

COMPUTATION WITH CONTINUOUS MODE  
CMOS CIRCUITS IN IMAGE PROCESSING  
AND PROBABILISTIC REASONING

A THESIS  
SUBMITTED TO THE UNIVERSITY OF MANCHESTER  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY  
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2014

**PRZEMYSŁAW MROSZCZYK**

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

# Contents

<b>Abstract</b>	<b>6</b>
<b>Declaration</b>	<b>7</b>
<b>Copyright</b>	<b>8</b>
<b>Acknowledgements</b>	<b>9</b>
<b>Author's publications</b>	<b>10</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Chapter overview .....	11
1.2 Reasons for alternative approaches .....	11
1.3 Alternative ways of information processing .....	15
1.3.1 Analogue computers .....	15
1.3.2 Artificial neural networks .....	16
1.3.3 Stochastic computers .....	17
1.4 Motivations .....	19
1.5 Research overview .....	19
1.5.1 Binary image processing .....	20
1.5.2 Bayesian inference .....	21
1.6 Contributions .....	24
1.7 Thesis structure .....	25
<b>2 Computation with MOS transistors under parameter variability</b>	<b>27</b>
2.1 Chapter overview .....	27
2.2 Analogue computation with MOS transistor .....	27
2.2.1 MOS model for hand calculations .....	29
2.2.2 Switched-current circuits .....	31
2.2.3 Analogue multipliers .....	32
2.3 Parameter variability in CMOS technologies .....	35
2.3.1 Physical model .....	36
2.3.2 Empirical model .....	36
2.3.3 Variability propagation .....	38
2.4 Mismatch modelling .....	39
2.4.1 Mismatch MOS model for hand calculations .....	39
2.4.1.1 Strong inversion and saturation .....	40
2.4.1.2 Strong inversion and linear region .....	42
2.4.1.3 Weak inversion .....	43
2.4.2 Statistical MOS models for CAD .....	43
2.4.3 Extended simplified model .....	45
2.5 Mismatch scaling .....	45
2.6 Mismatch versus noise .....	48
2.7 Mismatch versus temperature .....	50

2.8	Mismatch optimisation .....	50
2.8.1	Circuit design techniques .....	50
2.8.2	Layout drawing techniques .....	51
2.9	Conclusions .....	52
<b>3</b>	<b>Current-mode analogue multipliers</b> .....	<b>53</b>
3.1	Introduction and chapter overview .....	53
3.2	Continuous-time Gilbert multiplier .....	53
3.2.1	Circuit analysis and realisation .....	54
3.2.2	Design issues .....	57
3.2.3	V-AMS MOS model .....	58
3.2.4	Computational errors .....	62
3.2.5	Simulation results .....	63
3.3	Discrete-time Gilbert multiplier .....	66
3.3.1	Circuit analysis and realisation .....	66
3.3.2	Design issues .....	71
3.3.3	Simulation results .....	73
3.4	Fixed point digital implementation .....	77
3.5	Summary and conclusions .....	80
<b>4</b>	<b>Delay lines</b> .....	<b>81</b>
4.1	Chapter overview .....	81
4.2	Introduction .....	81
4.3	Circuit operation .....	83
4.3.1	Current starved inverter (CSI) .....	85
4.3.2	Output split inverter (OSI) .....	86
4.4	Mismatch analysis .....	86
4.4.1	Mismatch in CSI gate .....	88
4.4.2	Mismatch in OSI gate .....	89
4.4.3	Simplified analytical model .....	90
4.4.4	Charge sharing and S/D inherent capacitances .....	91
4.4.5	Model derivation .....	94
4.4.5.1	Mismatch model of a delay gate .....	96
4.4.5.2	Mismatch model of a delay line .....	98
4.4.5.3	Mismatch optimisation .....	98
4.5	Chip design and circuit implementation .....	99
4.6	Experimental results .....	102
4.6.1	Callibration .....	102
4.6.2	Normalised delay variance .....	104
4.6.3	Time jitter .....	107
4.6.4	Simulations versus measurements .....	108
4.7	Conclusions .....	112
<b>5</b>	<b>Asynchronous CMOS logic array for binary image processing</b> .....	<b>113</b>
5.1	Chapter overview .....	113
5.2	Introduction .....	113
5.2.1	Bio-inspired approach .....	113
5.2.2	SIMD paradigm .....	114
5.2.3	Vision chips .....	115

5.3	Wave propagation approach to skeletonization.....	116
5.3.1	Skeletonization (background knowledge).....	116
5.3.2	Trigger-wave propagation concept.....	117
5.3.3	Hardware realisations .....	120
5.4	Circuit design .....	120
5.4.1	Propagation gate .....	121
5.4.2	Collision-detectig gate.....	122
5.4.3	CMOS circuit realisation.....	126
5.4.4	Mismatch optimisation .....	128
5.5	Chip implementation .....	131
5.6	Test system and setup.....	134
5.7	Experimental results .....	136
5.7.1	Result comparison .....	136
5.7.2	Delay voltage tuning.....	140
5.7.3	Supply voltage variability.....	141
5.7.4	Temperature variability .....	142
5.8	Design issues .....	144
5.8.1	Power rail ringing.....	144
5.8.2	Design asymmetry .....	146
5.9	Performance and power .....	147
5.10	Design improvements and conclusions .....	149
5.10.1	Conclusions .....	149
5.10.2	Improvements and future work .....	150
<b>6</b>	<b>Wave propagation concept in arbitrary metrics</b>	<b>152</b>
6.1	Chapter overview .....	152
6.2	Introduction .....	152
6.3	Propagation and timing analyses .....	154
6.4	Simplified switched <i>RC</i> model.....	157
6.5	CMOS design and experimental results .....	159
6.6	Conclusions .....	160
<b>7</b>	<b>Probability and reasoning</b>	<b>162</b>
7.1	Introduction and chapter overview.....	162
7.2	Conditional probability and Bayes' rule .....	163
7.3	Bayesian networks.....	164
7.4	Bayesian inference .....	165
7.5	Belief propagation .....	167
7.6	Factor graphs .....	169
7.7	Conclusions .....	173
<b>8</b>	<b>VLSI systems for Bayesian inference</b>	<b>174</b>
8.1	Chapter overview .....	174
8.2	Introduction .....	174
8.3	Analogue circuits for arithmetic operations .....	176
8.3.1	Continuous-time circuits .....	176
8.3.2	Discrete-time circuits.....	177
8.4	Computational errors .....	179
8.5	Simulations.....	180

8.5.1	Simulation setup .....	180
8.5.2	Results .....	183
8.6	Networks in continuous-time analogue circuits .....	187
8.6.1	Accuracy versus redundancy .....	187
8.6.2	Accuracy versus area .....	189
8.6.3	Convergence in large networks .....	191
8.6.4	Complexity and power scaling .....	192
8.7	Performance of analogue systems .....	195
8.7.1	Computational efficiency .....	195
8.7.2	Performance of the continuous-time realisation .....	196
8.7.3	Performance of the discrete-time realisation .....	198
8.8	Digital implementation .....	200
8.8.1	Fixed point arithmetic .....	200
8.8.2	Hardware realisation .....	202
8.8.3	Computational accuracy .....	204
8.8.4	Digital synthesis and implementation .....	206
8.8.5	Performance of the digital realisation .....	209
8.9	Performance comparison .....	211
8.9.1	Analogue implementation .....	212
8.9.2	Digital implementation .....	212
8.9.3	Software implementation for PC .....	213
8.9.4	Discussion .....	215
8.10	Conclusions .....	220
<b>9</b>	<b>Conclusions</b> .....	<b>222</b>
9.1	Research summary .....	222
9.1.1	Binary image processing .....	223
9.1.2	Delay lines .....	223
9.1.3	Probabilistic reasoning .....	224
9.1.4	Contributions .....	225
9.2	Future work .....	226
9.2.1	Image processing .....	226
9.2.2	Bayesian inference .....	226
9.2.3	Noise based information processing .....	227
9.2.4	Hardware-accelerated network learning .....	228
<b>10</b>	<b>References</b> .....	<b>229</b>
<b>11</b>	<b>Appendix A</b> .....	<b>246</b>
<b>12</b>	<b>Appendix B</b> .....	<b>249</b>

# Abstract

*Computation with Continuous Mode CMOS Circuits  
in Image Processing and Probabilistic Reasoning*

A thesis submitted to The University of Manchester for the degree of  
Doctor of Philosophy  
Przemysław Mrozczyk  
May, 2014

The objective of the research presented in this thesis is to investigate alternative ways of information processing employing asynchronous, data driven, and analogue computation in massively parallel cellular processor arrays, with applications in machine vision and artificial intelligence. The use of cellular processor architectures, with only local neighbourhood connectivity, is considered in VLSI realisations of the trigger-wave propagation in binary image processing, and in Bayesian inference. Design issues, critical in terms of the computational precision and system performance, are extensively analysed, accounting for the non-ideal operation of MOS devices caused by the second order effects, noise and parameter mismatch. In particular, CMOS hardware solutions for two specific tasks: binary image skeletonization and sum-product algorithm for belief propagation in factor graphs, are considered, targeting efficient design in terms of the processing speed, power, area, and computational precision.

The major contributions of this research are in the area of continuous-time and discrete-time CMOS circuit design, with applications in moderate precision analogue and asynchronous computation, accounting for parameter variability. Various analogue and digital circuit realisations, operating in the continuous-time and discrete-time domains, are analysed in theory and verified using combined Matlab-Hspice simulations, providing a versatile framework suitable for custom specific analyses, verification and optimisation of the designed systems. Novel solutions, exhibiting reduced impact of parameter variability on the circuit operation, are presented and applied in the designs of the arithmetic circuits for matrix-vector operations and in the data driven asynchronous processor arrays for binary image processing. Several mismatch optimisation techniques are demonstrated, based on the use of switched-current approach in the design of current-mode Gilbert multiplier circuit, novel biasing scheme in the design of tunable delay gates, and averaging technique applied to the analogue continuous-time circuits realisations of Bayesian networks. The most promising circuit solutions were implemented on the PPATC test chip, fabricated in a standard 90 nm CMOS process, and verified in experiments.

# Declaration

## **The University of Manchester PhD Candidate Declaration**

**Candidate Name:** Przemyslaw Mroszczyk

**Faculty:** Engineering and Physical Sciences

**Thesis Title:** *Computation with Continuous Mode CMOS Circuits  
in Image Processing and Probabilistic Reasoning*

### **Declaration to be completed by the candidate:**

I declare that no portion of this work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Signed:

Date:

# Copyright

The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487> ), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see <http://www.manchester.ac.uk/library/aboutus/regulations>) and in The University’s policy on Presentation of Theses.

# Acknowledgments

I would like to express my great appreciation to my supervisor Dr. Piotr Dudek for his professional guidance, enthusiastic encouragement and constructive suggestions throughout this research work.

I express my gratitude to my colleagues Dr. Stephen Carey, Dr. Alexey Lopich, Dr. David Barr, Dr. Jayawan Wijekoon, Mr. Bin Wang and Mr. Declan Walsh for their assistance, help and support in this research, and for creating friendly environment.

I would like to extend my thanks to the academic, technical and administrative staff of The University of Manchester for offering me help, resources and assistance, necessary for successful completion of this research.

Finally, I wish to thank my parents (Danuta and Wieslaw) for their strong support, patience and love, and my friends, who greatly contributed to my professional and personal development.

## Author's Publications

The work and the results presented in this thesis were in parts subject of the following conference, journal, and poster contributions:

P. Mrosczyk, P. Dudek, "Trigger-Wave Collision Detecting Asynchronous Cellular Logic Array for Fast Image Skeletonization", ISCAS 2012, pp. 2653-2656, May 2012 (Received *Best Student Paper Award*).

P. Mrosczyk, P. Dudek, "Tunable CMOS Delay Gate with Reduced Impact of Fabrication Mismatch on Timing Parameters", NEWCAS 2013, pp. 1-4, Jun. 2013.

P. Mrosczyk, P. Dudek, "Trigger-wave propagation in arbitrary metrics in asynchronous cellular logic arrays", ECCTD 2013, pp. 1-4, Sep. 2013.

P. Mrosczyk, P. Dudek, "The Accuracy and Scalability of Continuous-Time Bayesian Inference in Analogue CMOS Circuits", accepted for ISCAS 2014.

P. Mrosczyk, P. Dudek, "Tunable CMOS Delay Gate with Improved Matching Properties", **IEEE Transactions on Circuits and Systems - I: Regular Papers** (in print), doi:10.1109/TCSI.2014.2312491, 2014.

P. Mrosczyk, P. Dudek, "Asynchronous Cellular Logic Array for Fast and Low-Power Global Binary Image Processing", in preparation for submission to **IEEE Transactions on Circuits and Systems I**.

"Bayesian Inference in Analogue VLSI", UK Neuroinformatics Node, poster presentation, Manchester, UK, Nov. 2011.

"Bayesian Inference in Analogue Networks with Gaussian Noise", Building Bridges to Build Brains, poster presentation, Edinburgh, UK, Nov. 2012.

## Chapter 1

---

# Introduction

---

### 1.1 Chapter overview

This chapter discusses reasons for departing from digital computation towards alternative approaches, operating in continuous domains using analogue and various bio-inspired ways of information processing. In particular, motivations and key contributions of this research are presented in the field of CMOS hardware design for image processing and probabilistic reasoning tasks. Research summary and the outline of the thesis structure concludes the chapter.

### 1.2 Reasons for alternative approaches

The application of Boolean logic in solving problems using networks of electro-mechanical switches, proposed by Claude Elwood Shannon in his Master's dissertation "A Symbolic Analysis of Relay and Switching Circuits" in 1940 [Shannon 38, 40], was one of the major pioneering steps in establishing "digital computing", providing the most reliable and precise way of information encoding and processing ever invented. This work gave foundation for the build of a versatile programmable computing machine (Turing machine), which principles of operation were proposed by Alan Turing [Turing 36] and its particular hardware realisation, known as the von Neumann architecture, was proposed by John von Neumann [von Neumann 45]. Merits of such approach were quickly recognised and electro-mechanical switches were replaced with

the best available electronic devices, first valves, and later bipolar and MOS transistors. This began the era of digital computing, the quick development of which, was on one hand propelled by the scientific curiosity and progress in the electronics manufacturing, and on the other, by the political situation in the world, where the applications of new computers, not only in science but also in security and defence, were highly expected. Examples of the most famous constructions from that time are Electronic Numerical Integrator And Computer (ENIAC), built by the scientists from the University of Pennsylvania in 1946, and the Small Scale Experimental Machine ("The Baby") built at the University of Manchester in 1948 as a prototype of a larger computing systems with magnetic data storage [Bowden 53]. In particular, ENIAC could perform about 5000 operations per second with a certain degree of processing parallelism, using several accumulators at the same time for different operations with 10 bit precision. It consumed 150 kW of power and consisted of over 17 000 vacuum tubes, costing an equivalent to 3 million pounds today. Due to the high cost, power consumption and large size, digital computers were solely used for specific scientific and military purposes and access to such devices was rather restricted only to small groups of scientists. The release of the first commercially available programmable 4 bit microprocessor, Intel 4004, by Intel Corporation in 1971, with processing capabilities comparable to ENIAC but consuming only 0.5 W of power [Intel 86], was a milestone achievement in the history of the digital computing. It denoted a new direction for the research and the industry leading to design of cheaper and more power efficient systems based on synchronous digital circuits, where low power consumption and high processing speed could be obtained by transistor size downscaling, and the cost could be reduced by mass production and further MOS process refinement. As a result, MOS technologies evolved, starting from 10  $\mu\text{m}$  down to nearly 10 nm feature size, supported by a highly reliable chip manufacturing processes and a variety of CAD tools for simulation, design and verification of the designed integrated circuits before fabrication.

Due to the long period of development and widespread use of systems based on synchronous digital circuits, such computers could already be called *classical* (term *classical* refers to a typical digital computer based on von Neumann or Harvard architecture, often associated with a single-processor Personal Computers (PCs) executing instructions serially). However, alternative machines, employing different classes of electronic circuits and various ways of information encoding and processing, have always been considered in parallel with the classical solutions. One reason for that,

concerning mostly the earliest digital constructions, was the limited processing speed and limited computational capacity of digital systems, unable to solve complex problems required in many practical applications. Another reason was the limited processing efficiency of the digital computers in comparison to the biological nervous systems, inherently dealing with complex tasks while consuming very low power. Therefore, many attempts have been made to mimic the operation of such systems using electronic circuits. In general, the main limitations of digital computers are their high power consumption, low processing speed and large size when compared to the asynchronous or analogue circuit implementations realised in the same technology and dedicated for particular tasks.

For a long time digital computing in CMOS technologies benefitted from the scaling process resulting in improved power efficiency and higher processing speed. In order to build faster and less power consuming computers, practically the same circuit designs, with some minor modifications, could have been used for chip fabrication in the finer technology nodes to achieve the desired effects. The continuous scaling process allowed for implementation of larger circuits on the same chip area, with complexity growing exponentially with time, as suggested by Gordon Moore in 1965 [Moore 65]. This observation, known as Moore's law, became a general scaling rule widely used in science and industry not only to predict the integration level of the future designs but also their speed, power performance, available memory and potential cost. It is obvious, however, that Moore's law is not a generic rule of progress but a self-propelling phenomenon driven by competing corporations trying to maintain reputation and maximise profits. Nonetheless, Moore's law provided a good estimate for parameters of future designs until certain factors, stemming from quantum physics and thermodynamics, started to constrain further performance growth of the CMOS circuits. The main goal of technology feature size miniaturisation is MOS gate capacitance scaling, which on one hand, allows for faster operation but, on the other hand, increases  $kT/C$  thermal noise. This, in addition to the reduced supply voltage, affects the operation of digital circuits reducing their statistical reliability. It should be noted that these effects do not constrain the size of a transistor, that can be manufactured, but only suggest to limit either the integration level, or to reduce the clock speed to maintain power dissipation and thermal noise [Kish 2002]. Since device scaling is a very cost-efficient process, the integration level grows exponentially following the Moore's law, however, other measures of performance, such as power and speed have to be compromised in order to meet thermal

constraints [Gea-Banacloche 2005]. This can be observed in Figure 1.1 showing the scaling trends of Intel processors, where only the number of transistors per chip follows the exponential growth.

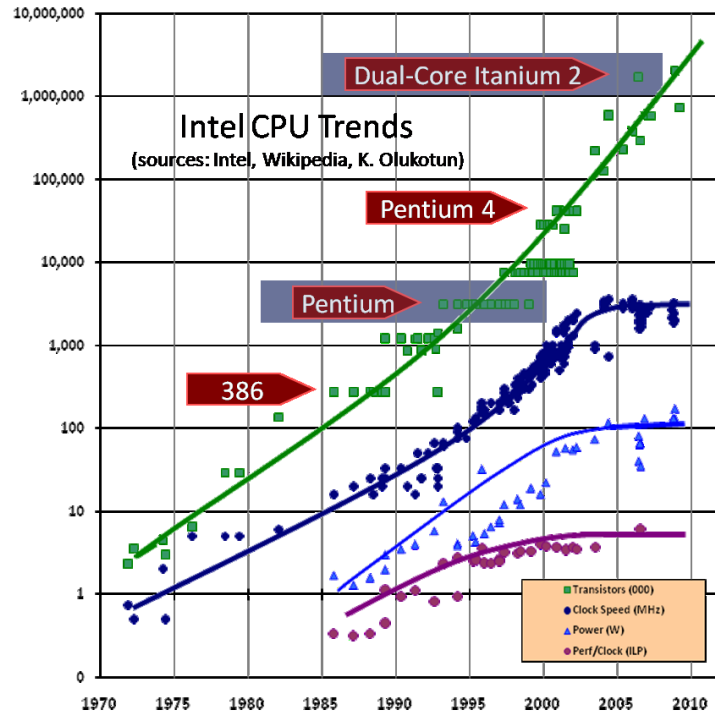


Figure 1.1. Scaling trends of Intel chips (green - number of transistors, navy - clock speed, blue - power, magenta - instruction-level parallelism), figure taken from [Shutter 2005].

Further scaling of the CMOS technologies will most probably continue for the next several decades, and will not be quickly replaced by any other known approach on an industrial scale. It is yet too early to assess to what extent novel propositions, based on graphene, single electron, or polymer transistors, will provide satisfactory solutions. At present, classical computing relies on CMOS technology and digital circuits operating close to their thermodynamic limits. Therefore, faster processing is usually attempted by novel programming techniques and better utilisation of the resources using multi core processors. In particular, better integration of the memory and CPU has been suggested to overcome so called *von Neumann architectural bottlenecks* [Backus 78], for example, by employing massively parallel *cellular* processor architectures, with only local or nearest neighbourhood connectivity, associative memories, and processing circuits optimised for particular tasks. Further performance boost is also searched for in the systems using asynchronous and analogue circuits, or even employing other ways of representing and processing information.

### 1.3 Alternative ways of information processing

In the following section more detailed review of the alternative processing methods, such as analogue computers, artificial neural networks and stochastic machines, will be provided. In particular, aspects concerning hardware design challenges and limitations, essential to the research presented in this theses, will be discussed.

#### 1.3.1 Analogue computers

Early realisations of digital computers were not fast and elaborate enough to handle complex mathematical models described by large sets of differential equations. On the other hand, it was observed that many such systems could easily be modelled by analogue circuits with passive and active elements. Since the parameters of the circuits can be tuned, they could inherently solve a set of equations while settling down to the steady state. Such analogue computers were popular even before the invention of the digital machines and used in many areas requiring system modelling and simulations, such as in real time power network simulations [Joetten 85], in nuclear physics experiments [Arbel 64], and in various control applications [Bissell 2004]. Examples of such generic analogue computers used for solving differential equations and system modelling are Newmark (1960, Cambridge, UK) and ELWAT-1 (1967, Wroclaw, Poland). In particular, there were 50 units of ELWAT-1 manufactured by Elwro company in Wroclaw (Poland) between 1967 and 1969 [Sienkiewicz 2009]. Each computer consisted of 19 valve operational amplifiers, 12 computational blocks for summation, differentiation and integration, one multiplier and 4 function generators, and operated with a precision of 0.1% to 5%, displaying the results on an oscilloscope or printing traces on a paper tape. For a long time such analogue computers were competitive with the digital ones in terms of speed and the complexity of the problem, that could be solved. However, relatively low precision, low dynamic range and strong dependency of the circuit parameters on temperature, precluded their use in many applications. Even though the motivations for using analogue circuits in computation and modelling were driven solely by the lack of alternative solutions at that time, or by a low speed and high cost of the early digital computers, the idea was not entirely abandoned after the rapid improvement of the digital computers. It returns as an alternative for classical computers in very complex applications such as bio-inspired processing systems and massively parallel processor arrays.

### 1.3.2 Artificial neural networks

Biological nervous systems are in many ways similar to electronic circuits, where the information is processed and transmitted between conductive neurons using electric charges. Given the very low power dissipation of the human brain, estimated to be about 20 W [Kish 2011], high complexity, and real time operation, the computational efficiency of such biological computers is much higher than the ones achieved using electronic circuits. One of the characteristic features adopted by nature in nervous systems is the high degree of redundancy in the hardware structure and in the information encoding scheme [von Neumann 52]. Such networks typically consist of a large number of neurons working in parallel to collect, process and exchange information with each other. Redundancy and parallelism assure high reliability of the system and its immunity to parameter fluctuations. This distinguishes them from typical electronic circuits, where failure of one component typically leads to an imminent failure of the whole system. One of the early attempts to improve understanding of the operation of neural networks was done in early 1940s by Warren McCulloch and Walter Pitts, who proposed a very simple mathematical model of neuron operating in a binary mode with Heaviside step activation function, used to the threshold sum of the active input signals [McCulloch 43]. A more elaborate model of perceptron was later proposed by Frank Rosenblatt in 1958 [Rosenblatt 58]. These works, among others, gave foundation for different types of artificial neural networks (ANN), such as multi-layer feed-forward networks and feedback networks as described in [Hecht 90]. Notable types of ANN are Hopfield networks [Hopfield 84] and cellular neural networks (CNN) [Chua 88a, 88b], [Roska 93]. In particular, VLSI implementations of CNN became popular due to their regular two-dimensional structure and only near neighbourhood connectivity between neurons. There are a number of propositions for hardware implementations of such networks, mainly using nonlinear analogue VLSI circuits, operating in continuous and discrete time modes, or specific high performance parallel digital implementations [Kinet 97]. It should be noted that analogue design in this area is very promising since many problems can be solved using dedicated circuits [Mead 89]. Examples of such circuits are single transistor multipliers based on the floating gate technique used as programmable weighted synaptic connections between neurons [Dominguez-Castro 98], rank-order extractors for winner-take-all networks [Hung 2002], and various nonlinear function realisations [Chang 96a].

In analogue networks, information is typically represented by an electric charge measured either as a voltage, when stored on a capacitance, or as a current, when flowing through a conducting element. This is sometimes used to differentiate between circuits operating in the current or voltage mode. However, such classification is rather conventional since the operation of any circuit requires constant transitions between current and voltage domains. [Toumazou 93b].

In biological neural networks, the information is carried by trains of electrical pulses (spikes) where the rate or probability of the pulse occurrence, measured over a period of time, or a correlation between trains of spikes, correspond to the information encoded by the signal. This shows the redundancy of biological systems not only in terms of the hardware but also in terms of the information encoding. Such systems require processing methods different from those used in the conventional electronic circuits [Kinget 97]. Following this approach, a separate class of electronic circuits, dedicated for the Spiking Neural Networks, have been proposed [Maass 2001]. Such circuits combine the processing methods of digital and analogue solutions and operate similarly to the asynchronous circuits, processing discrete signals (e.g. digital pulses) but in continuous time and representing continuous (analogue) values.

### 1.3.3 Stochastic computers

The probabilistic nature of the signals observed in biological neural networks was an inspiration to the design of specific type of arithmetic machines, considered already in 1960s, as an alternative to the analogue computers [Riberio 67], [Poppelbaum 67], [Gaines 67]. In the design of such stochastic computers it was assumed that the continuous variables could be represented as sequences of random pulses with probability proportional to an analogue value. Simple logic operations performed on such sequences correspond to inherent arithmetic operations on the probabilities. For example, if two random sequences of pulses are generated with probabilities  $p$  and  $q$ , the logical AND operation on these signals will generate a sequence with probability equal  $pq$  [Riberio 67]. Using more complex structures with logic gates, delay elements and memory circuits, different arithmetic operations such as addition, subtraction, multiplication, division, square root, integration and differentiation with respect to time or other variable, can also be realised [Gaines 67].

Stochastic computers operate according to very simple principles, however, their circuit realisations required additional hardware blocks such as generators of random

pulses with controlled probability distribution and signal randomisers. The generators were necessary to convert the analogue input signals to the sequences of random pulses. The randomisers had to be used to restore the statistical properties of the intermediate signals after each processing step [Riberio 67]. Since the computation is based on stochastic processes, the result converges to the expected solution, however, with limited precision depending on the length of the produced sequence. It has been shown that the precision of an analogue value represented by a random pulse string is proportional to the square root of the sequence length. In order to assure precision of 1% the sequence should consist of 10,000 pulses, and this length will have to increase by 100 times if the required precision is 0.1%. Even if longer sequences could be generated, due to the stochastic nature of the process, the probability of unforeseen random disturbances will gradually increase degrading the achievable precision [Poppelbaum 67]. The use of stochastic computers have been suggested in image transformers and artificial neural networks [Petriu 96]. They are promising alternatives to classical computers that could be considered in future designs, where moderate precision is sufficient. The use of modern CMOS technologies may facilitate some solutions to the aforementioned design challenges, seen as unsolvable at the time when the integrated circuits were not widely in use. It should be noted, however, that issues regarding area, power consumption and scalability of such stochastic machines in CMOS need investigation.

The idea of processing continuous random signals (i.e. noise) to evaluate logic functions has been proposed to address the problem of thermal noise in the modern CMOS technologies [Kish 2011]. Such techniques rely on the correlated noise information processing and are immune to the presence of the uncorrelated thermal noise. In principle, the statistical parameters of random orthogonal signals are used to encode logic state. In the processing, electronic circuits such as multipliers, low pass filters and switches, can be used to perform logic operations on the noise signals. Such systems, however, require analogue circuits operating in a very high bandwidth, to attain the processing speed comparable to existing computers and more complex hardware realisations of logic gates. It should be noted that there exist other approaches to stochastic data processing using probabilistic CMOS computers [Korkmaz 2008] or machines going beyond the scope of electronic circuits such as quantum computers [Shor 94]. Due to very specific principles of operation, the application domain of such computers is still limited.

## 1.4 Motivations

The main motivation for the research presented in this thesis is to address issues and challenges in the design of cellular processor arrays in standard CMOS technologies with applications in fast and power efficient image processing and probabilistic reasoning tasks. In particular, asynchronous and analogue circuits are of the main interest, since they can inherently solve many computationally demanding tasks faster and more efficiently than classical computers.

The work undertaken in this research consists of two parts. The first part considers the use of asynchronous pixel-parallel processor arrays in morphological operations on binary image, supported with the experimental results obtained from the fabricated test chip. The second part considers the use of the continuous-time and discrete-time analogue processor arrays for a particular set of matrix-vector operations required in loopy belief propagation algorithm for approximate inference in Bayesian networks. Aspects, such as processing speed, computational accuracy under fabrication mismatch and efficiency, accounting for power and area requirements, and scalability of such solutions with network size were investigated. In addition to that, equivalent synchronous digital circuits and software solutions on PC, were devised to provide the reference for comparison of the analogue and digital systems in terms of speed, power, area and computational efficiency.

## 1.5 Research overview

The progress of research work is not predictable and many initial assumptions, ideas and expectations for potentially promising results has to be revised, modified and sometimes abandoned. On the other hand, those unexpectedly encountered obstacles, triggered new ideas and solutions, splitting the research into new branches or even changing its main direction. In order to keep this thesis concise and consistent, some ideas and conclusions, that go slightly off the track, although interesting and certainly valuable, were not included but briefly summarised in the last chapter providing ideas and directions for future work. The outline of the research presented in this thesis indicating its turning points, encountered problems and proposed solutions, is presented further in this section.

### 1.5.1 Binary image processing

The idea of binary image processing using trigger-wave propagation concept was initially inspired by the properties of the light-sensitive chemical nonlinear system (a variant of the Belousov-Zhabotinski medium) capable of generating chemical reactions in the form of propagating wave-fronts when stimulated by light [Kuhnert 89], [Krinsky 91]. In literature, such systems are typically realised using asynchronous CMOS cellular processor arrays with applications in morphological image processing task such as geodesic reconstruction, hole filling, etc. [Dudek 2006].

The primary goal of the research was to use such arrays to evaluate skeletons of binary images by implementing a mechanism to detect collisions between the propagating waves. In such a system, it is essential for each processing cell to compute results within a short and precisely defined period of time. Since there is no time reference (e.g. clock signal) available in such asynchronous system, the idea was to implement a simple delay gate generating the required time interval on the arrival of the input data in each cell. By using a simple delay circuit based on a three-transistor current starved inverter, very satisfactory simulation results were obtained. However, when accounting for the parameter variability of MOS transistors, significant problems with the precision of the generated time intervals were observed, affecting the quality of the extracted images. This problem was solved by selective transistor scaling, where only critical transistors in terms of the timing parameters were enlarged. It was also observed that replacing the current limiting transistor with the switching one, in the delay gate circuit, slightly changed the dynamic behaviour of the gate, reducing the impact of the parameter variability on the generated time intervals. Such a gate has the current limiting transistor in between the switching ones *splitting* the output of the inverter, therefore, it is called *output split inverter* (OSI). The idea was further pursued in isolation from the image processing framework, and eventually evolved into a separate research subject of tunable delay lines with improved matching properties, applicable in delay locked loops, time to digital converters, readout systems for particle detection and neuromorphic circuit design.

In order to verify and optimise the operation of the designed system, investigate its asynchronous behaviour, timing parameters and dynamic operation, simulations of the actual circuit arrays rather than of a single processing cell were required. It seemed necessary and convenient to provide the inputs and represent the results of the

simulations in the form of binary images rather than time dependent signal traces. For this purpose, a separate set of tools was built in Matlab and C++ to communicate with Hspice circuit simulator and visualise the obtained results as a sequence of binary images. This allowed to perform the mismatch optimisation of the array and also aided investigating and solving problems related to current leakage, design asymmetry and initialisation process, not easily detectable at the processing cell level.

It was also observed that slightly different contours of the waves triggered from a single pixel can be generated depending on the timing parameters of the gate array. The theory behind this was further investigated based on the simplified switched  $RC$  timing model, generalising the principles of isotropic wave propagation in rectangular arrays in the context of particular distance measure norm. The results of this research were applied in the design of the asynchronous processor array, capable of generating circular propagation waves, significantly improving the quality of the evaluated skeletons and Voronoi diagrams, difficult to implement in synchronous circuits or even in software adaptations of the propagation-based image processing method.

### 1.5.2 Bayesian inference

Bayesian inference in networks representing systems with cause-effect relationships is often used in the applications requiring control, decision making, diagnoses and forecasting [Pearl 86, 88], [Jensen 2007]. In general, methods for exact inference are classified as NP-hard problems [Cooper 90], therefore, a lot of attention has been paid to the simplified methods for approximate reasoning such as loopy belief propagation and stochastic sampling [Neapolitan 2004]. In particular, belief propagation, relies on the repetitive information exchange between the nodes, using only locally available data in calculating probabilities. The algorithm performs algebraic operations including matrix-vector and vector-vector multiplications on the discrete probability distributions. In order to avoid underflow errors, belief propagation assumes normalisation of the computed probability distributions after each processing step. This, however, requires additional summations and divisions, increasing the computational complexity of the algorithm.

Despite extensive literature concerning principles of Bayesian inference by exact and approximate methods, the area of their hardware implementations is not well explored and limited to only a few publications. The idea of using sum-product algorithm in factor graph implementations in analogue CMOS circuits, for the realisation of loopy belief propagation, was initially proposed in [Kschischang 2001]. This approach was further

followed in [Luckenbill 2002], reporting very low computational accuracy achieved in such circuit realisations, however, relying on circuit simulations of a network with only three nodes and using generic BSIM3 MOS transistor model, not related to any technology. To the best of my knowledge, these are the only practical analogue circuit realisations, apart from several works considering Bayesian inference in spiking neural networks [Corneil 2012] and in digital domain using VLSI [Liang 2011] and field programmable gate arrays (FPGA) [Lin 2010], [Kulesza 2006]. Therefore, the primary motivation of this work was to revise the approach to Bayesian inference in analogue CMOS circuits, and propose a clear formalism in a consistent framework, providing the reference and the foundation for future research in this area.

In this thesis, the analogue circuits for arithmetic operations were implemented using CMOS realisation of the Gilbert multiplier, operating in continuous-time and discrete-time domains [Toumazou 93b]. The operation of such circuits was thoroughly analysed in theory and verified in simulations using BSIM4 MOS transistor models provided by the foundry. In particular, the operation of the continuous-time multiplier was verified using the author's simplified equivalent model V-AMS, built in Verilog AMS language, to account for second order effects in MOS transistors operating in weak inversion. It was observed that the computational errors result mainly from the channel length modulation effects and from the variable slope factor. These problems were addressed by employing cascode current mirrors, single stage differential pairs (to assure operation in saturation within the limited voltage headroom), and MOS transistors with different threshold voltages, to reduce the leakage currents. The optimised design of the multiplier was used to realise the matrix-vector and vector-vector operations in more complex arithmetic structures, required in the sum product algorithm. Based on that, scaling rules considering circuit complexity and power were derived.

At that stage it was also necessary to develop methods for quick verification of the simulation results and the computational accuracy of the implemented networks. To address this, a separate set of tools for Bayesian inference was built in Matlab. The verification of a particular Bayesian network implemented in analogue circuit, for a given set of input parameters, was performed using scripts for combined Matlab-Hspice simulations, where the inference results obtained from the circuit and software were used to evaluate the computational error. This allowed to further verify the operation and scalability of the circuit realisations of larger networks.

As was initially expected, accounting for the fabrication mismatch of MOS transistors resulted in a significant degradation of the computational accuracy, calling into question any practical use of such circuits. This concluded the preliminary stage of the research giving directions to design of more precise analogue systems for arithmetic operations. It brought attention to the parameter variability issues, setting the next research objective, which was to investigate possible ways of mismatch reduction and optimisation in such analogue circuits.

Several attempts to solve this problem were made, accounting for MOS transistor scaling, network parameters optimisation, redundant design approach and also migration to switched-current (SI) mode. The main disadvantage of the first method, based on enlarging the size of MOS transistors, was a significant increase of the circuit area and settling time. The idea of tuning network parameters to compensate for the parameter mismatch was initially adapted from neural network systems, where inaccuracies of neurons and synapses can be reduced by proper weight adjustment. Unfortunately, this concept failed because of relatively small number of network parameters, as compared to the number of MOS transistors used in the implementation. Another idea, benefiting from the design redundancy was based on the hypothesis that collecting and averaging results from several identical networks (affected by random parameter variability), will generate more precise result. This idea was successful on the simulation level, however, in order to obtain results with acceptable accuracy, large number of uncorrelated network copies were required, practically precluding the use of this technique in CMOS realisations.

Migration to switched-current mode was attempted due to the very high immunity of such circuits to fabrication mismatch reported in literature. It was done based on the observation that the continuous-time Gilbert multiplier can be realised using discrete-time current mirrors. The proposed idea was successful and the operation of the SI version of the multiplier, and two Bayesian networks consisting of 5 and 7 nodes were verified in simulations, giving promising results for the computational accuracy, efficiency, and power.

Since the assumed realisation of such systems is fully parallel, each arithmetic operation has its individual hardware block, which leads to a large area occupation. One idea to address this issue is to employ time multiplexing of the resources and serialise the processing flow. Another possibility is to investigate the hardware realisations of stochastic methods for approximate Bayesian inference, such as Gibbs sampling and

stochastic logic sampling. Such methods require less arithmetic operations per node than belief propagation, however, the hardware realisation of the random generators is still an open problem. So far, certain solutions for hardware random bit generators, based on amplified thermal noise [Wee 2001], metastability [Vasytsov 2008], and analogue chaos generators [Dudek 2003] have been proposed.

In this research, the results obtained from the continuous-time and discrete-time realisations of Bayesian networks were compared with their equivalent implementations in synchronous digital circuits operating in fixed point arithmetic with different bit precisions from 5 to 10 bits, and with software solutions realised in Matlab and C++. The objective of this comparison was to provide an overview and performance measure of three different approaches to the same problem realised in the same CMOS technology node using, in each case, non-trivial and optimised solutions.

## 1.6 Contributions

The major contributions of the research presented in this thesis are:

- Analysis and design of the collision detecting layer for trigger-wave propagation-based image processing algorithms in dynamic logic CMOS circuit combining logical AND function and 1 bit memory latch, using only 8 MOS transistors.
- Analysis and design of the propagation gate for trigger-wave propagation-based image processing algorithms with a novel bias scheme allowing for the generation of the circular wave contours, difficult to achieve in software or using generic SIMD processor arrays.
- Analysis and design of a delay gate employing a novel biasing scheme resulting in almost twice better matching properties when compared to the commonly used current starved inverter, with no penalty in terms of power or area.
- Analysis and design of the analogue CMOS discrete-time variant of the Gilbert multiplier, operating in current mode with computational accuracy comparable to its continuous-time equivalent but not affected by parameter mismatch.
- Analysis and design of an optimised digital fixed-point arithmetic circuits for matrix-vector operations with applications in probabilistic calculus and other areas requiring computation with normalised data.
- Analysis of the power, area and complexity scaling of the hardware realisations of the factor graphs for belief propagation in analogue circuits.

- Development and verification of the mismatch optimisation techniques based on the novel biasing scheme (OSI delay gates), results averaging (Bayesian networks in analogue continuous-time circuits) and switched-current technique (discrete-time current-mode multipliers).

In addition to the undertaken research, the most promising and successful circuit ideas were implemented on a test chip, fabricated using a standard 90 nm CMOS technology available through mini@sic program supervised by EUROPRACTICE. The layout of the PPATC chip (Parallel Processor Arrays Test Chip) was created in the full custom approach. The design was submitted for fabrication in November 2012 and the fabricated chips were received from the foundry in April 2013. The size of the PPATC chip was  $1875\text{ }\mu\text{m} \times 1875\text{ }\mu\text{m}$  including I/O ring and 64 pads, and accommodated three separate test designs:

- asynchronous logic array for binary image skeletonization and Voronoi tessellation ( $64 \times 96$  pixels),
- two arrays of 512 16-stage delay lines each implemented using current starved inverter (CSI) and output-split inverter (OSI) delay gates,
- two analogue processor arrays with various types of memory cells and Gilbert multipliers operating in switched current mode, dedicated for applications in Bayesian inference.

In order to test the fabricated chip, a separate test system based on Xilinx PicoBlaze (Kcpsm3) microcontroller was designed and implemented on a Development Board from Digilent, with Spartan 3 xc3s200 FPGA, operating in a command interpreting mode and providing a communication link between a PC and the PPATC chip. This thesis includes experimental results obtained from tests of the asynchronous logic array for binary image processing and delay line arrays.

## 1.7 Thesis structure

The background knowledge and the literature review concerning the analogue computation in CMOS circuits and a detailed discussion on parameter variability, are presented in Chapter 2. The analysis and design of the continuous-time and discrete-time current-mode analogue multipliers is presented in Chapter 3. The idea, analysis and the

obtained experimental results, concerning the operation of the OSI delay gates with improved matching parameters, are presented in Chapter 4. Chapter 5 provides an introduction to binary image processing based on the trigger-wave propagation concept, presents the proposed idea of detecting collisions, and discusses the obtained simulation and experimental results. Chapter 6 extends this discussion to propagation in arrays operating in different distance measure norms. An introduction to probabilistic reasoning, Bayesian networks and belief propagation is provided in Chapter 7. Various realisations of belief propagation algorithm, in analogue and digital systems, and the corresponding comparative analysis, is provided in Chapter 8. Chapter 9 concludes the thesis and discusses future work. Additional information concerning schematic diagrams and scaling rules of the analogue hardware for Bayesian inference is provided in Appendices A and B.

## **Chapter 2**

---

# **Computation with MOS transistors under parameter variability**

---

### **2.1 Chapter overview**

This chapter provides a background knowledge and the literature review on the computation in analogue circuits using MOS transistors and employing their switching capability, nonlinear characteristics and information storing properties. A detailed discussion will concern circuits such as analogue multipliers and switched-current memory cells, used in the realisations of the circuits considered in this research. Since the operation of the analogue and asynchronous circuits is highly determined by the accuracy of the fabricated hardware, the major part of this chapter deals with mismatch analysis, modelling and optimisation, providing foundation for CMOS circuit design discussed in this thesis.

### **2.2 Analogue computation with MOS transistor**

The function of MOS transistors in circuits for arithmetic operations is determined by the form of the arguments and the principles of computation employed. In digital domain, the computation on binary numbers is done using logic circuits with MOS transistors operating as switches. In such circuits, the dynamic behaviour of transistors during signal transitions and their nonlinear transfer characteristics, are usually not considered, unless the timing constraints of the designed system are affected. In the

analogue computing systems, the continuous arguments are represented by the magnitude of voltage or current, and the arithmetic operations such as addition, subtraction, integration etc., can be realised using linear  $RC$  circuits with operational amplifiers, where characteristics of MOS transistors are of second importance [Razavi 2001], [Allen 2002]. Such circuits are often used in communication, data converters and analogue signal processing, and are not frequently considered in the realisations of complex computing systems. This is mainly because of the high power and the large area requirements of the operational amplifiers.

In order to reduce the area and power consumption, and to accelerate the processing speed, circuits employing inherent features of MOS transistors (e.g. capacitance of the insulated gate, nonlinear transfer characteristics) are usually considered. In such approach, the mathematical relations between the drain currents and the gate, drain and source voltages, of a MOS transistor, depend on the operating point, and can be used in the realisations of different arithmetic operations, like addition, multiplication, division or log-linear conversion. There are a number of challenges in the design of such circuits, mainly stemming from biasing, dynamic range, noise and fabrication mismatch.

In the literature, fabrication mismatch and noise are often reported as the dominant factors affecting the operation of analogue circuits. Transfer characteristics of a MOS transistor, showing drain current  $I_D$  and the RMS values  $\sigma_{ID}$  (standard deviations) of the noise and the fabrication mismatch versus gate-source voltage, are presented in Figure 2.1. It can be observed that the variability of the drain current, resulting from mismatch, depends on the operation region and is nearly one order of magnitude higher than the variability caused by noise, when operating in strong inversion. In the designs assuming the operation of MOS transistors in weak inversion, both noise and mismatch should be taken into account. In particular, the exponential dependency of the drain current on the gate-source voltage magnifies the impact of parameter mismatch on the circuit performance. Also, a very small drain current in weak inversion approaches the noise level, reducing the dynamic range and degrading signal to noise ratio in the circuit. Nevertheless, the operation in weak inversion is often necessary, since the technology scaling imposes lower supply voltages, which leaves less headroom for the operation in strong inversion. Also, in weak inversion, the approximate exponential transfer characteristics are beneficial in the realisations of some arithmetic operations.

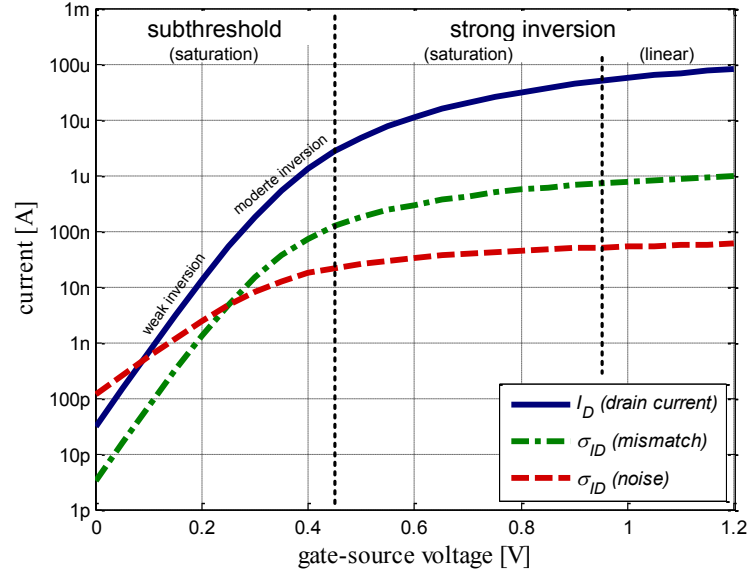


Figure 2.1. Drain current of MOS transistor and the RMS currents related to noise and fabrication mismatch vs. gate-source voltage (simulation results obtained using low-leakage MOS model from a 90 nm CMOS tech. assuming  $W = 1 \mu\text{m}$  and  $L = 1 \mu\text{m}$ , threshold voltage  $V_{th} = 0.45 \text{ V}$ , constant drain-source voltage  $V_{DS} = 0.5 \text{ V}$ , and gate-source voltage swept in range 0 - 1.2 V).

In the following section, an overview of different ideas utilising the inherent features of MOS transistors in the realisation of arithmetic operations in analogue circuits will be presented. In particular, switched-current circuits, advantageous in terms of mismatch immunity and various realisations of analogue multipliers, will be discussed. The discussion will be preceded by a brief introduction to the simplest MOS transistor model, based on the quadratic law (Spice Level 1), frequently used in analyses and hand calculations. The remainder of this chapter focuses on parameter variability in CMOS technologies, statistical models of MOS transistors, and mismatch optimisation techniques.

### 2.2.1 MOS model for hand calculations

In literature, theoretical analyses of CMOS circuits typically assume the use of the simplest MOS transistor model based on the square-law transfer characteristic, when operating in strong inversion [Shichman 68]. Such model was a sufficient analysis tool for hand calculations and computer simulations (known as Spice Level 1 model), useful in modelling circuits designed in CMOS technologies above  $1 \mu\text{m}$  feature size. In its basic form, it considers only four electrical parameters: threshold voltage  $V_{th}$ , current factor  $\beta$ , body effect parameter  $\gamma$ , and channel length modulation  $\lambda$ , which depend on the physical parameters of semiconductors [Allen 2002]. In order to simplify the analyses,

the body effect and the channel length modulation are usually not accounted for in the hand calculations, and that the bulk-source voltage equals zero (i.e. source is electrically connected to bulk). Such model describes the relations between the drain current  $i_D$  and the bias voltages  $v_{GS}$  and  $v_{DS}$  in strong inversion (when  $v_{GS} > V_{th}$ ) in two regions: saturation, when  $v_{DS} \geq v_{GS} - V_{th}$  and linear when  $v_{DS} < v_{GS} - V_{th}$ , and is given by the following equations:

$$i_D = \frac{\beta}{2} (u_{GS} - V_{th})^2, \text{ in saturation region where: } u_{DS} \geq u_{GS} - V_{th} \quad (2.1)$$

$$i_D = \beta \left[ (u_{GS} - V_{th}) u_{DS} - \frac{u_{DS}^2}{2} \right], \text{ in linear region where: } u_{DS} < u_{GS} - V_{th} \quad (2.2)$$

The current factor (also called large signal transconductance)  $\beta = \mu C_{ox} W/L$ , depends on the gate oxide capacitance  $C_{ox}$ , carrier mobility  $\mu$  and the device geometry defined by the channel width  $W$  and length  $L$ .

An extension to this model, accounting for the subthreshold operation in weak inversion, when  $v_{GS} \leq V_{th}$ , assumes the exponential dependency of the drain current on the gate-source voltage according to the equation [Allen 2002], [Mead 89]:

$$i_D = I_{D0} \frac{W}{L} \exp\left(\frac{u_{GS}}{nU_T}\right) \quad (2.3)$$

where  $I_{D0}$  is a specific current in weak inversion,  $W$  and  $L$  are the channel width and length respectively,  $U_T = kT/q$  is thermal voltage (equal approximately to 25.85 mV at a room temperature  $T = 300$  K), and  $n$  is a subthreshold slope factor, depending on the gate oxide capacitance  $C_{ox}$  and the depletion layer capacitance  $C_{dep}$ , according to relation:  $n = 1 + C_{dep}/C_{ox}$ . In small signal analysis, the slope factor  $n$  can be treated as constant and equal to a real number from interval 1 to 3 [Allen 2002]. In general, the slope factor  $n$  is in a convoluted relation with  $u_{GS}$ ,  $C_{dep}$  and  $C_{ox}$ , therefore, in a large signal analyses, it should be considered as a function of gate-source voltage, rather than a constant parameter [Toumazou 93b], [Mead 89]. The consequences of this effect will be further investigated in Chapter 3, when discussing the computational errors of CMOS multipliers.

Despite the limited accuracy of this model, equations (2.1) - (2.3) can be used in hand calculations providing results and conclusions useful for further circuit

optimisation, using more precise models such as BSIM (Berkeley Short-channel IGFET Model [Sheu 97]) and dedicated CAD tools.

### 2.2.2 Switched-current circuits

Computation in switched-current circuits typically employs memory cells for information storing and arithmetic operations. In such circuits, the information is represented by a current, which flows to the drain of a diode-connected transistor programming the gate voltage. After that, the gate disconnects from the drain and the transistor operates as a source generating the programmed current. Apart from the information storing purpose, such memory cells can also be used in the realisations of simple current-mode arithmetic operations such as addition, subtraction, multiplication and division by an integer number [Toumazou 93a, 93b].

There are many design challenges reported in the literature related to the realisations of SI circuits, such as charge injection, channel length modulation, gate leakage, noise, capacitive coupling, and parameter mismatch [Wegmann 89], [Daubert 88], [Fiez 91]. They affect the correct operation, accuracy and data retention time. Some of them, e.g. gate leakage, noise and parameter mismatch, are technology dependent and cannot be easily improved. However, effects such as charge injection, channel length modulation and parasitic coupling, can, to some extent, be reduced by using specific circuit solutions and by applying proper design and layout drawing techniques [Guggenbuhl 94], [Yang 90], [Toumazou 90a].

Charge injection is a complex process depending on the size and the driving scheme of the MOS switch connecting the gate and the drain of the information storing transistor [Wegmann 90]. Solutions addressing charge injection problem usually suggest the use of the minimum size switches, redundant dummy switches, individual switching scheme, or the use of additional gate capacitor [Guggenbuhl 94]. In some applications requiring higher precision, more complex structures were proposed including double (master-slave) memory cells [Leenaerts 94], algorithmic memory cells [Toumazou 90a], and S<sup>3</sup>I circuits [Carmona-Galan 2003]. These solutions employ multiple data storing transistors and more complex switching sequences to reduce charge injection errors.

Channel length modulation of the information storing transistor decreases its drain-source resistance, and hence, increases the error, depending on the voltage swing on the output node during transitions between read and write cycles. This can be reduced by employing cascode or regulated cascode memory cells, or by using cells with negative

feedback loops and DC servo amplifiers assuring a constant voltage on the analogue bus [Daubert 88], [Toumazou 90b]. Another circuit idea, addressing the charge injection, channel length modulation and signal dependent error cancellation, is based on the  $S^2I$  memory cell [Hughes 93]. Due to its compact structure, requiring only two information storing transistors, and simple switching sequence, such circuit is frequently considered in designs of analogue computing systems [Dudek 2000a].

Circuits based on the SI technique are usually less accurate than their continuous-time or switched-capacitor (SC) equivalents [Chang 96b]. The major disadvantage of the SI approach in current-mode computation is relatively slow operation in comparison to the continuous-time circuits. Especially, when the gate capacitances of the information storing transistors are enlarged (to prevent charge injection errors), time necessary to charge or discharge the gate increases. On the other hand, the realisation of the SI circuits, unlike switched-capacitor solutions, does not require precise linear capacitors nor high supply voltages to assure proper dynamic range [Leenaerts 96]. Most importantly, the correct operation of the switched-current circuits is practically not affected by fabrication mismatch, since the same transistor is used for storing and reading the information. Therefore, SI technique is usually considered in the analogue circuit realisations in standard CMOS technologies [Wegmann 89], [Dudek 2000].

### 2.2.3 Analogue multipliers

Analogue multipliers in CMOS technologies can be realised in various ways, employing different design strategies and ideas, usually based on the nonlinear characteristics of MOS transistors in continuous-time and discrete-time circuits, operating in switched current (SI) or switched capacitor (SC) modes. Such computational building blocks are used as analogue processing elements in adaptive filters, data converters, mixers and modulators in radio frequency (RF) and communication systems, and in parallel computing in neural networks and analogue processor arrays. In the realisations of analogue multipliers, meeting specific design requirements such as good linearity, wide dynamic range, low noise, high bandwidth and good matching, usually depends on a particular application. For example, in the analogue computation, linearity, dynamic range and parameter mismatch will be more important than noise or bandwidth [Han 98].

In the literature, analogue multipliers are classified depending on the operation range (single, two and four quadrant) and the operation mode, depending on whether the

voltage or the current represents the arguments. More generic classification, proposed in this review, differentiates between analogue multipliers based on the principles of signal multiplication. In particular, multiplication utilising variable gain amplifiers, nonlinear characteristics of MOS transistor, floating gate design and charge-based techniques will be discussed.

A multiplication of two analogue signals can be done using an amplifier with a variable gain, controlled by one of the signals, whereas the second one drives its input. There are a number of possible circuit realisations of this idea, however, some of them operate correctly only under the small signal assumption, where the characteristics of the amplifiers remain approximately linear. This often affects the precision and the dynamic range of the circuits, and usually requires differential representation of the signals to eliminate products stemming from nonlinear characteristics, therefore, such structures are mainly considered in RF and communication systems [Han 98].

Structures using MOS transistors operating in strong inversion and linear region, realise analogue multiplication based on the proportion between the drain current and the product of the gate and drain voltages, as shown in equation (2.2) [Shoemaker 91]. An example circuit realisation of such multiplier using two MOS transistors operating in linear region was presented in [Khachab 89]. There are many different possible realisations of such multiplier, based on differential approach with improved accuracy [Coban 95], [Lee 95], [Kub 90], switched capacitor (SC) circuits [Yasumoto 82], [Enomoto 85], and single transistor implementations used in programmable synaptic connections in CNN circuits [Dominguez-Castro 98], [Rodriguez-Vazquez 99], [Carmona-Galan 2003].

For structures using MOS transistors in strong inversion and saturation, the analogue multiplication is realised based on the proportion between the drain current and the square of the difference of the gate and the source voltages, as shown in equation (2.1) [Bult 87]. In practical realisation, the undesired components of the drain current can be removed assuming operation in differential mode, using crossed coupled differential pairs [Wang 93] or more complex structures based on multiple crossed-coupled circuits implementing quarter square rule for multiplication [Song 90]. Such circuits are typically realised using MOS squarers [Bult 86] or multiple-input floating gate transistors performing inherent voltage summations on the input gate capacitive dividers [Mehrvarz 95], [Ramirez-Angulo 96]. Discrete time realisation, based on a single squarer and the SI memory cells for data storage and current subtraction, was reported in

[Leenaerts 96], and its improved version using  $S^2I$  memory cells was presented in [Manganaro 98]. The use of the switched current technique allowed to store the intermediate results and use only one squarer circuit, which reduced the area and improved the matching, and the computational accuracy.

Structures using MOS transistors operating in weak inversion employ the exponential relation between the drain current and the gate voltage, given by the equation (2.3). Such multipliers are usually based on the Gilbert cell [Gilbert 68], realised as a set of crossed coupled differential pairs operating in the voltage-current (VCM) or current-current (CCM) mode. The VCM multipliers are highly nonlinear and operate correctly only in the small signal approximation [Mead 89]. Such circuits are used in the realisations of multipliers based on the quarter square identity [Liu 95]. The CCM multipliers, first convert the input currents to their voltage representations using diode connected MOS transistors, and then use these voltages to control the VCM circuits [Song 93], [Gravati 2005a]. Such multipliers belong to the class of *translinear* circuits, initially defined for bipolar transistors [Gilbert 75], and later extended to the circuits with MOS transistors, operating in weak inversion [Andreou 96]. In these circuits, MOS transistors are connected in loops, such that the gate source voltages around each loop sum to zero, therefore the corresponding drain currents remain in certain linear proportions [Toumazou 93b]. Such circuits allow for the operation in a wide dynamic range, thus are frequently considered in the realisations of multiplier and divider circuits for analogue computation [Gravati 2005b], [Andreou 96]. It is important to note that such multipliers, when operating in the current-current mode, perform not only multiplication but also normalisation of the computed results with respect to the input arguments [Gilbert 84]. Although such normalisation is usually undesired and avoided by using differential approach, it is advantageous in the realisations of the sum-product algorithm for belief propagation, discussed in Chapter 7 [Pearl 88], [Luckenbill 2002].

There exist other ways of calculating products of analogue quantities using electronic circuits, not necessarily employing any of the inherent features of MOS transistors. For example, a sampled data multiplier operating in voltage-voltage mode, based on time and current control in charging a capacitor, was proposed in [Brodarc 82]. A very compact but less accurate, single quadrant charge-based multipliers were proposed for the applications in spiking neural networks in the realisations of programmable synaptic connections [Dominguez-Castro 98], [Massengill 91]. Also, the application of the switched capacitor (SC) technique in multiplication and division was proposed in

[Watanabe 84]. Approach using ideal log amplifiers in realisations of multiplication, division and raising to a power was presented in [Grundy 94].

The most promising circuits for analogue computation can be found among structures operating in the large signal current-current mode (CCM) in weak inversion, and among structures realised in switched-current (SI) mode. In the former ones, algebraic operations can be realised using translinear principle and inherent summation of the current flowing into a common node. Unfortunately, the computational accuracy of such circuits is highly degraded by fabrication mismatch, unless very large MOS transistors are used [Gravati 2005b]. Switched-current circuits, on the other hand, are not affected by mismatch but operate slower and generate additional errors, resulting from charge injection, leakage and other effects. Therefore, hybrid solutions such as *translinear SI circuits*, should be considered in the designs of analogue computational systems. For example, the realisation of the analogue multiplier proposed in [Manganaro 98], used  $S^2I$  memory cells and a single arithmetic squarer operating in strong inversion, giving very good performance parameters. The idea and design of a multiplier based on the Gilbert cell and operating in switched-current mode is one of the contributions of this research, discussed in detail in Chapter 3 and used in the realisation of the sum-product algorithm for belief propagation in Chapter 8.

### 2.3 Parameter variability in CMOS technologies

An inherent shortcoming of any standard CMOS process is a certain degree of random variability of the physical parameters of the manufactured devices. Usually, the nature of these variations can be seen as a global, inter-die randomness (i.e. chip to chip, wafer to wafer or batch to batch), and a local (uncorrelated) one, randomly affecting the parameters of equally designed and closely laid out devices. Global variation can be attributed to the randomness in the manufacturing process, resulting in a systematic shift in the absolute values of all the device parameters within one wafer or one batch. Its influence on the correct circuit operation can be minimized when a design relies solely on the parameter ratios rather than their absolute values. Local parameter variation, known as fabrication mismatch, is more difficult to mitigate and may significantly degrade the accuracy and performance of a circuit [Pelgrom 89]. For example, it limits the accuracy of A/D and D/A converters [Pelgrom 98], increases the offset voltage in operational amplifiers, distorts the symmetry of current mirrors [Shyu 84], constraints the

speed-power-accuracy trade off in analogue systems [Kinet 97], and affects the voltage and timing margins in digital circuits [Christiansen 95], [Toifl 99], [Lovett 2000]. Moreover, the fabrication mismatch is being reported not to scale down linearly with the technology feature size but the variability of the parameters of a minimum-size MOS device increases when moving to finer nodes [Mead 94], [Rodriguez-Vazquez 2003]. Therefore, it becomes necessary to account for fabrication mismatch, its analysis, modelling and optimisation in a standard design flow of VLSI circuits, in order to avoid overdesign and properly estimate yield and the manufacturing costs [Cox 85].

In the following section, two approaches to mismatch modelling: based on the physical grounds [Lakshmikumar 86], and based on empirical analysis [Pelgrom 89], will be presented. Also, basic mathematical methods for quantifying the impact of parameter variability on a device operation will be provided.

### 2.3.1 Physical model

Mismatch model evaluated based on the study of the physical parameter randomness, was discussed in [Shyu 84] and [Lakshmikumar 86]. In the proposed analyses, it was assumed that the fabrication mismatch was dominated mainly by the randomness in the gate charge distribution and the channel doping concentration. Due to certain variability in the process of masking and ion implantation, dopants typically occupy random locations with distribution close to uniform across a given surface. It has been shown in theory [Nicollian 82] and verified in experiments [Shyu 84] that the variability of such charge density reduces with respect to the total device area. For example, the variability of the depletion charge density  $\sigma_{Qdep}^2$  over a channel area  $WL$  can be calculated as:

$$\sigma_{Qdep}^2 = \frac{Q_{dep}}{WL} \quad (2.4)$$

The same approach was applied to quantify the variability of other physical parameters such as oxide thickness  $t_{ox}$  and carrier mobility  $\mu$ . The variances of the threshold voltage  $V_{th}$  and current factor  $\beta$  were calculated using equations from Section 2.2.1 and the method of variance propagation from statistics (see Section 2.3.3).

### 2.3.2 Empirical model

The empirical model is based on the mathematical analysis assuming that the surface variability of a parameter  $P$  is given by a two-dimensional density function  $P(x, y)$

consisting of a fixed and random parts, with arguments  $x$  and  $y$  indicating the location on a plane [Pelgrom 89]. It is important to note that such abstract model was constructed with intention to show analogies to physical parameter variability across a chip die or a wafer. The value of the parameter  $P$ , in a certain location  $(x, y)$ , is represented by its average and calculated by integrating the corresponding density function  $P(x, y)$  over the area around this location. For two values of the same parameter  $P$  but obtained in different locations, the mismatch was defined as the difference between their respective mean values  $\Delta P(x, y)$ . The resulting variance of parameter  $\Delta P$  between two identically drawn rectangular devices of area  $WL$  in distance  $D$  between their centres, is given by the formula:

$$\sigma_{\Delta P}^2 = \frac{A_P^2}{WL} + S_P^2 D^2 \quad (2.5)$$

where  $A_P$  and  $S_P$  are the process dependent parameters corresponding respectively to the local random variation, which averages out with the device area, and to the long distance parameter variation, usually causing parameter offsets and gradients across the silicon dies. The derivation of the Pelgrom's model can be found in [Linares-Barranco 2007].

It is important to note that the distance dependent component  $D$  in equation (2.5), although verified in measurements [Pelgrom 89], [Bastos 95], has rather negligible impact on the total parameter variability. In order to quantify the magnitude of the long distance parameter variability, a corner distance  $D_m$  was introduced and defined as the distance between two identical smallest-size MOS transistors, manufactured in a CMOS technology of a particular  $\lambda_T$  feature size, where the components  $A_P^2/WL$  and  $S_P^2 D^2$ , in the equation (2.5), become equal [Kinet 97]:

$$D_m = \frac{A_P}{\lambda_T S_P} \quad (2.6)$$

The corner distance for the threshold voltage  $V_{th}$  and the current factor  $\beta$ , reported for a 2.5  $\mu\text{m}$  CMOS technology, was greater than 3 mm [Pelgrom 89], and already greater than 14 mm for a 0.7  $\mu\text{m}$  CMOS technology [Kinet 97]. Since the corner distance  $D_m$  is much bigger than the size of a typical integrated circuit or even a full chip, the distance dependent component in equation (2.5) can usually be neglected, and only the area dependent mismatch can be considered.

It can be concluded that, irrespective of the approach (either based on physical or empirical studies), the variability of the threshold voltage  $\sigma_{\Delta V_{th}}^2$  and the current factor  $\sigma_{\Delta \beta}^2$  of the square-law MOS transistor model are given by the following equations:

$$\sigma_{\Delta V_{th}}^2 = \frac{A_{V_{th}}^2}{WL} \quad (2.7)$$

$$\frac{\sigma_{\Delta \beta}^2}{\beta^2} = \frac{A_{\beta}^2}{WL} \quad (2.8)$$

where  $A_{V_{th}}$  and  $A_{\beta}$  are the technology dependent constant parameters, usually extracted from measurements.

### 2.3.3 Variability propagation

Standard MOS transistor models, such as Spice Level 1 or BSIM, define mathematical relations between the operating point of the device (i.e. drain current and gate-source voltage) and the electrical parameters of the transistors, such as threshold voltage  $V_{th}$  and current factor  $\beta$ . When building statistical models of MOS transistors, it is essential to "propagate" the variability of the model parameters (e.g.  $\sigma_{\Delta V_{th}}^2$ ,  $\sigma_{\Delta \beta}^2$ ) on the variability of the drain current or the gate-source voltage. This is typically done by calculating the approximate variance  $\sigma_F^2$  of a function  $f$  of  $n$  random variables  $f(P_1, P_2, \dots, P_n)$  with variances  $\sigma_{P_1}^2, \sigma_{P_2}^2, \dots, \sigma_{P_n}^2$  using formula [Abel 93], [Papoulis 2002]:

$$\sigma_F^2 = \sum_{i=1}^n \left( \frac{\partial f}{\partial P_i} \right)^2 \sigma_{P_i}^2 + \sum_{i=1}^n \sum_{j=i+1}^n 2\rho_{ij} \frac{\partial f}{\partial P_i} \frac{\partial f}{\partial P_j} \sigma_{P_i}^2 \sigma_{P_j}^2 \quad (2.9)$$

where  $\rho_{ij}$  is the correlation factor between parameters  $P_i$  and  $P_j$ . In practice, it may be the case that some of the correlation coefficients are negligibly small or zero. The physical parameters of CMOS circuits can, in the majority of cases, be represented as a set of independent random variables. However, the variability of the resulting electrical parameters, specific to a particular MOS transistor model, may be correlated. This happens when they rely on the same physical parameters and phenomena. For example, the variability of the gate oxide thickness affects the gate oxide capacitance, and hence, contributes to the variability of the threshold voltage  $V_{th}$ , and also to the variability of the current factor  $\beta$ . However, the experimental results have shown that the correlation between  $V_{th}$  and  $\beta$  is negligibly small [Kinget 2005]. In such a case, when all the

parameters  $P_1, \dots, P_n$  can be treated as uncorrelated random variables, the variance from (2.9) simplifies to the following equation:

$$\sigma_F^2 = \left( \frac{\partial f}{\partial P_1} \right)^2 \sigma_{P_1}^2 + \left( \frac{\partial f}{\partial P_2} \right)^2 \sigma_{P_2}^2 + \dots + \left( \frac{\partial f}{\partial P_n} \right)^2 \sigma_{P_n}^2 \quad (2.10)$$

In some mathematical considerations, it is important to differentiate between the variance of a random parameter  $P$  equal  $\sigma_P^2$  and the variance of the difference  $\Delta P$  of the random values of the same parameter  $P$  equal  $\sigma_{\Delta P}^2$  [Kinget 97]. Since equation (2.5) provides a method for calculating variance of the parameter difference, in some cases, it is necessary to calculate the variability of the parameter  $P$  itself. Assuming that  $\Delta P$  is a difference of two uncorrelated and randomly generated values of the same parameter  $\Delta P = P' - P''$  and using (2.10), the variance equals  $\sigma_{\Delta P}^2 = \sigma_{P'}^2 + \sigma_{P''}^2$ . Values  $P'$  and  $P''$  are samples of the same parameter  $P$ , therefore, their variances are equal:  $\sigma_{P'}^2 = \sigma_{P''}^2$ , and the variance  $\sigma_P^2$  of parameter  $P$  can be calculated as  $\sigma_P^2 = \sigma_{\Delta P}^2 / 2$ .

## 2.4 Mismatch modelling

The majority of works considering mismatch modelling focus mainly on constructing reliable statistical models applicable over a wide range of geometry sizes and different bias conditions. It should be noted, however, that mismatch modelling does not necessarily require constructing a new MOS transistor model. The main problem lies in the correct identification of different physical sources of randomness in the fabrication process, and in the analysis of their influence on the electrical parameters of MOS devices. This is specific to the model chosen and to the random parameter cross correlations.

In the following section, simplified MOS transistor model for mismatch analysis, used in this thesis, will be derived based on the Spice Level 1 model presented in section 2.2.1 and the method of variance propagation from equation (2.10). For simplicity zero correlation between variability of the threshold voltage and the current factor  $\beta$  will be assumed.

### 2.4.1 Mismatch MOS model for hand calculations

The implications of parameter mismatch on the operation of analogue circuits were extensively studied in [Kinget 96, 97, 2005]. Similarly to noise, the impact of parameter

mismatch on the drain current  $i_D$  or on the gate-source voltage  $u_{GS}$  is usually considered, assuming small signal approximation, depending on the operation mode of the transistor. Assuming a generic relation for the drain current:  $i_D = f(u_{GS}, u_{DS}, V_{th}, \beta)$  as a function of bias voltages  $u_{GS}$  and  $u_{DS}$ , and parameters  $V_{th}$  and  $\beta$ , the following difference equation can be derived:

$$\Delta I_D = \frac{\partial i_D}{\partial \beta} \Delta \beta + \frac{\partial i_D}{\partial V_{th}} \Delta V_{th} \quad (2.11)$$

Using (2.10), the variability of the drain current in (2.11) can be calculated as:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{\sigma_{\Delta \beta}^2}{\beta^2} + \left( \frac{g_m}{I_D} \right)^2 \sigma_{\Delta V_{th}}^2 \quad (2.12)$$

where  $g_m = \partial i_D / \partial u_{GS}$  is a small signal transconductance of MOS transistor calculated for the assumed bias point determined by  $u_{GS}$ ,  $u_{DS}$  and  $i_D$ . The variability of the gate-source voltage for a fixed drain current can be calculated based on the following difference equation:

$$\Delta U_{GS} = \frac{\partial u_{GS}}{\partial \beta} \Delta \beta + \frac{\partial u_{GS}}{\partial V_{th}} \Delta V_{th} \quad (2.13)$$

The partial derivatives in (2.13) can be calculated using  $i_D + f(u_{GS}, u_{DS}, V_{th}, \beta) = C$ , where  $C$  is a constant. Applying (2.10) to (2.13), the following equation for the variability of gate-source voltage can be derived:

$$\sigma_{\Delta U_{GS}}^2 = \sigma_{\Delta V_{th}}^2 + \left( \frac{I_D}{g_m} \right)^2 \frac{\sigma_{\Delta \beta}^2}{\beta^2} \quad (2.14)$$

The impact of the parameter variability on the drain current (for the fixed bias voltages) or on the gate-source voltage (for the fixed drain current), given by the equations (2.12) and (2.14) respectively, can be calculated for each region of operation of a MOS transistor, using equations (2.1), (2.2) and (2.3).

#### 2.4.2.1 Strong inversion and saturation

In strong inversion and saturation the drain current of a MOS transistor is given by equation (2.1) and the corresponding variability parameters of the drain current and gate-source voltage are equal:

$$\frac{\sigma_{\Delta ID}^2}{I_D^2} = \frac{\sigma_{\Delta \beta}^2}{\beta^2} + \frac{4\sigma_{\Delta V_{th}}^2}{(u_{GS} - V_{th})^2} \quad (2.15)$$

$$\sigma_{\Delta V_{GS}}^2 = \sigma_{\Delta V_{th}}^2 + \frac{(u_{GS} - V_{th})^2}{4} \frac{\sigma_{\Delta \beta}^2}{\beta^2} \quad (2.16)$$

Inserting the area dependent components  $\sigma_{\Delta V_{th}}^2$  and  $\sigma_{\Delta \beta}^2/\beta^2$  from the equations (2.7) and (2.8) respectively, to (2.15) and (2.16), the following equations can be derived:

$$\frac{\sigma_{\Delta ID}^2}{I_D^2} = \frac{1}{WL} \left[ A_{\beta}^2 + \frac{4A_{V_{th}}^2}{(u_{GS} - V_{th})^2} \right] \quad (2.17)$$

$$\sigma_{\Delta V_{GS}}^2 = \frac{1}{WL} \left[ A_{V_{th}}^2 + \frac{(u_{GS} - V_{th})^2}{4} A_{\beta}^2 \right] \quad (2.18)$$

It can be observed, that the propagation of the parameter mismatch on the drain current  $I_D$  and the gate-source voltage  $V_{GS}$  depends on the gate area  $WL$  of a MOS transistor but also on the operating point, determined by voltage  $u_{GS}$ . For very low or very high values of  $u_{GS} - V_{th}$ , the variability of one parameter, either the threshold voltage or the current factor becomes dominant. To address such cases, a corner gate-drive voltage  $u_{GST} = (u_{GS} - V_{th})_m$ , has been defined as the bias condition where the effects of mismatch in  $V_{th}$  and  $\beta$  are equal. In strong inversion and saturation, the corner gate-drive voltage equals  $u_{GST} = 2A_{V_{th}}/A_{\beta}$  [Kinget 97]. In particular, for bias conditions such that  $(u_{GS} - V_{th}) > u_{GST}$ , the variability in current factor  $A_{\beta}$  dominates in equation (2.17), whereas for  $(u_{GS} - V_{th}) < u_{GST}$ , the variability in the threshold voltage  $A_{V_{th}}$  is a major contributor to mismatch. Since the corner gate-drive voltage is usually high, the transistor will most likely operate on the gate-source bias voltages meeting relation  $(u_{GS} - V_{th}) < u_{GST}$ , where the mismatch in  $V_{th}$  is dominant. For example, the corner gate-drive voltage for low leakage nMOS transistor in a 90 nm CMOS technology is 0.66 V. For the threshold voltage  $V_{th} = 0.45$  V and supply voltage 1.2 V, the required gate-source voltage  $u_{GS}$  assuring  $(u_{GS} - V_{th}) > u_{GST}$  must be higher than 1.11 V. Assuming that, in most cases,  $(u_{GS} - V_{th}) < u_{GST}$ , the equations (2.17) and (2.18) can be simplified to the following form:

$$\frac{\sigma_{\Delta ID}^2}{I_D^2} = \frac{1}{WL} \frac{4A_{V_{th}}^2}{(u_{GS} - V_{th})^2} \quad (2.19)$$

$$\sigma_{\Delta V_{GS}}^2 = \frac{1}{WL} A_{V_{th}}^2 \quad (2.20)$$

#### 2.4.2.2 Strong inversion and linear region

In strong inversion and linear region, the drain current of a MOS transistor is given by equation (2.2) and the corresponding variability parameters are equal:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{\sigma_{\Delta \beta}^2}{\beta^2} + \frac{\sigma_{\Delta V_{th}}^2}{(u_{GS} - V_{th} - u_{DS}/2)^2} \quad (2.21)$$

$$\sigma_{\Delta V_{GS}}^2 = \sigma_{\Delta V_{th}}^2 + (u_{GS} - V_{th} - u_{DS}/2)^2 \frac{\sigma_{\Delta \beta}^2}{\beta^2} \quad (2.22)$$

Assuming deep linear operation, where  $u_{DS} \approx 0$ , the equations (2.21) and (2.22) can be simplified to:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{\sigma_{\Delta \beta}^2}{\beta^2} + \frac{\sigma_{\Delta V_{th}}^2}{(u_{GS} - V_{th})^2} \quad (2.23)$$

$$\sigma_{\Delta V_{GS}}^2 = \sigma_{\Delta V_{th}}^2 + (u_{GS} - V_{th})^2 \frac{\sigma_{\Delta \beta}^2}{\beta^2} \quad (2.24)$$

Inserting the area dependent components  $\sigma_{\Delta V_{th}}^2$  and  $\sigma_{\Delta \beta}^2/\beta^2$  from equations (2.7) and (2.8) respectively, to (2.23) and (2.24), the following equations can be derived:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{1}{WL} \left[ A_{\beta}^2 + \frac{A_{V_{th}}^2}{(u_{GS} - V_{th})^2} \right] \quad (2.25)$$

$$\sigma_{\Delta V_{GS}}^2 = \frac{1}{WL} \left[ A_{V_{th}}^2 + (u_{GS} - V_{th})^2 A_{\beta}^2 \right] \quad (2.26)$$

Based on the equations (2.25) and (2.26), two important conclusions can be formulated. Firstly, the corresponding corner gate-drive voltage  $u_{GST} = A_{V_{th}}/A_{\beta}$  is twice lower than in the saturation, therefore, the dominant source of the parameter variability is rather bias dependent. Secondly, for the same gate source voltage, the variability of the drain current in saturation (2.17) is higher than in linear region (2.25). Therefore, in some analogue circuits, for example in single transistor multipliers for synaptic connections in CNN, ohmic region of MOS transistor was suggested for better accuracy [Rodriguez-Vazquez 99], [Carmona-Galan 2003].

### 2.4.2.3 Weak inversion

In weak inversion and saturation the drain current of a MOS transistor is given by equation (2.3) and the corresponding variability parameters of the drain current and gate-source voltage are equal:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{\sigma_{\Delta \beta}^2}{\beta^2} + \frac{\sigma_{\Delta V_{th}}^2}{(nU_T)^2} \quad (2.27)$$

$$\sigma_{\Delta V_{GS}}^2 = \sigma_{\Delta V_{th}}^2 + (nU_T)^2 \frac{\sigma_{\Delta \beta}^2}{\beta^2} \quad (2.28)$$

Based on the experimental results, it has been shown that, in weak inversion, the variability of the drain current and gate-source voltage is dominated by the mismatch in the threshold voltage. For example, assuming  $n = 1.5$  for low leakage nMOS transistor from a 90 nm CMOS technology, and inserting the area dependent components  $\sigma_{\Delta V_{th}}^2$  and  $\sigma_{\Delta \beta}^2/\beta^2$  from equations (2.7) and (2.8) respectively, to (2.27) and (2.28), the ratio  $A_\beta nU_T/A_{V_{th}} \approx 0.12$  which means that the contribution of the threshold voltage component in (2.27) and (2.28) is almost ten times higher than the contribution of the current factor. Based on that, the equations (2.27) and (2.28) can be represented in the simplified forms:

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{1}{WL} \left[ \frac{A_{V_{th}}^2}{(nU_T)^2} \right] \quad (2.29)$$

$$\sigma_{\Delta V_{GS}}^2 = \frac{1}{WL} A_{V_{th}}^2 \quad (2.30)$$

The ratio of the drain current variability in strong inversion (2.19) to the current in weak inversion (2.29) equals  $nU_T/(u_{GS} - V_{th})$ . Assuming typical bias conditions of the low leakage nMOS transistor from a 90 nm CMOS technology operating in strong inversion and saturation ( $u_{GS} = 0.8$  V,  $V_{th} = 0.45$  V, see Figure 2.1), the variability of the drain current in weak inversion is almost ten times higher than in strong inversion.

### 2.4.3 Statistical MOS models for CAD

The simplified MOS model, considered in previous sections, was convenient for hand calculations and sufficiently precise for the technology feature size of 1  $\mu$ m or higher. However, it may no longer be practically justifiable for submicron and deep

submicron CMOS processes, where the second order effects in MOS devices, such as short channel effects, mobility degradation and body bias effect, start to play an important role. Therefore, the use of more complex, compact models such as BSIM, providing a better approximation of MOS transistor behaviour, is usually considered in circuit simulations [Bhattacharyya 2009].

Due to the high complexity and the convoluted form of the equations in BSIM model, the hand analysis of MOS transistor behaviour and circuit operation under model parameter variability is no longer feasible. Instead, statistical Monte Carlo sampling approach was proposed, using randomly generated sets of model parameters in multiple circuit simulations. The variability of particular circuit parameters (e.g. offset voltage in an operational amplifier) could be calculated directly from the statistics of the obtained simulation results.

Given the improved accuracy of the members of BSIM family such as BSIM4 and BSIM5 [He 2007], dedicated for CMOS processes below 65 nm feature size, the complexity of such models, and hence, the resulting simulation time increases. Since Monte Carlo analysis requires multiple simulation runs, it becomes critically important to devise a sensible trade off between the accuracy of the statistical model, and its complexity. This is usually achieved on the empirical bases assuming random fluctuation of only selected parameters, such that, the statistical model fits into the data obtained from the measurements [Drennan 2003]. The number of iterations in Monte Carlo simulations should also be chosen individually, according to the required accuracy of the parameter variability estimation. The relative error is inversely proportional to the square root of the number of simulation runs, therefore, one order of magnitude accuracy improvement requires two orders of magnitude more iteration runs. To address the problem of long simulation time, methods employing macromodel extraction, or approaches assuming limited number of runs, producing only critical circuit configurations, were considered [Michael 92, 96].

Even though Monte Carlo simulations using statistical BSIM models provides much more accurate tool for circuit variability estimation, the simplest square-law MOS transistor model is still frequently used in the hand analyses, providing insightful conclusions and directions for mismatch optimisation using more precise methods.

#### 2.4.4 Extended simplified model

In order to improve the accuracy of the simplified model discussed in section 2.2.1, certain improvements accounting for the second order effects such as gate roughness, effective mobility reduction, short and narrow channel effects and active gate area reduction, were proposed [Steyaert 94], [Bastos 95].

