TKK Dissertations 79
Espoo 2007

DENSE IMPLEMENTATIONS OF BINARY CELLULAR NONLINEAR NETWORKS: FROM CMOS TO NANOTECHNOLOGY

Doctoral Dissertation

Jacek Flak

Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory

TKK Dissertations 79
Espoo 2007

DENSE IMPLEMENTATIONS OF BINARY CELLULAR NONLINEAR NETWORKS: FROM CMOS TO NANOTECHNOLOGY

Doctoral Dissertation

Jacek Flak

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 29th of June, 2007, at 12 noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Piiritekniikan laboratorio

Distribution:
Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory
P.O. Box 3000
FI - 02015 TKK
FINLAND
URL: http://www.ecdl.tkk.fi/
Tel. +358-9-451 2271
Fax +358-9-451 2269
E-mail: jacek@ecdl.tkk.fi

ISBN 978-951-22-8853-3
ISBN 978-951-22-8854-0 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: http://lib.tkk.fi/Diss/2007/isbn9789512288540/

TKK-DISS-2321

Multiprint Oy
Espoo 2007

Abstract

This thesis deals with the design and hardware realization of the cellular neural/nonlinear network (CNN)-type processors operating on data in the form of black and white (B/W) images. The ultimate goal is to achieve a very compact yet versatile cell structure that would allow for building a network with a very large spatial resolution. It is very important to be able to implement an array with a great number of cells on a single die. Not only it improves the computational power of the processor, but it might be the enabling factor for new applications as well. Larger resolution can be achieved in two ways. First, the cell functionality and operating principles can be tailored to improve the layout compactness. The other option is to use more advanced fabrication technology – either a newer, further downscaled CMOS process or one of the emerging nanotechnologies.

It can be beneficial to realize an array processor as two separate parts – one dedicated for gray-scale and the other for B/W image processing, as their designs can be optimized. For instance, an implementation of a CNN dedicated for B/W image processing can be significantly simplified. When working with binary images only, all coefficients in the template matrix can also be reduced to binary values. In this thesis, such a binary programming scheme is presented as a means to reduce the cell size as well as to provide the circuits composed of emerging nanodevices with an efficient programmability. Digital programming can be very fast and robust, and leads to very compact coefficient circuits. A test structure of a binary-programmable CNN has been designed and implemented with standard 0.18 μm CMOS technology. A single cell occupies only 155 μm^2, which corresponds to a cell density of 6451 cells per square millimeter. A variety of templates have been tested and the measured chip performance is discussed.

Since the minimum feature size of modern CMOS devices has already entered the nanometer scale, and the limitations of further scaling are projected to be reached within the next decade or so, more and more interest and research activity is attracted by nanotechnology. Investigation of the quantum physics phenomena and development of new devices and circuit concepts, which would allow to overcome the CMOS

limitations, is becoming an increasingly important science. A single-electron tunneling (SET) transistor is one of the most attractive nanodevices. While relying on the Coulomb interactions, these devices can be connected directly with a wire or through a coupling capacitance. To develop suitable structures for implementing the binary programming scheme with capacitive couplings, the CNN cell based on the floating gate MOSFET (FG-MOSFET) has been designed. This approach can be considered as a step towards a programmable cell implementation with nanodevices. Capacitively coupled CNN has been simulated and the presented results confirm the proper operation. Therefore, the same circuit strategies have also been applied to the CNN cell designed for SET technology. The cell has been simulated to work well with the binary programming scheme applied. This versatile structure can be implemented either as a pure SET design or as a SET-FET hybrid. In addition to the designs mentioned above, a number of promising nanodevices and emerging circuit architectures are introduced.

Keywords: Integrated Circuits, Cellular Neural/Nonlinear Networks, Cellular Array Processors, CMOS, Nanotechnology, Single-Electron Tunneling

Preface

This thesis is the result of the research carried out at the Electronic Circuit Design Laboratory (ECDL) of the Helsinki University of Technology during the years 2001-2007. The research was funded by the Academy of Finland within the projects "Integrated Parallel Processors for Future Data Processing and Analyzing Systems" and "Low power biomorphic neural circuits based on floating gate MOS and SET transistors". Also, the financial support from the Jenny and Antti Wihuri Foundation is greatfully acknowledged.

Throughout the years, the working environment in the laboratory has been an amazing mixture of a great inspiration, challenges, continuous learning and relaxing moments of fun. All the staff at ECDL deserve my gratitude for creating such a fantastic atmosphere. However, I would like to name a few persons for their exceptional contribution. First of all, I need to mention Professor Veikko Porra and Dr. Witold Machowski, since without their idea for having a student exchange I wouldn't be writing these words. I want to thank Professor Kari Halonen for providing me with an interesting research topic, and supervising my postgraduate studies that will be crowned with this dissertation. My introduction to the world of CNNs and lots of efforts in organizing this project have been an invaluable contribution of Professor Ari Paasio, who has mentored my early research. Also, I want to express my gratefulness to Dr. Mika Laiho, who became my longtime tutor when Prof. Paasio has got overwhelmed by the Dean's splendor and duties at the University of Turku. It's been a real pleasure to work with Mika, and I've learned from him a lot. Also, my special thanks go to our dear secretary, Helena Yllö, who (so many times) has made my life easier by taking care of practical matters. She's the person I could rely on whenever facing a bureaucracy. I need to acknowledge Väinö Hakkarainen for all the years of being a great roommate and for the superb music we've used to listen to. Also, countless discussions with Dr. Marko Kosunen, Dr. Lauri Koskinen, Dr. Asko Kananen, Mika Länsirinne, Jonne Lindeberg and Mikko Talonen (to list just a few) have been both thoughtful and enjoyable.

For the hard work of pre-examiners, my special thanks go to Professor Stephen Goodnick (Arizona State University, AZ, USA) and Professor Victor Brea (University of Santiago de Compostela, Spain). Their devotion and a number of invaluable comments have helped me to improve this manuscript.

I want to mention my wife Anne and my son Patrik – my pride and joy, a sunshine of my life. I am grateful for their love, support, understanding and indulgence.

All my friends, here in Finland as well as back there in Poland, must be acknowledged for providing me with great social activities, sharing passions and fun.

Finally, I want to thank my parents Irena and Stanisław and my sister Katarzyna for their persistent support, motivation in the moments of doubt, and all the love they filled my life with.

Espoo, 02.06.2007

Jacek Flak

Contents

Symbols and Abbreviations

1-D	One dimensional
3IS	Three Island Structure
$\hbar$	reduced Planck's constant
μ_n	mobility of electron
μ_p	mobility of hole
ϕ_F	floating gate potential
τ	CNN cell time constant
ε	arbitrarily small value (disturbance)
A	CNN feedforward template
$A_{k,l}$	interconnection strength (weight); feedforward template term
AB	template matrix in binary programming scheme
$AB_{k,l}$	interconnection strength (weight); binary template term
B	CNN feedback template
$B_{k,l}$	interconnection strength (weight); feedback template term
C	unit capacitance
$C_{\overline{m}}$	parasitic gate capacitance used as dynamic memories
C_C	parasitic gate capacitance used as dynamic memories
C_c	coupling capacitance
C_g	gate capacitance
$C_{i,j}$	CNN cell at i^{th} row and j^{th} column

C_i	input coupling capacitance
C_L	load capacitance
C_m	parasitic gate capacitance used as dynamic memories
C_{sub}	substrate capacitance
C_S	parasitic gate capacitance used as dynamic memories
C_T	tunnel capacitance
C_X	CNN state capacitor
d	thickness of the tunnel barrier
e	electron (particle or charge)
E_C	Coulomb energy
E_F	Fermi level energy
F_i	weighted input signal (synapse output)
h	Planck's constant
I_{bias}	bias current
I_d	drain current
k_B	Boltzmann's constant
L_n	length of channel of NMOS
L_p	length of channel of PMOS
M	number of rows in array
N	number of columns in array
N_r	Neighborhood of CNN cell
R_Q	natural quantum unit of resistance
R_T	tunnel resistance
S	threshold level for the activation function
T	temperature
$u_{i,j}$	CNN cell input

V_{DD}	supply voltage
V_{ds-th}	threshold value of drain-source voltage
V_{ds}	drain-source voltage
V_g	gate voltage
V_{in}	input voltage
V_i	input voltage
V_m	comparator midpoint value
V_{SS}	ground level voltage
V_{th}	comparator (inverter) threshold voltage
V_{Tn}	NMOS threshold voltage
V_{Tp}	PMOS threshold voltage
V_w	voltage representing weight value
V_X	voltage representing cell state
V_Y	voltage representing cell output
W_i	weight value
W_n	width of channel of NMOS
W_p	width of channel of PMOS
$x_{i,j}$	CNN cell state
X_i	synapse input signal
Y	classified output signal
$y_{i,j}$	CNN cell output
z	CNN bias template
νMOS	neuron MOS
A/D	Analog-to-Digital
AER	Address-Event Representation
aka	also known as

ALU	Arithmetic Logic Unit
AND	logic AND operation
ANN	Artificial Neural Network
APE	Analog Processing Element
B/W	Black and White
C-SET	Capacitively coupled SET device
C●RAM	Computational RAM
CAM	Content Addressable Memory
CMOL	CMOS/nanowire/MOLecular hybrid
CMOS	Complementary MOS
CNN	Cellular Neural/Nonlinear Network
CNN-UM	CNN Universal Machine
CNT	Carbon NanoTube
CNTFET	Carbon NanoTube FET
D/A	Digital-to-Analog
DAC	Digital-to-Analog Converter
DC	Direct Current
DRAM	Dynamic RAM
DT-CNN	Discrete-Time CNN
FET	Field-Effect Transistor
FG-MOSFET	Floating Gate MOSFET
FSR	Full Signal Range
GAPU	Global Analogic Programming Unit
GLU	Global Logic Unit
HI	logic high signal level
I/O	Input/Output

I/V	current-to-voltage (converter)
IC	Integrated Circuit
IP	Internet Protocol
ITRS	International Technology Roadmap for Semiconductors
L	Length of a MOSFET channel
LLU	Local Logic Unit
LO	logic low signal level
LWR	Line Width Roughness
MDW	Moving Domain Wall
MIM	Metal-Insulator-Metal
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MWCNT	Multi-Wall Carbon NanoTube
N/A	Not Available
NAND	logic NAND operation
NDR	Negative Differential Resistance
NMOS	MOSFET with N-type channel
NOR	logic NOR operation
NSIP	Near-Sensor Image Processing
OR	logic OR operation
PE	Processing Element
PMOS	MOSFET with P-type channel
PRHG	Positive-Range High-Gain
PWL	PieceWise Linear
QCA	Quantum Cellular Automata
QCIF	Quarter Common Intermediate Format

QD	Quantum Dot
R-SET	Resistively coupled SET device
RAM	Random Access Memory
RBC	Random Background Charge
RC	Resistor Capacitor
RIE	Reactive Ion Etching
RTD	Resonant Tunneling Diode
RTT	Resonant Tunneling Transistor
SCAMP	SIMD Current-mode Analogue Matrix Processor
SE	South-East (direction)
SET	Single-Electron Tunneling
SFB	Self FeedBack
SIMD	Single-Instruction Multiple-Data
SOI	Silicon On Insulator
SRAM	Static RAM
SW	South-West (direction)
SWCNT	Single-Wall Carbon NanoTube
TLG	Threshold Logic Gate
W	Width of a MOSFET channel
W/L	Width over Length
XOR	logic eXclusive-OR operation

Chapter 1

Introduction

1.1 Motivation

For decades, the processing of information has been based on high precision computation. This approach has led to a development of digital processors with enormous computing power. However, a number of applications exist, in which the traditional strategy is insufficient. Especially in tasks of real-time image processing, e.g., vision systems, conventional computers are still far behind the neurobiological systems of living organisms. For instance, when a small insect flies, it performs an environment perception and navigation tasks among others. Such a complex computation is done at a fraction of time and power that would take with the advanced digital computer. Obviously, there is a huge gap in terms of the overall robustness, performance, and energy efficiency that engineers need to bridge. Through millions of years the evolution has developed and optimized the functionality and architectures of nervous systems. Therefore, a great lesson on the efficiency in complex computation can be learned from the nature. Studies of the physiological phenomena, on which biological receptors and nervous systems are based, reveal a substantial difference in conceptual and architectural approach to information processing between the systems developed by nature and engineers. In recent years, many endeavors to adapt the strategies known from neurobiology to the electronics have resulted in new theories and paradigms. Among them, the concept of cellular neural/nonlinear network (CNN) [1]-[3] is one to play a crucial role in the development of artificial vision systems [4]-[7].

The traditional information processing has largely been benefiting from a rapid progress in the semiconductor technology. For many years, a gain in computational power has been supported by the continuous down-scaling of CMOS technology. With smaller devices, higher chip densities and higher clock frequencies can be reached, yielding to more powerful processors. Even though the CMOS scaling limitation pre-

dicted in the past turned out not to be the ultimate boundary, there definitely are some physical limits beyond which no MOS device will be operational [8],[9]. Alternatively, the skyrocketing costs of fabrication will far exceed the performance improvements of scaled devices[1]. Since CMOS technology has already entered the nanometer scale, quantum effects are increasingly important and at some point will become the dominant phenomena. Therefore, they need to be investigated thoroughly. For the near future, techniques to prevent these effects from destroying the performance of a MOS device are sufficient. However, eventually designers will need ways of employing them instead of trying to avoid them. At a point where the rules of quantum physics become dominant, new devices will be required. Supposedly, such components will rely on different operation principles than MOS transistors, and therefore a new architectural approach will be needed as well. Classical digital designs are not suitable for devices and circuit blocks that lack the large gain, fan-in and fan-out, and worst of all are inherently unreliable. Moreover, traditional architectures become inefficient with scaling CMOS as well. Modern digital circuits as a whole scale down less than MOS transistors themselves due to the growing problem of interconnections. Longer and longer wires introduce larger parasitic components to the circuit. It is also increasingly difficult to route them efficiently. Moreover, the wire width is not scaling down as fast as the MOSFET channel length causing a new density bottleneck. Circuit speed is often limited by the RC delays of the connecting lines rather than by the device switching. This problem gets even worse with emerging nanostructures, due to their inherent low gain and driving capabilities. Hence, more and more attention is brought to the locally-interconnected array-architectures like CNN, which at given circumstances have multiple advantages over the traditional approach.

CNN theory allows to describe convolution-based local operations in a convenient way. It also stands for an architectural approach to realize a massively parallel processor. A computing array consists of (usually) identical and locally interconnected processing elements (PE) called cells. The state of a such cell has a nonlinear activation function and its evolution depends on the initial data, control terms and continuous-time spatio-temporal convolutions. The computing power of a CNN substantially depends on the array size. However, the implementation of a large array is a very challenging task. One of the possibilities to increase the cell density is to realize the gray-scale and black-and-white (B/W) image processing parts separately. Since the number of gray-scale operations used in many CNN applications is very limited and the majority of CNN algorithms contain many B/W image processing tasks, a full programmability seems to be needed only for the B/W processing part. Therefore, both

[1]Currently, high fabrication costs concern the nanotechnology as well. However, the tergeted nanodevices should have much simpler structures than MOSFETs, and thus would allow the use of alternative cost-effective fabrication processes like, e.g., a self-assembly.

gray-scale and B/W processing cores can be significantly simplified [10]. Reducing the number of analog memories and analog weight multipliers potentially brings down the total chip area as well as power consumption and the time spent for processing (performing B/W operations with general-purpose gray-scale image processor takes much more time and energy). Moreover, such dedicated cores can reach a higher robustness. With a focus on the core for processing data in the form of binary images, a binary-programming scheme has been developed to enable a significant simplification of the coefficient circuits. As the number of analog transistors is minimized, the cell density grows. All elements of the template matrices are limited to binary values. Many operations can be performed with a single binary template. Some more complex tasks are computed as a consecutive evaluation of subtasks, and thus the processor remain versatile. Due to digital programming, the template terms can be loaded fast and the overall performance stays competitive. Moreover, this approach can provide the nanodevice-based implementations with fast and robust programmability. The realization of a binary-programmable CNN in single-electron tunneling (SET) technology, presented in this thesis as a proof of concept, can be a starting point for a future development of programmable architectures for nanotechnology integration.

1.2 Research Contribution

This thesis proposes a solution to the cell density limitations encountered in the design of CNN hardware. The separation of the gray-scale and the B/W image processing parts of a CNN system provides the means to optimize their implementations. With a focus on the structures for processing B/W images, the minimum programmability requirements, which preserve the cell versatility, have been studied resulting in the proposed binary programming scheme. Based on this approach, novel compact structures of a binary CNN have been developed for extremely dense implementations with CMOS, floating gate MOSFET (FG-MOSFET), and SET technologies. The proposed CMOS realization enables the cell densities previously thought to be impossible to obtain with CMOS technology. The CNN cell based on FG-MOSFET structure illustrates the method for applying the binary programming scheme to effectively control the networks with capacitive interconnections. Following this method, the SET implementation of a binary CNN cell is proposed, which to the best knowledge of the author is the first programmable SET CNN presented so far and one of the most versatile CNN realizations with nanodevices. It is due to the binary programming scheme that the nanostructures can be provided with fast and robust programmability. The material presented in this thesis can be used to assess and develop the programing schemes for architectures based on other emerging nanodevices as well.

The idea of binary programming scheme had been proposed by Prof. Ari Paasio,

and it was further developed by the author, Prof. Paasio, and Dr. Mika Laiho. Cells in all of the presented designs use the positive-range high-gain output-nonlinearity developed by Prof. Paasio. The author's main contribution was in transforming the traditional templates into binary form. The mathematical formulation of the templates and cell modeling was done by Dr. Laiho. The coefficient structures for CMOS implementation as well as the basic architecture concept of a cell were proposed by Prof. Paasio and further developed by the author. The simulations, layout drawing, and test measurements were performed by the author. Structures developed for implementations with FG-MOSFET and SET transistors result from the author's independent research carried out under the guidance of Dr. Laiho and supervision of Prof. Kari Halonen. A majority of the material included within this thesis has already been published or has been submitted for publication [11]-[20].

1.3 Organization of the Thesis

The thesis consists of nine chapters. Chapter 2 introduces the basics of the CNN theory. Definitions and implementation oriented models are given. The importance of B/W image processing as a special subclass of CNN operations is emphasized.

Chapter 3 presents different approaches to the implementation of parallel image-processing. A selection of hardware realizations of array processors are described to give a view on the prior state-of-the-art.

The binary programming scheme is introduced in Chapter 4. The basic rules for designing the binary templates are given, and a selection of examples are used to form their classification.

Chapter 5 is focused on the CMOS implementation of a binary CNN. System architecture and functionality, cell circuit and operating principles are presented. A brief analysis of the robustness is given and a number of implementation issues are discussed. The measurement results of the fabricated chip are shown as well.

Chapter 6 briefly introduces the concept of the neuron MOSFET (νMOS) structure, i.e. the multi-input floating gate MOSFET (FG-MOSFET). Next, the FG-MOSFET implementation of a binary-programmable CNN is depicted. This design can be considered as a step toward a programmable CNN implementation with nanodevices. It allows for a conceptual development and tests of the capacitive couplings with ON/OFF programmability in the well known MOS technology. The designed cell structure is presented, and the simulation results of an 8×8 network are shown.

Chapter 7 is dedicated to the binary CNN implemented with SET technology. First, the basic features of the SET technology are introduced. Then, basics of artificial neural networks (ANN) and prior state-of-the-art SET ANN designs are presented. Finally, a structure of a binary-programmable CNN cell designed for implementation

either with SET transistors only or as a SET-FET hybrid is depicted and simulation results are shown.

Chapter 8 constitutes a compact review of other promising nanodevice and architecture concepts. Resonant tunneling diode (RTD) as a computing device is described in more detail due to its higher degree of maturity. A number of other device concept are shown to give a broader view on this exciting research field. Additionally, the interesting idea of quantum cellular automata (QCA) as an alternative computing approach and CMOS-nanowire-MOLecular (CMOL) hybrid architecture, which combines the standard CMOS logic and the processing array built with nanodevices, are presented.

Finally, Chapter 9 concludes the thesis and gives some projections of what the further research could be focused on.

This page is intentionally left blank.

Chapter 2

Cellular Neural/Nonlinear Networks

The Cellular Neural/Nonlinear Network (CNN) theory [1] describes an analog parallel processing paradigm. Its hardware implementation is potentially capable of very high speed computation, while a low power consumption is maintained. A CNN can obtain this due to its massively parallel architecture. The interconnected processors, called cells, are arranged in a regular array. Each cell evaluates a global instruction on its own local data. Therefore, a CNN can be classified as a single-instruction multiple-data (SIMD) type processor. Such an operating mode is naturally suitable for image processing, where each cell corresponds to a pixel of an image.

2.1 Definitions

A collection of basic definitions regarding the CNN are given in this section. However, some of these terms are general and can be used in describing other types of array processors as well.

2.1.1 Neural vs. Nonlinear

The term “neural” is often used to describe the network behavior, which can mimic a biological nervous system, while the term “nonlinear network” has a more general meaning. However, the difference between the ”neural” and the “nonlinear” network could also refer to how the interconnection weights are obtained. If the weights are defined by a learning algorithm (adaptation process) implemented within the system, we talk about a “cellular neural network”. Otherwise, i.e. weights are programmed to the predefined values, it is a “cellular nonlinear network”.

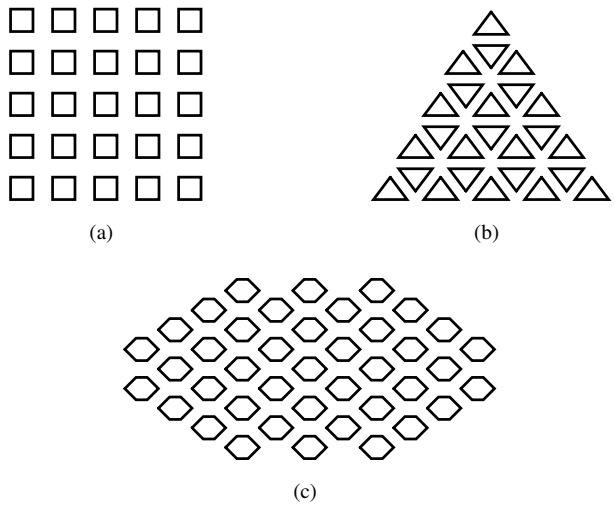


Figure 2.1 The examples of grids: a) rectangular, b) triangular and c) hexagonal.

2.1.2 Grid Type and Array Dimension

The processing elements of CNN are locally interconnected and organized into one- or two-dimensional grid. Higher dimensions are theoretically possible but not suitable for integration. Usually, the cells form a regular rectangular array. However, that is not the only possible layout. Cells can also be arranged in other grids, such as hexagonal or triangular. These examples are shown in Figure 2.1. In this thesis, only the rectangular arrangement is used. To identify an individual cell within the $M \times N$ sized array, a symbol $C_{i,j}$ is used to denote a cell in the i^{th} row and j^{th} column; $i \in [1,M]$ and $j \in [1,N]$.

2.1.3 Neighborhood and Interconnection Types

A cell within a CNN can have different types of neighborhoods and interconnections. The neighborhood of a cell defines the distance of the furthermost connections. Figure 2.2 presents examples of the 1- and 2-neighborhoods in a rectangular grid.

A cell is not necessarily connected to all of the neighboring cells. For instance, a cell in a rectangular grid can be 4- or 8-connected with the closest neighbors, as shown in Figure 2.3. Furthermore, an application may exist where only the cells at a certain distance within the neighborhood can be of interest. In case of a 3-neighborhood, a particular cell can interact with the cells at a distance of, say, 1 and 3 without dealing with the cells in between, i.e. at a distance of 2. An example of such a specific case can be found in Reference [21].

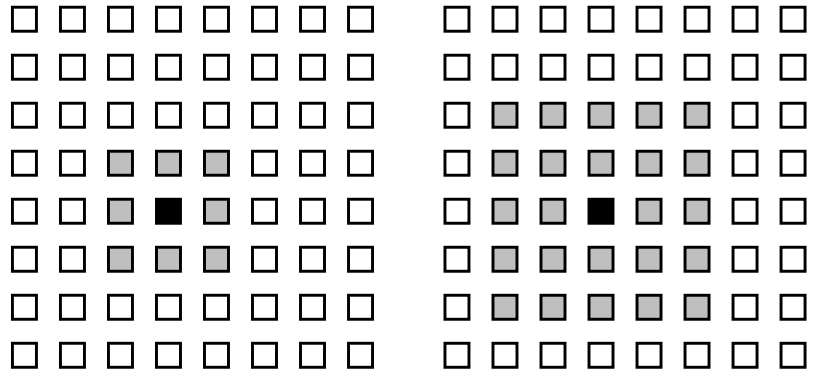


Figure 2.2 The gray cells are connected to the black cell, and thus form the a) 1-neighborhood, or b) 2-neighborhood.

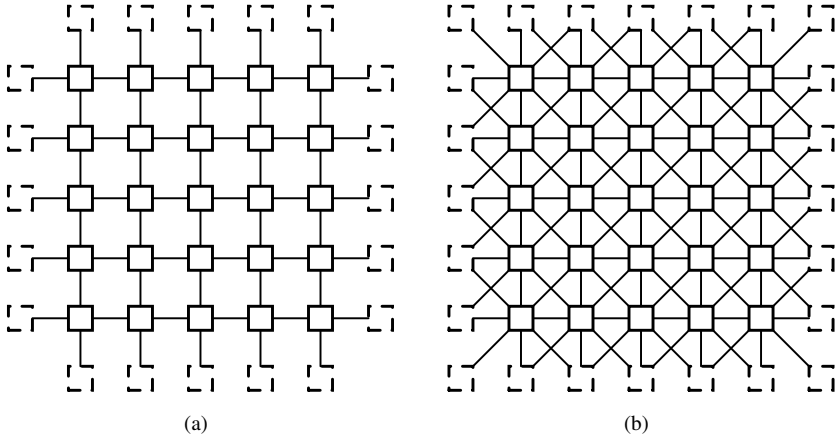


Figure 2.3 The CNN cells arranged in the a) 4-connected and b) 8-connected 1-neighborhood. Dashed line indicates that a cell is a border cell.