Gate roughness is defined as an irregularity of the polysilicon and diffusion regions forming MOS transistor gate. The impact of the resulting variations in the gate width and length on the mismatch of  $\beta$  is given by the equation [Pelgrom 89]:

$$\frac{\sigma_{\beta}^2}{\beta^2} = \frac{1}{WL} (A_{ox} + A_{\mu}) + \frac{A_W}{W^2 L} + \frac{A_L}{WL^2} \quad (2.31)$$

where  $A_W$  and  $A_L$  are constant process parameters. Important conclusion can be drawn when calculating minimum of the function  $\sigma_{\beta}^2/\beta^2$  from (2.31) for a constant area  $WL = A$ . It can be shown that there exists an optimal ratio  $W/L = A_W^2/A_L^2$ , minimising the variability of the current factor  $\beta$ . This is particularly important in the mismatch optimisation based on geometry scaling, where an improvement of the accuracy could be achieved by a proper selection of  $W/L$  ratio for a constant gate area.

Several important effects observed in the small geometry MOS transistors such as Drain Induced Barrier Lowering (DIBL) and Short Channel Effect (SCE) result from the interaction between the depletion charge  $Q_{dep}$  (induced by the gate-bulk bias voltage) and the depletion regions surrounding reverse biased drain-bulk and source-bulk  $p-n$  junctions. Such interaction between the depletion regions and the channel area can be seen as an additional source of randomness, contributing to the variability of the threshold voltage. Attempts to incorporate such effects in the simplest square-law model were proposed in [Steyaert 94] and [Bastos 95]. It was observed that the variability of the threshold voltage increases for wide and short channel devices, when the influence of the drain voltage on the channel depletion charge was higher. This is particularly important in analogue design, where wide and short channel transistors are often used (e.g. in the input stages of amplifiers to achieve large transconductance).

### 2.5 Mismatch scaling

Assuming constant field scaling rules in CMOS technologies, certain physical and electrical parameters of MOS devices increase or decrease by a constant factor  $K$ , which

scales with the technology feature size  $\lambda_T \sim 1/K$  [Wong 83]. In particular, lateral and vertical dimensions, such as channel width  $W$ , length  $L$ , and gate oxide thickness  $t_{ox}$  are assumed to scale proportionally to  $1/K$ , and the doping concentration  $N_D$  proportionally to  $K$ . In order to avoid punch through effects and drain-bulk junction breakdown, the bias and supply voltages should also scale proportionally to  $1/K$ . Although constant field scaling rules are not precise when applied to submicron technologies, they are sufficient for the approximate analyses presented in this section.

The variability of the threshold voltage of a MOS transistor depends mainly on the fluctuations of the depletion charge  $Q_{dep}$  stored on the gate capacitance  $C_{ox}$  and can be calculated from the approximate formula  $\sigma_{V_{th}}^2 \approx \sigma_{Q_{dep}}^2 / C_{ox}^2$  [Lakshmikumar 86]. On the other hand, the variability of the depletion charge  $Q_{dep}$ , given by (2.4), equals  $\sigma_{Q_{dep}}^2 = Q_{dep} / WL$ . Combining the last two equation and assuming  $C_{ox} = \epsilon_{ox} / t_{ox}$  and  $Q_{dep} = \sqrt{2\psi\epsilon_S q N_D}$  [Allen 2002], the following equation can be derived:

$$\sigma_{V_{th}}^2 = \frac{1}{WL} \left( \frac{t_{ox}}{\epsilon_{ox}} \right)^2 \sqrt{2\psi\epsilon_S q N_D} \quad (2.32)$$

where  $WL$  is the gate area,  $t_{ox}$  is the oxide thickness,  $\epsilon_{ox}$  and  $\epsilon_S$  is the electrical permittivity of the oxide layer and the semiconductor (silicon) respectively,  $\psi$  is the surface potential inducing the charge dislocation,  $q$  is the elementary electric charge, and  $N_D$  is the doping concentration of the semiconductor. Inserting the area dependent component of the threshold voltage variability from (2.7), equal  $\sigma_{V_{th}}^2 = A_{V_{th}}^2 / WL$ , to (2.32) it can be shown the technology parameter  $A_{V_{th}}$  equals:

$$A_{V_{th}} = \left( \frac{t_{ox}}{\epsilon_{ox}} \right) (2\psi\epsilon_S q N_D)^{1/4} \sim t_{ox} N_D^{1/4} \quad (2.33)$$

and is proportional to  $t_{ox} N_D^{1/4}$ . Assuming scaling rules:  $t_{ox} \sim 1/K$  and  $N_D \sim K$ , the mismatch parameter  $A_{V_{th}}$  is proportional to:

$$A_{V_{th}} \sim K^{-3/4} \sim \lambda_T^{3/4} \quad (2.34)$$

From the equation (2.34) it can be seen that the parameter  $A_{V_{th}}$  scales down with the technology feature size and becomes smaller when moving to finer nodes. Even though the intrinsic matching of the threshold voltage improves for a fixed gate size, there exist other physical limitations such as SCE and DIBL effects, introducing more sources of parameter variability, further reducing the precision of MOS transistor operation

[Bastos 95]. It should also be noted that technology scaling imposes lower supply and bias voltages reducing the dynamic range of analogue circuits, and hence, further magnifying the impact of parameter variability on the correct operation of such systems [Kinget 97].

In the case of the current factor  $\beta$ , there is no clear relation between the mismatch figure  $A_\beta$  and the process parameters. In the literature it has only been speculated that the variability of  $\beta$  could depend on the fluctuation of the gate oxide thickness  $t_{ox}$ , gate roughness, and most likely, on the variability of the carrier mobility. The measurement results reported in literature indicate a very weak scaling of  $A_\beta$  with the technology feature size, slightly improving when moving to finer nodes [Kinget 98], [Rodriguez-Vazquez 2003]. It should be noted, however, that these conclusions were drawn based on the measurements done in 1990s for CMOS technologies above 0.35  $\mu\text{m}$  feature size. When moving to finer nodes, certain reduction of the variability of parameter  $A_\beta$  can be observed (Table 2.1 and Figure 2.2b). Nevertheless, more experimental data for technologies below 0.35  $\mu\text{m}$  feature size is needed to confirm such observation.

The values of mismatch parameters  $A_{V_{th}}$  and  $A_\beta$ , for different standard CMOS technology nodes, reported in the literature and in the foundry documentation, are presented in Table 2.1. It should be noted that these parameters are technology dependent and may vary for the same node fabricated by different foundries [Pelgrom 2010]. Trends showing the scaling of the average values of mismatch parameters of nMOS and pMOS transistors in terms of the technology feature sizes  $\lambda_T$  are presented in Figure 2.2.

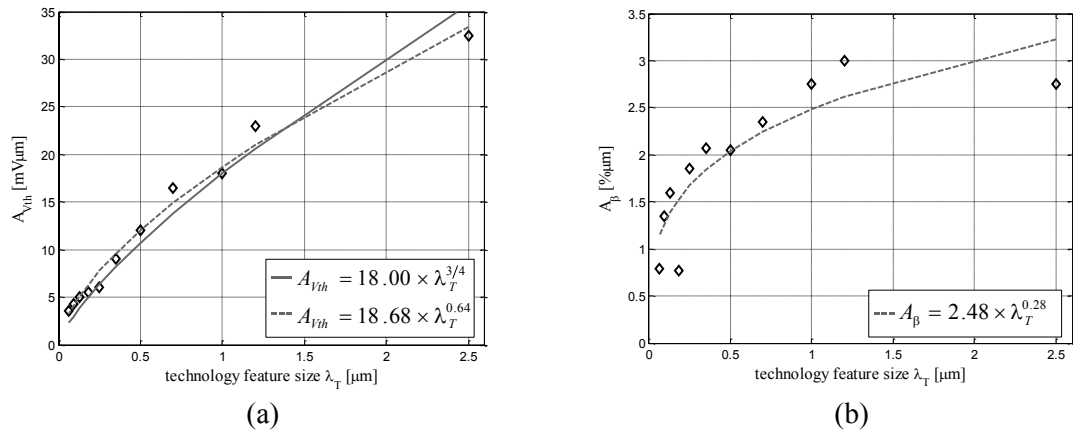


Figure 2.2. Scaling trends of mismatch parameters a)  $A_{V_{th}}$  and b)  $A_\beta$  in terms of technology feature size.

Table 2.1. Matching parameters  $A_{V_{th}}$  and  $A_\beta$  for standard CMOS processes of different technology feature size.

<i>Technology</i> ( $\lambda_T$ )	$t_{ox}$ [nm]	<i>Type</i>	$A_{V_{th}}$ [mV· $\mu$ m]	$A_\beta$ [%· $\mu$ m]	<i>Reference</i>
2.5 $\mu$ m	50	nMOS	30	2.3	[Pelgrom 89]
		pMOS	35	3.2	
1.2 $\mu$ m	22.5	nMOS	21	1.8	[Bastos 95]
		pMOS	25	4.2	
1 $\mu$ m	17.5	nMOS	13	2.5	[Bolt 96]
		pMOS	23	3.0	
0.7 $\mu$ m	17	nMOS	11	1.9	[Bastos 97a]
		pMOS	22	2.8	
0.5 $\mu$ m	12	nMOS	11	1.8	[Pelgrom 2010]
		pMOS	13	2.3	
0.35 $\mu$ m	7.7	nMOS	9	1.9	[Pelgrom 2010]
		pMOS	9	2.25	
0.25 $\mu$ m	6	nMOS	6	1.85	[Pelgrom 2010]
		pMOS	6	1.85	
0.18 $\mu$ m	3.3	nMOS	5.23	0.61	[FDK 2009]
		pMOS	5.85	0.93	
0.13 $\mu$ m	2.5	nMOS	5	1.6	[FDK 2009]
		pMOS	5	1.6	
90 nm	2.2	nMOS	5.14	1.56	[FDK 2009]
		pMOS	3.43	1.14	
65 nm	2.6	nMOS	4.18	0.89	[FDK 2009]
		pMOS	2.99	0.69	

In the case of the parameter  $A_{V_{th}}$ , the scaling rule derived in (2.34), represented by function  $y(\lambda_T) = C\lambda_T^{3/4}$  for  $C = 18.00$ , was evaluated using square error minimisation method and also added in the figure for reference. The same optimisation procedure was repeated for the function  $y(\lambda_T) = C\lambda_T^D$ , giving:  $C = 18.68$  and  $D = 0.64$ . It can be observed that the variability scaling of the threshold voltage extracted from the data is slightly higher than the variability given by equation (2.34). This results from the simplifications assumed in the derivation of (2.34), for example, not including the variability of gate oxide thickness, trapped charges in gate etc. The prediction of the parameter  $A_\beta$  scaling, with respect to the technology feature size, was made using function  $y(\lambda_T) = C\lambda_T^D$  and the same optimisation procedure. The obtained parameters are:  $C = 2.48$  and  $D = 0.28$ .

## 2.6 Mismatch versus noise

The experimental results reported in the literature shows that random errors caused by fabrication mismatch are about one to two orders of magnitude higher than the errors stemming from noise [Kinget 97]. Assuming that MOS transistor operates in saturation,

the variability of the equivalent input (gate) voltage can be calculated from equation (2.20) as  $\sigma_{\Delta V_{GS}}^2 = A_{V_{th}}^2 / WL$ , and the variability of this voltage as  $\sigma_{V_{GS}mismatch}^2 = A_{V_{th}}^2 / 2WL$ . For simplicity, only the total wideband thermal noise  $kT/C$  of the gate capacitance  $C_g$ , will be considered. The resulting noise voltage variability is equal to  $\sigma_{V_{GS}noise}^2 = kT/C_g$ . Assuming that the gate capacitance equals  $C_g = C_{ox}WL$ , the ratio of the input voltage variability caused by mismatch and noise is:

$$\frac{\sigma_{V_{GS}mismatch}^2}{\sigma_{V_{GS}noise}^2} = \frac{A_{V_{th}}^2 C_{ox}}{2kT} \quad (2.35)$$

For example, for the low leakage nMOS transistor from a 90 nm CMOS technology (assuming  $A_{V_{th}} = 5.14 \text{ mV} \cdot \mu\text{m}$  and  $t_{ox} = 2.2 \text{ nm}$ ), the ratio given by (2.35), equals approximately 50, at the room temperature  $T = 300 \text{ K}$ . This means that the error introduced by mismatch is 50 times larger than the error stemming from thermal noise. Using (2.34) and applying the constant field scaling rules to equation (2.35), and assuming  $C_{ox} = \epsilon_{ox}/t_{ox}$ , the following relation can be derived:

$$\frac{\sigma_{V_{GS}mismatch}^2}{\sigma_{V_{GS}noise}^2} \sim A_{V_{th}}^2 \frac{\epsilon_{ox}}{t_{ox}} \sim K^{-6/4} K^1 \sim K^{-1/2} \sim \sqrt{\lambda_T} \quad (2.36)$$

The mismatch to noise ratio from (2.35), with respect to technology feature size  $\lambda_T$  and calculated using data from Table 2.1, is presented in Figure 2.3. The scaling trend approximated by function  $y(\lambda_T) = C\sqrt{\lambda_T}$ , was evaluated using square error minimisation method giving  $C = 71.88$ . The optimisation procedure was also repeated for the function  $y(\lambda_T) = C\lambda_T^D$  giving  $C = 71.73$  and  $D = 0.37$ . It can be observed that the mismatch to noise ratio from (2.36) decreases when moving to finer nodes. Hypothetically, the error introduced by parameter mismatch will remain one order of magnitude above the error caused by thermal noise for technologies down to 3 nm feature size. It should be noted, however, that the rule derived in (2.36) is only an approximation not accounting for second order effects in MOS devices.

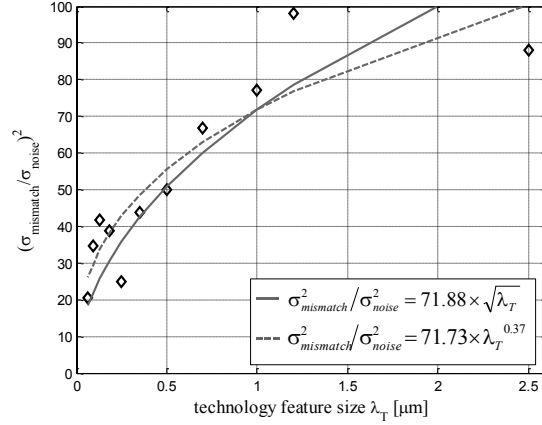


Figure 2.3. Scaling of the mismatch to noise ratio in terms of technology feature size.

## 2.7 Mismatch versus temperature

A temperature dependence of the process parameter variability is practically negligible. In particular, temperature affects the gate-semiconductor work function difference  $\Phi_{MS}$  and Fermi potential in the bulk  $\Phi_B$ , and hence, changes the threshold voltage  $V_{th}$  [Allen 2002]. The effect of temperature fluctuation can be seen as a systematic shift in the absolute values of physical parameters of semiconductor. Although  $\Phi_B$  and  $\Phi_{MS}$  linearly depend on temperature, they are proportional to the logarithm of the doping concentration  $N_D$  (e.g. substrate or gate doping) [Mead 89]. Therefore, the impact of the variability of  $\Phi_B$  and  $\Phi_{MS}$ , caused by fluctuations of  $N$ , on the threshold voltage is negligible. In the case of the current factor  $\beta$ , parameter  $A_\beta$  depends mainly on the fluctuations of carrier mobility caused by the channel edge roughness, which does not depend on temperature [Lakshmikumar 86].

Although parameter mismatch does not change with temperature, the generated thermal noise increases, reducing the mismatch to noise ratio discussed in the previous section. In theory, however, increase of the absolute temperature  $T$  by factor of 10 is necessary to equalise the effects of noise and mismatch, which is not realistic.

## 2.8 Mismatch optimisation

### 2.8.1 Circuit design techniques

One of the commonly used techniques of mismatch optimisation in CMOS circuits is scaling, based on the assumption that the variability of electrical and physical parameters of MOS transistor reduce with the gate area increase [Pelgrom 89]. Other methods,

account for circuit techniques such as auto-zero compensation, chopping, switched-current operation and trimming [Kinget 97].

Auto-zero compensation is used in comparators, A/D converters and switched capacitor circuits, mainly to eliminate DC offset errors, stemming from the mismatch of MOS transistors in the input differential pairs. Such circuits require additional calibration cycle, therefore, this technique applies mainly to discrete time circuits.

The main idea behind chopping technique is to reduce the effects of offset voltage and  $1/f$  noise in band limited analogue signal processing systems. It assumes an interchangeable signal phase inversions of  $0^\circ$  and  $180^\circ$  with frequency higher than the signal bandwidth at the input and output of the a processing block.

In discrete-time memory cells realised as switched-current circuits, the use of the same MOS transistor for both, read and write operation, practically eliminates the influence of parameter mismatch. This technique is further discussed in Chapter 3.

Post fabrication trimming is the most generic mismatch optimisation technique applicable to a variety of CMOS circuit. It is usually based on individual operating point trimming or based on design redundancy, where additional transistors or processing blocks can individually be added or excluded from the system. Despite the increased area occupation and power consumption, such circuits usually require individual post fabrication trimming step and parameter storage technique, increasing the manufacturing costs. Therefore, this approach is rarely used in practice, except very specific designs [Heijne 96].

### 2.8.2 Layout drawing techniques

An improvement in terms of parameter variability can be achieved by employing different layout drawing techniques, accounting for a symmetric component placement and common centroid design [Bastos 96]. In particular, mechanical stress caused by shallow trench isolation (STI) and metal coverage, can be reduced by using additional dummy devices separating the active circuit from trenches, and careful metal paths routing, leaving the critical transistors uncovered [Tuinhout 96, 2001]. An experimental study on the mismatch impact on the parameters of MOS transistors in different layout types was presented in [Yeh 2001]. Circuit design using self-aligned drain/source contacts of MOS devices was investigated in [Bolt 96]. It was concluded, however, that none of these techniques noticeably improves the design accuracy when compared to the typically used common-centroid approach. The effects of packaging and bonding on

circuit parameter variability were studied in [Bastos 97b]. More advanced layout drawing techniques using statistical models and numerical optimisation methods, targeting both systematic and random parameter variability reduction, have been developed mainly for capacitor arrays in modern CMOS data converters [Hsiao 2012].

## **2.9 Conclusions**

Despite a high parameter variability in deep submicron CMOS technologies and rather pessimistic conclusions concerning mismatch scaling with technology feature size, the methods and results presented in this thesis provide a positive outlook on the idea of using asynchronous and analogue circuits for specific information processing tasks. Conclusions and optimisation techniques presented in this chapter, will be discussed in the contexts of particular circuit applications presented further in this thesis.

## Chapter 3

---

# Current-mode analogue multipliers

---

### 3.1 Introduction and chapter overview

This chapter presents the idea and design of two current-mode CMOS multipliers for analogue computation, operating in a continuous-time and a discrete-time modes. Both circuits are based on the Gilbert multiplier cell, realised using MOS transistors working in the subthreshold region. The operation of both circuits and the computational errors, stemming from mismatch and the second order effects in MOS transistors, are analysed in theory and verified in simulations. A comparison to an equivalent structure, realised in digital domain, is provided for reference and estimation of the computational precision. The proposed multipliers will be of use in the hardware realisations of the sum-product algorithm for belief propagation in factor graphs discussed in Chapter 8.

### 3.2 Continuous-time Gilbert multiplier

In the following section, the operation of the continuous-time Gilbert multiplier, implemented using MOS transistors working in the subthreshold region, will be discussed. In particular, the design issues, such as limited power supply, gate leakage and some second order effects in MOS devices, mainly affecting the computational accuracy, will be further investigated based on theoretical analyses and simulations using MOS transistor models from a standard 90 nm CMOS technology.

### 3.2.1 Circuit analysis and realisation

Schematic diagram of a basic Gilbert multiplier cell (a current normaliser), realised using four MOS transistors connected in a translinear loop is shown in Figure 3.1.

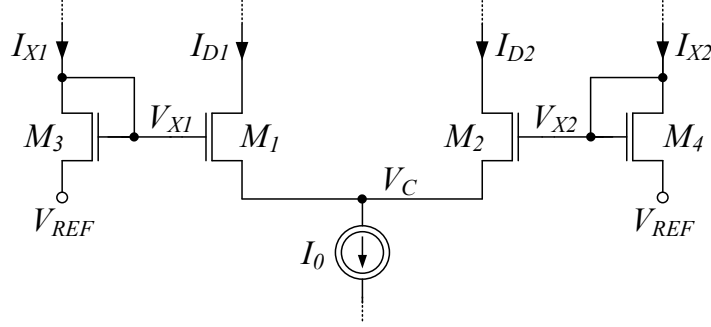


Figure 3.1. Schematic diagram of a Gilbert multiplier realised on four MOS transistors connected in a translinear loop.

In the following analysis, it is assumed that transistors  $M_1 - M_4$  have identical geometries and operate in weak inversion and saturation, where  $u_{GS} \ll V_{th}$  and  $u_{DS} \gg 4U_T \approx 100$  mV. The drain current of a MOS transistor in weak inversion can be then calculated using (equation (2.9) in Chapter 2):

$$i_D = I_{D0} \frac{W}{L} \exp\left(\frac{u_{GS}}{nU_T}\right) \quad (3.1)$$

where  $I_{D0}$  is a specific current in weak inversion,  $W$  and  $L$  are the channel width and length respectively,  $U_T = kT/q$  is thermal voltage equal approximately 25.85 mV at room temperature  $T = 300$  K, and  $n$  is a subthreshold slope factor. Gate voltages  $V_{X1}$ ,  $V_{X2}$  and the common source voltage  $V_C$  are measured in reference to the ground potential. The additional reference voltage  $V_{REF}$  was introduced to assure appropriate voltage headroom for the circuit realisation of the current source  $I_0$ . It will be shown that this voltage has no explicit impact on currents  $I_{D1}$  and  $I_{D2}$ , nevertheless, it is used later in the practical implementation, therefore, it should be accounted for in this derivation. Using equation (3.1), and assuming that parameters  $I_{D0}$ ,  $W$ ,  $L$ , and the slope factor  $n$  are the same for all four transistors in Figure 3.1, drain currents  $I_{X1}$  and  $I_{D1}$  can be calculated as:

$$I_{X1} = I_{D0} \frac{W}{L} \exp\left(\frac{V_{X1}}{nU_T}\right) \exp\left(\frac{-V_{REF}}{nU_T}\right) \quad (3.2)$$

$$I_{D1} = I_{D0} \frac{W}{L} \exp\left(\frac{V_{X1}}{nU_T}\right) \exp\left(\frac{-V_C}{nU_T}\right) \quad (3.3)$$

Using equations (3.2) and (3.3), the drain current  $I_{D1}$  and  $I_{D2}$  can be written as:

$$I_{D1} = I_{X1} \frac{\exp(-V_C/nU_T)}{\exp(-V_{REF}/nU_T)} \quad (3.4)$$

$$I_{D2} = I_{X2} \frac{\exp(-V_C/nU_T)}{\exp(-V_{REF}/nU_T)} \quad (3.5)$$

Using equations (3.4) and (3.5), and assuming that the drain currents  $I_{D1} + I_{D2} = I_0$ , the common source voltage  $V_C$  can be calculated as:

$$V_C = nU_T \ln\left(\frac{I_{X1} + I_{X2}}{I_0 \exp(-V_{REF}/nU_T)}\right) \quad (3.6)$$

Inserting (3.6) to (3.4) and (3.5), the output drain currents  $I_{D1}$  and  $I_{D2}$  are equal:

$$I_{D1} = I_0 \frac{I_{X1}}{I_{X1} + I_{X2}} \quad (3.7)$$

$$I_{D2} = I_0 \frac{I_{X2}}{I_{X1} + I_{X2}}$$

The obtained result could also be deduced from the translinear principle applied to the circuit from Figure 3.1. Assuming that transistors  $M_1 - M_4$  operate in weak inversion and saturation, and the drain current is an exponential function of the gate voltage, the diode-connected transistors  $M_3$  and  $M_4$  work as logarithmic  $I$ - $V$  converters of the input currents  $I_{X1}$  and  $I_{X2}$  generating voltages  $V_{X1} \sim \ln(I_{X1})$  and  $V_{X2} \sim \ln(I_{X2})$ , which in turn control the differential pair on  $M_1$  and  $M_2$ . Due to the exponential characteristics of  $M_1$  and  $M_2$ , the output currents  $I_{D1}$  and  $I_{D2}$  have the same proportions as the respective input ones and their sum is constant and equal  $I_0$ . This conclusion leads directly to the set of equations (3.7).

In the vector notation, the circuit from Figure 3.1 performs multiplication with normalisation given by the equation:

$$\begin{bmatrix} I_{D1} \\ I_{D2} \end{bmatrix} = \alpha \cdot I_0 \cdot \begin{bmatrix} I_{X1} \\ I_{X2} \end{bmatrix} \quad (3.8)$$

where  $\alpha = 1/(I_{X1} + I_{X2})$  is the normalising factor and  $I_0$  is the third input argument. The circuit presented in Figure 3.1 is the simplest realisation of a multiplier operating on two-

element vectors. Its extension to an  $n$ -element vector normaliser is straightforward and requires addition of more  $I$ - $V$  converters and the corresponding output transistors in the common source section. The schematic diagram of the multiplier proposed in this work, realised in a standard 90 nm CMOS technology is presented in Figure 3.2. It implements the simplest current normaliser from Figure 3.1 with additional current mirrors for input argument  $I_0$  and the output currents  $I_{D1}$  and  $I_{D2}$ . In the design, it was assumed that the input currents flow always to ground and the output currents are sourced from  $V_{DD}$ . This helps to avoid additional current mirrors in the constructions of larger systems, where several multipliers are connected together. Such structures are further discussed in Chapter 8. The details concerning the implementation of the multiplier in Figure 3.2 and various technology related issues will be explained in the next section.

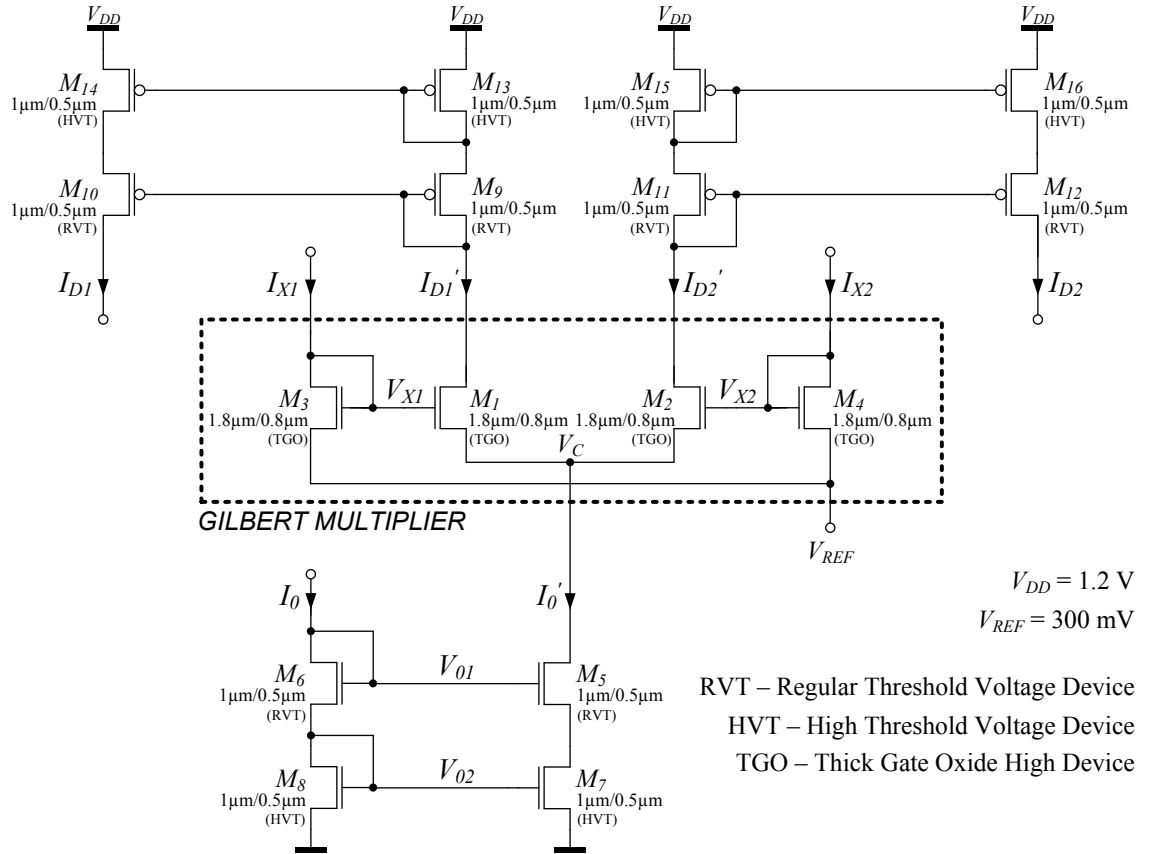


Figure 3.2. Schematic diagram of the proposed basic multiplier cell realised in a standard 90 nm CMOS technology.

### 3.2.2 Design issues

Computational precision of the analogue circuits used for arithmetic operations is usually limited by the systematic and random errors. Systematic errors can be attributed to issues such as non-ideal operation of current mirrors, variable operating point, leakage, non-ideal characteristics of MOS devices etc. Random errors result mainly from the fabrication mismatch and noise.

In order to reduce the level of systematic errors, several modifications to the basic multiplier circuit used by [Mead 89], [Loeliger 99, 2001] and [Luckenbill 2002], have been proposed. In the proposed solution, the input current  $I_0$  and the output currents  $I_{D1}$  and  $I_{D2}$  are replicated using cascode current mirrors, exhibiting higher output resistance and better linearity, but also requiring a higher voltage headroom, necessary to keep the transistors in saturation. This becomes critical especially for the tail current source, constructed on transistors  $M_5$  and  $M_7$ . Therefore, the sources of transistors  $M_3$  and  $M_4$  (the logarithmic  $I$ - $V$  converters) are connected to the reference voltage  $V_{REF} > 0$ , rather than directly to the ground potential. The reference voltage  $V_{REF}$  regulates the voltage  $V_C$ , and hence, can be used to adjust the operating point of the tail current source, preventing  $M_5$  and  $M_7$  from leaving the saturation region, especially for small input currents  $I_{X1}$  and  $I_{X2}$ . For very small input currents  $I_{X1}$  and  $I_{X2}$ , corresponding gate-source voltages of  $M_3$  and  $M_4$  are close to zero. For large tail currents, the  $V_C$  voltage is pulled down to assure gate-source voltages of  $M_1$  and  $M_2$  high enough to properly split  $I_0$  between the two branches of the differential pair. In practice,  $V_{REF}$  should be adjusted experimentally to assure proper operation of all the transistors in the circuit and minimise the generated computational errors.

The leakage currents of the MOS transistors in the current mirrors determine the minimum value of the arguments, that can be correctly replicated. In the proposed solution, to reduce the leakage below 1 nA, and assure the correct operation of the circuit for currents in range of 1 nA - 1  $\mu$ A, (necessary to encode values within range 0.1% - 100%, Chapter 8), transistors  $M_{7,8}$ ,  $M_{13,14}$ , and  $M_{15,16}$  were implemented as high threshold devices (HVT). The maximum value of the current is mainly limited by the size (width) of the transistors  $M_1$  -  $M_4$  (which operate in weak inversion for relatively low drain currents), and the size of transistors used in the current mirrors. In practice, the geometry of these transistors will be constrained by the timing requirements and the maximum circuit size. In the simulations of the multiplier presented in Figure 3.2 it was observed

that the current mirrors introduce minor computational errors (assuming no mismatch) for transistor size  $W = 1 \mu\text{m}$  and  $L = 0.5 \mu\text{m}$  or higher, in range from 1 nA to 1  $\mu\text{A}$ . The sizes of the transistors  $M_{1-4}$  are also not critical in terms of the computational precision of the multiplier, therefore, they were mainly dictated by the design of the discrete-time version of the multiplier (presented in Section 3.3), and for reference are kept the same and equal to  $W = 1.8 \mu\text{m}$  and  $L = 0.8 \mu\text{m}$ .

### 3.2.3 V-AMS MOS model

Yet another source of systematic errors results solely from the second order effects in MOS transistors  $M_1 - M_4$  (Figure 3.2), such as channel length modulation (i.e. Early effect) and the variable slope factor  $n$  [Mead 89], [Xi 2003]. This affects the symmetry of the Gilbert cell, where each output branch operates on a different gate voltages, unless both input currents  $I_{X1}$  and  $I_{X2}$  are equal. The level of errors caused by such effects was smaller for thick gate oxide devices (TGO), however this could be an issue specific to the model and technology used.

Given the structure of the BSIM model, the fundamental effects of a MOS device, such as channel length modulation, are not defined by one equation or a single parameter, that could be "activated" or "deactivated" by the user. Therefore, a simplified MOS transistor model was developed to allow for the simulations of the contribution of different effects to the total computational error. The proposed model, implemented in Verilog AMS language, is dedicated for the operation of MOS transistor in subthreshold region according to the formula [Mead 89]:

$$i_D = I_{D0} \frac{W}{L} \exp\left(\frac{u_{GS}}{nU_T}\right) \cdot \left(1 - \exp\left(-\frac{u_{DS}}{U_T}\right)\right) \cdot \left(1 + \frac{u_{DS}}{U_A}\right) + I_{OFF} \quad (3.9)$$

The equation (3.9) extends the basic formula for drain current given in (3.1) by accounting for the contribution of the drain-source voltage  $u_{DS}$ , the channel length modulation effect depending on the Early voltage  $U_A$ , and the off (leakage) current  $I_{OFF}$ . The variability of the slope factor  $n$ , in terms of the bias conditions, was first observed in the simulations of a simple test circuit for DC characteristics presented in Figure 3.3, using BSIM4 MOS transistor model provided by the foundry. Assuming the simplest relation between the drain current  $i_D$  and gate-source voltage  $u_{GS} = V_{GB} - V_{SB}$ , given by equation (3.1), the dependency of the inverted slope factor  $k = 1/n$  was evaluated from the simulations using the following relation:

$$k = U_T \frac{\partial \ln(i_D)}{\partial V_{GB}} \quad (3.10)$$

Based on the obtained results, it was observed that the traces showing the dependency of parameter  $k$  on the gate-bulk voltage  $V_{GB}$  resemble Gaussian functions, slightly elongated on the right hand side and shifted with the source-bulk voltage  $V_{SB}$ , as demonstrated in Figure 3.4. Therefore, in the constructed V-AMS MOS model, a Gaussian function was used as an approximation to reproduce the variability of the slope factor  $k$  according to the following empirical equation:

$$k = K \cdot \exp(-\alpha_k u_{GS}^2) \quad (3.11)$$

where  $K$  and  $\alpha_k$  are fitting parameters. The drain current  $i_D$  and the slope factor  $k$  of the proposed V-AMS model were calculated adapting the equations (3.9) and (3.11) and adding more fitting parameters and control flags, eventually leading to the following model equations:

$$i_D = I_{D0} \frac{W}{L} \exp\left(\frac{k(V_{GB} - V_{SB} - V_{SH})}{U_T}\right) \cdot \left(1 - \exp\left(-\frac{V_{DB} - V_{SB} - V_{SH}}{U_T}\right)\right) \cdot \left(1 + UAF \frac{V_{DB} - V_{SB} - V_{SH}}{U_A}\right) + I_{OFF} \quad (3.12)$$

$$k = (K_{MAX} - K_{MIN}) \cdot \exp(-KF \cdot \alpha_k (V_{GB} - V_{SB} - V_{SH})^2) + K_{MIN} \quad (3.13)$$

The values of the parameters used in the equations (3.12) and (3.13), are provided in Table 3.1. The values of the parameters were chosen experimentally to obtain qualitatively similar behaviour between BSM4 model, provided by the foundry, and the proposed V-AMS model. In the simulations, the circuit from Figure 3.3 was used assuming constant drain-bulk voltage  $V_{DB} = 1.2$  V and  $V_{GB}$  swept in range 0 to 1.2 V for four different values of the source-bulk voltage  $V_{SB} = 0$  V, 100 mV, 200 mV and 300 mV. The results showing drain currents  $i_D$  and slope factors  $k$ , calculated using equation (3.10), are presented in Figure 3.4. It should be noted the equation (3.11) was derived based on the assumption that MOS transistor operates according to the equation (3.1). Since BSIM4 model accounts for the presence of various second order effects, they may to some extent, contribute to the variability of the slope factor calculated using (3.10). Therefore, equation (3.11) is an approximation of the slope factor variability, suitable for the proposed model, but its use for other purposes should further be investigated.

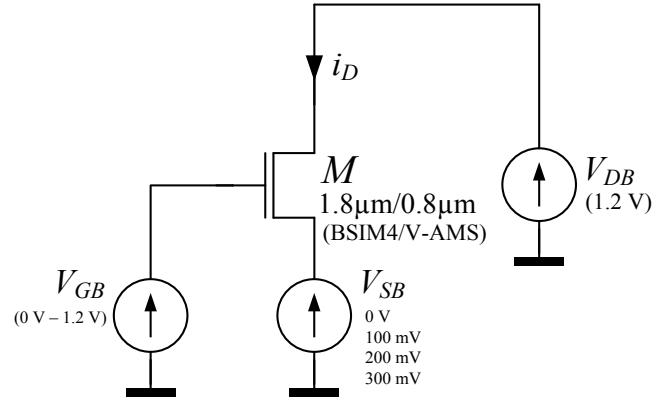


Figure 3.3. Schematic diagram of the test circuit used in V-AMS MOS transistor model verification.

Based on the simulation results presented in Figure 3.4, it can be seen that the proposed V-AMS model, constructed for the thick gate oxide transistor of a particular geometry, remains in a good qualitative agreement with its BSIM4 counterpart provided by the foundry. A deviation of the simulated drain current above the off (leakage) floor in the weak inversion region stems from second order effects such as Drain Induced Barrier Lowering (DIBL), accounted for in the BSIM4 model but not included in the proposed V-AMS model. Also, a distinct inflexion region of the slope factor profile around the threshold voltage is not predicted by the V-AMS model, which assumes a smooth transition of  $k$  from moderate to strong inversion.

A better fit of the proposed model could possibly be achieved by employing numerical optimisation, or by extending the proposed model to more elaborate form. Nevertheless, for the purpose of this research, the proposed model is sufficient in its current form, providing good qualitative description of MOS transistor behaviour. It is applicable to the subthreshold region and allows for "controlling" the channel length modulation, off leakage and variable slope factor, simply by setting the corresponding flags (see Table 3.1). Simulation results showing the impact of the systematic and random effects of MOS transistors on the computational error of analogue multipliers are presented and discussed in Section 3.2.5.

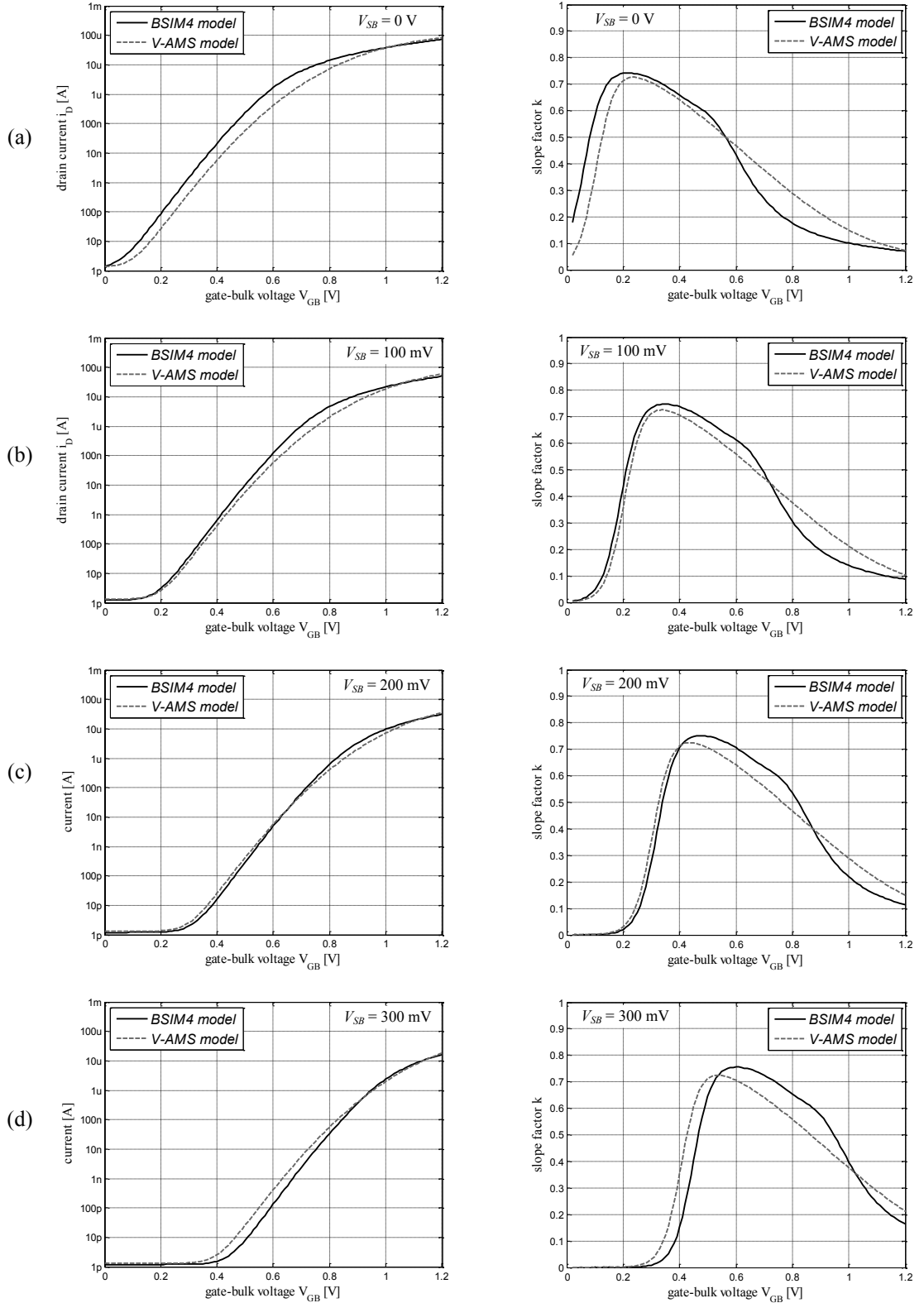


Figure 3.4. Drain current and slope factor  $k$  simulated for the thick gate oxide transistor using BSIM4 model (provided by the foundry) and the proposed V-AMS model versus gate-bulk voltage  $V_{GB}$  for  $V_{DB} = 1.2$  V and different values of the source-bulk voltage equal: a)  $V_{SB} = 0$  V, b)  $V_{SB} = 100$  mV, c)  $V_{SB} = 200$  mV, and d)  $V_{SB} = 300$  mV.

Table 3.1. Parameters of the V-AMS MOS transistor model.

<i>Parameter</i>	<i>Value</i>	<i>Unit</i>	<i>Remarks</i>
$W$	1.8	$\mu\text{m}$	MOS transistor channel width
$L$	0.5	$\mu\text{m}$	MOS transistor channel length
$V_{GB}$	0 - 1.2	V	Gate-bulk voltage
$V_{DB}$	0 - 1.2	V	Drain-bulk voltage
$V_{SB}$	0 - 1.2	V	Source-bulk voltage
$I_{D0}$	0.15	pA	Specific current of MOS transistor in subthreshold
$U_T$	25.85	mV	Thermal potential equal $kT/q$ for $T = 300$ K (constant)
$U_A$	8	V	Early voltage
$I_{OFF}$	1.3	pA	Off (leakage) current
$K_{MAX}$	0.78	---	Maximum slope factor value (also the value of $k$ when $KF = 0$ )
$K_{MIN}$	0.28	---	Minimum slope factor
$\alpha_K$	0.90	---	Fitting parameter of the slope factor $k$
$V_{SH}$	60	mV	Source voltage shift (fitting parameter)
$UAF, KF$	0/1	---	Flags used to "enable" or "disable" channel length modulation and variable slope factor effect (for $KF=0$ the slope factor is constant and equals $K_{MAX}$ )

### 3.2.4 Computational errors

In the simulations, two definitions of the computational error are considered: the *relative current error* (RCE), and the *normalised current error* (NCE). The relative current error (RCE) is defined as the maximum difference between the vector of currents obtained from the circuit simulation  $I_{SIM}$  (in the case of the multiplier from Figure 3.2, the elements of  $I_{SIM}$  are the drain currents  $I_{D1}$  and  $I_{D2}$ ) and the vector  $I_{MAT}$  (evaluated using equation (3.8)), with respect to the reference current  $I_{REF}$ , defining the maximum value of the computed signals (i.e.  $I_{REF}$  is equivalent to unity in the performed arithmetic operations):

$$\text{RCE}[\%] = \frac{\max |I_{SIM} - I_{MAT}| [\mu\text{A}]}{I_{REF} [\mu\text{A}]} \cdot 100\% \quad (3.14)$$

The normalised current error (NCE) is defined as the maximum difference between the normalised current  $I_{SIM}/\|I_{SIM}\|$ , obtained from the simulation, and the normalised result  $I_{MAT}/\|I_{MAT}\|$ , obtained from equation (3.8), where  $\|X\|$  equals the sum of the elements in  $X$ :

$$\text{NCE}[\%] = \max \left| \frac{I_{SIM}}{\|I_{SIM}\|} - \frac{I_{MAT}}{\|I_{MAT}\|} \right| \cdot 100\% \quad (3.15)$$

The first definition of the computational error (RCE) represents a disparity between the current obtained from the circuit simulation and the exact one, computed from the equation (7.8). The second definition of the computational error (NCE) was introduced to address applications where the information is represented by the ratios of the vector

elements, rather than by their absolute values. This definition will be used in the calculations of the computational errors generated by the circuit realisations of the sum-product algorithm for belief propagation in Chapter 8.

### 3.2.5 Simulation results

In the simulations of the continuous-time multiplier, presented in Figure 3.2, the input currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  were provided directly from ideal current sources. The output currents  $I_{D1}$  and  $I_{D2}$  were sunk using additional diode connected transistors of the same size and type as  $M_1 - M_4$ , with sources connected to the reference voltage  $V_{REF}$ . The simulation results were obtained using MOS transistor models provided by the foundry for a typical process corner ( $TT$ ) and the V-AMS model discussed in section 3.2.3. The simulations were performed using a predefined set of 5000 random pairs of input currents  $[I_{X1} I_{X2}]$ , generated in range 1 nA - 1  $\mu$ A, and divided into ten subsets, each with different constant current  $I_0$ , in range from 50 nA to 1  $\mu$ A.

The simulation results showing mean values of the computational errors RCE and NCE (section 3.2.4), in terms of the third input current  $I_0$ , generated using BSIM4 and V-AMS models applied to transistors  $M_1 - M_4$ , are presented in Figure 3.5. In particular, V-AMS model was used with different settings of flags  $KF$  and  $UAF$  covering four cases: 1) both the variable slope factor ( $KF = 1$ ) and the channel length modulation effects ( $UAF = 1$ ) are "activated", 2) only variable slope factor ( $KF = 1$ ,  $UAF = 0$ ) is accounted for, 3) only channel length modulation ( $KF = 0$ ,  $UAF = 1$ ) is considered, and 4) MOS transistor with pure exponential characteristics are used ( $KF = 0$ ,  $UAF = 0$ ).

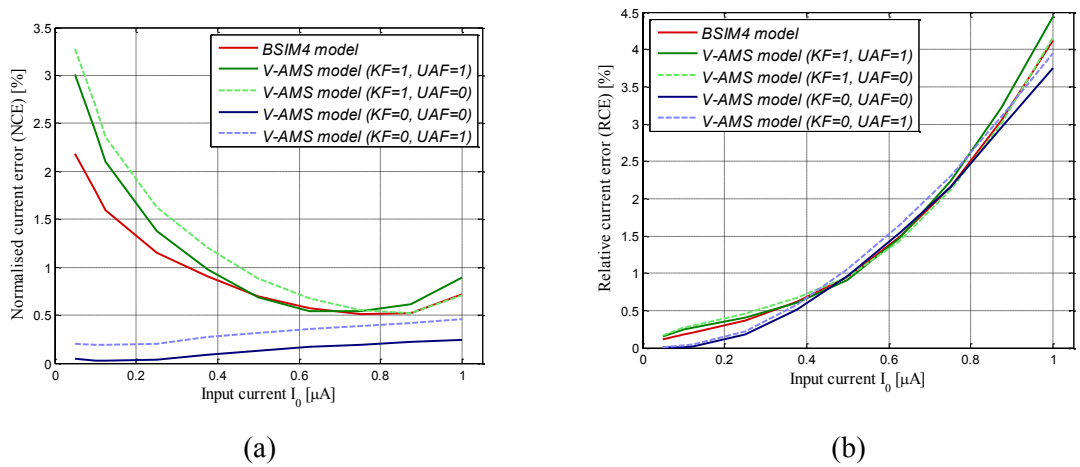


Figure 3.5. Simulations of the test circuit from Figure 3.3 using BSIM4 and V-AMS models showing the mean value of the computational errors vs. input current  $I_0$ : a) NCE, and b) RCE.

It can be observed that the impact of the variable slope factor in transistors  $M_1 - M_4$  on the computational accuracy (green dashed trace in Figure 3.5a) is much higher than the impact of the channel length modulation (blue dashed trace in Figure 3.5a). It can be observed that traces (dashed green and continuous green), that were generated assuming only the variable slope factor (dashed green trace) and assuming both the variable slope factor and the channel length modulation (continuous green trace), are close to each other. Therefore, it can be concluded that the slope factor variability dominates the precision of the multiplier. Slightly higher values of NCE, obtained for only slope factor variability (dashed green trace), result from the fact that this effect may, to some extent, be compensated by the Early effect, when both are accounted for in the model. It can also be observed that the NCE is the lowest when MOS transistors with pure exponential characteristics are used (continuous blue trace). A non-zero level of the NCE error (continuous blue trace) stems mainly from a non-ideal operation of the current mirrors built on pMOS transistors, since their performance degrades for lower and higher output currents due to leakage and nonlinearities such as Early effect.

The relative current error (RCE) provides more generic measure of the circuit precision, also accounting for the non-ideal behaviour of the tail current mirror (Figure 3.2). In particular, for very low input currents  $I_{X1}$  and  $I_{X2}$ , the corresponding values of voltages  $V_{X1}$  and  $V_{X2}$  are also low, and the common voltage  $V_C$  will decrease to enable proper splitting of the tail current  $I_0$ . If  $I_0$  is high, then  $V_C$  drops further down to increase the gate-source voltages of  $M_1$  and  $M_2$ . This, on the other hand, reduces the headroom for transistors  $M_5$  and  $M_7$ , operating in saturation (Figure 3.2), and hence, affects the bottom current mirror reducing the replicated value of  $I_0'$ . As a result, the RCE error reaches its maximum for the larger values of  $I_0$ , indicating a dominant impact of the tail current on the generated computational errors. In the definition of the RCE error, the division by the fixed reference current  $I_{REF}$  (representing the maximum argument value) is assumed, therefore, the computational errors evaluated for  $I_0 < I_{REF}$  will be reduced (Figure 3.5b). It should be noted that the normalisation of the output vector  $[I_{D1} I_{D2}]$ , done in the computation of the NCE error, does not account for the inaccuracy of the tail current mirror.

Finally, it can be observed that the traces showing the computational errors generated using BSIM4 and V-AMS models (for  $KF = 1$  and  $UAF = 1$ ), remain in qualitative agreement, which confirms the validity of the proposed model.

Histograms showing distribution of the computational errors NCE and RCE of the continuous-time multiplier from Figure 3.2, are presented in Figure 3.6. Histograms of the NCE and RCE error distribution, generated accounting for parameter mismatch in all MOS transistors in the circuit from Figure 3.2 are presented in Figure 3.7. The distribution of RCE error is represented on a log scale due to a long tail of very rare cases with higher error. The mean value of the computational errors NCE and RCE, in terms of the transistor size scaling, is presented in Figure 3.8. In the simulations, widths and lengths of all the transistors in the circuit from Figure 3.2 were multiplied by the scaling factor  $\alpha$  in range from 1 to 20.

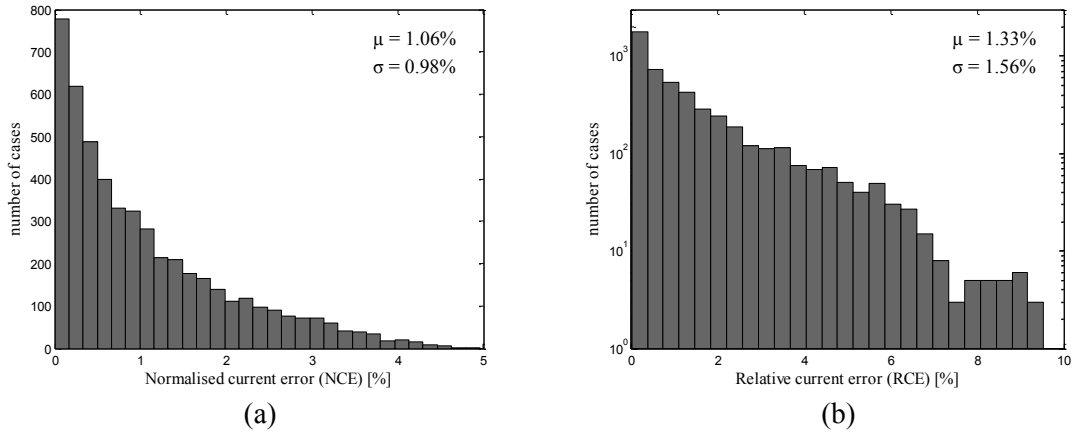


Figure 3.6. Histograms of the computational errors of the continuous-time multiplier from Figure 3.2: a) NCE ( $\mu = 1.06\%$ ,  $\sigma = 0.98\%$ ), and b) RCE ( $\mu = 1.33\%$ ,  $\sigma = 1.56\%$ ).

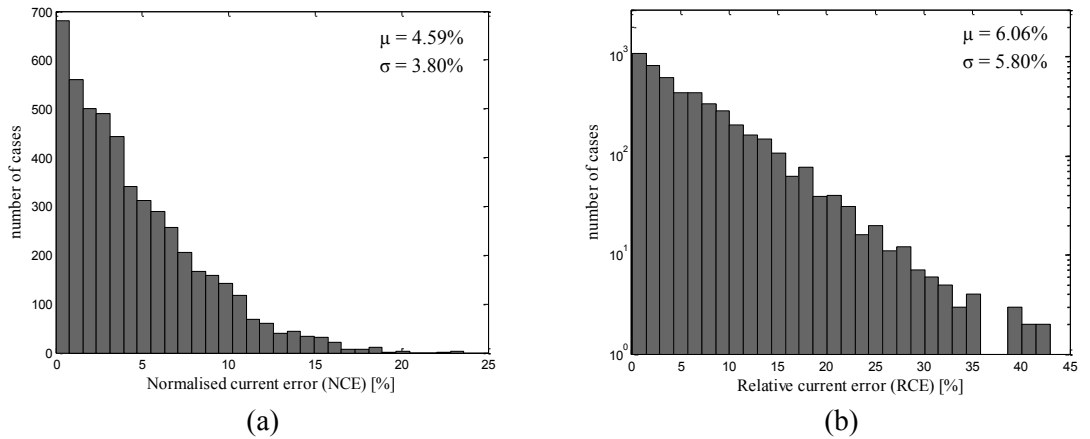


Figure 3.7. Histograms of the computational errors of the continuous-time multiplier from Figure 3.2 accounting for fabrication mismatch of MOS transistors: NCE ( $\mu = 4.59\%$ ,  $\sigma = 3.80\%$ ) and b) RCE ( $\mu = 6.06\%$ ,  $\sigma = 5.80\%$ ).

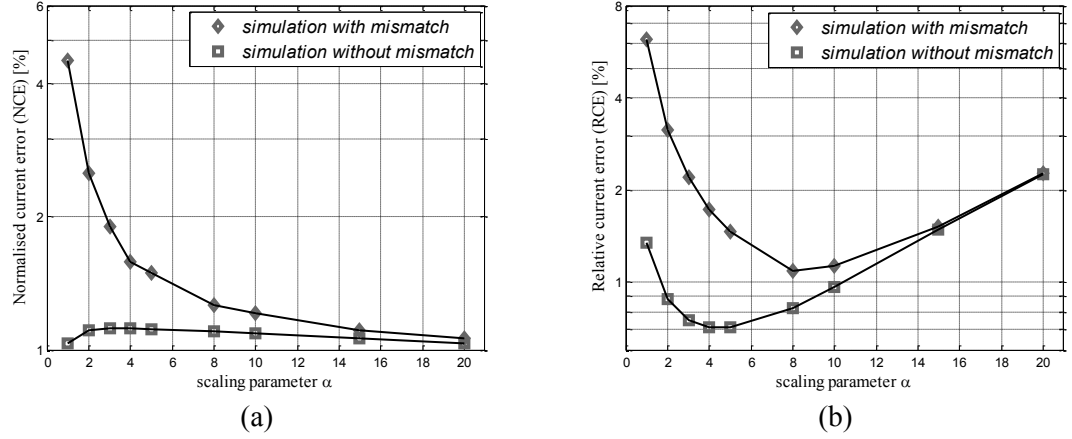


Figure 3.8. a) Mean value of the NCE, and b) RCE computational errors versus transistor size scaling factor  $\alpha$  (the corresponding gate area is proportional to  $\alpha^2$ ).

It can be observed that mismatch strongly affects the correct operation of the circuit reducing its accuracy. Transistor size scaling improves the precision of the multiplier, however, at the expense of the device area. In particular, the impact of the fabrication mismatch becomes comparable to the level of the systematic errors for the values of parameter  $\alpha$  higher than 8. This means that the total area of the multiplier ought to be increased more than 64 times to notably reduce mismatch. In practice, the magnitude of the generated error in such multiplier may be too high for many applications requiring precise computation under strict area constraints.

### 3.3 Discrete-time Gilbert multiplier

#### 3.3.1 Circuit analysis and realisation

The basic Gilbert multiplier cell, presented in Figure 3.1, can also be seen as a combination of two structures resembling current mirrors, built on transistor pairs  $M_1$ - $M_3$  and  $M_2$ - $M_4$ . It is important to note, however, that these pairs operate with different source potentials, therefore, the corresponding gate-source voltages are different and the input currents  $I_{X1}$  and  $I_{X2}$  do not simply copy to the output. This similarity can be advantageous in terms of the discrete-time realisation of the circuit, where the transistor pairs can be replaced with dynamic current mirrors operating in the switched-current mode. The schematic diagram of a continuous-time current mirror and its dynamic, discrete-time equivalent, are presented in Figure 3.9. [Toumazou 93b].

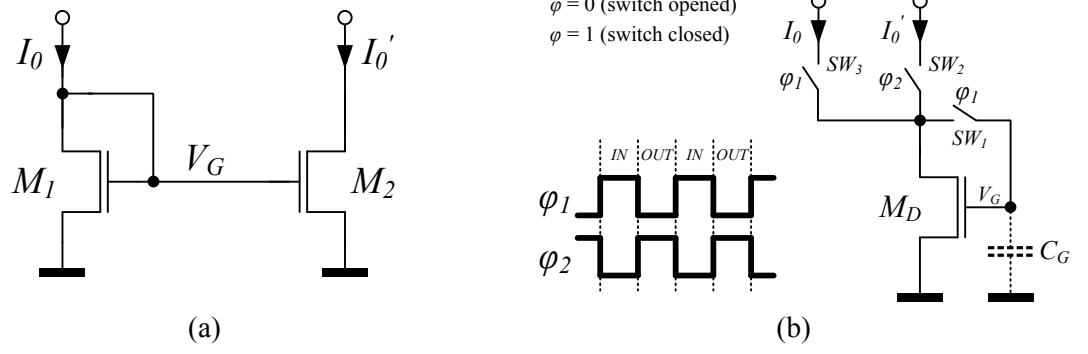


Figure 3.9. Schematic diagram of a current mirror realised as a) continuous-time and b) discrete-time circuit.

The continuous-time current mirror, presented in Figure 3.9a, consists of two transistors: the diode-connected  $M_1$ , converting the input current  $I_0$  to the gate voltage  $V_G$ , and  $M_2$ , controlled by  $V_G$  and generating the output current  $I_0'$ . Ideally, for the identical transistors  $M_1$  and  $M_2$ , the output current  $I_0' = I_0$  (not accounting for the second order effects in MOS transistors). The discrete time realisation, presented in Figure 3.9b, works in two phases, first it reads the input current  $I_0$  (*IN* phase) and then it generates the output current  $I_0'$  (*OUT* phase). In the following analysis, ideal switches controlled by signals  $\phi_1$  and  $\phi_2$  and no charge injection nor gate current leakage are assumed. When  $\phi_1 = 1$  and  $\phi_2 = 0$ , the circuit is in the current reading mode (*IN* phase), switch  $SW_2$  is opened and  $SW_1$  and  $SW_3$  are closed. The input current  $I_0$  flows through the diode-connected transistor  $M_D$  to the ground, charging the gate capacitance  $C_G$  to a particular voltage  $V_G$ , corresponding to the input current  $I_0$ . In this phase, transistor  $M_D$  operates similarly to  $M_1$  in the continuous-time realisation in Figure 3.9a. When  $\phi_1 = 0$  and  $\phi_2 = 1$ , the circuit generates the output current (*OUT* phase), switch  $SW_2$  is closed and  $SW_1$  and  $SW_3$  are opened, and the output current  $I_0'$  is controlled by the voltage  $V_G$  of the floating gate of  $M_D$ , storing some electric charge on the gate capacitance  $C_G$ . In this phase, transistor  $M_D$  operates similarly to  $M_2$  in the continuous-time realisation in Figure 3.9a.

It can be observed that the structure and the operation of the dynamic current mirror is identical to the analogue switched-current memory cell. The capacitance  $C_G$  is usually an inherent gate capacitance of a MOS transistor. Due to the gate leakage, charge injection from the switch  $SW_1$ , and parasitic coupling between gate and drain/source regions, external gate capacitors, may be needed in practice. The fundamental advantage of the dynamic current mirror is the use of the same transistor  $M_D$  in both phases (*IN* and *OUT*), which makes the process of replicating currents indifferent to the parameters of the circuit, and hence, highly immune to the fabrication mismatch. It is important to note,

however, that the switched current approach exhibits a range of design challenges related to charge injection from switches, gate leakage, and second order effects of MOS transistors (e.g. channel length modulation), which usually degrade the precision of such circuits. More detailed analysis and characterisation of the effects and their optimisation techniques have been a subject of a broad study and can be found in textbook positions such as [Toumazou 93a, 93b].

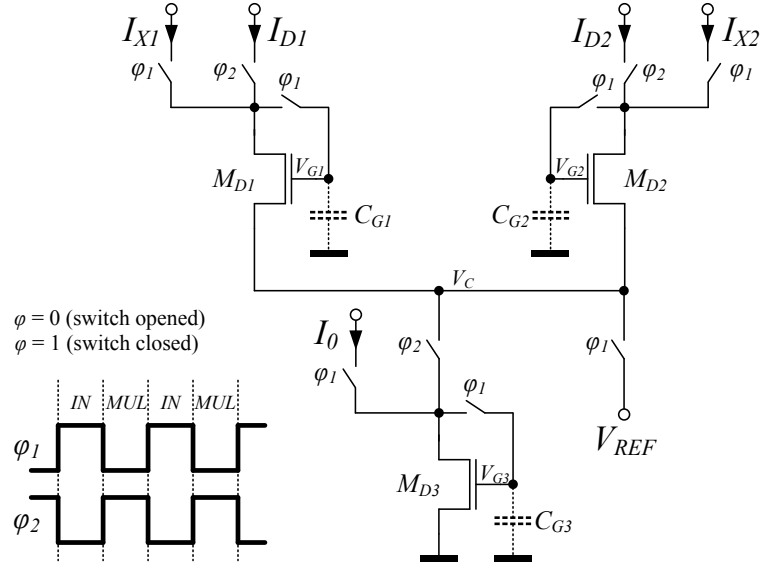


Figure 3.10. Schematic diagram of the discrete-time Gilbert multiplier.

The schematic diagram of the discrete-time realisation of the Gilbert multiplier is shown in Figure 3.10. The proposed circuit consists of three memory cells built on transistors  $M_{D1}$ ,  $M_{D2}$  and  $M_{D3}$ , and operates in two phases: reading the input currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  (*IN* phase), and performing multiplication (*MUL* phase), generating the output currents  $I_{D1}$  and  $I_{D2}$ . The additional cell with transistor  $M_{D3}$  is a replacement for the ideal current mirror  $I_0$  from Figure 3.1. When  $\phi_1 = 1$  and  $\phi_2 = 0$  (*IN* phase), the input currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  charge the gate capacitances  $C_{G1}$ ,  $C_{G2}$  and  $C_{G3}$  of the respective transistors  $M_{D1}$ ,  $M_{D2}$  and  $M_{D3}$  to voltages  $V_{G1}$ ,  $V_{G2}$  and  $V_{G3}$ . In particular, in this phase the transistors  $M_{D1}$  and  $M_{D2}$  operate as logarithmic  $I$ - $V$  converters of the input currents  $I_{X1}$  and  $I_{X2}$ , similarly to transistors  $M_3$  and  $M_4$  in Figure 3.1. The sources of transistors  $M_{D1}$  and  $M_{D2}$  are connected to the bias voltage  $V_{REF}$  rather than directly to the ground to assure proper voltage headroom for the transistor  $M_{D3}$  during the *MUL* phase. When  $\phi_1 = 0$  and  $\phi_2 = 1$ , the circuit is in the multiplying mode (*MUL* phase) and transistors  $M_{D1}$  and  $M_{D2}$  are connected to the drain of  $M_{D3}$  forming a differential pair, similar to the one built on

transistors  $M_1$  and  $M_2$  in Figure 3.1. In this phase, the common source voltage  $V_C$  will increase or decrease from  $V_{REF}$ , depending on the relation between currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  (see discussion in section 3.2.2). Assuming constant gate voltages  $V_{G1}$  and  $V_{G2}$ , the corresponding gate source voltages of  $M_{D1}$  and  $M_{D2}$  will change to split the current  $I_0$  into two branches of the differential pair, generating the output currents  $I_{D1}$  and  $I_{D2}$ , proportional to the input currents  $I_{X1}$  and  $I_{X2}$ . The proposed circuit operates according to the same principles as the continuous-time Gilbert multiplier, utilising the exponential characteristic of MOS transistors in weak inversion. Therefore, the output currents can also be calculated using equation (3.8). The only difference is the decomposition of the multiplication process into two independent phases, where the input currents are first converted to their logarithmic voltage representations, and then, these voltages drive the differential pair. The proof-of-concept implementation of the discrete-time multiplier realised in a standard 90 nm CMOS technology is presented in Figure 3.11.

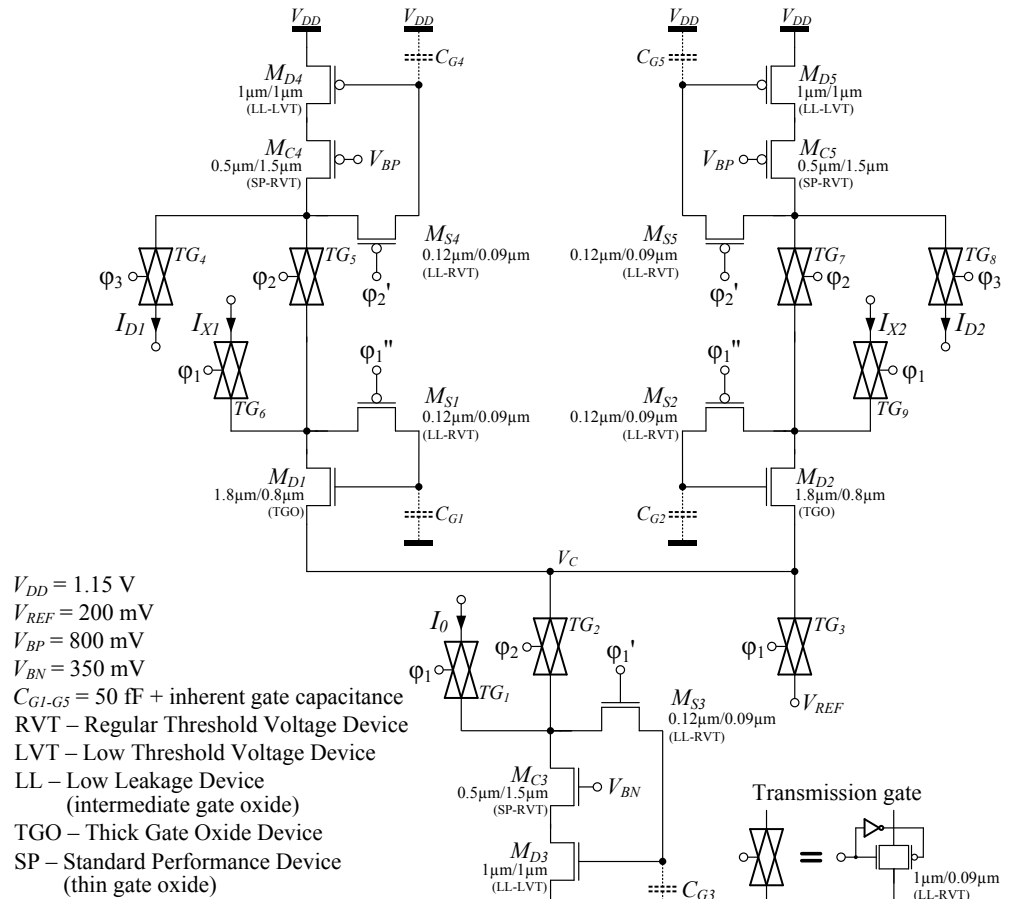


Figure 3.11. Schematic diagram of the discrete-time multiplier realised in a standard 90 nm CMOS technology.

The circuit consists of five memory cells built on transistors  $M_{D1} - M_{D5}$ . Their function is the same as in the simplified realisations in Fig. 3.10. The additional two cells, built on transistors  $M_{D4}$  and  $M_{D5}$ , are used to store and output the computed results. The circuit operates in three phases *IN*, *MUL* and *OUT* using control signals  $\phi_1$ ,  $\phi_2$  and  $\phi_3$  turning on or off the transmission gates to obtain particular circuit configurations. The detailed timing diagram of the control signals used in the circuit is presented in Figure 3.12.

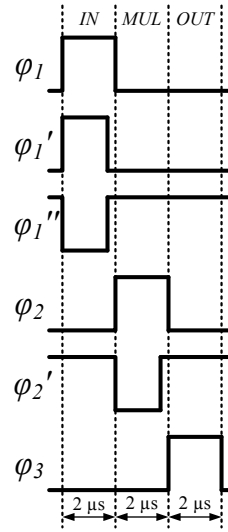


Figure 3.12. Timing diagram of a single *IN-MUL-OUT* sequence of the discrete-time multiplier.

In the first phase (*IN*), denoted by the control signals  $\phi_1 = 1$ ,  $\phi_2 = 0$  and  $\phi_3 = 0$ , the transmission gates  $TG_1$ ,  $TG_6$  and  $TG_9$  are turned on providing the input currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  to the memory cells  $M_{D1}$ ,  $M_{D2}$  and  $M_{D3}$ . The additional signals  $\phi_1'$  and  $\phi_1''$  turn on the switches built on transistors  $M_{S1}$ ,  $M_{S2}$  and  $M_{S3}$ , setting a diode-connected configuration of these memory cells. In order to reduce the effects of charge injection, these switches turn off before the transition from *IN* to *MUL* phase. Also, the transmission gate  $TG_3$  connects the sources of  $M_{D1}$  and  $M_{D2}$  to the bias voltage  $V_{REF} = 200$  mV, assuring proper operation of the transistor  $M_{D3}$  during *MUL* phase. In the second phase (*MUL*), denoted by the signals  $\phi_1 = 0$ ,  $\phi_2 = 1$  and  $\phi_3 = 0$ , the gates  $TG_2$ ,  $TG_5$  and  $TG_7$  turn on, reconfiguring the circuit such that the transistor  $M_{D3}$  works as the tail current source for the differential pair built on  $M_{D1}$  and  $M_{D2}$  splitting the tail current  $I_0$  into the two branches according to the values of previously read  $I_{X1}$  and  $I_{X2}$ . These currents flow through the diode connected transistors  $M_{D4}$  and  $M_{D5}$  as long as the control signal  $\phi_2'$  keeps transistors  $M_{S4}$  and  $M_{S5}$  turned on. Similarly as before, the control signal

$\phi_2'$  turns  $M_{S4}$  and  $M_{S5}$  off before the transition from *MUL* to *OUT* phase. In the last phase (*OUT*), denoted by the control signals  $\phi_1 = 0$ ,  $\phi_2 = 0$  and  $\phi_3 = 1$ , only the output gates  $TG_4$  and  $TG_8$  turn on, and the transistors  $M_{D4}$  and  $M_{D5}$  work as current sources supplying the computed results. It is important to note that the circuit configurations in phases *IN* and *OUT* are independent and can be evaluated at the same time. This is advantageous in larger computing systems, where sequential operations can be pipelined, requiring only two cycles per single multiplication. The duration of each phase (the cycle time) was set to  $T_C = 2 \mu s$  (Figure 3.12) based on the assumption that the 10 nA current (representing value 0.01 assuming  $I_{REF} = 1 \mu A$ ) charges the capacitance  $C = 50$  fF for  $\Delta V = 0.4$  V in 2  $\mu s$ . In practice  $\Delta V$  may be smaller than 0.4 V due to the previously stored charge.

### 3.3.2 Design issues

There are many challenges in the sampled-current circuit design, mainly related to the fact that the information, at some point, is represented by an electric charge stored on the gate of a MOS transistor. Effects, such as charge injection and gate leakage, can significantly affect this charge and change or destroy the information stored. Therefore, techniques aiming at the reduction of these phenomena are usually employed in the design of such circuits. In the proposed implementation, the charge injection effect was reduced by using the transistors  $M_{S1} - M_{S5}$  implemented as low leakage devices (with intermediate oxide thickness) and of the smallest size. Devices with higher oxide thickness exhibit lower inherent gate capacitance, and hence, less coupling between the gate and the source/drain regions. Also, the off current of such transistors is lower due to the higher threshold voltage. The types of particular transistors were chosen based on the range of the operating voltages of the gate and drain of the corresponding information storing transistor  $M_{D1} - M_{D5}$ . Therefore, only transistor  $M_{S3}$  is nMOS, whereas the remaining ones are of pMOS type. Transistors  $M_{D1}$  and  $M_{D2}$  are thick gate oxide devices with a high threshold voltage. This, together with the bias voltage  $V_{REF}$ , shifts the operation range of  $M_{S1}$  and  $M_{S2}$  closer to  $V_{DD}$ , which makes the *p*-type switch a better choice (this was verified in simulations). Thick gate oxide transistors were used to reduce the computational error of the multiplier caused by the slope factor variation and the gate-drain and gate-source capacitive coupling which results in an additional charge injection to the gates of  $M_{D1}$  and  $M_{D2}$ . The downside of the thick gate oxide devices is the lower inherent gate capacitance, which makes the use of some additional capacitors necessary to reduce the effects of charge injection. Therefore, the capacitances  $C_{G1} - C_{G5}$ ,

including the inherent gate capacitances of MOS transistors, were increased. Such additional capacitance can improve the precision of a memory cell, however, at the expense of the longer programming time and additional area (or volume) occupation. The efficiency of the proposed solution is technology dependent, and for standard CMOS technologies, such additional capacitance can be obtained using MIM (Metal-Insulator-Metal) and MOM (Metal-On-Metal) capacitors.

Another critical design issue of the analogue memories is charge leakage caused by the quantum effects observed between the gate and the channel (i.e. hot electron injection and tunnelling), and also caused by the subthreshold conductivity of the switching transistor connecting the gate to the drain. In order to reduce these effects, transistors  $M_{D3}$ ,  $M_{D4}$  and  $M_{D5}$  were implemented as intermediate gate oxide devices to reduce the gate tunnelling current and meet particular voltage headroom requirements.

It is important to note that the level of systematic disparity, observed between the written and read currents results also from the high output conductance of the MOS transistors (Early effect), which affects the output (read) current when the drain voltage changes. When writing to a cell, the drain of the diode-connected nMOS transistor is on the same potential as the gate. Assuming that the source is on the ground potential and the gate voltage is around the threshold voltage, the drain voltage (equal to the gate voltage) will usually be closer to zero than  $V_{DD}$ . When reading from the cell, the information storing transistor works as a current source and the drain voltage will depend on the circuit reading this current. If the current is read by another memory cell, which is usually the case, the diode connected pMOS transistor of that cell will pull up the drain voltage closer to  $V_{DD}$ . The observed drain voltage variation causes additional drain current variation, depending on the output conductance of the information storing transistor. In order to reduce this current variability, in the proposed circuit, the additional cascode transistors  $M_{C3}$  -  $M_{C5}$ , operating with a fixed bias voltages  $V_{BN}$  and  $V_{BP}$ , were used. This problem could also be solved by using the  $S^2I$  memory cells, consisting of two charge storing transistors, one for the information and one for the correction compensating for the drain voltage variation [Manganaro 98], or  $S^3I$  memory cells, dedicated for high precision applications [Carmona-Galan 2003]. Such circuits, however, occupy more area and require more complex control sequence, therefore, in the proposed proof-of-concept design the simplest solution has been chosen.

### 3.3.3 Simulation results

In the simulations of the discrete-time multiplier presented in Figure 3.11, the input currents  $I_{X1}$ ,  $I_{X2}$  and  $I_0$  were generated using additional cascode current mirrors built on pMOS transistors with regular threshold voltage (SP-RVT) of the size  $W = 4 \mu\text{m}$  and  $L = 0.5 \mu\text{m}$  and supplied from voltage  $V_{DDM} = 1.3 \text{ V}$ . The use of the circuit current mirrors was necessary since the ideal sources, used directly to provide input currents to the multiplier, may cause convergence problems in the simulations when the corresponding transmission gates are disconnected. In order to avoid additional errors caused by these auxiliary current mirrors (in practice very low), in the mathematical calculations, the input currents flowing directly to the multiplier in Figure 3.11 were used. The output currents  $I_{D1}$  and  $I_{D2}$  were sunk through additional diode-connected SP-RVT nMOS transistors of the size  $W = 2 \mu\text{m}$ ,  $L = 80 \text{ nm}$ . The simulation results were generated using MOS transistor models provided by the foundry for the typical process corner (*TT*). For inputs, a set of 5000 random pairs  $[I_{X1} I_{X2}]$  in range  $1 \text{ nA} - 1 \mu\text{A}$  and  $I_0$  in range  $50 \text{ nA} - 1 \mu\text{A}$ , the same as in section 3.2.5, was used.

The histograms showing the distribution of the computational errors NCE and RCE, calculated for the proposed discrete-time multiplier circuit, are presented in Figure 3.13. The histograms of the error distribution, accounting for the fabrication mismatch in the MOS transistors in the multiplier, except for the auxiliary current mirrors, are presented in Figure 3.14. The presented results were obtained based on the simulations of two full cycles (*IN-MUL-OUT*) of the discrete time multiplier to prevent additional errors caused by the limited time charging the gate capacitances for very small input currents.

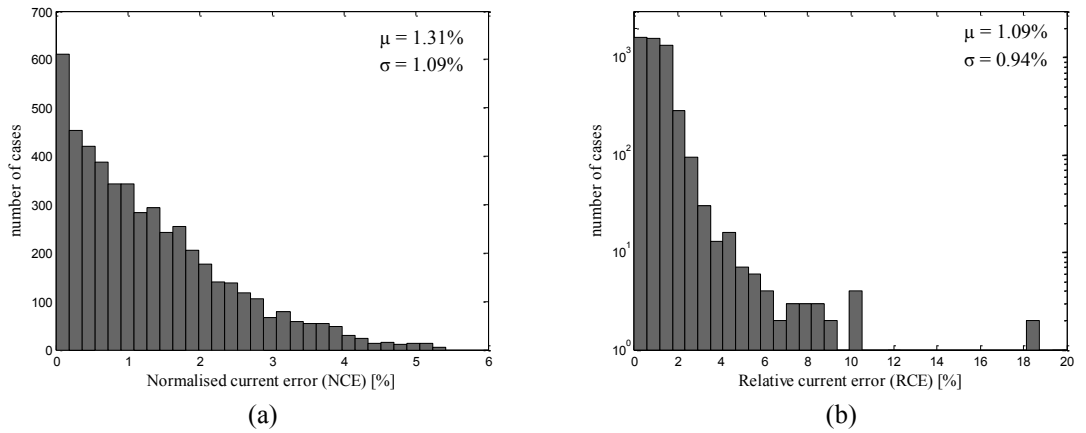


Figure 3.13. Histograms of the computational error of the discrete-time multiplier from Figure 3.11: a) NCE ( $\mu = 1.31\%$ ,  $\sigma = 1.09\%$ ), and b) RCE ( $\mu = 1.09\%$ ,  $\sigma = 0.94\%$ ).

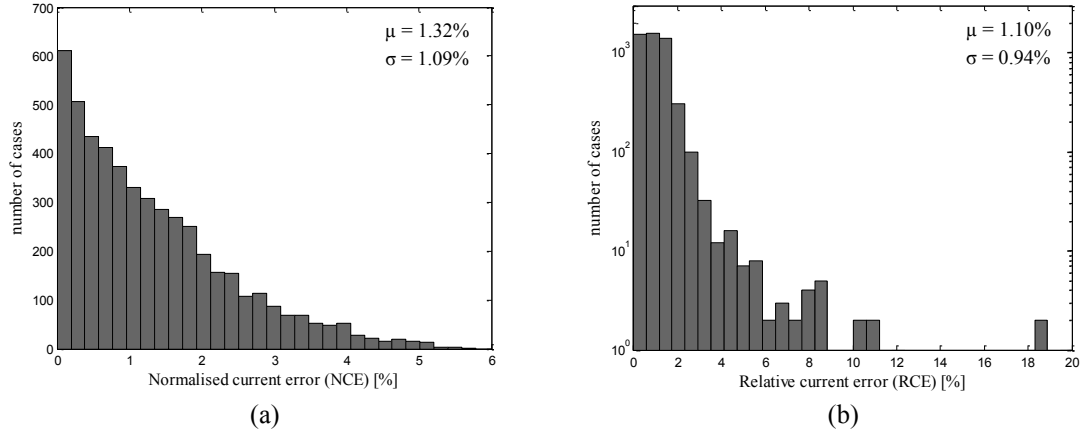


Figure 3.14. Histograms of the computational error of the discrete-time multiplier from Figure 3.11, accounting for mismatch in MOS transistors: a) NCE ( $\mu = 1.32\%$ ,  $\sigma = 1.09\%$ ), and b) RCE ( $\mu = 1.10\%$ ,  $\sigma = 0.94\%$ ).

It can be observed that the normalised current error (NCE) of the discrete-time multiplier is below 6% with only small amount of samples crossing the level of 5% (Figure 3.14a). The mean value of the NCE error of the discrete-time multiplier is equal to 1.32%, and is higher than the respective mean value of NCE error of the continuous-time realisation, equal 1.06% (Figure 3.6a). The observed increase of the computational errors result mainly from the charge injection effects. The discrete-time implementation does not deal well with certain cases generating high RCE error reaching up to 20% (Figure 3.14b). This, however, is a very rare case, which happens when the voltage  $V_C$  drops down to several tens of millivolts, as a result of very low input currents  $I_{X1}$  and  $I_{X2}$ , and very high  $I_0$ . On average, the discrete time realisation generates lower RCE (1.09%) than its continuous- time counterpart (1.33%).