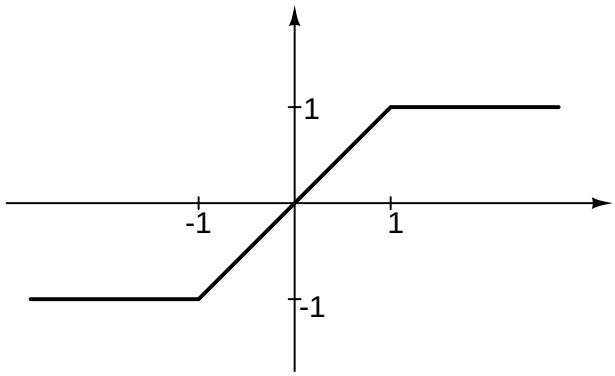


Figure 2.4 CNN cell output nonlinearity

2.2 Continuous-Time CNN

Each cell in the array has a state and a constant input. The state of a cell is determined by the inputs and outputs of the directly connected neighborhood cells. All of these contribute to define the cell output. The standard CNN dynamics can be described with the state equation [1]

$$\frac{dx_{i,j}}{dt} = -x_{i,j} + \sum_{C_{k,l} \in N_r(i,j)} A(i,j;k,l) \cdot y_{k,l} + \sum_{C_{k,l} \in N_r(i,j)} B(i,j;k,l) \cdot u_{k,l} + z \qquad (2.1)$$

where $x_{i,j}$ is the cell state, $u_{k,l}$ is the input and $y_{k,l}$ is the output of each cell $C_{k,l}$ within the neighborhood $N_r(i,j)$ of the cell $C_{i,j}$. A and B are the feedback and feedforward templates, respectively. These templates are the collections of coefficient weights, which determine the strength of each interconnection between the cell $C_{i,j}$ and its neighborhood. Finally, z determines the operating point of the cell and is called a bias template. The cell output has nonlinear activation function (relation between $x_{i,j}$ and $y_{i,j}$), usually in the form of piecewise linear (PWL) sigmoid as shown in Figure 2.4 and given by

$$y_{i,j} = f(x_{i,j}) = \frac{1}{2}(|x_{i,j}+1| - |x_{i,j}-1|). \qquad (2.2)$$

If the feedback and feedforward templates have coefficients set identical for each cell regardless of its position in the network, the templates are space invariant. In such a case, assuming the 1-neighborhood, these templates take the form of

$$A = \begin{bmatrix} A_{-1,-1} & A_{0,-1} & A_{1,-1} \\ A_{-1,0} & A_{0,0} & A_{1,0} \\ A_{-1,1} & A_{0,1} & A_{1,1} \end{bmatrix} \quad B = \begin{bmatrix} B_{-1,-1} & B_{0,-1} & B_{1,-1} \\ B_{-1,0} & B_{0,0} & B_{1,0} \\ B_{-1,1} & B_{0,1} & B_{1,1} \end{bmatrix} \quad z. \tag{2.3}$$

So, only 19 terms are required. An operation performed by a CNN is determined by programming these terms. The matrices A and B are called "cloning templates" and z is called bias template or threshold. The center element of the cloning template matrix represents a self-feedback term. A number of CNN templates with the names describing the corresponding network-operation have already been proposed [22].

2.3 CNN Model Modifications

2.3.1 CNN Universal Machine

The original CNN has been supplemented by a number of additional functionalities resulting in the CNN Universal Machine (CNN-UM) [23]. The set of cells' new features include Boolean logic, control logic, and memories (both analog and digital). Moreover, the global analogic programming unit (GAPU) comprising program registers, configuration registers, etc. has been added to the system. These extensions enable algorithmic operations consisting of several templates and logic operations.

2.3.2 Full Signal Range CNN

Even though the CNN theory is a powerful tool for many applications operating on images, its realization on silicon is a challenging task. Therefore, designers have proposed some implementation-oriented simplifications, e.g., the full signal range (FSR) model [24], which truncates the cell state between two saturation values: -1 and 1. In this way, the cell state and cell output become equivalent. The employment of the FSR model has led to the realization of a CNN-UM with 128 $\times$ 128 cells [6].

2.3.3 Positive-Range High-Gain CNN

Positive-range high-gain CNN [25], [26] is another interesting simplification of the original CNN model. The output nonlinearity is modified to

$$y_{i,j}(t) = f(x_{i,j}(t)) = \begin{cases} 0, & x_{i,j}(t) < -\varepsilon \\ 1, & x_{i,j}(t) > \varepsilon \end{cases}, \quad 0 < \varepsilon \ll 1, \tag{2.4}$$

resulting in a step function. The cell input and output (I/O) become binary, while the template terms remain continuously valued. As a consequence of the analog design being simplified, yielding a more compact layout and improved reliability, gray-scale operations are no longer possible with binary I/O. Therefore, a part of the functionality of the network is lost.

2.3.4 Discrete-Time CNN

A Discrete-Time CNN (DT-CNN) [27] has continuously valued inputs and weights while its output is bipolar. It can be realized with either analog or digital implementation. In a digital DT-CNN the state can be updated through a variety of digital integration methods.

2.3.5 Binary CNN

The general purpose approach to CNN implementation, which combines the capability of processing both black and white (B/W) and gray-scale images, is easy to model with the CNN theory, but challenging to realize. The large number of in-cell multipliers extorts a large cell size. Moreover, the coefficient circuits are scaled for a precision required in analog computation, and thus their power consumption and speed are not optimized for processing binary images. An alternative approach is to separate the B/W and the gray-scale processing parts and optimize their implementations.

Binary CNN takes a B/W image as an input and yields a B/W result. The B/W data processing is an important class of CNN operations covering a wide range of applications [22]. In fact, most of the proposed templates handle mainly binary data. It is also an integral part of many algorithms based on sequential execution of templates and Boolean logic functions.

A CNN implementation dedicated for B/W image processing can benefit from the programmability of template terms reduced to binary values $\{0,1\}$. The coefficient circuits, thus cell structure, can be significantly simplified resulting in a very compact layout. Additionally, the power dissipation is significantly reduced. The limitation of the network functionality due to one-bit programmable template terms is not severe. More complex templates can be split into sequentially executed simple subtasks and the processor remains versatile. Moreover, since the weights are binary values, the reprogramming can be very fast and robust. Therefore, the overall performance can be competitive. An array with an extremely high spatial resolution can be implemented on a single chip for low-power, high frame-rate applications.

Chapter 3

Hardware Realizations of Array Processors

This chapter presents a selection of the state-of-the-art hardware realizations of array processors. These chips are described to broaden the view on this research field as well as to give a context in the discussion of motivation for the approach proposed later in this thesis.

3.1 ACE Series Processors

The appealing features of the CNN paradigm [1]-[3] have served as a driving force in forming an active research field of CNN hardware realizations. Despite years of development, the CNN hardware design remains challenging, and alternative approaches are considered. The series of ACE[1] processors [4]-[6] is an attempt to efficiently implement a network that would resemble the CNN model. The largest and the most advanced processor in this series, codenamed ACE16k, is a hardware realization of a CNN-UM with 128×128 cells. It combines image acquisition with processing capabilities. In addition, versatile programming makes ACE16k an attempt to obtain a vision system on a chip [6]. Each cell in the array incorporates a reconfigurable optical input module, in which the user can select among two types of photodiodes and a phototransistor. Moreover, either a normal linear integration or a logarithmic compression sensing mode can be used. In addition to an optical sensor, each processing element (PE) includes:

- a local analog memory (LAM) with a capacity for 8 gray-scale pixel values with an 8-bit resolution,

[1] ACE is the name of a family of CNN chips.

- a local logic unit (LLU) consisting of a programmable two-input one-output logic operator,
- a resistive grid module that allows for continuous-time diffusion in a resistive-grid like manner,
- an address event downloading module to simplify the information extraction from B/W images (instead of images, the addresses of active array locations are provided[2]),
- a configurable bank of analog multipliers and adders for interaction with other cells in the neighborhood,
- local masks for conditional execution of certain operations upon a locally defined value.

The analog core (array) is complemented by 128 A/D and 128 D/A data converters (one of each per array column), instructions memory, a memory for convolution masks and references with 24 digital-to-analog converters (DACs), and control circuits.

ACE16k is a general purpose SIMD-type processor that can operate on binary and gray-scale data. Despite the fact that only local cell interconnections are available, operations with a larger region of influence can be computed by means of the asynchronous information propagation.

3.2 Near-Sensor Image Processing

Another very interesting approach to realizing array-type image processors is the concept of near-sensor image processing (NSIP) [29]-[31]. Unlike the ACE chips, the NSIP implementation does not try to realize any predefined mathematical model. Instead, the functionality of a PE is based on functional and hardware-oriented requirements. Since a clear contradiction exists between the PE complexity and a high spatial resolution, a combination of a nondestructive photodiode readout and a binary image processing is proposed as a tradeoff solution. The proposed structure can perform local and global binary operations. Local operations include Boolean logic functions as well as interactions with the four nearest neighbors (towards principal compass points). Global operations (e.g., threshold, smallest circumscribing rectangle, selecting one of many objects) are handled by a global logic unit (GLU), which provides the asynchronous propagation of binary data over the array and thus working in parallel on the entire image. Due to a nondestructive sensor readout, the processor can also perform

[2]Event-based communication is becoming increasingly important. A recent example is the address-event representation (AER) protocol that has been developed for efficient communication in multichip neuromorphic systems [28].

a number of simple gray-scale operations by, for example, comparing in time the evolution of sensor outputs at different pixel locations. However, the results are always binary. Since the gray-scale values are coded as bit sequences in time, more complex gray-scale operations require an algorithm with extensive number of steps or even become impossible to perform. Additionally, a limited amount of in-cell memories restricts the array programmability, but NSIP can be considered as an image processor for certain simple applications.

3.3 SIMD Current-mode Analogue Matrix Processor

The SIMD Current-mode Analogue Matrix Processor (SCAMP) series is a very interesting example of realizing a general-purpose vision chip [32]-[34]. The implementation consists of an array of cells combining an image sensor with processing capabilities for analog operations. Each cell, called analog processing element (APE), contains a photodetector, a comparator with activity-flag latch, nine registers (eight general-purpose ones and one used for exchanging data with the nearest neighbors), and I/O circuits. The photodetector circuit uses an n-type diffusion diode and can operate in either linear integration mode or in continuous-time log-compression mode. The registers are designed using switched-current technique (compact memory cells). In this way, an additional dedicated circuitry such as arithmetic logic unit (ALU) is not needed to perform the arithmetic operations. Instead, the basic operations (inversion, addition, division, etc.) are executed in the registers and analog bus. Each register can store a gray-scale pixel value or a variable with 7 bits of accuracy. The local I/O register allows for data exchange with only one neighboring APE at a time. Moreover, connections are possible towards cardinal directions only. Therefore, examining the pixel values of multiple neighbors requires a time-domain multiplexing. As the computing speed is traded for silicon area, a very compact APE and arrays with reasonably high spatial resolutions can be achieved. However, operations relying on the asynchronous propagation of information throughout the array are no longer possible. To overcome this limitation, a new architecture for cellular image processor has recently been proposed, which allows for both synchronous and asynchronous operation modes [35]. Additionally, the SCAMP chip is equipped with simple and robust digital control/programming.

An interesting feature of SCAMP chips is the flexible global readout architecture [36] designed to enhance global communication and data extraction. The addressing is extended by the capability of having *don't care* bits in the address word. Therefore, a number of rows/columns in the array can be addressed simultaneously. This enables addressing groups of APEs in the array (block section of an image or periodic pixel-locations) in addition to single APE and entire array selections. Such a flexible

readout is useful in performing certain global operations. For instance in the binary readout mode, a logic OR can be computed on the selected group. In the analog readout mode, a global summation is performed. Moreover, this addressing scheme allows for calculating global image descriptors (pixel counts, histograms, etc.), object coordinate extraction, multiresolution image processing, and control of iterative procedures. It can be also used together with a global input, e.g., to replace a local value in selected APEs with a global one.