The main advantage of the discrete-time multiplier is the high immunity to the parameter variability. When mismatch modelling is accounted for, the mean value of the NCE error of this circuit increases by only a fraction (from 1.31% to 1.32%) whereas for the continuous-time implementation this error increases over four times (from 1.06% to 4.59%). Also, the mean value of the RCE error of the discrete-time circuit increases only from 1.09% to 1.10%, but for the continuous-time circuit, it increases from 1.33% to 6.06%. It can be concluded that the computational precision of the proposed discrete-time multiplier is comparable with its continuous-time equivalent but does not degrade under fabrication mismatch. The additional source of error results from the effects degrading the performance of the memory cells and could be further reduced by employing more robust and better optimised circuits.

The influence of the external gate capacitances  $C_{G1} - C_{G5} = C_G$  on the computational errors NCE and RCE, is presented in Figure 3.15. The obtained results were generated in the simulations of a single *IN-MUL-OUT* sequence using the same set of 5000 input vectors and the values of the capacitances  $C_{G1} - C_{G5} = C_G$ , and the cycle time  $T_C = 2 \mu\text{s}$ . It can be observed that the additional capacitance  $C_G$  improves the precision of the multiplier, reducing the effects of charge within range. The increase of  $C_G$  over 10 fF, still reduces the relative current error (RCE) but does not reduce the normalised current error (NCE). This results from the definitions of these errors (see section 3.2.4). In the definition of the RCE, the difference between the simulated and the calculated currents is divided by  $I_{REF} = 1 \mu\text{A}$ , representing the maximum signal value. For example, for the current vector  $I_{MAT} = [5 \text{ nA } 15 \text{ nA}]$ , obtained from the calculations using equation (3.8), and for the simulated one  $I_{SIM} = [4 \text{ nA } 14 \text{ nA}]$ , different due to effects such as leakage charge injection, etc., the computational error  $\text{RCE} = 1\text{nA}/1\mu\text{A} = 0.1 \%$ . However, normalising both results gives  $\|I_{MAT}\| = [0.25 + 0.75]$  and  $\|I_{SIM}\| = [0.22 + 0.78]$  resulting in NCE equal 3 %. When the results are very small, even very low disparity in the simulated and the calculated currents generates high normalised error. When  $C_G$  increases, the NCE decreases for large output currents (typically for large  $I_0$ ) and increases when the output currents are small (typically for small  $I_0$ ). These effects tend to compensate each other and the mean value of the RCE remains on the same level for  $C_G$  higher than 10 fF.

The effect of shortening the single cycle time  $T_C$  from 2  $\mu\text{s}$  to 200 ns on the mean value and standard deviation of the computational errors, is presented in Figure 3.16. It can be observed that the relative current error (RCE) does not increase significantly until the cycle time becomes shorter than 0.5  $\mu\text{s}$ , whereas the normalised current error (NCE) keeps increasing within the entire time interval. This results from the fact that the shorter cycle time does not allow the gate capacitances  $C_G$  to charge properly when the input current  $I_0$  is very low. For very low input currents, however, the relative current error becomes less significant (due to the division by  $I_{REF}$ ), therefore, the level of RCE remains constant in the right hand side segment of the plot in Figure 3.16b. Since the NCE performs normalisation of the obtained result, even small variations of the low currents are magnified giving higher computational error.

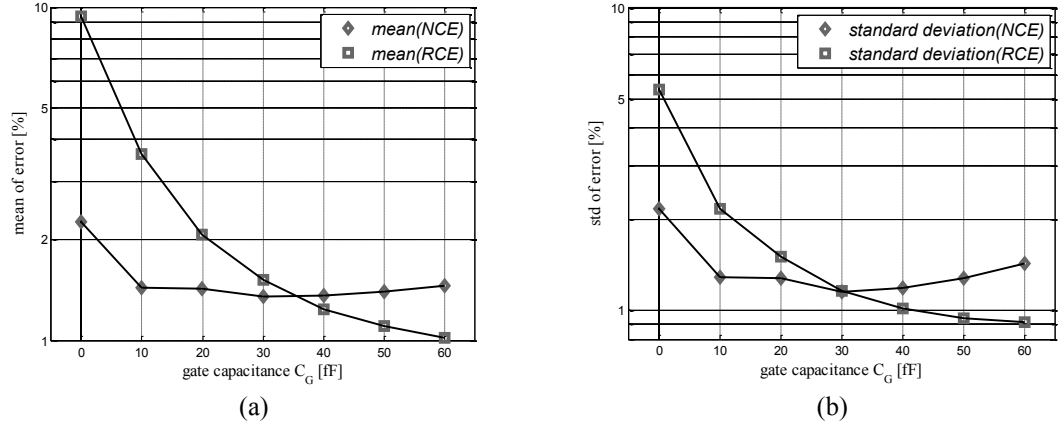


Figure 3.15. The influence of the gate capacitance  $C_G$  on the measures of the computational error NCE and RCE expressed by: a) the mean value, and b) the standard deviation.

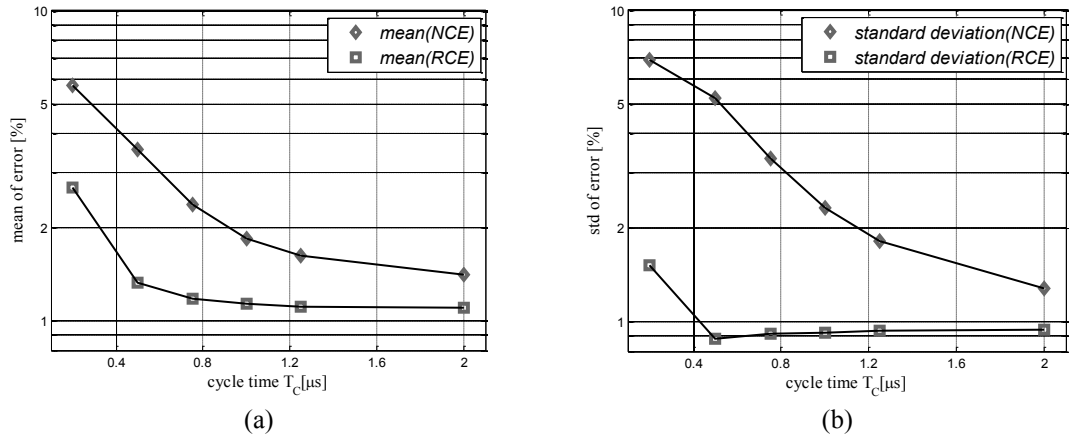


Figure 3.16. The influence of shortening the cycle time  $T_C$  on the measures of the computational error NCE and RCE expressed by a) the mean value, and b) the standard deviation.

The relation between the computational error and the range of the input currents, expressed by the reference current  $I_{REF}$  defining the maximum signal value, is presented in Figure 3.17. It should be noted that the assumed value of  $I_{REF}$  affects the operation region of the transistors and the range of input and output currents. The accuracy of the multiplier, when operating with smaller currents will depend more on the charge injection errors and reduced charging time. For the larger currents, the accuracy starts degrading when the circuit does not operate in the assumed region (i.e. in the saturation and weak inversion). Also, when moving the operation region towards very low currents may degrade the computational accuracy due to noise, whereas moving it towards higher currents will increase the power consumption. Therefore, the selection of the reference current  $I_{REF}$ , defining the operation range of the multiplier, should be seen as an optimisation process aiming maximisation of the computational accuracy and efficiency

at the same time. For the proposed design of the discrete time multiplier, the reference current  $I_{REF} = 1 \mu\text{A}$  was chosen experimentally in the simulations to assure the minimum computational errors.

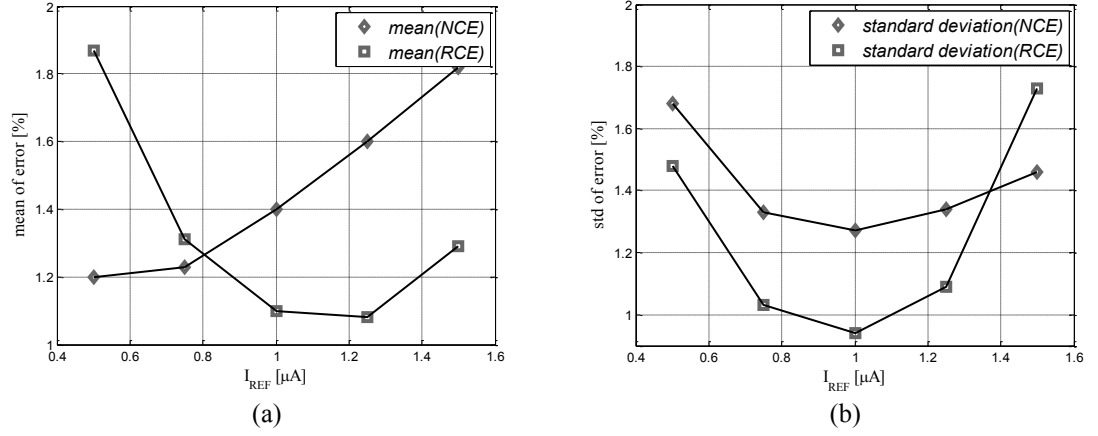


Figure 3.17. The influence of the value  $I_{REF}$  on the measures of the computational error NCE and RCE expressed by a) the mean value, and b) the standard deviation.

### 3.4 Fixed point digital implementation

The computational accuracy of the analogue realisations of the Gilbert multiplier, discussed in this chapter, was compared with its functionally equivalent digital counterpart, implemented in software and using a fixed precision arithmetic and integer numbers represented by binary words of the number of bits  $N$  in range from 5 to 10. The block diagram of the digital multiplier is presented in Figure 3.18.

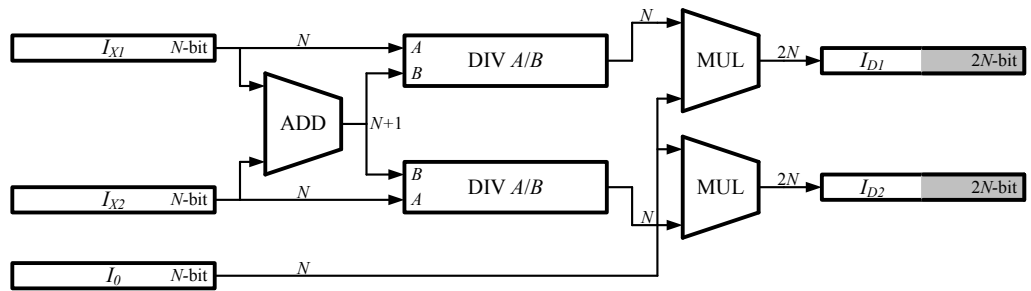


Figure 3.18. Block diagram of the digital equivalent of the Gilbert cell from Figure 3.1 realising multiplication and input vector normalisation using  $N$  bit fixed point arithmetic.

The input vector  $[I_{X1} \ I_{X2}]$  is normalised using  $N + 1$  bit adder and two  $N$  bit dividers, and then multiplied by  $I_0$  using two multipliers. The two  $2N$  bit numbers obtained from the multiplication represent the output currents  $[I_{D1} \ I_{D2}]$ . Since the system operates on  $N$  bit arguments, the result of the multiplication is truncated, leaving only the higher halves

of the computed words. This introduces an additional error with maximum value of 1 LSB (least significant bit). This error could be reduced down to 0.5 LSB, if an additional rounding circuit was employed. It should also be noted that the implementation of the divider block is not straightforward, like the adder or the multiplier. One way of realising such division is first to multiply the input argument  $A$  by  $2^N$  (equivalent to logic left shift by  $N$  bits), and then divide such  $2N$  bit number by the  $N + 1$  bit argument  $B$  using integer divider. The computational errors of a digital implementation cannot be attributed to only one functional block or strictly associated with a particular argument bit length  $N$ . It may be possible to implement the functionality of the system presented in Figure 3.18 using blocks operating with different levels of bit precision, achieving higher computational accuracy for a given number of gates, circuit area, and speed. Nevertheless, such design optimisation goes beyond the scope of this research, since the main objective here is to compare the computational precision of the analogue multipliers with their digital equivalent, operating with a fixed bit precision  $N$ . Digital design optimisation in terms of the structural complexity will be considered in Chapter 8, dealing with realisations of larger arithmetic systems, dedicated for matrix-vector operations in the sum-product algorithm.

The mean and standard deviation measures of the computational errors NCE and RCE, in terms of the bit precision  $N$  of the system, generated by the proposed digital multiplier, are shown in Figure 3.19.

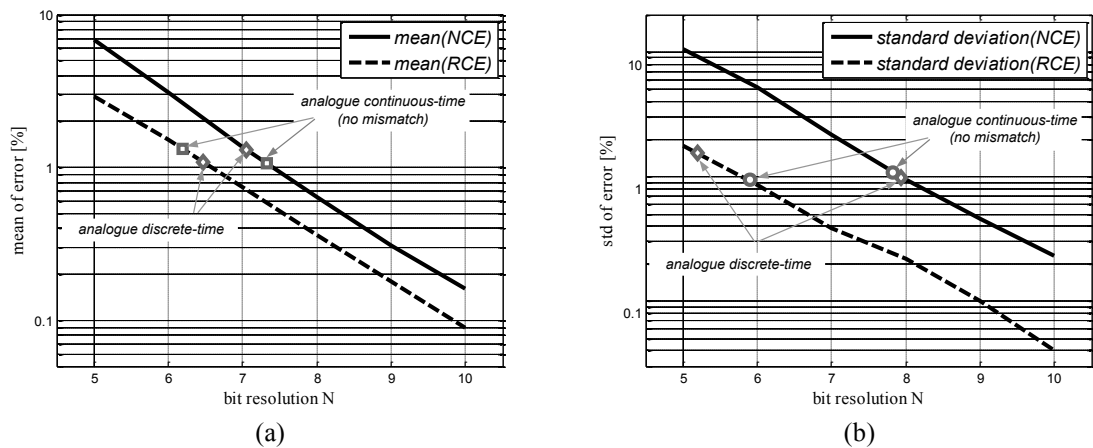


Figure 3.19. a) Mean value, and b) standard deviation of the computational errors NCE and RCE in terms of the precision  $N$  of the digital implementation of the Gilbert multiplier.

For comparison, the computational errors of the analogue continuous-time and discrete-time realisations have also been inserted in the Figure 3.19. It should be noted, however, that the analogue and the digital realisations are substantially different systems,

generating computational errors with different distributions. Therefore, statistical parameters such as mean value and standard deviation of the error provide a rather baseline comparison of the analogue and digital realisations. In particular, according to the NCE measure, the mean value of the analogue realisations corresponds to the 7 bit precision of the digital multiplier (Figure 3.19a), but with error spread corresponding to 8 bit precision (Figure 3.19b). In the case of the NCE error, these figures correspond to precision of 6 bits and 5 - 6 bits respectively. The distributions of the generated computational errors NCE and RCE of the digital multiplier are presented in Figure 3.20.

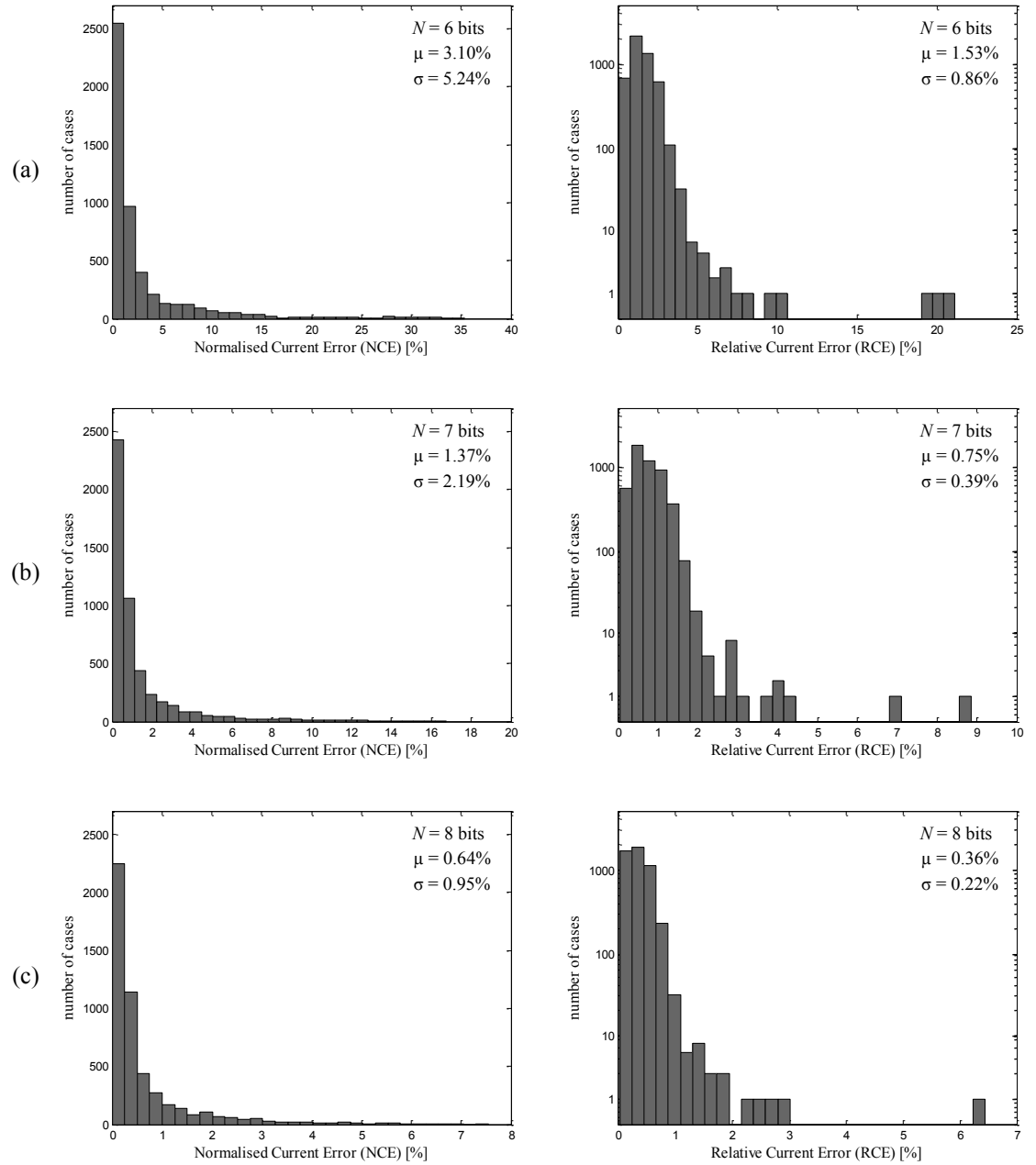


Figure 3.20. Histograms of the absolute computational error of the fixed point digital multiplier from Figure 3.18 for precision  $N$  equal: a) 6 bits, b) 7 bits, and c) 8 bits.

### 3.5 Summary and conclusions

In this chapter two approaches to analogue computation in CMOS circuits employing continuous-time and discrete-time processing, were presented. The idea, design and the circuit-related issues of the current-mode multipliers were discussed. In particular, the impact of the second order effects of MOS devices and the parameter mismatch on the computational error was further investigated. It has been concluded that parameter variability of a CMOS process highly affects the operation of the continuous-time circuits. This could be improved by transistor size scaling, however, at the expense of the excessive area increase. The employment of the switched-current technique, in the realisation of the discrete-time multiplier, allowed to minimise the effects of mismatch in the circuit operation, but at the expense of the reduced processing speed and slightly increased computational errors. The analyses and simulations reported the computational precision equivalent to 6 - 7 bits in the fixed point arithmetic, which is sufficient for moderate precision applications with strict power and area requirements.

The presented analogue multipliers will be of use in Chapter 8, dealing with the designs of arithmetic circuits for continuous-time and discrete-time realisations of the sum-product algorithm for belief propagation. The experimental verification of the discrete-time multiplier in analogue computation is necessary and will be done in the future using the prototype processor array implemented on the PPATC test chip.

## Chapter 4

---

# Delay lines

---

### 4.1 Chapter overview

This chapter presents the idea and design of a tunable CMOS delay gate (*output-split inverter*, OSI) exhibiting lower impact of fabrication mismatch on the generated delay time intervals than the commonly used *current starved inverter* (CSI) of the same size. The operation of the CSI and OSI circuits under parameter variability is analysed in theory and verified in simulations accounting for the realisations in different technology nodes, and in experiments using delay line arrays implemented on the PPATC chip fabricated in a 90 nm CMOS technology. The proposed circuit will be of use in the design of the asynchronous processor array presented in Chapter 5.

### 4.2 Introduction

Tunable CMOS delay gates and delay lines are important functional sub-blocks in various applications requiring generation and measurement of the controlled delay time intervals, such as delay locked loops (DLL) [Christiansen 95], time-to-digital converters (TDC) [Dudek 2000b], silicon pixel readout circuits for particle detection [Heijne 96], neuromorphic circuits [Indiveri 2006], [Wang 2013] and asynchronous processor arrays (discussed in Chapter 5). Due to parameter variability caused by fabrication process, an array of identically designed delay gates or delay lines will generate delay time intervals with randomly varying offsets, even under the same supply and bias conditions. Such

mismatch of the generated time intervals is usually reported as a dominant factor limiting precise operation of a systems relying on timing parameters [Cantatore 97]. The majority of solutions found in the literature employ a typical structure of a delay gate based on the current starved inverter (CSI) circuit, shown in Figure 4.1a.

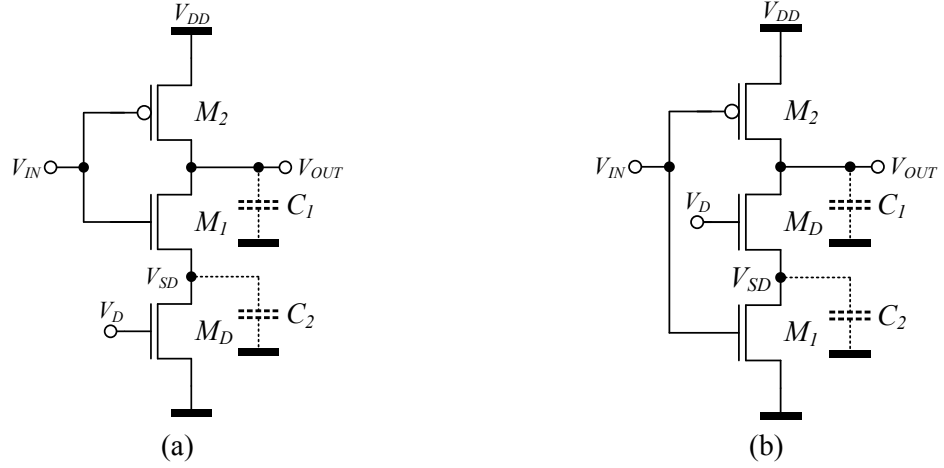


Figure 4.1. Schematic diagrams of the nMOS asymmetric delay gate circuits: a) the commonly used current starved inverter (CSI), and b) the proposed output-split inverter (OSI).

In the majority of designs employing the current starved inverter circuit, mismatch optimisation is done by proper scaling of the current limiting transistor  $M_D$  (Figure 4.1a), mostly contributing to the generated time variability [Bolt 96]. An extensive research on the fabrication mismatch in the CSI gates was done for the design of particle detectors used in the experiments in nuclear physics. It has been shown that the accuracy of such circuits, and hence the precision of a detector, is dominated by the variability of timing parameters of the delay lines used for event buffering and synchronisation. In particular, various approaches to design of delay lines, immune to parameter mismatch, were investigated in [Toifl 99] and [Cantatore 97], as a preliminary step towards building detectors for the Large Hadron Collider (LHC) in CERN. Optimisation techniques, other than transistor scaling, account for the use of delay locked loops for individual biasing [Christiansen 95], post fabrication trimming [Heijne 96], and layout drawing techniques considering design symmetry, shadow effects and drain/source contact resistances [Bolt 96]. Reduced mismatch, is usually achieved at the expense of additional circuit area, higher power consumption and more complex tuning scheme, which in some applications is not desired.

In this thesis, an alternative structure of a delay gate, the *output-split inverter* (OSI) shown in Figure 4.1b, is presented. The only topological difference between the CSI and the OSI circuits is the location of the current limiting transistor  $M_D$  on the drain rather

than source side of the switching transistor  $M_I$ , which separates or *splits* the output of the inverter.

In principle, the operation of both circuits is similar and, for the rising input edge, the transistor  $M_D$  controls the current discharging the output capacitance  $C_I$  regulating the output signal falling edge slope and discharge time. However, during the transient state, both circuits behave differently, which has a direct impact on their performance under the process parameter variability. It should be noted that the proposed OSI structure was previously used, for example, in the build of charge pump circuits [Christiansen 95], [Indiveri 2006], [Wang 2013] and linearly tuned delay elements [Jovanovic 2006]. Nevertheless, it has never been used in the applications where matching is of critical importance, nor its operation has ever been analysed in detail, providing particular description or mathematical model. Therefore, the primary goal of the work presented in this chapter is to fill this gap by presenting theoretical analysis and experimental verification of the operation and performance of the delay lines constructed using CSI and OSI gate.

### 4.3 Circuit operation

The implementation of both delay gates presented in Figure 4.1 uses the structure of a logic inverter (transistors  $M_I$  and  $M_2$ ) and employs the idea of delaying the output slope by discharging the capacitance  $C_I$  with the drain current of  $M_D$  controlled by the bias voltage  $V_D$ . Assuming that the transistors  $M_I$  and  $M_2$  work as ideal switches, the operation of both circuits is very similar. Depending on the transition of the input signal, the slew of the output signal is either controlled by the current limiting transistor  $M_D$  (for the rising input edge) or is determined by the strength of  $M_2$  pulling the output node up to  $V_{DD}$  (for the falling input edge). However, more detailed analysis of these gates reveals substantial differences in their operation which are of high importance in terms of the process parameter variability and its influence on circuit performance. The simulation results showing the transitions of  $V_{IN}$ ,  $V_{OUT}$  and  $V_{SD}$  signals of the asymmetric CSI and OSI delay gates from Figure 4.1, for the rising input slope (when the output load  $C_I$  discharges through the current limiting transistor  $M_D$ ), are presented in Figure 4.2. In the simulations, MOS transistor models from a standard 90 nm CMOS technology were used assuming the same sizes for the current limiting transistors  $W_D/L_D = 1\mu\text{m}/0.5\mu\text{m}$  and for the switching transistors  $W_{I,2}/L_{I,2} = 1\mu\text{m}/80\text{nm}$ . Capacitances  $C_I$  and  $C_2$  are always

present due to the junction capacitances of drain and source areas of MOS transistors. An additional capacitance of 1 fF has been added to represent the external load of the output node. In the full custom layout design, both transistors  $M_D$  and  $M_I$  may share the same diffusion stripe, therefore, there is no additional capacitance attached to the  $V_{SD}$  node, apart from the geometry-dependent one associated with the drain and source regions, already included in MOS transistor model. In the simulations, the bias voltage  $V_D = 300$  mV was used for both gates and the generated delay time  $T_D$  was measured between 50% levels of the input ( $V_{IN}$ ) and output ( $V_{OUT}$ ) signals.

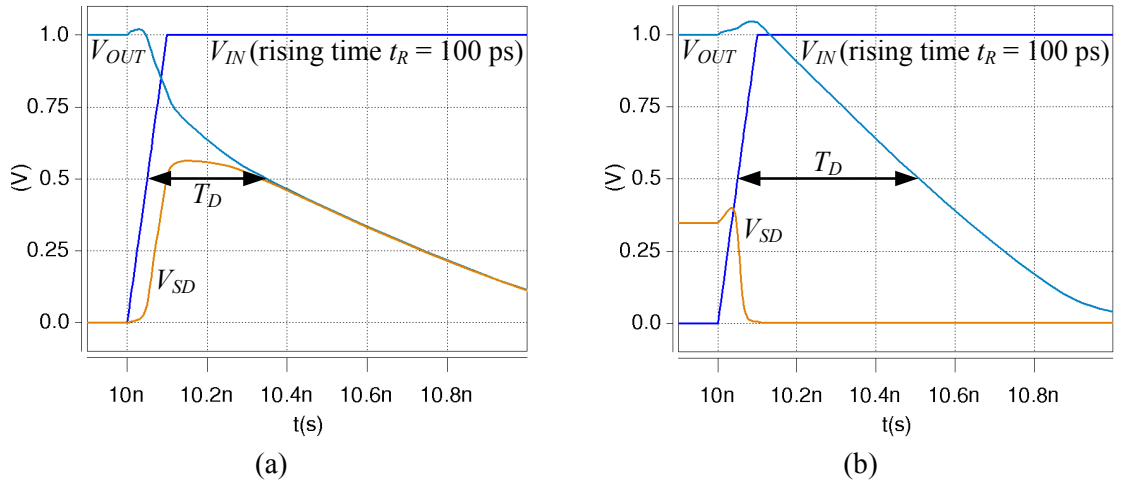


Figure 4.2. Simulation results of the circuits from Figure 4.1: a) CSI gate, and b) OSI gate for the rising edge of the input voltage  $V_{IN}$ ,  $V_D = 300$  mV,  $V_{DD} = 1.0$  V and  $t_R = 100$  ps (MOS transistor models from a 90 nm CMOS technology were used assuming  $W_D/L_D = 1\mu\text{m}/0.5\mu\text{m}$ ,  $W_{I,2}/L_{I,2} = 1\mu\text{m}/80\text{nm}$ ;  $C_1 = 1$  fF,  $C_2 = 0$  plus inherent source/drain MOS capacitances).

The main difference in the operation of both delay gates can be observed during the transient state, when the rising input edge turns  $M_I$  on and  $M_2$  off. In the case of the proposed OSI gate, capacitance  $C_2$  (initially charged close to  $V_D$  through  $M_D$  when  $M_I$  is off for  $V_{IN} = 0$ ) quickly discharges to zero whereas capacitance  $C_1$  is gradually discharged from  $V_{DD}$  to zero by the drain current of  $M_D$  (Figure 4.2b). In the CSI circuit, however, capacitance  $C_2$  is initially discharged since  $M_D$  is always on, and the rising input edge causes the transistor  $M_I$  to short its drain and source nodes such that the voltages  $V_{OUT}$  and  $V_{SD}$  converge close to the common value  $V_{CM}$ , before the load capacitance starts discharging (Figure 4.2a). As a result, in the CSI circuit capacitance  $C_1$  will discharge quicker and the output voltage  $V_{OUT}$  will drop faster to 50% of  $V_{DD}$  for the same current of  $M_D$  as compared to the OSI one. The variability of the slew of the output signal is mostly dependent on the parameter variability of  $M_D$  and the output load, but in

the CSI circuit the discharge time will also be affected by the variability of  $V_{CM}$ , resulting from the mismatch between transistors  $M_1$  and  $M_2$ . Therefore, the commonly used CSI circuit tends to generate shorter and more variable delay intervals than the proposed OSI structure, where capacitance  $C_1$  of the output node always discharges from the constant  $V_{DD}$  voltage. In the following a simplified analysis presenting only the first-order behaviour will be provided addressing the major differences between circuits in Figure 4.1 and their operation under the presence of the process parameter variability.

#### 4.3.1 Current starved inverter (CSI)

A simplified analysis of the CSI circuit showing the transitions of voltages  $V_{IN}$ ,  $V_{OUT}$  and  $V_{SD}$  is presented in Figure 4.3. The timeline can be divided into three phases: the initial phase ( $t < t_1$ ), the switching phase ( $t_1 < t < t_2$ ), and the discharge phase ( $t > t_2$ ).

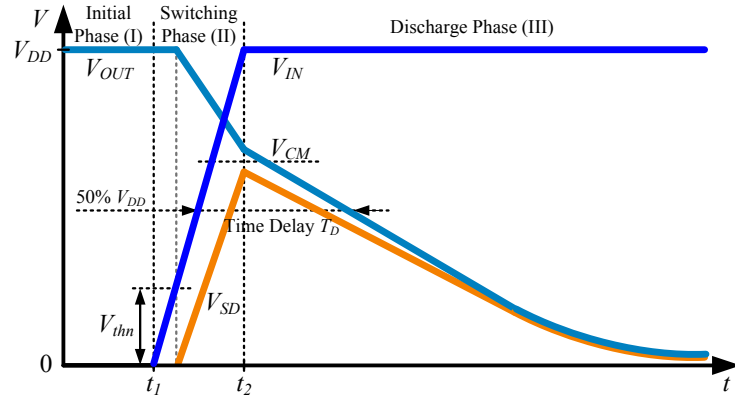


Figure 4.3. The behaviour of the CSI delay gate in a transient state for the rising input edge.

In the initial phase, the capacitance  $C_2$  is discharged to zero through the current limiting transistor  $M_D$  (for  $V_D > 0$ ) and capacitance  $C_1$  is charged to  $V_{DD}$  through  $M_2$ . In the switching phase, the rising edge of  $V_{IN}$  turns  $M_1$  on and  $M_2$  off. As a result  $V_{OUT}$  and  $V_{SD}$  converge closer to the common level  $V_{CM}$  denoting the starting point for the discharge phase. In a closer view, the transistor  $M_1$  turns on first (when  $V_{IN} > V_{thn}$ ) quickly increasing the conductance between nodes  $V_{OUT}$  and  $V_{SD}$ . In the same time the drain-source conductance of  $M_2$  decreases practically disconnecting the output node from the power rail. Due to the current limiting transistor  $M_D$ , the total current flowing through  $M_1$  and  $M_2$  is reduced and the observed output voltage drop (from  $V_{DD}$  to  $V_{CM}$ ) can practically be attributed to the charge sharing between capacitances  $C_1$  and  $C_2$ . During the discharge phase the high logic level on the input keeps transistor  $M_1$  fully turned on which connects capacitances  $C_1$  and  $C_2$  in parallel. The discharge rate of these

capacitances depends mainly on the bias voltage  $V_D$  controlling the current limiting transistor  $M_D$ .

### 4.3.2 Output split inverter (OSI)

The analysis of the OSI circuit showing the transitions of voltages  $V_{IN}$ ,  $V_{OUT}$  and  $V_{SD}$ , is presented in Figure 4.4. In the following only the first-order effects will be discussed indicating the major differences in the operation between the CSI and OSI structures.

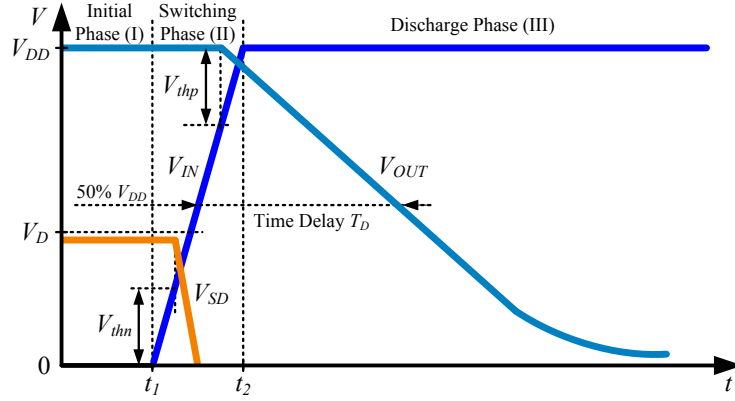


Figure 4.4. The behaviour of the OSI delay gate in a transient state for the rising input edge.

In the initial phase, voltage  $V_{IN}$  equals zero assuring that capacitance  $C_1$  is charged to  $V_{DD}$  through the transistor  $M_2$ . The current limiting transistor  $M_D$  operates in weak inversion with its gate-source voltage high enough above zero to conduct the small off current of  $M_1$ , therefore the capacitance  $C_2$  remains charged closely to the gate bias voltage  $V_D$ . In the switching phase the rising edge of the input signal turns  $M_1$  on (when  $V_{IN} > V_{thn}$ ), which quickly discharges  $C_2$  to zero and, after that, turns  $M_2$  off (when  $V_{IN} > V_{DD} - |V_{thp}|$ ) and  $C_1$  starts discharging with rate dependent mainly on the drain current of  $M_D$  controlled by the bias voltage  $V_D$ . Further operation of this gate is practically the same as in the case of the CSI one.

## 4.4 Mismatch analysis

The effects of parameter mismatch in the realisations of the CSI and OSI delay gates are presented in Figure 4.5. In the simulations, the same circuit realizations as before (Figure 4.2) were used, but with the mismatch Monte Carlo MOS transistor models and bias voltages  $V_D$  tuned for both gates to ensure the same mean value of the generated

delays  $T_D$ . Based on 5000 Monte Carlo simulation runs, in Figure 4.5, it can be seen that the random variability of the generated delay is larger in the CSI gate.

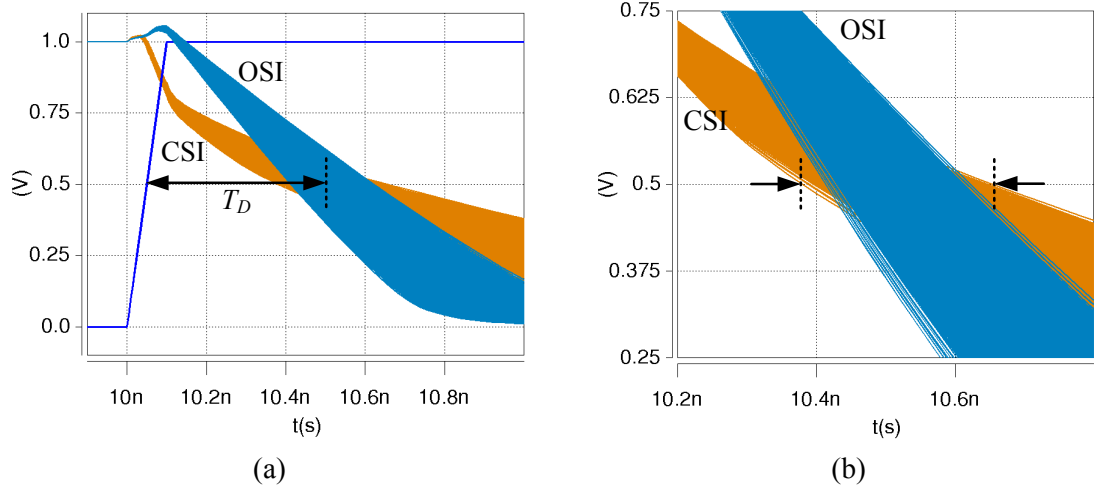


Figure 4.5. Transient mismatch Monte Carlo simulations (5000 runs) of the delay gates from Figure 4.2 tuned to generate equal mean delays  $T_D \approx 0.46$  ns ( $V_D = 280$  mV for the CSI and  $V_D = 300$  mV for the OSI gate): a) input and output signal transitions, b) detailed view of the output signals crossing the 50%  $V_{DD}$  threshold.

The detailed simulation results, accounting for the variability of the generated delay  $T_D$  caused by mismatch of individual transistors in the delay gates are presented in Table 4.1. In particular, the effects of the input signal slope variability were verified using additional buffer BUFF consisting of two inverters (designed using the same transistor sizes as  $M_1$  and  $M_2$  in Figure 4.1) and driving the input of the delay gate with a fixed load capacitance of 1 fF. The results were obtained in the simulations of the circuit presented in Figure 4.6, based on 500 Monte Carlo runs using mismatch MOS transistor models with mismatch flag (one of the input parameters of the model) set to 1 or 0, in order to individually activate or deactivate random generators in the transistors. It can be concluded that the variability of the generated delay time  $T_D$  in both gates depends mainly on the variability of the current limiting transistor  $M_D$ .

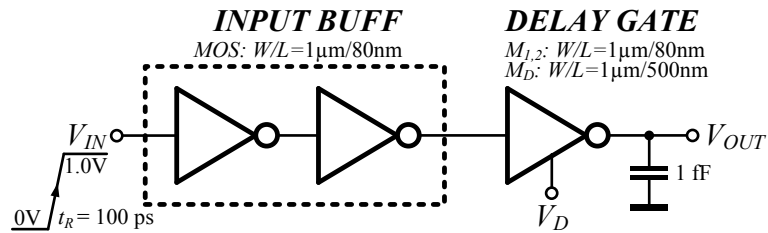


Figure 4.6. Schematic diagram of the circuit used in simulations of the time delay variability of the CSI and OSI delay gates presented in Figure 4.1.

Table 4.1. Mismatch Monte Carlo simulation results of the CSI and OSI gates.

<i>Mismatch in</i>	CSI ( $V_D = 280$ mV)		OSI ( $V_D = 300$ mV)	
	$T_{DMEAN}$ [ps]	$\sigma_{TD}$ [ps]	$T_{DMEAN}$ [ps]	$\sigma_{TD}$ [ps]
$M_D$	459.00	41.923	461.53	32.407
$M_D + M_1$	459.35	42.381	461.64	32.438
$M_D + M_2$	459.08	41.986	461.59	32.410
ALL	459.43	42.434	461.69	32.441
ALL + BUFF	477.19	45.259	475.62	34.695
$M_{1,2} + \text{BUFF}$	475.04	2.187	474.21	1.820
<i>no mismatch</i>	474.81	$5 \times 10^{-12}$	474.15	$3 \times 10^{-12}$

In the following section, a qualitative analysis of the switching and discharge phases will be provided, supporting the obtained simulation results and explaining the influence of the MOS parameter variability on the precision of the generated delays.

#### 4.4.1 Mismatch in CSI gate

The analysis of the CSI circuit is presented in Figure 4.7 showing the influence of the process parameter fluctuation on the variability of the  $V_{CM}$  voltage and the generated delay time  $T_D$ .

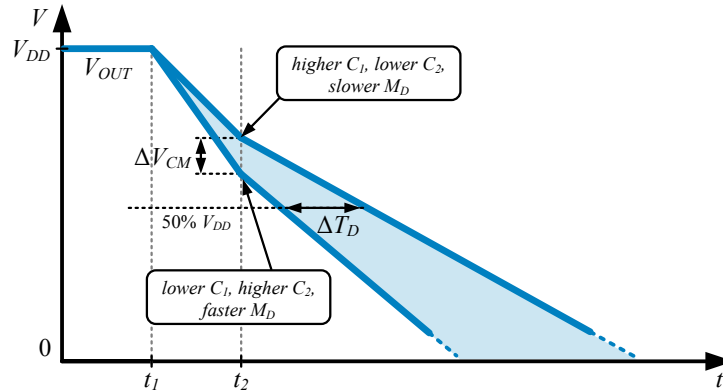


Figure 4.7. The transient state of the CSI circuit showing the influence of the MOS parameter variability on the  $V_{CM}$  voltage and the generated delay  $T_D$ .

During the switching phase ( $t_1 < t < t_2$ ), both voltages  $V_{SD}$  and  $V_{OUT}$  converge towards the common level  $V_{CM}$  which, in a crude approximation, can be estimated from the charge sharing between  $C_1$  and  $C_2$  (this will be further explained in Section 4.4.4). Therefore, the variability of the voltage  $V_{CM}$  will be affected by the variability of these capacitances but also by the variability of all the MOS transistors, especially  $M_D$ , and other effects such as off (leakage) current of  $M_2$ , and the capacitive coupling between the input and output. The discharge phase will mainly be affected by the variability of the current limiting transistor  $M_D$ . For example, due to the random variation of the threshold

voltage  $V_{thn}$  of  $M_D$ , this transistor may be slightly "faster" (higher drain current for lower values of  $V_{thn}$ ) or slightly "slower" (lower drain current for higher values of  $V_{thn}$ ) than a regular one. For the faster  $M_D$ , the corresponding discharge slope will be steeper and also the  $V_{CM}$  voltage will be lower (due to the higher current of  $M_D$  discharging the output node during the switching phase), whereas for the slower  $M_D$ , the  $V_{CM}$  voltage will reach a higher value and the discharge phase will take a longer time. As a result, it can be observed that not only the parameter variability of the current limiting transistor  $M_D$  but also the variability of  $V_{CM}$  voltage ( $\Delta V_{CM}$ ) will affect the precision  $\Delta T_D$  of the generated time delay  $T_D$ .

#### 4.4.2 Mismatch in OSI gate

The analysis of the OSI circuit showing the influence of the MOS parameter fluctuations on the variability of the generated time delay  $T_D$  is presented in Figure 4.8.

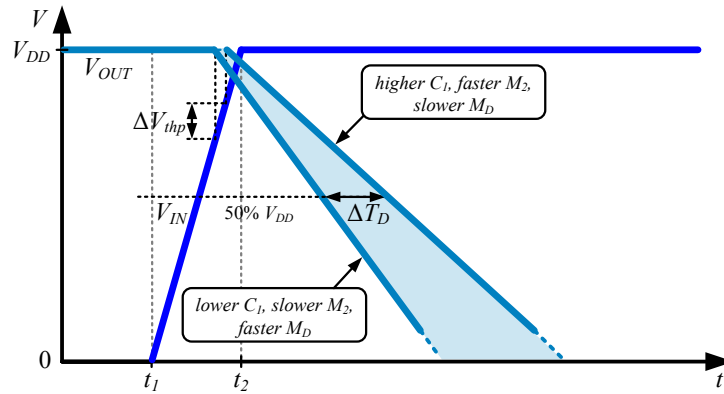


Figure 4.8. The transient state of the OSI circuit showing the influence of the MOS parameter variability on the generated delay time  $T_D$ .

Due to the current limiting transistor  $M_D$ , "splitting" the output of the inverting stage, the rising edge of  $V_{IN}$  may not force an immediate transition of  $V_{OUT}$ , as it was observed in the case of the CSI circuit. While the capacitance  $C_2$  quickly discharges to zero,  $M_2$  still pulls the output node up to  $V_{DD}$ , postponing the discharge phase roughly until  $V_{IN}$  crosses the threshold  $V_{DD} - |V_{thp}|$  switching transistor  $M_2$  off. The variability  $\Delta T_D$  in the generated delay time will mainly depend on the parameter mismatch in transistors  $M_D$  and  $M_2$ . Similarly as before, the slightly faster transistor  $M_D$  will force the discharge phase earlier and will discharge the output capacitance  $C_1$  faster. Additionally, for the slightly slower transistor  $M_2$  with a higher threshold voltage  $|V_{thp}|$ , the discharge phase may begin earlier than for the slightly faster one, further increasing the variability of the

generated delay time  $T_D$ . The influence of the variability of the threshold voltage  $\Delta V_{thp}$  of  $M_2$  is usually suppressed by a sharp slope of  $V_{IN}$ . Also, the discharge phase always begins for the same output voltage  $V_{OUT} = V_{DD}$  irrespective of the variability in  $C_1$  and  $C_2$ . Because of this, the starting point of the discharge phase is more stable ( $C_1$  is always charged to the constant voltage  $V_{DD}$ ) and the discharge time of  $C_1$  is longer for the same current of  $M_D$  as compared to the CSI structure where  $V_{CM} < V_{DD}$ . This makes the generated delay time of the OSI circuit less prone to mismatch.

#### 4.4.3 Simplified analytical model

In the proposed simplified analytical model only the dynamic behaviour of the CSI and OSI delay gates will be considered during the discharge phase (with the initial conditions determined by the switching phase). It is assumed that the current limiting transistor  $M_D$  operates in saturation and strong inversion regions for the generated delay time interval  $T_D$ . This assumption holds for typical applications where the output signal triggers the next stage (e.g. the next gate in a delay line) at the 50% signal level, which is higher than a value of the saturation voltage of  $M_D$  (usually  $V_{DSAT} \ll V_{DD}/2$ ). Also, the gate-source voltage of  $M_D$  (equal to  $V_D$  when  $M_I$  is fully turned on) is usually higher than the threshold voltage  $V_{thn}$ , in order to avoid the increased impact of parameter mismatch on the circuit operation, when  $M_D$  is in the subthreshold region. The schematic diagrams of the simplified CSI and OSI delay gates, representing the state of each circuit after the switching phase, are shown in Figure 4.9.

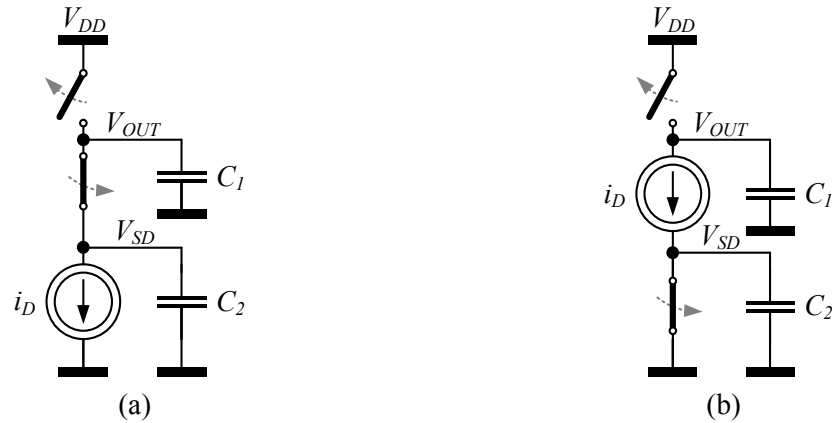


Figure 4.9. Schematic diagrams of the simplified delay gates representing the state of the circuits after the switching phase for: a) CSI delay gate (initial condition:  $V_{OUT} = V_{CM}$ ), and b) OSI delay gates (initial condition:  $V_{OUT} = V_{DD}$ ).

In both cases, the current limiting transistor  $M_D$  was replaced with an ideal current source  $i_D$ . The transistors  $M_I$  and  $M_2$  were replaced with switches, where the non-ideal

behaviour of these devices, in the case of the CSI gate, can be seen as an additional factor affecting the  $V_{CM}$  voltage. The assumption of charge sharing between  $C_1$  and  $C_2$ , as the primary reason for the output voltage drop, was verified in simulations and remains valid almost within the entire tuning range except for the very short delays for  $V_D \gg V_{thn}$  when  $M_D$  starts pulling the node  $V_{SD}$  and  $V_{OUT}$  closer to zero during the switching phase. In particular, for the gate design implemented on the test chip, the inherent geometry dependent drain/source capacitances are approximately equal to 2.5 fF for the  $V_{OUT}$  node, and 1.5 fF for the  $V_{SD}$  node. The values were calculated assuming charge sharing effect with the resulting  $V_{CM}$  voltage obtained from simulations for different load capacitances, as discussed in the following section.

#### 4.4.4 Charge sharing and S/D inherent capacitances

In the proposed model of the CSI gate, the input signal slope turns  $M_1$  on and  $M_2$  off such that the output voltage  $V_{OUT}$  drops down to a certain common value  $V_{CM}$ , as a result of charge sharing between fully charged  $C_1$  and discharged  $C_2$ . However, these transistors are non-ideal (resistive) switches, and also  $M_1$  turns on before  $M_2$  turns off, for the rising edge of the input signal. This will disrupt the charge sharing process and the resulting  $V_{CM}$  voltage will be different than  $V_{DD} \times C_1 / (C_1 + C_2)$ . In order to verify the relevance of the proposed charge sharing approximation, the test circuit of the CSI gate, presented in Figure 4.10, will be considered.

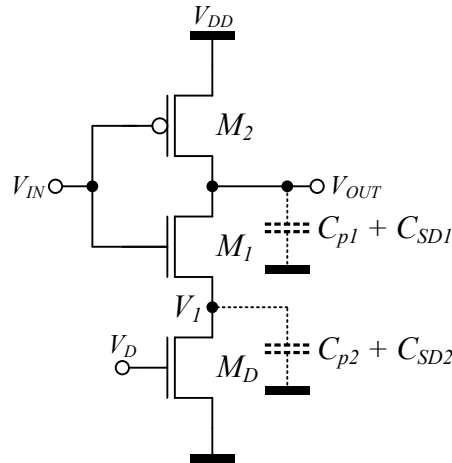


Figure 4.10. Schematic diagram of the CSI delay gate used in the verification of the charge sharing effect and extraction of the inherent source/drain capacitances  $C_{SD}$ .

Due to parasitics, in this circuit the external capacitances  $C_{p1}$  and  $C_{p2}$  relate to the internal node capacitances, whereas  $C_{SD1}$  and  $C_{SD2}$  denote the inherent, geometry

dependent capacitances of the drain and source regions. It was assumed that  $C_{SD1}$  and  $C_{SD2}$  are constant and the output voltage drop results from charge sharing between  $C_{p1} + C_{SD1}$  and  $C_{p2} + C_{SD2}$ . The asymmetric CSI delay gate, presented in Figure 4.10, was simulated for different configurations of  $C_{p1}$  (from 0 to 1 fF) and  $C_{p2}$  (from 0 to 0.2 fF) shown in Table 4.2. In the simulations, the size of the switching transistors  $M_1$  and  $M_2$  was  $1\mu\text{m}/80\text{nm}$ , and the size of the current limiting transistor  $M_D$  was  $1\mu\text{m}/500\text{nm}$ . Voltages  $V_{OUT}$  and  $V_I$  converge towards the common value  $V_{CM}$  during the transition state but usually don't meet. Therefore, the  $V_{CM}$  voltage was calculated as the mean value of  $V_{OUT}$  and  $V_I$ , at the point where  $V_I$  reached maximum. The values of the capacitances  $C_{SD1}$  and  $C_{SD2}$  were calculated individually for the cases presented in Table 4.2, assuming the following relations:

$$\frac{V_{CM0}}{V_{DD}} = \frac{C_{SD1}}{C_{SD1} + C_{SD2}} = A \quad (4.1)$$

$$\frac{V_{CM}}{V_{DD}} = \frac{C_{p1} + C_{SD1}}{C_{p1} + C_{SD1} + C_{p2} + C_{SD2}} = B \quad (4.2)$$

where  $V_{CM0}$  is the common voltage obtained from the simulation assuming  $C_{p1} = C_{p2} = 0$ . Using equations (4.1) and (4.2), the values of  $C_{SD1}$  and  $C_{SD2}$  can be calculated in the following way (for simplicity the voltage ratios were replaced with parameters  $A$  and  $B$ ):

$$C_{SD1} = \frac{C_{p1}(B-1) + C_{p2}B}{1 - B/A} \quad (4.3)$$

$$C_{SD2} = C_{SD1}(1/A - 1) \quad (4.4)$$

The traces showing the variability of the extracted inherent drain/source capacitances  $C_{SD1}$  and  $C_{SD2}$  for the 16 configurations (cases 1 - 16) of  $C_{p1}$  and  $C_{p2}$  from Table 4.2, and bias voltages  $V_D$  within range from 150 mV - 300 mV, covering the tuning range from 0.3 ns to 8 ns, are presented in Figure 4.11. Values of the simulated mean delay time  $T_D$  calculated over all the cases from Table 4.2 and time when the maximum value of  $V_I$  voltage occurred  $T_{VIMAX}$  for bias voltages  $V_D$  considered in the simulations are presented in Table 4.3.

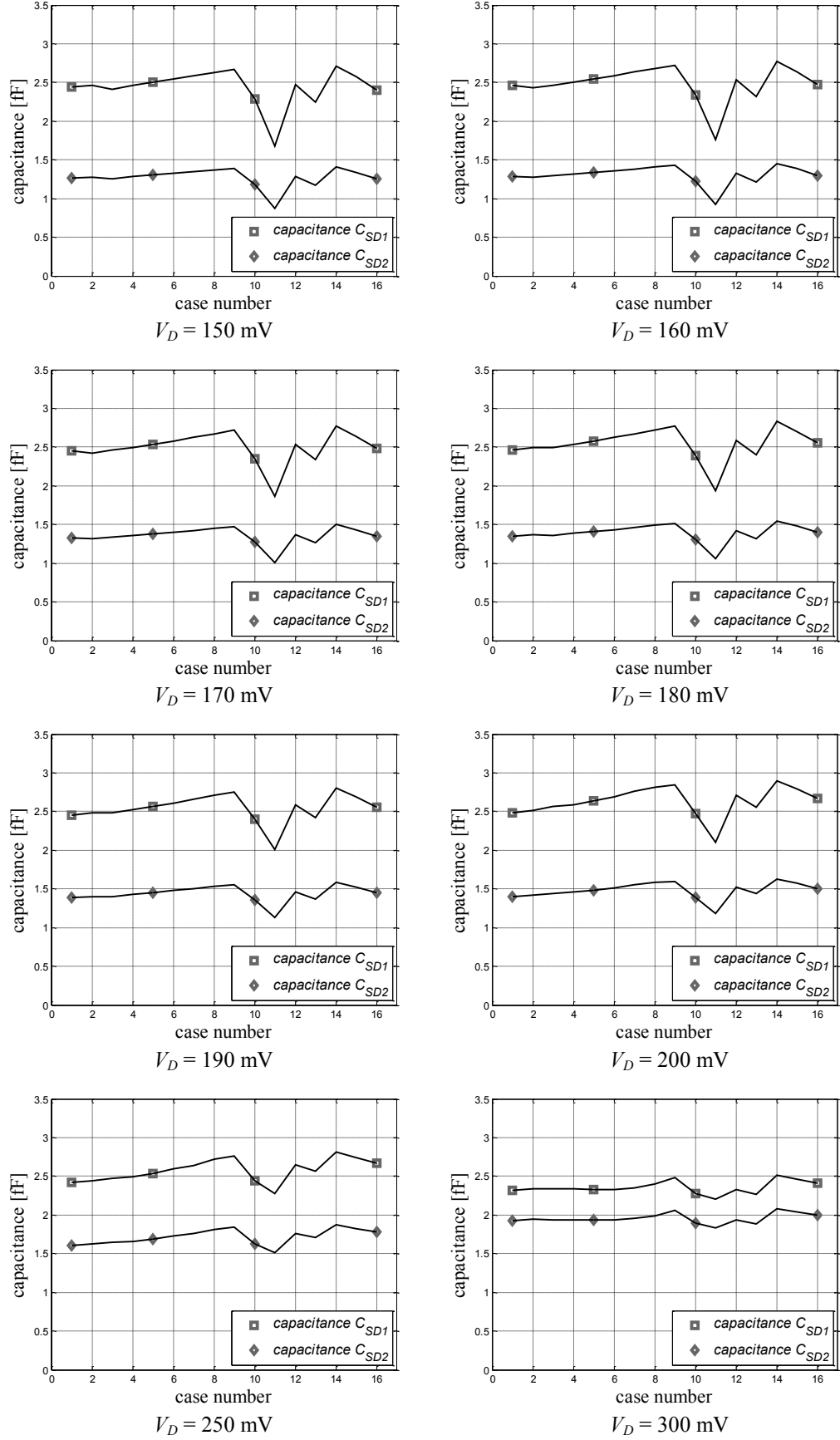


Figure 4.11. Values of the extracted inherent capacitances  $C_{SD1}$  and  $C_{SD2}$  for bias voltages from 150 mV to 300 mV.

Table 4.2. Configurations of capacitances  $C_{p1}$  and  $C_{p2}$  considered in the simulations.

case	(*)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$C_{p1}$ [fF]	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.5	0.5	0.8	0.8	1	1	1
$C_{p2}$ [fF]	0	0	0	0	0	0	0	0	0	0	0.1	0.2	0.1	0.2	0	0.1	0.2

Table 4.3. Timing parameters of the CSI delay gate obtained from simulations.

$V_D$ [mV]	$T_{V1MAX}$ [ns]	$mean(T_D)$ [ns]
150	11.232	8.27
160	10.946	6.22
170	10.829	4.70
180	10.644	3.58
190	10.565	2.75
200	10.425	2.13
250	10.233	0.65
300	10.155	0.28

Assuming charge sharing effect, the calculated values of the inherent source/drain MOS capacitances are approximately equal to  $C_{SD1} \approx 2.5$  fF and  $C_{SD2} \approx 1.5$  fF with variability of  $\pm 30\%$  in range from 150 mV to 300 mV. Since these capacitances are geometry dependent, in a particular circuit realisation they remain constant, which can be observed in the traces in Figure 4.11. That confirms the validity of the charge sharing concept.

For the higher values of the  $V_D$  voltage, the bias current of  $M_D$  increases removing some charge from  $C_{SD1}$  and  $C_{SD2}$  during the switching phase. As a result, some part of the charge is sunk to the ground, which "increases" the equivalent capacitance  $C_{SD2}$  to about 2 fF, and "decreases" the capacitance of  $C_{SD1}$  to about 2.4 fF for  $V_D = 300$  mV. Under such conditions, the bias current of  $M_D$  becomes significant, precluding the use of the proposed model. Nevertheless, for  $V_D > 300$  mV the generated delay  $T_D$  is shorter than 0.3 ns. This means that the proposed approximate model will be invalid only for very short delay time intervals.

#### 4.4.5 Model derivation

In the proposed approach, the delay time  $T_D$  will be derived for the simplified CSI and OSI circuits from Figure 4.9 assuming the discharge scheme presented in Figure 4.12.

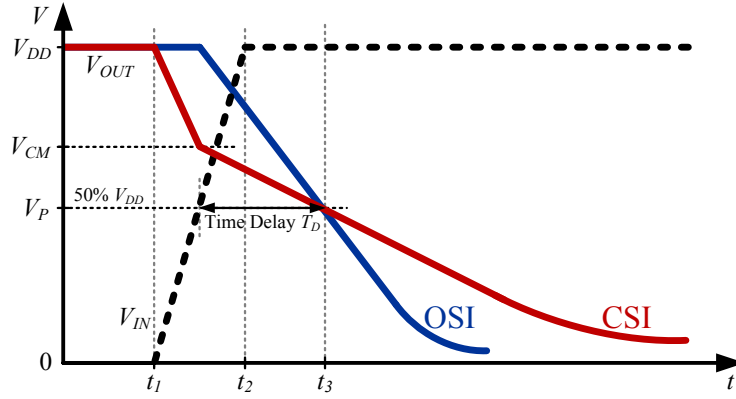


Figure 4.12. The transient state of the CSI and OSI circuit models generating the delay time interval  $T_D$  measured at 50% input and output signal level and assuming the discharge of the respective capacitances by the  $i_D$  current.

For the purpose of this analysis, the discharge phase is assumed to begin when the input voltage crosses 50% of  $V_{DD}$  and to terminate (generating time delay  $T_D$ ) when the output voltage crosses the same threshold (the capacitances continue to discharge to 0 V after that). The discharge phase of the current starved inverter (CSI) delay gate starts for the output voltage  $V_{OUT} = V_{CM} = V_{DD} \times C_1 / (C_1 + C_2)$ . Assuming the ideal operation of the switches, the generated delay time  $T_D = T_{CSI}$  depends only on the current  $i_D$  discharging the capacitances  $C_1 + C_2$  from the initial voltage  $V_{CM}$  to  $V_P$  (typically  $V_P = V_{DD}/2$ , terminating the generated time interval  $T_D$ , Figure 4.12) and is given by the formula:

$$T_{CSI} = (C_1 + C_2) \frac{V_{CM} - V_P}{i_D} \quad (4.5)$$

In the case of the proposed OSI circuit, the discharge phase always starts for the output voltage  $V_{OUT} = V_{DD}$  and terminates when the current  $i_D$  discharges the capacitance  $C_1$  down to  $V_P$  (the capacitance  $C_2$  is already shorted to the ground and does not participate in the discharge phase). The generated delay time  $T_D = T_{OSI}$  equals:

$$T_{OSI} = C_1 \frac{V_{DD} - V_P}{i_D} \quad (4.6)$$

One of the advantages of the proposed OSI circuit is its capability of generating longer delay time intervals  $T_D$  for the same bias conditions. This is mainly caused by the fact that  $C_1$  is usually larger than  $C_2$  due to the additional load of the node  $V_{OUT}$ , and the voltage  $V_{CM}$  is usually lower than  $V_{DD}$ . In particular, inserting  $V_{CM} = V_{DD} \times C_1 / (C_1 + C_2)$  to the equation (4.5) and  $V_P = V_{DD}/2$  to (4.5) and (4.6), and assuming  $V_{CM} \geq V_P$  (to ensure

discharging of the load capacitance from  $V_{CM}$  to  $V_P$ ), the ratio  $T_{CSI}/T_{OSI}$  will simplify to the following formula:

$$\frac{T_{CSI}}{T_{OSI}} = \frac{C_1 - C_2}{C_1} \quad (4.7)$$

It is important to note that equation (4.7) was derived assuming  $C_1 > C_2$  (which is usually the case due to the additional load capacitance) and switching at 50% of the maximum signal level ( $V_P = V_{DD}/2$ ). It can be observed that, in such case, the OSI gate will generate a longer delay time interval ( $T_{OSI} > T_{CSI}$ ). The proposed charge sharing approach can be further extended to account for other systematic effects affecting  $V_{CM}$  voltage. For example, for a high coupling between the input and output nodes (due to e.g. high gate-source and gate-drain capacitances in MOS transistors) the additional charge injected to nodes  $V_{OUT}$  and  $V_{SD}$  from the input may increase the  $V_{CM}$  voltage which, from the perspective of the proposed model, can be seen as an increase of the capacitance  $C_1$  in equation (4.7). Assuming charge sharing between  $C_1$  and  $C_2$  any other factor affecting the charge stored on these capacitances can be theoretically accounted for by modifying the value of  $C_1$  or  $C_2$ . In the extreme case, when the charge injected to the output node causes  $V_{CM} \approx V_{DD}$ , the operation of both CSI and OSI delay gates will become similar due to the fact that both gates will start the discharge phase for the same initial condition  $V_{OUT} = V_{DD}$ .

#### 4.4.5.1 Mismatch model of a delay gate

The variability of the generated time delay  $T_D$  of the CSI and OSI circuits can be estimated by applying the equation (2.10) from Chapter 2 to the calculated delays  $T_{CSI}$  and  $T_{OSI}$ . For both circuits it is assumed that the variability of the delay time  $T_D$  results mainly from the variability of the parameters of the current limiting transistor  $M_D$  and the capacitive load of the output node, therefore only the variability of the current  $i_D$  and capacitances  $C_1$  and  $C_2$  will be accounted for in the following calculations. Despite its limitations, the proposed approach covers all major contributors to the  $T_D$  time variability including all MOS transistors (e.g. the capacitive load of the next stage in a delay line will depend on the variability of  $M_1$  and  $M_2$ ) and the interconnecting tracks. The normalized delay variances derived for the CSI and OSI circuits (equations (4.5) and (4.6)) assuming  $V_{CM} = V_{DD} \times C_1 / (C_1 + C_2)$ ,  $V_P = V_{DD}/2$  and  $V_{CM} \geq V_P$  are equal to:

$$\frac{\sigma_{T_{CSI}}^2}{T_{CSI}^2} = \frac{\sigma_{I_D}^2}{i_D^2} + \frac{\sigma_{C_1}^2 + \sigma_{C_2}^2}{(C_1 - C_2)^2} \quad (4.8)$$

$$\frac{\sigma_{T_{OSI}}^2}{T_{OSI}^2} = \frac{\sigma_{I_D}^2}{i_D^2} + \frac{\sigma_{C_1}^2}{C_1^2} \quad (4.9)$$

where  $\sigma_{I_D}^2$ ,  $\sigma_{C_1}^2$  and  $\sigma_{C_2}^2$  are the variances of the current  $i_D$  and the capacitances  $C_1$  and  $C_2$ . Assuming the simplest square law model of the transistor  $M_D$  operating in strong inversion and saturation, the drain current  $I_D$ , and its relative variability  $\sigma_{I_D}^2/I_D^2$  caused by the variability of the threshold voltage  $\sigma_{V_{th}}$  and the transconductance  $\sigma_\beta$ , are given by equations (see equations 2.1 and 2.15 in Chapter 2 for reference):

$$i_D = \frac{\beta}{2}(u_{GS} - V_{th})^2 \quad (4.10)$$

$$\frac{\sigma_{\Delta I_D}^2}{I_D^2} = \frac{\sigma_{\Delta\beta}^2}{\beta^2} + \frac{4\sigma_{\Delta V_{th}}^2}{(u_{GS} - V_{th})^2} \quad (4.11)$$

From (4.5), (4.6), (4.8), (4.9), (4.10) and (4.11) the following equations can be obtained:

$$\frac{\sigma_{T_{CSI}}^2}{T_{CSI}^2} = \frac{4\beta\sigma_{V_{th}}^2 T_{CSI}}{V_{DD}(C_1 - C_2)} + \frac{\sigma_\beta^2}{\beta^2} + \frac{\sigma_{C_1}^2 + \sigma_{C_2}^2}{(C_1 - C_2)^2} \quad (4.12)$$

$$\frac{\sigma_{T_{OSI}}^2}{T_{OSI}^2} = \frac{4\beta\sigma_{V_{th}}^2 T_{CSI}}{V_{DD}C_1} + \frac{\sigma_\beta^2}{\beta^2} + \frac{\sigma_{C_1}^2}{C_1^2} \quad (4.13)$$

Equations (4.12) and (4.13) show a linear dependence of the normalised delay variance in terms of the generated delay time  $T_D$ . For a typical circuit implementation (when  $C_1 > C_2$ ) the slope and the  $y$ -intercept component in (4.12) is greater than the respective one in (4.13), due to the dependency of the delay time of the CSI circuit on both  $C_1$  and  $C_2$ . As a result, the delay variance of the CSI gate is higher than the one of the OSI gate for the same generated delay. It can be concluded that the OSI gate generates delay time intervals that are longer, as shown in equation (4.7), and less affected by parameter variability (equations 4.12 and 4.13). It is important to note that approximately linear dependency of the normalised delay variance on the generated delay can be observed in practice (Figure 4.20), for short delays, when the current limiting transistors operate in strong inversion.

#### 4.4.5.2 Mismatch model of a delay line

The proposed analysis can also be applied to designs of delay lines where a certain number of delay gates are connected in series creating a chain. For  $N$  gates with delay  $T_D$  each connected in a chain, the delay of the entire line is equal to  $T_N = NT_D$ , i.e. it will increase linearly with the number of stages. Assuming that delays  $T_D$  generated by different stages are normally distributed and independent random variables with variance  $\sigma_{TD}^2$ , the total delay variance of a line will be equal to  $\sigma_{TN}^2 = N\sigma_{TD}^2$ , and hence the normalized delay variance of the line consisting of  $N$  such stages can be calculated as:

$$\frac{\sigma_{TN}^2}{T_N^2} = \frac{1}{N} \frac{\sigma_{TD}^2}{T_D^2} \quad (4.14)$$

It should be noted that, in some applications, the symmetric delay gates with two current limiting transistors on both pull-up and pull-down sides may be used [Bolt 96], [Cantatore 97]. Their operation is practically the same as the operation of the discussed asymmetric circuits but the delaying of the output signal occurs on both falling and rising slopes controlled either by transistor  $M_{DN}$  or  $M_{DP}$  (e.g. for the falling output slope transistor  $M_{DN}$  controls the delay time whereas  $M_{DP}$  does not participate in the discharge phase). Therefore, the analysis of the asymmetric circuit, presented in this thesis, applies also for the symmetric version used in the realisations of the delay lines on the test chip.

#### 4.4.5.3 Mismatch optimisation

The mismatch optimisation technique based on scaling of the current limiting transistor assumes that the variability of the generated delay time intervals decreases with the increase of the gate area  $WL$  of  $M_D$ . In general, this will reduce the variability of  $\beta$ ,  $V_{th}$ ,  $C_1$  and  $C_2$  in equations (4.12) and (4.13), but will also affect the value of the generated delay  $T_D$ . Assuming only the variability in threshold voltage  $\sigma_{V_{th}}^2 = A_{V_{th}}^2 / 2WL$  and  $\beta = \mu C_{ox} W / L$ , the normalised variance of the generated delay time  $T_D$  in 4.12 and 4.13 can be written in a simplified form:

$$\frac{\sigma_{TD}^2}{T_D^2} = \frac{2T_D}{V_{DD}C_L} \frac{\mu C_{ox} A_{V_{th}}^2}{L^2} \quad (4.15)$$

where  $C_L$  is a load capacitance representing  $C_1$  and  $C_2$  from equations (4.12) and (5.13). Based on the equation (4.15), it can be observed that the impact of the threshold voltage variability is only dependent on the length  $L$  of the current limiting transistor  $M_D$  for a

constant delay  $T_D$ . This is because enlarging of the area of  $M_D$  by increasing  $W$ , requires also proper adjustment of the bias voltage  $V_D$  to assure constant delay time  $T_D$ . For example, increasing  $W$  by the factor of  $n^2$  will increase the discharge current  $n^2$  times. In order to reduce this current to assure the same delay time  $T_D$ , the corresponding bias voltage  $V_D$  has to be reduced  $n$  times (see equation (4.10)). This, in turn, will increase the variability of the drain current caused by mismatch of the threshold voltage  $n^2$  times (see equation (4.11)). As a result, the obtained mismatch reduction, caused by the area increase of  $M_D$ , will be compensated by the same increase of the variability in the drain current, caused by the reduced bias voltage. Therefore, scaling of the current limiting transistor by increasing the channel length rather than width should be considered [Cantatore 97].

## 4.5 Chip design and circuit implementation

In order to compare the operation and statistical parameters of the CSI and OSI delay gates, two arrays of delay lines employing these structures were fabricated in a standard 90 nm CMOS technology. The architecture of the test system, implemented on the chip, including the arrays of 512 CSI and OSI delay lines and the additional control logic, is shown in Figure 4.13 and the circuit layout in Figure 4.14.

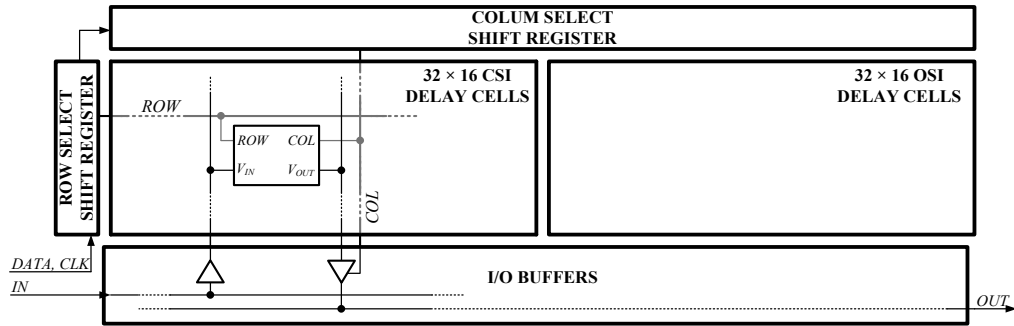
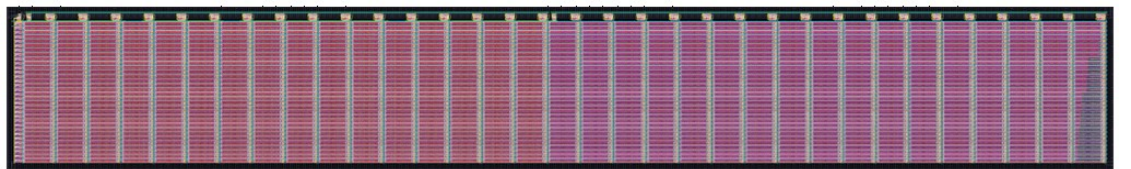


Figure 4.13. The test system with CSI and OSI delay line arrays implemented on the chip.



Array of  $32 \times 16$  CSI delay lines

Array of  $32 \times 16$  OSI delay lines + additional column with lines of different lengths (four lines of: 16, 15, 8, 7, 4, 3, 2 and 1 gate)

Figure 4.14. Full layout of the test system (area:  $160 \mu\text{m} \times 1140 \mu\text{m}$ ).

Each array consists of  $32 \times 16$  delay cells selected using the row/column addressing maintained by the boundary shift register on the left and top sides of the arrays. The schematic diagram of the delay cell including a 16-stage delay line and additional control and I/O logic is presented in Figure 4.15.

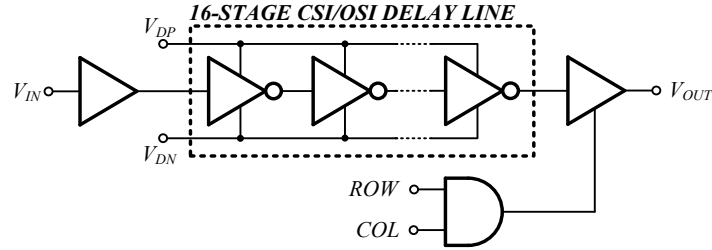


Figure 4.15. The schematic diagram of the delay cell including the 16-stage CSI/OSI delay line, I/O buffers and the AND gate used in the row/column addressing.

A particular cell from any of the arrays can be selected by shifting the programming sequence into the register using signals *DATA* and *CLK* which sets the appropriate column and row lines to the high logic state. An additional AND gate, implemented in each delay cell, will detect this condition and connect the output of the delay line to the output line through a tri-state buffer. In order to reduce the capacitive load of this buffer, the output line is shared only for the lines from the same column. The *COL* signal is then used to enable another tri-state buffer in the I/O block, which connects the selected column to the global output. The input signal is buffered at the input of each delay cell and also individually for each column in the I/O block. In order to assure the uniform propagation times of the input and output signals, the same numbers of buffers were used for each delay line irrespective of its position in the array. Each delay cell includes a 16-stage delay line implemented using the symmetric variant of the CSI or OSI delay gate, presented in Figure 4.16.

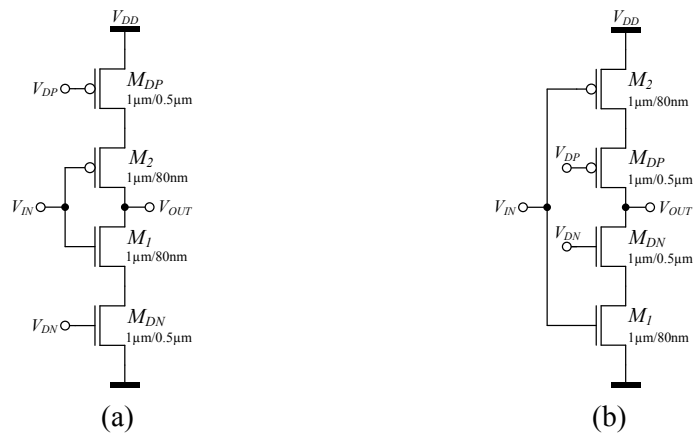


Figure 4.16. Schematic diagrams of the delay gates with two complementary current limiting transistors  $M_{DN}$  and  $M_{DP}$ : a) CSI variant, b) OSI variant.

In this implementation there are two current limiting transistors  $M_{DN}$  and  $M_{DP}$  of the size  $W/L = 1\mu\text{m}/0.5\mu\text{m}$  controlled by the bias voltages  $V_{DN}$  and  $V_{DP}$  respectively. The width of the switching transistors  $M_{I,2}$  is the same as the current limiting ones to assure compact layout structure where both transistors share the same diffusion stripe. The layouts of the designed cells with delay lines consisting of 16 CSI and OSI gates used in the experiments are presented in Figure 4.17. The chip micrograph showing the region where the delay line arrays were implemented is presented in Figure 4.18. The area of the test system is  $160\mu\text{m} \times 1140\mu\text{m}$ .

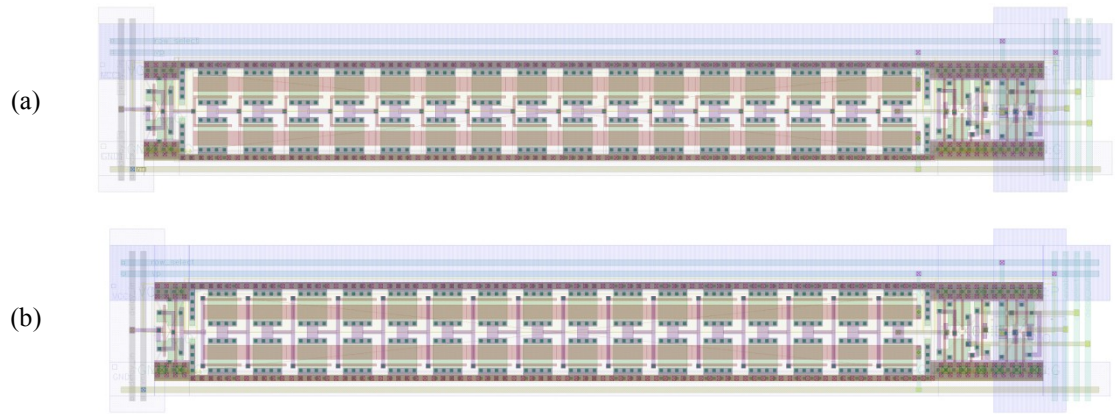


Figure 4.17. Layouts of the delay cells with lines consisting of 16 serially connected a) CSI gates and b) OSI gates (the size of each delay line is  $3.7\mu\text{m} \times 27\mu\text{m}$ ).

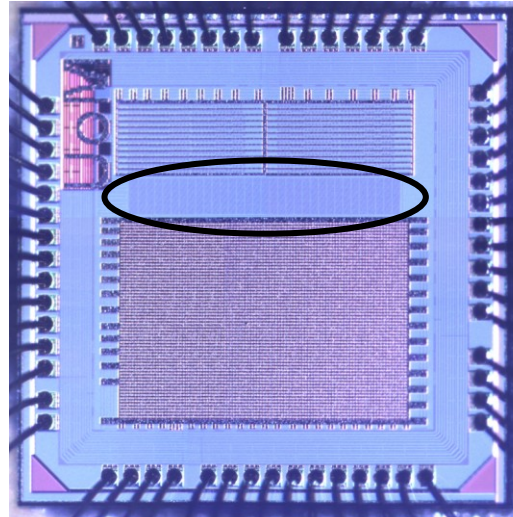


Figure 4.18. Chip micrograph showing the location the delay cell arrays.

## 4.6 Experimental results

The measurement results of the fabricated chip were obtained in a laboratory environment using an Agilent 54641D oscilloscope and a KCPSM3 (Xilinx PicoBlaze) controller implemented on a Spartan 3 FPGA development board. The block diagram showing the setup used in the experiments is presented in Figure 4.19. The KCPSM3 system communicates with a PC application (e.g. Matlab) via an RS-232 serial port, executes received commands and, based on that, provides communication with the chip by programming the boundary shift register and generating the square wave signal of 5  $\mu$ s period to drive a selected delay line. The delay time was measured on rising edges from 50% to 50% of the fixed level representing high logic state of input and output signals. The data acquisition setup of the oscilloscope assumed full bandwidth and averaging based on 64 samples to suppress the time jitter. Both the KCPSM3 system and the oscilloscope were working in a loop controlled by a Matlab script selecting delay lines in turn and collecting the measured delay times.

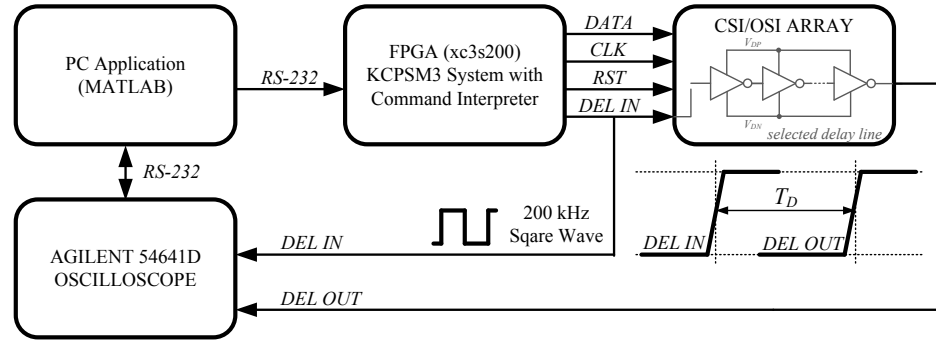


Figure 4.19. Block diagram of the setup used for the delay time measurements.