3.4 Digital SIMD Processor

An image processor performing a single instruction on multiple data can also be realized with digital circuits only. The implementation proposed in references [37] and [38] has an array of 64 $\times$ 64 processing elements, column adders as well as column and row shift registers. Each PE in the array occupies 67.4 $\times$ 67.4 μm^2 on a 0.35 μm CMOS process. The vision system is completed by an off-chip real-time controller and a DAC. Each PE consists of a photodetector with readout circuit, a bit-wise local RAM and a bit-serial ALU composed of a full adder combined with I/O multiplexers and carry register. The ALU has two binary inputs and executes a logical or arithmetic operation. Different types of image processing tasks can be performed as a sequence of such single-bit operations. The PE can also act as an analog-to-digital converter (ADC), which utilizes a software-controlled conversion of pixel illumination to digital pixel values. Additionally, each PE can communicate with its four neighbors at cardinal directions for spatial image processing.

Interestingly, by keeping the output latches transparent, adjacent PEs can be joined into blocks and treated as a single logic circuit. Each PE in such a block has two functions. It can be used as a pixel of an image or as a bit in a word, which provides effective memory use. The PE-joining feature enables the obtained blocks to execute global cumulative operations such as global summation and global OR, or multi-bit addition. This architecture can also emulate the column-parallel processor by chaining PEs in columns. Moreover, self-generated PE chains are possible based on the results from image processing.

3.5 Binary I/O CNN-UM

An alternative approach to the hardware realization of a CNN-UM has been proposed [25], in which the cells are based on the positive-range high-gain (PRHG) CNN model. A revised version of this design resulted in a standardized QCIF resolution (176 $\times$ 144 cells) CNN-UM for very low bit-rate video coding systems [39]. An algorithm for

such applications of CNN has been proposed in [40]. The observation that a dominant portion of this algorithm relies on manipulating B/W images explains the design focus on the B/W image processing part. Additional motivation comes from the fact that some of the required gray-scale operations are not implemented in any CNN hardware so far, and thus a separate design dedicated for such operations is assumed.
In this approach, each PE consists of:

- a comparator in form of a CMOS inverter to realize the cell output nonlinearity,
- synapse multipliers (fully-connected first-order neighborhood) based on a three-transistors structure (one for each positive and negative coefficient, plus a control switch),
- four local memories for storing intermediate results (the number is due to the algorithm of [40]),
- a memory controlling unit, which provides a cell-level buffer for global OR operation, memory I/O control as well as refreshing or inverting the memory content,
- programmable local logic structure,
- a cell I/O circuit with buffer and a number of control switches.

In addition to the QCIF-sized processing array, the chip contains a ring of constant border cells, DACs, a block of I/O buffers for driving the pads, row selection unit for addressing purposes, and a global OR evaluation block. While the PRHG CNN model limits the cell state to a single-bit value, the template coefficients remain analog with 6-bit programmability (5 magnitude bits and the sign) and the bias term is controlled with 9 bits. To increase the speed and reliability these analog values are generated by the on-chip DACs. The converters comprise a 6-bit coefficient memory (9 bits for bias term), binary weighted current sources, current steering switches, I/V converters for P- and N-type device in synapse, and buffers (over 25000 transistor gates load each voltage-line). The block of I/O buffers incorporate multiplexing of the 16-bit incoming data to a 176-bit wide image row for writing the initial values to the array.

Unlike the SCAMP or ACE16k chip, this implementation is not a general purpose image processor due to the lack of capabilities for gray-scale data processing. Neither, it belongs to the class of vision systems as the cell design does not incorporate any optical sensor. However, this limited functionality and novel design techniques allowed to increase the cell density to over 3000 cells/mm^2, a significant step towards arrays with a large spatial resolution.

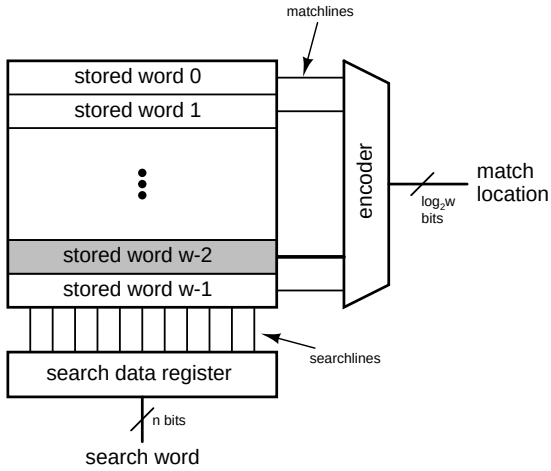


Figure 3.1 Conceptual view of a content-addressable memory containing w words. Shaded box indicates the location $(w-2)$ with data matching the search word [41].

3.6 Memory-based Approaches to Parallel Processing

This section presents two very interesting methods of exploiting the regular structure of a memory for implementation of parallel data processing – the content addressable memory and the computational RAM. Although, they are not directly related to the CNN paradigm, they offer properties (parallel search/write, bit-serial processing) that might be considered as useful extensions of the CNN functionality.

3.6.1 Content Addressable Memory

Content addressable memory (CAM) [41] can be regarded as a hardware search engine that can be much faster than algorithmic approaches for search-intensive applications. It is composed of a conventional memory (usually SRAM) with added comparison circuitry that enable performing a complete search operation in a single clock cycle, as shown in Figure 3.1. The search operation takes a data word as input and returns the address to a memory location containing the matching data. In case of multiple hits, the priority encoder selects the highest priority matching location.

The primary commercial application of CAMs is to classify and forward Internet protocol (IP) packets in network routers. However, if a CAM architecture is complemented with a parallel write function, it becomes a promising candidate for a compact hardware realization of a highly parallel image processing system with high data throughput. An example of such a processor is presented in [42]. The system consists of a chip controller and 32 CAM blocks, each with 512 words. The adjacent blocks are connected by an 8-bit horizontal bus and 1-bit vertical bus. Since these buses are also

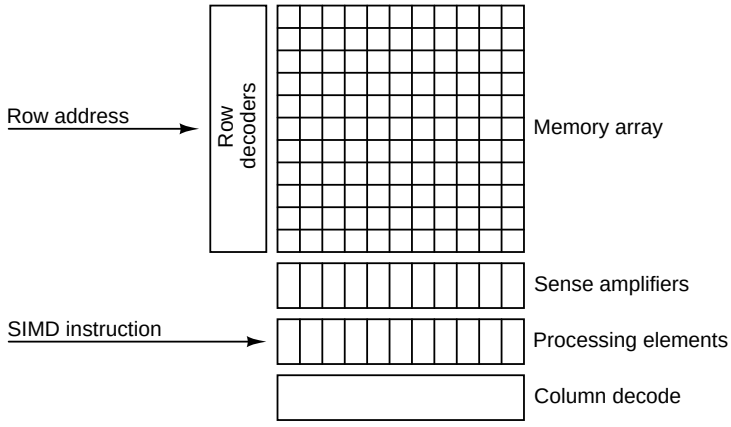


Figure 3.2 Integration of PEs within a basic DRAM structure [43].

distributed outside the chip, a larger CAM array can be obtained by using several chips. This processor can handle 128 $\times$ 128 pixels in parallel, and perform two-dimensional SIMD-type processing including global data operations. Various advanced functions such as bidirectional hit-flag shift, block read/write, and hit-flag counting are implemented in the chip.

3.6.2 Computational RAM

The computational RAM (C•RAM) architecture [43] is an attempt to implement processing elements within a memory chip. The motivation comes from the appealing idea of utilizing the very high internal bandwidth of a memory. This can be achieved only through a tight integration of logic and memory. For example, a pitch-matched PE can be placed at each column of a memory, as shown in Figure 3.2. In this way, the capabilities of a conventional DRAM chip are extended to the functionality of a SIMD-type processor with distributed, nonshared and uniformly addressed memory. It is worth noting that this extension can be obtained at the cost of only slightly increased area and power consumption.

Figure 3.3 presents the structure of a C•RAM processing element which supports bit-serial computation. It consists of an ALU based on an 8-to-1 multiplexer, and three registers. The ALU can perform a Boolean operation of three inputs coming from X and Y registers and memory. The result can be written back to either the memory or the X, Y or write-enable register. The X and Y registers can also act as destinations for left and right shift operations, providing the means for communication with neighboring PEs. The write-enable register is used for conditional writing to the memory. The bidirectional bus transceiver drives or receives from the broadcasting bus. During communication operations, the ALU is used to route signals.

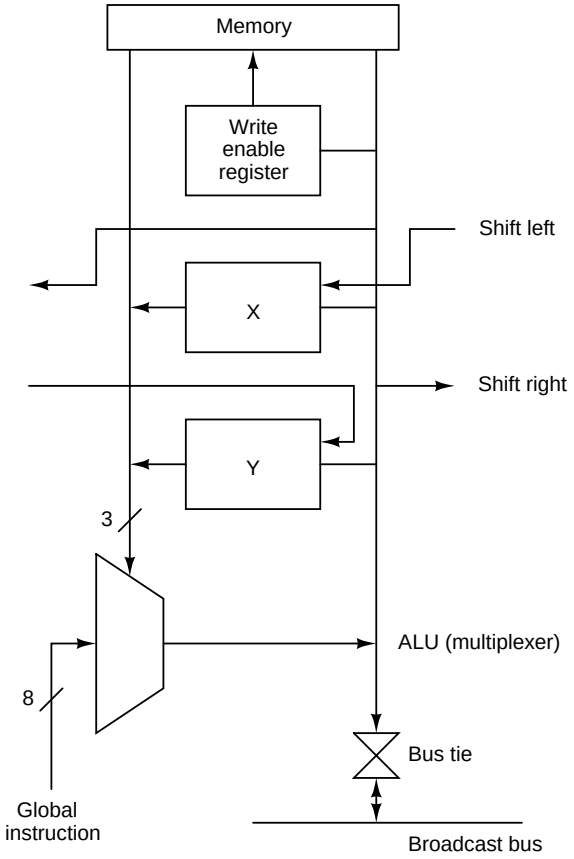


Figure 3.3 A simple processing unit for C•RAM implementation [43].

The C•RAM chips can be applied in fields of e.g., image processing, computer graphics, databases or even a real-time video-processing [44], [45]. However, this architecture does not allow for pixel-parallel computation since only one PE per memory column and not per memory cell (pixel) is available.

This page is intentionally left blank.

Chapter 4

Binary Programming Scheme

In this chapter, a new approach to control the function performed by a CNN – the binary programming scheme is described. With minimum hardware requirements, this scheme provides a processor with sufficient versatility to perform all black-and-white (B/W) operations proposed so far. The basic cell model is presented and template robustness issues are discussed. Based on the properties, the templates are classified into groups and the guidelines for template design are presented. The library of binary templates is given in Appendix A. This chapter is based on Reference [20]. However, the model cell structure described here differs somewhat from the version proposed in [20], as to keep a better correspondence with the implementations presented in the following chapters. Examples of template processing with images used in simulations as well as the analysis of memory usage in template evaluation can be found in [14], [20].

4.1 Motivation

The hardware realization of a cellular nonlinear network (CNN) can significantly benefit from the separation of B/W and gray-scale image processing parts as it allows for their optimization. In turn, the areal density, power consumption, and/or speed of operation can be improved. A great example of such optimization is the positive-range high-gain CNN dedicated for processing B/W data [25], [26]. In this model, the output nonlinearity takes the form of a threshold function resulting in binary cell output. However, the coupling weights remain analog, i.e. real numbers (positive and/or negative). It is a real challenge to perform fast programming of analog weights since their values need to be accurately settled before use. The coefficient circuits are seen as large capacitive loads attached to the weight lines, and thus slow down the programming due to RC delays. Reference [10] proposed that the multiplier coefficients are reduced to binary values as well. In this way, the coefficient circuits, and thus the cell structure,

can be further simplified, resulting in a very compact layout. That enables a single chip implementation of an array with extremely high spatial resolution. At the same time, a very fast reprogramming of the weights comes as a consequence of toggling the bit lines with minimum load capacitance.

4.2 Introduction

It is an interesting observation that most templates handling B/W images can be modified so that all of the terms are 1-bit values. The more-complex templates, which cannot be directly transformed into one binary template, need to be divided into a set of 2 or 3 subtasks (binary templates) run successively. Such a division can be done by, e.g. separating the positive template terms from the negative ones. The result of each subtask is either used as an initial state for the following one or it is stored in a local memory of a cell to be later combined with others (by means of Boolean logic operations) into the final result. In this programming scheme, also the threshold value is programmed digitally with 2 bits. Interestingly, in the binary programming scheme there is no need for a separate control matrix B and a feedback matrix A as in a classical CNN template. They are replaced with just one template matrix AB, which contains the elements of either A or B template matrix depending on the type of operation. Selecting the type of operation is done with a dedicated global control signal. Therefore, programming the template becomes as simple as loading 12 bits (9 for template terms, 2 for bias, and 1 for choosing the type of template). Obviously, toggling the digital lines can be fast (in the order of a few nanoseconds) and leads to competitive evaluation times of the multiple template algorithms.

4.3 Cell Model

4.3.1 Model Definitions

In the presented model, $x_{i,j}$ denotes the state of a cell in the i^{th} row and j^{th} column of an $M \times N$ array ($i \in [1,M]$, $j \in [1,N]$) and is determined as follows:

$$\begin{array}{rcll} x_{i,j}(t) & = & y_{i,j}(0) & \text{if} \quad e_{i,j} + M\_E = 2 \\ x_{i,j}(t) & = & \overline{y_{i,j}(0)} & \text{if} \quad e_{i,j} + IM\_E = 2 \\ C_X \cdot dx_{i,j}(t)/dt & = & \sum_{k,l}[AB_{k,l} \cdot yu_{i,j;k,l}(t)] - z & \text{else} \end{array} \tag{4.1}$$

where $k \in [-1,1]$, $l \in [-1,1]$, $y_{i,j}(0)$ is the initial cell output ($\overline{y_{i,j}(0)}$ is its inverse), $e_{i,j}$ is a transient-mask control-value, and C_X is a state capacitance. The variables M_E and IM_E control (enable) the normal and the inverted mask operation. They have

binary values and only one of them can be active at a time ($M\_E + IM\_E \leq 1$). For $(M\_E + IM\_E) \cdot e_{i,j} = 1$, the transient mask is active and the cell state is forced to the initial cell output $y_{i,j}(0)$ or its inverse $\overline{y_{i,j}(0)}$. Otherwise, the transient mask is inactive and the cell state evaluates. The inverted mask operations are not available when processing an A-template. The term $yu_{i,j;k,l}(t)$ is the neighborhood contribution defined as

$$\begin{aligned} yu_{i,j}(t) &= u_{i,j} \quad \text{if} \quad A\_B = 0 \\ yu_{i,j}(t) &= y_{i,j}(t) \quad \text{if} \quad A\_B = 1 \end{aligned} \tag{4.2}$$

where A_B is a binary value controlling the template type selection. For $A\_B = 1$ an A-template and for $A\_B = 0$ a B-template is being processed. The term $u_{i,j}$ is the cell input loaded before a B-template is selected with A_B. Upon A_B, terms $AB_{k,l}$ are elements of either a feedback matrix A or a control matrix B. The coefficients $AB_{k,l}$ can be independently programmed to be either 0 or 1. The negative bias term z defines the threshold of comparison and can be set to four different values: 0.5, 1.5, 2.5 and 3.5. It should be noticed that this bias programmability is sufficient, since any operation requiring threshold larger than 3.5 can be performed with inverted images. The cell model uses a unity step function as the output nonlinearity, and the relationship between the state and output takes the form

$$y_{i,j}(t) = f(x_{i,j}(t)) = \begin{cases} 0, & x_{i,j}(t) < V_m - \varepsilon \\ 1, & x_{i,j}(t) > V_m + \varepsilon \end{cases}, \; 0 < \varepsilon \ll 1 \tag{4.3}$$

where ε represents an arbitrarily small disturbance, and V_m is a comparator midpoint value. The introduction of V_m (for comparison, see Equation (2.4)) makes the model more universal and at the same time more accurate for hardware realizations, since usually $V_m \neq 0$ unless complementary supply voltages are used in the cell circuit.

4.3.2 Basic Cell Structure

The concept cell structure implementing the proposed model is sketched in Figure 4.1. It is just one of many possible realizations. As in most CNN hardware designs, cells interact with each other using currents. Such a technique is very suitable for a CMOS implementation. However, the cell model allows for alternative approaches that would fit best a specific technology and/or application.

The presented cell structure consists of a programmable negative bias, a state capacitance, and a pair of cascaded inverters working as a comparator and providing the desired output nonlinearity. The structure also includes local memories, 9 coefficient circuits and a number of switches partially grouped in a transient mask block. The switches in the figure are drawn in a position for unset control signal (logic LO). Additionally, it is assumed that nodes marked as x, x', e, and yu can be utilized as dynamic

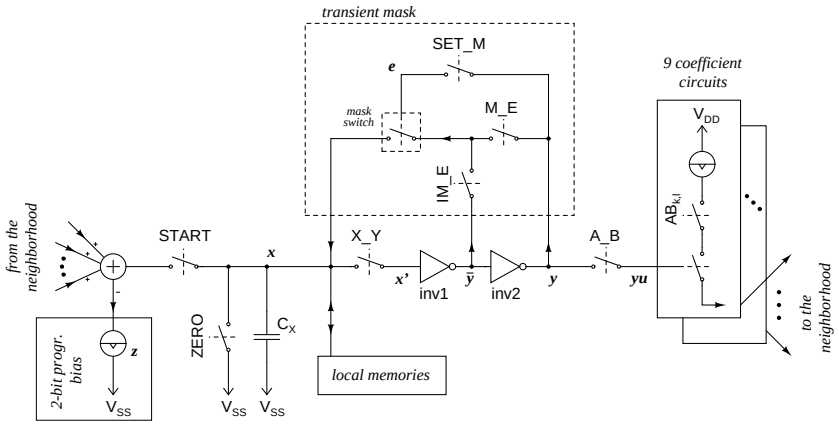


Figure 4.1 Cell structure implementing functionality required for binary programming scheme. For clarity, the I/O parts (image acquisition, data transfer) are omitted as implementation/application specific and irrelevant to the programmability model description.

memories. The values of x', e, and yu are stored using parasitic capacitances.

The signal *START* is used to initiate the template evaluation by connecting the neighborhood contribution and the negative bias to the cell input. The control signal *ZERO* sets the cell state to LO. The signal *SET_M* is used to program the transient mask by writing y to e. The control signal *X_Y* enables the nodes x' and y to follow the cell state x. For inverted mask operations this signal is unset. When the mask is active the cell state is forced to the value stored at the node x', i.e. $y(0)$ because the inverters *inv*1 and *inv*2 are strong enough to override the contribution from neighborhood and bias. All other signals reflect exactly the corresponding variables described in the previous section.

4.4 Template Robustness

In a hardware realization, manufacturing process variations may cause devices to have characteristics different from the desired ones. These imperfections can be classified into two groups: systematic and random. The influence of systematic differences, e.g. gradients and wafer-to-wafer variations, can be eliminated (or at least minimized) with proper circuit topologies, biasing and layout techniques [46]. Random variations exhibit themselves at a device level, and therefore are called ”device mismatch”. Their influence depends on the device size, thus limiting the minimum layout area. As the mismatch is random in nature, it is hard to predict accurately. Therefore, a computation result of a system is affected, and the probability of incorrect outcome is finite.

To account for the random mismatch and minimize the error probability, the designed system should be insensitive to a certain level of imperfections. Only two mea-

Bias value z	Minimum relative robustness $r_{rel,min}$
0.5	33.3%
1.5	14.3%
2.5	9.1%
3.5	6.7 %

Table 4.1 The values of the programmable bias and the corresponding minimum relative robustness.

sures of the system insensitivity, namely the absolute and the relative robustness are considered here as they are sufficient for comparison of the robustness of different templates. A thorough analysis of mismatch impact on the reliability of computation in binary CNN is described in [47]. The error probability measures described therein are particularly useful in the design process of very large arrays, as they allow to determine the optimal transistor sizes for requirements of a certain application.

The absolute robustness describes the minimum separation between the two logic states. In the presented binary CNN model it can be calculated as

$$r_{abs} = \min|D - z| = 0.5 \tag{4.4}$$

where D is the number of the neighborhood black pixels marked by "1" in the template matrix, i.e. $D = \sum_{k,l} yu_{i,j;k,l} \cdot AB_{k,l}$. For given template (and thus a bias value), the minimum separation between the logic states occurs when $D = z \pm 0.5$.
The relative robustness refers the absolute robustness r_{abs} to all contributions at the cell input, i.e. the sum of the positive and negative terms.

$$r_{rel} = \frac{r_{abs}}{D + z} \tag{4.5}$$

Therefore, in the presented model the minimum relative robustness given by

$$r_{rel,min} = \min\frac{0.5}{D+z} = \frac{0.5}{2 \cdot z + 0.5} \tag{4.6}$$

occurs when $D = z + 0.5$. The advantage of relative robustness is that it can be used to qualitatively compare the robustness of different templates. As can be seen, the template robustness depends on the value of the required bias z. Since all the operations can be performed with $z \leq 3.5$, the worst case in the binary programming scheme takes place for $z = 3.5$, which corresponds to $r_{rel,min} = 6.7\%$. The programmable bias values and the corresponding minimum relative robustness values are collected in Table 4.1.

The bias values of 0.5, 1.5, 2.5 and 3.5 seem a natural choice as the absolute robustness is maximized and equals 0.5 for both cases of $D = z + 0.5$ and $D = z - 0.5$. However, the corresponding values of r_{rel} differ. Reference [48] has recently suggested

that the bias values should be chosen with respect to the relative (and not the absolute) robustness, as the $r_{rel,min}$ determines the minimum device sizes. For example, the bias value $z = 1.5$ after the proposed optimization equals 1.41. In this way, a significant improvement in chip area and/or power consumption can be achieved.

4.5 Performing Local Logic Operations

The proposed cell can perform typical Boolean logic operations locally. In this case, the control signal *START* is inactive, and thus there is no information exchange between the cells. The cell model yields

$$\begin{aligned} Y_L(X(0),Y(0),E) &= [X(0) \wedge \overline{E}] \vee [Y(0) \wedge E] \quad \text{if} \quad M\_E = 1 \\ Y_{\overline{L}}(X(0),Y(0),E) &= [X(0) \wedge \overline{E}] \vee [\overline{Y(0)} \wedge E] \quad \text{if} \quad IM\_E = 1 \end{aligned} \tag{4.7}$$

where $X(0)$ and $Y(0)$ are matrices containing the initial state and the initial output, respectively. E is the transient mask matrix, $\wedge$ and $\vee$ denote logic AND and OR operations performed for matrices element-by-element. The bar over the matrix symbol denotes the inversion of matrix elements.

If $IN1$ and $IN2$ denote the operands, the basic logic operations can be obtained with

$$\begin{aligned} Y_{L,NOT} &= Y_{\overline{L}}(IN1, IN1, 1) \\ Y_{L,AND} &= Y_L(0, IN1, IN2) \\ Y_{L,OR} &= Y_L(IN1, IN2, IN2) \\ Y_{L,XOR} &= Y_{\overline{L}}(IN1, IN1, IN2) \\ Y_{L,NAND} &= Y_{\overline{L}}(1, IN1, IN2) \\ Y_{L,NOR} &= Y_{\overline{L}}(0, IN1, \overline{IN2}) \end{aligned} \tag{4.8}$$

Of course, the NAND and the NOR operations can also be obtained by inverting the results of the AND and the OR functions, respectively. However, computing NAND or NOR in the way presented in Equation (4.8) is potentially faster than inverting the results of AND or OR, respectively. In case of NAND, it is clear since no additional inversion is needed. For the NOR operation, it is not so obvious as the operand written to the mask needs to be inverted. However, if the NOR function is performed on the results of consecutive subtasks, there is a possibility that the inverse of one of the operands is already available. In that case, computing the NOR as described in Equation (4.8) can result in an algorithm with fewer steps.

The logic functions are also useful for algorithmic template evaluation. For instance, templates that conditionally turn pixels white or black can be performed using XOR and OR functions between the input image and the result of a subtemplate. The example of such a case is the Line Removal operation [22].

4.6 Processing B-Templates

In this section, the evaluation of different B-templates is presented. Based on the operation properties, three classes are identified: neighborhood threshold logic functions, operations with transient mask, and templates using pattern matching.

4.6.1 Neighborhood Threshold Logic

Assuming the mask is inactive, the cell model can be simplified to a neighborhood threshold logic gate described by

$$Y(AB,U,z) = \varphi(AB * U - z) \tag{4.9}$$

where U is the input matrix, AB is the template matrix, $*$ is the convolution operator, and z is the negative bias that determines the threshold of comparison. If the number of black neighborhood pixels marked by the nonzero template terms D is larger than the desired threshold T, the resulting pixel will be black (and otherwise white). As seen in Section 4.4, the larger the bias term the less robust is the operation. Therefore, if a large threshold T is desired, the entire operation should be performed on the inverted image. Namely, an inverted input $\overline{U}$ should be used instead of U, and the outcome of template evaluation should also be inverted for the final result. Also, the border condition should be changed. In this way, the value of the negative bias term z can be minimized. This procedure can be expressed as

$$Y(AB,U,z) = \begin{cases} \varphi(AB * U - (T + 0.5)) & T < Q/2 \\ \overline{\varphi(AB * \overline{U} - (Q - T - 0.5))} & T \geq Q/2 \end{cases} \tag{4.10}$$

where $Q = \sum_{k,l} AB_{k,l}$. In case the resulting pixel should be black when more than T neighborhood pixels are white, Equation (4.10) can still be used after replacing U with $\overline{U}$ ($\overline{\overline{U}} = U$). Examples of such operations are the Junction Extraction and Corner Detection [22].