### 4.6.1 Calibration

In the test system a particular delay line can be tested by programming the boundary shift register with a sequence addressing the corresponding cell in the array. The output signal of the selected delay line goes through two tri-state buffers (one in the cell and one in the I/O block) output buffer, the digital cell in the I/O ring including additional buffers and level shifters (to match the external and core logic standards) and through a buffer driving the capacitance of the test probe. Similarly the input signal is provided through the digital I/O cell and a series of buffers until it reaches the selected delay line. Due to the fact that it is not possible to measure the generated delays  $T_D$  directly at the input and output nodes of the selected line, the delay time  $T_{MEAS}$  measured between the input and

output slopes of the signals outside the chip will include additional offset time  $T_{OFF}$  introduced by the buffers between the test points on the PCB, such that  $T_{MEAS} = T_D + T_{OFF}$ . The offset time also slightly depends on the location of a particular delay line in the array due to different lengths of I/O paths. This variability, however, is only a small fraction of the entire length of the path between the array and the probes on the PCB, therefore it can be neglected in this analysis. In order to estimate the offset time  $T_{OFF}$ , an additional column with OSI delay lines with different number of stages was implemented. Based on the measurement results of the delays  $T_{MEAS}$  for four different OSI 16-stage and 1-stage delay lines for the bias voltages  $V_{DN} = V_{DD}$  and  $V_{DP} = 0$ , the obtained mean values are  $T_{MEAS-16} = 11.73$  ns (for 16-stage lines) and  $T_{MEAS-1} = 9.50$  ns (for 1-stage lines). Assuming that the 16-stage line generates delays 16 times longer than the 1-stage one, the delay  $T_D$  of a single gate and  $T_{OFF}$  were calculated using the following relations:

$$T_{MEAS-16} = 16T_D + T_{OFF} \quad (4.16)$$

$$T_{MEAS-1} = T_D + T_{OFF} \quad (4.17)$$

The obtained values of the delay and offset time were  $T_D = 0.149$  ns and  $T_{OFF} = 9.35$  ns respectively. It should be noted that the input driver and the output buffer of the delay line slightly affect the propagation times  $T_D$  of the first and last delay stage. The impact of the delay time variability  $\Delta T_D/T_D$  on the accuracy of the offset time estimation can be calculated from:

$$T_{OFF}(T_D) = T_{MEAS-1} - T_D \quad (4.18)$$

Developing (4.18) into Taylor series for the first two elements gives:

$$T_{OFF}(T_D + \Delta T_D) = T_{MEAS-1} - T_D - \Delta T_D \quad (4.19)$$

Inserting (4.18) into (4.19) and dividing both sides of (4.19) by  $T_{OFF}$  the following relation can be derived:

$$\frac{\Delta T_{OFF}}{T_{OFF}} = -\frac{\Delta T_D/T_D}{T_{MEAS-1}/T_D - 1} \quad (4.20)$$

Inserting the obtained values  $T_D = 0.149$  ns and  $T_{MEAS-1} = 9.50$  ns into (4.20), the relation between the relative variability of  $T_D$  and  $T_{OFF}$  is given by the approximation:

$$\frac{\Delta T_{OFF}}{T_{OFF}} \approx -\frac{1}{63} \frac{\Delta T_D}{T_D} \quad (4.21)$$

From the equation (4.21) it can be seen that the relative variability of the measured time delay  $T_D$  has negligible impact on the accuracy of the offset time estimation. For example  $\pm 50\%$  variability of  $T_D$  will affect the precision of the  $T_{OFF}$  estimation by less than  $\pm 1\%$ . Therefore, the impact of the non uniform delay time  $T_D$  along the line caused by the input and output buffers can practically be neglected in the calculations of the offset time. The obtained delay time offset  $T_{OFF} = 9.35$  ns was subtracted from the raw data obtained from the measurements prior to any further statistical computations and analyses.

#### 4.6.2 Normalised delay variance

In order to compare the performance of the CSI and OSI delay lines, the normalized delay variance will be calculated based on the measurement result obtained from the entire array containing 512 CSI and 512 OSI delay lines for symmetric bias voltages  $V_{DN} = 150 \dots 430$  mV and  $V_{DP} = 850 \dots 570$  mV ( $V_{DP} = V_{DD} - V_{DN}$ ). The core supply voltage of the chip is  $V_{DD} = 1.0$  V. The diagram showing the normalized delay variance as a measure of the relative time variability versus mean delay time, computed based on the obtained results of the CSI and OSI delay gates accounting for the offset time  $T_{OFF} = 9.35$  ns, is presented in Figure 4.20.

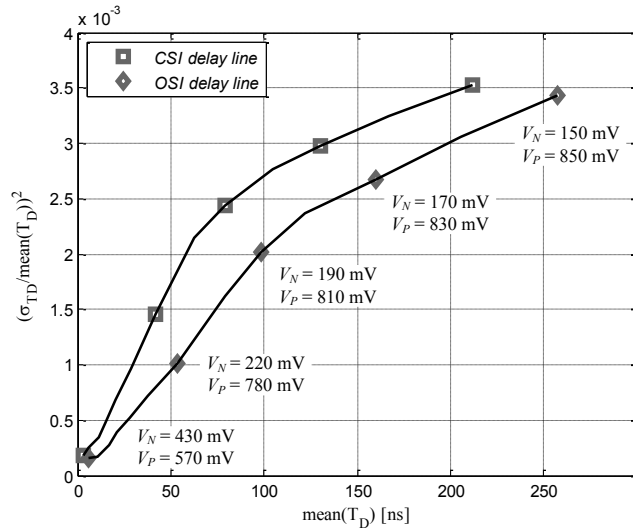


Figure 4.20. Normalized delay variance vs. mean delay time obtained from the measurements of the CSI and OSI delay lines.

The traces of the normalised delay variances of the CSI and OSI circuits, presented in Figure 4.20, can be divided in two sections depending on the operation region of the current limiting transistors  $M_{DN}$  and  $M_{DP}$  during the discharge phase. In the first section,

for the measured delays  $T_D$  below 100 ns, the current limiting transistors operate in strong inversion and the relation of the normalised delay variance in terms of mean delay is linear, as suggested by the proposed model in equations (4.12) and (4.13). For higher delays, transistors  $M_{DN}$  and  $M_{DP}$  enter subthreshold region where the drain current variability, and hence, the normalised delay variance, given by equations (4.6) and (4.7), is constant and independent on the bias voltages (see equation (2.29) in Chapter 2). However, the traces shown in Figure 4.20 obtained from measurements do not level out immediately but slightly bend towards the horizontal axis. Reasons for the observed phenomenon are mainly the operation of the MOS transistors in moderate inversion for bias around the threshold voltage.

In the CSI and OSI circuits, the generated delay time  $T_D$  increases when lowering the bias voltage  $V_{DN}$  or increasing the voltage  $V_{DP}$ . From equations (4.12) and (4.13), the variability of this current depends on the operating point of the transistor, and increases with the generated delay time, also increasing the corresponding relative time variability. This means that the precision of the circuit array degrades for longer delays. Therefore, the tuning range of the fabricated test arrays, will most probably be restricted to 20 - 30 ns, where the current limiting transistors operate in strong inversion. Practically, in order to generate longer delays of the same precision, longer delay lines should be used. The normalized delay variance versus mean delay time limited to the 30 ns tuning range, is shown in Figure 4.21.

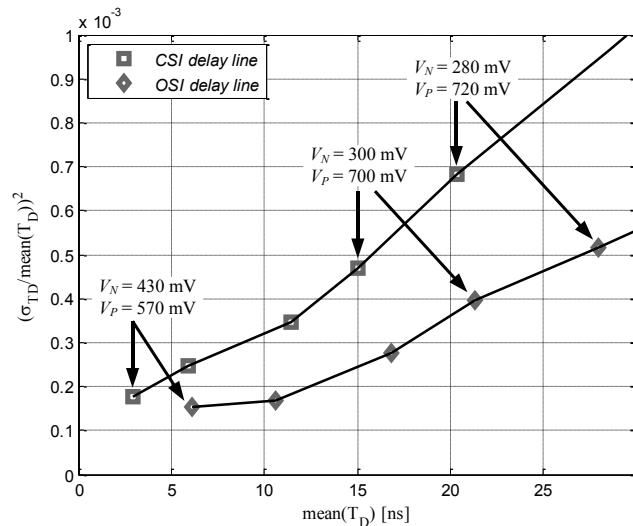


Figure 4.21. Normalized delay variance vs. mean delay time obtained from the measurements for the tuning range limited to 30 ns.

The visual representation of the generated delay times across the CSI and OSI arrays on the same gray scale map is shown in Figure 4.22. The corresponding histograms of the generated delays are presented in Figure 4.23. Both arrays were tuned to generate delay time intervals of approximately 11 ns. In the experiment it was difficult to tune both arrays precisely and the mean values extracted from the measurements of the delay times were  $T_{DCSI} = 11.463$  ns and  $T_{DOSI} = 10.633$  ns for the CSI and OSI arrays respectively. To facilitate comparison, the difference of  $T_{DCSI} - T_{DOSI}$  was added to the results obtained from the OSI array to align both histograms and set the range of the gray scale map to cover the corner cases from both data sets. In the map, black colour represents the fastest line of  $T_{DMIN} = 10.885$  ns delay, and white colour represents the slowest line of  $T_{DMAX} = 12.070$  ns delay (after the alignment). It can be observed that the image representing delay time in the OSI array has lower contrast and looks more uniform in comparison to the image obtained from the CSI array. Also, the corresponding distribution of the generated delays is narrower for the OSI array indicating less effect of the physical parameter variability on the circuit performance.

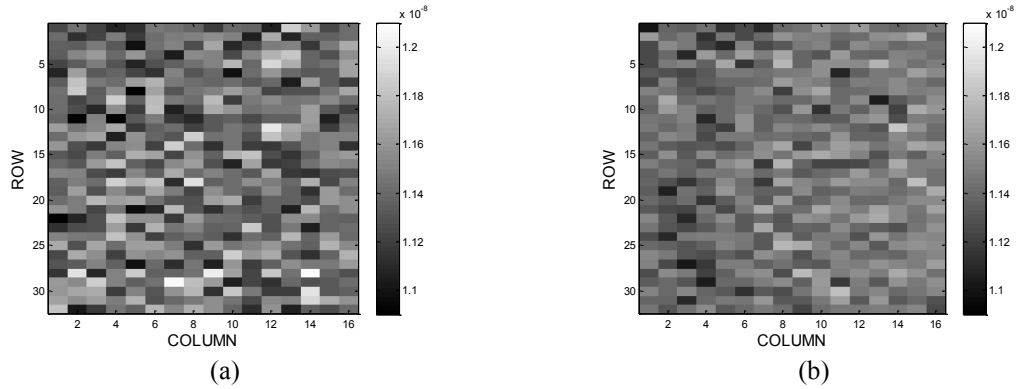


Figure 4.22. Visual representation of the delay time variability on the same gray scale measured for a:) CSI array and b) OSI array (aligned results, see text for details, ROW and COLUMN correspond to the physical location on the chip).

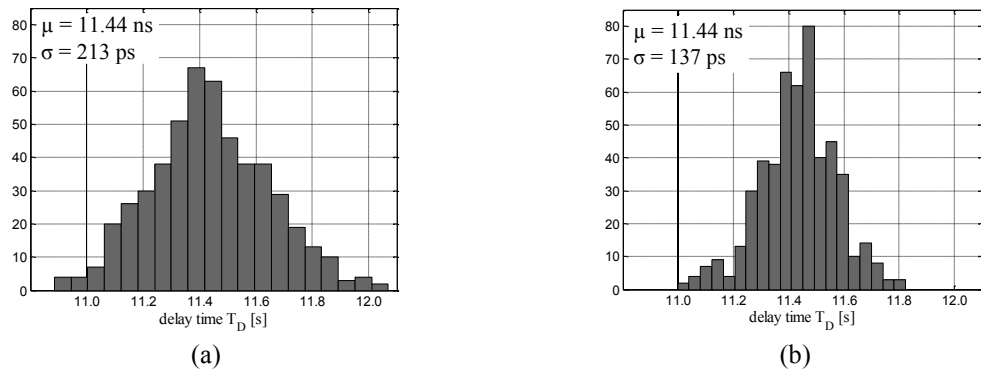


Figure 4.23. Histograms of the generated delay time distribution for: a:) CSI array and b) OSI array (aligned results, see text for details).

In the experiments, four other chips from the same fabrication run were tested. The results presented above were obtained from the measurements of chip (#1). The measured statistical parameters of the CIS and OSI delay lines (tuned to generate delays around 11 ns) for chips #1-#5 are presented in Table 4.4. For all the chips (#1 - #5) the offset time  $T_{OFF}$  was measured individually. The ratios of the standard deviation to the generated mean delay time of the CSI and OSI delay lines in each chip for fixed delays 20, 30, 40, 50 ns are presented in Table 4.5.

Table 4.4. Measurement results of five different chips from the same batch.

<i>Chip</i>	$T_{OFF}$ [ns]	<i>CSI</i>		<i>OSI</i>	
		$T_{DMEAN}$ [ns]	$\sigma_{TD}$ [ps]	$T_{DMEAN}$ [ns]	$\sigma_{TD}$ [ps]
#1	9.35	10.63	203	11.46	154
#2	9.15	11.05	231	11.70	151
#3	9.24	10.91	216	11.01	151
#4	9.20	11.16	224	11.31	169
#5	9.28	11.37	220	11.45	155

Table 4.5. Delay time variability measured for five different chips from the same batch.

$T_{DMEAN}$	<i>Chip #1</i>		<i>Chip #2</i>		<i>Chip #3</i>		<i>Chip #4</i>		<i>Chip #5</i>	
	$\sigma_{TD}/T_D$ [%]		$\sigma_{TD}/T_D$ [%]		$\sigma_{TD}/T_D$ [%]		$\sigma_{TD}/T_D$ [%]		$\sigma_{TD}/T_D$ [%]	
	<i>CSI</i>	<i>OSI</i>	<i>CSI</i>	<i>OSI</i>	<i>CSI</i>	<i>OSI</i>	<i>CSI</i>	<i>OSI</i>	<i>CSI</i>	<i>OSI</i>
20 ns	2.60	1.90	2.75	1.80	2.70	1.80	2.80	2.10	2.75	1.90
30 ns	3.33	2.43	3.43	2.37	3.47	2.30	3.57	2.63	3.47	2.47
40 ns	3.75	2.75	3.90	2.73	4.03	2.70	4.13	2.98	4.00	2.83
50 ns	4.42	3.08	4.24	3.02	4.48	3.00	4.58	3.32	4.46	3.18

### 4.6.3 Time jitter

Apart from the fabrication mismatch, the generated delay time intervals are also affected by noise (from the power supply, bias voltages and thermal effects in the circuit), which causes random variation of the output signal slope (jitter). In previous experiments jitter was removed by applying averaging over 64 samples, which was sufficient to obtain stable delay measurement readouts. The influence of noise on the generated delay time was calculated based on 1000 measurements of the delay of a particular line working under constant bias conditions. The measurement results showing the calculated standard deviation of the delay times as a result of fabrication mismatch and jitter within 50 ns tuning range are presented in Figure 4.24. For the time jitter estimation the standard deviation and the mean value of the generated delay time was measured for the fastest and the slowest delay line in both CSI and OSI arrays, and the respective average values were calculated. The obtained results show that the delay time

variability is dominated by the fabrication mismatch and remains almost one order of magnitude higher than the measured time jitter. It was observed that the measured time jitter was also affected by the noise of the setup (oscilloscope, probes and IO buffers), introducing a baseline noise level of  $\sigma_{TD} = 30$  ps for delay  $T_D = 3.44$  ns measured in the system without the chip and with the corresponding input and output pins shorted in the socket on the PCB. The same measurement repeated with averaging over 64 samples (set up in the oscilloscope) reduced the baseline noise level to  $\sigma_{TD} = 4.8$  ps. Therefore, the measurements of the time delay variability caused by mismatch (done with averaging over 64 samples) are practically not affected by the noise but the time jitter measurements in Figure 4.24 can, to some extent, be overestimated due to the baseline noise level of the setup and the limited sampling rate of the oscilloscope (1 GSa / s in the dual channel mode).

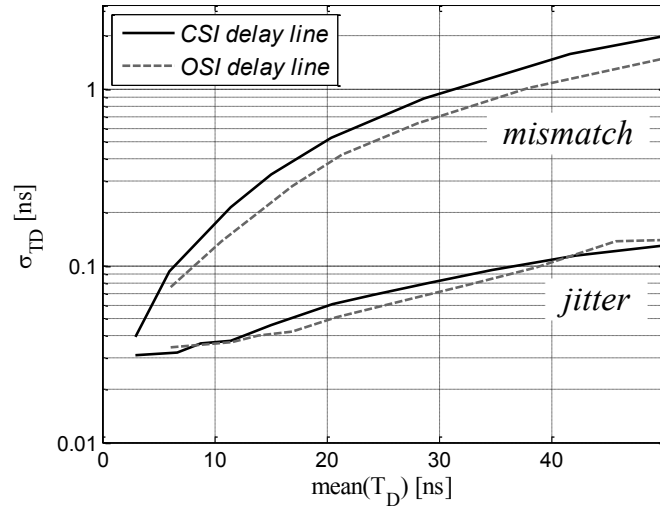


Figure 4.24. Generated time variability caused by the fabrication mismatch and noise (time jitter) measured for CSI and OSI arrays within 50 ns tuning range.

#### 4.6.4 Simulations versus measurements

The operation of the CSI and OSI delay lines implemented on the test chip was verified in simulations using their circuit models extracted from the layouts, accounting for the parasitic resistances and capacitances. The schematic diagram of the test circuit used in the simulations is presented in Figure 4.25. The input signal was buffered using two inverters. The output of the delay line was driving additional dummy inverter providing a capacitive load. The generated delay time  $T_D$ , for particular bias voltages  $V_{DN}$  and  $V_{DP}$ , was measured between the input and the output of the delay line. In the simulations, statistical MOS transistor models provided by the foundry were used with

switches  $MISMATCH = 1$  and  $PROCESS = 0$ , accounting only for the parameter mismatch, not including batch to batch process variability. In the simulations, 200 Monte Carlo runs were performed for a each bias voltage pair  $V_{DN}$  and  $V_{DP}$ , covering the entire tuning range, and repeated for process corners: *typical-typical* (TT), *fast-fast* (FF) and *slow-slow* (SS). The obtained results, in comparison with the measurements, are presented in Figures 4.26 and 4.27.

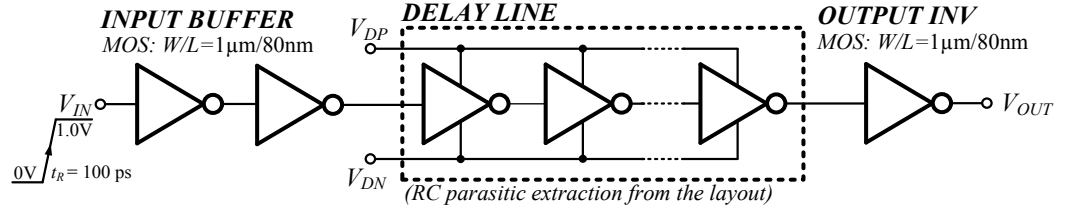


Figure 4.25. Schematic diagram of the test circuit used in the simulations of the delay lines.

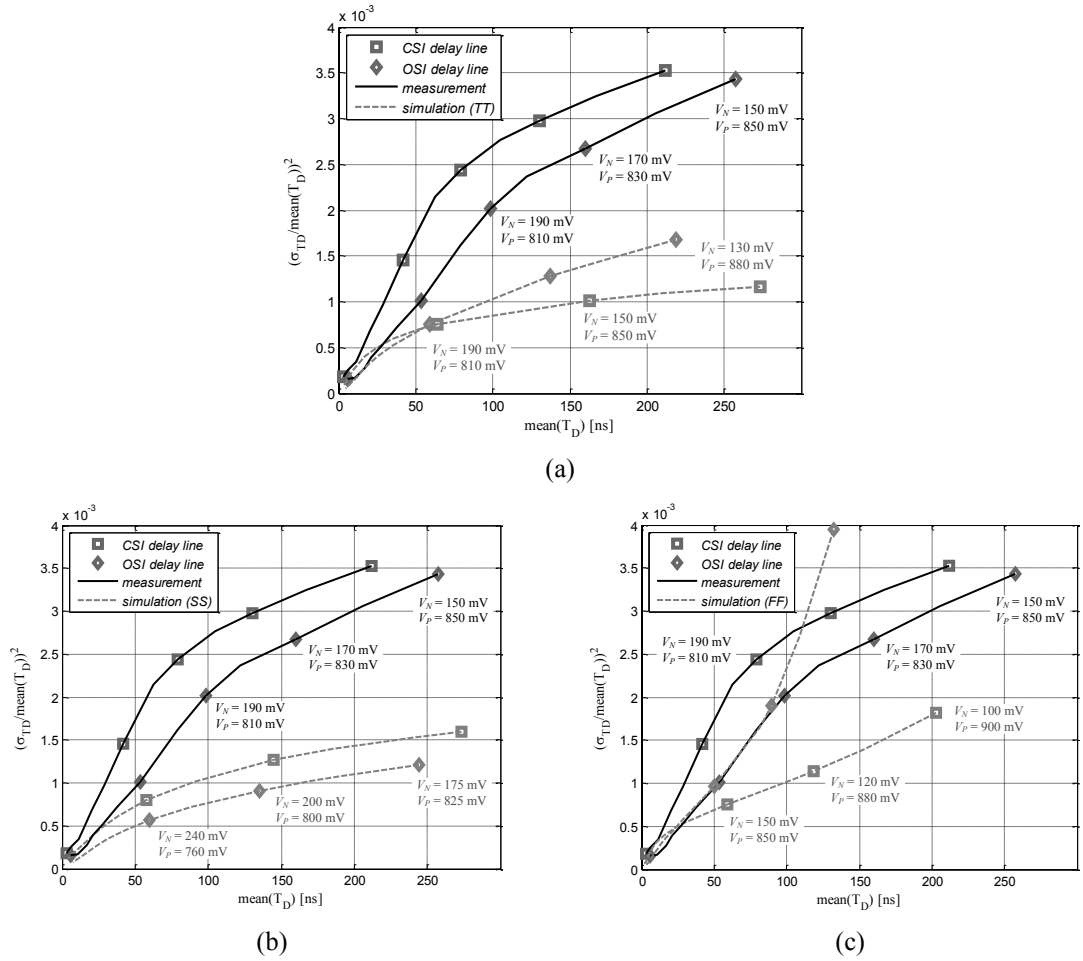


Figure 4.26. The normalised delay variance of the CSI and OSI delay lines versus mean delay time obtained from the measurements and simulations assuming mismatch MOS transistor models operating in a) TT process corner, b) SS process corner and c) FF process corner.

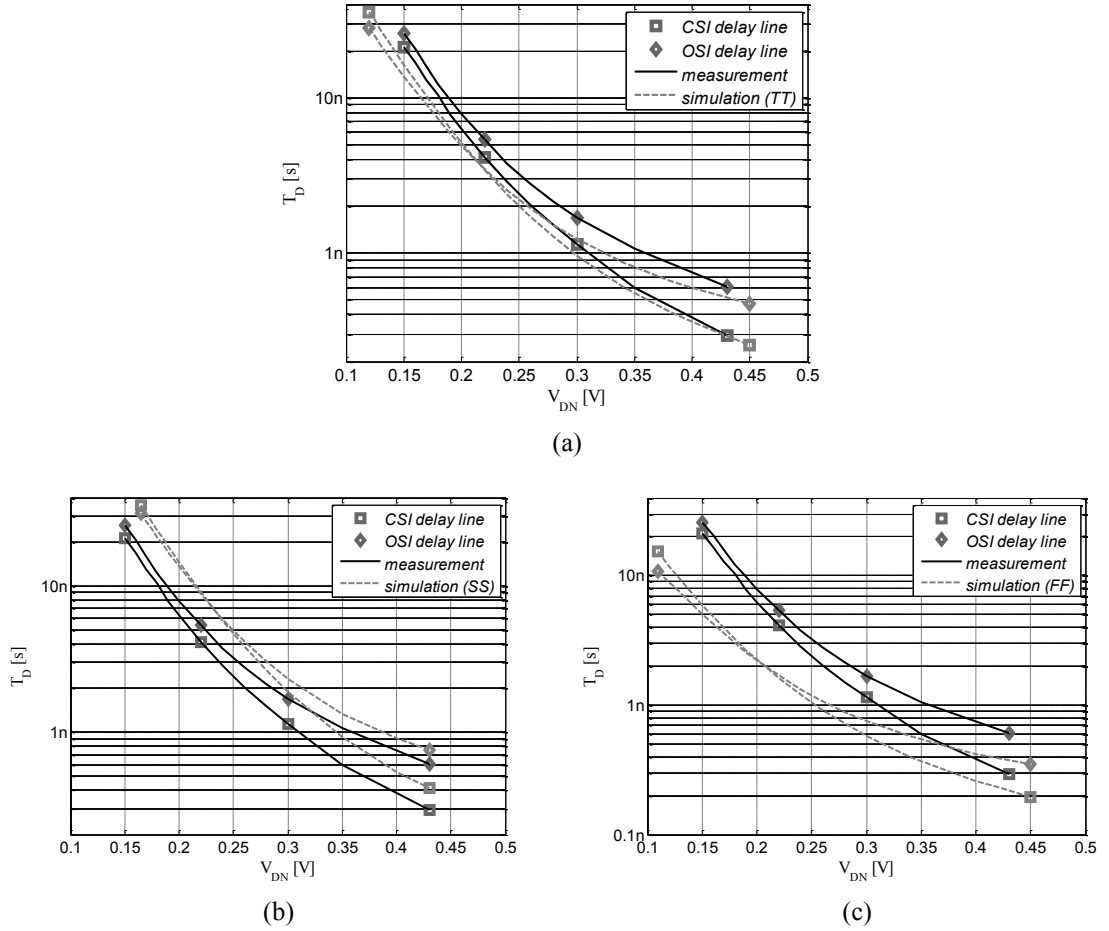


Figure 4.27. Tuning range of the CSI and OSI delay lines obtained from the measurements and simulations assuming mismatch MOS transistor models operating in a) TT process corner, b) SS process corner and c) FF process corner.

It can be observed that the measurement results, in terms of the tuning range, are the closest to the simulations obtained for the typical (TT) process corner (Figure 4.27a). Mismatch results, however, show higher disparities between the simulations and the measurements. There are several potential reasons for the observed differences, including limited model precision, distance dependent variability and parasitics.

Mismatch Monte Carlo models account only for the local parameter variability in the adjacent MOS devices. For larger circuits, such as the test array of delay lines, also distance dependent variability component, predicted by Pelgrom's model, could be seen as a potential contributor of the increased mismatch. Nevertheless, its impact on the circuit performance has already been shown rather insignificant for standard CMOS technologies (Chapter 2).

For simulations, parameters of BSIM4.3.0 MOS transistor model [Xi 2003], provided by the foundry, were used. However, this model is not accurate in the

subthreshold region. It can be observed that matching between simulation and measurement results improves for larger bias voltages  $V_N$ , when the current limiting transistors operate in strong inversion. It should also be noted that there were no up-to-date transistor models for this particular technology available, since the foundry is terminating fabrications in this node. The models used in the simulations were from May 2009 (the PPATC chip was fabricated almost four years later in January 2013).

Finally, simulations account only for the variability in the parameters of MOS transistor, whereas the parasitic  $RC$  components are fixed. In the fabricated circuit, the random variability of the parasitic components will additionally increase the variability of the generated delay time.

It is important to note that the situation results indicating better matching properties of the CSI gate than the proposed OSI circuit (see Figure 4.26a and c), were observed only when using the standard performance transistor models from the 90 nm design kit. One characteristic feature of these transistors is their very thin gate oxide ( $t_{ox} = 1.6$  nm), thinner than in other CMOS technologies, such as 65 nm ( $t_{ox} = 2.6$  nm) or 180 nm ( $t_{ox} = 3.3$  nm). Such thin gate oxide is used mainly to manufacture fast transistors with applications in high speed digital systems. In the design of the CSI delay gate, it increases the gate capacitances of the switching transistors, and hence, the coupling between the input and output nodes. As a result, some part of the input charge is transferred to the output, charging the load capacitance above the supply voltage (the overshoot effect [Huang 2010]). Consequently, it elongates the discharge time by pulling voltage  $V_{CM}$  closer to  $V_{DD}$ . Since this effect is not significantly affected by the parameter variability, the generated delay intervals are longer and less variable. In the OSI circuit, however, the effect of overshooting is much lower due to the current limiting transistors separating the output node from the switching transistors, therefore, the overall performance of the CSI gate is seemingly better than the proposed OSI one.

The operation of the 16-stage CSI and OSI delay lines designed in standard 180 nm, 90 nm and 65 nm CMOS technologies was verified in simulations of the post layout models including  $RC$  parasitics, and using the test circuit presented in Figure 4.25. The size of the transistors in the delay gates was the same as shown in Figure 4.16. Only the channel length of the switching transistors  $M_1$  and  $M_2$  was defined by the minimum feature size of a particular technology. The ratio of the delay variance of the OSI and CSI lines  $(\sigma_{TDOSI}/\sigma_{TDCSI})^2$  versus mean delay time  $T_D$ , obtained from the simulations in three different technology nodes and measurements, in the same tuning range 10 ns - 200 ns is

presented in Figure 4.28. It can be observed that the operation of the proposed OSI structure is less affected by the fabrication mismatch. The delay variance ratio remains within interval of 60% - 75% for 65 nm technology and 50% - 90% for 180 nm and 90 nm technologies (both in simulations and measurements). Delay lines implemented in 180 nm and 65 nm nodes were simulated for TT corner and in 90 nm node for SS corner.

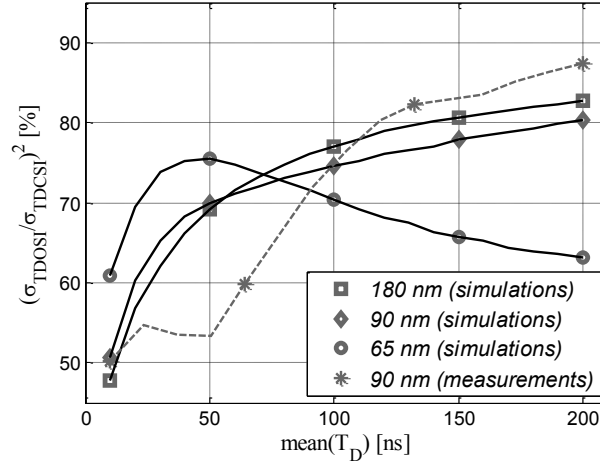


Figure 4.28. Ratio of the delay variances of the OSI and CSI circuits vs. mean delay time obtained from the post layout simulations of the circuits designed in 180 nm, 90 nm and 65 nm standard CMOS technologies and from the measurements.

## 4.7 Conclusions

The superior performance of the proposed OSI delay gate was achieved by inserting the current limiting transistors in between the switching transistors, unlike in the case of the CSI circuit, where the current regulating transistors are on the side of the power rails. Despite the similar operation of both designs, significant differences in their dynamic behaviour could be observed during the signal transitions, which are of a high importance when process parameter variability is concerned. The analyses and the simulation results were confirmed in measurements of 512 16-stage CSI and OSI delay lines implemented on a test chip and fabricated in a standard 90 nm CMOS technology. The experimental results have shown that the proposed OSI delay lines generate 10% - 50% less variable delay intervals than the CSI ones, with no penalty in terms of additional area, power or complexity increase. The proposed OSI structure could be considered in applications where multiple tunable delay elements of matched parameters are required, for example, in the build of readout systems for particle detectors, and in neuromorphic circuits. In this thesis, the proposed OSI structure will be used in the design of the asynchronous processor array for binary image skeletonization discussed in Chapter 5.

## **Chapter 5**

---

# **Asynchronous CMOS logic array for binary image processing**

---

### **5.1 Chapter overview**

This chapter presents a concept of the trigger-wave propagation in binary image processing using asynchronous cellular logic arrays in CMOS technologies. The idea of detecting collisions between wave-fronts is proposed as an extension to the propagation mechanism, with applications in fast object skeletonization and Voronoi diagram extraction. Discussions concerning hardware realisation and circuit design issues are supported by simulation analyses and experimental results, confirming the correct operation of the prototype array fabricated in a 90 nm CMOS technology.

### **5.2 Introduction**

#### **5.2.1 Bio-inspired approach**

Processing visual information in nervous systems, developed by vertebrates, can typically be divided into three stages accounting for low, medium and high level tasks. The earliest processing step is associated with chemical reactions that occur in rod and cone photodetector cells, creating the outer layer of a retina. Its inner layers perform low level spatial and temporal tasks, decomposing the received image into sets of features considering colour, brightness, shape, orientation, and movement, and hence, reducing the amount of data necessary for transmission and further processing. Such pre-processed

information is then transmitted via optic nerve to areas performing higher level cognitive functions in brain [Roska 2001]. Due to its efficiency and robustness, the structure and the processing flow of such visual system has been a subject of an extensive study and inspired the development of cameras, vision sensors and many image processing algorithms. In particular, in the field of VLSI circuit design, the idea of integration of the light sensor with processing elements, in the form of a monolithic electronic retina, gained high popularity. Such pixel-processor arrays are particularly suited for low and medium level image processing tasks, providing a very high processing efficiency. Ideally, their task is to process visual data directly on the focal plane, reducing the amount of primarily captured information to only a set of abstract descriptors transmitted off the chip [Bernard 1993].

### **5.2.2 SIMD paradigm**

Image processing tasks can be classified depending on the form, size and the complexity of the visual information taken as input and generated as a result. In such classification, low level tasks perform simple image processing operations, for example filtering, thresholding, edge detection, expansion and dilation, where the amount of input and the output information is practically the same. Medium level tasks are more application specific and attempt image interpretation, recognition and more advanced feature extraction [Fernandez-Berni 2011]. Their input is usually a pre-processed image of its original size (e.g. after thresholding and binarisation), whereas the output is only a set of abstract descriptors. In their subsequent routines, medium level tasks usually employ algorithms requiring global information of the image [Manzanera 2002]. High level tasks can be associated with the functionality of the visual cortex in brain, receiving and interpreting visual data from retina to perform more complex cognitive functions such as image reconstruction, understanding and correlation with other sensory data.

Although classical computers can be used in image processing, they usually require the use of the pixel-wise serialising procedures, which has a negative impact on the processing time. To alleviate this, architectures employing processor arrays operating in accordance with Single Instruction Multiple Data (SIMD) paradigm are usually considered [Unger 58]. Such computers employ uniform arrays of processing elements (PE), executing the same series of elementary instructions but processing the locally available data.

The majority of low and medium level image processing tasks fit well into the SIMD paradigm, requiring pixel-wise operations with only nearest neighbourhood connectivity. Therefore, vision chips are typically realised as SIMD pixel-processor arrays, trying to retain the high visual data throughput on chip, and send only the extracted sets of abstract features off the chip. Furthermore, such systems are often expected to provide high processing speed, efficiency, computational accuracy, and to consume low power, especially in applications requiring real time image capturing and processing (e.g. in robotics or industrial inspection and control [Carey 2013]).

### **5.2.3 Vision chips**

In the literature there are several propositions of the vision systems realised as programmable "processor-per-pixel" SIMD arrays in CMOS technologies. They consist of a regular (typically rectangular) array of cells, each incorporating an image sensor and a processing element responsible for data storage, neighbourhood communication and pixel-wise operation [Zarandy 2011], [Moini 97], [Belbachir 2010]. Examples of such generic integrated pixel-parallel processor arrays for variety of computational tasks are SCAMP, realised using discrete time, switched-current analogue circuits [Dudek 2000, 2005], ASPA, realised in digital domain, employing architectural solutions for optimised design such as bit serial arithmetic and asynchronous global summation [Lopich 2010a], and MIPA4k, designed as a mixed-mode circuit, accommodating asynchronous propagation mechanism and separate cores for greyscale and binary image processing [Poikonen 2009]. Other vision chips, based on generic purpose processor arrays, were also presented in [Ishikawa 99], [Komuro 2003, 2009], [Astrom 93] and [Zhang 2011].

Many image processing tasks, including low and medium level operations, can efficiently be solved using cellular neural networks universal machines (CNN-UM), where the operation of the network can be controlled by a set of global parameters (templates), used by each processing cell when operating on the local image data and signals from the neighbourhood [Chua 88a, 88b]. Although, both CNN-UM and SIMD architectures are processor arrays with a neighbourhood connectivity, the main difference between these approaches lies in the principles of computation employed. In SIMD arrays, each processor acts according to the globally issued instruction and performs some elementary operations on the image. Cellular neural networks, on the other hand, settle to a particular state satisfying the input data (image) according to the set of globally defined templates. Even though cellular neural networks can be implemented in digital

domain using numerical computation on processor arrays, the unconstrained continuous information flow in such systems strongly suggests the use of analogue and asynchronous circuits in their hardware realisations. Therefore, the majority of vision chips such as APAP [Carmona-Galan 2003] and ACE [Rodriguez-Vazquez 2004], were designed in mixed-mode approach using continuous and discrete time analogue circuits dedicated for particular arithmetic tasks. The main advantages of the analogue solutions are small area of the processing element and low power operation, when compared to their digital counterparts. On the other hand, the operation of analogue circuit arrays is affected by systematic and random errors, such as noise and fabrication mismatch, which degrades the computational precision and quality of the obtained results [Rodriguez-Vazquez 2003].

### **5.3 Wave propagation approach to skeletonization**

Feature recognition in visual images usually requires global routines, operating on images in some already pre-processed forms. Such feature analysis, often employs binary image skeletonization (structurization), converting objects into more abstract forms representing their structures, shapes and sizes. In practice, skeletonization have been used in character recognition [Arora 2010], biological cell analysis [Xiong 2010] and human action recognition [Chen 2006]. Various methods for binary image skeletonization and their applications were discussed in [Davies 90] and [Lam 92].

#### **5.3.1 Skeletonization (background knowledge)**

There are several ways of defining skeletons. In the following, binary images where objects in the foreground are already segmented and clearly distinct from the background will be considered, as shown in Figure 5.1. In general a skeleton consists of lines and curves creating continuous structure describing the shape and the size of an object [Davies 90]. Skeleton of an object is usually defined as a set of points equally distant from the edges and associated with the "ridges" on the distance transformation map of the object (white lines in Figure 5.1b) [Blum 67]. These points can also be interpreted as the centres of circles drawn into the object (Figure 5.1c) or as a result of collisions between isotropic waves triggered from the boundary and propagating to the inside of the object with a constant speed (Figure 5.1d) [Krinsky 91], [Rekeczky 99].

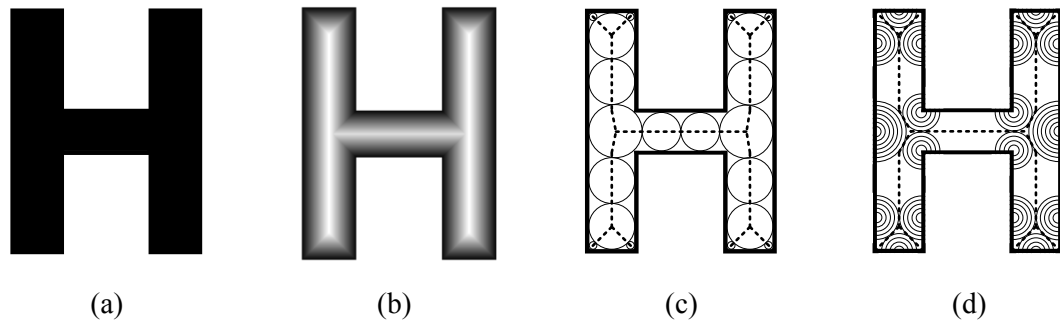


Figure 5.1. a) Binary object, b) skeleton denoted by white "ridges" on the distance transformation map, c) method of circles, d) method of trigger-wave propagation.

There are many different methods of binary image skeletonization exhibiting different levels of complexity and returning slightly different results. The majority of them are too complex for direct and compact hardware implementation. The most common ones, usually considered in VLSI realisations, are based on iterative thinning or distance transformation [Davies 90], [Lam 92]. Thinning algorithms require only logical operations (binary morphology) and one bit per pixel memory. However, a large number of templates, sometimes of two pixels radius, may increase the number of physical interconnections required between cells and the complexity of the corresponding hardware realization. In the case of the distance transformation, every processing element has to store the information about its distance relative to the object's edge. This usually involves numerical computation and makes the implementation more complex and image-size variant. A skeleton extraction, based on the trigger-wave propagation and the wave-front collision detection was initially proposed in [Blum 67]. In this approach, each boundary point triggers a wave that propagates to the inside of the object. It can be assumed that the points where the propagation waves collide form the skeleton of the object. In principle, propagation can be seen as a generic computational engine dedicated for medium level image processing tasks applicable for both thinning and distance transformation. It can be observed that the propagation method aggregates some benefits of both approaches, which makes it particularly promising in terms of efficient hardware implementation in synchronous and asynchronous logic circuits.

### 5.3.2 Trigger-wave propagation concept

There are at least two rather disjoint areas in literature, independently exploring trigger-wave propagation approach in image processing. The idea of wave propagation in feature interpretation tasks was initially discussed in [Blum 67], and later re-introduced

in [Krinsky 91], proposing also VLSI circuits realisations based on works considering experiments with propagation in light sensitive chemical solutions [Kuhnert 89]. Both articles cite previous works, nevertheless these particular publications seem the most comprehensive and influential in terms of further research, defining trigger-wave propagation mechanisms suitable for image processing. Despite different approaches and formalisms used in these works, the discussed concepts are practically the same.

Blum suggested a new revised approach to object perceiving, applicable to feature recognition and interpretation. In his view, Euclidean geometry relying on vectors, and distance measures, is too detailed in terms of the size, position and location, when describing objects. Instead, in order to obtain global attributes of an image, the medial axis function (MAF) based on the trigger-wave propagation and the wave-front collision detection was proposed [Blum 67]. In particular, MAF gave foundation for the distance transformation (DT) and object's structure (skeleton) extraction. It was assumed that MAF and DT are computed in the most "natural" Euclidean metric. Further works found in the literature concerned mainly software methods for DT, showing a number of difficulties with even approximate computation in Euclidean metric. In particular, quasi-Euclidean Distance Transformation (quasi-EDT), was proposed in [Montanari 68]. Optimised sequential algorithms for Euclidean Distance mapping were proposed in [Danielsson 80]. The calculation of DT in metrics different than Euclidean and the application of DT for Voronoi (Dirichlet) tessellation was considered in [Borgefors 86]. Hardware oriented approach, developing the approximation of Euclidean Distance Transformation for parallel architectures working in SIMD mode, was presented in [Razmjooei 2010].

The idea of image processing using trigger-wave propagation presented in [Krinsky 91] resulted in a variety of software and hardware realisations extensively described in [Astrom 96], [Dudek 2006], [Lopich 2009], [Carey 2013]. The majority of works considering SIMD and CNN based solutions are mainly inspired by the properties of the light-sensitive chemical nonlinear system, a variant of the Belousov-Zhabotinski medium, and try to build its electronic hardware replicas. Originally, such system is capable of generating chemical reactions in the form of the propagating wave-fronts stimulated by the intensity of incident light [Kuhnert 89]. It was observed that its behaviour resembles the operation of a special kind of a parallel image processor, capable of performing global image processing tasks. Krinsky proposed more formalized definition of such waves, initially discussed by Blum, treating them as a subclass of the

nonlinear waves (autonomous waves - autowaves). Such waves, contrary to the mechanical waves, exhibit a series of interesting properties. They propagate with a constant speed, utilizing the locally stored energy of the active medium, thus they can expand infinitely preserving their initial amplitudes and contours, in uniform media. The wave-front, moving across the medium, leaves it behind "discharged", inhibiting any other propagation (Figure 5.2a). It means that the only visible dynamic effect of such propagation is the locally moving wave-front separating charged and discharged areas. In particular, the backward propagation is not possible without recharging, therefore, interference and reflection is not observed. Also, when two wave-fronts meet, the propagation cannot proceed further due to the local medium discharge on both sides, and hence, the waves annihilate. The basic properties of autowaves and mechanical waves, are presented in Table 5.1, and graphically illustrated in Figure 5.2b. Autowave propagation is a natural phenomenon and can often be observed as combustion waves (e.g. forest fire), nerve impulses and epidemic spreads [Krinsky 91].

Table 5.1 Properties of mechanical waves and nonlinear autowaves [Krinsky 91].

<i>Property</i>	<i>Mechanical waves</i>	<i>Autowaves</i>
Conservation of energy	+	-
Conservation of amplitude	-	+
Reflection	+	-
Annihilation	-	+
Interference	+	-
<b>Diffraction</b>	+	+

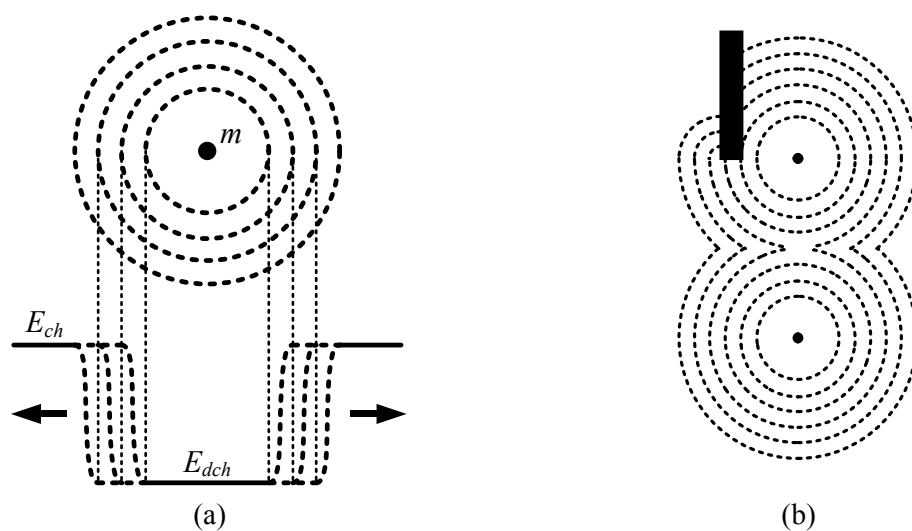


Figure 5.2. Autowave properties (dotted contours illustrate wave-fronts at successive time intervals): a) the expansion of the wave-front in an active medium starting from trigger point marker  $m$  (the wave-front separates charged and discharged areas of energies  $E_{ch}$  and  $E_{dch}$  respectively), b) annihilation of two colliding waves and diffraction on the object.

### 5.3.3 Hardware realisations

Further works following the idea of trigger wave propagation in image processing include various VLSI synchronous and asynchronous implementations of cellular arrays such as 2D CNN array of coupled Chua's circuits [Perez-Munuzuri 93], CNN universal machine with special cloning templates [Rekeczky 99], 2D Global Logic Unit (GLU) arrays [Astrom 96] and Asynchronous Cellular Logic Array (ACLA) [Lopich 2009, 2010a, 2010b, 2011], [Dudek 2006]. Despite some successful realizations of trigger-wave-based algorithms in VLSI circuits, the efficient implementation of the wave-front collision-detection mechanism, essential for skeleton and Voronoi diagram extraction, remains an open problem. Realisation of the propagation and the collision-detection in CNN was discussed in [Rekeczky 99], but the required feedback operator was of two pixels radius. For practical reasons, most of the fabricated VLSI implementations of CNN circuits are limited to only the nearest neighbourhood operators [Rodriguez-Vazquez 93, 2003, 2004], [Halonen 90], [Harrer 92]. The design of a versatile CNN machine with cloning templates of two-pixels radius was reported in [Paasio 2002], however, circuit simplifications preclude the use of the "collision detecting" feedback template from [Rekeczky 99].

## 5.4 Circuit Design

In the approach presented in this thesis, rather than attempting to inhibit the propagation at the point where two wave-fronts meet, as is typically done in the iterative thinning [Davies 90], [Lam 92] or CNN-based methods [Rekeczky 99], it is proposed to employ a separate layer dedicated for detecting collisions between waves in the propagation layer. This simplifies the design of the single cell and enables the asynchronous operation of the entire system. The array consisting of the proposed cells can be arranged by placing cells on a rectangular grid with only four neighbours connectivity.

In this section the design and implementation of the asynchronous processing module (APM) will be described. It consists of two logic circuits connecting with the neighbouring cells in the array and creating two hardware layers, one for wave propagation and one for collision detecting. Due to their structural and operational simplicity, in this thesis, these circuits are also called *gates*, where array of propagation

gates creates the propagation layer and array of collision detection gates creates collision detecting layer.

#### 5.4.1 Propagation gate

In VLSI systems the idea of trigger-wave propagation is typically implemented using pixel-parallel rectangular array of OR gates, where each cell connects to only four nearest orthogonal neighbours [Dudek 2006], as shown in Figure 5.3. Such circuit remains stable when the inputs and outputs of all the gates in the array are in low logic state. The output  $p$  of a particular OR gate depends on the logic state of the outputs of its neighbours  $p_N$ ,  $p_E$ ,  $p_S$  and  $p_W$  and can be set up to high logic state (activated) when a propagation signal is received, or by using the additional input  $m$ , individually triggering a selected gate, according to the formula:

$$p = p_N \vee p_E \vee p_S \vee p_W \vee m \quad (5.1)$$

In such a case, when triggering propagation via marker  $m$ , the neighbours of the activated gate will detect the high logic states on their inputs and turn on, triggering their respective neighbours. The expansion of the activated region around the triggered cell resembles a wave propagation mechanism which continues until all the gates in the array are in high logic state (Figure 5.4). After that the array remains stable and a new propagation is not possible until all the gates are again set to the low logic state (i.e. the array requires initialization before the next evaluation cycle).

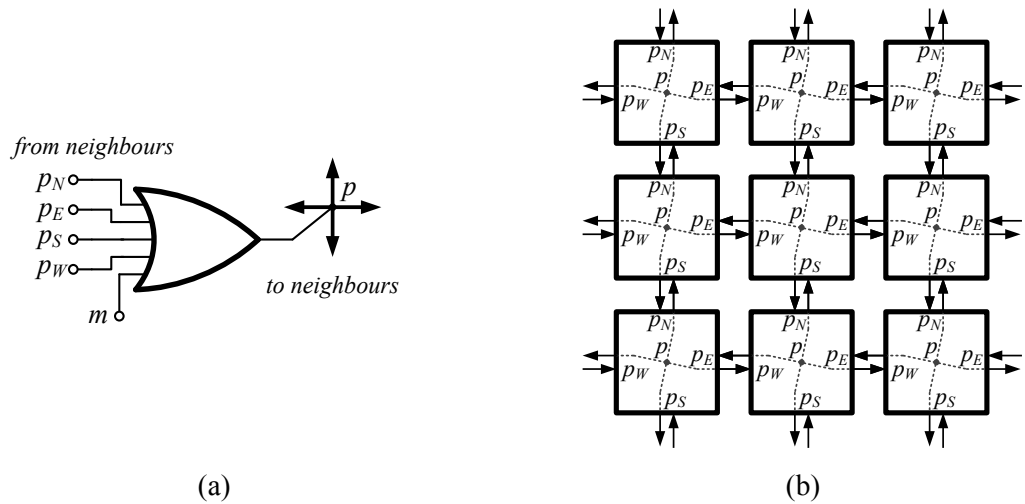


Figure 5.3. a) Schematic diagram of the propagation OR gate and b) the network connectivity of the pixel-parallel logic array realising the wave propagation concept.

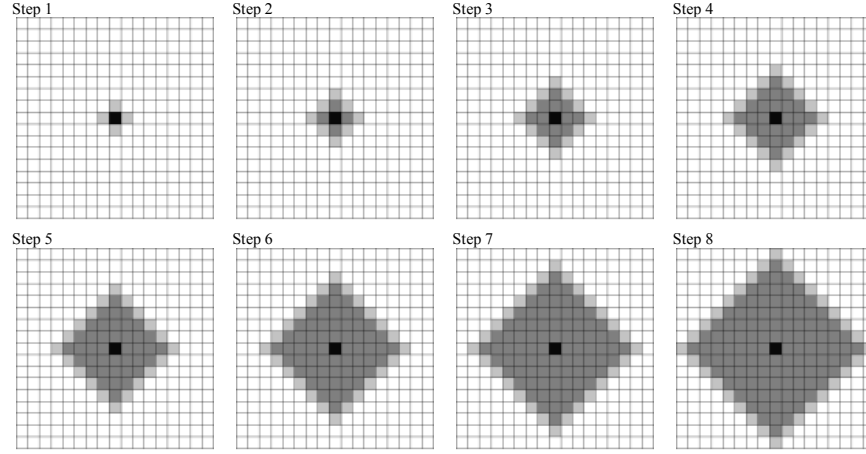


Figure 5.4. The expansion of the propagation wave triggered from the centre of the array (colour map used in the figure: white  $\square$  - charged (not yet activated) cell, light gray  $\square$  - cell that receives the propagation signal, medium gray  $\blacksquare$  - discharged (activated) cell, black  $\blacksquare$  - cell triggered from the marker).

In terms of the trigger-wave propagation concept, logic '0' at the output  $p$  corresponds to the charged cell, and logic '1' denotes a discharged cell. The transition from low to high logic state occurs when the gate receives high logic signal from its neighbour(s), however, the opposite transition (from high to low state) is not possible. Therefore, it can be concluded that a wave propagates at the expense of the locally stored charge and, in order to repeat the propagation, all the cells should be set to low logic state (i.e. the circuit array should be initialised). In particular, all the properties of the autowaves such as amplitude conservation, annihilation etc. (see Table 4.1) can also be observed in such arrays.

Usually, the logical OR gate arrays, used in image processing, require some additional features which, for example, enable to define the propagation space or set a particular direction of the propagation [Dudek 2006], [Astrom 96]. In this work, a design of a test array, solely dedicated to binary image skeletonization and Voronoi tessellation is considered, therefore the propagation space is by default set to the full array size and the binary input image corresponds to the logic states of markers  $m$  used to trigger the propagation.

#### 5.4.2 Collision-detecting gate

The mechanisms of trigger-wave propagation and wave-front collision detection can be found in many implementations of binary image skeletonization. Using the CNN-based methods [Rekeczky 99] or the iterative thinning approach [Davies 90], [Lopich 2009], the execution of the algorithm resembles the propagation which

terminates shortly before the two wave-fronts meet; in order to inhibit the propagation before the collision, each cell needs to monitor the state of its neighbourhood within at least two pixels radius. In the solution proposed in this work, the wave-fronts are allowed to meet and annihilate, whereas a separate logic circuit detecting collisions, based on the logic states of its four nearest neighbours is used. In such approach, if the logic state of all four neighbours of a particular cell turns to '1', it means that a collision has occurred and it could have been caused by two independent wave-fronts meeting at this point. However, it can also mean that a wave has simply passed by this cell and discharged all its neighbours, as observed in a typical propagation scheme. In terms of a circuit implementation, the state of an AND gate, used for detecting the state of the neighbour cells, determines only the necessary but not sufficient condition  $A$  for the occurrence of a collision, given by equation:

$$A = p_N \wedge p_E \wedge p_S \wedge p_W \quad (5.2)$$

It is important to note that the *wave-front collision*, between different wave-fronts, will be indicated by the AND gate before or immediately after the cell becomes activated. On the other hand, when only one wave-front is passing through the cell, the AND gate will indicate the *wave-front pass* condition, after all its neighbours become activated. Therefore, in the proposed solution it is critical to determine the specific time when the collision condition is valid and the state of the AND gate should be saved to discriminate between wave-front collision and wave-front pass situations. The logic diagram of the proposed cell with the propagation and collision-detection mechanisms is presented in Figure 5.5. The corresponding diagram illustrating the timing relations between signals is presented in Figure 5.6. For simplicity, 1D chain of the propagation gates is analyzed.

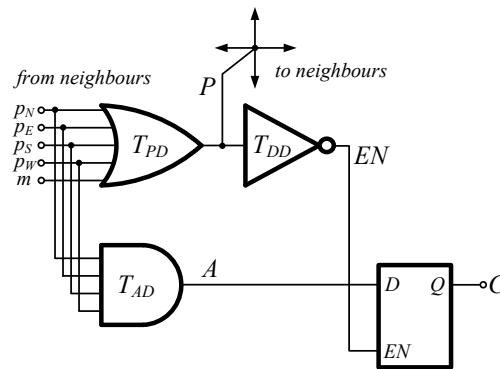


Figure 5.5. Logic diagram of the proposed APM cell.

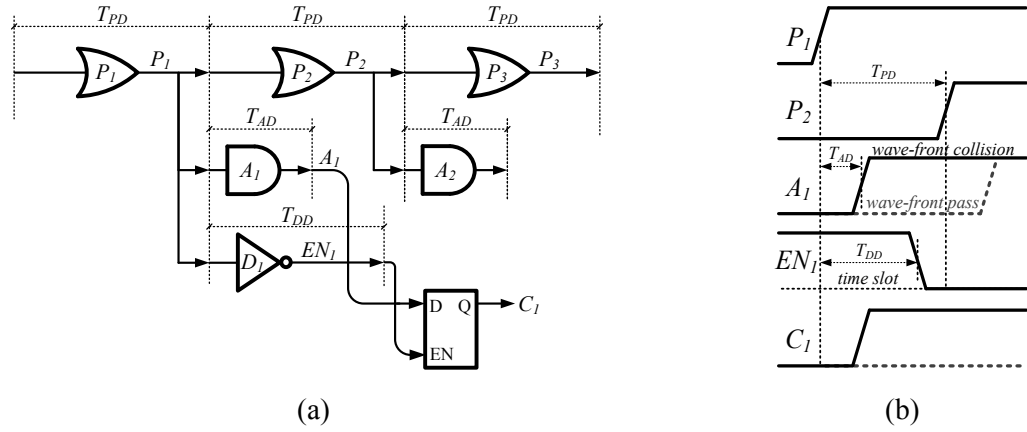


Figure 5.6. The timing analysis of the propagation chain: a) schematic of a 1D array illustrating timing relations, b) timing diagram (dashed traces of  $A_1$  and  $C_1$  show the rejection of the wave-front pass case).

When the propagation gate  $P_1$  of the first cell in the chain receives a signal from its preceding neighbour (or from the marker  $m$ ), it sets the propagation bit  $P_1$  to the high state after a certain delay time  $T_{PD}$ . This indicates the activation state of this cell denoting the beginning of the *time slot* when the collision-detecting circuit is supposed to react. The state of the AND gate  $A_1$  depends on the state of the propagation bits of the neighbouring cells and, if the collision condition is met (i.e. all the neighbours are in the high state), it turns to the high state after a certain propagation delay  $T_{AD}$ . The state of the signal  $A_1$  determines the state for the collision bit  $C_1$ . The corresponding D-latch (storing the value of  $C_1$ ) remains "transparent" when the signal  $EN_1$  is in the high state. This signal is generated by the inverting gate  $D_1$  and turns to the low state after the delay time  $T_{DD}$  after the cell became activated. This terminates the *time slot* for collision-detecting mechanism and latches the value of  $A_1$ . As the propagation progresses, all the inputs of the AND gate eventually receive signals from their neighbours however, the value of the respective collision bit  $C_1$  can only be modified during the *time slot*  $T_{DD}$  (Figure 5.6b). For correct operation it must be ensured that:  $T_{AD} < T_{DD} < T_{PD}$ . It is important to note that in the dedicated VLSI hardware implementation the operation of such an asynchronous structure will be sensitive to process parameter variation (mismatch) and noise, leading to the propagation speed variability across the array. As a result the neighbouring cells may not produce correct logic states right at the beginning of the time slot, but slightly later with a certain random time offset. For that reason, in the practical realization of the circuit, the delay time  $T_{DD}$  will be tuned experimentally to obtain the most satisfactory results (see Figure 5.24).

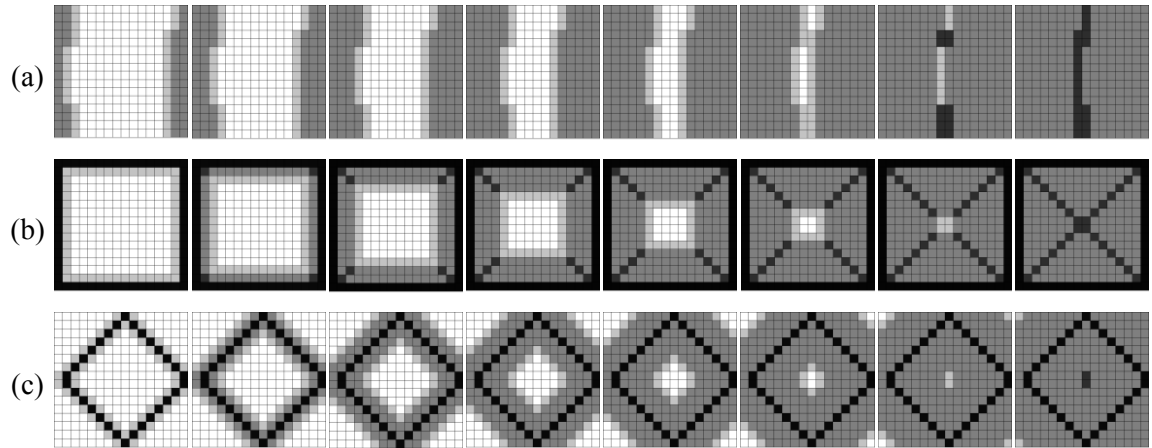


Figure 5.7. The operation of the propagation and collision detection mechanisms in different cases: a) the collision of two irregular parallel wave-fronts (the resulting collision line is of one or two pixels width depending on whether there was odd or even number of cells between colliding wave-fronts), b) square (the proposed mechanism correctly recognises the collision points), c) 45° rotated square (the proposed collision detecting mechanism does not recognise collisions because all the cells in the wave-front have at least one inactive (white) neighbour). The gray scale colour map used in the figure: white  $\square$  - inactive (charged) cell, light gray  $\square$  - cell that receives the propagation signal, medium gray  $\square$  - activated (discharged) cell, dark gray  $\square$  - cell that detects collision, black  $\blacksquare$  - cell triggered from the marker.

The proposed solution is in principle similar to the iterative thinning methods, but far more simplified. Instead of a set of matching templates (usually of two pixels radius), only one, logical AND of the nearest neighbourhood state, is defined. It is assumed that the resulting collision line should be continuous and of (at most) two pixels width, depending on whether there is an even or odd number of pixels in between parallel wave-fronts. An exemplar case, showing the collision of two such waves propagating from the opposite directions, is presented in Figure 5.7a.

The use of only one template applied to the four orthogonal neighbours may occasionally lead to confusions, especially for non-frontal collisions, where at the meeting point of two wave-fronts not all the neighbouring cells are activated. Two cases of the propagation with non-frontal collisions, triggered from edges of a square and a 45° degrees rotated square, are shown in Figure 5.7b and c respectively. In the first case, collisions are detected correctly because all the neighbouring cells of the points where two perpendicular wave-fronts meet are activated at the same time. In the second case (Figure 5.7c), however, the perpendicular waves collide, but the wave-fronts are 45° angled to the cell lattice, therefore each cell, where the collision should normally be detected, has at least one inactive neighbour. In such cases, the logic states of cells located further than the nearest neighbourhood should be considered. Nevertheless, despite the simplicity of the proposed method, it produces very satisfactory results

### 5.4.3 CMOS Circuit realisation

Figure 5.8. Schematic diagram of the proposed APM cell consisting of propagation gate, delay gate and AND-Latch gate (transistor dimensions  $W/L$  are given in micrometers).

The size of each transistor is given in micrometers as the ratio of the channel width to the channel length. The sizes of the transistors were chosen to address the leakage and parameter mismatch issues, discussed further in this chapter. The circuit is realised using two-phase dynamic logic approach, therefore, each cell has to be initialized before the array can perform any operation. During the initialization phase, signal  $V_P$  (precharge) is set to  $V_{DD}$  discharging the parasitic capacitances of nodes  $P$  and  $C$  through  $M_6$  and  $M_{24}$  respectively, and the capacitances of  $NOR$  and  $NAND$  nodes are precharged to  $V_{DD}$  through  $M_8$  and  $M_{22}$ . After that, signal  $V_P$  is set to '0', which terminates the precharge phase and the array is ready to process the input image (i.e. the array is able to carry out one propagation cycle). In order to prevent charge leakage, transistors  $M_8$  and  $M_{22}$  work as weak keepers assuring the high logic state of nodes  $NOR$  and  $NAND$  as long as all the inputs remain inactive.

If any of the input signals turns to a high state (the cell receives the propagation signal), the node  $NOR$  discharges, turning on  $M_7$  and setting the propagation bit  $P$  to  $V_{DD}$ . If all the inputs turn to the high state, the node  $NAND$  can be discharged depending on the state of the signal  $EN$  (enable). This signal is generated by the inverting delay gate with delay time controlled by voltage  $V_{DELAY}$ . As long as  $P$  remains in the low state, transistor  $M_{14}$  pulls up signal  $EN$  to  $V_{DD}$  "enabling" the AND-Latch gate. This enables the latch to "follow" the output  $A$  of the AND gate determining the collision condition. The high state of this signal (when all the inputs  $p_N$ ,  $p_E$ ,  $p_S$  and  $p_W$  in a high state) discharges the node  $NAND$  setting the collision bit  $C$  to a high state. Once this node is discharged, the output state cannot be changed until the next initialization cycle. Such a limitation does not inhibit the cell from proper operation because all the signals received from the neighbours can change only once (from the low to the high logic state).

A critical parameter of the circuit is the duration of the time slot defining how long the AND-Latch gate remains transparent for the signal  $A$  (Figure 5.6b). When the signal  $P$  turns to a high state (denoting the beginning of the time slot) the output capacitance of the delay gate (node  $EN$ ) starts discharging. While the discharge time depends on the current controlled by the transistor  $M_{15}$ , the time slot length can be tuned using voltage  $V_{DELAY}$  biasing the gate of this transistor.

#### 5.4.4 Mismatch optimisation

It was observed that the fabrication mismatch is one of the major contributors affecting the operation of a circuit and degrading the quality of the extracted images. It introduces the variability of the timing parameters of the array which are critical in terms of the correct extraction of the collision lines. Matching between devices can be improved by proper transistor scaling (enlarging). The obtained improvement of the circuit's precision comes, however, at the price of the increased area and power consumption, and is usually limited by other design constraints. Therefore, only the critical transistors in the APM cell, which mostly contribute to the timing parameter variability, were scaled in order to keep the design area small. It was observed that the propagation speed uniformity across the array depends mostly on matching between transistors in the propagation gate whereas the precision of the generated time slot can be improved by proper scaling of the transistors in the delay gate (Figure 5.8). In particular, the evaluation transistors  $M_{1-4}$ ,  $M_7$ , and  $M_{16}$ , and the current limiting transistors (biased from the analogue voltage sources)  $M_{9-13}$  and  $M_{15}$  are the dominant contributors to the performance degradation. The mismatch optimisation of the APM was performed based on the simulation results of a reduced size array of  $32 \times 64$  cells using statistical Monte Carlo MOS transistor models provided by the foundry, and assuming a simplified version of the propagation gate not including current regulating transistors  $M_{9-13}$  and  $M_{15}$ . The sizes of these transistors, however critical, are not of the prime interest at this stage. The goal was to eliminate the artefacts in the resulting images stemming from the random variability of the timing parameters of the cells in the array. Figure 5.9 shows the results of the Monte Carlo mismatch simulations of five circuit arrays consisting of MOS transistors with different sizes. Transistors  $M_{1-5}$ ,  $M_{10-13}$  and  $M_{15,16}$ , in the actual circuit, were oversized due to the available space in the APM layout (Figure 5.8 shows sizes used eventually in the circuit realisation after expanding the critical transistors).

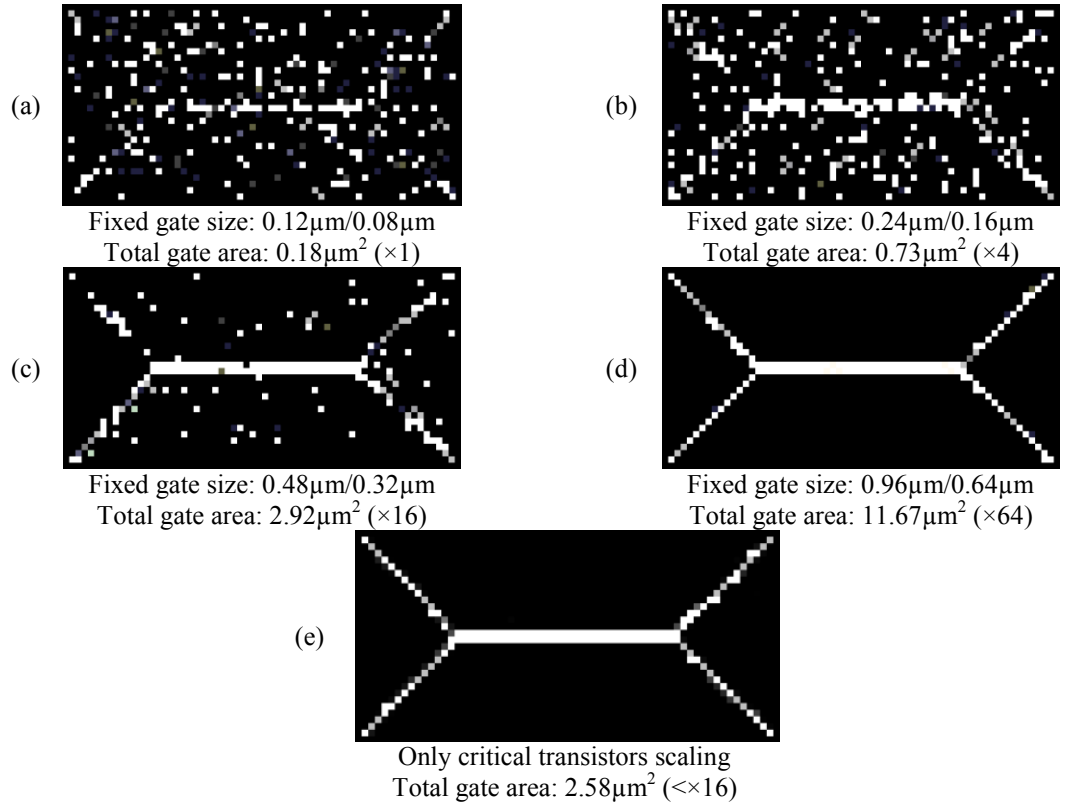


Figure 5.9. Mismatch Monte Carlo simulation results of the full image size rectangle obtained from the  $32 \times 64$  cell arrays consisting of: a) minimum-size transistors, b) transistors of  $4\times$  larger gate array, c) transistors of  $16\times$  larger gate array, d) transistors of  $64\times$  larger gate array, e) transistors scaled using the proposed approach.

It can be observed that the use of the OSI delay gate (presented in Chapter 3) further helped to reduce the variability of the critical timing parameters of the array without additional area increase.

In order to verify the correct operation of the final design presented in Figure 5.8, arrays of  $32 \times 32$  cells using post-layout models of APMs including only parasitic capacitances with delay gates realised based on current starved inverter (CSI) and the proposed output split inverter (OSI) circuits were simulated accounting for fabrication mismatch. The corresponding layouts of the APM modules, with CSI and OSI delay gates, are presented in Figure 5.10a and b respectively. The obtained results of two Monte Carlo runs, showing the collision lines extracted when triggering wave from the border of the array, are presented in Figure 5.11.

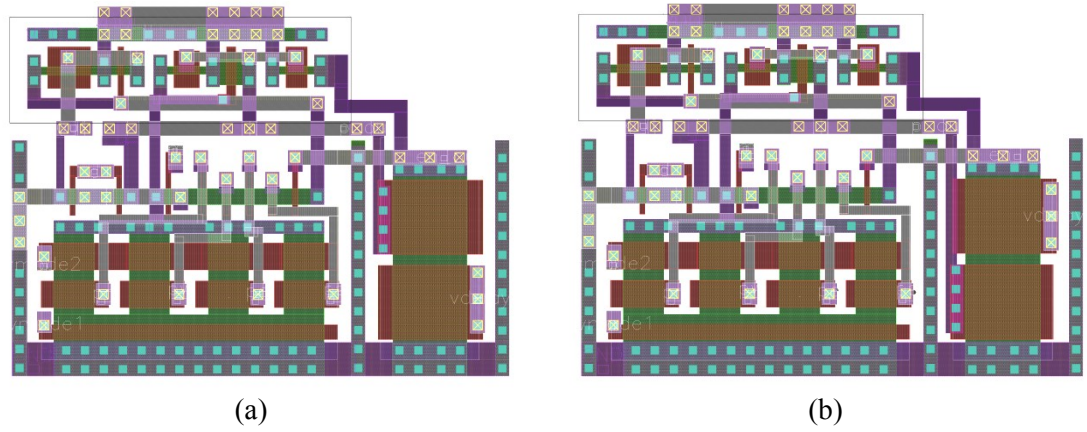


Figure 5.10. Layout of the APM module with delay gate design based on a structure of a) current starved inverter (CSI), and b) output-split inverter (OSI).

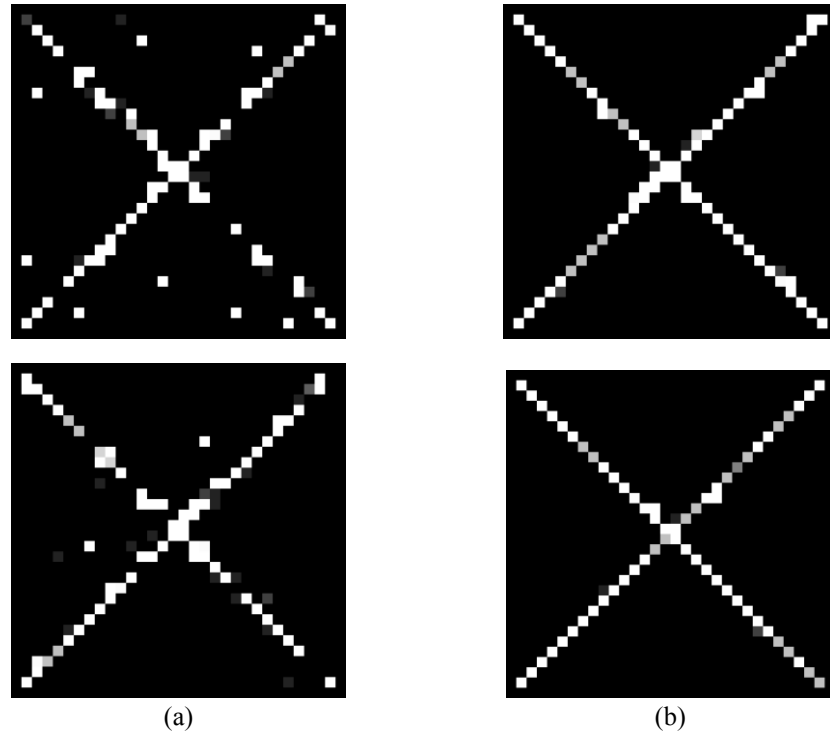


Figure 5.11. Mismatch Monte Carlo simulation results of a full size square (propagation triggered from the boundaries) consisting of  $32 \times 32$  APM cells of the same size with delay gate design based on a structure of a) current starved inverter (CSI), and b) output-split inverter (OSI).

It can be concluded that the proposed APM module with OSI circuit is less affected by the fabrication mismatch and extracts skeletons of better quality. The variability of the timing parameters of the AND-Latch gate does not affect the operation of the circuit, therefore, the weak keeping and the output stage transistors were sized to assure that the gate operates correctly, whereas transistors  $M_{17-21}$  in the NAND pull-down network were slightly widened only to speed up the gate switching process, when the collision condition is met.

## 5.5 Chip implementation

A test array consisting of  $64 \times 96$  APM cells from Figure 5.8, was designed and fabricated in a standard 90 nm CMOS technology. The proposed circuit forms an asynchronous processing module (APM), suitable for integration within a dedicated processor array. In order to communicate with the array implemented on the test chip, each cell accommodates an additional I/O circuit consisting of two D flip flops, a multiplexer and two logic gates. The schematic diagram of the processor used in the implementation of the test array is shown in Figure 5.12. The input image bit is stored in DFF1 D flip flop, and if it is set to '0' (image background), it triggers the APM on the falling edge of the global *START* signal. The APM generates two signals: *P* and *C*, corresponding to the propagation and collision bits respectively. Depending on the state of the lines *S1* and *S0* (see table in Figure 5.12), either bit *P* or *C* can be saved in the second D flip flop DFF2. In the array both D flip flops are serially connected with their respective neighbours and form two separate shift registers REG1 and REG2. The register REG1 is used to shift the input image into the array (one bit per processor). The second register (REG2) can be used to send the captured result off the chip, when both signals *S1* and *S0* are in the high state. The generic structure of the global shift register used in the image data transfers in the chip is presented in Figure 5.13. The rising edge of the clock signal *CLK2* is used to capture the value of bit *P* or *C* in DFF2 after or during the processing, which enables to observe the intermediate results of the propagation. The design of the processing cell (including the APM, I/O logic and signal routing) occupies  $12.5 \mu\text{m} \times 12.5 \mu\text{m}$ . The layout of the cell is shown in Figure 5.14. The proposed APM cell consists of 24 MOS transistors and occupies  $5.5 \mu\text{m} \times 7.4 \mu\text{m}$  which is less than the area of three D flip flops in this technology.

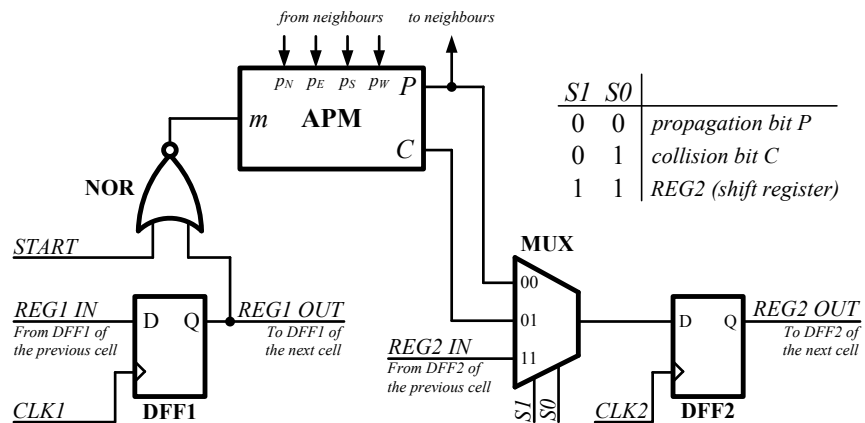


Figure 5.12. Schematic diagram of the processing element including the APM and I/O logic.

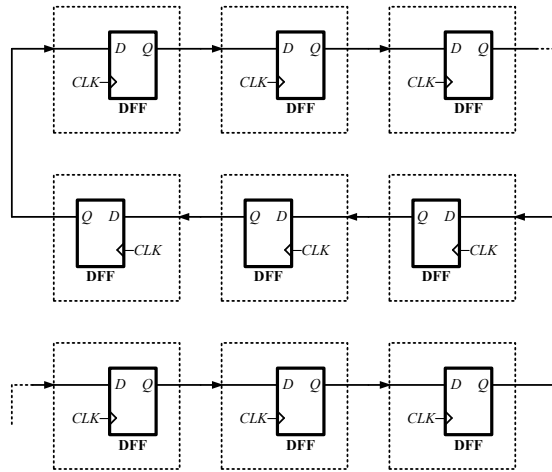


Figure 5.13. The structure of the scan register for serial I/O data exchange used in the test chip design.

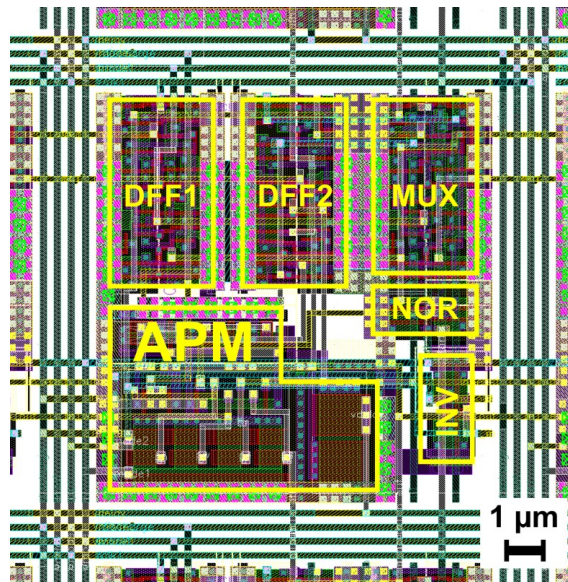


Figure 5.14. The layout of the processing cell including the APM and I/O logic (for clarity the power rails from the top 3 metal layers are not shown, core area:  $12.5 \mu\text{m} \times 12.5 \mu\text{m}$ ).

The additional inverter in the bottom right corner (Figure 5.14) is by default inactive (the output is floating and the input is set to '0'). In some cells these inverters were used as buffers in the signal distribution network, assuring the uniform propagation times of *START* and *CLK2* signals from the respective I/O pads of the chip to each processor. The network is based on the "H-pattern" routing topology shown in Figure 5.15a, where the distance between the centre of a cluster of four cells and the input of each cell (e.g. the clock input of a D flip flop) is the same. This was repeated for each four clusters creating a second level of the signal distribution network covering 16 cells (Figure 5.15b). Following this idea, the uniform signal distribution network can be created for an

arbitrary size square array. Each time the signal descends to the lower level in such hierarchy it is buffered, therefore each buffer drives only the inputs of four buffers from the lower level. The top level of the design consists of  $2 \times 3$  clusters with  $32 \times 32$  cells each. The signals to all 6 clusters were routed manually, also assuring the same path lengths. It can be observed that the correct operation of the system is very sensitive to timing parameters, especially to the uniformity of the falling edge of the *START* signal, triggering the propagation, which can affect the quality of the obtained skeletons. Proper distribution of the *CLK2* signal is also important in experiments when capturing images showing the intermediate states of the propagation and collision detecting layers. Therefore, it was critically important to assure that the propagation delays of both signals are uniform across the array. The chip micrograph showing the designed test array of  $64 \times 96$  processing cells, occupying  $840 \mu\text{m} \times 1200 \mu\text{m}$  area, is presented in Figure 5.16. In the design two separate power supply rails were used for the APMs and the I/O logic blocks respectively.