Since in all threshold logic operations the in-cell feedforward term can be realized with the aid of transient mask (as explained in the next section), the center element of the template matrix $AB_{0,0}$ can be kept zero. As a result, the $Q \leq 8$ and $z \leq 3.5$ [1].

For the case of bias $z = 0.5$, Equation (4.9) describes a multiple input OR gate. In this way, one can process B-templates that determine the output according to one of the following neighborhood conditions of a pixel in the image data IN:

[1] This justifies that the bias programmable with two bits (to either 0.5, 1.5, 2.5, or 3.5) is sufficient for all operations.

- There is at least one black pixel in the neighborhood that is marked by a nonzero entry in the template matrix AB. Such locations are represented by black pixels in $Y_{B>0} = Y(AB, IN, 0.5)$. The Object Increase and Dilation are examples of such templates [22].

- There is at least one white pixel in the neighborhood that is marked by a nonzero template term. In this case, an inverted image $\overline{IN}$ is applied as a network input, and black pixels in $Y_{B>0} = Y(AB, \overline{IN}, 0.5)$ indicate such locations.

Another two conditions can be determined by inverting the results of the above operations, so that

- Black pixels in $Y_{B=0} = \overline{Y_{B>0}}$ indicate that there were no black neighborhood pixels marked by a nonzero entry in AB.

- Black pixels in $Y_{W=0} = \overline{Y_{W>0}}$ show the locations where no white neighborhood pixels were marked by a nonzero template term. This type of operations include, e.g., Erosion and Peel templates [22].

4.6.2 Processing with the Transient Mask

The transient mask is also useful in processing templates that perform conditional neighborhood logic operations. For instance, it might be desired that only black (or only white) pixels of an image will able to change their state during the processing.

The use of the noninverting transient mask ($M\_E = 1$) during a B-template evaluation allows the cell model to be written as

$$Y_e(AB, U, Y(0), E, z) = [\varphi(AB * U - z) \wedge \overline{E}] \vee [Y(0) \wedge E] \tag{4.11}$$

Data stored in $Y(0)$ determines whether the pixels with active transient mask are black or white. Here, the $Y(0) = 0$ yields a white pixel. For instance, the Point Removal and Edge Detection templates belong to this group [22].

When the inverting transient mask is used ($IM\_E = 1$), the cell model takes the form of

$$Y_e(AB, U, Y(0), E, z) = [\varphi(AB * U - z) \wedge \overline{E}] \vee [\overline{Y(0)} \wedge E] \tag{4.12}$$

As can be observed, the only difference between Equations (4.11) and (4.12) is the use of $\overline{Y(0)}$ instead of $Y(0)$, i.e. an opposite action takes place for the pixels with active mask. In this case, the $Y(0) = 0$ yields a black pixel. The Point Extraction template represents such operations [22].

4.6.3 Templates Using Pattern Matching

A number of B-templates rely on pattern matching only. These operations can be processed with the minimum bias value $z = 0.5$. In general, such templates check if the pixels at certain neighborhood locations are black as required. Then, another set of pixels in the neighborhood (which should be white) are tested. The outcomes of both operations are combined into a final result. This procedure can conveniently be described by the General Pattern Matching template [22], which can be expressed as

$$\begin{aligned} match &= \overline{Y(AB^{(b)}, \overline{IN}, 0.5) \wedge Y(AB^{(w)}, IN, 0.5)} \\ &= \overline{Y(AB^{(w)}, IN, 0, \varphi(AB^{(b)} * \overline{IN} - 0.5), 0.5)} \end{aligned} \tag{4.13}$$

where nonzero elements of $AB^{(b)}$ indicate neighborhood locations at which the pixels should be black, and nonzero elements of $AB^{(w)}$ point to locations for which the white pixels are required. To clarify, it should be mentioned that these two tests do not need to cover the entire neighborhood. In particular operation, states of some pixels may be indifferent. The examples of templates using pattern matching include: Diagonal Detection, Right Edge Detection, Local Concave Place Detection, and Skeleton [22].

4.7 Processing A-Templates

The binary-programmable CNN can process the A-templates that rely on a propagating information wave or a threshold logic. Additionally, the cell allows for the algorithmic evaluation of A-templates and logic operations.

4.7.1 Computing with the Propagating Wave

The binary-programmable CNN is well suited for evaluation of A-templates in which the propagating wave changes the pixel values unidirectionally, e.g., from white to black. Although the operation is continuous in time, it is convenient to describe the process with an iterative (i.e. discrete time) expression. With $A\_B = 1$ an A-type operation is selected, i.e. AB contains the A-template coefficients. Assuming that a noninverting mask is in use ($M\_E = 1$) the following model is obtained.

$$Y_{t+1}(AB, Y(0), E, z) = [\varphi(AB * Y_t - z) \wedge \overline{E}] \vee [Y(0) \wedge E] \tag{4.14}$$

As in the case of B-templates, the transient mask can be used to disable selected pixels from changing their state during the processing, while other pixels evaluate. Additionally, the mask enables operations with two input images. In that case, one image (or its inverse) is written to the mask E and the other image (or its inverse) is

used as an initial state.

The propagating black wave can be initiated from either an object in the input image (marker), as in the case of Selected Object Extraction [22], or from the network border (Hole Filler [22]) and stopped by the object contour.

A-templates can also use threshold logic (e.g., Concave Location Filler [22]) and be combined with local logic operations (e.g., Global Connectivity Detection [22]).

4.7.2 A-Templates with Positive and Negative Coefficients

The proposed programming scheme does not allow for asynchronous processing of the A-templates that contain both positive and negative coefficients, such as Concentric Contour Detection or Connected Component Detection [22]. However, these operations can be performed as a series of B-templates, following the guidelines of Section 4.6. Although in this way, the speed of operation gets lower, the robustness is increased. Moreover, this can be done without an excessive usage of memory for intermediate results [14], [20].

4.8 Conclusions

The guidelines presented in this chapter can be used to design a binary template for any B/W image operation. Additionally, it has been shown that the binary programming scheme provides the cell with versatility sufficient for all B/W image manipulations typical of CNN applications. With the coefficient weights and the cell state reduced to 1-bit values, the requirements for coefficient circuits are significantly loosened. The implications of this programming scheme on the hardware realization of a CNN are discussed in the following chapters.

Chapter 5

CMOS Implementation of Binary CNN

This chapter is devoted to development of the binary CNN implementation with CMOS technology. The presented array processor is dedicated for operations performed on B/W images, especially, in case where very fast processing and very limited power consumption are required. In addition to a single-bit representation of an image pixel, the coefficient weights are also one-bit values. The bias value, which defines the threshold, can be set to either 0.5 or 1.5. Except for the 1-bit bias programmability, this chip implementation follows the guidelines set by the binary programming scheme presented in Chapter 4. The main objective of this chapter is to demonstrate the realizability and measured performance of the proof-of-concept test structure.

Programming analog weights can easily take hundreds of microseconds due to the accuracy requirements. In the time it takes to program analog weights, the proposed binary-programmable CNN can be reprogrammed a couple of times and a few operations can be performed. Additionally, the binary-valued weights of the cells' interconnections lead to a very compact structure of the coefficient circuits, and thus a very dense layout is obtained. Also, the aspect of power consumption, very important in array computing, is addressed. The proposed structure can operate with a very low supply voltage limiting the currents in the circuit. A peak dissipation in the range of microwatts per cell and the small cell-size open the door for implementation of an array with extremely high spatial resolution suitable for high-frame-rate applications.

This chapter is organized in the following manner. Section 5.1 describes the architecture of the proposed CNN hardware realization. The silicon implementation issues are covered in Section 5.2. A number of operations are described and visualized by the measurement results in Section 5.3. Discussion regarding the chip performance is

given in Section 5.4.

5.1 System Architecture

Figure 5.1 presents a block diagram of the chip consisting of a 4×4 array of processing elements (PEs) surrounded by border cells. The cells are organized in a rectangular grid and they are 8-connected with their 1-neighborhood. In addition, a shift register used for programming the template terms was placed as a means to minimize the required number of pins. That is because two other test structures were implemented along with this one on the same chip. Only 11 bits (9 for the coefficients, 1 for the bias term, and 1 for the control signal A_B to select the type of template) are needed for the template programing. The supply voltage of the PEs, V_{DD1}, is kept at least one NMOS threshold voltage lower than the high level of the global control signals. This way the NMOS switches can convey full logic levels. The supply voltage of shift register, V_{DD2}, is the same as the logic high level (HI) of the global signals.

5.1.1 Cell Model

The cells in the presented approach use a modified version of the positive-range high-gain output nonlinearity resulting in an inverting threshold function (see Figure 5.2(a)). This nonlinearity can be mathematically expressed as:

$$V_Y = f(V_X) = \begin{cases} V_{DD1}, & 0 \leq V_X < V_{th} \\ V_{SS}, & V_X \geq V_{th} \end{cases} \tag{5.1}$$

where V_Y is the cell output, V_X stands for the cell state, V_{th} represents the comparator threshold, and V_{SS} is the ground potential.

With the modification shown in Figure 5.2(b), the state equation can be written in the form of

$$C_X \cdot \frac{d(\sim V_X)}{dt} = \sum_{k,l \in N(i,j)} AB_{k,l} \cdot V_{Y_{k,l}} - z = D - z \tag{5.2}$$

where C_X is the state capacitance, $N(i,j)$ is the first neighborhood of a cell in the i^{th} row and j^{th} column, $AB_{k,l}$ are the coefficients i.e., the elements of either A or B template matrix (depending on the global signal A_B), z is the threshold set by the positive bias, and D is the number of black neighborhood pixels (cells with low logic level at V_X) marked by the template.

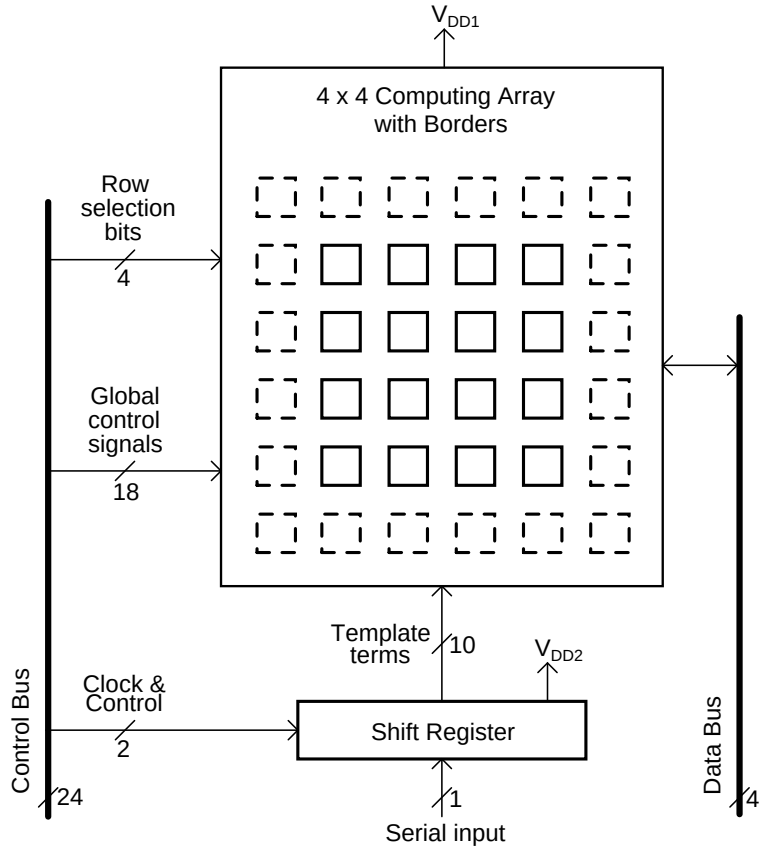
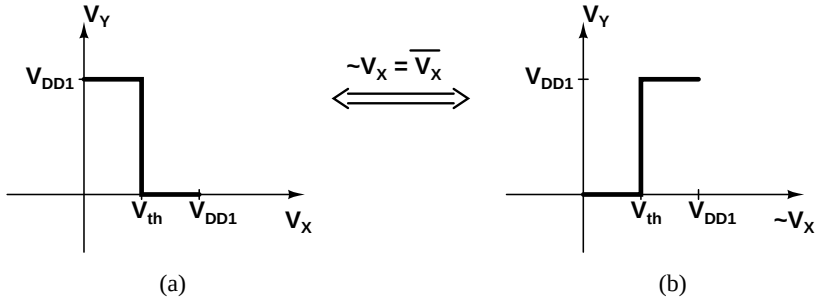


Figure 5.1 Block diagram of the chip architecture.



(a) (b)

Figure 5.2 Cell output nonlinearity.

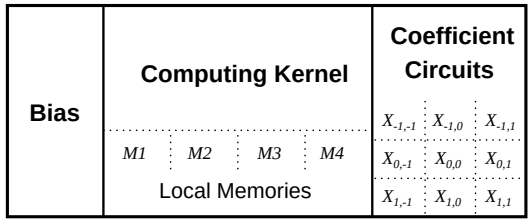


Figure 5.3 Block diagram of the processing element structure.