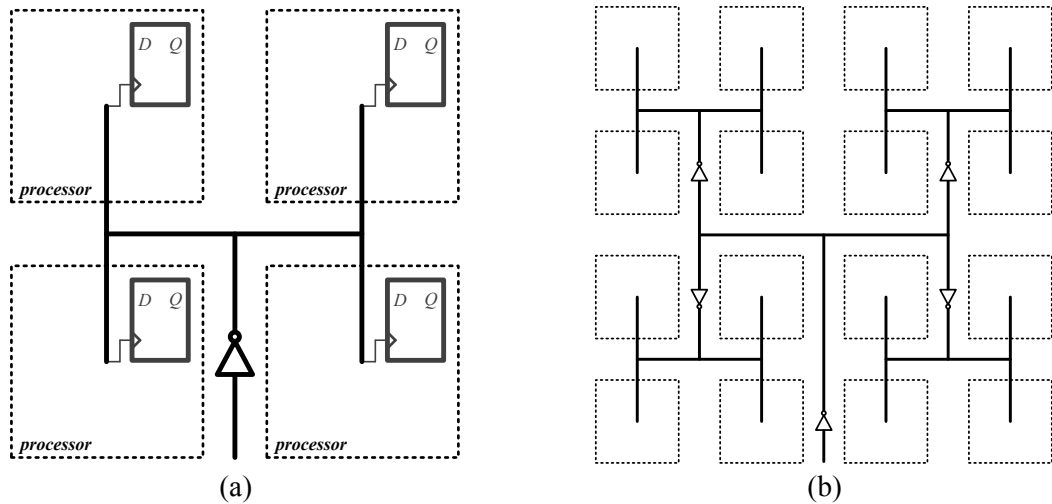


Figure 5.15. The proposed "H-pattern" routing topology for uniform-delay distribution of global signals across the array of processing elements: a) cluster of 4 cells, b) cluster of 16 cells.

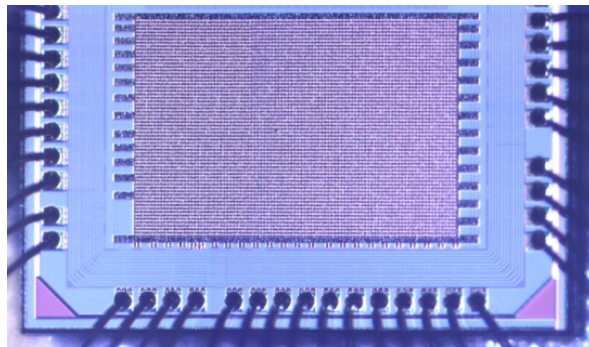


Figure 5.16. Micrograph showing the array of  $64 \times 96$  processing cells (size:  $840 \mu\text{m} \times 1200 \mu\text{m}$ ).

## 5.6 Test system and setup

In order to verify the operation and measure the performance of the fabricated chip, a test system was designed to generate the programming sequences and the control signals for the circuit array, and to provide communication with a PC. The block diagram, showing the structure of the test system and its internal architecture, is presented in Figure 5.17. The design is based on the KCPSM3 (Xilinx PicoBlaze) controller with I/O interfaces and RAM memories implemented on a Spartan 3 XC3S200 FPGA. For communication with a PC, the RS-232 serial interface was chosen due to its simplicity and sufficient speed for this particular test application. The program stored in the ROM memory of the KCPSM3 controller is a command interpreter working in a text mode, executing commands received from the host through the serial port and sending back the results. Therefore, any PC application capable of accessing the serial port (e.g. Hyper Terminal, Matlab) can be used to communicate with the designed system. In particular, the Program Memory Manager module (PMM) can be accessed and any user's program can be uploaded to a separate  $18 \times 1\text{k}$  PRAM memory (Program RAM) and executed by switching between PRAM and ROM. This allows to develop and debug the software for KCPSM3 controller *online* without repeating the synthesis and implementation steps. Also, a dedicated MATLAB application was built to simplify the communication with the test system and to enable image data exchange and visualization. The two 1kB memory banks (RAM A and B) are used to store the binary input and output images.

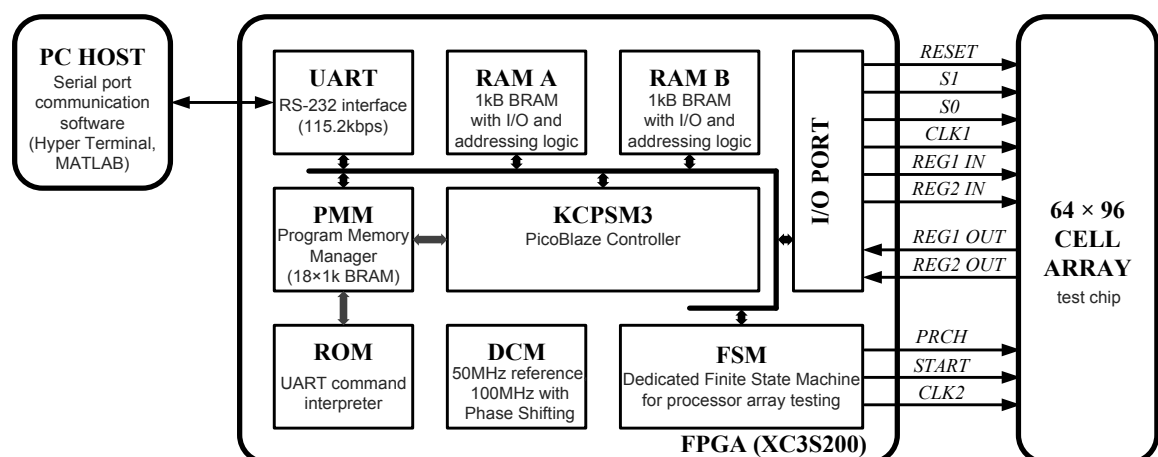


Figure 5.17. Block diagram of the test system.

The reset and I/O sequences controlling *REG1* and *REG2* registers are generated entirely by the KCPSM3 system, reading and writing to particular pins in the generic I/O PORT module. Due to the limited time resolution of such solution, the initialization and evaluation strobes for the chip are generated by a dedicated finite state machine (FSM) unit. The sequence generated by the FSM is shown in Figure 5.18. For the system clock frequency 50 MHz, the minimum time resolution of the FSM is 20 ns. It starts the cycle by bringing all the lines to their initial state (first 20 ns). Then it precharges the array (80 ns) and generates *START* signal to begin the propagation. The inserted delay of 160 ns after the *PRCH* falling edge and before the *START* falling edge is necessary due to ringing observed on the power rails (see Section 5.8). The duration of the evaluation phase (when the *START* signal is in low state) is fixed to 80 ns, which is sufficient for the array to finish the propagation. The rising edge of the *CLK2* signal is used to capture the processed image to register *REG2*. In order to observe the intermediate results of the propagation, an additional circuit generating the rising edge of *CLK2* signal, shifted in phase with reference to the system clock, was designed and implemented using the Phase Shifter from the DCM module on the FPGA (Figure 5.19). Together with the digital delay line, consisting of a series of D flip flops, it generates the delay time equal:

$$\Delta T = (N + 1) \cdot 10\text{ns} \pm P_S \cdot 75\text{ps} \quad (5.3)$$

where  $P_S$  is an 8 bit parameter controlling the Phase Shifter in the DCM module, and can be set in range -128...+127, and  $N$  is a number defining the length of D flip flop chain. After the evaluation phase, the FSM can generate from 1 to 256 wait states of 20 ns each before it returns to the initialization state. This enables to modify the length of the generated initialization-evaluation cycle from 360 ns to 5.46  $\mu\text{s}$ .

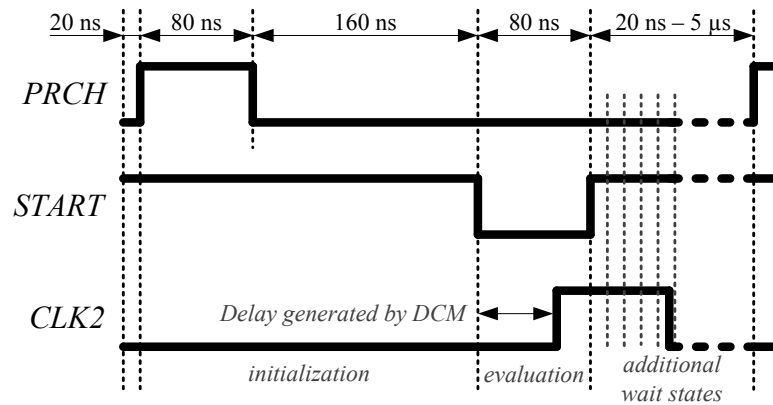


Figure 5.18. Time diagram showing the initialization and evaluation sequences generated by the dedicated FSM.

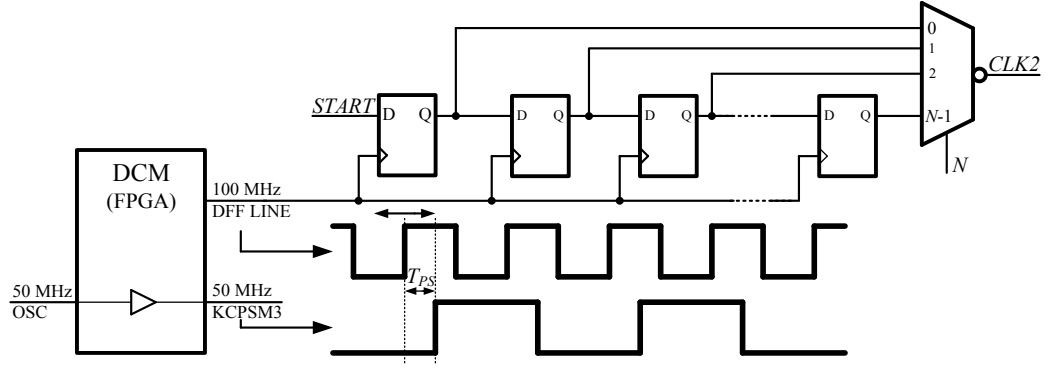


Figure 5.19. Schematic diagram of the CLK2 slope signal generator.

## 5.7 Experimental results

The operation of the fabricated chip was verified in a laboratory environment using the test system described in Section 5.6 and a dedicated set of voltage regulators providing required biases and power supplies working in range 0 - 2.5 V. The bias voltages  $V_{MODE1}$  and  $V_{MODE2}$  were adjusted experimentally to assure the circular contours of the propagation waves (the details will be provided in Chapter 6). The  $V_{DELAY}$  voltage was adjusted to minimise the occurrence of the wave pass conditions, misclassified as collisions. The supply and bias voltages used in the tests presented in this paper are summarized in Table 5.2. The reasons for selection of such values will be discussed further in this and the next chapter.

Table 5.2. The supply and bias voltages used for testing the fabricated chip assuring circular propagation.

<i>Parameter</i>	<i>Value</i>	<i>Remarks</i>
$V_{DD}$	930 mV	Power supply voltage (APMs)
$V_{CC}$	930 mV	Power supply voltage (control and I/O logic)
$V_{DELAY}$	396 mV	Delay gate bias voltage
$V_{MODE2}$	511 mV	Propagation gate bias voltage (serial input transistors)
$V_{MODE1}$	315 mV	Propagation gate bias voltage (common input transistor)
$V_{IO}$	2.5 V	I/O ring supply voltage of the test chip

### 5.7.1 Results comparison

The skeletons obtained from the designed asynchronous array were compared with the results from two skeletonization algorithms. The first algorithm is the implementation of the propagation and collision detection mechanisms (discussed in Section 5.4) but executed synchronously. The second algorithm is more complex and extracts octagonal skeletons based on the iterative thinning using 8 structuring-element pairs [Haralic 92].

Its software implementation, available in Matlab Image Processing Toolbox is *bwmorph* function called with a parameter *skel*, is used here as reference. The skeletons of several "natural" objects and geometric shapes, extracted by the fabricated circuit array and computed using the aforementioned algorithms, are presented in Figure 5.20.

The differences in skeletons extracted by the iterative thinning method (Figure 5.20c) and the synchronous propagation-based method (Figure 5.20b) result mainly from the simplicity of the proposed collision detecting mechanism. It is equivalent to a single template matching approach, limited to only four nearest neighbours and logical AND operation. The differences between the synchronous (software) and asynchronous (hardware) implementations of the propagation-based method result from circular contours of the trigger-waves generated in the circuit array. In a synchronous implementation, a wave propagates with a constant cell-to-cell speed, therefore, it takes it twice as much time to reach the nearest cell on the diagonal direction than the nearest cell in the cardinal direction. As a result, the contour of the wave-front triggered from a single cell resembles a 45° rotated square (diamond), as presented in Figure 5.4, indicating anisotropic wave propagation speed (lower in the diagonal direction). In the asynchronous realisation, the diagonal cell receives signals from two nearest neighbours simultaneously which shortens the propagation time through this cell. As a result, the contour of the trigger-wave is closer to circular, as shown in Figure 5.21. The bias voltages  $V_{MODE1}$  and  $V_{MODE2}$  can be used to tune the timing parameters of the propagation gate, and hence, the shape of the generated wave-fronts. The principles of such circular propagation are further discussed in Chapter 6. In the case of skeletonization, the use of isotropic waves produces more accurate results with fewer discontinuities in the extracted skeletons when compared to the results obtained from the synchronous implementation of the propagation-based method.

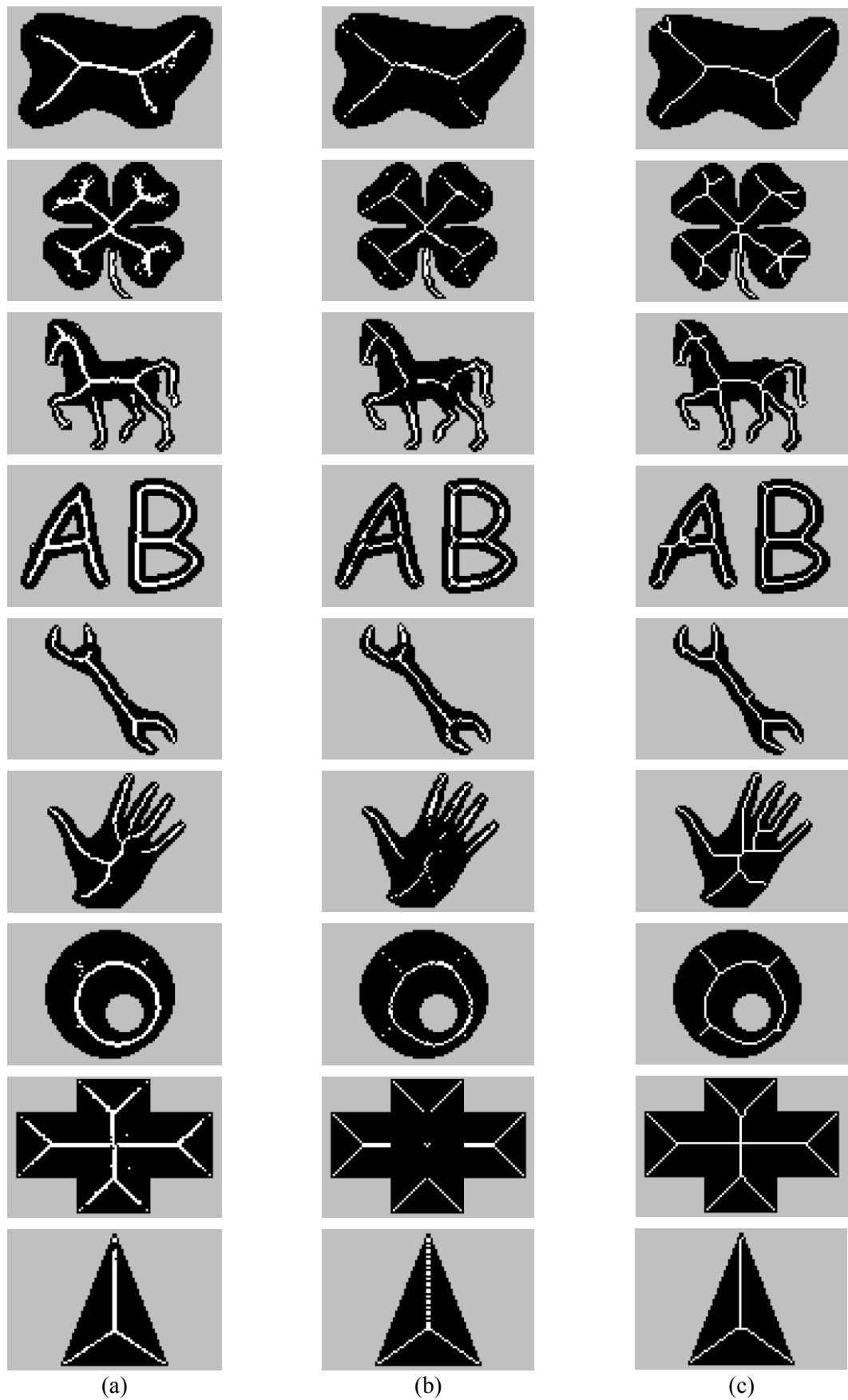


Figure 5.20. The skeletons of different objects extracted by: a) the asynchronous array on the test chip, b) the synchronous implementation of the propagation and collision detection algorithm, c) the iterative thinning method (reference).

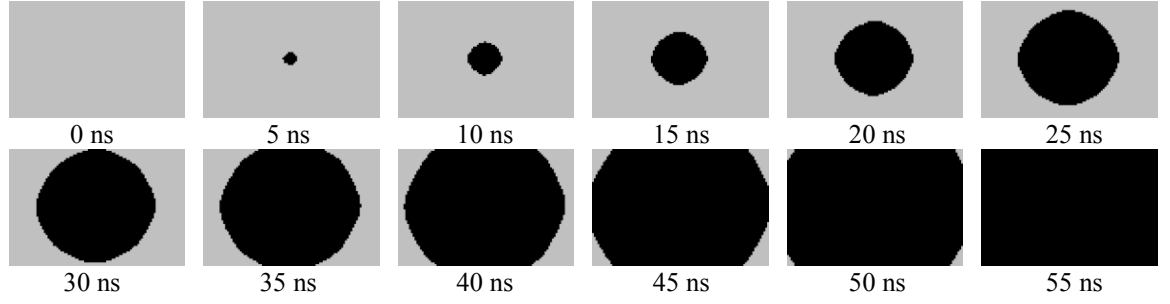


Figure 5.21. The intermediate states of the propagation wave triggered from a single pixel in the middle of the array and captured after each 5 ns.

The proposed propagation-based method can also be applied to generate Voronoi diagrams of binary images. In such a case, the waves are triggered from a set of points and the resulting collision lines create the tessellation of the propagation space. The experimental results showing the effects of the propagation triggered from several points in the array are presented in Figure 5.22.

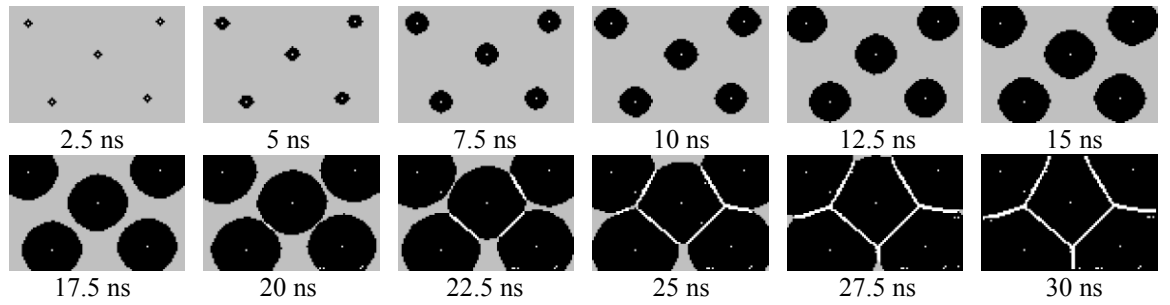


Figure 5.22. The intermediates steps of the propagation and collision detection mechanisms captured after each 2.5 ns showing the generation of the Voronoi diagram.

For comparison, three different tessellation diagrams obtained from the test chip and computed using *voronoi* function in Matlab are shown in Figure 5.23. It can be observed that the proposed method using isotropic propagation and collision detection scheme correctly extracts tessellation diagrams. It can be observed that the lines segmenting images widen towards the edges of the array. This results from the unbalanced load of the border cells, where the propagation speed along the periphery of the array increases, confusing the collision detecting circuits, which misclassify the wave-front pass as a collision. This and other design issues will be discussed in section 5.8 of this chapter.

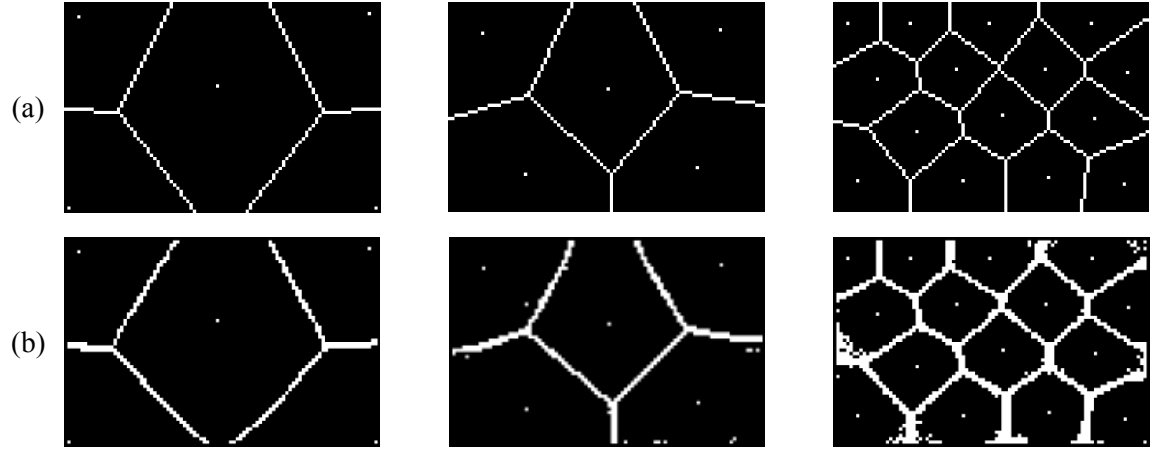


Figure 5.23. The results of the Voronoi tessellation obtained using: a) *voronoi* function in Matlab, b) asynchronous processor array on the test chip.

### 5.7.2 Delay voltage tuning

The quality of the obtained skeletons and tessellation diagrams depends on the  $V_{DELAY}$  voltage regulating the delay time slot for the cells in the array. The experimental results showing skeletons of four different images, extracted for  $V_{DELAY}$  bias from 270 mV to 930 mV, are presented in Figure 5.24. For lower values of  $V_{DELAY}$ , the generated time slot becomes longer and cells may start to misclassify the wave-front pass condition as a collision. As a result, the detected collision lines become wider and, due to the fabrication mismatch, can also be surrounded by some artefacts resembling the presence of noise. For higher values of  $V_{DELAY}$ , the extracted collision lines become fractured and eventually disappear. It is important to note that such artefacts, contrary to the random noise, are stationary. The presence of random noise has been observed in the circuit but its level is very low and practically does not affect the obtained results. It can be observed that the value of  $V_{DELAY}$  is critical in terms of the quality of the obtained results (Figure 5.24). The best skeletons can be extracted for  $V_{DELAY} \approx 400$  mV. Based on a number of experiments with different images, for the particular chip used in all the tests, the value of  $V_{DELAY}$  was fixed to 396 mV.

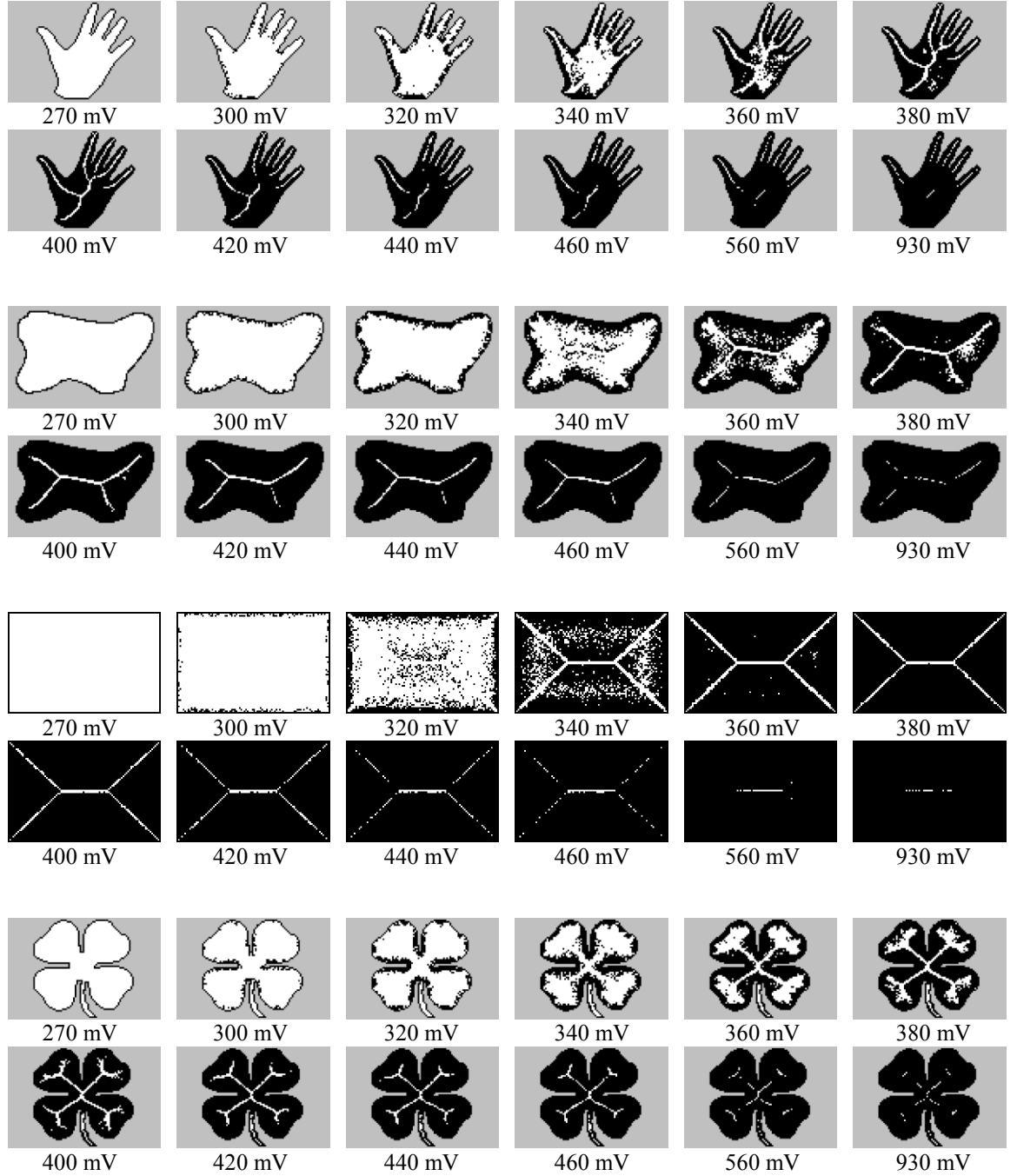


Figure 5.24. The skeletons of four images extracted for different  $V_{DELAY}$  bias voltages from 270 mV to 930 mV.

### 5.7.3 Supply voltage variability

The operation and robustness of the prototype array was verified for the variable supply voltage  $V_{DD}$  (supplying APMs) and the variable temperature. In the experiments bias voltages  $V_{DELAY}$ ,  $V_{MODE1}$  and  $V_{MODE2}$ , and the digital supply  $V_{CC}$  were constant. The respective values of these voltages are provided in Table 5.2. The results of the  $V_{DD}$  voltage variability within range  $\pm 100$  mV, corresponding to the relative variability of

+/- 10%, are presented in Figure 5.25. The supply voltage of APMs affects the propagation speed, which decreases for smaller and increases for higher values of  $V_{DD}$ . As a result, the time slot of the collision detecting mechanism, tuned by  $V_{DELAY}$  to a particular propagation speed, will either be too short or too long, giving incomplete skeletons (Figure 5.25a) or excessive number of artefacts (Figure 5.25c) in the extracted images. This can be fixed by proper adjustment of  $V_{DELAY}$  voltage. In the experiments, when the array worked at  $V_{DD} = 830$  mV and 1030 mV, the correct skeleton results (the same as for the nominal bias and supply conditions from Table 5.2) could be obtained for  $V_{DELAY}$  equal 370 mV and 430 mV respectively.

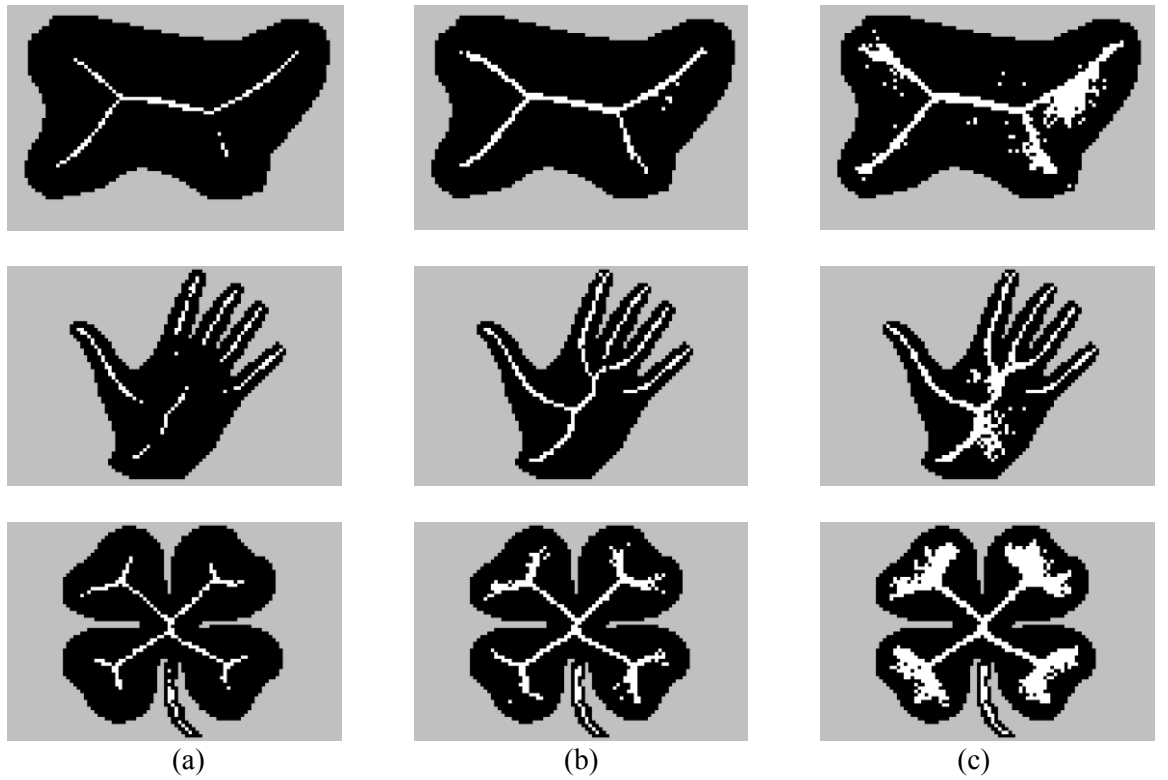


Figure 5.25. Images obtained for different supply voltages  $V_{DD}$  equal: a) 830 mV, b) 930 mV (nominal value), c) 1030 mV.

#### 5.7.4 Temperature variability

The effects of the temperature variation on the quality of the obtained results are presented in Figure 5.26. The operation of the chip was tested in three different temperatures: 25°C (room temperature), 40°C and 60°C, assuming constant bias and supply voltages from Table 5.2. It was observed that the increased temperature does not significantly affect the propagation speed but it lengthens the generated time slot  $T_D$ . Assuming that the OFF (leakage) current of a MOS transistor increases and the ON

current decreases with temperature [Allen 2002], the leakage currents of transistors  $M_{1-4}$  in the first stage of propagation gate (Figure 5.8) will increase and the ON currents of the weak keeper  $M_8$  and the output transistor  $M_7$  will decrease. For a rising slope on any of the inputs of the propagation gate, the *NOR* node will be discharged faster to zero and the output node  $P$  will be pulled up to  $V_{DD}$  slower. These two effects, to some extent, compensate each other with the temperature increase, resulting in almost constant propagation speed. The temperature increase affects also the operation of delay gate, where the current of the transistor  $M_{15}$  decreases, elongating the generated time slot. Consequently, for too long time slot, the circuit may start to mistakenly recognise the wave-front pass conditions as a collisions, leading to wider skeletons and additional artefacts, as shown in Figures 5.26b and c. The effects of the temperature increase can be compensated by proper adjustment to the bias voltage  $V_{DELAY}$ . It was observed that the correct skeletons could be obtained for  $V_{DELAY} = 410$  mV and 420 mV for temperatures 40°C and 60°C respectively. In practical applications (e.g. in vision chips) such calibration of  $V_{DELAY}$  could be done automatically to minimise the disparities between the extracted skeletons and the reference results, using a set of input images and reference results. Based on such comparisons, bias voltages minimising the number of errors could be found.

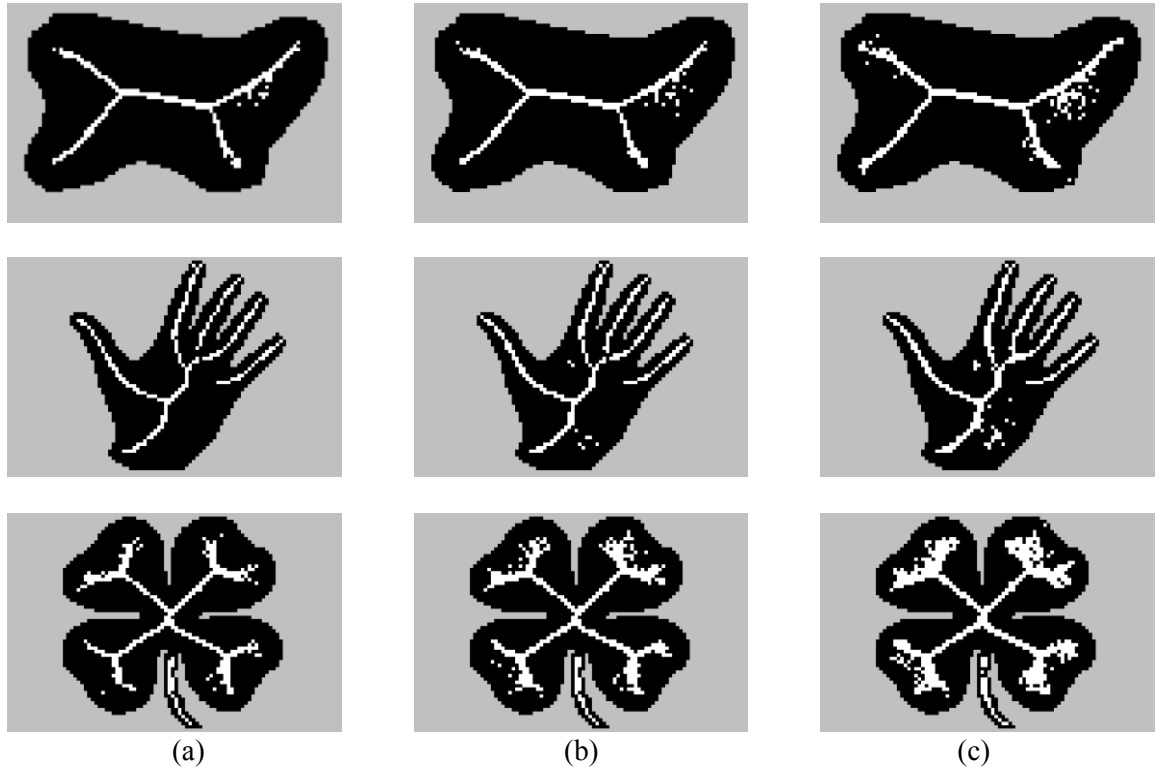


Figure 5.26. Images obtained for different temperatures assuming constant bias and supply voltages from Table 3.2: a)  $T = 25^\circ\text{C}$  (room temperature), b)  $T = 40^\circ\text{C}$ , c)  $T = 60^\circ\text{C}$ .

## 5.8 Design issues

### 5.8.1 Power rail ringing

In the experiments with the fabricated chip, it has been observed that the quality of the obtained results is strongly affected by the voltage oscillation (ringing) on the power rail, which occurs after a larger group of cells switches at the same time. Such situation occurs during the precharge phase, when the rising edge of the *DSCH* signal initializes all the cells in the array simultaneously, and during the evaluation cycle, when the falling edge of the *START* signal triggers the propagation from the markers. In the first case, the additional delay of 160 ns was inserted to assure that the power rail settles before the evaluation phase (Figure 5.18). This reduced the influence of the precharge cycle on the quality of the computed results. However, the influence of ringing caused by the falling edge of the *START* signal on the extracted image cannot be easily reduced. It modulates the propagation speed, confusing the collision detection circuit. In the resulting images, the intensity of artefacts will increase in the regions where the wave-front passes faster (due to the maximum in the supply voltage oscillations), and decrease in the regions where the propagation slows down.

The oscillations of the power rail can be visually observed in the resulting image for lower  $V_{DELAY}$  voltages. The generated time slot of each cell increases and the circuit array is more prone to misclassifying the wave-front pass conditions as collisions. This "side effect", however, can be used in observing the changes of the propagation speed across the array. In particular, for a certain value of  $V_{DELAY}$ , the mistakenly captured collision pixels, caused by the increased propagation speed from the power rail oscillations, can be observed. In addition to that, the fabrication mismatch adds some random offsets to the generated time slots for all the cells individually, and the propagation speed variability can be observed in more "analogue" way as a variable concentration of the collision pixels, intensifying in the regions where the propagation accelerates. The experimental results showing such mistakenly captured pixels, when triggering propagation from a column of cells and a larger block of cells on the left hand side of the array, for  $V_{DELAY}$  reduced to 330 mV, are presented in Figure 5.27. For a small number of markers (Figure 5.27a) there is a large dark area around the object, suggesting a drop of the power supply voltage resulting in a lower propagation speed. The observed increase of the intensity of white pixels indicates the first maximum of the supply voltage. After that, the oscillations practically settle and the propagation speed remains constant. For a large number of

markers, the obtained image shows several distinct and white stripes (Figure 5.27b), indicating higher amplitude of the supply voltage oscillations.

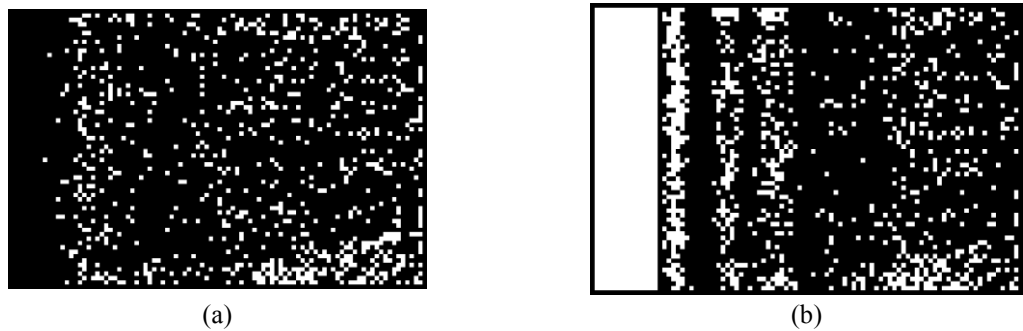


Figure 5.27. The oscillations of the supply voltage affecting the propagation speed in the designed array when triggering propagation from: a) a column of cells, and b) from a larger block of cells on the left hand side of the array.

The effects of ringing on the extracted skeleton are shown in Figure 5.28. When the propagation is triggered from the background pixels (Figure 5.26a), the oscillations of the power rails impede the correct recognition of collisions. One way of reducing this undesirable effect is to reduce the number of marker pixels and trigger the propagation only from the objects boundary (Figure 5.26b). This requires additional logic operations on the input and output images, such as binary edge detection and masking, but these operations can easily be performed by a standard SIMD processor. Proper adjustment of the bias voltages  $V_{MODE1}$  and  $V_{MODE2}$  also helped to reduce the propagation speed, and hence, the switching current of the propagation gates, further reducing ringing.



Figure 5.28. The skeletons of the same object extracted for: a) wave triggered from all the background pixels, and b) wave triggered only from the pixels on the border of the image.

### 5.8.2 Design asymmetry

The proposed test array was designed for binary image skeletonization, where the waves are always triggered from the objects' boundary and propagate to its interior, but never along the borders of the array. However, in the case of the Voronoi diagram extraction, it is required that the propagation speed remains constant even along the border lines. In the designed test array the border line cells, unlike the rest, trigger only three or two nearest neighbours. Due to the lower output load, the propagation speed increases along the borders. The intermediate steps of the propagation, triggered from a column of pixels on the left hand side of the array, are presented in Figure 5.29.

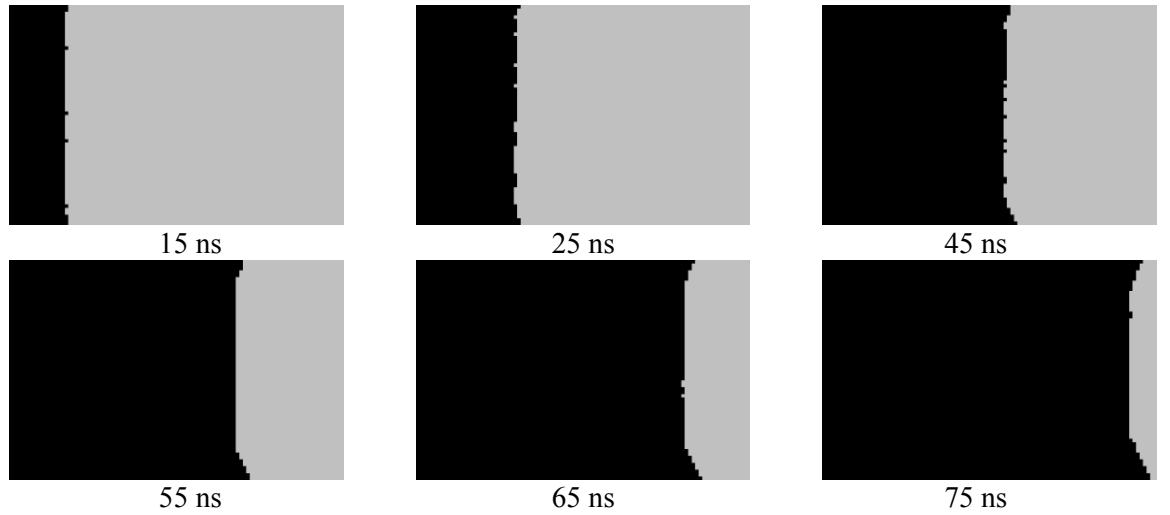


Figure 5.29. Bending of the propagation wave contour caused by the unbalanced load of the border cells.

It can be observed that the cells located along the borders propagate faster triggering other cells inside the array. As a result, the wave-front bends towards the borders and the propagation speed increases. This can confuse the collision detecting mechanism, leading to widening the extracted collision lines, as shown in Figure 5.23b. This could easily be fixed by inserting an additional set of dummy gates around the array balancing the load, or by implementing the propagation space control mechanism, switching off the border line pixels if necessary.

It should also be noted that transistors  $M_{17-21}$  in the pull-down network of the AND-LATCH gate (Figure 5.8) have slightly different gate capacitances during switching, due to the body effects resulting from the serial connection and a common substrate potential. Such systematic asymmetry of the array can affect the propagation speed and make it direction dependent. For example, the wave contour in Figure 5.29 bends faster at the

bottom side than at the top side of the array (i.e. the wave propagates faster to the north-east than to south-east direction). This, however, has a minor effect on the quality of the obtained results, as long as transistors  $M_{17-21}$  are much smaller than  $M_{1-4}$ , which is usually the case, since only  $M_{1-4}$  are enlarged to reduce parameter mismatch.

## 5.9 Performance and power

The performance of the designed integrated circuit was verified using a dedicated FSM module on the FPGA test system (Figure 5.17), working in a loop and generating repetitive signal sequences for the initialization and evaluation cycles of the processor array. The respective timing of a single initialization and evaluation cycles (Figure 5.18) was adjusted to assure the correct extraction of the collision lines. The shortest sequence generated by the FSM (with no additional wait states), takes 360 ns and the longest (with 256 wait states) 5.46  $\mu$ s. In the power estimation, only the current of the power rail supplying the array of the APMs in the processor array ( $I_{APM}$ ) was considered. The correct operation of the prototype chip was verified for constant supply and bias voltages (see Table 5.2), temperatures 25°C, 40°C and 60°C, and values of  $V_{DELAY}$  tuned to compensate for the temperature effects. During tests, when the FSM was working in a loop, the obtained results were saved to register REG2 (on the rising edge of  $CLK2$  signal) overwriting the previous result without shifting it out. Only the last result, captured during the last cycle of the FSM after which it was stopped, was transmitted off chip for verification. For power measurements, four different input images were used: white (with no markers triggering the propagation), single dot in the middle (only one marker triggering the propagation), full size rectangle (markers on the boundary of the array) and black (all pixels are markers). The operation of the array was tested at the maximum processing speed (2.78 MHz at 360 ns FSM cycle) and in the idle state (when  $START = '1'$  and  $DSCH = '0'$ ) in temperatures 25°C, 40°C and 60°C. The summary of the tests and the obtained results are presented in Table 5.3

Table 5.3. Performance and power results measured at 25°C, 40°C and 60°C.

<b>Test Condition (input image)</b>	<b><math>I_{APM}</math> @ 2.78 MHz</b>			<b><math>I_{APM}</math> @ 0Hz</b>		
	$T = 25^{\circ}\text{C}$	$T = 40^{\circ}\text{C}$	$T = 60^{\circ}\text{C}$	$T = 25^{\circ}\text{C}$	$T = 40^{\circ}\text{C}$	$T = 60^{\circ}\text{C}$
<b>white</b> (no markers)	0.20 mA	0.42 mA	0.77 mA	0.31 mA	---	---
<b>single pixel</b> (propagation from the centre)	1.75 mA	1.95 mA	2.25 mA	0.31 mA	0.47 mA	0.70 mA
<b>full size rectangle</b> (markers around the array)	1.78 mA	2.00 mA	2.27 mA	0.32 mA	---	---
<b>black</b> (all pixels are markers)	1.95 mA	2.13 mA	2.40 mA	0.43 mA	---	---

In the case of the white input image, there is no propagation triggered in the array. It can be observed, however, that the corresponding current consumption is lower during the operation of the array (0.20 mA) than in the idle state (0.31 mA). This is caused by the leakage currents affecting the initial state of the array after the initialization cycle. In particular, nodes  $P$  (outputs of the propagation gates) are slowly pulled up through transistors  $M_7$ , which eventually triggers a "spontaneous" propagation, discharging the array. After such discharge, the state of the array settles depending on the balance between the leakage currents, leaving all the transistors not fully turned on or off and creating DC paths between the power and ground rails. To assure low power operation in the idle state of the circuit, signal  $DSCH$  should be kept at high logic state, limiting the total leakage current of the APM modules to about 0.20 mA ( $\sim 33$  nA/pixel) at 25°C. This current increases with temperature to 0.40 mA at 40°C and to 0.68 mA at 60°C.

The differences of the current consumption, measured at the maximum processing speed for the last three images in Table 5.3, result mainly from different number of collision pixels detected in each of the images. For example, for the single dot image, the measured average current is the lowest (1.75 mA) because there are no collisions detected. Each time a collision is detected, the respective AND-Latch gate is discharged, which requires additional amount of energy to overcome the weak keeping transistor  $M_{22}$  (Figure 5.8). When the processed image generates collision pixels, the corresponding supply current will be higher (e.g. processing a full image size rectangle increases the supply current  $I_{APM}$  to 1.78 mA). The black input image was used to measure the worst case of the power consumption when all the pixels are markers and detect collisions. The respective supply current during the operation is 1.95 mA. Also, the corresponding supply current in the idle state (0.43 mA) increases due to the increased leakage of the AND-Latch gate when the collision bit  $C$  is set to '1' ( $\sim 70$  nA/pixel). The main design parameters and power performance measured at 25°C of the prototype chip are presented in Table 5.4.

The maximum processing speed of the designed array is mainly limited by the oscillations after the initialization (precharge) cycle of the power rail requiring additional 160 ns delay between the slopes of the  $PRCH$  and  $START$  signals. The propagation speed of about 1.1 pixels/ns typically requires less than 60 ns time to complete processing of an image. In the experiments, the delay of maximum 80 ns was assumed for testing. At such speed the array performs 2.78 million initialization and evaluation cycles per second. In practical implementations (e.g. in vision SoC), the speed will be limited by other factors

such as off chip data transfer, and the required maximum frame rate. In such a case, the design can benefit from a very low processing power, theoretically below 1 nW/fps, however, only applicable to frame rates over 200 kfps, where the dynamic power is higher than the static leakage losses. For lower speeds, the total power will mainly be limited by the leakage remaining on the constant level of 0.2 - 0.4 mW, depending on the number of the detected collision pixels. For designs with strict power constraints, the quiescent current can be reduced by supply voltage scaling or by using high threshold voltage and low leakage transistors in the design of APM. It should be noted that supply voltage scaling will affect the propagation speed and the use of MOS devices other than regular may increase the parameter mismatch requiring larger transistors to assure correct operation of the array.

Table 5.4. Performance, power and design parameters of the fabricated test array.

<i>Parameter</i>	<i>Value</i>	<i>Remarks</i>
Technology	CMOS 90 nm	Dynamic logic full custom design
No. transistors in APM	24	Sized accordingly to reduce effects of mismatch
APM size	$5.5 \mu\text{m} \times 7.4 \mu\text{m}$	Less than 3 D flip flops in the same technology
Test processor size	$12.5 \mu\text{m} \times 12.4 \mu\text{m}$	Including APM, I/O and control logic
Test array size	$840 \mu\text{m} \times 1200 \mu\text{m}$	$64 \times 96$ processor array
Image resolution	$64 \times 96$	
Propagation speed	1.1 pixel/ns	1.4 pixel/ns for $V_{MODE1}$ and $V_{MODE2}$ set to $V_{DD}$
Current consumption	33 nA/pixel	idle mode ( <i>DSCH</i> in high state)
	50 nA/pixel	propagation without collision (idle state)
	70 nA/pixel	propagation with collision (idle state)
Max. processing speed	2.78 Mfps	With no quality degradation
Power consumption	< 1mW/1Mfps	for the supply voltage 930 mV and > 200 kfps
Min. power consumption	0.2 - 0.4 mW	For < 200 kfps, depending on the number of markers in the image ( $V_{DD} = 930 \text{ mV}$ )

## 5.10 Design improvements and conclusions

### 5.10.1 Conclusions

The proposed circuit array implements the propagation and collision detection mechanisms suitable for a variety of morphological operations on binary images. In particular, its ability to detect the collisions between trigger-waves, can be used in binary image skeletonization and Voronoi tessellation. Despite the simplicity of the proposed method, the employment of the asynchronous circuit, generating circular propagation waves rather than square-like, produces good quality results when compared to its synchronous implementation. Low power, low area and a very high processing speed have been achieved employing full custom, dynamic logic design. The prototype array,

consisting of  $64 \times 96$  APMs with additional I/O and control logic, was designed and fabricated in a standard 90 nm CMOS technology using standard performance design kit (SP) with very thin gate oxide MOS devices ( $t_{ox} = 1.6$  nm). The experimental results confirmed the correct operation of the proposed circuit capable of processing up to  $2.78 \times 10^6$  binary images per second consuming less than 1 nJ/image.

### 5.10.1 Improvements and future work

Several improvements to the APM design could be considered in the future implementations to reduce the effects of ringing, eliminate boundary effects and reduce the power consumption in the idle state. The schematic diagram of the improved propagation gate is presented in Figure 5.30.

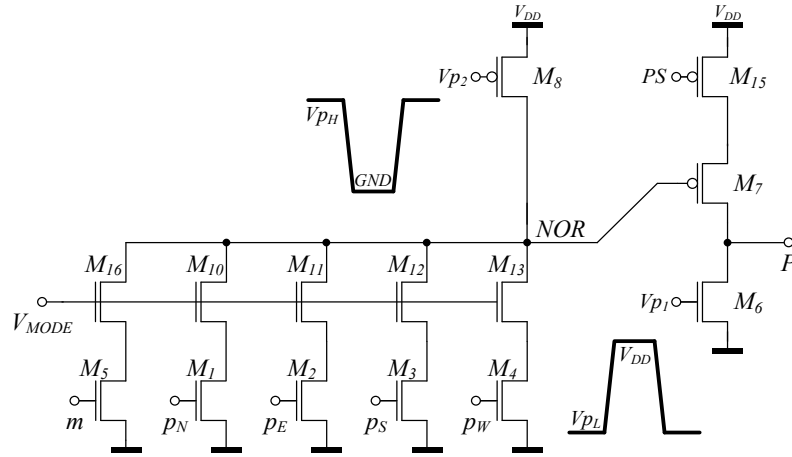


Figure 5.30. The improved design of the proposed propagation gate from Figure 5.8.

The observed power rail oscillations result mainly from the use of transistor  $M_5$ , pulling down the node  $NOR$  directly to ground when the weak keeping transistor  $M_8$  is still in operation. For a short while, before  $M_7$  will charge the output  $P$  to  $V_{DD}$ , transistors  $M_9$  and  $M_5$  (Figure 5.8) create a DC path between the power rails. In order to reduce this current, it is proposed to add another initialization signal  $V_{p2}$  to control the gate of  $M_8$  separately. Both transistors  $M_6$  and  $M_8$  are controlled individually by signals  $V_{p1}$  and  $V_{p2}$ , initializing the cell and working as weak keepers during the evaluation phase with gate bias voltages  $V_{pL}$  and  $V_{pH}$  respectively. This can also reduce the leakage current in the idle state when  $M_6$  and  $M_8$  are turned off.

The border effects resulting from the unbalanced load of the cells located around the array can be solved by inserting dummy cells around the array, or by inserting an

additional transistor  $M_{15}$ , controlled by bit  $PS$  (propagation space), switching on or off the output stage. For  $PS$  in high logic state, transistor  $M_{15}$  will be turned off and the gate will not generate propagation signal. Setting the border cells into such state will resolve the symmetry problems. The propagation space is also used in many morphological operations, therefore, such design extension would be beneficial. The current limiting transistor  $M_9$  (Figure 5.8), used solely to control the timing parameters of the gate can be removed, since proper biasing of transistors  $M_{10-13}$  is sufficient to assure circular wave propagation. It is also suggested to add transistor  $M_{16}$  in series with  $M_5$  to improve the timing parameter uniformity and further reduce the power rail oscillations.

For designs with strict power constraints, the quiescent current can further be reduced by supply voltage scaling or by using high threshold voltage and low leakage MOS transistors with thicker gate oxide. It should be noted, however, that supply voltage scaling will affect (reduce) the propagation speed and the use of MOS devices other than regular may result in higher parameter mismatch, requiring larger transistors to assure correct operation of the array. Therefore, the use of the design kits dedicated for low leakage purposes should rather be considered in practical implementations of such processor arrays.

## Chapter 6

---

# Wave propagation concept in arbitrary metrics

---

### 6.1 Chapter overview

This chapter extends the discussion on the trigger-wave propagation in asynchronous CMOS arrays, presented in Chapter 5. The propagation mechanism is considered in the context of isotropic propagation in spaces employing different distance measure norms. Theoretical analysis of the propagation mechanism and the proposed simplified timing model of the propagation gate are verified in circuit simulations, and are confirmed in the experiments with the fabricated prototype chip.

### 6.2 Introduction

Shape recognition usually involves medium level image processing algorithms requiring global operations such as distance transformation (DT), skeletonization or Voronoi tessellation. An interesting approach to global image attributes extraction, based on the trigger-wave propagation and wave-front collision detection concepts, was considered in the evaluation of the medial axis function (MAF) [Blum 67], and later practically observed in chemical solutions reacting with incident light (the Belousov-Zhabotinsky reaction) [Kuhnert 89], [Krinsky 91]. Ideally, such waves (autowaves), when triggered from the edges of an object, propagate isotropically with a constant speed in every direction, utilizing the locally stored energy of a medium, and collide or bend

denoting the medial axis points [Blum 67] (see Sections 5.2 and 5.3 in Chapter 5). In this chapter propagation mechanism is considered in the context of isotropic propagation in spaces operating in different distance measure norms.

There are several characteristic norms, typically used in image processing, such as the Euclidean norm, Manhattan norm and Chessboard norm [Borgefors 86], being particular cases of a generic  $p$ -norm, defined by a real number  $p$  (with a constraint  $p \geq 1$ ) and, for a 2-dimensional vector  $(x, y)$ , given by the formula:

$$\|(x, y)\|_p = \left( |x|^p + |y|^p \right)^{\frac{1}{p}} \quad (6.1)$$

Assuming the isotropic propagation in different  $p$ -norm spaces, the wave-front contours triggered from a single point are equidistant from that point, and hence, their shape corresponds to a particular value  $p$ . For example, for  $p = 2$ , the norm describes Euclidean space and the resulting contours are circular. The shapes of the propagation waves in typical distance measure norms are presented in Figure 6.1.

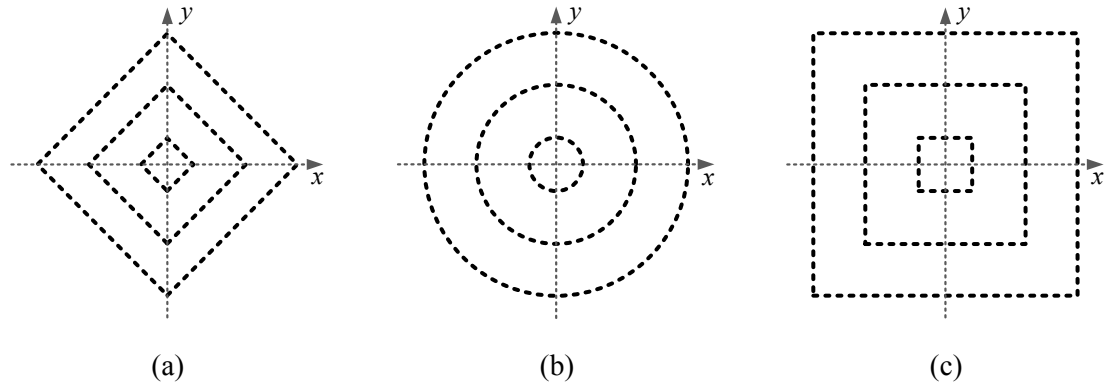


Figure 6.1. Contours of the 2-dimensional propagation waves in different  $p$ -norms: a) Manhattan ( $p = 1$ ), b) Euclidean ( $p = 2$ ), Chessboard ( $p \rightarrow \infty$ ).

Since the Euclidean metric is the most "natural" to use, several algorithms for calculating the approximate Euclidean distance measure were considered in [Montanari 68], [Danielsson 80] and [Borgefors 86]. Also, hardware oriented approach for Single Instruction Multiple Data (SIMD) fine-grain processor arrays was presented in [Razmjooei 2010]. Direct hardware implementations of the trigger-wave propagation mechanism, using asynchronous logic arrays (presented in Chapter 4), were previously discussed in [Eklund 96] and [Dudek 2006]. Such arrays provide fast and energy efficient computational engine for image processing algorithms, e.g. hole filling, geodesic reconstruction, closed shape detection, where the correct operation is independent of the assumed metric. Also, the CNN implementations using the trigger-wave propagation,

usually do not consider the distance measure norm applied in image processing tasks [Rekeczky 99]. Some algorithms, however, such as distance transformation and skeletonization, typically require a circular (Euclidean) propagation [Blum 67], when using the trigger-wave and collision detecting scheme.

Rounded shapes of the trigger-wave contours in asynchronous VLSI hardware realisations of processor arrays were reported in [Dudek 2006] and [Lopich 2010], and in CNN implementation in [Carmona-Galan 2003]. Attempts aiming the implementation of wave-front collision detection in CNN were presented in [Rekeczky 99], however, the contours of the waves in the propagation layer were highly distorted.

### 6.3 Propagation and timing analyses

The analysis of the propagation mechanism and the timing parameters will be presented assuming a regular 2-D array with only four nearest neighbours connectivity with a constant pixel pitch  $x$ . Two cases of the wave-front propagation: in the cardinal direction (A), and in the diagonal direction (B), triggered from a reference point  $O$  will be considered (Figure 6.2).

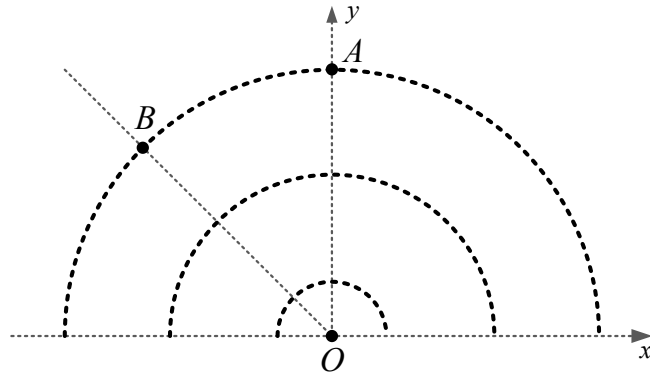


Figure 6.2. The propagation of the wave triggered from the point  $O$  in the cardinal and in the diagonal directions considered in points  $A$  and  $B$  respectively.

The cells located in the cardinal directions are always triggered from only one neighbour and the wave-front propagates with a constant cell-to-cell speed  $v_A$ . The cells located in the diagonal directions, are triggered from two neighbours simultaneously and propagate the signal with the higher respective cell-to-cell speed  $v_B > v_A$ . As a result, wave triggered in a circuit array tends to accelerate towards the diagonal directions, which makes the propagation contours more circular [Dudek 2006]. More detailed

analysis of the propagation mechanism considered in points  $A$  and  $B$  in Figure 6.2, is shown in Figure 6.3.

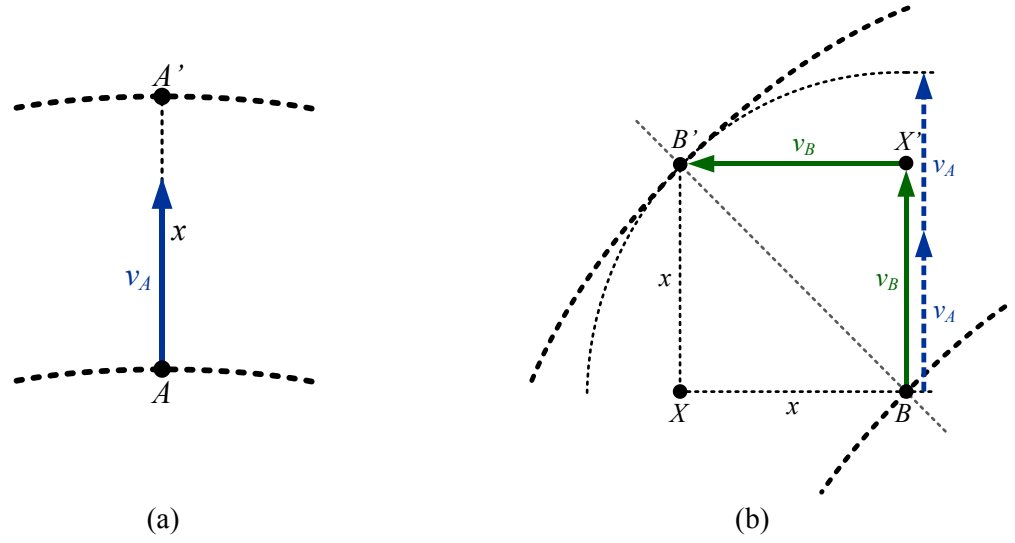


Figure 6.3. The mechanism of the wave propagation in a regular four-connected circuit array in the: a) cardinal, b) diagonal directions.

For a cardinal direction, assuming that a wave travels distance  $x$  with a constant speed  $v_A$ , the propagation time  $T_C$  can be calculated as (Figure 6.3a):

$$T_C = \frac{x}{v_A} \quad (6.2)$$

For a diagonal direction (Figure 6.3b), the wave propagates through its nearest neighbours according to the assumed rectangular structure of the array with only four neighbourhood connectivity passing the distance  $2x$ . Since the propagation in this direction is triggered from two neighbours simultaneously, the resulting speed in cardinal directions  $v_B$  will be higher than  $v_A$  and the corresponding propagation time  $T_D$  equals:

$$T_D = \frac{2x}{v_B} \quad (6.3)$$

In the wave propagation context, assuming isotropic medium and constant propagation speed, the distance can be determined by means of the propagation time. Therefore, the diagonal distance  $d = |BB'|$  in Figure 6.3b can be calculated from the propagation time ratio  $d = (T_D/T_C)x$ , which leads to the following relation:

$$d = \frac{2x}{v_B / v_A} \quad (6.4)$$

Assuming that the array operates in an arbitrary metric defined by the parameter  $p$  of the  $p$ -norm in (6.1), the distance  $d$  between two diagonal neighbours in the regular array with the pixel pitch  $x$  is given by:

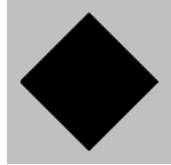
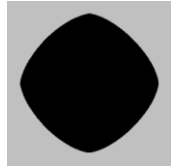
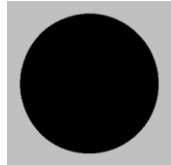
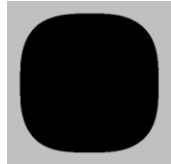

$$d = 2^{1/p} x \quad (6.5)$$

Combining equations (6.4) and (6.5) the following relation can be derived:

$$\gamma = 2^{1-1/p} \quad (6.6)$$

where  $\gamma = v_B/v_A$  is the speed ratio parameter and  $p \in [1, \dots, \infty]$  defines the  $p$ -norm from (6.1). The propagation speed  $v_A$  and  $v_B$  can be determined from the timing parameters of the circuit array, therefore, the equation (6.6) combines the circuit parameters with the geometric properties of the generated wave contours. Several characteristic isotropic propagation contours, generated numerically for particular distance measured norms defined by parameters  $p$  and  $\gamma$ , are presented in Table 6.1.

Table 6.1. Isotropic propagation contours in different distance measure norms.

<i>Parameters <math>p</math> and <math>\gamma</math></i>	<i>Distance measure norm</i>	<i>Wave contour</i>
$p = 1$ $\gamma = 1$	Manhattan (City Block)	
$1 < p < 2$ $1 < \gamma < \sqrt{2}$	Manhattan/Euclidean	
$p = 2$ $\gamma = \sqrt{2}$	Euclidean	
$p > 2$ $\gamma > \sqrt{2}$	Euclidean/Chessboard	
$p \rightarrow \infty$ $\gamma = 2$	Chessboard	

## 6.4 Simplified switched $RC$ model

In order to verify the relation between the timing parameters of the circuit array and the properties of the generated wave-fronts, the operation of an asynchronous array, consisting of simplified switched  $RC$  propagation gates, will be considered. In particular, the timing parameters of the propagation gate presented in Figure 6.4, corresponding to the CMOS implementation from Figure 5.8 in Chapter 5, will be analyzed and used in the calculations of parameter  $\gamma$ .

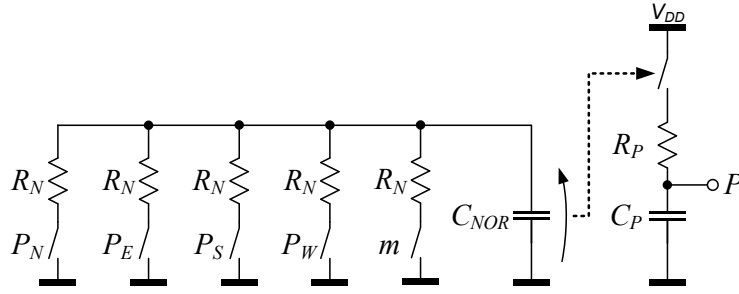


Figure 6.4. Schematic diagram of the ideal switched  $RC$  propagation gate.

The initial conditions, specified for the proposed circuit array, assure the state just after initialisation phase where the capacitor  $C_{NOR}$  is charged to  $V_{DD}$  and the capacitor  $C_P$  is discharged (the output  $P$  is at the zero logic level). This is the state just before the propagation phase. Transistors  $M_{1-5}$  (nMOS) and  $M_7$  (pMOS) from Figure 5.8 are implemented as ideal voltage controlled switches with fixed channel resistances  $R_N$  and  $R_P$  respectively. A switch turns on when its corresponding control voltage (any of the input signals for switches  $P_N$ ,  $P_E$ ,  $P_S$ ,  $P_W$  and  $m$ , or the voltage across  $C_{NOR}$  for switch  $R_P$ ) exceeds a certain threshold. When this is the case, for any of the input signals, the respective switch turns on discharging  $C_{NOR}$  with a time constant  $\tau_1 = R_N C_{NOR}$ . When any two inputs are driven simultaneously, the capacitance  $C_{NOR}$  discharges at twice the speed with a time constant  $\tau_1/2$ . When the voltage across  $C_{NOR}$  falls below a certain value, the switch in the second stage turns on, charging the output capacitance  $C_P$  with a time constant equal to  $\tau_2 = R_P C_P$ . When the rising slope of the output signal  $P$  crosses the threshold voltage of the input switches, all the neighbouring gates will be triggered and the mechanism of the propagation will continue. The proposed circuit realisation of the gate consists of two stages, thus the propagation speed is inversely proportional to the sum of respective time constants  $v \sim 1/(\tau_1 + \tau_2)$ . Based on this relation, the speed ratio  $\gamma$  for the propagation speeds  $v_A$  (when only one out of four switches  $P_N$ ,  $P_E$ ,  $P_S$ ,  $P_W$  closes), and

$v_B$  (when two out of four switches closes simultaneously), of the proposed switched  $RC$  circuit equals to:

$$\gamma = \frac{v_B}{v_A} = \frac{\tau_1 + \tau_2}{\tau_1/2 + \tau_2} = \frac{R_N C_{NOR} + R_P C_P}{R_N C_{NOR}/2 + R_P C_P} \quad (6.7)$$

For example, assuming that both of the time constants  $\tau_1$  and  $\tau_2$  are equal, the speed ratio is  $\gamma \approx 1.33$  which means that the observed propagation contour will be close to a circle, and the array will operate in the approximate Euclidean metric (see Table 6.1).

The operation of the proposed ideal switched  $RC$  model of the propagation layer was verified in simulations using Hspice. Certain input circuit parameters such as supply voltage  $V_{DD} = 1$  V, *ON/OFF* resistance of the switches equal 10k $\Omega$ /1G $\Omega$ , and the basic time constant  $\tau_1 = 1$  ns were chosen arbitrarily. In order to implement switched resistance  $R_N = 10$  k $\Omega$  with a threshold voltage 500 mV, a *VCR* (voltage-controlled resistor) element with resistance attribute changing from 1 G $\Omega$  (switch open) to 10 k $\Omega$  (switch closed) within the control voltage range from 499.5 mV to 500.5 mV was used (half of the initial voltage on charged  $C_{NOR}$  capacitance). The values of the  $RC$  elements were calculated to assure fixed propagation time of each stage of 1 ns ( $R_N = 10$  k $\Omega$  and  $C_{NOR} = C_P = 144.27$  fF). Inserting the assumed numerical values, the parameter  $\gamma$  can be set to any value between 1 and 2, depending on  $R_P$ , given in k $\Omega$ , according on the equation:

$$\gamma = \frac{1 + (R_P / 10\text{k}\Omega)}{0.5 + (R_P / 10\text{k}\Omega)} \quad (6.8)$$

The snapshots of the propagation contours, obtained from the simulations of the array consisting of  $33 \times 33$  switched  $RC$  delay gates, when triggering a wave from the centre, are shown in Fig 6.5. It can be observed that different wave contours are generated depending on the value of  $R_P$ , and hence on the parameter  $\gamma$ , corresponding well with the shapes shown in Table 6.1.

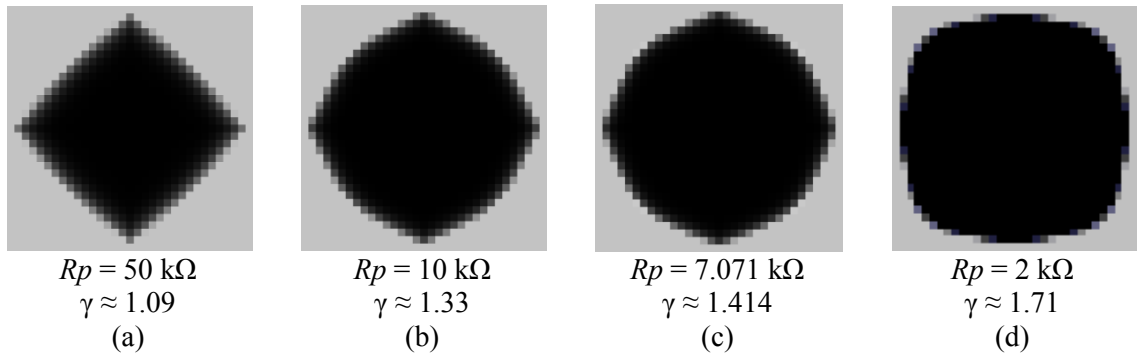


Figure 6.5. Propagation contours observed in the  $33 \times 33$  cell array of the proposed switched  $RC$  gates for different  $\gamma$  values.

## 6.5 CMOS design and experimental results

In the proposed design of the switched  $RC$  gate, the propagation speed and the generated wave contours depend solely on the value of  $R_P$  and the time constant of the second stage  $\tau_2 = R_P C_P$ . In particular, for very large values of  $R_P$ , the time constant  $\tau_2 \gg \tau_1$  and dominates the resulting cell-to-cell propagation speed, where the contribution of  $\tau_1$  becomes negligible. This is typical to any synchronous implementation of the propagation mechanism, where the propagation speed is strictly denoted by a clock period. When reducing the value of  $R_P$ , the time constant  $\tau_2$  becomes lower and the contribution of the first stage in propagation becomes more dominant. This makes the corresponding cell-to-cell speed more dependent on the operation of the first stage of the propagation gate, depending on the number of triggering neighbours. As a result, the propagation across the diagonal direction accelerates and makes the wave contours more circular. In such model, in order to achieve any rounded shape of the propagation contour, e.g. to achieve propagation in approximate Euclidean metric, the corresponding time constant of the second stage  $\tau_2$  has to be smaller than  $\tau_1$ . Since in practical realisations it seems much easier to elongate the propagation time of a logic circuit rather than shorten it, in the CMOS implementation of the propagation gate presented in Figure 5.8 in Chapter 5, the parameter  $\gamma$  is controlled using additional current limiting transistors  $M_{9-13}$  in the first stage, slowing its propagation time according to the bias voltages  $V_{MODE1}$  and  $V_{MODE2}$ . In particular, transistors  $M_{10-13}$  (in series with  $M_{1-5}$ ) increase the corresponding resistances  $R_N$  of each pull-down branch in the simplified switched  $RC$  model in Figure 6.4, which increases the time constant  $\tau_1$  depending on  $V_{MODE2}$ . Transistor  $M_9$ , controlled by the voltage  $V_{MODE1}$ , limits the total current discharging the corresponding capacitance  $C_{NOR}$ , which makes  $\tau_1$  less dependent on the number of the triggering neighbours. A similar effect can be observed in the proposed switched  $RC$  model when the time constant  $\tau_2$  is much longer than  $\tau_1$ .

In the experiments, the same measurement setup and methodology, as presented in Chapter 5, was used. The obtained images showing snapshots of the waves triggered from the centre of the array for different bias voltages  $V_{MODE1}$  and  $V_{MODE2}$ , tuned to achieve operation in approximate Manhattan, Manhattan-Euclidean, Euclidean and Chessboard norms, are shown in Figure 6.6. The results showing the extracted Voronoi diagrams, based on the collisions of the propagation waves triggered from several pixels in the array, operating in four different distance measure norms, are presented in

Figure 6.7. The corresponding bias voltages used in the experiments are grouped in Table 6.2 (the supply voltages were as given in Table 4.2 in Chapter 4).

Table 6.2. Bias voltages of used to achieve different distance measure norms.

<b>Parameter</b>	<b>Manhattan</b>	<b>Mixed</b>	<b>Euclidean</b>	<b>Square</b>
$V_{DELAY}$	372 mV	496 mV	396 mV	343 mV
$V_{MODE1}$	279 mV	930 mV	315 mV	371 mV
$V_{MODE2}$	1000 mV	930 mV	511 mV	359 mV

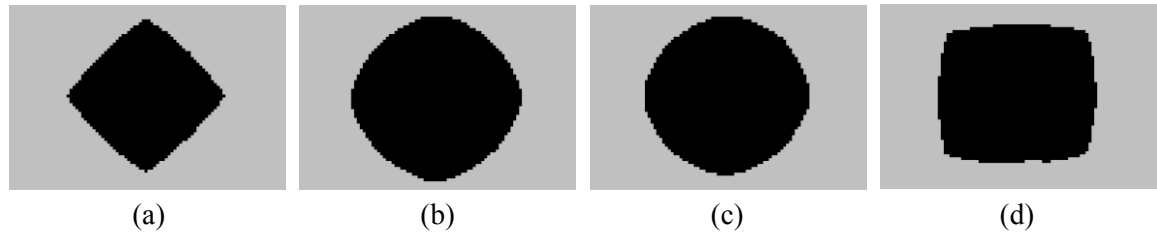


Figure 6.6. Propagation contours observed for different bias voltages  $V_{MODE1}$  and  $V_{MODE2}$  resulting in a) approximate Manhattan, b) mixed Manhattan-Euclidean, c) approximate Euclidean, and d) approximate Chessboard metrics.

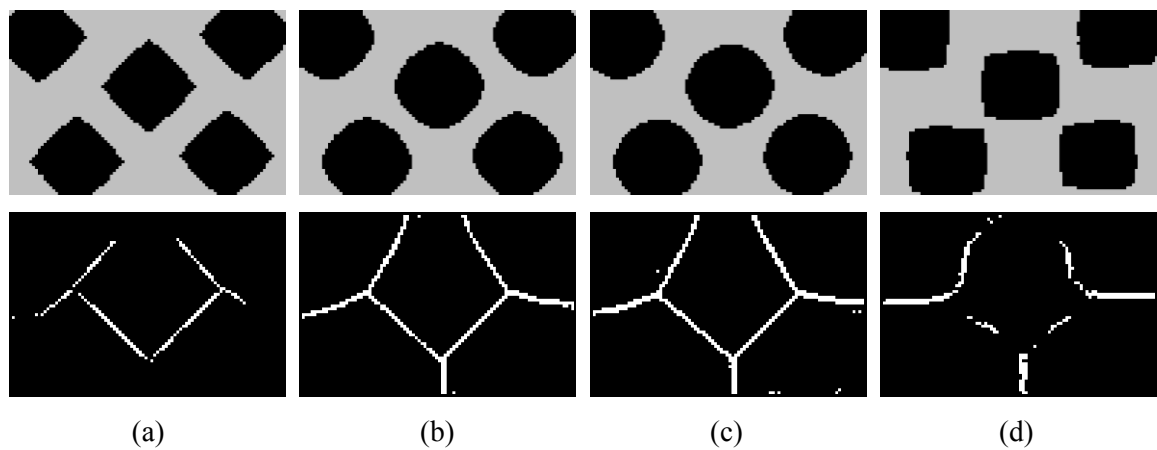


Figure 6.7. The results of binary image tessellation evaluated in different distance measure norms: a) approximate Manhattan, b) mixed Manhattan-Euclidean, c) approximate Euclidean, d) approximate Chessboard.

## 6.6 Conclusions

Based on the obtained experimental results, it can be observed that the Voronoi diagrams and skeletons (discussed in Chapter 5) of the best quality, can be extracted when array operates in approximate Euclidean metric. In such a case, the waves collide frontally, which is less confusing for the collision detection mechanism employed in the design. This can also be observed for the mixed Manhattan-Euclidean metric, assuming bias voltages  $V_{MODE1} = V_{MODE2} = V_{DD}$ . Therefore transistors  $M_9 - M_{13}$  could be removed from the design, if the area constraints were more strict, however, losing the control of

the wave contour. In particular, transistor  $M_9$ , limiting the total current discharging the first stage of the propagation gate, could be removed from the design, since it is only necessary to assure operation in Manhattan metric (typical to synchronous systems). It can be concluded that transistors  $M_{10-13}$  are sufficient to assure the operation in approximate Euclidean metric and, when controlled individually, can also be used as switches defining the space and the direction of the propagation, allowing the implementation of variety of image processing algorithms discussed in [Astrom 96].

## Chapter 7

---

# Probability and reasoning

---

### 7.1 Introduction and chapter overview

This chapter introduces the notions of probability, uncertainty and reasoning in networks modelling systems with cause-effect relationships between the variables. In particular, it discusses the probabilistic calculus, conditional probability and Bayes' rule, and its applications in reasoning under uncertainty in Bayesian networks. The theory of probability and the probabilistic reasoning have been a subject of many publications. The mathematical foundations for the methods and the algorithms used in Bayesian inference can be found in [Pearl 88], [Neapolitan 2004], [Jensen 2007] and [Darwiche 2009].

Two methods: one for the exact inference, based on the chain rule (used as reference in this research), and one for the approximate inference, using belief propagation approach, will be further discussed in detail. Methods of exact inference usually exhibit a very high computational complexity, growing nonlinearly with a network size [Cooper 90], therefore, they are typically not considered in hardware realisations, unless the network is very small. The approximate methods require less extensive computation, and their complexity scales slower with the network size [Jensen 2007]. Since in many applications approximate inference provides sufficient accuracy, approximate methods are frequently considered in practical realisations. The background knowledge provided in this chapter will be of use in Chapter 8, dealing with analogue and digital hardware realisations of the sum-product algorithm for Bayesian inference in analogue hardware realisations of factor graphs.

## 7.2 Conditional probability and Bayes' rule

The probability  $P(A)$  of a particular outcome  $A$  in an experiment is always conditioned on all other known factors, which may affect the result of the experiment. For example, when tossing a coin, the probability of getting heads or tails is  $1/2$ , assuming that the coin is fair. A conditional probability  $P(A|B)$ , defining the probability of event  $A$  conditioned on the result of  $B$ , can be calculated using the fundamental rule given by the equation:

$$P(A|B) = \frac{P(A, B)}{P(B)} \quad (7.1)$$

where  $P(A, B) = P(A \cap B)$  is the probability of events  $A$  and  $B$  occurring simultaneously. From the law of alternation, the probability  $P(A, B)$  equals  $P(B, A)$ . Applying this to (7.1), multiplying and dividing the right hand side of (7.1) by  $P(A)$ , assuming that  $P(A) \neq 0$ , the following formula can be obtained:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (7.2)$$

The equation (7.2) is known as Bayes' rule of inverse probability. It says how an initial or a subjective knowledge of an event  $A$ , represented by  $P(A)$ , can be improved given the observation  $B$ . In particular, in the equation (7.2),  $P(A)$  is called the *prior probability* of event  $A$ , usually estimated based on some "prior knowledge", resulting from an experiment, experience or guess. The inverse conditional probability of  $A$ , given by  $P(B|A)$ , is the *likelihood* of  $A$  given  $B$  (equivalent to the conditional probability of  $B$  given  $A$ ), and  $P(B)$  is the probability of the conditioning event  $B$ . The calculated conditional probability  $P(A|B)$  is called *posterior probability* of  $A$  given  $B$  or *belief* of  $A$ , accounting for the observation  $B$  [Jensen 2007].

Bayes' rule is essential in statistics but also finds applications in science, medicine and engineering. The probability updating, known as *Bayesian inference*, can be used in computer vision [Chow 68], [Koeser 2004], [Richardson 72], [Geman 85], [Felzenszwalb 2006], robotics [Zhou 2007], [Lee 2009], bioinformatics [Lin 2010], [Friedman 2004] navigation and tracking [Bergman 99], search for lost objects [Frost 96], medicine and health care [Beinlich 89], [Olesen 89], [Kim 87], and administration and management [Acid 2004].

### 7.3 Bayesian networks

Bayes' rule, in its canonical form given by equation (7.3), is applicable to simple cases, where the underlying models consist of just one observation conditioned on a particular set of events. More elaborate systems, including many mutually dependent observations, are usually represented in a graphical form of *causal networks*. In such networks, events are associated with the variables or nodes, and are connected by links representing the underlying cause-effect relationships. Bayesian networks are particular class of causal networks. They consists of a set of variables (nodes)  $\{A_1, \dots, A_n\}$ , representing particular observations or events, and a set of directed edges (links, arcs), indicating causal relations between these variables. The links always point from a *parent* to the *child* node, indicating the direction of causation [Pearl 88]. No cycles in causation are assumed, which means that starting from any node and following the path denoted by directed links, one must not come back to the same node. The structure of the network is mathematically defined as a *directed acyclic graph* (DAG). Depending on the number of states, variables can be discrete, with a countable and finite number of mutually exclusive states, or continuous, with an infinite (continuum) number of states. In many practical cases, networks including discrete variables are typically used. In some applications, networks with continuous variables or hybrid networks with mixed type of variables are considered, however, often limited to account only for the Gaussian probability density functions [Neapolitan 2004]. Each variable in a network has its individual *conditional probability distribution* (CPD) representing the strength of the relations between a node and its parents. If a variable and its parents are discrete, its CPD becomes a *conditional probability table* (CPT). In the case of root variables, which do not have parents, the corresponding CPTs are simpler and represent only the prior probabilities.

An exemplar Bayesian network, illustrating the relations between four observations: *Cloudy* ( $C$ ), *Sprinkler* ( $S$ ), *Rain* ( $R$ ) and *Wet Grass* ( $G$ ) is shown in Figure 7.1. It consists of four variables  $\{C, S, R, G\}$  with two states  $\{T, F\}$  referring to *true* and *false* respectively. It describes a simple, real life system indicating potential reasons for wet grass ( $G$ ), caused either by working sprinkler ( $S$ ) or rain ( $R$ ). The prior and conditional probabilities of the system can be obtained based on experience, analysis or in the process of parameter learning from statistical data. In this network, the occurrence of cloudy sky ( $C$ ), with prior probability of 0.2 (e.g. estimated based on the weather

patterns), can frequently cause rain, therefore, the probability of such a case is 0.8. Sometimes, when the clouds are very thick, they can occasionally dim the sunlight and trigger the sprinkler ( $S$ ), which is supposed to wet the grass after the sunset. However, such situation is rare, therefore, its probability is equal to 0.1. Since both rain and sprinkler can make the grass wet, node  $G$  has two parents:  $S$  and  $R$ , and accordingly larger CPT accounting for  $2^3 = 8$  possible cases of the conditional probability  $P(G|S,R)$ . In such network, the probability of each node is represented by a discrete density function with two points corresponding to the probabilities of the states *true* and *false*.

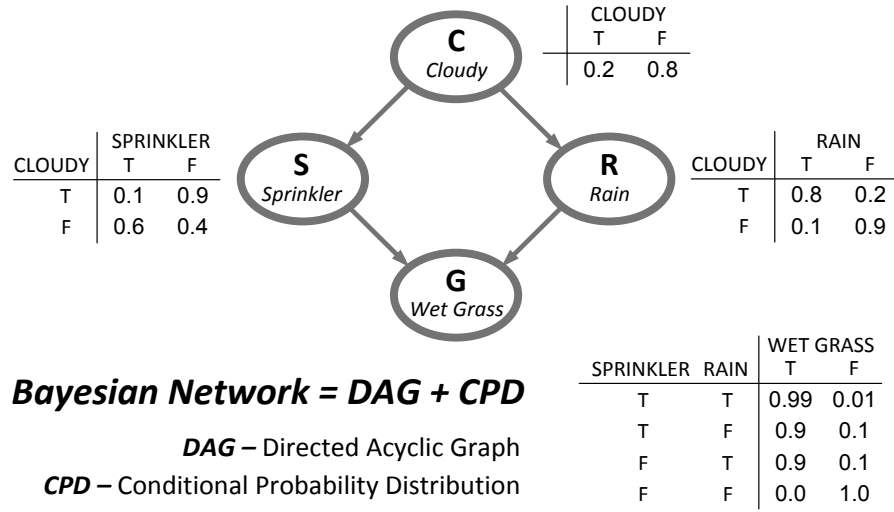


Figure 7.1. Example Bayesian network representing system consisting of four two-state variables *Cloudy* ( $C$ ), *Sprinkler* ( $S$ ), *Rain* ( $R$ ) and *Wet Grass* ( $G$ ).

## 7.4 Bayesian inference

The task of Bayesian inference is to compute posterior probabilities of the network nodes, given the incorporated knowledge (i.e. the network structure and the conditional probabilities), and accounting for the inserted observations. Such observations can instantiate nodes to particular fixed states. Whenever a new observation is received, the state of the network changes and new posterior probabilities of the nodes (beliefs) can be calculated. This makes Bayesian inference particularly useful in answering probabilistic queries. By inserting evidence or hypothetical observations into a network, the behaviour of the modelled systems can be examined under different assumptions, usually not possible to verify in practice. For example, clamping the variable  $G$  to state  $T$ , inherently assumes that the grass is wet and, based on that, the probability of cloudy sky  $P(C|G^T)$  can be computed. From equations (7.1) and (7.2), considering the simplest case of two events  $A$  and  $B$ , it can be seen that the corresponding conditional probabilities can be

calculated using *joint probability*  $P(A, B)$ , optionally *marginalised* in order to obtain probabilities of single observations  $P(A)$  or  $P(B)$ . Assuming that a Bayesian network is defined for a particular *universe* of variables  $U = \{A_1, A_2, \dots, A_n\}$ , its unique joint probability distribution  $P(U)$  can be calculated using chain rule, given by the product of the conditional probabilities specified for all the nodes [Jensen 2007]:

$$P(U) = \prod_{i=1 \dots n} P(A_i | \text{pa}(A_i)) \quad (7.3)$$

where  $\text{pa}(A_i)$  is the set of parents of node  $A_i$  in the network. Chain rule can directly be used in the calculations of the conditional probabilities of variables in the network. For example, the probability of variable  $A_1$  being in  $i$ -th state, and denoted as  $A_1^i$ , can be calculated as:

$$P(A_1^i) = \sum_{\{A_2, A_3, \dots, A_n\}} P(A_1^i, A_2, \dots, A_n) \quad (7.4)$$

where  $\{A_2, A_3, \dots, A_n\}$  indicates the summation over the possible states of variables  $A_2, A_3, \dots, A_n$ , whereas the state  $A_1$  is fixed to  $i$ . The probability of variable  $A_1$  being in  $i$ -th state, and conditioned on a variable  $A_2$  being in  $j$ -th state, can be calculated using the equation (7.1):

$$P(A_1^i | A_2^j) = \frac{\sum_{\{A_3, \dots, A_n\}} P(A_1^i, A_2^j, A_3, \dots, A_n)}{\sum_{\{A_1, A_3, \dots, A_n\}} P(A_1, A_2^j, A_3, \dots, A_n)} \quad (7.5)$$

The posterior probability of a node in a Bayesian network can be calculated assuming particular states of other variables, accounting also for the observations, evidence or assumptions given by the probabilistic queries. In such scheme, conditioning can only be done by instantiating selected variables into particular fixed states. For example, in the network from Figure 7.1, the variable *Rain* ( $R$ ) can either be *true* or *false*. Therefore, it is not possible to explicitly insert so called *likelihood evidence* into (7.5), saying for example that the variable *Rain* is 90% *true* and 10% *false*. The likelihood evidence can be dealt with using, for example, auxiliary nodes with variable CPTs representing the inserted evidence. The method of Bayesian inference, directly applying the equations (7.4) and (7.5) in the calculations of the conditional probabilities, is called *global marginalisation* and provides foundation for other methods of exact Bayesian reasoning. These methods usually attempt to optimise the process of variable elimination by pre-computing the probability tables to avoid redundant calculations. In particular,

methods such as variable elimination, following the *topological order* of the network (i.e. with respect to hierarchy moving from parents to children), avoid redundant computation and reduce the size of the joint probability tables [Jensen 2007]. Other methods operate directly on the structure of the network and implement node clustering or conditioning to optimise the process of variable elimination [Pearl 88]. In general, methods for exact inference try to reduce the amount of computation, and hence, to reduce the processing time but also keep balance between the speed and memory consumption. Since it has been proven that the complexity of exact inference quickly becomes intractable and grows nonlinearly with the size of a network [Cooper 90], more attention has been paid to the development of methods for approximate reasoning, such as belief propagation, and stochastic methods based on Monte Carlo approach.

## 7.5 Belief propagation

The idea of belief propagation in Bayesian networks lies in the properties of the graphical models defining conditional dependencies between nodes. In particular, recognising the conditional independencies between variables can simplify calculations, practically to the set of the nearest neighbourhood of each node. Such situation is illustrated in Figure 7.2 showing the neighbourhood of the variable  $A$  in a larger network.

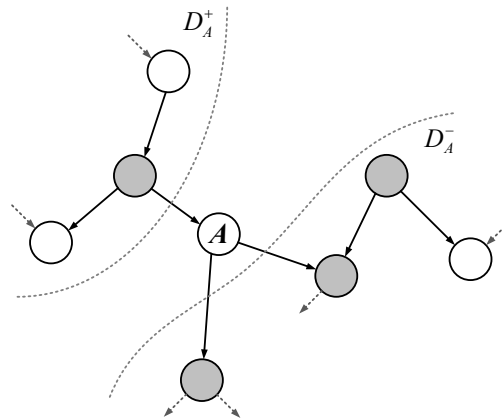


Figure 7.2. Neighbourhood of the node  $A$  in a singly connected network split into two conditionally independent parts.

The shaded nodes in Figure 7.2 denote the *Markov blanket* of the variable  $A$ , defined as a set consisting of the parents of  $A$ , the children of  $A$  and its children parents. When all the nodes belonging to the Markov blanket of  $A$  are instantiated,  $A$  is *d-separated* from the rest of the network, and hence, the state of  $A$  can be evaluated based solely on the

states of the neighbouring nodes [Jensen 2007]. This implies that only the local information received from the neighbours of  $A$  is necessary to compute belief of  $A$ . Assuming that the network, which part is shown in Figure 7.2, is *singly connected*, i.e. there exists at most one path between any two nodes, the variable  $A$  unambiguously separates the network into two sets including its predecessors  $D_A^+$  and successors  $D_A^-$  [Pearl 86, 88]. Based on that, the universe of all variables of this network can be presented as  $U = \{A, D_A^+, D_A^-\}$  and the joint probability, from the chain rule, is equal to  $P(U) = P(A | D_A^+) P(D_A^- | A)$ . The conditional probability of the variable  $A$ , for its particular state  $i$ , given the states of its successors and predecessors, can be calculated as:

$$P(A^i | D_A^+, D_A^-) = \frac{P(A^i, D_A^+, D_A^-)}{\sum_{\{A\}} P(A, D_A^+, D_A^-)} \quad (7.6)$$

The summation in the denominator of (7.6) is done over all states of the variable  $A$ , and the result of this operation is constant for a given network configuration, defined by  $D_A^+$  and  $D_A^-$ . Therefore, in the calculation of the probability distribution of  $A$ , being a vector of elements  $A$  with all possible states, the summation in the denominator in (7.6) can be omitted, and obtained vector can be normalised. The posterior probability distribution of  $A$ , can be calculated based on equation (7.6), using the following formula:

$$Bel(A) = \alpha P(A | D_A^+) P(D_A^- | A) \quad (7.7)$$

where  $Bel(A)$  represents a vector of conditional probabilities computed using (7.6) for all possible states of  $A$ , and  $\alpha$  is the normalising factor. The components  $P(A | D_A^+) = \pi(A)$  and  $P(D_A^- | A) = \lambda(A)$  in (7.7) represent the top-down and bottom-up propagation of the probability density functions, sent respectively from the predecessors and successors of  $A$ . As a result, each link in the network conducts messages in both directions. The flow of the probabilistic information resembles the mechanism of propagation, where changes in belief of one node propagate across the network updating beliefs of other nodes. In such scheme, nodes operate as data-driven arithmetic processors, reacting to each new incoming message. If, as a result of such local computation, any of the outgoing messages will change, it will be detected by the neighbouring nodes, which in turn will perform appropriate calculations and send new messages to their respective neighbours. The process of belief propagation stops when the network attains equilibrium and no new updates for the output messages can be generated.

In belief propagation mechanism in singly connected networks, the computed beliefs always converge to the posterior probabilities of the variables [Pearl 88]. When a network contains *loops*, i.e. there exists more than one path between two nodes (e.g. in the network in Figure 7.1 there are two paths between nodes  $A$  and  $D$ ), it is no longer singly connected, and the local propagation scheme may occasionally become problematic. It should be noted that these loops refer to the structure of the network, not loops in causation. In the networks with such structural loops, the messages exchanged between nodes may circulate indefinitely, also leading to oscillations, and the process may not converge to a stable equilibrium. In fact, it has been shown that such oscillations are usually not observed in probabilistic networks modelling real systems [Pearl 88]. It should be noted, however, that multiply connected networks may, for some nodes, violate the conditional independencies between subsets  $D_A^+$  and  $D_A^-$ , assumed in the derivation of the belief propagation scheme, therefore, the asymptotic equilibrium, even if attained in such networks, may not always be coherent with the posterior probabilities of the variables. Nevertheless, it has been demonstrated that the corresponding error is often very small and can be neglected in many practical applications.

## 7.6 Factor graphs

*Factor graph* is a graphical model providing a convenient way of representing algorithms using complex global functions of many variables, which can be decomposed (*factored*) into a product of local functions requiring only local computation. In particular, the sum-product algorithm, based on the message the passing scheme, can be used in a variety of applications requiring computation of global functions [Loeliger 2004]. It has been shown that many tasks in signal processing, digital communication and artificial intelligence, can be represented as factor graphs and solved using particular adaptation of the sum-product algorithm. The merits of such approaches were demonstrated in the realisations of error correcting codes such as Viterbi [Shakiba 98] and BCJR [Moerz 2000] algorithms, iterative decoding using low density parity check (LDPC) codes [Srinivas 97], [Loeliger 2001], forward/backward algorithm for hidden Markov model, Kalman filtering, Fast Fourier Transformation and belief propagation in Bayesian networks [Kschischang 2001].

Factor graph representation of a Bayesian network provides a systematic and perhaps simpler approach to belief propagation mechanism, where the arithmetic operations

performed by each node, can be decomposed into two separate sub-blocks, one corresponding to the *variable* and one to the *factor* node. The factor graph representation of the network from Figure 7.1 is shown in Figure 7.3. In the figure, squares denote factor nodes and circles refer to variable nodes. It can be observed that each Bayesian node consists of two nodes, one factor and one variable, from the underlying factor graph.

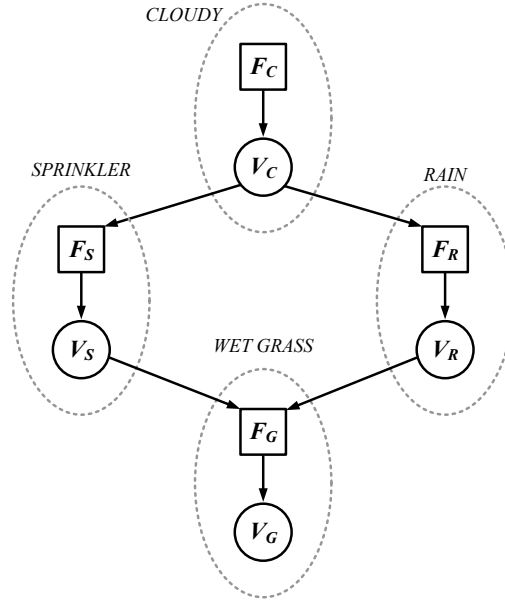


Figure 7.3. Factor graph representation of the Bayesian network from Figure 7.1.

In the following, one particular case will be considered in detail, where both factor and variable nodes communicate with only three neighbours and perform operations using two-state variables with elements from a set  $\{T, F\}$  denoting *true* and *false*. The connectivity and the corresponding arithmetic operations of such three-way factor and variable nodes are presented in Figures 7.4 and 7.5. Since variable nodes ( $V$ ) communicate only with factor nodes ( $F$ ), and factor nodes communicate only with variable nodes (see Figure 7.3), the factor node  $F$  in Figure 7.4 receives messages from its three neighbours  $V_1$ ,  $V_2$  and  $V_3$ , denoted as  $VF_1$ ,  $VF_2$ , and  $VF_3$ , and generates output messages  $FV_1$ ,  $FV_2$  and  $FV_3$  respectively. Assuming that nodes  $V$  and  $F$  form a Bayesian node  $N$ , and  $V_2$  and  $V_3$  belong to the parents of  $N$ , the factor node  $F$  performs matrix-vector multiplications, using conditional probabilities  $P(N|V_2, V_3)$ . The probability  $P^{ijk}$  refers to a particular entry from a CPT given by  $P^{ijk} = P(N^i | V_2^j V_3^k)$  where  $i, j$  and  $k$  define a particular configuration of states. For example, the probability of variable  $N$  in state  $T$ , denoted as  $N^T$ , given states of its parents  $V_2^T$  and  $V_3^F$  is  $P^{TTF} = P(N^T | V_2^T V_3^F)$ . The

messages sent to all three neighbours  $FV_1$ ,  $FV_2$  and  $FV_3$  of the factor node can be calculated using the following equations:

$$\begin{bmatrix} FV_1^T \\ FV_1^F \end{bmatrix} = \alpha_1 \cdot \begin{bmatrix} P^{TTT} & P^{TTF} & P^{TFT} & P^{TFF} \\ P^{FTT} & P^{FTF} & P^{FFT} & P^{FFF} \end{bmatrix} \cdot \begin{bmatrix} VF_2^T \cdot VF_3^T \\ VF_2^T \cdot VF_3^F \\ VF_2^F \cdot VF_3^T \\ VF_2^F \cdot VF_3^F \end{bmatrix} \quad (7.8)$$

$$\begin{bmatrix} FV_2^T \\ FV_2^F \end{bmatrix} = \alpha_2 \cdot \begin{bmatrix} P^{TTT} & P^{TTF} & P^{FTT} & P^{FTF} \\ P^{TFT} & P^{TFF} & P^{FFT} & P^{FFF} \end{bmatrix} \cdot \begin{bmatrix} VF_1^T \cdot VF_3^T \\ VF_1^T \cdot VF_3^F \\ VF_1^F \cdot VF_3^T \\ VF_1^F \cdot VF_3^F \end{bmatrix} \quad (7.9)$$

$$\begin{bmatrix} FV_3^T \\ FV_3^F \end{bmatrix} = \alpha_3 \cdot \begin{bmatrix} P^{TTT} & P^{TFT} & P^{FTT} & P^{FFT} \\ P^{TTF} & P^{TFF} & P^{FTF} & P^{FFF} \end{bmatrix} \cdot \begin{bmatrix} VF_1^T \cdot VF_2^T \\ VF_1^T \cdot VF_2^F \\ VF_1^F \cdot VF_2^T \\ VF_1^F \cdot VF_2^F \end{bmatrix} \quad (7.10)$$

where  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$  are the normalising factors ensuring probabilities sum to 1. It can be observed that each outgoing message sent to a particular neighbour is calculated based on the two input messages received from the two remaining neighbours, and particular configurations of the states in the column vector in (7.8)-(7.10) correspond to the respective configurations of the states of the fixed conditional probabilities.

In the case of the variable node  $V$ , the outgoing messages can be calculated using the following equations:

$$\begin{bmatrix} VF_1^T \\ VF_1^F \end{bmatrix} = \alpha_1 \cdot \begin{bmatrix} FV_2^T \cdot FV_3^T \\ FV_2^F \cdot FV_3^F \end{bmatrix} \quad (7.11)$$

$$\begin{bmatrix} VF_2^T \\ VF_2^F \end{bmatrix} = \alpha_2 \cdot \begin{bmatrix} FV_1^T \cdot FV_3^T \\ FV_1^F \cdot FV_3^F \end{bmatrix} \quad (7.12)$$

$$\begin{bmatrix} VF_3^T \\ VF_3^F \end{bmatrix} = \alpha_3 \cdot \begin{bmatrix} FV_1^T \cdot FV_2^T \\ FV_1^F \cdot FV_2^F \end{bmatrix} \quad (7.13)$$

where  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$  are the normalising factors. In Bayesian perspective, the belief of a variable  $N$ , representing its posterior probability distribution normalised by factor  $\alpha$ , can be calculated as a dot product accounting for all the messages received:

$$\begin{bmatrix} Bel^T \\ Bel^F \end{bmatrix} = \alpha \cdot \begin{bmatrix} FV_1^T \cdot FV_2^T \cdot FV_3^T \\ FV_1^F \cdot FV_2^F \cdot FV_3^F \end{bmatrix} \quad (7.14)$$

Equations (7.8) - (7.14) provide a complete set of mathematical operations performed by a particular realisation of Bayesian processor communicating with two parents and two children, and operating using two-state variables. Block diagram of such processor implemented in node  $N_l$  is shown in Figure 7.6. It exchanges messages with parents  $N_2$  and  $N_3$ , and with children  $N_4$  and  $N_5$ . Extension to a general case, accounting for an arbitrary number of parents, children, variable states of node  $N_l$  and its neighbours is straightforward and requires proper modifications of the equations (7.8) - (7.14).

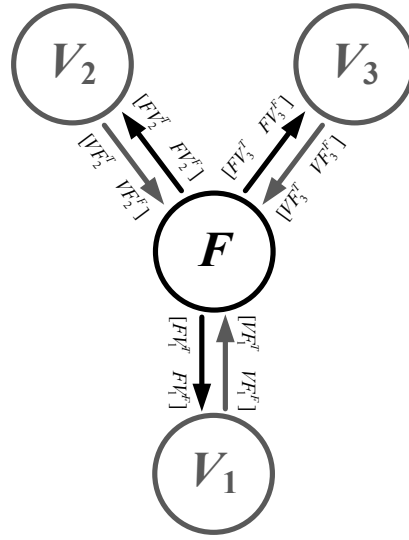


Figure 7.4. Connectivity of the factor node in factor graph implementation of belief propagation.

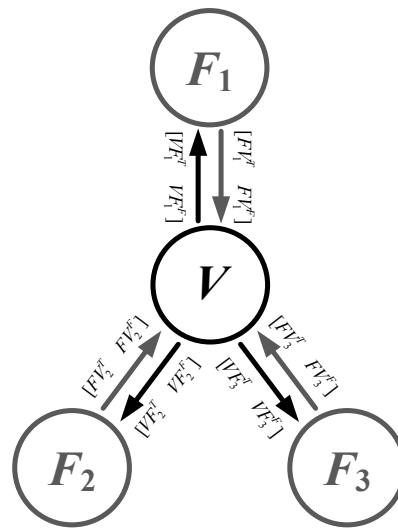
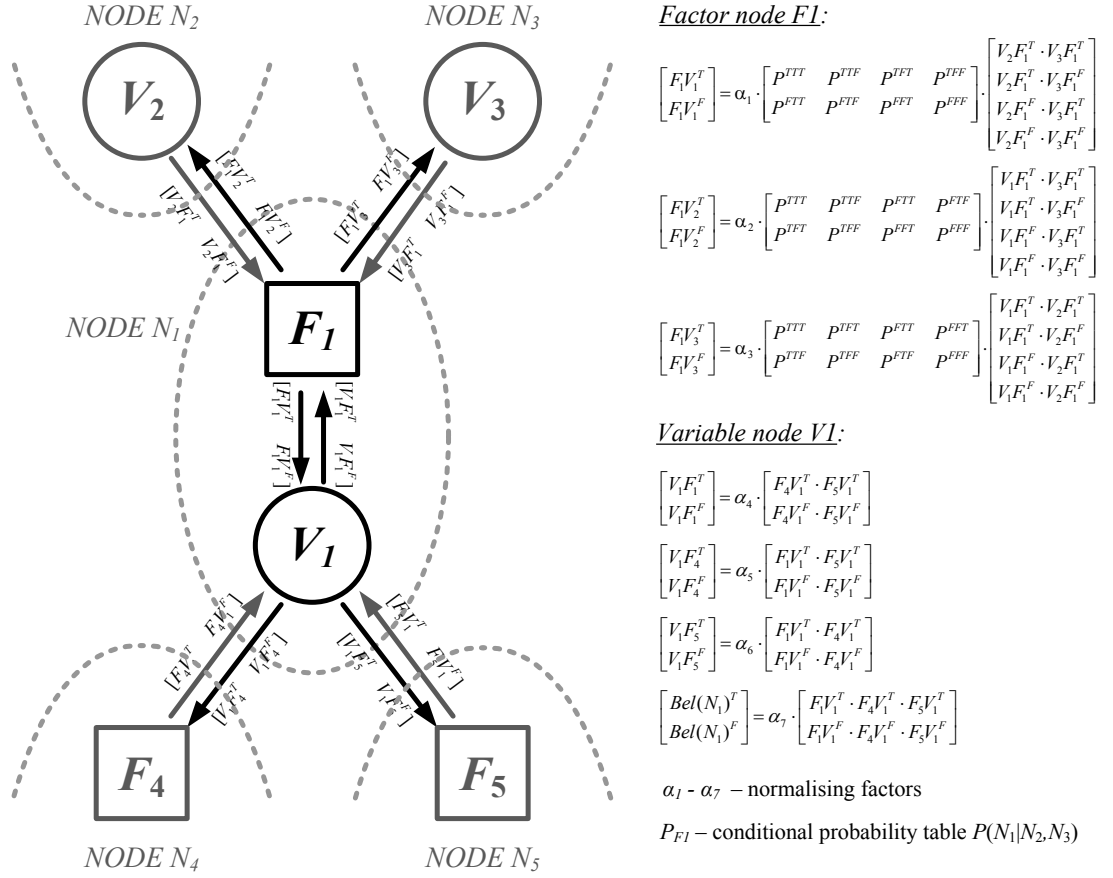


Figure 7.5. Connectivity of the variable node in factor graph implementation of belief propagation.

Figure 7.6 Block diagram of a Bayesian processor  $N_1$  with parents  $N_2, N_3$  and children  $N_4, N_5$ .

## 7.7 Conclusions

This chapter discussed the notion of the conditional probability, introduced Bayes' rule of inverse probability, and provided mathematical foundations for reasoning in networks representing cause-effect relationships. In particular, algorithm for belief propagation method was discussed in detail providing the background knowledge for its hardware and software implementations in factor graphs. The presented mathematical description of the sum-product algorithm and the message passing scheme will be used in Chapter 8 in the realisations of analogue and digital arithmetic circuits dedicated for probabilistic reasoning in Bayesian networks.

## Chapter 8

---

# VLSI systems for Bayesian inference

---

### 8.1 Chapter overview

This chapter discusses two design concepts of analogue CMOS arithmetic circuits using the Gilbert multiplier cell, presented in Chapter 3, in continuous-time and discrete-time hardware realisations of belief propagation in factor graphs. Design issues, such as computational accuracy, power consumption, processing speed, area occupation and complexity scaling are further investigated. The performance of the analogue solutions is compared with the equivalent digital hardware, synthesised using the same CMOS technology, and with two software implementations on PC. The obtained figures of performance provide a baseline comparison between the realisations of the same computationally demanding task, using different hardware architectures and operating according to different principles.

### 8.2 Introduction

Message passing and belief updating in the sum-product realisation of Bayesian inference in factor graphs, discussed in Chapter 7, is an iterative process requiring high processing power. It becomes particularly important in applications involving complex network structures [Lin 2010] or real-time operation [Felzenszwalb 2006]. The idea of continuous-time processing in the implementation of the sum-product algorithm was initially considered in [Haygenauer 98]. Its VLSI hardware realisations in BiCMOS [Hagenauer 2002], [Moertz 2000], [Lustenberger 99a, 99b, 2001] and in standard CMOS

technologies [Loeliger 2001] became a promising alternative to the classical systems in applications related to the iterative decoding in digital communication. The power and speed performance of a dedicated analogue computing circuit was reported almost two orders of magnitude better than in the case of its digital equivalent [Loeliger 99].

Although the analogue solutions provide very promising figures of performance, there are many design issues in standard CMOS technologies affecting the operation of such circuits and impeding scaling for more complex problems. In particular, the impact of parameter variability on the correct operation of the analogue decoders was discussed in [Lustenberger 2001]. The research showed a high robustness of such circuits to fabrication mismatch. It should be noted, however, that these systems process only binary data. Therefore, their sensitivity to parameter mismatch is usually lower than in the case of the circuits processing analogue signals. A preliminary research on that topic, done by [Luckenbill 2002], indicated the advantages and challenges in the build of such analogue circuits. The computational task of such systems is usually limited to perform a particular type of matrix-vector and vector-vector operations on real numbers from a unity interval  $[0...1]$  (see Chapter 7). Therefore, such systems have typically been realised in the current domain, where the arithmetic sums can be calculated by current additions in nodes and products can be evaluated using circuits employing the Gilbert cell (see Chapter 3).

The sum-product algorithm for belief propagation in factor graphs, discussed in Chapter 7, exhibits a high degree of parallelism on the network level, as well as in terms of the operations performed by nodes. On the network level, the message exchange and belief computation are independent processes. Also "inside" the nodes, messages are processed for each output link individually. Therefore, various ways of implementing such systems, with different levels of "hardware virtualisation", understood as the degree of time multiplexing of particular hardware computing blocks, with direct consequences in processing time, can be considered [Zaveri 2010]. In particular, continuous-time solutions, operating based on inherent circuit settling, require fully parallel realisations. Discrete-time analogue and synchronous digital solutions, also discussed in this chapter, have potential for processing with time multiplexing, however, analysis of such realisations goes beyond the scope of this work, which focuses mainly on the estimation of the processing efficiency, not significantly depending on the level of hardware virtualisation (this will be further discussed in this chapter). In the following sections, the realisations of arithmetic circuits for analogue continuous-time and discrete-time

processing, and equivalent digital and software implementations will be presented, verified in simulations, and analysed in terms of processing power, speed, efficiency, size and accuracy.

### 8.3 Analogue circuits for arithmetic operations

The arithmetic operations required in the sum-product algorithm for belief propagation, account for matrix-vector and vector-vector multiplications with normalisation of the intermediate results. Such operations can directly be implemented in dedicated analogue hardware using the proposed current-mode, continuous-time and discrete-time multipliers, discussed in Chapter 3.

#### 8.3.1 Continuous-time circuits

The realisations of the arithmetic circuits for 2-element continuous-time vector-vector and matrix-vector multiplications, are presented in Figure 8.1.

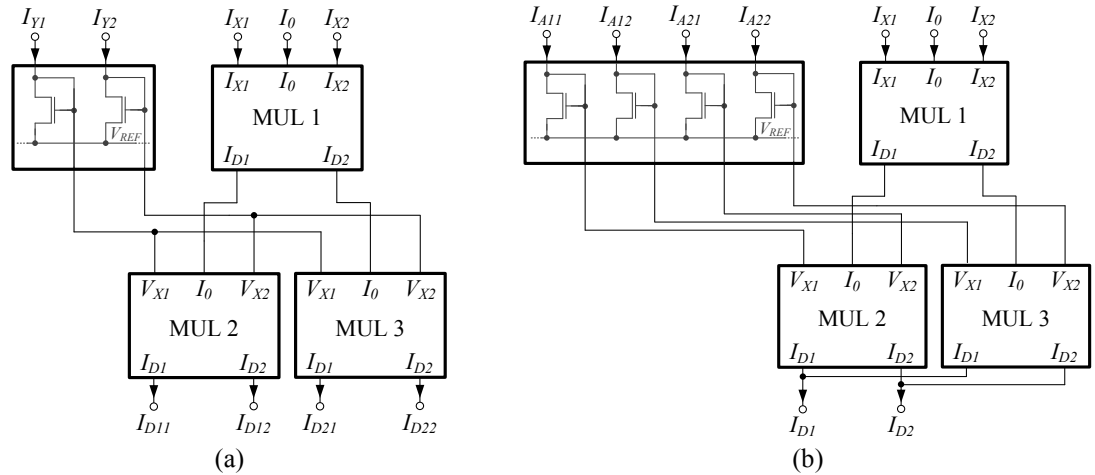


Figure 8.1. Block diagrams of the arithmetic circuits, realised using continuous-time current-mode multiplier circuit from Figure 3.2 in Chapter 3, dedicated for: a) vector-vector multiplication, and b) matrix-vector multiplication.

One of the main advantages of using continuous-time circuits is their low area, low power consumption and high processing speed, depending on the settling time of the circuit. However, such realisation requires fully parallel implementations, where each arithmetic operation has its individual hardware block. Since the information is encoded using currents, copying requires the use of current mirrors. In order to reduce the complexity of the circuits, arguments are distributed using their voltage representations rather than currents. Therefore, multipliers MUL 2 and MUL 3 in Figure 8.1 have voltage inputs  $V_{X1}$  and  $V_{X2}$ , and the corresponding log-linear  $I$ - $V$  converters are shared.

The structure presented in Figure 8.1a performs vector-vector multiplication, accounting for all element permutations, given by equation:

$$\begin{bmatrix} I_{D11} \\ I_{D12} \\ I_{D21} \\ I_{D22} \end{bmatrix} = \begin{bmatrix} I_{X1} \cdot I_{Y1} \\ I_{X1} \cdot I_{Y2} \\ I_{X2} \cdot I_{Y1} \\ I_{X2} \cdot I_{Y2} \end{bmatrix} \quad (8.1)$$

The structure from Figure 8.1b performs matrix-vector multiplication given by formula:

$$\begin{bmatrix} I_{D1} \\ I_{D2} \end{bmatrix} = \begin{bmatrix} I_{A11} & I_{A12} \\ I_{A21} & I_{A22} \end{bmatrix} \cdot \begin{bmatrix} I_{X1} \\ I_{X2} \end{bmatrix} \quad (8.2)$$

It should be noted that the circuits in Figure 8.1 perform inherent normalisation of the input arguments  $[I_{X1} \ I_{X2}]$   $[I_{Y1} \ I_{Y2}]$ . Moreover, the matrix-vector multiplier (Figure 8.1b) assumes normalised coefficient pairs  $A_{11}$ - $A_{21}$  and  $A_{12}$ - $A_{22}$ . This limitation, however, is advantageous in the operations on discrete probability densities, requiring intermediate result normalisations.

### 8.3.2 Discrete-time circuits

The realisations of 2-element vector-vector and matrix-vector multipliers using the discrete-time circuits are presented in Figure 8.2. They perform the same arithmetic operations as the continuous-time structures from Figure 8.1, given by equations (8.1) and (8.2). The timing diagram of the control signals used for circuit reconfigurations is presented in Figure 8.3.

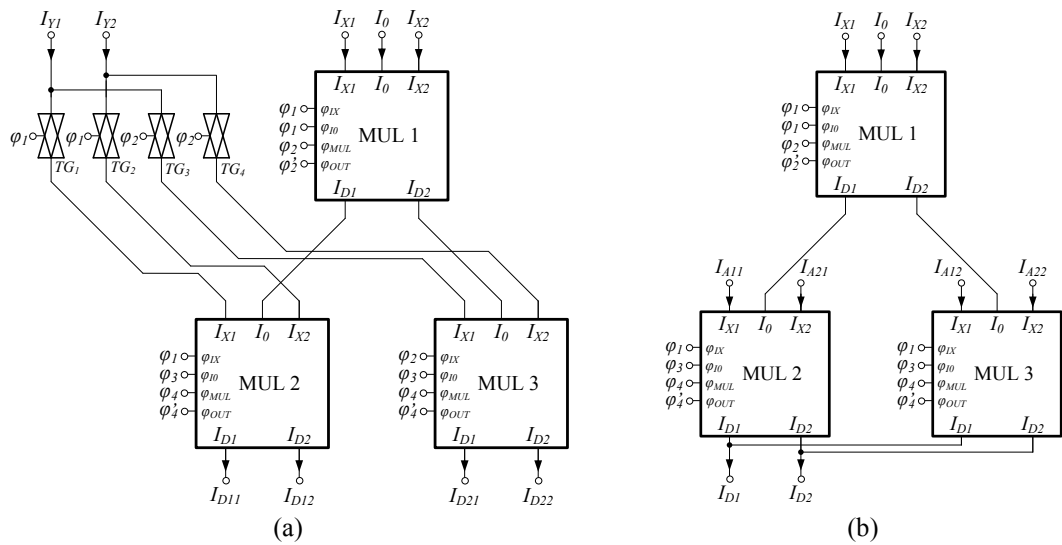


Figure 8.2. Block diagrams of the arithmetic circuits, realised using discrete-time current-mode multiplier circuit from Figure 3.11 in Chapter 3, dedicated for: a) vector-vector multiplication, and b) matrix-vector multiplication.

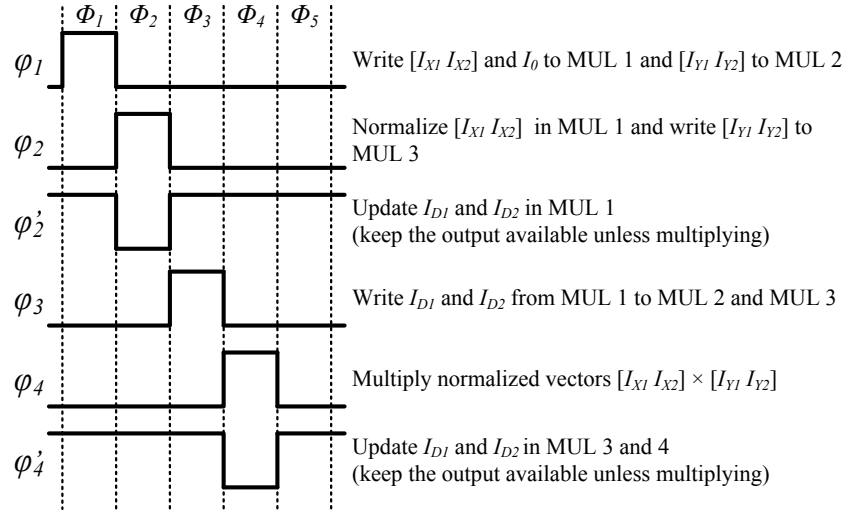


Figure 8.3. Timing diagram of the control signals  $\phi_1 - \phi_5$  of the arithmetic circuits from Figure 8.2 (the comments refer to the vector-vector multiplier in Figure 8.2a).

The proposed realisations of the discrete-time vector-vector and the matrix-vector multipliers operate in five phases  $\Phi_1 - \Phi_5$  using control signals  $\phi_1 - \phi_5$ . The matrix-vector multiplier is a simplified version of the vector-vector multiplier. The only difference is the set of transmission gates  $TG_1 - TG_4$  in the vector-vector multiplier, providing the second input argument  $[I_{Y1} I_{Y2}]$  to MUL 2 and MUL 3. This, however, is not required in the matrix-vector multiplier, where the second input is a set of fixed parameters provided individually for both multipliers. In the following, the operation of the vector-vector multiplier will be discussed.

In the implementation of the multipliers MUL 1, 2 and 3, the control signal  $\phi_{IN}$ , common for the memory cells built on transistors  $M_{D1}$ ,  $M_{D2}$  and  $M_{D3}$  (see Figure 3.11 in Chapter 3), has been replaced with two signals  $\phi_{IX}$  and  $\phi_{I0}$ , independently controlling the writing process to the memory cells built on transistor pair  $M_{D2} - M_{D3}$ , and  $M_{D1}$  respectively. The additional signals controlling the switches  $M_{S1} - M_{S5}$  (Figure 3.11) of the multipliers, correspond to signals  $\phi_{IX}$ ,  $\phi_{I0}$  and  $\phi_{MUL}$ , thus are not shown in Figures 8.2. and 8.3. In the first phase ( $\Phi_1$ ) the control signal  $\phi_1 = 1$  configures MUL 1 to save the input vector  $[I_{X1} I_{X2}]$  and the reference current  $I_{REF}$ , and also configures MUL 2 to save the second input vector  $[I_{Y1} I_{Y2}]$ , provided through the transmission gates  $TG_1$  and  $TG_2$ . In the belief propagation scheme, each intermediate result of the computation is a vector representing probability distribution and ought be normalised in order to avoid operations on very low numbers leading to underflow errors. Such normalisation, in the hardware realisation of the arithmetic circuit, is actually not performed at the output but inherently

at the input of the Gilbert multiplier. Since the vector  $[I_{Y1} I_{Y2}]$  is written to the symmetric inputs of the multipliers MUL 2 and MUL 3, it will naturally be normalised, however, the elements of vector  $[I_{X1} I_{X2}]$  feed the asymmetric inputs of these multipliers and will not be normalised. Therefore, the multiplier MUL 1 is necessary to perform such normalisation before the actual vector-vector multiplication. In the second phase ( $\Phi_2$ ), the control signals  $\varphi_2 = 1$  configure MUL 1 to perform normalisation of  $[I_{X1} I_{X2}]$  and write the results to the memory cells built on  $M_{D4}$  and  $M_{D5}$  in MUL 1. The results are available on the outputs  $I_{D1}$  and  $I_{D2}$  of MUL 1 in the next phases when  $\varphi_3 = 1$ . In the third phase ( $\Phi_3$ ), the control signal  $\varphi_3 = 1$  configures the asymmetric inputs of MUL 2 and MUL 3 to the reading mode, in order to copy the results computed by MUL 1 to MUL 2 and MUL 3. In the fourth phase ( $\Phi_4$ ), the control signals  $\varphi_4 = 1$  setting MUL 2 and MUL 3 into the multiplying mode, where the vector-vector multiplication is performed and the results are saved in the corresponding output memory cells of these multipliers. In the last phase ( $\Phi_5$ ), the computed result is available on the four outputs of MUL 2 and MUL 3. It can be observed that the first phase  $\Phi_1$  and the last  $\Phi_5$  are independent and could overlap in a pipelined computation scheme. The operation of the matrix-vector multiplier from Figure 8.2b is practically the same, only the matrix coefficients  $I_{A11} - I_{A22}$  are written to MUL 2 and MUL 3 in the first phase  $\Phi_1$ .

In the realisations of the arithmetic circuits presented in Figure 8.2, it has been assumed that the input arguments and the matrix parameters are available during the first and the second phases  $\Phi_1$  and  $\Phi_2$ . This assumption can be met when using the output memory cells with transistors  $M_{D4}$  and  $M_{D5}$  in the multipliers as a temporary data storage, providing the computed results to other circuits between the update cycles.

## 8.4 Computational errors

The *normalised computational error* of belief evaluation using the sum-product algorithm ( $NCE_{SPA}$ ) in hardware, is defined as the maximum difference between the elements of the normalised current vector  $I_{SIM}/\|I_{SIM}\|$ , representing belief of a particular node obtained from the circuit simulation, and the corresponding normalised vector  $BEL_{SPA}$ , obtained from the software implementation of the sum-product algorithm. The computational error  $NCE_{SPA}$  provides the same accuracy measure as the *normalised current error*, defined in equation (3.15) in Chapter 3, and can be seen as its extension

applicable to more complex systems. The computational error  $NCE_{SPA}$  is calculated for each node individually, according to the equation:

$$NCE_{SPA}[\%] = \max \left| \frac{I_{SIM}}{\|I_{SIM}\|} - BEL_{SPA} \right| \cdot 100\% \quad (8.3)$$

The disparities between the results obtained using the software realisation of the sum-product algorithm ( $BEL_{SPA}$ ) and beliefs computed using the global marginalisation algorithm  $BEL_{GMA}$  for exact Bayesian inference, can be seen as an additional source of computational errors. This error, however, does not result from hardware issues, but is inherent to the inference method employed, and will be calculated as:

$$NCE_{GMA}[\%] = \max |BEL_{GMA} - BEL_{SPA}| \cdot 100\% \quad (8.4)$$

In this thesis, the results obtained using the sum-product algorithm for belief propagation are considered as reference for the analysis of the accuracy of the hardware circuit realisations. The detailed analysis of the computational errors and the convergence issues in belief propagation scheme itself, goes beyond the scope of this research and will not be further discussed. The computational error  $NCE_{GMA}$  will be evaluated for reference and comparison with  $NCE_{SPA}$  of the hardware realisations.

## 8.5 Simulations

### 8.5.1 Simulation setup

In the simulations, two exemplar Bayesian networks presented in Figure 8.4 will be considered. The first network, shown in Figure 8.4a, is called *TN-5*, consists of five nodes and contains a (structural) loop, existing between nodes *A*, *B*, *C* and *D*. The second network is called *TN-7*, consists of seven nodes and has a tree structure. Both networks have fixed conditional probability tables (CPTs) and operate on two-state variables with values from the set  $\{T, F\}$  representing the probability of *true* and *false* respectively. It should be noted that the typical network size used in the decision and control systems in robotics is usually 10 to 20 nodes [Lazkano 2006], [Lebeltel 2004]. Larger networks, consisting of several thousand nodes and more, are considered in simulations and modelling complex systems for example in bioinformatics [Nikolova 2011].

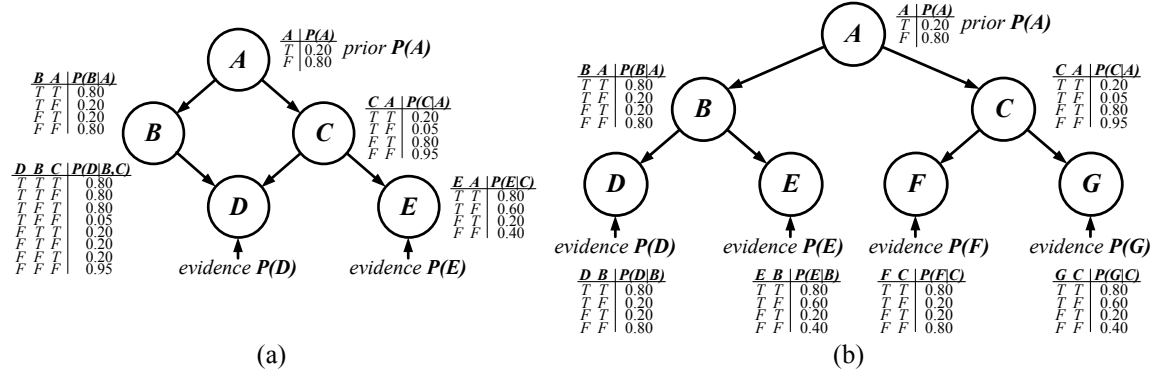


Figure 8.4. Bayesian networks used in the analogue circuits implementations consisting of a) five nodes with a loop (TN-5), and b) seven nodes singly connected forming a tree structure (TN-7).

In the circuit realisations of the networks from Figure 8.4, Bayesian nodes were implemented as pairs of three-way factor and three-way variable nodes. In such configuration, one pair of links connects the two nodes together and the remaining ones allow for connections with up to two parents and two children (see Figure 7.6 in Chapter 7). Each link passes over two-element vectors, one for the input and one for the output message. All the unconnected links, depending on the direction, were terminated either by a diode connected transistors or by pairs of current sources, generating currents  $[0.5 \ 0.5] \times I_{REF}$ . The diode-connected transistors were used to sink the output currents of the nodes, whereas the current sources were used to provide input messages, neutral in terms of the performed arithmetic operations. The prior probability for node  $A$ , in both networks, was inserted in its conditional probability table. The probabilities representing evidence for the bottom (leaf) nodes were inserted using one of the spare links for two potential descendants.

In the analogue circuit implementation, variable and factor nodes consist of three independent computational blocks performing the arithmetic operations for each output link individually. Additionally, in the variable node a separate block has been instantiated to compute belief. Circuit schematics and block diagrams of the continuous-time analogue implementation of the three-way factor and variable nodes are presented in Appendix A. The architecture of the system using discrete-time multipliers is the same.

In the designed system, the arithmetic blocks in the discrete-time implementation of three-way variable node require 12 phases to complete the computation of the output messages (including belief). The three-way factor node realisation requires 14 phases for the same task. Assuming that both nodes operate in parallel with cycle time  $T_C = 2 \mu s$  (the cycle time  $T_C$  was introduced in Chapter 3), the total processing time of a Bayesian node, necessary to update the output messages is  $28 \mu s$ . The additional two phases were

intentionally added to the control sequence of the variable node to assure equal processing times and synchronisation between nodes in the network. It is important to note that the message update time of a Bayesian node is twice longer and equal to  $56 \mu\text{s}$ , due to the propagation of the input messages through the variable and factor nodes.

The simulation results of the *TN-5* network were generated based on the total number of 110 input vectors and several different sets of CPTs summarised in Table 8.1. Here the input vector is defined as the set of the input information including the prior probability  $P(A)$  of node  $A$  and the evidence  $E(D)$  and  $E(E)$  for nodes  $D$  and  $E$  respectively. All the probabilities provided in the input vectors were normalised with respect to the reference current  $I_{REF}$ . In the calculations of the computational error, each node was taken as a separate case, therefore, the estimation of the statistical parameters of the corresponding  $NCE_{SPA}$ , for the *TN-5* network, is based on the total number of generated results, equal  $110 \times 5 = 550$  (110 input vectors, see Table 8.1, and 5 nodes, each generating one result). Similarly, the simulation results of the *TN-7* network were generated based on the total number of 90 input vectors and several different sets of CPTs summarised in Table 8.2. The computational error  $NCE_{SPA}$  of the *TN-7* network was based on the total number of results equal  $90 \times 7 = 630$  (the number of cases is 90, see Table 8.2).

Table 8.1. Input parameters and vectors used in the simulations of the *TN-5* network.

<b><i>Input set</i></b>	<b><i>CPT</i></b>	<b><i>P(A)</i></b>	<b><i>E(D), E(E)</i></b>	<b><i># cases</i></b>	<b><i>range</i></b>
#1	fixed (Figure 8.4a)	fixed	fixed	10	1% - 99 %
#2	fixed (Figure 8.4a)	fixed	random	10	5% - 95 %
#3	random (set #1)	random	random	30	5% - 95 %
#4	random (set #2)	random	random	30	5% - 95 %
#5	random (set #3)	random	random	30	5% - 95 %

Table 8.2. Input parameters and vectors used in the simulations of the *TN-7* network.

<b><i>Input set</i></b>	<b><i>CPT</i></b>	<b><i>P(A)</i></b>	<b><i>E(D), E(E)</i></b>	<b><i># cases</i></b>	<b><i>range</i></b>
#1	fixed (Figure 8.4b)	random	random	30	5% - 99 %
#2	random (set #1)	random	random	30	5% - 95 %
#3	random (set #2)	random	random	30	5% - 95 %

The number of cycles (iterations), required to attain convergence, can be determined only for the singly connected networks. It corresponds to the number of nodes on the longest path. If a network has (structural) loops, the messages start to circulate and the state of the network converges to the solution asymptotically. In such a case, an additional mechanisms controlling the convergence ought to be used, for example, terminating the computation when the result of a satisfactory precision has been

achieved. In the continuous-time analogue realisations, the convergence is attained inherently by the circuit, therefore, mechanisms for convergence monitoring are not necessary. In the discrete-time analogue realisations, the computed result slightly fluctuates around the target solution. Such behaviour has also been observed in the software implementations of the belief propagation, assuming reduced computational precision. In such systems, an arbitrary number of iterations for a particular network could be set to terminate the computations rather than measure the convergence rate, which may be difficult due to these oscillations. In the simulations, a fixed number of iterations has been assumed, equal 16, for the *TN-5* network, and equal 12 for the *TN-7* network. As a result, the total processing time of the discrete-time realisation is  $16 \times 28 \mu\text{s} = 448 \mu\text{s}$  (*TN-5* network) and  $12 \times 28 \mu\text{s} = 336 \mu\text{s}$  (*TN-7* network). For these numbers, the software implementation of the message passing algorithm converges to the result with error much below 1%.

### 8.5.1 Results

The histograms showing the distribution of the  $\text{NCE}_{\text{SPA}}$  error of the networks *TN-5* and *TN-7*, realised using the continuous-time and the discrete-time analogue circuits, are shown in Figures 8.5 and 8.6 respectively.

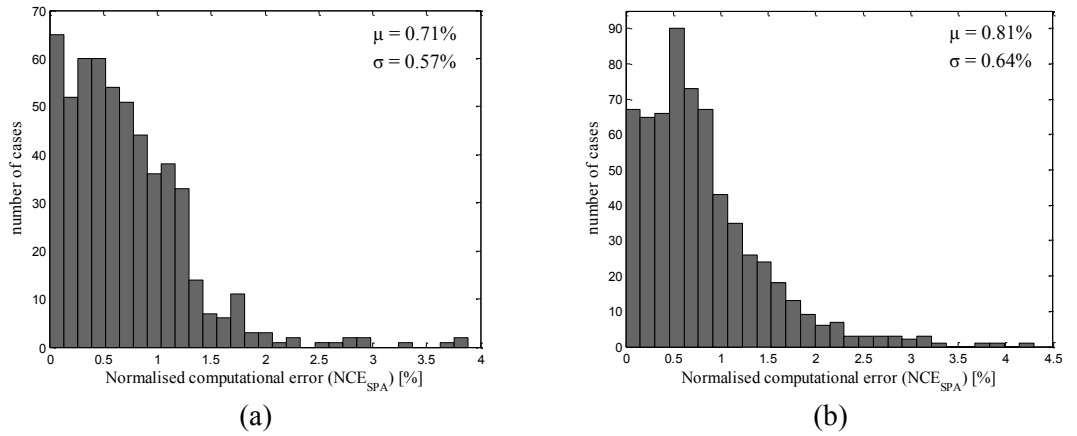


Figure 8.5. Histograms of the computational error  $\text{NCE}_{\text{SPA}}$  of the analogue continuous-time networks: a) *TN-5* ( $\mu = 0.71\%$ ,  $\sigma = 0.57\%$ ), and b) *TN-7* ( $\mu = 0.81\%$ ,  $\sigma = 0.64\%$ ).

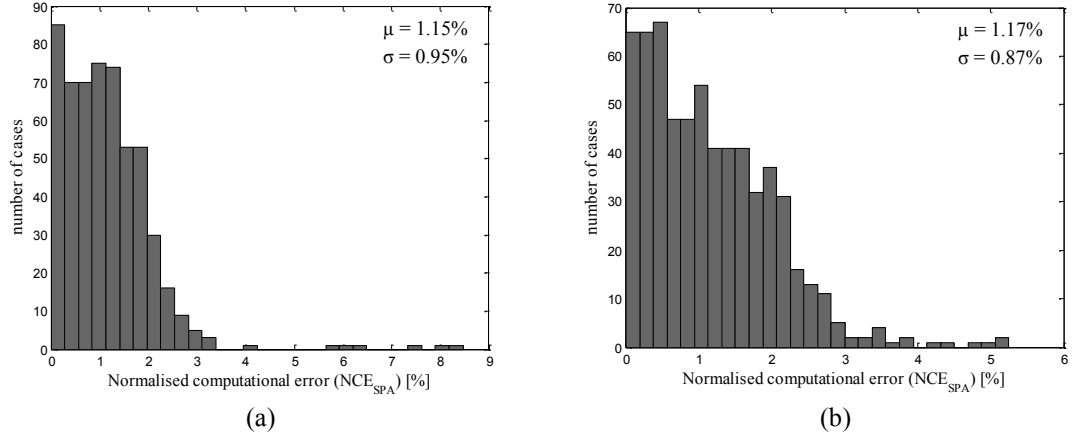


Figure 8.6. Histograms of the computational error  $NCE_{SPA}$  of the analogue discrete-time networks for  $T_C = 2 \mu s$ : a) *TN-5* ( $\mu = 1.15\%$ ,  $\sigma = 0.95\%$ ) and b) *TN-7* ( $\mu = 1.17\%$ ,  $\sigma = 0.87\%$ ).

It can be observed that the distribution of the  $NCE_{SPA}$  is similar for both circuit realisations. In the case of the discrete-time solution the generated error is slightly higher, mainly due to the additional effects related to charge injection, leakage and the channel length modulation of the MOS transistors used in the analogue memory cells. The precision of the obtained result degrades when very small probabilities occur in the CPTs or evidence. This results from the limited accuracy of the analogue multipliers when operating on small currents. The outlying bars in the histograms towards the higher computational error can be associated with the cases where the input parameter set contains probabilities below 5 %.

The simulations of the discrete-time circuit realisations were repeated assuming the cycle time  $T_C = 0.2 \mu s$ . The histograms of the  $NCE_{SPA}$ , generated for the same parameter sets from Tables 8.1 and 8.2, are shown in Figure 8.7. It can be observed that shortening the processing time degrades the precision of the obtained results, roughly by 30 %.

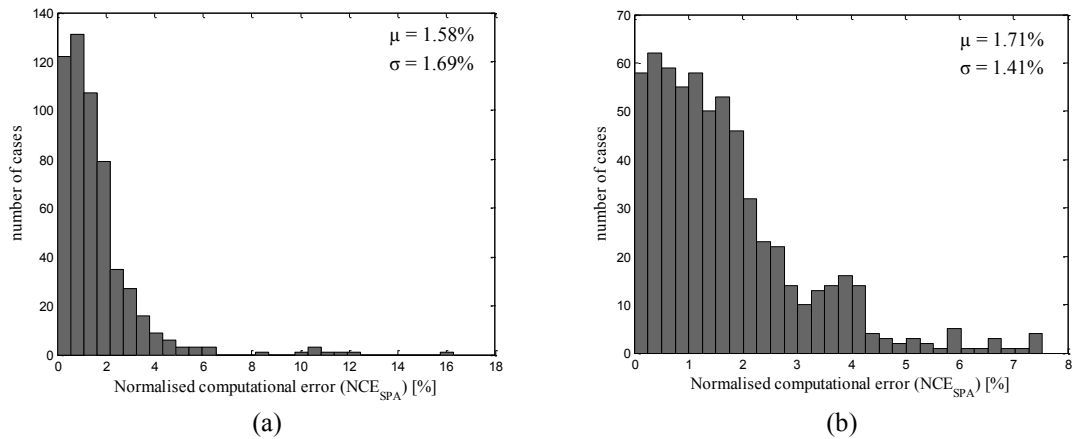


Figure 8.7. Histograms of the computational error  $NCE_{SPA}$  of the analogue discrete-time networks for  $T_C = 0.2 \mu s$ : a) *TN-5* ( $\mu = 1.58\%$ ,  $\sigma = 1.69\%$ ) and b) *TN-7* ( $\mu = 1.71\%$ ,  $\sigma = 1.41\%$ ).

The histogram of the computational error  $NCE_{GMA}$ , inherent to the sum-product algorithm, in comparison to the exact inference method based on global the marginalisation algorithm, is presented in Figure 8.8. In the analysis, the *TN-5* network containing a loop was simulated with the parameter sets provided in Table 8.1. The *TN-7* network is singly connected and the inference result returned by the sum-product algorithm is always exact (see Chapter 7). The obtained disparities between the results computed using belief propagation for approximate inference are typically below 1%, with a few samples above this level. It is important to note that the analogue circuit realisations of Bayesian networks, considered in this research, implement an approximate inference method. Therefore, besides the circuit design issues, affecting the precision of the obtained results, the error introduced by the method itself should also be considered when targeting more precise and better optimised circuit designs.

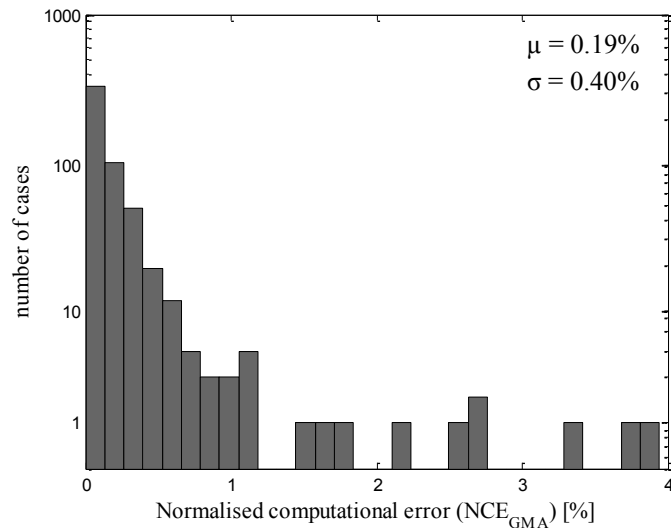


Figure 8.8. Histogram of the computational error  $NCE_{GMA}$  of the belief propagation in comparison to the global marginalisation method for exact inference ( $\mu = 0.19\%$ ,  $\sigma = 0.40\%$ ).

The process of convergence observed in nodes *A* and *D*, in the analogue realisations of the *TN-5* network, is shown in Figures 8.9 and 8.10. The results obtained from the reference software implementation, and the currents representing the corresponding beliefs, computed by the continuous-time and the discrete-time circuit realisations, were normalised and plotted on one graph in terms of a normalised time. Although the traces do not represent the timing relations between the traces correctly, they illustrate the process of network settling in three different realisations. The timescale of the continuous-time analogue traces were modified to assure fitting with the discrete-time and software results.

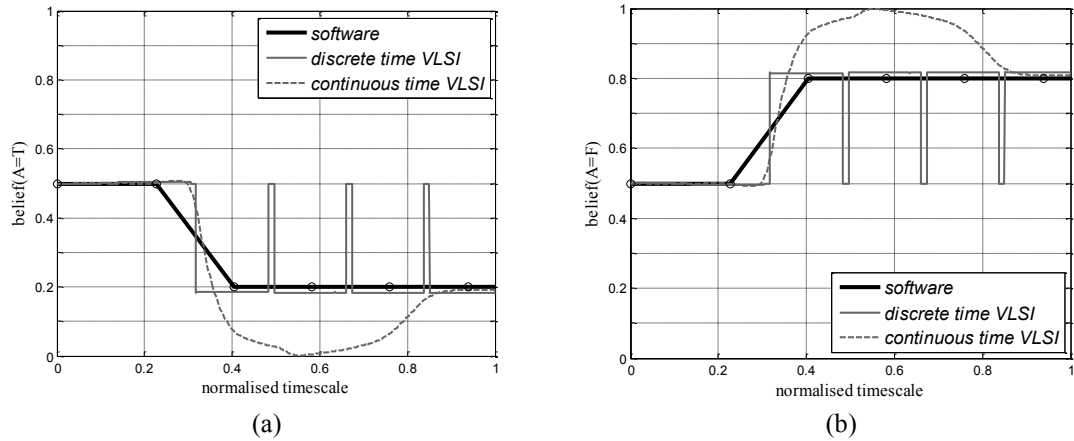


Figure 8.9. The convergence of the belief propagation mechanism in node  $A$  observed in the software and the analogue hardware implementations for the  $TN$ -5 network.

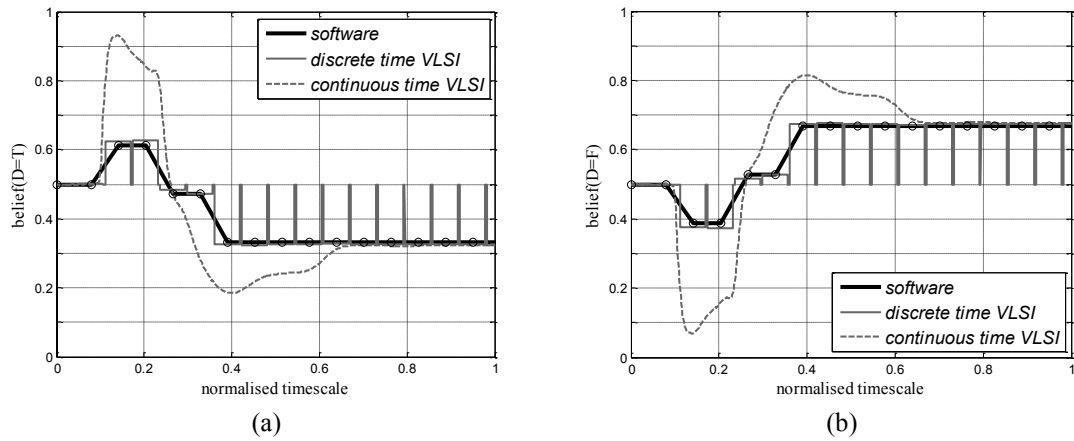


Figure 8.10. The convergence of the belief propagation mechanism in node  $D$  observed in the software and the analogue hardware implementations for the  $TN$ -5 network.

It can be observed that both analogue solutions correctly converge to the expected results, denoted by the software implementation of the sum-product algorithm. The additional bars on the traces of the discrete-time circuit, result from the normalisation of current during the message update phases. When the results computed by each node are being written to the output memory cells, the output currents are undefined for one cycle. In the simulations, these undefined currents are equal (due to the circuit symmetry) and after normalisation they are equal to 0.5.

When the network settles to the solution, the intermediate results oscillate both in hardware and software realisations (see Figure 8.10). This is caused by the faster processing of the local data of the nodes and slower information flow across the network. This effect is particularly visible in the continuous-time analogue implementation.

## 8.6 Networks in continuous-time analogue circuits

In Chapter 3, random parameter variability has been identified as the dominant factor degrading the accuracy of the continuous-time analogue multiplier circuit. It has been shown that the precision can be improved by transistor size scaling or by employing switched-current technique. This section deals with the continuous-time implementation of the sum-product algorithm and provides more detail characterisation of such arithmetic circuits in terms of the accuracy, power, speed and area. In particular, two techniques of reducing the impact of parameter mismatch on the computational accuracy, based on redundant design and area scaling, will be presented. Also, the convergence and scalability of larger networks will be discussed.

### 8.6.1 Accuracy versus redundancy

The idea of improving computational accuracy, based on redundant design, assumes that a more precise result can be obtained when averaging the results generated by the same circuit realisations but affected by random parameter variability. In such approach, rather than suppressing the parameter variability by averaging over a MOS device area, it is suggested to average the set of obtained results. In the experiments, the continuous-time circuit realisations of the networks *TN-5* and *TN-7*, with mismatch Monte Carlo MOS transistor models were used. Due to the long simulation time, the analysis of the *TN-5* network was limited to a set of 30 input vectors from case #3 in Table 8.1. The analysis of the *TN-7* network was performed for the case #2 from Table 8.2. The simulations were performed assuming fixed seed of the random number generator, starting always from the same point to assure the same set of mismatch parameters for each input vector. In other words, such setup allowed to simulate a system consisting of a set of 500 identical network copies (operating in parallel), where the result was calculated by averaging over 1, 2, 3,...,500 such circuit copies.

The simulation results showing the mean and the standard deviation of the normalised computational error ( $NCE_{SPA}$ ), in terms of the number of network copies used in the result averaging, are shown in Figures 8.11 and 8.12. It can be observed that both, the mean value and the standard deviation of  $NCE_{SPA}$ , decrease with the number of network copies used for averaging, and converge closely to the values  $\mu_{CT}$  and  $\sigma_{CT}$ , obtained from the simulations not accounting for parameter mismatch (the corresponding levels are denoted in the Figures by the horizontal dashed lines). In general, the obtained

results do not improve significantly for the number of network copies higher than 100. For comparison, the mean  $\mu_{DT}$  and the standard deviation  $\sigma_{DT}$  of the  $NCE_{SPA}$ , obtained from the simulations of the discrete-time realisations of these networks for the same input sets, are also shown in the Figures. The mean value of the computational error, generated by the discrete-time network realisations, corresponds to averaging over approximately 100 network copies. This, however, is only a rough estimation since both solutions operate based on different principles and are affected by different factors degrading the computational precision. Also, the standard deviation measure, representing the spread of the computational error, after some point, does not improve further with the number of network copies.

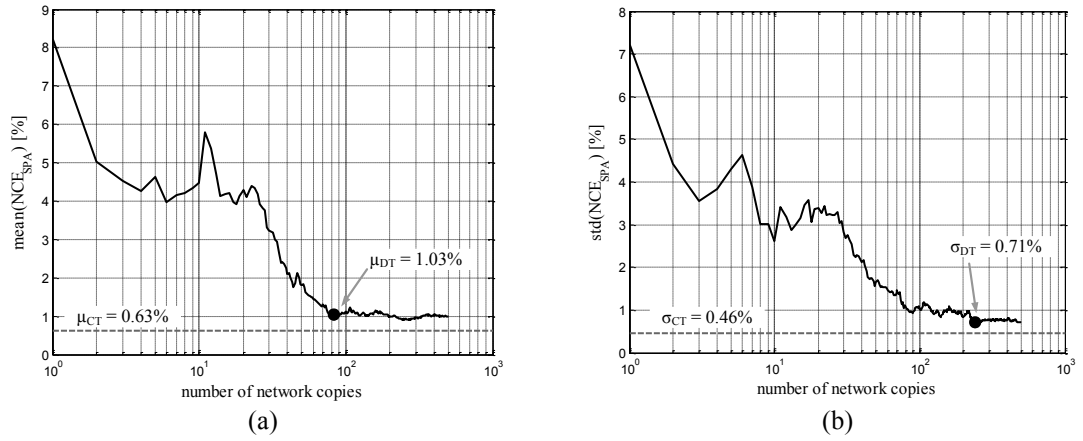


Figure 8.11. Computational error  $NCE_{SPA}$  of the TN-5 network versus the number of networks copies used for result averaging: a) mean, and b) standard deviation.

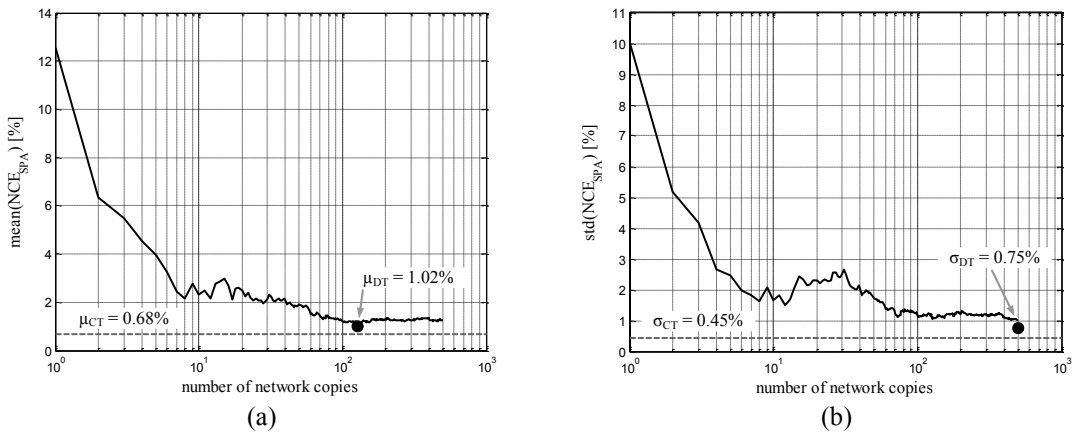


Figure 8.12. Computational error  $NCE_{SPA}$  of the TN-7 network versus the number of networks copies used for result averaging: a) mean, and b) standard deviation.

It is important to note that the benefit of such design redundancy, resulting in the improved computational precision, comes at a very high cost of power and area, and hence, decreases the processing efficiency of such realisations. In fact, power and area could be traded for time in the time-multiplexed realisation of such systems. Rather than building a set of fixed identical networks, a generic reconfigurable array of multipliers and current mirrors could be considered. In such approach, however, networks should be synthesised using different components to minimise the similarities between their particular realisations, in order to assure convergence of the averaged result. Therefore, the resources of such reconfigurable system, and its size, may be much higher than the maximum size of the network that could be efficiently implemented and solved.

### 8.6.2 Accuracy versus area

The simplest method of improving parameter matching in analogue circuits is transistor size scaling (see Chapter 2). In such approach, it is assumed that the random parameter variability of MOS devices averages out with the gate area increase. In the analyses presented in this section, widths and lengths of all the MOS transistors of the continuous-time circuit realisations of the test networks *TN-5* and *TN-7* were multiplied by an integer scaling factor  $\alpha$  with values from 1 to 25. In the simulations, the same input parameter sets, as in the previous section, were used, assuming only one Monte Carlo run for each input vector and the same starting point of the random number generator for parameter variability modelling. In other words, circuits with the same mismatch parameters were generated for different transistor sizes defined by the parameter  $\alpha$ , and simulated for the same set of input vectors.

The simulation results showing the mean and the standard deviation of the normalised computational error ( $NCE_{SPA}$ ), in terms of the area scaling factor  $\alpha^2$ , proportional to the circuit area, are shown in Figures 8.13 and 8.14. It can be observed that both, the mean value and the standard deviation of  $NCE_{SPA}$ , decrease with the circuit area, and converge closely to the respective values obtained from the simulations, not accounting for parameter mismatch (dashed black traces). For comparison, the results of the computational error, obtained from the method of averaging (from previous section), were also plotted (dashed grey traces). It can be assumed that the number of network copies used for averaging is equal to the area scaling factor  $\alpha^2$ , if no circuits other than the networks are required for such realisations. It can be observed that the efficiency of error suppression of both methods is similar in terms of the circuit area. In particular, the

approach based on the gate area scaling provides systematic improvement of the accuracy, asymptotically converging to the result generated by circuit not affected by mismatch. Due to the stochastic nature of the average-based method, the error reduction process is not deterministic and may not asymptotically converge to the expected result, but oscillate slightly above it. However, the average-based approach tends to converge quicker than the area scaling method. In particular, the accuracy of the continuous-time solution is comparable with the results of the discrete-time realisation (in terms of the mean error value) for area scaling factor approximately higher than 300. The same level of precision could be achieved using the average-based method for approximately 100 network copies (see previous section). This, however, is based on the analysis of a particular case of two test networks.

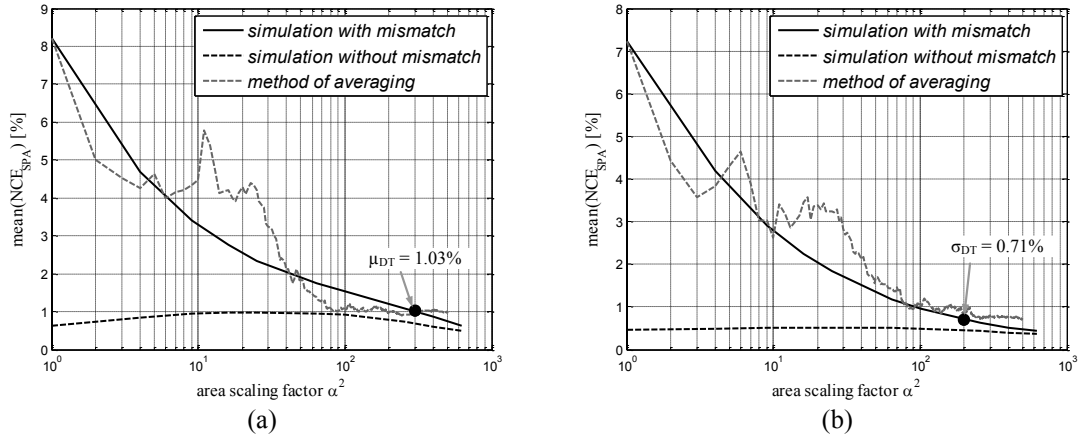


Figure 8.13. Computational error  $NCE_{SPE}$  of the TN-5 network versus area scaling factor  $\alpha^2$ : a) mean, and b) standard deviation.

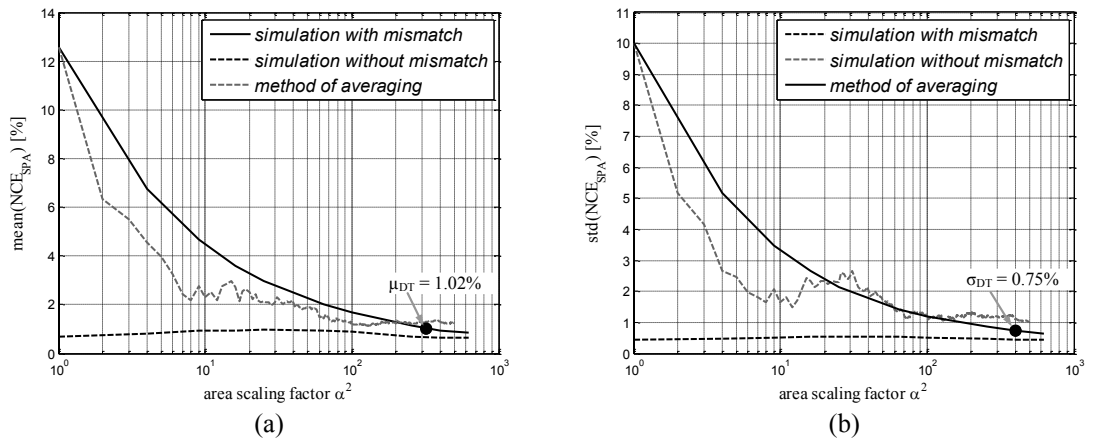


Figure 8.14. Computational error  $NCE_{SPE}$  of the TN-7 network versus area scaling factor  $\alpha^2$ : a) mean, and b) standard deviation.

More thorough characterisation of these methods should be considered in the future research, nevertheless, it should be noted that better accuracy is obtained at a very high cost of area. This may be prohibitive in many applications, precluding the use of such methods in practice. The processing speed and the implications of the area scaling on the convergence time in such networks will be discussed later in this chapter.

### 8.6.3 Convergence in large networks

The accuracy of continuous-time analogue circuits for Bayesian inference was verified in the simulations of several synthetic networks of a regular structure shown in Figure 8.15. Five networks of such structure were generated with number of nodes from 9 ( $3 \times 3$ ) to 121 ( $11 \times 11$ ) and random CPTs with entries within range 5% - 95%. The input test vector consisted of the prior probability of the middle top node  $A$  and the evidence for the middle bottom node  $B$  (Figure 8.15). These nodes were chosen to assure the message propagation across the entire network to verify the settling process and the correctness of the obtained results. In order to reduce the simulation time, DC operating point analysis was used to achieve the final state of the network, conceptually equivalent to the propagation after infinite time. The corresponding transient analysis takes much longer time and may not always be conclusive since the convergence time of a network depends on its parameters and input data, and cannot be easily estimated. The simulations of the settling time were performed for the  $TN-5$  and  $TN-7$  networks to estimate the processing efficiency.

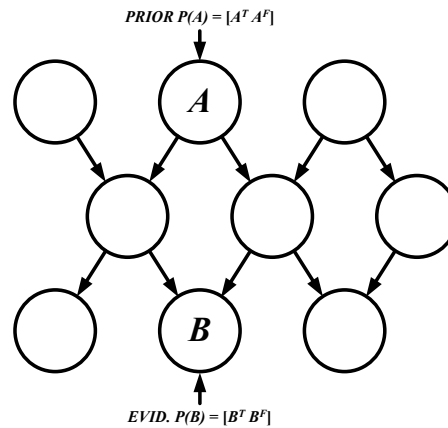


Figure 8.15. The structure of the Bayesian network  $TN-3 \times 3$  consisting of 9 nodes generated on a regular grid (the structure of other test networks is similar).

The simulation results of 5 synthetic networks, with structure shown in Figure 8.15 and consisting of 5 to 121 nodes implemented using 3-way factor and variable nodes, are

presented in Table 8.3. The reported number of transistors and the DC supply currents in the steady state account for the entire circuit with additional terminating blocks of unused links and banks of current sources for CPTs (the corresponding schematic and block diagrams are presented in Appendix A). The computational error was calculated based on 100 random input vectors randomly generated for each network with values from 5% to 95%.

Table 8.3. Simulation results of the synthetic test networks showing the complexity, power and accuracy scaling.

<i>Network</i>	<i>#MOS</i>	<i>I<sub>VDD</sub></i>	<i>mean(ANE<sub>MPA</sub>)</i>	<i>std(ANE<sub>MPA</sub>)</i>
<i>TN-3×3</i>	5,918	431.7 $\mu$ A	0.523 %	0.346 %
<i>TN-5×5</i>	16,406	1.166 mA	0.692 %	0.524 %
<i>TN-7×7</i>	32,126	2.256 mA	0.578 %	0.392 %
<i>TN-9×9</i>	53,078	3.701 mA	0.618 %	0.444 %
<i>TN-11×11</i>	79,262	5.500 mA	0.541 %	0.434 %

It can be observed that the mean value and the standard deviation, used as a measure of the error spread, remain at the same level despite the increased complexity of the computing hardware. This is mainly because belief propagation requires only local computation and the respective arithmetic circuits perform normalizations of the intermediate results at each processing step, which keeps the corresponding currents within the operating range of the circuit.

#### 8.6.4 Complexity and power scaling

Despite the limited computational accuracy, the continuous-time circuit implementations exhibit several promising advantages such as low power and short convergence time resulting in a high processing speed [Loeliger 2001], [Luckenbill 2002]. On the other hand, such hardware realisations of factor graphs become area and complexity prohibitive for larger networks. In this section, the analysis of the key scaling aspects of such hardware implementations, specific to the fully parallel, continuous-time realisation, is provided. The equations describing different complexity issues, derived for  $n$ -way factor and  $n$ -way variable nodes operating on  $k$ -state variables, are presented in Tables 8.4 and 8.5 respectively. It should be noted that the provided rules were derived based on the features of a particular implementation considered in this work. Other ways of realising the same functionality may also be possible. The details concerning the implementations of factor and variable nodes and the derivations of the equations for complexity and power scaling are provided in Appendices A and B.

Table 8.4. Scaling rules of  $n$ -way factor node operating on  $k$ -state messages (see Appendix B).

<b>Feature</b>	<b>Exact equation</b>	<b>Complexity</b>
# basic multipliers of type A	$n \cdot k \cdot \frac{k^{n-2} - 1}{k - 1} + k^{n-1}$	$O(n \cdot k^n)$
# basic multipliers of type B	$(n-1) \cdot (k^n + 1) + 1$	$O(n \cdot k^n)$
# two-argument real number multiplications	$n \cdot (n-2)k^{n-1} + k^n$	$O(n^2 \cdot k^n)$
# two-argument real number additions	$n \cdot k \cdot (k^{n-1} - 1)$	$O(n \cdot k^n)$
# $k$ element vector normalisations	$n$	$O(n)$
supply current	$(2 \cdot n + 2 \cdot (n-1) \cdot (n+k-1)) \cdot I_{REF}$	$O(n^2 + n \cdot k)$

Table 8.5. Scaling rules of  $n$ -way variable node operating on  $k$ -state messages (see Appendix B).