5.1.2 Cell Design

Figure 5.3 presents a block diagram of a cell. Three main building blocks can be distinguished in the cell. Namely, the bias circuit, the cell's computing kernel with four digital local memories, and nine coefficient circuits. This CMOS implementation of a cell is very similar to the basic cell structure proposed in Section 4.3.2. However, a number of small differences exist. Mainly, the bias is 1-bit programmable and it is implemented with a pull-up structure, while the coefficient circuits are of pull-down type. Due to these, a black pixel is represented by a low voltage at the state capacitance. The cell has three basic operating modes:

- a multi-input threshold logic gate (TLG) with programmable inputs
- a multi-input threshold logic gate (TLG) with programmable inputs and with a fixed state map
- a local logic device (transient mask operations).

The first two modes are used for template evaluations. The cell can operate either in discrete time (B-templates) or in a way that allows asynchronous propagation (A-templates). The local logic device mode is used for computing Boolean functions without the neighborhood interactions.

5.1.2.1 Computing Kernel

Figure 5.4 shows the detailed structure of the cell's computing kernel. It comprises the state capacitance C_X implemented with a 3 μm $\times$ 3 μm NMOS, two inverters $INV1$ and $INV2$, a transient mask based on the programmable transmission gate, four SRAMs grouped in the local memories block, and a number of switches. Additionally, four parasitic gate capacitances C_S, C_C, C_m, and $C_{\overline{m}}$ utilized as dynamic memories are shown with the small dashed-line symbols. Since these nodes store temporary signal values for a very short time, no refresh cycles typical for DRAM implementations are needed.

Most of the control signals are named the same as in Section 4.3.2, and the cell operates as follows. In the multi-input TLG mode, the $START$ switch is used for con-

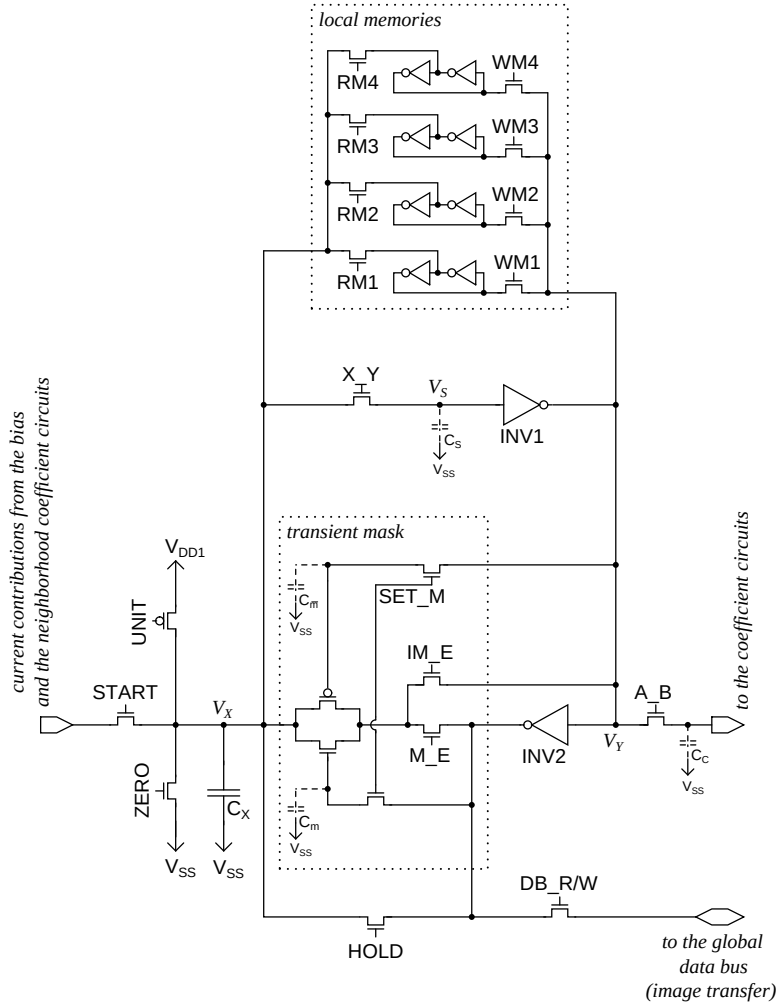


Figure 5.4 Schematic of the computing kernel of a cell.

necting the neighborhood outputs and the bias to the cell input, thus initializing the processing. The input currents set the voltage at the state capacitance C_X. The voltage V_X is thresholded by $INV1$, which provides the output voltage V_Y. The cell state can be latched in the SRAM-like loop formed by the inverters $INV1$ and $INV2$, and the switches $HOLD$ and X_Y. Control signal A_B determines whether an A- or a B-template is run. When it is kept HI, the cell output contribution is enabled to change during the processing, and therefore an A-template is evaluated. For a B-template this signal is active only for a short while in order to write the V_Y into the parasitic gate capacitance of the coefficient circuits, C_C. Transistor DB_R/W is used to enable the image data transfer to and from a data bus. The control signals $UNIT$ and $ZERO$ are used to preset the cell state V_X to either V_{DD1} or V_{SS}, respectively. Although only one switch (either $UNIT$ or $ZERO$) would be sufficient, both are implemented as it boosts the performance (inversion is a rather complex operation with this kernel structure – see Section 5.3.1) at the minimum area penalty. The transient mask is used for implementing a fixed state map. It can be programmed by writing the desired values via the SET_M switches into capacitors C_m and $C_{\overline{m}}$. When the mask is inactive (transmission gate is not conducting), the cell state V_X evaluates according to Equation (5.2). When the mask is active (transmission gate is conducting), V_X is forced to either V_S or its inverse $\overline{V_S}$ depending on whether the M_E or the IM_E signal is set HI. Such enforcement is possible because the transistor controlled by the $START$ signal has a small W/L ratio. The mask is also used for performing Boolean logic operations by means of conditional writing to the cell state. When the cell operates in the local logic device mode, the signals $START$ and A_B are kept LO, thus there is no interaction between the cells.

5.1.2.2 Coefficient Circuits

In the presented approach the coefficient circuits, shown in Figure 5.5, are implemented with pull-down circuits. The output of $INV1$ controls the gate of the analog transistor (shown using a larger symbol than the template term-controlled switch) in each coefficient circuit. Assuming the corresponding template term ($AB_{k,l}$) to be "1", the coupling becomes active when the cell output is HI i.e., the cell state V_X is LO. Such an activated coefficient circuit works as an unit current sink (with the analog transistor serving as an unit current source) at the input of the corresponding neighbor. Since a low supply voltage is used, the gate of the NMOS current source can be driven to V_{DD1}. In this way, the unit current is determined by the supply voltage of the cell. Therefore, the tradeoff between speed and power consumption can be controlled with the value of V_{DD1}. When a B-template is run, the parasitic gate capacitances of the analog transistors maintain the desired control value (either zero or V_{DD1} volts). Since the processing

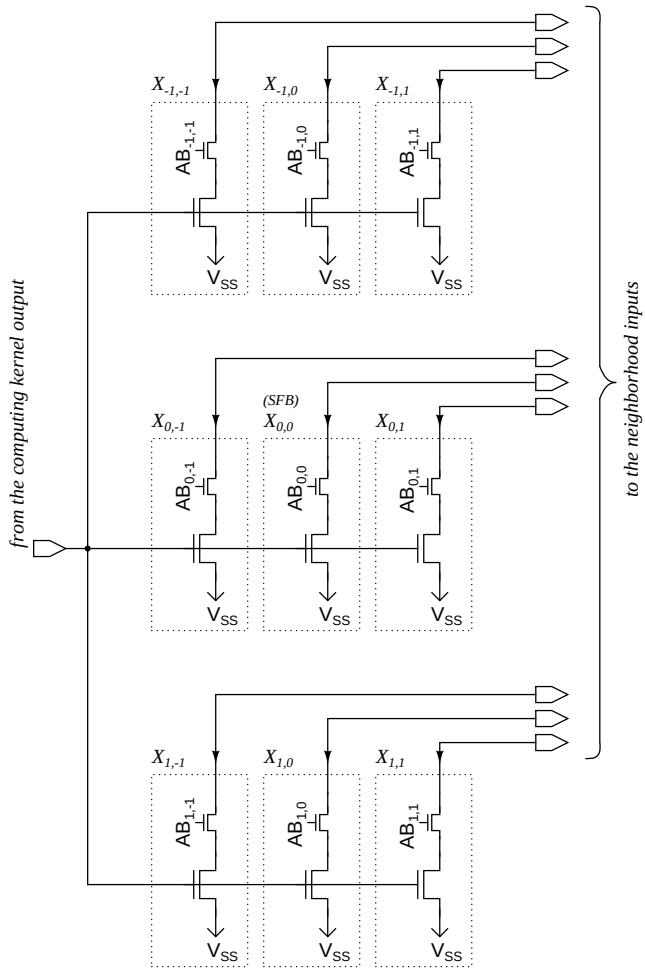


Figure 5.5 Schematic of the coefficient circuits.

is fast (see Section 5.3.3), the gate parasitics do not need to hold the voltage for a long time. Therefore, the transistor leakage is not a critical issue.

5.1.2.3 Bias Circuit

The pull-up bias circuit, shown in Figure 5.6, comprises a simple current mirror with a scaled replication of the current defined by an NMOS current source (that is identical as in the coefficient circuits). The total bias current is either 0.5 or 1.5 times the unit current depending on the global signal *BIASBIT*. The bias programmability could easily be extended to two bits, by placing another (scaled $\times$ 2) current replication path. The power consumption of a cell after settling is upper bounded by the bias current.

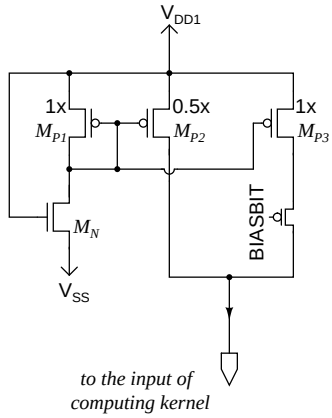


Figure 5.6 Schematic of the bias circuit.

5.1.3 Border Cells

Each cell in the border ring surrounding the computing array has a structure identical to the coefficient circuit. A border cell switch is controlled by a suitable template term (e.g., $AB_{0,-1}$ in left border), and the analog transistor is driven by the global *BORDER* signal. For black border conditions this signal is set to HI, and for white border it is set to LO.

5.2 Silicon Implementation

5.2.1 Robustness

When implementing an array processor, mismatch caused by processing variations has to be taken into account. To estimate the probability of an error due to mismatch, the cell was Monte Carlo-simulated with 1000 iterations. The mismatch parameters and models provided by the foundry were used within the Eldo simulator. Transistors with the following sizes (given in micrometers) were used: NMOS transistors with $W/L = 0.5/0.5$ (for both the analog transistors in the coefficient circuits and M_N in the bias circuit), and PMOS transistors with $W/L = 2.0/0.3$ (for M_{P1}), $W/L = 2.17/0.28$ (for M_{P3}), and $W/L = 2.17/0.56$ (for M_{P2}). Since the minimum relative robustness occurs when $D = z + 0.5$ with bias z at its maximum value, the bias term set to 1.5 and two black neighbors can be considered as the worst case here. Table 5.1 presents the percentage of incorrect cell states at given bias and neighborhood vs. the supply voltage. With the bias programmed to 0.5, failure-free results were obtained at each supply voltage value and neighborhood condition. For the case of the bias programmed to 1.5, failure-free results were obtained with a supply voltage $V_{DD1} \geq 0.8$ V. With

Bias and no. of black neighbors	V_{DD1} [V] 0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2
$z = 1.5, D = 1$	0.8%	0	0	0	0	0	0	0
$z = 1.5, D = 2$	21.1%	11.3%	1.1%	0	0	0	0	0

Table 5.1 Simulated Failure State Percentage vs. Supply Voltage

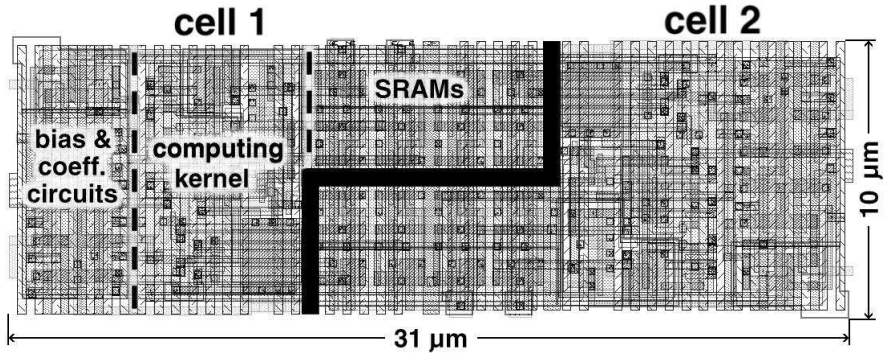


Figure 5.7 Layout of two cells with marked regions occupied by each of the PE's components.

scaling the supply voltage down the failure percentage increases. As can be deducted, the increase of supply voltage (if programmed differently for different templates) can be used as a means to increase the robustness of a certain template evaluation.