<b>Feature</b>	<b>Exact equation</b>	<b>Complexity</b>
# basic multipliers of type A	$n \cdot (n-2) \cdot k + (n-1) \cdot k$	$O(n^2 \cdot k)$
# basic multipliers of type B	$n + 1$	$O(n)$
# two-argument real number multiplications	$n \cdot (n-2) \cdot k + (n-1) \cdot k$	$O(n^2 \cdot k)$
# $k$ element vector normalisations	$n + 1$	$O(n)$
supply current for $n = 2$	$n = 2 : 8 \cdot I_{REF}$	---
supply current for $n = 3$	$n = 3 : (12 + (4 + 1/k + 1/k^2)) \cdot I_{REF}$	---
supply current for $n \geq 4$ (worst case approximation, error < 12 %)	$n \geq 4 : (n+1) \cdot (4 + 1/k + 2/(k^2 \cdot (1-1/k))) \cdot I_{REF}$	$O(n)$

The complexity of the proposed continuous-time hardware realisation of factor graph for Bayesian inference does not only depend on the number of nodes  $N$  in the network but also on the structure of each node. Based on the rules provided in Tables 8.4 and 8.5, some estimates concerning the power, area, and the computational complexity of a generic Bayesian node can be derived. In particular, the hardware complexity and the area requirements grow proportionally to  $O(n^2 k^n)$  with the number of links  $n$ . Interestingly, the power consumption of the analogue hardware grows much slower and proportionally to  $O(n^2 + nk)$ , which is very promising in terms of low power designs. This stems from the effect of current partitioning in the pyramid-structure multipliers (see Appendix B for reference) for vector-vector operations. It is achieved, however, at the expense of the computational precision, degrading for larger circuits since the information is gradually represented by smaller currents decreasing in geometric progress with the number of input vectors. This could be fixed by breaking larger pyramids into smaller sub-multipliers with intermediate current normalisation.

Figures showing complexity of the two-way and three-way variable and factor nodes, operating on two-state variables and derived using the exact equations from Tables 8.4

and 8.5, are presented in Table 8.6. It should be noted that the equations in Table 8.5 consider variable node implementation assuming optimised pyramid-structure multipliers, without cells generating products of unused element permutations (see Appendix A and B). In the continuous-time circuit realisations full pyramid-structure multipliers were used. As a result, the statistics reported by the circuit simulator (Table 8.3) show higher number of MOS transistors and power supply currents (including also additional bias  $I_{REF}$ , terminating blocks for links, beliefs and CPT currents) than those predicted using scaling rules from Table 8.5. This disparity, however, is negligible for very low number of links  $n$ .

Table 8.6. Complexity figures for two-way and three-way factor and variable nodes operating on two-state variables.

<i>Feature</i>	<i>Factor node</i>		<i>Variable node</i>	
	<i>2-way</i>	<i>3-way</i>	<i>2-way</i>	<i>3-way<sup>1)</sup></i>
# basic multipliers of type A	2	10	2	10/12
# basic multipliers of type B	6	19	3	4
# MOS transistors	118	416	72	206/236
# 2-arg MUL operations	8	36	2	10
# 2-arg ADD operations	4	18	---	---
# 2-element vector normalisations	2	3	3	4
Supply current	$10 \cdot I_{REF}$	$22 \cdot I_{REF}$	$8 \cdot I_{REF}$	$16.75/18 I_{REF}$

<sup>1)</sup> reduced pyramid-structure/full pyramid-structure multipliers (used in the circuit realisations)

Continuous-time hardware realisations of the test networks used in this work and four example networks (*Mendel Genetics*, *Car Diagnostic*, *Alarm* and *Hail Finder* [Norsys 2014]), were analyzed in terms of the complexity and the computational requirements. The estimated figures of implementation requirements are presented in Tables 8.7 and 8.8. In the area prediction, it was assumed that implementation of a single continuous-time multiplier (Figure 3.2 in Chapter 3) in a standard 90 nm CMOS technology, occupies  $6.8 \mu\text{m} \times 10 \mu\text{m}$  area, which is comparable with three D flip-flop cells in the same technology.

Table 8.7. Implementation requirements of the synthetic Bayesian networks used in this work.

<i>Network</i>		<i>Analogue hardware</i>			<i>Operations/iteration</i>		
<i>Name</i>	<i>Complexity<sup>1)</sup></i>	<i>#MOS</i>	<i>I<sub>VDD</sub></i>	<i>Area [<math>\mu\text{m}^2</math>]</i>	<i>#mul</i>	<i>#add</i>	<i>#norm</i>
<i>TN-1</i>	5/2/2	1,516	$119.5 \cdot I_0$	$84 \times 84$	94	34	28
<i>TN-7</i>	7/2/2	1,732	$152.3 \cdot I_0$	$90 \times 90$	94	28	38
<i>TN-3×3</i>	9/2/2	3,438	$245 \cdot I_0$	$127 \times 127$	234	92	53
<i>TN-5×5</i>	25/2/2	11,662	$782 \cdot I_0$	$234 \times 234$	826	324	157
<i>TN-7×7</i>	49/2/2	24,862	$1,629 \cdot I_0$	$342 \times 342$	1,786	700	317
<i>TN-9×9</i>	81/2/2	43,038	$2,786 \cdot I_0$	$450 \times 450$	3,114	1,220	533
<i>TN-11×11</i>	121/2/2	66,190	$4,253 \cdot I_0$	$560 \times 560$	4,810	1,884	805

<sup>1)</sup> Complexity: no. of nodes, max. no. of parents and max. no. of children

Table 8.8. Implementation requirements of four example Bayesian networks with assuming realisations with two-state variables only [Norsys 2014].

<i>Network</i>		<i>Analogue hardware</i>			<i>Operations/iteration</i>		
<b>Name</b>	<b>Complexity<sup>1</sup></b>	<b>#MOS</b>	<b><math>I_{VDD}</math></b>	<b>Area [<math>\mu\text{m}^2</math>]</b>	<b>#mul</b>	<b>#add</b>	<b>#norm</b>
<i>Mendel</i>	6/2/2	1,706	$135.5 \cdot I_0$	90×90	104	38	33
<i>Car Diag.</i>	18/5/3	14,244	$513 \cdot I_0$	260×260	1,656	572	104
<i>Alarm</i>	37/3/5	19,676	$1,111 \cdot I_0$	304×304	1,598	594	226
<i>HailFinder</i>	56/4/16	34,344	$1,539 \cdot I_0$	401×401	2,804	906	330

<sup>1)</sup> Complexity: no. of nodes, max. no. of parents and max. no. of children

## 8.7 Performance of analogue systems

### 8.7.1 Computational efficiency

A comparison of the performance of the computational systems realised in different architectures and operating according to different principles, is usually not straightforward. A baseline performance estimation of such systems can be done assuming that a particular task, in general, consists of some elementary arithmetic operations (e.g. additions, multiplications, divisions etc.), and is executed in a certain time consuming certain power. The measure of the computational efficiency CE, used in the comparison of the systems considered in this research, is defined as a ratio of the number of arithmetic operations performed per second per consumed power according to the generic formula:

$$\text{CE} = \frac{\#operations / time}{power} = \frac{processing\ speed\ [OPS]}{power\ [W]} = \frac{\#operations\ [OP]}{energy\ [J]} \quad (8.5)$$

In the equation (8.5), the unit of the processing speed is OPS (i.e. operations-per-second), and the computational efficiency CE is measured in OPS/W. The proposed definition of the computational efficiency requires only basic parameters of a system such as the number of the executed operations and the consumed energy. In particular, the level of the system parallelisation, which is not of the main interest here, will be marginalised in the calculation of the parameter CE. It should be noted that such comparison is valid only if the considered systems generate results of a similar precision. For example, a comparison of the analogue continuous-time and discrete-time systems should account for the effects of fabrication mismatch. Therefore the computational efficiency of the continuous-time solutions should be estimated for the systems employing the average-based or scaling-based mismatch optimisation techniques, discussed in previous sections. The precision of the continuous-time analogue circuit increases almost to the level where such comparison becomes justifiable, assuming either the result averaging over 100

network copies, or the area sizing by a factor of 300. Similarly, in order to compare the analogue and digital realisations, the later ones should be implemented assuming computation with reduced precision, "matching" the accuracy of the analogue solutions. In the following sections, the computational efficiency CE will be calculated based on the equation (8.5), with respect to the number of two-argument multiplications required for one message update (one cycle) of a three-way factor and three-way variable node operating on two-state variables. It is assumed that one such cycle requires the execution of the equations (7.8) - (7.10), in the case of the factor node, and equations (7.11) - (7.14), in the case of the variable node.

### 8.7.2 Performance of the continuous-time realisation

The computational efficiency of the continuous-time and discrete-time analogue realisations of the *TN-5* and *TN-7* test networks was verified in the simulations assuming five input vectors including different sets of network parameters. Due to the circuit symmetry, the continuous-time realisations consume a constant power in the steady state, and also when settling to the solution. The measurement of the settling time in the continuous-time circuits is not straightforward since it depends on the network parameters and can vary in a wide range for the same network structure. The settling time was simulated using transient analysis assuming zero initial condition (i.e. all circuit nodes with zero initial voltage). This allows to measure the worst case convergence time where all the MOS gate capacitances has to be charged. The DC operating point analysis was used to obtain the steady state solution of the circuit. The currents obtained from both (transient and DC) analyses were normalised to represent beliefs. The results of the transient analyses were then sampled with a fixed time interval of 20 ns to search for the moment where the difference between the obtained result and the asymptotic solution from the DC analysis is less than 1%. This analysis was repeated for both networks for five different input vectors (defining the network parameters) assuming area scaling factor  $\alpha^2$  equal 1 and 300, to estimate the convergence time of the networks employing the average-based and scaling-based mismatch optimisation techniques. The obtained results are summarised in Tables 8.9, 8.10 and 8.11.

In the considered hardware implementations, each Bayesian node consists of one variable and one factor node with three ways each, where one message update sequence requires 46 two-argument multiplications (36 for the factor node and 10 for the variable node, see Table 8.6 for reference). Due to the continuous-time operation of the circuit, it

is not possible to determine the number of operations performed before the network settles to the solution. Therefore, in the estimations of the processing speed and efficiency, the number of operations performed will be calculated based on the discrete-time realisation whereas the power and convergence time will be taken from the continuous-time circuit simulations. In the discrete-time realisation (and in software), it was assumed that the *TN-5* network converges in 16 message update cycles, and the *TN-7* network converges in 12 message update cycles. The total number of operations performed by each network equals  $46 \times 5 \times 16 = 3680$  (for the *TN-5* network), and  $46 \times 7 \times 12 = 3864$  (for the *TN-7* network). Assuming the estimated average supply current  $I_{DD}$ , supply voltage  $V_{DD} = 1.2$  V and the convergence time  $T_{CONV}$  from Table 8.9, the processing speed of the *TN-5* network is  $3680 / 0.56 = 6571$  MOPS (mega operations per second) and the processing speed of the *TN-7* network is  $3864 / 0.62 = 6232$  MOPS. The computational efficiency of the *TN-5* network is  $CE = 6571 \text{ MOPS} / 277 \text{ } \mu\text{W} = 23.7 \text{ TOPS/W}$ , and the computational efficiency of the *TN-7* network is  $CE = 6232 \text{ MOPS} / 393 \text{ } \mu\text{W} = 15.9 \text{ TOPS/W}$ . It is important to note that the computational efficiency is a measure of the system performance and does not necessarily provide an information concerning its scalability in terms of the processing speed. Therefore, it should be carefully used in the estimations of the speed. For example, a processing efficiency of 1 TOPS/W does not necessarily mean that a system will perform  $10^{15}$  operations per second at the power of 1kW. In particular, in the continuous-time systems, the processing speed and the power consumption depend on the inherent RC circuit parameters, and hence, cannot easily be controlled.

Ideally, the continuous-time analogue computing systems exhibit a very high processing efficiency of about 20 TOPS/W. Nevertheless, the effects of parameter mismatch significantly affect the accuracy of such circuits. Mismatch optimisation employing the average-based and the scaling-based techniques, discussed in this chapter, improve the precision but degrades the processing speed and efficiency. In particular, the average-based method, assuming the use of 100 network copies, increases the power consumption and reduces the processing efficiency by factor of 100 respectively, but do not affect the speed. The scaling-based technique, assuming gate size scaling by factor of 300, increases the convergence time and decreases the processing speed and efficiency roughly by factor of 300, but do not increase the power consumption. This results mainly from the fact that the gate capacitances and the settling time of a continuous-time circuit scales with the gate area.

Table 8.9. The computational efficiency of the continuous-time realisations of *TN-5* and *TN-7* networks (no mismatch assumed).

case	<i>TN-5</i>					<i>TN-7</i>				
	$I_{DD}$ [ $\mu$ A]	Power [ $\mu$ W]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]	$I_{DD}$ [ $\mu$ A]	Power [ $\mu$ W]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]
#1	231.11	277.33	0.76	4842	17.5	327.64	393.17	0.68	5682	14.5
#2	230.87	277.04	0.44	8364	30.2	328.25	393.90	0.50	7728	19.6
#3	231.37	277.64	0.40	9200	33.1	327.63	393.16	0.60	6440	16.4
#4	231.06	277.27	0.52	7077	25.5	328.08	393.70	0.52	7431	18.9
#5	231.02	277.22	0.66	5576	20.1	327.49	392.99	0.78	4954	12.6
<b>average</b>	<b>231</b>	<b>277</b>	<b>0.56</b>	<b>6571</b>	<b>23.7</b>	<b>328</b>	<b>393</b>	<b>0.62</b>	<b>6232</b>	<b>15.9</b>

Table 8.10. The computational efficiency of the continuous-time realisations of *TN-5* and *TN-7* networks (average-based method using 100 network copies).

case	<i>TN-5</i>					<i>TN-7</i>				
	$I_{DD}$ [mA]	Power [mW]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]	$I_{DD}$ [mA]	Power [mW]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]
#1	23.111	27.733	0.76	4842	0.175	32.764	39.317	0.68	5682	0.145
#2	23.087	27.704	0.44	8364	0.302	32.825	39.390	0.50	7728	0.196
#3	23.137	27.764	0.40	9200	0.331	32.763	39.316	0.60	6440	0.164
#4	23.106	27.727	0.52	7077	0.255	32.808	39.370	0.52	7431	0.189
#5	23.102	27.722	0.66	5576	0.201	32.749	39.299	0.78	4954	0.126
<b>average</b>	<b>23.1</b>	<b>27.7</b>	<b>0.56</b>	<b>6571</b>	<b>0.237</b>	<b>32.8</b>	<b>39.3</b>	<b>0.62</b>	<b>6232</b>	<b>0.159</b>

Table 8.11. The computational efficiency of the continuous-time realisations of *TN-5* and *TN-7* networks (scaling-based method for  $\alpha^2 = 300$ ).

case	<i>TN-5</i>					<i>TN-7</i>				
	$I_{DD}$ [ $\mu$ A]	Power [ $\mu$ W]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]	$I_{DD}$ [ $\mu$ A]	Power [ $\mu$ W]	$T_{CONV}$ [ $\mu$ s]	Speed [MOPS]	CE [TOPS/W]
#1	219.76	263.71	187	19.68	0.074	309.78	393.17	121	31.93	0.086
#2	219.92	263.90	211	17.44	0.066	309.68	393.90	154	25.09	0.068
#3	219.69	263.63	131	28.09	0.107	309.67	393.16	114	33.89	0.091
#4	219.70	263.64	131	28.09	0.107	309.64	393.70	125	30.91	0.083
#5	219.74	263.69	147	25.03	0.095	309.80	392.99	152	25.42	0.068
<b>average</b>	<b>219</b>	<b>264</b>	<b>161</b>	<b>22.85</b>	<b>0.087</b>	<b>328</b>	<b>393</b>	<b>133</b>	<b>29.05</b>	<b>0.074</b>

### 8.7.3 Performance of the discrete-time realisation

The architectures and the corresponding block diagrams of the discrete-time and the continuous-time network realisations, considered in this research, are identical, and can be referenced to Appendices A and B. The only difference is the operation of the basic multiplier cell. The discrete-time circuit realisations were implemented, using only three-way factor and three-way variable nodes. It was assumed that the *TN-5* network converges after 16 message update cycles, and the *TN-7* network after 12 message update cycles. The number of two-argument multiplications performed for each message update was equal to  $46 \times 5 \times 16 = 3680$  (for the *TN-5* network) and  $46 \times 7 \times 12 = 3864$  (for the *TN-7* network). The power consumption was calculated based on the average supply

current  $I_{DD}$ , obtained from the simulations assuming the cycle time  $T_C = 2 \mu\text{s}$  and supply voltage  $V_{DD} = 1.15 \text{ V}$ . Since the implemented three-way variable and three-way factor nodes require 14 cycles to compute the output messages, the total processing time of the *TN-5* network is  $14 \times 2 \mu\text{s} \times 16 \approx 450 \mu\text{s}$  and the total processing time of the *TN-7* network is  $14 \times 2 \mu\text{s} \times 12 \approx 340 \mu\text{s}$ . Therefore, the *TN-5* network performs  $3680/450 = 8.18 \text{ MOPS}$ , and the *TN-7* network performs  $3864 / 340 = 11.36 \text{ MOPS}$ . The computational efficiency of the *TN-5* network is  $\text{CE} = 8.18 \text{ MOPS} / 50.77 \mu\text{W} = 0.161 \text{ TOPS/W}$ , and of the *TN-7* network is  $\text{CE} = 11.36 \text{ MOPS} / 74.46 \mu\text{W} = 0.153 \text{ TOPS/W}$ . The obtained results and the corresponding are summarised in Table 8.12. Figures for the same parameters but assuming a shorter cycle time  $T_C = 0.2 \mu\text{s}$  are presented in Table 8.13.

Table 8.12. The computational efficiency of the discrete-time realisations of *TN-5* and *TN-7* networks for  $T_C = 2 \mu\text{s}$ .

	<i>TN-5</i>				<i>TN-7</i>			
<i>case</i>	$I_{DD} [\mu\text{A}]$	$Power [\mu\text{W}]$	$Speed [\text{MOPS}]$	$\text{CE} [\text{TOPS/W}]$	$I_{DD} [\mu\text{A}]$	$Power [\mu\text{W}]$	$Speed [\text{MOPS}]$	$\text{CE} [\text{TOPS/W}]$
#1	44.07	50.68	8.18	0.161	62.10	74.52	11.36	0.153
#2	44.67	51.37	8.18	0.159	61.50	73.80	11.36	0.154
#3	43.81	50.38	8.18	0.162	61.95	74.34	11.36	0.153
#4	44.01	50.61	8.18	0.162	61.66	73.99	11.36	0.154
#5	44.20	50.83	8.18	0.161	63.05	75.66	11.36	0.150
<b>average</b>	<b>44.20</b>	<b>50.77</b>	<b>8.18</b>	<b>0.161</b>	<b>62.05</b>	<b>74.46</b>	<b>11.36</b>	<b>0.153</b>

Table 8.13. The computational efficiency of the discrete-time realisations of *TN-5* and *TN-7* networks for  $T_C = 0.2 \mu\text{s}$ .

	<i>TN-5</i>				<i>TN-7</i>			
<i>case</i>	$I_{DD} [\mu\text{A}]$	$Power [\mu\text{W}]$	$Speed [\text{MOPS}]$	$\text{CE} [\text{TOPS/W}]$	$I_{DD} [\mu\text{A}]$	$Power [\mu\text{W}]$	$Speed [\text{MOPS}]$	$\text{CE} [\text{TOPS/W}]$
#1	48.78	50.68	81.8	1.46	68.94	82.73	113.6	1.37
#2	49.77	51.37	81.8	1.43	68.34	82.00	113.6	1.39
#3	48.54	50.38	81.8	1.47	68.95	82.74	113.6	1.37
#4	48.86	50.61	81.8	1.46	68.32	81.94	113.6	1.39
#5	48.87	50.83	81.8	1.46	69.03	82.84	113.6	1.37
<b>average</b>	<b>48.96</b>	<b>50.61</b>	<b>81.8</b>	<b>1.45</b>	<b>68.72</b>	<b>82.46</b>	<b>113.6</b>	<b>1.38</b>

It can be observed that shortening the cycle time by the factor of 10 decreases the processing time and increases the processing speed 10 times. However, it causes only a small increase of the power consumption, which improves the computational efficiency. It should be noted that this improvement of the circuit performance comes at the cost of the reduced computational precision, therefore, it should be compared with an equivalent continuous-time implementation generating results of a similar precision. This could be

obtained by decreasing the number of the network copies used in the average-based approach or reducing the value of the area factor  $\alpha$  in the scaling-based approach. For example, for the *TN-5* network in the discrete-time realisation (assuming cycle time  $T_C = 0.2 \mu\text{s}$ ), the parameters of the computational error are:  $\mu = 1.58\%$  and  $\sigma = 1.69\%$  (Figure 8.7a). Such level of the computational error could be obtained from the continuous-time realisation assuming the result averaging over 50 network copies (Figure 8.11a) or area scaling by factor  $\alpha^2 = 200$  (Figure 8.13a). Assuming the averaging over 50 network copies, the power consumption will reduce two times, giving only twice higher efficiency equal 0.50 TOPS/W (see Table 8.10 for reference) which is almost three times lower than the efficiency of the discrete-time system equal 1.45 TOPS/W (see Table 8.13) for the same computational accuracy.

## 8.8 Digital implementation

This section presents several realisations of the sum-product algorithm for Bayesian inference in the dedicated digital hardware and in software for PC. In particular, digital circuits for fixed-point arithmetic are considered in the realisations of the computational systems with reduced precision, trading accuracy for speed and processing efficiency. The figures of performance of such digital solutions are compared with the analogue realisations designed in the same technology, and with the software implementations in Matlab and C++.

### 8.8.1 Fixed point arithmetic

The sum-product algorithm for belief propagation operates on arguments from the unity interval  $[0...1]$ , therefore, the required arithmetic operations could be performed using unsigned  $N$ -bit integer numbers, representing probabilities within range 0 to  $(2^N - 1)/2^N$ . This, however, imposes specific design and requires particular solutions on the hardware level, concerning issues such as rounding errors, under-flow and over-flow cases, and the realisations of the arithmetic operations such as addition, multiplication and division.

In the realisations considered in this research, the multiplication of two  $N$ -bit arguments returns a  $2N$  bit result, preceded by the decimal point. The truncation to  $N$  bit number is done simply by taking only the first  $N$  most significant bits of the obtained  $2N$  bit result. This introduces some rounding error, but simplifies the hardware design. A

critical operation is vector normalisation, which requires addition of all the elements of a vector and division of each element by the obtained sum. This is usually time, power and area consuming, even in the fixed point arithmetic circuits. The normalisation is necessary to avoid over-flow and under-flow errors. Since the probabilistic information is encoded in the ratios of the vector elements, rather than their absolute values, in this work, a simplified approach, based on a successive bit shifting, is proposed. Assuming that the elements of a vector (before normalisation) are within the unity interval (i.e. for unsigned integer notation are smaller or equal  $(2^N - 1)/2^N$ ), they are logically shifted to the left, filling the rightmost bits with zeros, which is equivalent to successive multiplications by 2. After each shift, the overflow occurrence is checked and, if any of the elements is higher than one, the elements from the previous iteration (still within the unity interval) are taken as the final result. Such mechanism can easily be implemented as a state machine using  $N + 1$  bit shift registers for the vector elements and a logic circuit performing OR operation on the most significant bits of the elements to indicate the overflow occurrence. It should be noted that more precise normalisation procedure could be implemented assuming multiplications not by 2 (realised as logic shifts) but by an argument higher than one. Also, in the bit shifting approach, rather than filling the least significant bits of the register with zeros, the lower half of the  $2N$  bit result after multiplication could be used. These improvements, however, require additional hardware and increase the complexity of a design. In this research, the simplest hardware realisation was chosen. In some cases, as a result of the previously performed operation (e.g. dot product multiplication) and the rounding errors, a vector may only have zero elements, which should be detected before the normalisation. If that is the case, then the normalisation process should be skipped, issuing a vector with equal and non-zero elements representing uniform probability distribution (a neutral element in the performed calculations).

Slightly more complex state machine is required to normalise results of the matrix-vector multiplications, which involve additions, and may return numbers higher than one. In such a case, the left or right logic shifts of all the elements may be required, depending on the initial value of the largest element in the vector. If, for example, the largest element is higher than one, then all its elements have to be successively divided by 2 (i.e. iteratively right shifted) until all the elements are smaller or equal to one. Also, in order to avoid overflow errors in the matrix-vector multiplications, the corresponding

adders should operate on  $N + m$  bit words, where  $m$  is the number of added arguments, with a decimal point fixed after  $N$  less significant bits of the  $N + m$  bit result.

### 8.8.2 Hardware realisation

The test networks  $TN-5$  and  $TN-7$  were implemented as fully parallel synchronous digital circuits, where each Bayesian node has a fixed hardware module, consisting of a three-way variable and a three-way factor nodes, providing connectivity with up to four neighbouring Bayesian nodes (i.e. two parents and two children). Such approach was chosen because its structure corresponds to the analogue realisations considered in this research. The arithmetic operations performed by the variable and factor nodes are implemented using dedicated state machines computing the output messages and beliefs according to the equations (7.8) - (7.14) from Chapter 7. Each state machine is a simple digital processor equipped with a bank of registers and processing blocks for fixed point operations required by the implemented algorithm. In the realisations considered in this work, the three-way variable node consists of a single state machine, called  $V-3W$ , performing serial computation of the output messages and the corresponding beliefs. The three-way factor node consists of three identical state machines, called  $F-1W$ , computing the output messages in parallel for each link. Since the number of clock cycles required by the  $V-3W$  and  $F-1W$  state machines to complete a single message update is almost the same, such approach maximises the processing speed of the system.

The diagram showing the processing flow of a simplified version of the  $V-3W$  state machine, calculating only belief of the three-way variable node, is presented in Figure 8.16. Belief is calculated based on the three input messages  $[I_{X1}^T \ I_{X1}^F]$ ,  $[I_{X2}^T \ I_{X2}^F]$  and  $[I_{X3}^T \ I_{X3}^F]$  according to the equation (see Chapter 7 for reference):

$$\begin{bmatrix} Bel^T \\ Bel^F \end{bmatrix} = \alpha \cdot \begin{bmatrix} I_{X1}^T \cdot I_{X2}^T \cdot I_{X3}^T \\ I_{X1}^F \cdot I_{X2}^F \cdot I_{X3}^F \end{bmatrix} \quad (8.5)$$

where  $\alpha$  is a normalising factor. The computation of the three output messages (not shown in the diagram in Figure 8.16) is analogous and requires dot product of the three pairs of the input messages. The operations are performed using fixed point arithmetic with precision determined by the argument length  $N$ . The state diagram in Figure 8.16 consists of 9 states #1 - #9 representing particular arithmetic and logic operations performed by the circuit. In states #1 to #4, the dot product of the three input vectors is

calculated. Rounding to the  $N$  most significant bits after multiplication is done by saving only the higher halves of the obtained results.

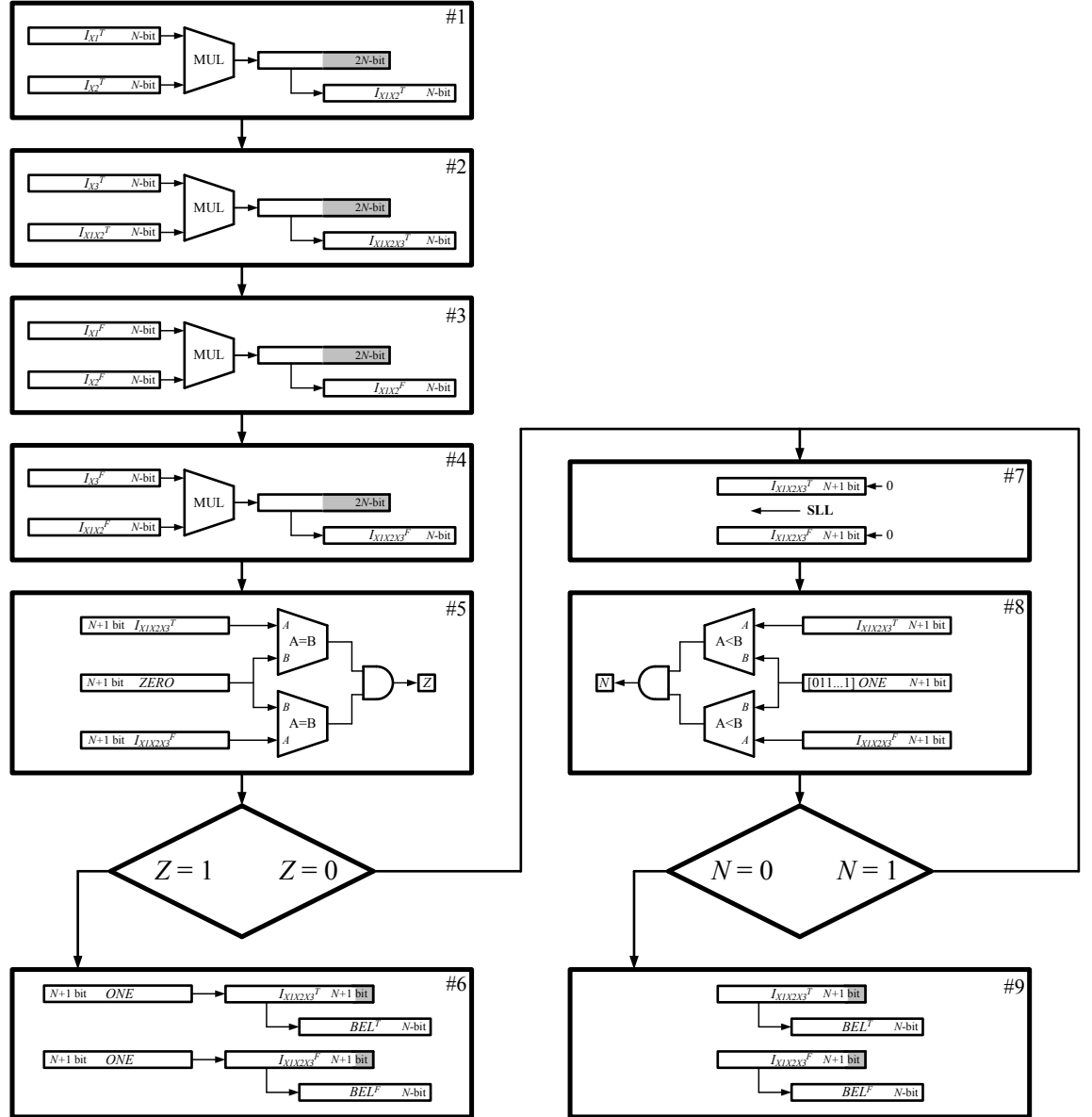


Figure 8.16. Block diagram of the simplified  $V$ -3 $W$  state machine for belief computation.

When the input messages include elements close or equal zero, the dot product  $[I_{X1X2X3}^T I_{X1X2X3}^F]$  may occasionally consist of only zero elements. This will be detected in state #5, where the elements of the computed result are compared with zero. If both of them are zero, the normalisation of the result is skipped and the vector representing the computed belief is assigned elements of the maximum value to represent the uniform probability distribution. In fact, any nonzero value from the unity interval could be assigned to the elements of the vector representing belief. The same approach applies to

the output messages computed in the network and was introduced to avoid possibility of clamping the state of a network or its part to zero, which may occur when zero messages start to circulate. In the majority of cases, the computed vector  $[I_{X1X2X3}^T I_{X1X2X3}^F]$  will consist of non zero elements requiring normalisation. The normalisation is performed in states #7 and #8, where the elements of the vector are first shifted logically to the left (multiplication by 2) and then compared with the value representing the probability of one. This process is repeated in a loop until any of the elements exceeds one. The result of the performed normalisation is the  $N$  most significant bits from the  $N + 1$  bit shift register, equal the result from the previous iteration.

### 8.8.3 Computational accuracy

The operation and the computational accuracy of the test networks  $TN-5$  and  $TN-7$ , implemented using simplified fixed point arithmetic approach, was verified in software (Matlab) for the input data from Tables 8.1 and 8.2. The simulations were performed for different values of the argument bit length  $N$ , in range from 5 to 10, defining the precision of the system. The obtained mean values and standard deviations of the computational error  $NCE_{SPA}$  are presented in Figure 8.17. In the evaluation of the computational error, the definition given by equation (8.3) was used assuming that  $I_{SIM}/||I_{SIM}||$  is the belief evaluated by the fixed point implementation and  $BEL_{SPA}$  is the reference result obtained using double precision floating point realisation. In the simulations of the fixed point realisations, a constant number of 20 iterations for both networks was used to assure convergence. For comparison, the parameters of the analogue implementations of the same networks (for the same data sets) are also included.

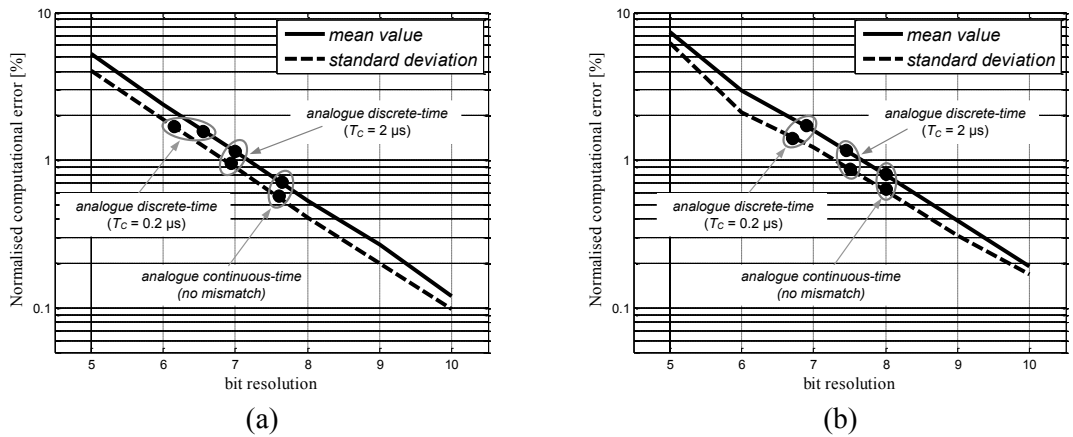


Figure 8.17. The computational error  $NCE_{SPA}$  in terms of the precision of the digital implementation obtained for the networks a)  $TN-5$ , and b)  $TN-7$ .

It can be observed that the computational precision of such systems improves exponentially with the number of bits  $N$ . The computational accuracy of the continuous-time analogue realisations reaches almost the same precision as the 8 bit digital equivalent, however, the simulations of the analogue circuit do not account for the parameter variability, therefore, this result does not reflect the real circuit behaviour. The precision of the continuous-time analogue system employing the average-based and scaling-based mismatch optimisation techniques decreases to about 6 - 7 bits, depending on the network. The accuracy of the discrete-time implementations remains within the range of 7 - 8 bits, assuming the cycle time  $T_C = 2 \mu\text{s}$  and decreases to 6 - 7 bits for  $T_C = 0.2 \mu\text{s}$ . The histograms of the error distribution of the  $TN$ -5 and  $TN$ -7 networks, for different values of parameter  $N$ , are presented in Figures 8.18 and 8.19. respectively.

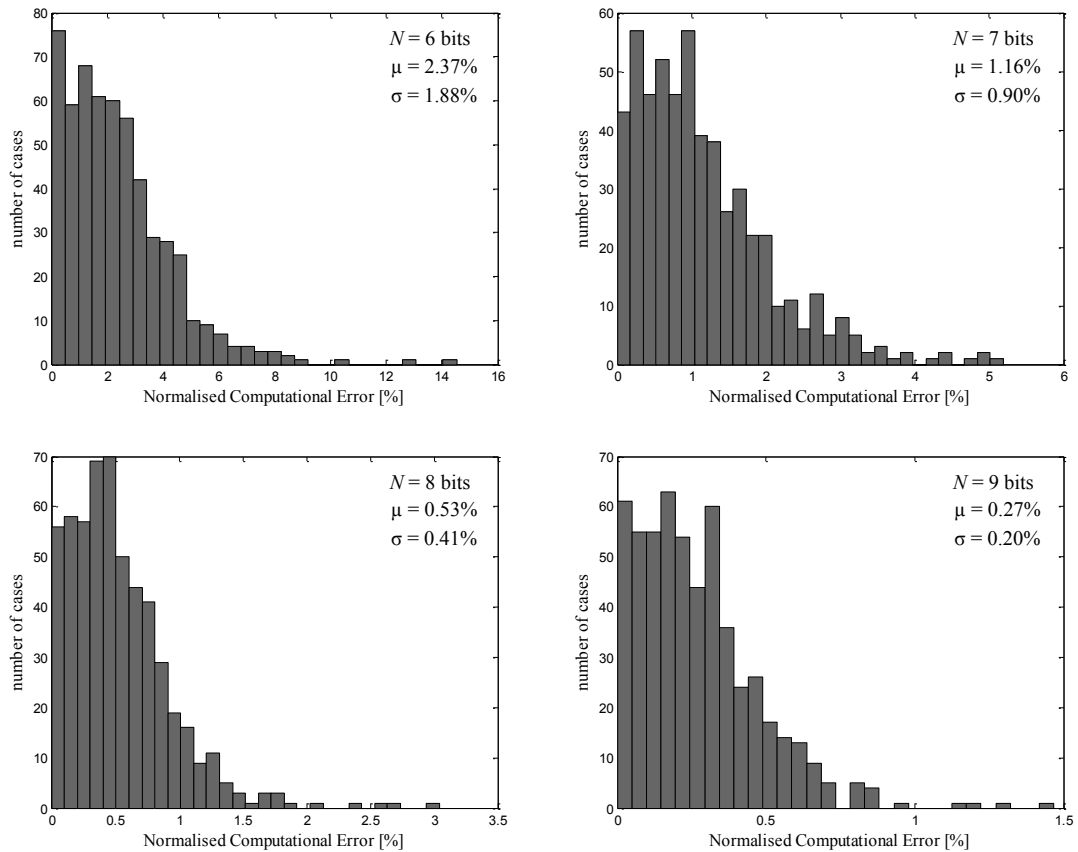


Figure 8.18. The histograms of the normalised computational errors of the  $TN$ -5 network implemented in the fixed point arithmetic.

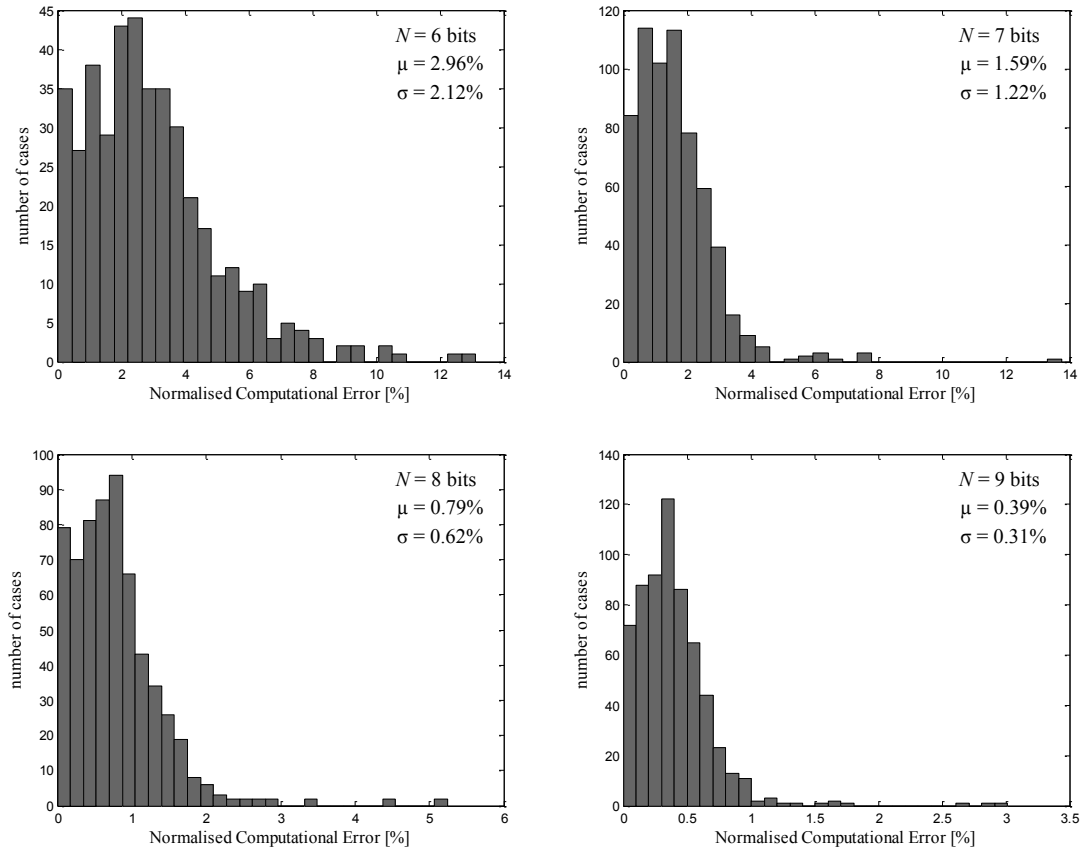


Figure 8.19. The histograms of the normalised computational errors of the *TN-7* network implemented in the fixed point arithmetic.

#### 8.8.4 Digital synthesis and implementation

The state machines *V-3W* and *F-1W* were synthesised from a parameterised VHDL behavioural description for argument length  $N$  equal from 5 to 10. For the logic synthesis and implementation Design Compiler and IC Compiler tools from Synopsys and RVT (regular threshold voltage,  $V_{CC} = 0.9$  V) standard cell libraries designed in a standard 90 nm CMOS technology were used. The synthesis report specifying the type, quantity and the size of the components recognised in the VHDL description, is summarised in Table 8.14.

Table 8.14. Components recognised by the synthesis tools in the *V-3W* and *F-1W* state machines.

<i>component</i>	<i>variable node V-3W</i>		<i>factor node F-1W</i>	
	<i>size</i>	<i>number</i>	<i>size</i>	<i>number</i>
multiplier	$N \times N$	1	$N \times N$	1
adder	---	0	$2N+4$	2
register	$N$	21	$N$	8
	$N+1$	2	$2N+4$	4
comparator	$N+1$	2	$2N+4$	2

The power prediction of the synthesised circuits was performed at two design stages: after synthesis and mapping, and after placement and routing. The former one provides estimation based solely on the power dissipation of the logic gates used in the design. The later one accounts also for the losses of the clock distribution network and wire switching. In both cases, the power consumption was estimated based on the assumed switching activity of the input ports. This could either be generated from a testbench or assigned manually, given the toggle rates of particular input lines. In this work, the second approach was chosen. The required switching activities of the input ports were estimated assuming the continuous operation of the state machines (i.e. there is no waiting time before each message update). Since the duration of the normalisation procedure depends on the elements of the input vector, in the behavioural simulations of the state machines, three uniform sets of input messages and parameters (equal 0.1, 0.5 and 0.9) were considered. The simulation results, showing the number of clock cycles required to complete one message update sequence in  $V\text{-}3W$  and  $F\text{-}1W$  state machines, are summarised in Table 8.15. Each parameter set is represented in percents.

Table 8.15. The number of clock cycles required to complete one message update sequence for three values of the input parameters equal 10%, 50% and 90%.

<b>Resolution <math>N</math></b>	<b># clock cycles (<math>V\text{-}3W</math>)</b>			<b># clock cycles (<math>F\text{-}1W</math>)</b>		
	10%	50%	90%	10%	50%	90%
5 bit	24	42	32	26	37	41
6 bit	24	42	32	26	39	43
7 bit	66	42	32	31	41	45
8 bit	66	42	32	29	43	47
9 bit	66	42	32	29	45	49
10 bit	66	42	32	31	47	51

It can be observed that the number of clock cycles required by the state machines depends on the input parameters and the resolution  $N$ . The shortest processing time of only 24 clock cycles (for the  $V\text{-}3W$ ) and 26 clock cycles (for the  $F\text{-}1W$ ) occurs for the lowest bit precision of 5 and 6 bits, and the smallest values of the input parameters, equal 0.1. In these cases, the low value of the input parameters and the reduced precision of the arithmetic operations generate zero vectors as a result of the dot product multiplications. Therefore, the normalisation sequence is omitted, shortening the processing time. The constant numbers of clock cycles for the input parameters equal 50% and 90% in the  $V\text{-}3W$  state machine, stems from the fact that the obtained dot product result scales linearly with  $N$ , and hence, requires the same number of logic shifts during the normalisation process. Also, larger input parameters (e.g. 90%) produce larger dot

product elements and the number of logic shifts required to normalise the result is lower. The situation is more complex in the case of the  $F-1W$  state machine, where small input values (e.g. 10%), as a result of the performed matrix-vector multiplication, generate vectors with elements smaller than one requiring logic left shifts during the normalisation. For larger input parameters (e.g. 50% and 90%) the results of the matrix-vector multiplications are larger than one and require divisions by 2 (right logic shifts) during the normalisation.

The  $V-3W$  and  $F-1W$  state machines were implemented in a standard 90 nm CMOS technology for the resolutions  $N$  from 5 bits to 10 bits, the same timing constraints and the clock period equal 10 ns. The synthesis and mapping steps were performed assuming leakage power and area optimisation. The placement and routing was performed to meet the specified timing constraints. The power prediction was performed assuming the switching activities corresponding to the input parameters of 50% for both state machines (see Table 8.15). The summaries of the obtained results for the  $V-3W$  and  $F-1W$  state machines, including the predicted dynamic and leakage power, area, and the time slack of the critical paths, are presented in Tables 8.16 and 8.17.

Table 8.16. The summary of the synthesis and implementation of the  $V-3W$  state machine in a standard 90 nm CMOS technology.

Resolution	Power [ $\mu$ W] @ 100 MHz		Area [ $\mu\text{m} \times \mu\text{m}$ ]	Slack [ns]	Max freq. [MHz]	Max speed [MOPS]	CE [TOPS/W]
	Synthesis <sup>1)</sup>	Implementation <sup>1)</sup>					
5 bit	123 + 10	142 + 11	67 $\times$ 65	7.23	360	69	0.124
6 bit	177 + 12	182 + 13	74 $\times$ 74	7.04	337	64	0.098
7 bit	205 + 14	212 + 14	80 $\times$ 79	6.71	304	58	0.084
8 bit	234 + 17	234 + 17	87 $\times$ 85	6.66	300	57	0.076
9 bit	261 + 19	266 + 19	92 $\times$ 90	5.91	244	46	0.067
10 bit	291 + 22	296 + 23	99 $\times$ 96	5.61	227	43	0.060

<sup>1)</sup> Power estimation: dynamic + leakage

Table 8.17. The summary of the synthesis and implementation of the  $F-1W$  state machine in a standard 90 nm CMOS technology.

Resolution	Power [ $\mu$ W] @ 100 MHz		Area [ $\mu\text{m} \times \mu\text{m}$ ]	Slack [ns]	Max freq. [MHz]	Max speed [MOPS]	CE [TOPS/W]
	Synthesis <sup>1)</sup>	Implementation <sup>1)</sup>					
5 bit	110 + 11	113 + 12	71 $\times$ 71	6.77	310	101	0.257
6 bit	129 + 13	178 + 14	76 $\times$ 76	6.71	304	94	0.160
7 bit	178 + 15	186 + 16	84 $\times$ 82	5.95	247	72	0.144
8 bit	199 + 18	219 + 19	90 $\times$ 90	6.13	258	72	0.117
9 bit	217 + 20	232 + 20	95 $\times$ 93	5.22	209	56	0.106
10 bit	241 + 23	253 + 23	101 $\times$ 99	4.86	189	48	0.093

<sup>1)</sup> Power estimation: dynamic + leakage

The maximum operating frequency and the maximum processing speed were calculated assuming that the clock period could be shortened by the estimated time slack. It is important to note, however, that faster circuits could be synthesised assuming different timing constraints at the beginning of the synthesis process. The maximum processing speed and the computational efficiency were estimated assuming that the  $V\text{-}3W$  state machine requires 42 clock cycles to complete the operation and performs 8 multiplications (operations). In the case of the  $F\text{-}IW$  circuit, the number of multiplications performed in one message update is 12 and the corresponding numbers of clock cycles were taken from Table 8.15 for the 50% column according to the bit resolution  $N$ . The power estimation, provided in Tables 8.16 and 8.17, account for the dynamic and leakage power respectively.

### 8.8.5 Performance of the digital realisation

The computational efficiency and the power performance of a single Bayesian node, consisting of one state machine  $V\text{-}3W$  and three state machines  $F\text{-}IW$ , was estimated based on the data from Tables 8.16 and 8.17. In the following, parameters of such system will be calculated for bit resolution  $N = 8$ . For the clock speed of 100 MHz, the power consumption of a Bayesian node processor (after implementation) is  $1 \times (234 + 17) + 3 \times (219 + 19) = 965 \mu\text{W}$ . The supply current is  $I_{VDD} = 1.07 \text{ mA}$  (for the default value of the supply voltage  $V_{CC} = 0.9 \text{ V}$  of the standard cell library used in the implementation). Such Bayesian processor performs  $1 \times 8 + 3 \times 12 = 44$  multiplications to generate belief and the output messages, and it requires (on average) 43 clock cycles. It should be noted that 8 bit implementations of  $V\text{-}3W$  and  $F\text{-}IW$  state machines require 42 and 43 clock cycles respectively (see Table 8.15 for  $N = 8$  and 50% input parameters), therefore,  $V\text{-}3W$  will have to wait one extra cycle. therefore, the number of cycles of the corresponding Bayesian processor is 43. Also, in the digital realisation of the  $V\text{-}3W$  state machine, first, the output messages are calculated and, after that, beliefs. Since the dot product of the message pairs are already computed, one of them can be used and multiplied by the third input message to evaluate belief. Therefore, only two additional multiplications are required to compute belief (not four as in the case of the analogue solution, see Table 8.6). For the assumed clock speed of 100 MHz, the considered Bayesian processor performs  $(44 \times 100) / 43 = 102 \text{ MOPS}$  with efficiency  $\text{CE} = 102 \text{ MOPS} / 965 \mu\text{W} = 0.106 \text{ TOPS/W}$ . The maximum processing speed is limited by the maximum operating frequency of the  $F\text{-}IW$  state machine, equal 258 MHz, and can be calculated as

$(44 \times 258) / 43 = 264$  MOPS. The area of such single processor can be estimated as  $1 \times (87 \times 85) + 3 \times (90 \times 90) = 178 \mu\text{m} \times 178 \mu\text{m}$ , and the processing speed  $43 \times 10 \text{ ns} = 430 \text{ ns}$ . The parameters of Bayesian nodes, for  $N = 5$  to 10, are summarised in Table 8.18

Table 8.18. The parameters of Bayesian processors estimated for different bit resolutions.

<b>Resolution</b>	<b>Power @ 100 MHz [<math>\mu\text{W}</math>]</b>	<b>Area [<math>\mu\text{m} \times \mu\text{m}</math>]</b>	<b># MUL/ # CLK</b>	<b>Max clock [MHz]</b>	<b>Speed @ 100 MHz [MOPS]</b>	<b>Max speed [MOPS]</b>	<b>CE [TOPS/W]</b>
5 bit	528	$140 \times 140$	44/42	310	105	325	0.199
6 bit	771	$151 \times 151$	44/42	304	105	318	0.136
7 bit	832	$164 \times 164$	44/42	247	105	259	0.126
8 bit	965	$178 \times 178$	44/43	258	102	264	0.106
9 bit	1041	$187 \times 187$	44/45	209	97.8	204	0.094
10 bit	1147	$199 \times 199$	44/47	189	93.6	177	0.082

The realisations of Bayesian processors communicating with larger number of neighbours than considered in this research, can easily be done by extending the state machines  $V\text{-}3W$  and  $F\text{-}IW$ . In general, a Bayesian processor communicating with  $K$  parents and  $L$  children will require  $(L+1)$ -way variable node and  $(K+1)$ -way factor nodes. Due to the fact that the number of the required operations does not scale in the same way for the variable and factor nodes (see Tables 8.4 and 8.5), the system assuring maximum processing speed requires proper design of the state machines, "synchronised" in terms of the processing speed. In order to assure maximum processing speed, all the state machines should ideally finish their tasks at the same time (i.e. after the same number of clock cycles). In practice, for different  $K$  and  $L$ , and different numbers of states of the processed variables, such designs could be optimised by using time multiplexing, serialising short computation sequences and parallelising the long ones. For example, in the diagram presented in Figure 8.16, states #1, #3, and #2, #4 could be executed in parallel, reducing the number of the required clock cycles, but at the expense of the additional area and power consumed by the second multiplier. For larger number of links, e.g.  $L = 12$ , each element of the vector dot product requires multiplication of 12 numbers which can be done in parallel in 4 steps using 6 multipliers, or in 5 steps using only 3 multipliers. In such cases, the normalisation of the intermediate results should also be considered in order to avoid underflow errors, which may further increase the number of the required clock cycles. Such optimisation could also be done for the factor node, however, the uniform processing speed of the system may be difficult to achieve when the number of parents and the number of children of a particular Bayesian processor are significantly different. In such a case, the node with the larger number of links will have

to perform more operations in parallel to produce the result in the same time as the node with the lower number of links. This will increase the power and area of the hardware realisation, which in some practical realisations may not be acceptable.

In the case of the networks considered in this research, using fixed implementation of Bayesian processors, the power and area will scale linearly with the number of nodes in the network. In the mini asic design of  $1500\text{ }\mu\text{m} \times 1500\text{ }\mu\text{m}$  active circuit area (excluding the I/O ring), the possible number of nodes operating with 8 bit precision, that could be implemented, assuming  $20\text{ }\mu\text{m}$  spaces between the nodes for I/O and memory registers, is approximately equal  $(1500 \times 1500) / (190 \times 190) = 62$ . The approximate power consumption of such circuit operating at 100 MHz clock is  $62 \times 965\text{ }\mu\text{W} = 59\text{ mW}$ , and may increase up to 147 mW for the maximum clock speed of 250 MHz. In the full chip realisation of  $10\text{ mm} \times 10\text{ mm}$  die size, the number of Bayesian nodes is 2770, and the consumed power is 2.6 W for the clock speed of 100 MHz. The estimated maximum processing speed is 730 GOPS consuming 6.6 W of power. It should be noted that, in the provided estimations, the number of operations is equal to the number of two-argument multiplications. The actual processing capabilities of such system are much higher, when accounting for the data summations, transfers and normalisations.

Although the estimated performance figures are very promising in terms of network scaling, the scalability of the computational error, resulting from the use of the fixed point arithmetic, need to be further investigated. The use of the simplest approach to the implementation of the fixed point arithmetic, presented in this thesis, was mainly motivated by the possibility of its straightforward comparison to the analogue solutions. The design of more complex circuits for fixed point arithmetic could be considered in order to reduce the computational errors. In particular, designs aiming optimisation of the processing error, area and power consumption by using computational blocks of mixed bit precision and variable time multiplexing, could be investigated in the future designs.

## 8.9 Performance comparison

This section provides the overview and comparison of three different approaches to computation considered in this research, accounting for the analogue and digital hardware realisations, and two software solutions, implemented in Matlab and C++ language for PC. The corresponding figures of performance are estimated for the implementations of the *TN-5* and *TN-7* test networks. Details concerning each particular

realisation are discussed further in the following sections. The performance analysis and comparison is provided in a separate section discussing the obtained results.

### 8.9.1 Analogue implementation

The implementations of the three-way variable and three-way factor nodes in analogue circuits, require 14 and 29 Gilbert multipliers respectively (see Tables 8.4 and 8.5). Assuming that the area of a Gilbert multiplier, realised in a standard 90 nm CMOS technology, is  $6.8 \mu\text{m} \times 10 \mu\text{m}$ , the approximate area of a three-way variable node is  $31 \mu\text{m} \times 31 \mu\text{m}$  and the area of the three-way factor node is  $44 \mu\text{m} \times 44 \mu\text{m}$ . Based on that, the area of a single Bayesian node with 2 parents and 2 children is  $54 \mu\text{m} \times 54 \mu\text{m}$ . It should be noted that some additional area will be required to implement current sources to store and generate the network parameters.

The architecture of the discrete-time analogue realisation is practically the same as the continuous-time one and also requires 14 Gilbert multipliers for the three-way variable node and 29 multipliers for the three-way factor node. The discrete-time Gilbert multiplier consists of 5 memory cells (see Figure 3.11). Three of them use an information storing transistor of size  $1 \mu\text{m} \times 1 \mu\text{m}$  and two of them use a thick gate oxide transistors of the size  $1.8 \mu\text{m} \times 0.8 \mu\text{m}$ . Assuming, for simplicity, that each memory cell requires area of  $2 \mu\text{m} \times 2 \mu\text{m}$  (accounting for the information storing transistor, transmission gates and the cascoding transistors), the estimated area of a three-way variable node is  $17 \mu\text{m} \times 17 \mu\text{m}$ , and the estimated area of the three-way factor node is  $24 \mu\text{m} \times 24 \mu\text{m}$ . Based on that, the area of a single Bayesian processor communicating with 2 parents and 2 children is  $30 \mu\text{m} \times 30 \mu\text{m}$ .

Based on the provided estimations, the area of the continuous-time realisation of the *TN-5* network is  $5 \times 54 \mu\text{m} \times 54 \mu\text{m} = 121 \mu\text{m} \times 121 \mu\text{m}$ , and of the *TN-7* network is  $7 \times 54 \mu\text{m} \times 54 \mu\text{m} = 143 \mu\text{m} \times 143 \mu\text{m}$ . The area of the discrete-time realisation of the *TN-5* network is  $5 \times 30 \mu\text{m} \times 30 \mu\text{m} = 67 \mu\text{m} \times 67 \mu\text{m}$ , and the area of the *TN-7* network is  $7 \times 30 \mu\text{m} \times 30 \mu\text{m} = 79 \mu\text{m} \times 79 \mu\text{m}$ .

### 8.9.2 Digital implementation

The performance figures of the digital implementation of the *TN-5* and *TN-7* test networks were estimated based on the parameters of a single Bayesian processor realised in a standard 90 nm CMOS technology and operating at the maximum clock frequency (see Table 8.18). In the following, the calculations concerning realisations with 8 bit

precision will be provided assuming linear scaling of the power and processing speed with the clock frequency. The maximum power consumption of the *TN-5* network implementation is  $5 \times 965 \mu\text{W} \times 258 / 100 = 12.45 \text{ mW}$  and of the *TN-7* network is  $7 \times 965 \mu\text{W} \times 258 / 100 = 17.43 \text{ mW}$ . Assuming 43 clock cycles for each message update (in the 8 bit architecture), the minimum processing time of the *TN-5* network is  $43 \times 3.88 \text{ ns} \times 16 = 2.67 \mu\text{s}$ , and in the case of the *TN-7* network, the minimum processing time is  $43 \times 3.88 \text{ ns} \times 12 = 2.00 \mu\text{s}$ . The maximum processing speed of the *TN-5* network is  $5 \times 264 \text{ MOPS} = 1320 \text{ MOPS}$  and of the *TN-7* network is  $7 \times 264 \text{ MOPS} = 1848 \text{ MOPS}$ . The area of the *TN-5* implementation is  $5 \times (178 \mu\text{m} \times 178 \mu\text{m}) = 398 \mu\text{m} \times 398 \mu\text{m}$ , and the area of the *TN-7* network is  $7 \times (178 \mu\text{m} \times 178 \mu\text{m}) = 471 \mu\text{m} \times 471 \mu\text{m}$ .

### 8.9.3 Software implementation for PC

The software realisations of belief propagation algorithm in the *TN-5* and *TN-7* networks were implemented and tested in Matlab 2012a environment, and in C++ using Microsoft Visual Studio 2008. Similarly as before, it was assumed that the networks consist of Bayesian nodes communicating with two parents and two children, and perform a fixed set of arithmetic operations (see Figure 7.6 in Chapter 7). In both cases, only the fundamental data types and standard coding techniques were employed, assuming no parallelisation or GPU use. In Matlab, the profiler tool was used to optimise the code in terms of speed. The compilation of the C++ sources was performed assuming speed optimisation. The performance of both solutions was verified using an off-the-shelf PC with Intel Core i7 950 processor, 6 GB RAM, running Windows 7 operating system. In the processing time estimation, only the algorithm runtime was measured, excluding the variable initialisation and the output log generation. In the Matlab environment, functions *tic* and *toc* were used for code timing. Such method allowed to measure the execution time with resolution of about 1  $\mu\text{s}$ . In the C++ implementations, the state of the *Time Stamp Counter* (TSC) was used for code timing [Paoloni 2010]. In order to improve the precision of the time measurement, multiple iterations of the message passing scheme were executed. In particular, in Matlab realisations,  $10^5$  iterations were assumed, and in C++ implementations,  $10^7$  iterations were assumed. The processing speed was calculated assuming that each Bayesian node performs 44 operations (i.e. two-argument multiplications, see Section 8.8.5) for each message update. The obtained mean processing speed figures (calculated based on five runs), were: 1.44 MOPS for the *TN-5*

and *TN-7* networks in the Matlab realisations. The reported processing time of the *TN-5* network was 1.53 s and of the *TN-7* network was 2.13 s (for  $10^5$  iterations). For the C++ implementations, the processing speed was equal to 662 MOPS, for the *TN-5* network, with processing time of 3.34 s, for  $10^7$  iterations, and 647 MOPS, for the *TN-7* network, with processing time of 4.84 s, for  $10^7$  iterations. The maximum power dissipation of the Intel i7 950 processor is equal to 130 W and the chip die area is  $263 \text{ mm}^2$  [Intel 2014]. The power of one core can be estimated as  $130 \text{ W} / 4 = 32.5 \text{ W}$ , and the area as  $263 \text{ mm}^2 / 4 \approx 8110 \text{ } \mu\text{m} \times 8110 \text{ } \mu\text{m}$ . The corresponding processing efficiency of both networks is approximately equal to 20 MOPS/W (C++ implementations) and 0.04 MOPS/W (Matlab implementation). The processing speed and power efficiency of the hardware and software realisations discussed in the previous sections are summarised in Tables 8.19 and 8.20 for the *TN-5* and *TN-7* test networks respectively.

Table 8.19. The parameters of the *TN-5* network in analogue and digital realisations (comparison).

<b>Realisation</b>	<b>Power [mW]</b>	<b>Time [<math>\mu\text{s}</math>]</b>	<b>Area [<math>\mu\text{m} \times \mu\text{m}</math>]</b>	<b>Speed [MOPS]</b>	<b>CE [TOPS/W]</b>
Analogue continuous-time (no mismatch)	0.231	0.56	$121 \times 121$	6,571	23.7
Analogue continuous-time (average-based technique)	23.1	0.56	$1210 \times 1210$	6,571	0.237
Analogue continuous-time (scaling-based technique)	0.219	161	$2096 \times 2096$	22.85	0.087
Analogue discrete-time ( $T_C = 2 \text{ } \mu\text{s}$ )	0.051	450	$67 \times 67$	8.18	0.161
Analogue discrete-time ( $T_C = 0.2 \text{ } \mu\text{s}$ )	0.056	45	$67 \times 67$	81.8	1.45
Digital 5 bit @ 310 MHz	8.18	2.17	$313 \times 313$	1,625	0.199
Digital 6 bit @ 304 MHz	11.72	2.21	$338 \times 338$	1,590	0.136
Digital 7 bit @ 247 MHz	10.28	2.72	$367 \times 367$	1,295	0.126
Digital 8 bit @ 258 MHz	12.45	2.67	$398 \times 398$	1,320	0.106
Digital 9 bit @ 209 MHz	10.88	3.44	$418 \times 418$	1,020	0.094
Digital 10 bit @ 189 MHz	10.84	3.98	$445 \times 445$	885	0.082
Digital (Intel i7, C++) 64 bit FP @ 3.00 GHz	32,500	5.34	$8110 \times 8110$	662	$20 \times 10^{-6}$
Digital (Intel i7, Matlab) 64 bit FP @ 3.00 GHz	32,500	245	$8110 \times 8110$	1.44	$4 \times 10^{-8}$

Table 8.20. The parameters of the *TN-7* network in analogue and digital realisations (comparison).

<i>Realisation</i>	<i>Power</i> [mW]	<i>Time</i> [ $\mu$ s]	<i>Area</i> [ $\mu$ m $\times$ $\mu$ m]	<i>Speed</i> [MOPS]	<i>CE</i> [TOPS/W]
Analogue continuous-time (no mismatch)	0.393	0.62	143 $\times$ 143	6,232	15.9
Analogue continuous-time (average-based technique)	32.8	0.62	1430 $\times$ 1430	6,232	0.159
Analogue continuous-time (scaling-based technique)	0.393	133	2477 $\times$ 2477	29.05	0.074
Analogue discrete-time ( $T_C = 2 \mu$ s)	0.074	340	79 $\times$ 79	11.36	0.153
Analogue discrete-time ( $T_C = 0.2 \mu$ s)	0.082	34	79 $\times$ 79	113.6	1.38
Digital 5 bit @ 310 MHz	11.46	1.62	370 $\times$ 370	2,275	0.199
Digital 6 bit @ 304 MHz	16.41	1.66	400 $\times$ 400	2,226	0.136
Digital 7 bit @ 247 MHz	14.39	2.04	434 $\times$ 434	1,813	0.126
Digital 8 bit @ 258 MHz	17.43	2.00	471 $\times$ 471	1,848	0.106
Digital 9 bit @ 209 MHz	15.23	2.58	495 $\times$ 495	1,428	0.094
Digital 10 bit @ 189 MHz	15.17	2.98	527 $\times$ 527	1,239	0.082
Digital (Intel i7, C++) 64 bit FP @ 3.00 GHz	32,500	5.8	8110 $\times$ 8110	647	$20 \times 10^{-6}$
Digital (Intel i7, Matlab) 64 bit FP @ 3.00 GHz	32,500	256	8110 $\times$ 8110	1.44	$4 \times 10^{-8}$

### 8.9.4 Discussion

The relation between the processing speed and the consumed power is presented in Figure 8.20 (the data from both tables are included). The fastest solutions, providing the processing speed of over 6 GOPS, are realised in analogue circuits operating in the continuous-time mode, assuming no parameter mismatch, and employing the average-based mismatch optimisation. In such solutions, the settling time of the analogue circuit does not depend on the number of network copies, therefore, it preserves the processing speed at the expense of the consumed power. In other words, in the average-based mismatch reduction method, power is traded for precision, shifting the solution along the horizontal axis in the design space presented in Figure 8.20. The second approach to mismatch optimisation, based on the transistor size scaling, preserves power but at the

expense of the processing time. Larger transistors exhibit higher gate capacitances, therefore the corresponding settling time of the circuit increases. As a result, scaling-based approach trades processing speed for accuracy and shifts the solution down along the vertical axis, as shown in Figure 8.20.

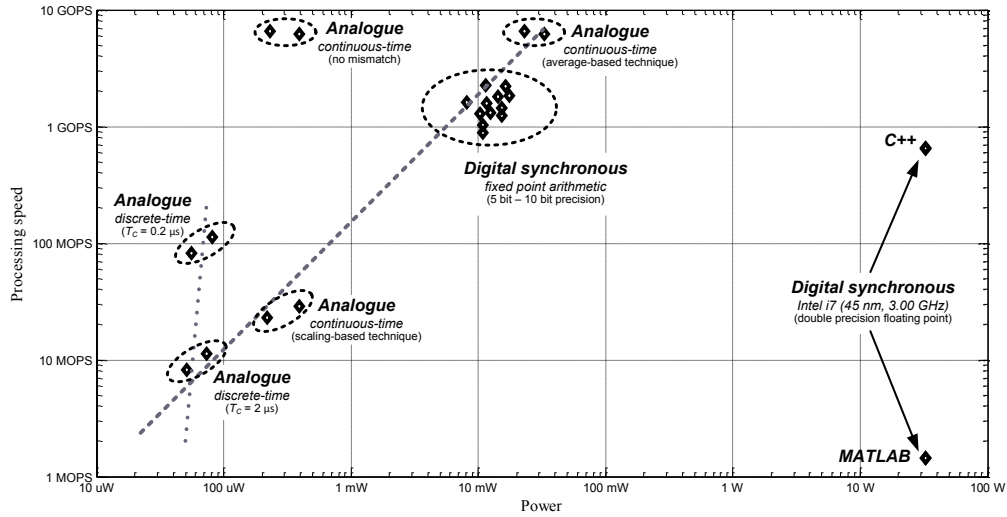


Figure 8.20. The design space of the computational systems defined by the processing speed and power consumption.

Hardware realisations, such as the digital synchronous (with fixed point arithmetic), the discrete-time analogue (with  $T_C = 2 \mu s$ ), and the continuous-time analogue (with mismatch optimisation applied), were designed to operate with a similar computational accuracy. It can be observed that all three solutions are located close to the line representing a proportion between the power and the processing speed (dashed grey line in Figure 8.20). It suggests that the processing efficiency, defined as the ratio of the processing speed and power, may be limited for a given computational accuracy in a particular technology, irrespective of the assumed approach and circuit realisation. Such conclusion, however, refers to the solutions considered in this research. One possibility to depart from the observed trend and improve the processing speed or efficiency, is based on scaling of the cycle time  $T_C$  in the discrete-time realisations. It can be observed that shortening the cycle time from  $2 \mu s$  to  $200 \text{ ns}$  increases the processing speed by factor of ten but reduces the computational accuracy by 30% (see Section 8.5.1). In such approach, the corresponding proportion between the processing speed and the power consumption is much steeper (dotted grey line in Figure 8.20) than in the previous case. This is very promising in terms of the high efficiency analogue computing, however, needs to be further investigated, especially in terms of the processing accuracy.

The software realisations for PC, considered in this research, exhibit significantly different processing speed and power efficiency, even though both were implemented and tested on the same hardware platform. In particular, the implementation in C++ language is almost three order of magnitude faster than its equivalent realised in Matlab environment. In both cases, programs were written in the same way assuming fixed, hard coded matrix-vector operations with no generic structures such as loops, branching or dynamic memory preallocation (i.e. the code was written entirely for a particular network structure). Since the program execution in Matlab is based on code interpreting, the corresponding overhead reduces the achievable processing speed, when compared to the approach employing code compilation (e.g. C++ language). In the C++ realisations, the number of the clock cycles spent on the message computation and update for one Bayesian node can be estimated assuming that  $10^7$  iterations of the *TN-5* network realisation takes 3.34 s resulting in  $(3 \times 10^9 \times 3.34) / (5 \times 10^7) \approx 200$  clock cycles per node. Given that such node requires 79 arithmetic operations (i.e. 44 two-argument multiplications, 28 two-argument additions, and 7 divisions, see Figure 7.6 in Chapter 7), the average number of cycles per arithmetic operation and the related data traffic, is about 2.5. For a regular network structure, presented in Figure 8.15, and consisting of  $N$  nodes, the number of iterations to attain convergence (proportional to the number of nodes along the diagonal of the square) is equal to  $2\sqrt{2N}$  (accounting for the forward and backward propagation). Assuming that each Bayesian node requires 200 clock cycles for message update, and 3 GHz clock speed of the processor, networks consisting of about 30,000 nodes could be solved in one second using an off-the-shelf PC. This, however, is only an estimation assuming the use of fixed Bayesian nodes communicating with only two parents and two children. In general, the number of clock cycles and iterations will depend on the size of the nodes and the structure of the network.

A comparison of the processing speed and the efficiency of the hardware and software solutions, considered in this research, are presented in Figures 8.21, 8.22 and 8.23. Figure 8.21 shows the processing speed of different solutions. It should be noted, however, that different realisations exhibit different design approaches and different levels of time-multiplexing. Figure 8.22 shows the computational efficiency as a ratio of the processing speed and the consumed power, as defined in the equation (8.5). Figure 8.23 shows the processing efficiency calculated as a ratio of the operation speed and the

product of power and area occupation. It can be interpreted as a measure of energy and silicon utilisation in terms of the performed computation.

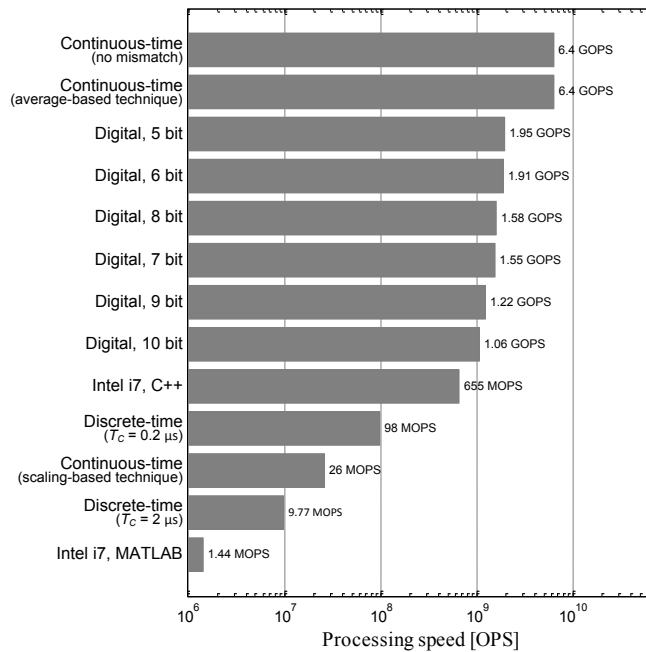


Figure 8.21. The comparison of the processing speed of different hardware and software solutions (measured as the number of two-argument multiplications per second).

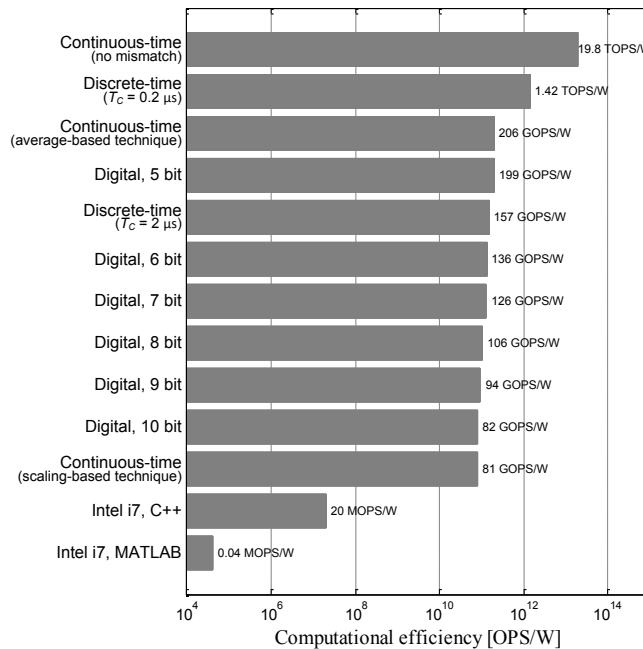


Figure 8.22. The comparison of the processing efficiency of different hardware and software solutions (measured as the number of two-argument multiplications per second per watt of power).

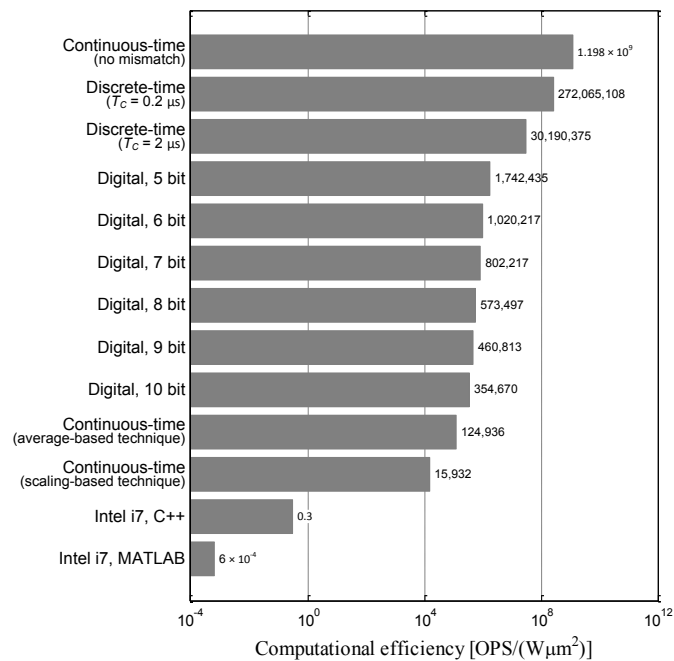


Figure 8.23. The comparison of the processing efficiency of different hardware and software solutions (measured as the number of two-argument multiplications per second per watt of power per square micron of the silicon area).

In the Figures, the continuous-time analogue realisation (not accounting for mismatch) exhibits the highest processing speed and power/area efficiency. Such solution, however, is only theoretical, since parameter variability cannot be avoided in practice. The processing speed of the discrete-time analogue realisation is lower and outperforms only the software realisation in Matlab. However, due to its very low power consumption and area occupation, the corresponding processing efficiency is over one order of magnitude higher than any other practical solution. The processing efficiency of the synchronous digital and the continuous-time analogue solutions (with average-based mismatch optimisation) remains within the same order of magnitude and varies in range from 80 to 200 GOPS/W. These solutions are located along the dashed grey line in Figure 8.20. The processing speed of the C++ software realisation is comparable with the performance of the synchronous digital solutions, however, the efficiency is over four orders of magnitude lower than the least efficient hardware solution. It should be noted however, that the software realisations employ double precision floating point operations, and are executed on a processor realised in a 45 nm technology. Therefore, a direct comparison of these solutions is not straightforward. More reasonable comparison could

be made using PC with a processor fabricated in a 90 nm technology or implementing the proposed hardware solutions in a 45 nm technology.

The dedicated hardware solutions, considered in this research, exhibit several orders of magnitude higher processing efficiency and area utilisation than their functional equivalents in software running on a PC. Assuming comparable computational accuracy, the continuous-time analogue circuit realisations exhibit the highest processing speed, almost 6 times faster than the dedicated fixed point digital, and almost one order of magnitude faster than a floating point solution running on a general purpose processor.

The processing speed of the continuous-time analogue circuit is only limited by the currents representing computed quantities and capacitances of the gates of MOS transistors and wires. However, a high impact of the fabrication mismatch reduces the computational precision of such solutions below a typically acceptable level. The average-based and the scaling-based mismatch optimisation methods, proposed in this research, improved the computational accuracy but at the expense of the increased power consumption or area occupation, which in turn reduced the processing efficiency. Despite a very high processing speed, the continuous-time analogue approach requires large area, and hence, is probably too expensive for realisations in standard CMOS technologies, given the fact that its 8 times slower digital equivalent (for  $N = 7$ ) occupies 10 times less area and consumes about 7 times less power.

The processing speed of the discrete-time analogue realisations is the lowest among the three hardware solutions. However, the low power consumption results in a high processing efficiency of about 1.4 TOPS/W. Assuming that the processing accuracy of the discrete-time circuits is comparable to a 7 bit fixed point digital realisation, the efficiency of the discrete-time solution is almost 30 times higher. Given their very low power and potential for time-multiplexing, they provide a good alternative for digital computation in applications where power and area is of the main concern.

## 8.10 Conclusions

This chapter presented the comparison of different realisations of analogue and digital computing hardware, dedicated for belief propagation algorithm. Based on the obtained results, no practical circuit realisation could be favoured in terms of the processing speed and efficiency at the same time.

The discrete-time analogue solutions are the most promising in terms of the processing efficiency. Nevertheless, more research is needed to investigate the scalability of the computational accuracy, both with the processing speed, and the network size. The dedicated synchronous digital solutions provide a good processing speed and efficiency, sufficient for many practical applications. However, more research is needed to verify the convergence of larger networks, when using fixed point arithmetic with reduced computational precision. Also, structures exhibiting different levels of time-multiplexing should be considered for more efficient network realisations.

The software solutions employing the hard coded arithmetic matrix-vector operations could be considered in the applications requiring high computational precision. In particular, the scalability of such software realisations in terms of the processing speed and efficiency, with network complexity and the size of the resulting program, should further be investigated.

## Chapter 9

---

# Conclusions

---

### 9.1 Research summary

The objective of the research presented in this thesis was to investigate the alternative ways of information processing employing asynchronous (data driven), and analogue computation in massively parallel cellular processor arrays, with applications in machine vision and artificial intelligence. The use of cellular processor architectures, with only local neighbourhood connectivity, was considered in the VLSI hardware realisations of the trigger-wave propagation in binary image processing, and in belief propagation in Bayesian inference. Design issues, critical in terms of the computational precision and system performance, were extensively analysed, accounting for the non-ideal operation of MOS devices caused by the second order effects, noise and parameter mismatch. In particular, CMOS hardware solutions for two specific tasks: binary image skeletonization and sum-product algorithm for belief propagation in factor graphs, were considered, targeting efficient design in terms of the processing speed, power, area, and computational precision. In the research, various analogue and digital circuit realisations, operating in the continuous-time and discrete time domains, were analysed in theory and verified using combined Matlab-Hspice simulation environment, providing a versatile framework, suitable for arbitrary analyses, verification, optimisation of the designed systems. Novel circuit solutions, exhibiting a reduced impact of parameter variability, such as discrete-time current-mode Gilbert multiplier and output-split inverter delay gate, were used in the designs of the arithmetic circuits for matrix-vector operations, and in the

data driven asynchronous processing arrays. The most promising circuit ideas were implemented on the PPATC test chip, fabricated in a standard 90 nm CMOS process, and verified in experiments.

### 9.1.1 Binary image processing

In this thesis, the implementation of the trigger-wave propagation for morphological operations on binary images was considered using asynchronous CMOS cellular logic arrays. The proposed hardware realisation of the trigger-wave propagation mechanism extends its functionality to detect collisions between the wave-fronts, and hence, enables the binary image skeletonization and Voronoi diagrams generation. Low power and high processing speed requirements were achieved by employing asynchronous dynamic logic design. Critical design issues, such as current leakage and parameter variability, affecting the correct circuit operation, were resolved by proper MOS transistor scaling, and by employing the delay gate design based on the output-split inverter circuit, exhibiting reduced impact of fabrication mismatch on the generated time intervals. The quality of the obtained skeletons was further improved by a novel biasing scheme of the propagation gate, enabling the generation of approximately circular waves. The operation of the circuit was verified in simulations and in experiments with the fabricated prototype array consisting of  $64 \times 96$  cells. The fabricated prototype logic array is capable of processing up to  $2.78 \times 10^6$  images per second consuming less than 2 mW of power. The proposed asynchronous logic array could be of use in the future designs of vision chips as a co-processing layer, dedicated for fast and low power morphological operations on binary images, extending the application domain of such circuits to skeletonization and Voronoi diagram extraction.

### 9.1.2 Delay lines

The operation of the output-split inverter delay gate (OSI), exhibiting less timing parameter variability than the commonly used current starved inverter (CSI), was verified in simulations, for three different technology nodes (180 nm, 90 nm and 65 nm), and in experiments with the prototype delay line arrays, implemented on the PPATC test chip. The obtained experimental results showed almost twice better matching properties of the proposed structure, achieved solely by modifying the biasing scheme of the current starved inverter gate, with no additional cost in terms of the consumed power or circuit area. The theoretical analysis of the dynamic operation of the CIS and OSI delay gates,

presented in this thesis, showed significant differences in their dynamic behaviour, which are of a high importance when process parameter variability is concerned. The proposed OSI gate structure could be considered in the applications requiring multiple tunable delay elements of matched parameters with strict area constraints, for example, in the build of readout systems for particle detectors, or in neuromorphic circuits. In this thesis, the proposed OSI structure was used in the design of the asynchronous logic array for data-driven image processing, improving the precision of the collision detecting layer, and hence, improving the quality of the extracted skeletons.

### 9.1.3 Probabilistic reasoning

In this thesis, several approaches to hardware and software realisations of the sum-product algorithm, dedicated for belief propagation in factor graphs, were considered. In particular, hardware implementations of the factor graphs in the dedicated analogue and digital cellular processor arrays, operating in the continuous-time and discrete-time modes, were further investigated. In general, the continuous-time analogue realisations have the potential for processing with a high speed and power-efficiency, easily outperforming any other approach, of a comparable precision, considered in this research. However, the high susceptibility to parameter mismatch, and no alternative for the time multiplexed realisations, are the major factors limiting the use of such circuits in many practical applications. Two mismatch optimisation techniques, applicable to the continuous-time solution, based on the transistor size scaling and the result averaging, were proposed and tested. Both techniques were shown successful in terms of improving the computational precision, however, at the cost of processing efficiency and significant area increase.

The most promising solution, in terms of power consumption and area occupation, is based on the discrete-time analogue processing. Although the processing speed of these solutions is relatively low, it has been shown that it scales up with power much better than other solution considered in this work and has potential for time multiplexed computation. Therefore, it is a very promising alternative, outperforming other solutions in terms of processing efficiency, which could be considered in the applications requiring very low power and low area designs. The major issue of such approach is the limited computational accuracy, caused mainly by the charge injections and second order effects in MOS devices. This could be further reduced employing more robust memory cells and using CMOS technologies dedicated for low leakage designs.

Based on the results presented in this thesis, it can be concluded that the processing efficiency of different hardware realisations remains within the same order of magnitude, irrespective of the processing method employed, as long as the computational precision remains on the same level. Therefore, not only analogue but also dedicated digital solution, employing the fixed point arithmetic operations, should definitely be considered in the search for faster and more efficient ways of information processing, especially in parallel and time multiplexed processor architectures.

#### 9.1.4 Contributions

The major contributions of the research presented in this thesis are:

- Analysis and design of the collision detecting layer for trigger-wave propagation-based image processing algorithms in dynamic logic CMOS circuit combining logical AND function and 1 bit memory latch, using only 8 MOS transistors.
- Analysis and design of the propagation gate for trigger-wave propagation-based image processing algorithms with a novel bias scheme allowing for the generation of the circular wave contours, difficult to achieve in software or using generic SIMD processor arrays.
- Analysis and design of a delay gate employing a novel biasing scheme resulting in almost twice better matching properties, when compared to the commonly used current starved inverter, with no penalty in terms of power or area.
- Analysis and design of the analogue CMOS discrete-time variant of the Gilbert multiplier, operating in current mode with computational accuracy comparable to its commonly used continuous-time equivalent.
- Analysis and design of an optimised digital fixed-point arithmetic circuits for matrix-vector operations with applications in probabilistic calculus and other areas requiring computation with normalised data.
- Analysis of the power, area and complexity scaling of the hardware realisations of the factor graphs for belief propagation in analogue circuits.
- Development and verification of the mismatch optimisation techniques based on the novel biasing scheme (OSI delay gates), results averaging (Bayesian networks in analogue continuous-time circuits) and switched-current technique (discrete-time current-mode multipliers).

## 9.2 Future work

### 9.2.1 Image processing

The proposed prototype array for binary image skeletonization could easily be adopted to the design of a generic asynchronous co-processing unit, applicable to a variety of morphological operations, aiding fast and low power processing on SIMD arrays in vision chips. In particular, design improvements discussed in Chapter 5, addressing boundary effects, power rail oscillations and modified initialisation scheme, should be considered in the future designs. Special attention should also be paid to the leakage currents in larger arrays, occasionally triggering spurious propagations. This could be addressed by the use of low leakage and/or high threshold voltage devices, and power supply reduction. It should be noted, however, that the use of transistors other than regular, may result in increased parameter mismatch. It is also important to verify the scalability of such solution in terms of the uniformity of the propagation speed, impact of the systematic errors on the quality of the extracted collision lines, and the circular shape of the propagation wave contours, for larger array sizes, realised in a particular CMOS technology node.

### 9.2.2 Bayesian inference

The scope of the research concerning the probabilistic reasoning in analogue VLSI, presented in this thesis, was limited to the analysis of networks represented by factor graphs, operating on two-state variables and consisting of only three-way nodes. It provided a valuable contribution to the current state-of-the-art literature, extending the application domain of the hardware realisations of factor graphs to account for belief propagation in Bayesian inference. Although the main objective of this research was to provide a baseline comparison of the performance and computational accuracy of different analogue and digital realisations, several issues could be seen as a direct continuation, building upon the presented results.

In this research, the scalability of factor graph realisations in analogue continuous-time CMOS circuits was investigated in detail in terms of power, size and computational complexity and processing accuracy. Since the discrete-time analogue and the dedicated digital solutions (employing the fixed point arithmetic) were found promising in terms of the processing efficiency, the scalability of the processing speed, accuracy and convergence time with the network size and complexity, should be verified.

The average-based mismatch optimisation technique, applied to the continuous-time analogue circuits, improved the computational accuracy while preserving the processing speed but at the cost of increased power consumption and silicon area occupation. Despite its practical disadvantages, it should further be investigated, especially in terms of the convergence and scalability. It is important to check if averaging will still work for larger networks, and, if the number of network copies required to achieve a particular level of precision will depend on the network size. Since the idea of result averaging is not related to a particular implementation or technology, the obtained results may become beneficial for future designs in technologies other than CMOS, possibly offering lower fabrication cost and higher integration level, where design redundancy may become feasible.

The use of cellular architectures for factor graph representations, where nodes maintain data traffic independently and perform operations in parallel, is a very efficient solution. Nevertheless, the possibilities for fast and more area efficient computation, should also be verified in terms of the time-multiplexed processing, applicable to the discrete-time analogue and digital realisations. In such approach, rather than using area expensive parallelisation, a generic reconfigurable state machines for factor and variable nodes could be used as an accelerator, aiding inference in a generic digital or analogue computer.

In general, the software or hardware solutions for Bayesian inference should be devised individually, depending on the application. In robotics, most probably power efficient analogue solutions, providing moderate accuracy, will be of main interest. However, in forecasting or bioinformatics, very large networks and precise computation may be needed. In such cases, the use of the optimised software realisations and dedicated digital hardware realisations of Bayesian inference, employing HPC clusters, FPGA or ASIC chips may be necessary.

Apart from the research presented in this thesis, some exploratory work has been undertaken in the areas of approximate inference in networks using stochastic signals and in hardware-accelerated structure learning from statistical data.

### **9.2.3 Noise based information processing**

Inference in networks with continuous variables is usually done in two ways. The first method assumes discretization of the probability density functions and the use of methods typical for discrete variables. Such approach is very efficient and frequently

used in software applications. The second approach is based on processing probability distributions in their analytical forms, which usually requires lengthy calculations of multiple integrals. In the approach proposed for future consideration, rather than processing probability distributions, it is suggested to process directly random noise signals representing these distributions. Noise processing techniques can be adapted from the algebra of random variables, where it is known that addition of two or more uncorrelated random variables is equivalent to the convolution of their probability density functions. Products of the probability density functions can be calculated using coincidence detectors, used in spiking neural networks. Based on these two mechanisms, the inference in Gaussian Bayesian network could be implemented. Aspects such as signal generation, randomisation, convergence and scalability of such networks, and analogies to spiking neural networks should further be investigated.

#### **9.2.4 Hardware-accelerated network learning**

There are a number of different algorithms for learning Bayesian networks, which have been developed and refined to provide efficient and robust tools for networks discovery. Irrespective of the learning approach, it has been observed that the majority of the algorithm's runtime is largely dominated by statistical tests on the data bases. Such operations require counting the number of records in the data base which fit to a particular pattern. Such operations could be solved by employing associative processing and by using content addressable memory (CAM), with additional hardware counting the number of matched cases. Distributed data processing, performed directly on memory arrays, significantly reduces the data traffic in database operations, and hence, reduces power and increases the processing speed of the implemented system. In particular, counting methods and realisations of fast CAM circuits in CMOS technologies, dedicated for Bayesian network learning, is an important and promising subject of future research.

---

## References

---

- [Abel 93] C. J. Abel, C. Michael, M. Ismail, C. S. Teng, R. Lahri, "Characterisation of Transistor Mismatch for Statistical CAD of Submicron CMOS Analog Circuits", IEEE International Symposium on Circuits and Systems (ISCAS) Vol. 2, pp. 1401 - 1404, May 1993.
- [Acid 2004] S. Acid, L. M. de Campos, J. M. Fernandez-Luna, S. Rodriguez, J. M. Rodriguez, J. L. Salcedo, "A comparison of learning algorithms for Bayesian networks: a case study based on data from emergency medical service", Artificial Intelligence in Medicine, Vol. 30, No. 3, pp. 215-232, Mar. 2004.
- [Allen 2002] P. E. Allen, D. R. Holberg, "CMOS Analogue Circuit Design Second Edition", Oxford University Press, 2002.
- [Andreou 96] A. G. Andreou, K. A. Boahen, "Translinear Circuits in Subthreshold MOS", Analog Integrated Circuits and Signal Processing, Vol. 9, No. 2, pp. 141 - 166, Mar. 1996.
- [Arbel 1964] A. F. Arbel, "Current operated nucleonic modules", Nuclear Instruments and Methods, vol. 32, pp. 341-346, Sep. 1964.
- [Arora 2010] S. Arora, L. Malik, D. Bhattacharjee, M. Nasipuri, "A novel approach to handwritten Devnagari character recognition", Computer Vision and Pattern Recognition, arXiv:1006.5924, Jun. 2010.
- [Astrom 93] A. Astrom, R. Forchheimer, P. Ingelhart, "An Integrated Sensor/Processor Architectures Based on Near-Sensor Image Processing" Proceedings of Computer Architectures for Machine Perception, pp. 147-154, Dec. 1993.

- [Astrom 96] A. Astrom, R. Forchheimer, J. E. Eklund, "Global Feature Extraction Operations for Near-Sensor Image Processing" *IEEE Transactions on Image Processing*, Vol. 5, No. 1, pp.102 - 110, Jan. 1996.
- [Backus 78] J. Backus, "Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs", *Communications of the ACM*, Vol. 21, No. 8, pp. 613 - 641, 1978.
- [Bastos 95] J. Bastos, M. Steyaert, R. Roovers, P. Kinget, W. Sansen, B. Graindourze, A. Pergoot, Er. Janssens, "Mismatch characterisation of small size MOS transistors", *Proceeding of the International Conference on Microelectronic Test Structures (ICMTS)*, pp. 271 - 276, Mar. 1995.
- [Bastos 96] J. Bastos, M. Steyaert, B. Graindourze, W. Sansen, "Matching of MOS Transistors with Different Layout Styles", *Proceedings of the IEEE International Conference on Microelectronic Test Structures (ICMTS 1996)*, pp. 17 - 18, Mar. 1996.
- [Bastos 97a] J. Bastos, M. S. J. Steyaert, A. Pergoot, W. M. Sansen, "Mismatch Characterization of Submicron MOS Transistors", *Journal of Analog Integrated Circuits and Signal Processing*, Vol. 12, pp. 95 - 106, 1997.
- [Bastos 97b] J. Bastos, M. S. J. Steyaert, A. Pergoot, W. M. Sansen, "Influence of Die Attachment on MOS Transistor Matching", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 10, No. 2, pp. 209 - 218, May 1997.
- [Beinlich 89] I. A. Beinlich, H. J. Suermondt, R. Martin Chavez, G. F. Cooper, "The ALARM Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks", *Lecture Notes in Medical Informatics*, Vol. 38, pp. 247 - 256, 1989.
- [Belbachir 2010] A. N. Belbachir, "Smart Cameras", Springer, 2010.
- [Bergman 1999] N. Bergman, "Recursive Bayesian Estimation Navigation and Tracking Applications", Department of Electrical Engineering, Linkoping University, Dissertation No. 579, 1999.
- [Bernard 93] T. M. Bernard, B. Y. Zavidovique, F. J. Devos, "A Programmable Artificial Retina", *IEEE Journal of Solid-State Circuits*, Vol. 28, No. 7, pp. 789 - 798, Jul. 1993.
- [Bhattacharyya 2009] A. B. Bhattacharyya, "Compact MOSFET Models for VLSI Design", John Wiley & Sons, 2009.
- [Bissell 2004] C. C. Bissell, "A great disappearing act: the electronic analogue computer", *IEEE Conference on the History of Electronics*, Bletchley, UK, Jun. 2004.
- [Blum 67] H. Blum, "A transformation for extracting new descriptors of shape", *Symposium on Models for Perception of Speech and Visual Form* (W. Wathen-Dunn, Ed.), pp. 362-380, MIT Press, Cambridge, Massachusetts, 1967.
- [Bolt 96] E. Bolt, E. Cantatore, M. Socha, C. Aussems, J. Solo, "Matching Properties of MOS Transistors and Delay Line Chains with Self-Aligned Source/Drain Contacts",

IEEE International Conference on Microelectronic Test Structures (ICMTS), pp. 21 - 25, Mar. 1996.

[Borgefors 86] G. Borgefors, "Distance Transformations in Digital Images", *Computer Vision, Graphics and Image Processing* 34, pp. 344 - 371, Feb. 1986.

[Bowden 53] B. V. Bowden, "Faster than thought", *A Symposium on Digital Computing Machines*, London, 1953.

[Brodarc 82] D. Brodarc, D. Herbst, B. J. Hosticka, B. Hoefflinger, "Novel Sampled-data MOS multiplier", *Electronics Letters*, Vol. 18, No. 5, pp. 229 - 230, Mar. 1982.

[Bult 86] K. Bult, H. Wallinga, "A CMOS Four-Quadrant Analog Multiplier", *IEEE Journal of Solid-State Circuits*, Vol. SC-21, No. 3, Jun. 1986.

[Bult 87] K. Bult, H. Wallinga, "A Class of Analog CMOS Circuits Based on the Square-Law Characteristics of MOS Transistor in Saturation", *IEEE Journal of Solid-State Circuits*, Vol. SC-22, No. 3, Jun. 1987.

[Cantatore 97] E. Cantatore, et al., "Statistical analysis and optimisation of delay line chains for pixel readout electronics", *Nuclear Instruments and Methods in Physics Research A* 395, pp. 318 - 323, 1997.

[Carey 2013] S. J. Carey, D. R. W. Barr, A. Lopich and P. Dudek, "A 100,000 fps Vision Sensor with Embedded 535 GOPS/W 256x256 SIMD Processor Array", *VLSI Circuits Symposium 2013*, Kyoto, June 2013.

[Carmona-Galan 2003] R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, S. Espejo, T. Roska, C. Rekeczky, I. Petras, A. Rodriguez-Vazquez, "A Bio-Inspired Two-Layer Mixed-Signal Flexible Programmable Chip for Early Vision", *IEEE Transactions on Neural Networks*, Vol. 14, No. 5, pp. 1313 - 1336, Sep. 2003.

[Chang 96a] R. C. Chang, B. J. Sheu, J. Choi, D. C. H. Chen, "Programmable-Weight Building Blocks for Analog VLSI Neural Network Processors", *Analog Integrated Circuits and Signal Processing*, vol. 9, pp. 215 - 230, Kulwer Academic Publishers, 1996.

[Chang 96b] S. T. Chang, B. R. Hayes-Gill, C. J. Paull, "Multi-Function Block for Switched Current Field Programmable Analogue Array", *IEEE 39th Midwest Symposium on Circuits and Systems*, Vol. 1, pp. 158 - 161, Aug. 1996.

[Chen 2006] H. S. Chen, H. T. Chen, Y. W. Chen, S. Y. Lee, "Human action recognition using star skeleton", *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pp. 171 - 178, 2006.

[Chow 68] C. K. Chow, C. N. Liu, "Approximating Discrete Probability Distributions with Dependence Trees", *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462-467, May 1968.

[Christiansen 95] J. Christiansen, "An Integrated CMOS 0.15 ns Digital Timing Generator for TDC's and Clock Distribution Systems", *IEEE Transactions on Nuclear Science*, Vol. 42, No. 4, pp. 753 - 757, Aug. 1995.

- [Chua 88a] Chua. L., Yang L., "Cellular Neural Networks: Applications", IEEE Transactions on Circuits and Systems, vol. 35, no. 10, Oct. 1988.
- [Chua 88b] L. Chua, L. Yang, "Cellular Neural Networks: Theory", IEEE Transactions on Circuits and Systems, vol. 35, no. 10, Oct. 1988.
- [Coban 95] A. L. Coban, P. E. Allen, X. Shi, "Low-Voltage, Analog IC Design in CMOS Technology", IEEE Transactions on Circuits and Systems I, Vol. 42, No. 11, pp. 955 - 958, Nov. 1995.
- [Cooper 90] G. F. Cooper, "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks", Artificial Intelligence, No. 42, pp. 393 - 405, 1990.
- [Corneil 2012] D. Corneil, D. Sonnleithner, E. Neftci, E. Chicca, M. Cook, G. Indiveri, R. Douglas, "Real-time inference in a VLSI spiking neural network", IEEE International Symposium on Circuits and Systems (ISCAS) pp. 2425 - 2428, May 2012.
- [Cox 85] P. Cox, P. Yang, S. S. Mahant-Shetti, P. Chatterjee, "Statistical Modeling for Efficient Parametric Yield Estimation of MOS VLSI Circuits", IEEE Journal of Solid-State Circuits, Vol. SC-20, No. 1, pp. 391 - 398, Feb. 1985.
- [Danielsson 80] P. E. Danielsson, "Euclidean Distance Mapping", Computer Graphics and Image Processing 14, pp. 227 - 248, Feb. 1980.
- [Darwiche 2009] A. Darwiche, "Modeling and reasoning with Bayesian networks", Cambridge University Press, 2009.
- [Daubert 88] S. J. Daubert, D. Vallancourt, Y. P. Tsvividis, "Current copier cells", Electronics Letters, Vol. 24, No. 25, Dec. 1988.
- [Davies 90] E. R. Davies, "Machine Vision: Theory , Algorithms, Practicalities", Cambridge University Press, 1990.
- [Dominguez-Castro 98] R. Dominguez-Castro, A. Rodriguez-Vazquez, S. Espejo, R. Carmona, "Four-Quadrant One-Transistor-Synapse for High-Density CNN Implementations", IEEE Proceedings of Fifth International Workshop on Cellular Neural Networks and Their Applications, pp. 243 - 248, Apr. 1998.
- [Drennan 2003] P. G. Drennan, C. C. McAndrew, "Understanding MOSFET Mismatch for Analog Design", IEEE Journal of Solid-State Circuits, Vol. 38, No. 3, pp. 450 - 456, Mar. 2003.
- [Dudek 2000a] P. Dudek, P. J. Hicks, "A CMOS General-Purpose Sampled-Data Analog Processing Element", IEEE Transactions on Circuits and Systems II, Vol. 47, No. 5, pp. 467 - 472, May 2000.
- [Dudek 2000b] P. Dudek, S. Szczepanski, J. V. Hatfield, "A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line", IEEE J. Solid-State Circuits, Vol. 35, No. 2, pp. 240-247, Feb. 2000.
- [Dudek 2003] P. Dudek and V. D. Juncu, "Compact Discrete-Time Chaos Generator Circuit", Electronics Letters, Vol.39, No. 20, pp. 1431-1432, Oct. 2003.

- [Dudek 2005] P. Dudek, "A General-Purpose Processor-per-Pixel Analog SIMD Vision Chip", IEEE Transactions on Circuits and Systems I, Vol. 52, No. 1, pp. 13 - 20, Jan. 2005.
- [Dudek 2006] P. Dudek, "An Asynchronous Cellular Logic Network for Trigger-Wave Image Processing on Fine-Grain Massively Parallel Arrays", IEEE Transactions on Circuits and Systems II, vol. 53, no.5, pp. 354-358, May 2006.
- [Eklund 96] J. E. Eklund, C. Svensson, A. Astrom, "VLSI Implementation of a Focal Plane Image Processor-A Realization of the Near-Sensor Image Processing Concept", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 4, No. 3, pp. 322 - 335, Sep. 1996.
- [Enomoto 85] T. Enomoto, M. Yasumoto, "Integrated MOS Four-Quadrant Analog Multiplier Using Switched Capacitor Technology for Analog Signal Processor IC's", IEEE Journal of Solid-State Circuits, Vol. SC-20, No. 4, pp. 852 - 859, Aug. 1985.
- [FDK 2009] Foundry Design Kit, NDA confidential.
- [Felzenszwalb 2004] P. F. Felzenszwalb, D. P. Huttenlocher, "Efficient Belief Propagation for Early Vision", IEEE Computer Society Conference of Computer Vision and Pattern Recognition (CVPR), Vol. 1, pp. 261 - 268, Jul. 2004.
- [Fernandez-Berni 2011] J. Fernandez-Berni, R. Carmona-Galan, L. Carranza-Gonzalez, "FLIP-Q: A QCIF Resolution Focal-Plane Array for Low-Power Image Processing", IEEE Journal of Solid State-Circuits, Vol. 46, No. 3, pp. 669 - 680, Mar. 2011.
- [Fiez 91] T. S. Fiez, G. Liang, D. J. Allstot, "Switched-Current Circuit Design", IEEE Journal of Solid-State Circuits, Vol. 26, No. 3, Mar. 1991.
- [Friedman 2004] N. Friedman, M. Linial, I. Nachman, D. Pe'er, "Using Bayesian Networks to Analyze Expression Data", Journal of Computational Biology, Vol. 7, No. 3-4, pp. 601 - 620, Jul. 2004.
- [Frost 96] J. R. Frost, "The Theory of Search", Soza & Company, Ltd., of Airfax Virginia, U.S.A, in cooperation with the U.S. Coast Guard, Oct. 1996.
- [Gaines 67] B. R. Gaines, "Stochastic computing", Proceedings of the Spring Joint Computer Conference, pp. 149 - 156, Apr. 1967.
- [Gea-Banacloche 2005] J. Gea-Banacloche, L. B. Kish, "Future Directions in Electronic Computing and Information Processing", Proceedings of IEEE, Vol. 93, No. 10, pp. 1858-1863, Oct. 2005.
- [Geman 85] S. Geman, D. E. McClure, "Bayesian Image Analysis: An Application to Single Photon Emission Tomography", Proceedings of the Statistical Computing Section, pp. 12 - 18, 1985.
- [Gilbert 68] B. Gilbert, "A Precise Four-Quadrant Multiplier with Subnanosecond Response", IEEE Journal of Solid-State Circuits, Vol. SC-3, No. 4, pp. 365 - 373, Dec. 1968.

- [Gilbert 75] B. Gilbert, "Translinear circuits: a proposed classification", *Electronics Letters*, Vol. 11, No. 1, pp. 14 - 16, Jan. 1975.
- [Gilbert 84] B. Gilbert, "A Monolithic 16-Channel Analog Array Normalizer", *IEEE Journal of Solid-State Circuits*, Vol. SC-19, No. 6, Dec. 1984.
- [Gravati 2005a] M. Gravati, M. Valle, "Modeling mismatch effects in CMOS translineat loops and current mode multipliers", *Proceedings of the European Conference on Circuit Theory and Design (ECCTD)*, Vol. 3, pp. 373 - 376, Sep. 2005.
- [Gravati 2005b] M. Gravati, M. Valle, G. Ferri, N. Guerrini, L. Reyes, "A Novel Current-Mode Very Low Power Analog CMOS Four Quadrant Multiplier", *Proceedings of the 31st European Solid-State Circuits Conference (ESSCIRC)*, pp. 495 - 498, Sep. 2005.
- [Grundy 94] D. L. Grundy, "A Computational Approach to VLSI Analog Design", *Journal of VLSI Signal Processing*, Vol. 8, No. 1, pp. 53 - 60, 1994.
- [Guggenbuhl 94] W. Guggenbuhl, J. Di, J. Goette, "Switched-Current Memory Circuits for High-Precision Applications", *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 9, pp. 1108-1116, Sep. 1994.
- [Hagenauer 98] J. Hagenauer, "Decoding of binary codes with analogue networks", *Proceeding of Information Theory Workshop*, pp. 13 - 14, Feb. 1998.
- [Hagenauer 2002] J. Hagenauer, M. Moerz, A. Schaefer, "Analog Decoders and Receivers for High Speed Applications", *Proceedings of International Zurich Seminar on Broadband Communication*, 2002.
- [Halonen 90] K. Halonen, V. Porra, T. Roska, L. Chua, "VLSI Implementation of a Reconfigurable Cellular Neural Network Containing Local Logic (CNNL)", *IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA-90*, pp. 206 - 215, Dec. 1990.
- [Han 98] G. Han, E. Sanchez-Sinencio, "CMOS Transconductance Multipliers: A Tutorial", *IEEE Transactions on Circuits and Systems II*, Vol. 45, No. 12, pp. 1550 - 1563, Dec. 1998.
- [Haralick 92] R. M. Haralick, L. G. Shapiro, "Computer and Robot Vision", Vol. 1, Addison-Wesley, 1992.
- [Harrer 92] H. Harrer, J. A. Nossek, R. Stelzl, "An Analog Implementation of Discrete-Time Cellular Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 3, No. 3, pp. 466 - 476, May 1992.
- [Hsiao 2012] W. H. Hsiao, Y. T. He, M P. H. Lin, R. G. Chang, S. Y. Lee, "Automatic Common-Centroid Layout Generation for Binary-Weighted Capacitors in Charge-Scaling DAC". *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 173 - 176, Sep. 2012.

- [He 2007] J. He, X. Xi, H. Wan, M. Dunga, M. Chan, A. M. Niknejad, "BSIM5: An advanced charge-based MOSFET model for nanoscale VLSI circuit simulation", *Journal of Solid-State Electronics*, Vol. 51, pp. 433 - 444, 2007.
- [Hecht 90] R. Hecht-Nielsen, "Neurocomputing", Addison Wesley, 1990.
- [Heijne 96] E. H. M. Heijne, et al., "LHC1: A semiconductor pixel detector readout chip with internal, tunable delay providing a binary pattern of selected events", *Nuclear Instruments and Methods in Physics Research A* 383, pp. 55 - 63, 1996.
- [Hopfield 84] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 81, pp. 3088 - 3092, May 1984.
- [Huang 2010] Z. Huang, A. Kurokawa, M. Hashimoto, T. Sato, M. Jiang, Y. Inoue, "Modeling the Overshooting Effect for CMOS Inverter Delay Analysis in Nanometer Technologies", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Vol. 29, No. 2, pp. 250 - 260, Feb. 2010.
- [Hughes 93] J. B. Hughes, K.W. Moulding, "S<sup>2</sup>I: A switched-current technique for high performance", *Electronics Letters*, Vol. 29, No. 16, Aug. 1993.
- [Hung 2002] Y. C. Hung, B. D. Liu, "An analog CMOS rank-order extractor with O(N) complexity using maximum/winner-take-all circuit", *Asian-Pacific Conference on Circuit and Systems (APCCAS)*, Vol. 2, pp. 389 - 394, 2002.
- [Indiveri 2006] G. Indiveri, E. Chicca, R. Douglas, "A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity", *IEEE Transactions on Neural Networks*, Vol. 17, No. 1, pp. 211-221, Jan. 2006.
- [Intel 86] "4004 Single Chip 4-Bit P-Channel Microprocessor", Intel Corporation, Mar. 1987.
- [Intel 2014] [http://ark.intel.com/products/37150/Intel-Core-i7-950-Processor-8M-Cache-3\\_06-GHz-4\\_80-GTs-Intel-QPI](http://ark.intel.com/products/37150/Intel-Core-i7-950-Processor-8M-Cache-3_06-GHz-4_80-GTs-Intel-QPI)
- [Ishikawa 99] M. Ishikawa, K Ogawa, T. Komuro, I. Ishii, "A CMOS Vision Chip with SIMD Processing Element Array for 1ms Image Processing", *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 206 - 207, Feb. 1999.
- [Jensen 2007] F. V. Jensen, T. D. Nielsen, "Bayesian Networks and Decision Graphs Second Edition", *Information Science & Statistics*, Springer, 2007.
- [Joetten 85] R. Joetten, T. Weiss, J. Wolters, H. Ring, B. Bjoernsson, "A New Real-Time Simulator for Power System Studies", *IEEE Power Engineering Review*, vol PER-5, no. 9, pp. 58 - 59, Sep. 1985.
- [Jovanovic 2006] G. S. Jovanovic, M. K. Stojcev, "Current Starved Delay Element with Symmetric Load", *International Journal of Electronics*, Vol. 93, No. 3, pp. 167-175, Mar. 2006.

- [Khachab 89] N. I. Khachab, M. Ismail, "MOS multiplier/divider cell for analogue VLSI", *Electronics Letters*, Vol. 25, No. 23, pp. 1550 -1552, Nov. 1989.
- [Kinget 96] P. Kinget, M. Steyaert, "Impact of transistor mismatch on the speed-accuracy-power trade-off of analog CMOS circuits", *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 333 - 336, May 1996.
- [Kinget 97] P. Kinget, M. Steyaert, "Analog VLSI Integration of massive parallel processing systems", Kluwer Academic Publishers, 1997.
- [Kinget 2005] P. Kinget, "Device Mismatch and Tradeoffs in the Design of Analog Circuits", *IEEE Journal of Solid-State Circuits*, Vol. 40, No. 6, pp. 1212 - 1224, Jun. 2005.
- [Kim 87] J. H. Kim, J. Pearl, "CONVINCE: A Conversational Inference Consolidation Engine", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-17, No. 2, pp. 120 - 132, Apr. 1987.
- [Kish 2002] L. B. Kish, "End of Moore's law: thermal (noise) death of integration in micro and nano electronics", *Physics Letters A*, 305, pp. 144-149, 2002.
- [Kish 2011] L. B. Kish, "Noise-based Information Processing", *21st International Conference on Noise and Fluctuations (ICNF)*, pp. 28-33, Jun. 2011.
- [Koeser 2004] K. Koeser, C. Perwass, G Sommer, "Dense Optic Flow with a Bayesian Occlusion Model", *SCVMA, Lecture Notes in Computer Science*, Vol. 3667, pp. 127 - 139, 2004.
- [Komuro 2003] T. Komuro, I. Ishii, M. Ishikawa, A. Yoshida, "A Digital Vision Chip Specialised for High-Speed Target Tracking", *IEEE Transactions on Electron Devices*, Vol. 50, No. 1, pp. 191 - 199, Jan. 2003.
- [Komuro 2009] T. Komuro, A. Iwashita, M. Isikawa, "A QVGA-Size Pixel-Parallel Image Processor for 1,000-FPS Vision", *IEEE Computer Society*, pp. 58 - 67, Dec. 2009.
- [Korkmaz 2008] P. Korkmaz, B. E. S. Akgul, K. V. Palem, "Energy, Performance, and Probability Tradeoffs for Energy-Efficient Probabilistic CMOS Circuits", *IEEE Transactions on Circuits and Systems I*, Vol. 55, No. 8, pp. 2249-2262, Sep. 2008.
- [Krinsky 91] V. I. Krinsky, V. N. Biktashev, I. R. Efimov, "Autowaves Principles for Parallel Image Processing", *Physica D*, vol. 49, pp. 247-253, 1991.
- [Kschischang 2001] F. R. Kschischang, B. J. Frey, H. A. Loeliger, "Factor Graphs and the Sum-Product Algorithm", *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 498 - 519, Feb. 2001.
- [Kub 90] F. J. Kub, K. K. Moon, I. A. Mack, F. M. Long, "Programmable Analog Vector-Matrix Multipliers", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 1, pp. 207 - 214, Feb. 1990.
- [Kuhnert 89] L. Kuhnert, K. I. Agladze, V. I. Krinsky, "Image processing using light-sensitive chemical waves", *Letters to Nature, Nature* vol. 337, pp. 244-247, 1989.

- [Kulesza 2006] Z. Kulesza, W. Tylman, "Implementation of Bayesian Network in FPGA Circuit", Proceedings of the International Mixed Design of Integrated Circuits and Systems (MIXDES), pp. 711 - 715, Jun. 2006.
- [Lakshmikumar 86] K. R. Lakshmikumar, R. A. Hadaway, M. A. Copeland, "Characterisation and Modeling of Mismatch in MOS Transistors for Precision Analog Design", IEEE Journal of Solid-State Circuits, Vol. SC-21, no. 6, pp. 1057 - 1066, Dec. 1986.
- [Lam 92] L. Lam, S. W. Lee, C. Y. Suen, "Thinning Methodologies - A Comprehensive Survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 9, pp. 869 - 885, Sep. 1992.
- [Lazkano 2006] E. Lazkano, B. Sierra, A. Astigarraga, J. M. Martinez-Otzeta, "On the use of Bayesian Networks to develop behaviours for mobile robots", Robotics and Autonomous Systems, Vol. 55, pp. 253 - 265, 2007.
- [Lebeltel 2004] O. Lebeltel, P. Bessiere, J. Diard, E. Mazer, "Bayesian Robot Programming", Autonomous Robots, Vol. 16, No. 1, pp. 49 - 79, Kluwer Academics Publisher, Jan. 2004.
- [Lee 95] S. T. Lee, K. T. Lau, L. Siek, "Four-quadrant CMOS analogue multiplier for artificial neural network", Electronics Letters, Vol. 31, No. 1, pp. 48 - 49, Jan. 1995.
- [Lee 2009] S. H. Lee, I. H. Suh, "Bayesian Network-based Behavior Control for Skilligent Robots", IEEE International Conference on Robotics and Automation, pp. 2910 - 2916, May 2009.
- [Leenaerts 94] D. M. W. Leenaerts, A. J. Leeuwenburgh, G. G. Cell, "A High-Performance SI Memory Cell", IEEE Journal of Solid-State Circuits, Vol. 29, No. 11, Nov. 1994.
- [Leenaerts 96] D. M. W. Leenaerts, G. H. M. Joordens, J. A. Hegt, "A 3.3 V 625 kHz Switched-Current Multiplier", IEEE Journal of Solid-State Circuits, Vol. 31, No. 9, Sep. 1996.
- [Liang 2011] K. Liang, C. C. Cheng, Y. C. Lai, L. G. Chen, "Hardware-Efficient Belief Propagation", IEEE Transactions on Circuits and Systems, Vol. 21, No. 5, pp. 525 - 537, 2011.
- [Lin 2010] M. Lin, I. Lebedev, J. Wawrzynek, "High-throughput bayesian computing machine with reconfigurable hardware", Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays (FPGA), pp. 73 - 82, 2010.
- [Linares-Barranco 2007] B. Linares-Barranco, T. Serrano-Gotarredona, "A Physical Interpretation of the Distance Term in Pelgrom's Mismatch Model results in very Efficient CAD", IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1561 - 1564, May 2007.

- [Liu 95] S. I. Liu, C. C. Chang, "CMOS subthreshold four-quadrant multiplier based on unbalanced source-coupled pairs", *International Journal of Electronics*, Vol. 78, No. 2, pp. 327 - 332, Feb. 1995.
- [Loeliger 99] H. A. Loeliger, F. Tarkoy, F. Lustenberger, M. Helfenstein, "Decoding in Analog VLSI", *IEEE Communication Magazine*, Vol. 37, No. 4, pp. 99 - 101, Apr. 1999.
- [Loeliger 2001] H. A. Loeliger, F. Lustenberger, M. Helfenstein, F. Tarkoy, "Probability Propagation and Decoding in Analog VLSI", *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 837 - 843, Feb. 2001.
- [Loeliger 2004] H. A. Loeliger, "An Introduction to Factor Graphs", *IEEE Signal Processing Magazine*, Vol. 21, No. 1, pp. 28 - 41, Jan. 2004.
- [Lopich 2009] A. Lopich, P. Dudek, "Hardware Implementation of Skeletonization Algorithm for Parallel Asynchronous Image Processing", *Journal of Signal Processing Systems*, Vol. 56, No. 1, pp. 91-103, Jul. 2009.
- [Lopich 2010a] A. Lopich, P. Dudek, "An 80×80 general-purpose digital vision chip in 0.18  $\mu\text{m}$  CMOS technology", *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp 4257-4260, May 2010.
- [Lopich 2010b] A. Lopich, P. Dudek, "Asynchronous Cellular Logic Network as a Co-Processor for a General-Purpose Massively Parallel Array", *International Journal of Circuit Theory and Applications*, DOI: 10.1002/cta.679, Apr. 2010.
- [Lopich 2011] A. Lopich and P. Dudek, "A SIMD Cellular Processor Array Vision Chip With Asynchronous Processing Capabilities", *IEEE Transactions on Circuits and Systems - I*, vol 58, issue 10, pp. 2420-2431, October 2011.
- [Lovett 98] S. J. Lovett, M. Welten, A. Mathewson, B. Mason, "Optimizing MOS Transistor Mismatch", *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 1, pp. 147 - 150, Jan. 1998.
- [Lovett 2000] S. J. Lovett, G. A. Gibbs, A. Pancholy, "Yield and Matching Implications for Static RAM Memory Array Sense-Amplifier Design", *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 8, pp. 1200 - 1204, Aug. 2000.
- [Luckenbill 2002] S. B. Luckenbill, "Building Bayesian Networks with Analog Subthreshold CMOS Circuits", Yale University, 2002.
- [Lustenberger 99a] F. Lustenberger, M. Helfenstein, G. S. Moschytz, H. A. Loeliger, F. Tarkoy, "All- Analog Decoder for Binary (18, 9, 5) Tail-Biting Trellis Code", *Proceedings of the European Solid-State Circuits Conference*, pp. 362 - 365, Sep. 1999.
- [Lustenberger 99b] F. Lustenberger, M. Helfenstein, H. A. Loeliger, F. Tarkoy, G. S. Moschytz, "An analog VLSI decoding technique for digital codes", *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 2, pp. 424 - 427, Jun. 1999.
- [Lustenberger 2001] F. Lustenberger, H. A. Loeliger, "On Mismatch Errors in Analog-VLSI Error Correcting Decoders", *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 4, pp. 198 - 201, 2001.

- [Maass 2001] W. Maass, C. M. Bishop, "Pulsed Neural Networks", Massachusetts Institute of Technology, 2001.
- [Manganaro 98] G. Manganaro, J. P. de Gyvez, "A Four-Quadrant  $S^2I$  Switched-Current Multiplier", IEEE Transactions on Circuits and Systems II, Vol. 45, No. 7, Jul. 1998.
- [Manzanera 2002] A. Manzanera, "Morphological Segmentation on the Programmable Retina: Towards Mixed Synchronous/Asynchronous Algorithms", 6<sup>th</sup> International Symposium on Mathematical Morphology, pp. 389-399, 2002.
- [Massengill 91] L. W. Massengill, "A Dynamic CMOS Multiplier for Analog VLSI Based on Exponential Pulse-Decay Modulation", IEEE Journal of Solid-State Circuits, Vol. 26, No. 3, pp. 268 - 276, Mar. 1991.
- [McCulloch 43] W. D. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity", The Bulletin, of Mathematical Biology, Vol. 5, No. 5, pp. 115 - 133, Dec. 1943.
- [Mead 89] C. Mead, "Analog VLSI and Neural Systems", Addison-Wesley Publishing Company, 1989.
- [Mead, 94] C. A. Mead, "Scaling of MOS Technology to Submicrometer Feature Size", Journal of VLSI Signal Processing, Vol. 8, pp. 9 - 25, 1994.
- [Mehrvarz 95] H. R. Mehrvarz, C. Y. Kwok, "A Large-Input-Dynamic-Range Multi-Input Floating-Gate MOS Four-Quadrant Analog Multiplier", IEEE International Solid-State Circuits Conference (ISSCC), pp. 60 - 61, Feb. 1995.
- [Michael 92] C. Michael, M. Ismail, "Statistical Modeling of Device Mismatch for Analog MOS Integrated Circuits", IEEE Journal of Solid-State Circuits, Vol. 27, No. 2, pp. 154 - 166, Feb. 1992.
- [Michael 96] C. Michael, H. Su, M. Ismail, A. Kankunen, M. Valtonen, "Statistical Techniques for the Computer-Aided Optimization of Analog Integrated Circuits", IEEE Transactions on Circuits and Systems I, Vol. 43. No. 5, pp. 410 - 413, May 1996.
- [Moerz 2000] M. Moerz, J. Hagenauer, E. Offer, "On the Analog Implementation of the APP (BCJR) Algorithm", IEEE International Symposium on Information Theory, pp.425, 2000.
- [Moini 97] A. Moini, "Vision Chips or Seeing Silicon", The Centre for High Performance Integrated Technologies and Systems Department of Electrical & Electronics Engineering The Univ. of Adelaide, SA 5005, Australia, 1997.
- [Montanari 68] U. Montanari, "A Method for Obtaining Skeletons Using a Quasi-Euclidean Distance", Journal of the Association for Computing Machinery, Vol. 15, No. 4, pp. 600 - 624, Oct. 1968.
- [Moore 65] G. E. Moore, "Cramming more components onto integrated circuits", Electronics, Vol. 38, No. 8, pp 114 ff., Apr. 1965.

- [Neapolitan 2004] R. E. Neapolitan, "Learning Bayesian Networks", Pearson Prentice Hall, 2004.
- [von Neumann 45] J. von Neumann, "First Draft of a Report on the EDVAC", Contract No. W-670-ORD-4926, between the United States Army Ordnance Department and the University of Pennsylvania, Moore School of Electrical Engineering, Jun. 1945.
- [von Neumann 52] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," lectures delivered at the California Institute of Technology, January 4-15, 1952. Notes by R. S. Pierce, Caltech Eng. Library, QA.267.V6.
- [Nicollian 82] E. H. Nicollian, J. R. Brews, "MOS Physics and Technology", New York: Wiley, 1982.
- [Nikolova 2011] O. Nikolova, S. Aluru, "Parallel Discovery of Direct Casual Relations and Markov Boundaries with Applications to Gene Networks" International Conference on Parallel Processing (ICPP), pp. 512-521, Sep. 2011.
- [Norsys 2014] Resources of Norsys Software Corp. (networks library) available at: <https://www.norsys.com/netlibrary/index.htm>
- [Olesen 89] K. G. Olesen, U. Kjaerulff, F. Jensen, F. V. Jensen, B. Falck, S. Andreassen, S. K. Andersen, "A MUNIN network for the median nerve - a case study on loops", Journal of Applied Artificial Intelligence, Vol. 3, No. 2-3, pp. 385 - 403, 1989.
- [Paasio 2002] A. Paasio, M. Laiho, A. Kananen, K. Halonen, "An Analog Array Processor Hardware Realization with Multiple New Features", Proceedings of the International Joint Conference on Neural Network (IJCNN), Vol. 2, pp. 1952 - 1955, May 2002.
- [Paoloni 2010] G. Paoloni, "How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures", Intel White Paper, Sep. 2010.
- [Papoulis 2002] A. Papoulis, "Probability, random variables, and stochastic processes", McGraw-Hill 4th edition, 2002.
- [Pearl 86] J. Pearl, "Fusion, Propagation, and Structuring in Belief Networks", Artificial Intelligence, Vol. 29, pp. 241 - 288, Sep. 1986.
- [Pearl 88] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann Publishers, 1988.
- [Pelgrom 89] M. J. M. Pelgrom, A. C. J. Duinmaijer, A. P. G. Welbers, "Matching Properties of MOS Transistors", IEEE Journal of Solid-State Circuits, Vol. 24, No. 5, pp. 1433 - 1440, Oct. 1989.
- [Pelgrom 98] M. J. M. Pelgrom, H. P. Tuinhout, M. Vertregt, "Transistor matching in analog CMOS applications", IEEE International Electron Devices Meeting (IEDM'98), pp. 915 - 918, Dec. 1998.
- [Pelgrom 2010] M. J. M. Pelgrom, "Analog-to-Digital Conversion", Springer, 2010.

- [Perez-Munuzuri 93] V. Perez-Munuzuri, V. Perez-Villar, L. O. Chua, "Autowaves for Image Processing on Two-Dimensional CNN Array of Excitable Nonlinear Circuits: Flat and Wrinkled Labyrinths", *IEEE Transactions on Circuits and Systems I*, Vol. 40, No. 3, pp. 174 - 181, Mar. 1993.
- [Petriu 96] E. M. Petriu, K. Watanabe, T. H. Yeap, "Applications of Random-Pulse Machine Concept to Neural Network Design", *IEEE Transactions on Instrumentation and Measurement*, Vol. 45, No. 2, pp. 665-669, Apr. 1996.
- [Poikonen 2009] J. Poikonen, M. Laiho, A. Paasio, "MIPA4k: A 64×64 Cell Mixed-mode Image Processor Array", *IEEE International Symposium on Circuits and Systems ISCAS 2009*, May 2009.
- [Poppelbaum 67] W. J. Poppelbaum, C. Afuso, J. W. Esch, "Stochastic computing elements and systems", *Proceedings of the Joint Computer Conference*, pp. 635 - 644, Nov. 1967.
- [Ramirez-Angulo 96] J. Ramirez-Angulo, "±0.75V BiCMOS Four Quadrant Analog Multiplier with Rail-Rail Input Signal-Swing", *IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol.1, pp. 242 - 245, May 1996.
- [Ramirez-Angulo 97] J. Ramirez-Angulo, G. Gonzalez-Altamirano, S. C. Choi, "Modelling Multiple-Input Floating-Gate Transistors for Analog Signal Processing", *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 2020 - 2023, Jun. 1997.
- [Razavi 2001] B. Razavi, "Design of Analogue CMOS Integrated Circuits", McGraw-Hill, International Edition, 2001.
- [Razmjooei 2010] S. Razmjooei, P. Dudek, "Approximating Euclidean Distance Transform with Simple Operations in Cellular Processor Arrays", *IEEE Workshop on Cellular Nanoscale Networks and Applications, CNNA 2010*, Feb. 2010.
- [Rekeczky 99] C. Rekeczky, L. O. Chua, "Computing with Front Propagation: Active Contour and Skeleton Models in Continuous Time CNN", *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, Volume 23, Numbers 2-3, pp. 373-402, 1999.
- [Riberio 67] S. T. Riberio, "Random-Pulse Machines", *IEEE Transactions on Electronic Computers*, Vol. EC-16, No. 3, Jun. 1967.
- [Richardson 72] W. H. Richardson, "Bayesian-Based Iterative Method of Image Restoration", *Journal of the Optical Society of America*, Vol. 62, No. 1, pp. 55 - 59, Jan. 1972.
- [Rodriguez-Vazquez 93] A. Rodriguez-Vazquez, S. Espejo, R. Dominguez-Castro, J. L. Huertas, E. Sanchez-Sinencio, "Current-Mode Techniques for the Implementation of Continuous- and Discrete-Time Cellular Neural Networks", *IEEE Transactions on Circuits and Systems II*, Vol. 40, No. 3, pp.132 - 146, Mar. 1993.
- [Rodriguez-Vazquez 99] A. Rodriguez-Vazquez, E. Roca, M. Delgado-Restituto, S. Espejo, R. Dominguez-Castro, " MOST-Based Design and Scaling of Synaptic

Interconnections in VLSI Analog Array Processing CNN Chips" *Journal of VLSI Signal Processing*, Vol. 23, pp. 239 - 266, 1999.

[Rodriguez-Vazquez 2003] A. Rodriguez-Vazquez, G. Linan, S. Espejo, R. Dominguez-Castro, "Mismatch-Induced Trade-Offs and Scalability of Analog Processing Visual Microprocessor Chips", *Journal of Analog Integrated Circuits and Signal Processing*, Vol. 37, pp. 77-83, 2003.

[Rodriguez-Vazquez 2004] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, S. Espejo Mena, "ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Towards VSoCs", *IEEE Transactions on Circuits and Systems - I: Regular Papers*, vol. 51, no. 5, May 2004.

[Rosenblatt 58] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, Vol. 65, Nov. 1958.

[Roska 93] T. Roska, L. Chua, "The CNN Universal Machine: An Analogic Array Computer", *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 40, no. 3, Mar. 1993.

[Roska 2001] B. Roska, F. Werblin, "Vertical interactions across ten parallel, stacked representations in the mammalian retina", *Nature*, Vol. 410, pp. 583 - 587, Mar. 2001.

[Shakiba 98] M. H. Shakiba, D. A. Johns, K. W. Martin, "BiCMOS Circuits for Analog Viterbi Decoders", *IEEE Transactions on Circuits and Systems II*, Vol. 45, No. 12, pp. 1527 - 1537, Dec. 1998.

[Shannon 38] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits", *Transactions of AIEE*, vol. 57, pp. 713 - 723, May 1938.

[Shannon 40] C. E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits", Master's Thesis, Massachusetts Institute of Technology, 1940.

[Sheu 87] B. J. Sheu, D. L. Scharfetter, P. K. Ko, M. C. Jeng, "BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors", *IEEE Journal of Solid-State Circuits*, Vol. SC-22, No. 4, pp. 558 - 566, Aug. 1987.

[Shichman 68] H. Shichman, D. A. Hodges, "Modeling and Simulation of Insulated-Gate Filed-Effect Transistor Switching Circuits", *IEEE Journal of Solid-State Circuits*, Vol. SC-3, No. 3, pp. 285 - 289, 1968.

[Shoemaker 91] P. A. Shoemaker, G. L. Haviland, R. L. Shimabukuro, I. Lagnado, "A Simple CMOS Analog Four-Quadrant Multiplier", *Analog Integrated Circuits and Signal Processing I*, Kulwer Academic Publishers, Vol. 1, No. 2, pp. 107 - 117, Oct. 1991.

[Shor 94] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring", *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, Nov. 1994.

[Shutter 2005] H. Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software", *C/C++ Users Journal*, vol. 23, Feb. 2005.

[Shyu 84] J. B. Shyu, G. C. Tmes, F. Krummenacher, "Random Error Effects in Matched MOS Capacitors and Current Sources", IEEE Journal of Solid-State Circuits, Vol. SC-19, No. 6, pp. 948 - 955, Dec. 1984.

[Sienkiewicz 2009] P. Sienkiewicz, J. S. Nowak, "Sixty Years of Cybernetics and Polish Informatics", Scientific Magazine no. 3, pp. 9 - 23, WWSI, Warsaw, 2009.

(in original: P. Sienkiewicz, J. S. Nowak, "Sześćdziesiąt lat cybernetyki i polskiej informatyki", Zeszyt Naukowy nr 3, str. 9 - 23, WWSI, Warszawa, 2009).

[Song 90] H. J. Song, C. K. Kim, "An MOS Four-Quadrant Analog Multiplier Using Simple Two-Input Squaring Circuits with Source Followers", IEEE Journal of Solid State Circuits, Vol. 25, No. 3, Jun. 1990.

[Song 93] L. Song, M. I. Elmasry, A. Vennelli, "Analog Neural Network Building Blocks Based on Current Mode Subthreshold Operation", IEEE International Symposium on Circuits and Systems (ISCAS), Vol. 4, pp. 2462 - 2465, May 1993.

[Srinivas 97] M. A. Srinivas, R. J. McEliece, "A General Algorithm for Distributing Information in a Graph", IEEE International Symposium on Source, pp. 6, 1997.

[Steyaert 94] M. Steyaert, J. Bastos, R. Roovers, P. Kinget, W. Sansen, B. Graindourze, A. Pergoot, Er. Janssen, "Threshold voltage mismatch in short-channel MOS transistors", Electronics Letters, Vol. 30, No. 18, pp. 1546 - 1547, Sep. 1994.

[Toifl 99] T. Toifl, R. Vari, P. Moreira, A. Marchioro, "4-Channel Rad-Hard Delay Generation ASIC with 1ns Timing Resolution for LHC", IEEE Transactions on Nuclear Science, Vol. 46, No. 3, pp. 139 - 143, Jun. 99.

[Toumazou 90a] C. Toumazou, N. C. Battersby, C. Maglaras, "High-performance algorithmic switched-current memory cell", Electronics Letters, Vol. 26, No. 19, pp. 1593 - 1595, Sep. 1990.

[Toumazou 90b] C. Toumazou, J. B. Hughes, D. M. Pattullo, "Regulated cascode switched-current memory cell", Electronics Letters, Vol. 26, No. 5, Mar. 1990.

[Toumazou 93a] C. Toumazou, J. B. Hughes, N. C. Battersby, "Switched-Currents an analogue technique for digital technology", IEE Circuits and Systems Series 5, Peter Peregrinus Ltd, 1993.

[Toumazou 93b] C. Toumazou, F. J. Lidgely, D. G. Haigh, "Analogue IC Design: the current-mode approach", IEE Circuits and Systems Series 2, Peter Peregrinus Ltd, 1993.

[Tuinhout 96] H. Tuinhout, M. Pelgrom, R. Penning de Vries, M. Vertregt, "Effects of Metal Coverage on MOSFET Matching", International Electron Devices Meeting (IEDM), pp. 735 - 738, Dec. 1996.

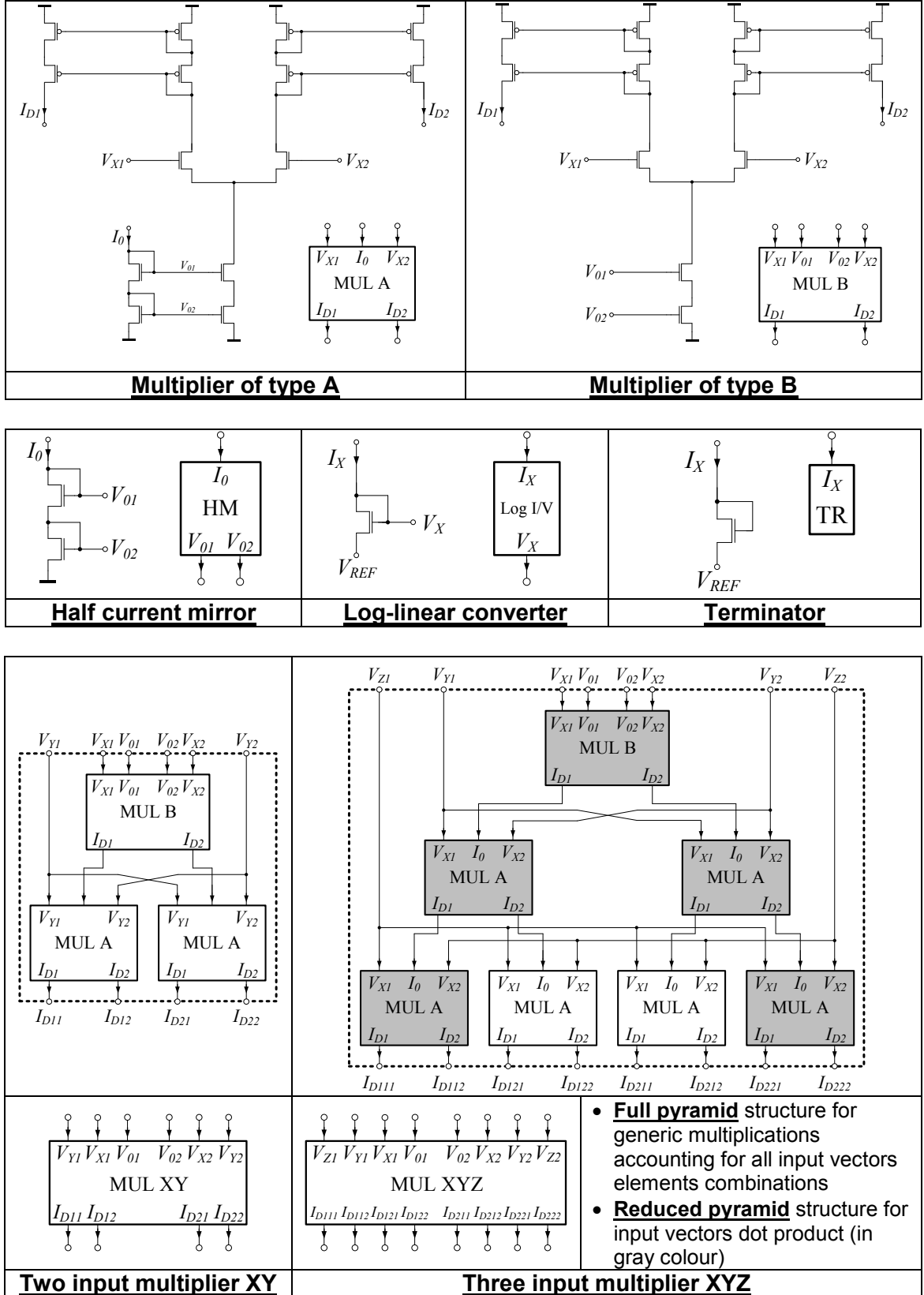
[Tuinhout 2001] H. P. Tuinhout, M. Vertregt, "Characterisation of Systematic MOSFET Current Factor Caused by Metal CMP Dummy Structures", IEEE Transactions on Semiconductor Manufacturing, Vol. 14, No. 4, pp. 302 - 310, Nov. 2001.

- [Turing 36] A. Turing, "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society, Series 2, pp. 230 - 265, Nov. 1936.
- [Unger 58] S. H. Unger, "A Computer Oriented Toward Spatial Problems", Proceedings of IRE, Vol. 46, No. 10, pp. 1744 - 1750, Oct. 1958.
- [Vasyiltsov 2008] I. Vasyiltsov, E. Hambardzumyan, Y. S. Kim, B. Karpinsky, "Fast Digital TRNG Based on Metastable Ring Oscillator", Lecture Notes in Computer Science, Vol. 5154, pp. 164 - 180, 2008.
- [Wang 93] Z. Wang, "A Four-Quadrant Analog Multiplier Using MOS Transistors in the Saturation Region", IEEE Transactions on Instrumentation and Measurement, Vol. 42, No. 1, Feb. 1993.
- [Wang 2013] R. Wang, G. Cohen, T. J. Hamilton, J. Tapson, A. van Schaik, "An Improved aVLSI axon with programmable delay using spike timing dependant delay plasticity", International Symposium on Circuits and Systems (ISCAS), May 2013.
- [Watanabe 84] K. Watanabe, G. C. Temes, "A Switched-Capacitor Multiplier/Divider with Digital and Analog Outputs", IEEE Transactions on Circuits and Systems, Vol. CAS-31, No. 9, pp. 796 - 800, Sep. 1984.
- [Wee 2001] K. H. Wee, T. Shibata, T. Ohmi, "A Simple Random Noise Generator Employing Metal-Oxide-Semiconductor-Field-Effect-Transistor Channel  $kT/C$  Noise and Low-Capacitance Loading Buffer", Japanese Journal of Applied Physics, Vol. 40, Part 1, No. 7, Jul. 2001.
- [Wegmann 89] G. Wegmann, E. A. Vittoz, "Very accurate dynamic current mirrors", Electronics Letters, Vol. 25, No. 10, May 1989.
- [Wegmann 90] G. Wegmann, E. A. Vittoz, "Analysis and Improvements of Accurate Dynamic Current Mirrors", IEEE Journal of Solid-State Circuits, Vol. 25, No. 3, Jun. 1990.
- [Weste 85] N. Weste, K. Eshragian, "Principles of CMOS VLSI Design A System Design Perspective", AR&T Bell Laboratories, 1985.
- [Wong 83] S. Wong, C. Andre, T. Salama, "Impact of Scaling on MOS Analog Performance", IEEE Journal of Solid-State Circuits, Vol. SC-18, No. 1, pp. 106 - 114, Feb. 1983.
- [Xi 2003] X. Xi, M. Dunga, J. He, W. Liu, K. M. Cao, X. Jin, J. J. Ou, M. Chan, A. M. Niknejad, C. Hu, "BSIM4.3.0 MOSFET Model", Department of Electrical Engineering and Computer Sciences, University of Berkeley, 2003.
- [Xiong 2010] Y. Xiong, C. Kabacoff, J. Franca-Koh, P. Devreotes, D. N. Robinson, P. A. Iglesias, "Automated characterisation of cell shape change during amoeboid motility by skeletonization", BMC System Biology, 4:33, Mar. 2010.

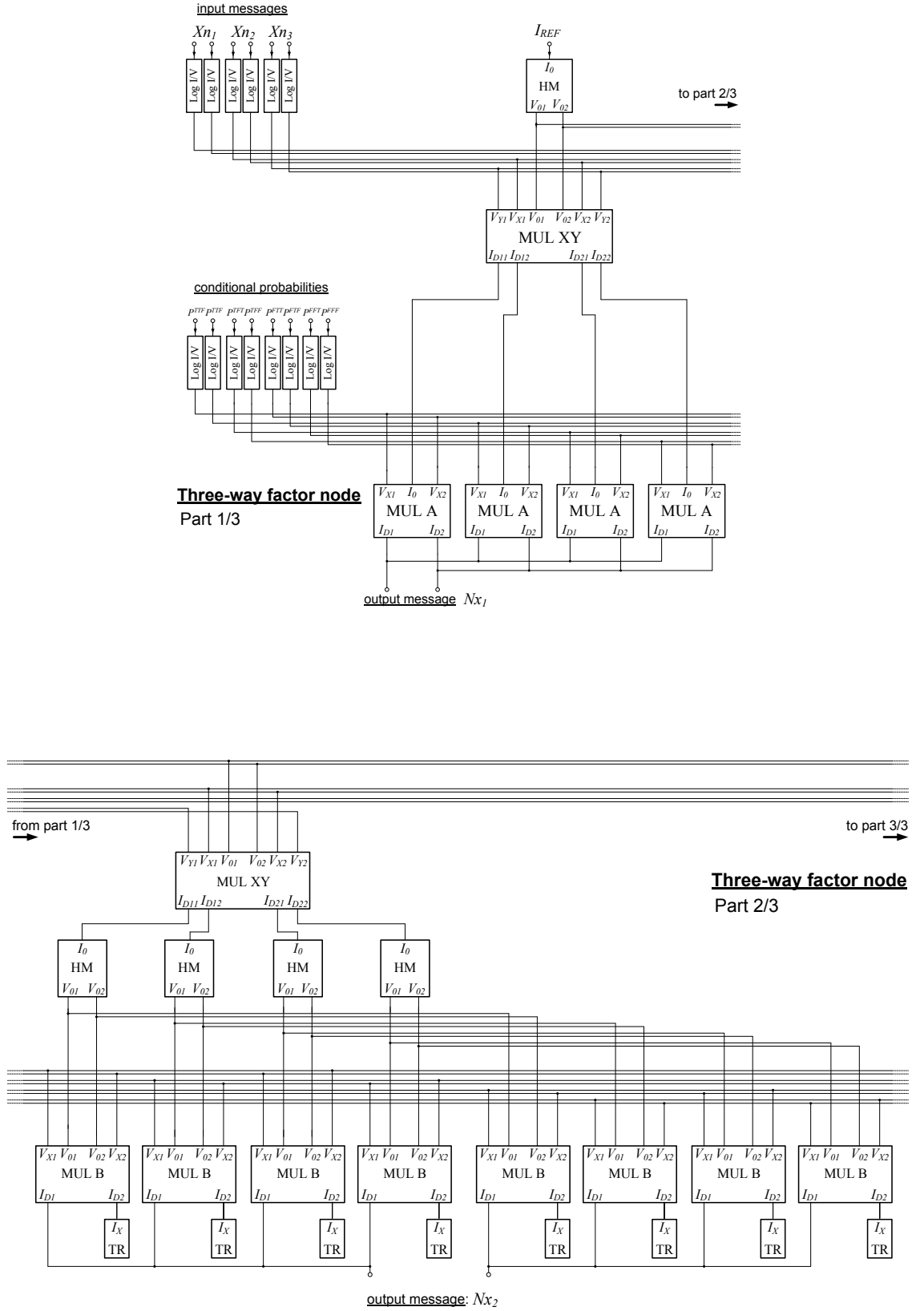
- [Yang 90] H. C. Yang, T. S. Fiez, D. J. Allstot, "Current-feedthrough effects and cancellation techniques in switched-current circuits", IEEE International Symposium on Circuits and Systems (ISCAS), Vol. 4, pp. 3186 - 3188, May 1990.
- [Yasumoto 82] M. Yasumoto, T. Enomoto, "Integrated MOS four-quadrant analogue multiplier using switched-capacitor technique", Electronics Letters, Vol. 18, No. 18, Sep. 1982.
- [Yeh 2001] T. H. Yeh, J. C. H. Lin, S. C. Wong, H. Huang, J. Y. C. Sun, "Mis-match Characterisation of 1.8V and 3.3V Devices in 0.18 $\mu$ m Mixed Signal CMOS Technology", Proceedings of the International Conference on Microelectronic Test Structures (ICMTS), pp. 77 - 82, 2001.
- [Zarandy 2011] A. Zarandy, "Focal-Plane Sensor-Processor Chips", Springer, 2011.
- [Zaveri 2010] M. S. Zaveri, D. Hammerstrom, "CMOL/CMOS Implementations of Bayesian Polytrees Inference: Digital and Mixed-Signal Architectures and Performance/Price", IEEE Transactions on Nanotechnology, Vol. 9, No. 2, pp. 194 - 211, Mar. 2010.
- [Zhang 2011] W. C. Zhang, Q. Y. Fu, N. J. Wu, "A Programmable Vision Chip Based on Multiple Levels of Parallel Processors", IEEE Journal of Solid State Circuits, Vol. 46, No. 9, pp. 1 - 16, Sep. 2011.
- [Zhou 2007] H. Zhou, S. Sakane, "Mobile robot localization using active sensing based on Bayesian networks inference", Journal of Robotics and Autonomous Systems, Vol. 55, pp. 292 - 305, 2007.

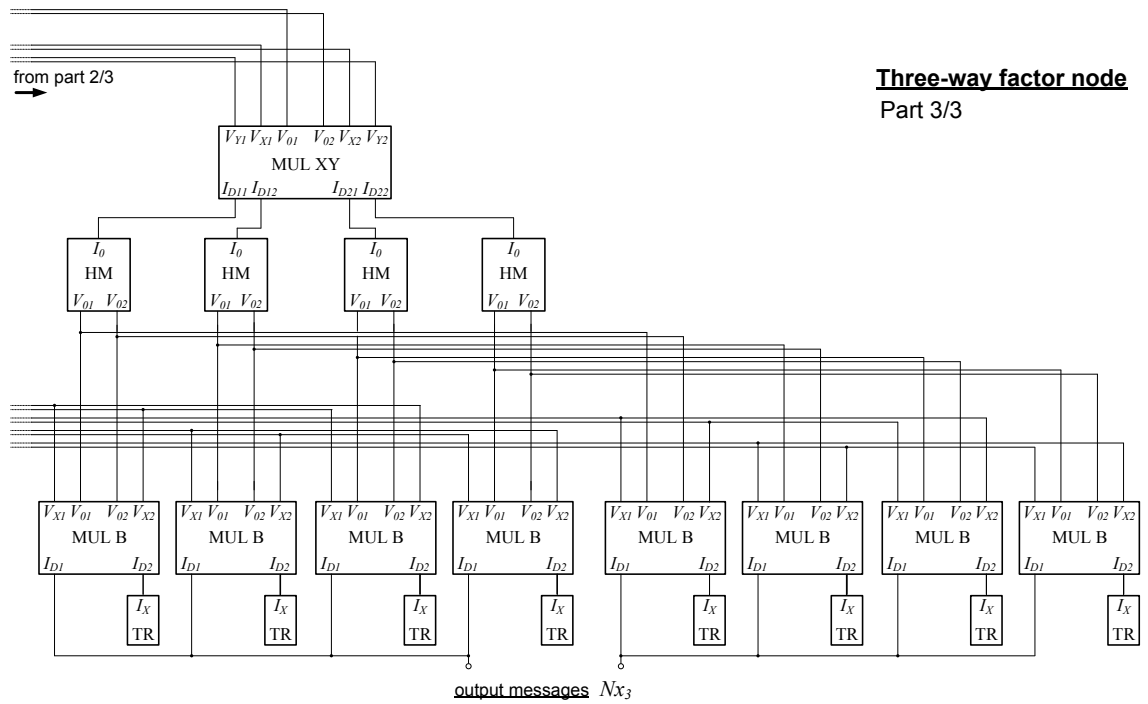
## Appendix A: Continuous-time analogue VLSI circuits for Bayesian inference - schematics and structures

### A.1 Schematic and block diagrams of the continuous-time circuit structures

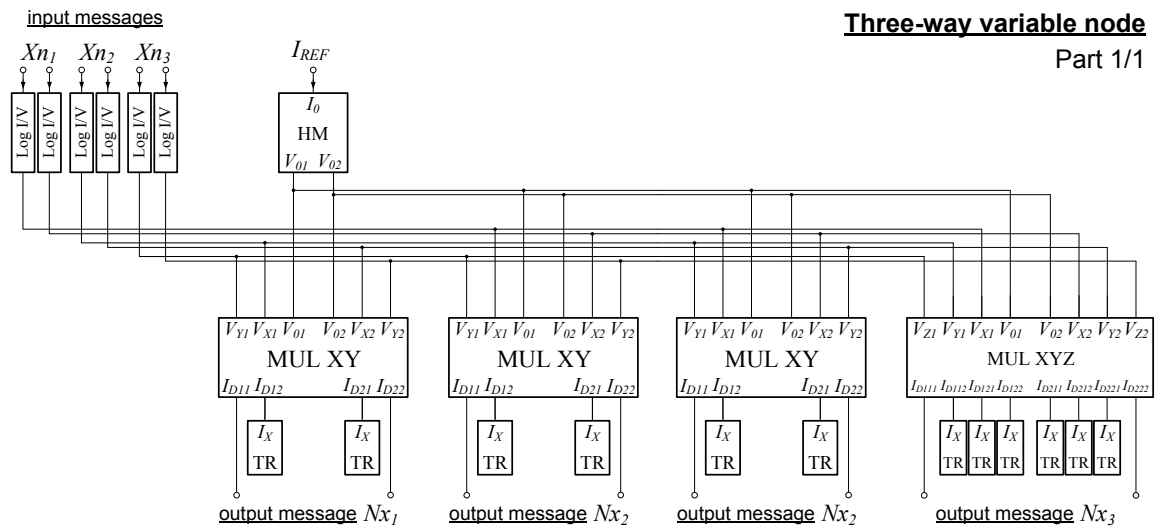


## A.2 Block diagram of the three-way factor node





### A.3 Block diagram of the three-way variable node



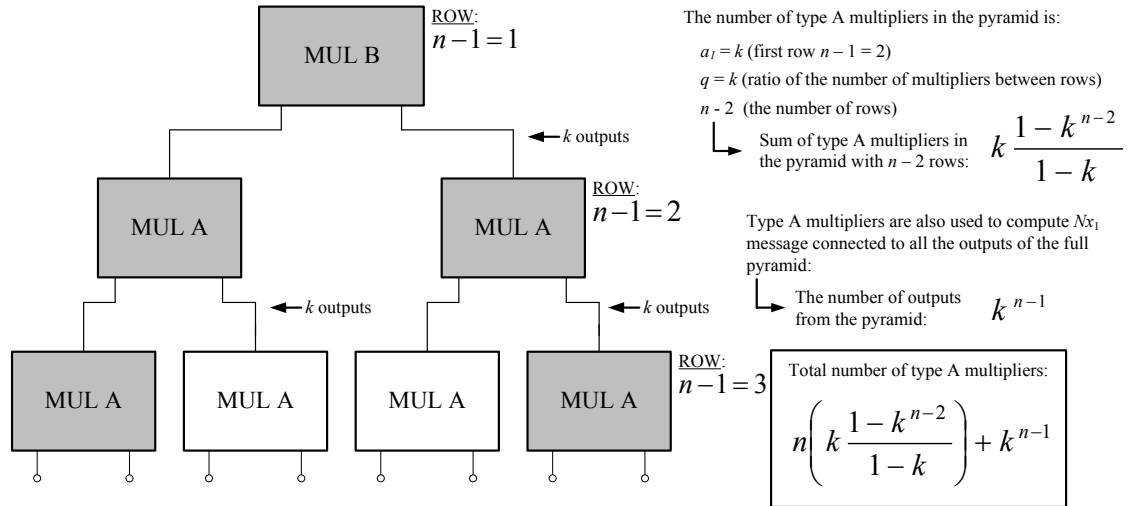
## Appendix B: Continuous-time analogue VLSI circuits for Bayesian inference - scaling rules

### Assumptions and general rules:

- $n$  - the number of ways ( $n \geq 2$ ).
- $k$  - the number of states of the represented variable ( $k \geq 2$ ).
- Type A multiplier:  $5k + 4$  MOS transistors (each branch for the element from the  $k$ -element input vector consists of 5 MOS transistors plus 4 MOS transistors in the bottom current mirror).
- Type B multiplier:  $5k + 2$  MOS transistors (each branch for the element from the  $k$ -element input vector consists of 5 MOS transistors plus 2 MOS transistors in the bottom current source).

### A.5 Scaling rules for factor node

#### A.5.1 The number of type A multipliers



#### A.5.2 The number of type B multipliers

##### Derivation:

- Type B multipliers are connected to the bottom of the pyramid in each link apart from the first one (computing message  $Nx_1$ ). Each output expands through half current mirrors to  $k$  type B multipliers. Since there are  $k^{n-1}$  outputs from each pyramid, the number of type B multipliers in each link is  $k^{n-1} \times k = k^n$ .
- Each link, apart from the first one (computing message  $Nx_1$ ), has one more type B multiplier at the top of the pyramid, therefore, the number of type B multipliers is  $k^{n+1}$ .
- There is only one type B multiplier in the first link computing message  $Nx_1$ .

The total number of type B multipliers is:  $(n-1)(k^n + 1) + 1$ .

### A.5.3 The number of terminating blocks TN

Derivation:

- Terminating blocks are used to terminate unused outputs of type B multipliers in all  $n - 1$  output links (except for the first link computing message  $Nx_1$ ).
- Since each link has  $k^n$  type B multipliers connected to the bottom of the pyramid (see rule A.5.2), there are  $k^n \times (k - 1)$  terminators (in each link).

The total number of terminators is:  $(n-1)(k-1)k^n$ .

### A.5.4 The number of log-linear converters

Derivation:

Log-linear converters are used to convert input currents into voltages of the input messages and the conditional probabilities:

- Since  $n$   $k$ -element vectors generated  $nk$  signals, there are  $nk$  log-linear converters used at the input.
- Since the number of elements in the CPT is  $k^n$ , there are  $k^n$  log-linear converters used for parameters.

The total number of log-linear converters is:  $k^n + nk$ .

### A.5.5 The number of half current mirrors

Derivation:

- Half mirrors are used at the outputs of the pyramids in  $(n - 1)$  links (apart from the first one computing message  $Nx_1$ ). Since there are  $k^{n-1}$  outputs from the pyramid (see rule A.5.1), there are  $(n - 1) k^{n-1}$  half mirrors in each link.
- There is one half mirror used to distribute the reference current  $I_{REF}$ .

The total number of half current mirrors is:  $(n-1)k^{n-1} + 1$ .

### A.5.6 Supply current and power

Assumptions:

- Each row in the pyramid consumes in total  $2I_{REF}$  current ( $1 \times I_{REF}$  to bias the differential pair and  $1 \times I_{REF}$  that splits evenly to the outputs).
- Input currents are not accounted for.

Derivation:

- The first link (computing message  $Nx_1$ ) consists of  $n - 1$  rows in the pyramid ( $n - 1$  input messages to multiply) and one row of multipliers at the bottom of the pyramid. This gives  $n$  rows in total and  $2nI_{REF}$  current consumption.
- The remaining  $n - 1$  links have pyramids with  $n - 1$  rows each, and  $k$  extra rows of the multipliers connected to the bottom of the pyramid (each half current mirror splits to  $k$  multipliers generating  $k$  separate rows, see rule A.5.4). This gives  $n - 1 + k$  rows and results in  $2(n - 1 + k)I_{REF}$  supply current.

The total supply current is:  $(2n + 2(n - 1)(n - k + k))I_{REF}$ .

**A.5.7 Two-argument multiplications**Derivation (one link):

$$\begin{bmatrix} NX_1^T \\ NX_2^F \end{bmatrix} = \alpha \cdot \begin{bmatrix} P^{ITT} & P^{ITF} & P^{TFT} & P^{TFF} \\ P^{FTT} & P^{FTF} & P^{FFT} & P^{FFF} \end{bmatrix} \cdot \begin{bmatrix} XN_2^T \cdot XN_3^T \\ XN_2^T \cdot XN_3^F \\ XN_2^F \cdot XN_3^T \\ XN_2^F \cdot XN_3^F \end{bmatrix}$$

$\downarrow$  matrix has  $k$  rows and  $k^{n-1}$  columns

$\rightarrow (n-2)$  columns of two-argument dot products  
 $k^{n-1}$  rows of two-argument dot products  $\rightarrow (n-2)k^{n-1}$  two-argument multiplications

$k \cdot k^{n-1}$  two-argument multiplications in matrix-vector operations

The total number of two-argument multiplications is:  $n((n-2)k^{n-1} + k^n)$ .

**A.5.8 Two-argument additions**Derivation:

- The number of two-argument additions per row of the probability matrix is:  $k^{n-1} - 1$
- The number of rows of the probability matrix is:  $k$

The total number of two-argument additions is  $nk(k^{n-1} - 1)$ .

**A.5.9 Normalisations**

The number of  $k$ -element vector normalisations is equal to the number of links  $n$ .

**A.5.10 Coefficients**

The number of coefficients in CPT is  $k^n$ .

## A.6 Scaling rules for variable node

### A.6.1 The number of type A multipliers

#### Assumptions:

- Since variable node does not need a full pyramid to evaluate a dot product of  $k$ -element vectors, two options of implementation can be considered based on:
  - full pyramid-structure multipliers (including all multipliers inside the pyramid),
  - reduced pyramid-structure multipliers (including only outer multipliers necessary to compute dot product).
- The approach using reduced pyramid-structure multipliers provides more compact implementation and will be considered in the formulation of scaling properties.

#### Derivation:

<i>Feature</i>	<i>Full pyramid</i>	<i>Reduced pyramid</i>
The number of type A multipliers in one link: (in reduced pyramid: $k$ - number of multipliers in a row, $(n - 2)$ - number of rows)	$k \frac{1 - k^{n-2}}{1 - k}$	$k(n - 2)$
The number of type A multipliers used to compute belief: (all $n$ input links are processed)	$k \frac{1 - k^{n-1}}{1 - k}$	$k(n - 1)$
The total number of type A multipliers:	$nk \frac{1 - k^{n-2}}{1 - k} + k \frac{1 - k^{n-1}}{1 - k}$	$nk(n - 2) + k(n - 1)$

The number of type A multipliers in the realisation of variable node with reduced pyramids is:  $nk(n - 2) + k(n - 1)$ .

### A.6.2 The number of type B multipliers

There is only one type B multiplier per link (at the top of each pyramid) and one at the top of the pyramid for belief calculation, therefore the number of type B multipliers is  $n + 1$ .

### A.6.3 The number of terminating blocks TN

#### Derivation:

In the full pyramid realisation, terminators connect to all unused outputs at the bottom of the pyramids:

- The number of bottom links is  $k^{n-1}$ , out of which  $k$  is used for output. Therefore the number of terminators is  $k^{n-1} - k$ .
- Since computing belief requires the same operations but for all the inputs, the number of terminators will be:  $k^n - k$ .

The total number of terminators in the full pyramid realisation is:  $n(k^{n-1} - k) + k^n - k$ .

In the half pyramid realisation, each type A multiplier is terminated with  $k - 1$  terminators.

- The number of type A multipliers in one pyramid is  $k(n - 2)$ , see rule A.6.1.
- The number of terminator in each link is  $k(n - 2)(k - 1)$ .
- Since computing belief requires the same operations but for all the inputs, the number of terminators will be:  $k(n - 1)(k - 1)$ .

The total number of terminators in the half pyramid realisation is:

$$n(k(n - 2)(k - 1)) + k(n - 1)(k - 1).$$

#### A.6.4 The number of log-linear converters

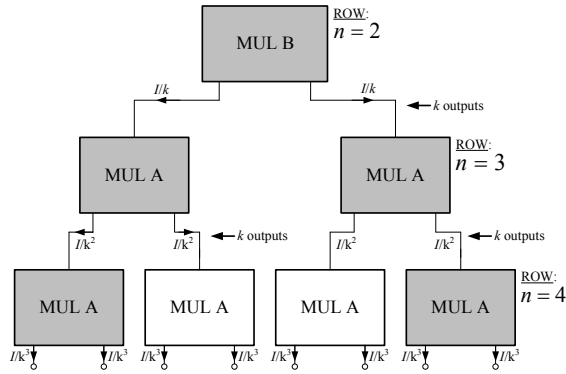
Log-linear converters are used to convert input currents into voltages of the input messages, therefore, the number of converters is:  $nk$ .

#### A.6.5 The number of half current mirrors

There is only one half current mirror used to distribute the reference current  $I_{REF}$ .

#### A.6.6 Supply current and power

Derivation:



Total current of variable node with  $n$  links using full pyramid structures:  $[n(2(n - 1)) + 2n]I_{REF}$

Total current of variable node with  $n$  links using reduced pyramid structures:

$$\begin{aligned} n = 2, 3 & \quad [n(2(n - 1)) + 2n]I_{REF} \\ n = 4 & \quad 12I_{REF} + \left[4 + \frac{1}{k} + \frac{2}{k^2} - \frac{1}{k^2}\right]I_{REF} \\ n > 4 & \quad (n + 1) \left[4 + \frac{1}{k} + \frac{2}{k^2(1 - (1/k))}\right]I_{REF} \end{aligned}$$

FULL	REDUCED
$2I$	$2I$
$2I$	$2I$
$2I$	$I/k + I/k^2$

$$\begin{aligned} n = 4 & \quad \begin{array}{l} \hookrightarrow 4 + 2 \end{array} \quad \begin{array}{l} \hookrightarrow 4 + \frac{1}{k} + \frac{2}{k^2} - \frac{1}{k^2} \end{array} \\ n = 5 & \quad \begin{array}{l} \hookrightarrow 4 + 4 \end{array} \quad \begin{array}{l} \hookrightarrow 4 + \frac{1}{k} + \frac{2}{k^2} + \frac{2}{k^3} - \frac{1}{k^3} \end{array} \\ n = 6 & \quad \begin{array}{l} \hookrightarrow 4 + 6 \end{array} \quad \begin{array}{l} \hookrightarrow 4 + \frac{1}{k} + \frac{2}{k^2} + \frac{2}{k^3} + \frac{2}{k^4} - \frac{1}{k^4} \end{array} \\ & \quad \begin{array}{l} \hookrightarrow 2(n-1) \end{array} \quad \begin{array}{l} \hookrightarrow 4 + \frac{1}{k} + \frac{2}{k^2} \frac{1 - (1/k)^{n-3}}{1 - (1/k)} - \frac{1}{k^{n-2}} \end{array} \\ & \quad \begin{array}{l} \hookrightarrow 4 + \frac{1}{k} + \frac{2}{k^2(1 - (1/k))} \end{array} \end{aligned}$$

### A.6.7 Two-argument multiplications

Derivation:

$$\begin{bmatrix} Bel^T \\ Bel^F \end{bmatrix} = \alpha \cdot \begin{bmatrix} Xn_1^T \cdot Xn_2^T \cdot Xn_3^T \\ Xn_1^F \cdot Xn_2^F \cdot Xn_3^F \end{bmatrix}$$

→ (n - 2) columns of two-argument dot products

→ k rows of two-argument dot products

→

$k(n - 2)$  two-argument multiplications per link

$k(n - 1)$  two-argument multiplications to compute belief

The total number of two-argument multiplications is:  $nk(n - 2) + k(n - 1)$ .

### A.6.8 Normalisations

The number of  $k$ -element vector normalisations is equal to the number of links  $n$  plus one normalisation of belief.