5.2.2 Layout

For maximal density the cells are grouped into pairs of partially overlapping cells, which share the global control signals for their (identical) SRAM subcircuits. It is important to mention here that the cell array does not require any additional spacing (e.g. for signal wires) between those pairs. In fact, they also slightly overlap each other thus reducing the array area. Figure 5.7 presents the layout of two connected cells designed for a standard digital 0.18 μm CMOS process from ST Microelectronics, which was used for fabrication. The process has a single poly layer, while six metal layers are available for routing. In this design only four metal layers were used, thus there is room for an improved distribution of the global signals and supply voltage or a shielding layer, if a large network is to be built. The area of such a two-cell pair is 310 μm^2. The entire 4×4 array occupies 62.2 μm $\times$ 38.5 μm without the border cells, and 67 μm $\times$ 46 μm with the border ring. The layout was designed in a full-custom manner.

5.3 Measurements

The chip measurements were conducted with a dedicated test board comprising external buffers for a bidirectional data bus. All of the control signals as well as the image data were provided by a pattern generator with 8ns minimum pulse width. The images resulting from processing were read out from the chip and the cell states were determined by a logic analyzer. The presented array has been provided with separate supply rails. A microampere-meter in series with the power supply source was used for the measurements of the power consumption.

5.3.1 Logic Operations: NOT, XOR, NAND, NOR

The logic operations (Boolean combinations) can be computed by a CNN-UM either with the use of templates or by a local logic unit (LLU) placed within each PE as in [6]. In the presented approach the logic operations do not use templates, neither is there a dedicated LLU. Instead, a sequence of global control signals makes use of the programmable transmission gate for computing these functions as explained in Section 4.5.

Figure 5.8 shows the simplified equivalent cell structures for different logic operations. For the sake of clarity, the complementary control of the transmission gate's PMOS (values stored on $C_{\overline{m}}$) was omitted as obvious. In the XOR case, the computing kernel performs a conditional inversion of one operand if the other operand is in a logic HI state. Analogically, the NAND function means conditional inversion of one operand when the other is HI or otherwise the cell state is set HI. NOR is done by inverting one operand when the other is LO or otherwise the cell state is set LO.

In the presented approach, the NOT operation is used in many algorithms. Though it is a trivial function, its evaluation with the proposed cell structure is not so straightforward. Therefore, the detailed NOT algorithm is given here to show how this logic operation translates into a sequence of control signals. The inversion is done with the aid of the transient mask programmed to conduct.

1. Set the cell state HI: $HOLD \searrow$, $UNIT \searrow\nearrow$
2. Program the transient mask: $SET\_M \nearrow\searrow$
3. Read-out the operand from memory: $rd1 \nearrow\searrow$
4. Separate C_X from C_S: $X\_Y \searrow$
5. Perform the inversion (force V_X to $\overline{V_S}$): $IM\_E \nearrow \ldots \searrow$
6. Charge sharing between C_X and C_S: $X\_Y \nearrow$

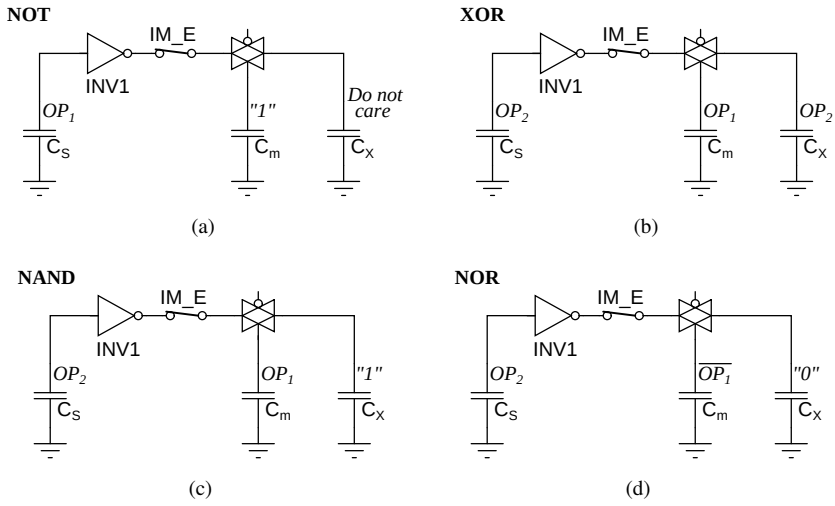


Figure 5.8 The equivalent cell structures for different logic functions: a) NOT, b) XOR, c) NAND, and d) NOR.

Since $C_X/C_S \approx 5$, the voltage V_S at the input of $INV1$ will become close to V_X. Then $HOLD \nearrow$ and the result is latched in the cell ready to be read out or to be used in further calculation. The evaluation of NOT becomes straightforward if the transient mask can be bypassed with an additional switch as proposed in [14].

The signal sequences for the two-operand logic functions are designed in a similar way. In the XOR case the first operand instead of unity is used to program the mask. A similar sequence is required for a NAND function. The only difference lies in the need to initialize the cell to HI ($UNIT \searrow\nearrow$) before starting the inversion with $IM\_E \nearrow$. That will ensure the HI state of the cells where no inversion takes place i.e., the first operand is LO. Only a bit more complicated algorithm is needed to compute the NOR function. Namely, the first operand needs to be inverted before being written to the transient mask to ensure the inversion takes place for its LO state (i.e., the NOR algorithm begins with computing the NOT). Also, the cell state needs to be set LO ($ZERO \nearrow\searrow$) before the signal $IM\_E \nearrow$. Figure 5.9 shows the operands and the measured results of the two-operand logic functions. In this implementation black pixels correspond to logical ”0” and are represented by a LO cell state (voltage at V_X), and white pixels correspond to logical ”1” and are represented by a HI cell state.

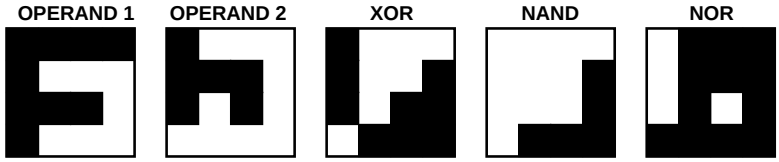


Figure 5.9 The operands and the measured results of the logic operations.

Supply voltage	Measured propagation time per cell	Measured 0.5 bias current per cell
1.2 V	4.0 ns	6.58 μA
1.1 V	4.7 ns	5.19 μA
1.0 V	5.3 ns	3.92 μA
0.9 V	6.3 ns	2.78 μA
0.8 V	9.0 ns	1.07 μA
0.7 V	16.3 ns	0.68 μA
0.6 V	41.7 ns	0.35 μA
0.55 V	78.3 ns	0.34 μA

Table 5.2 Measured Wave Propagation Speed and 0.5 Bias Current vs. Supply Voltage

5.3.2 Processing of Templates

5.3.2.1 Wave Propagating A-Templates

The first examined template is the Shadow into South-West (SW) direction, which takes the form of

$$AB = A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = 0.5 \tag{5.3}$$

Each cell checks the state of the neighbor specified by the non-zero template term. If the state is LO (black pixel) the cell also turns black. The center element of a template matrix is non-zero to assure that pixels without the required black neighbor remain black. However in this particular case, the same effect could be obtained with the aid of the transient mask.

The Shadow template gives an opportunity to measure the speed of a wave propagating throughout the network. The measurements were conducted with a supply voltage V_{DD1} varying from 0.55 to 1.2 volts. At each V_{DD1} value, the minimum time required for the propagation of a black pixel from one cell through the rest of the array (3 cells) was measured. Depending on the supply voltage, a wave needs from 4 ns at 1.2 V to 78.3 ns at 0.55 V to propagate a distance of one cell (see Table 5.2). The table also shows the measured 0.5 bias currents at the given voltages. The Shadow template evaluations into the SW and SE directions are shown in Figure 5.10.

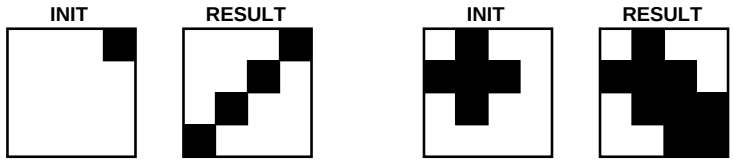


Figure 5.10 Evaluation of the Shadow template: a) into SW direction, b) into SE direction.

5.3.2.2 A-Templates With a Fixed State Map

The Hole Filler is another tested template. It relies on a black wave propagating from the image borders that is stopped by the object contours [30]. It has the form of

$$AB = A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad z = 0.5 \tag{5.4}$$

The Hole Filler causes the white areas surrounded by black pixels to turn black. Therefore, the white holes within black objects are being filled. Since the operation is performed on an inverted image, the algorithm is more complex than for Shadow, and thus it is given here along with the corresponding sequence of control signals.

1. Load an image: DB_R/W $\nearrow\searrow$
2. Invert the image: *NOT operation as depicted in Section 5.3.1*
3. Set the transient mask condition to prevent the cells corresponding to the black pixels of an image from changing their state during the processing: SET_M $\nearrow\searrow$
4. Initialize the network to contain white pixels only: $HOLD$ $\searrow$, $UNIT$ $\searrow\nearrow$
5. Separate C_X from C_S: X_Y $\searrow$
6. Upon the mask condition force V_X to V_S: M_E $\nearrow$
7. Select the template type: A_B $\nearrow$
8. Set the black border condition (this can be done concurrently with the previous step): $BORDER$ $\nearrow$
9. Perform the template evaluation (black wave propagates from the borders throughout the network, pixels with active transient mask stay white and block the further propagation): $START$ $\nearrow$ ... $\searrow$
10. Invert the result: *NOT operation*

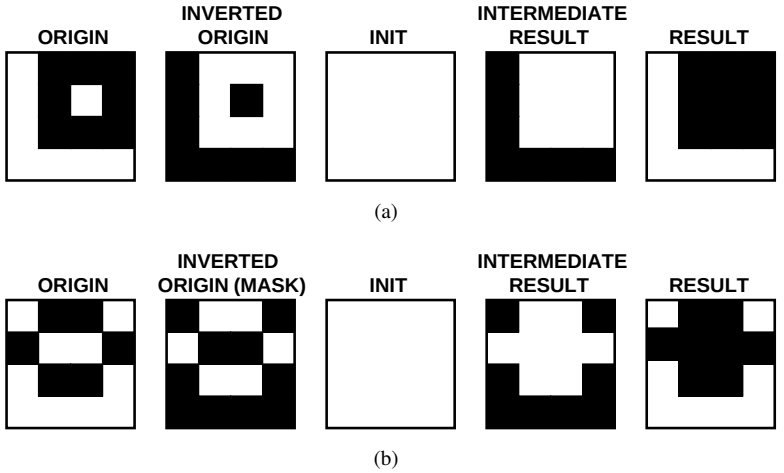


Figure 5.11 Two examples of the Hole Filler template evaluation.

After the algorithm evaluation is completed the result can be read out from the cell or used in further processing. Figure 5.11 presents the original image and the measured result of Hole Filler evaluation.

The Figure Reconstruction template (also known as Selected Object Extraction) extracts from the image the objects marked by the black pixels in the marker image. In other words, the marked objects are preserved and appear in the result, while all the other objects are being erased. This template relies on similar operating principles as the Hole Filler. With the original image written to the mask and the marker as the initial image the template of Equation (5.5) is evaluated. In this way, the marker initiates the wave propagation that is stopped by the object contour. The measured examples are shown in Figure 5.12.

$$AB = A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad z = 0.5 \tag{5.5}$$

5.3.2.3 B-Templates

With the proposed implementation, every discrete time CNN (DT-CNN) template [22] is evaluated as a B-template. For test measurements, the Object Increase template was chosen, which is expressed as

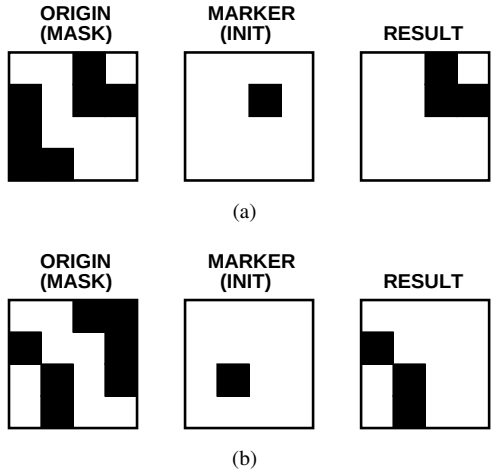


Figure 5.12 Evaluation of the Figure Reconstruction template.

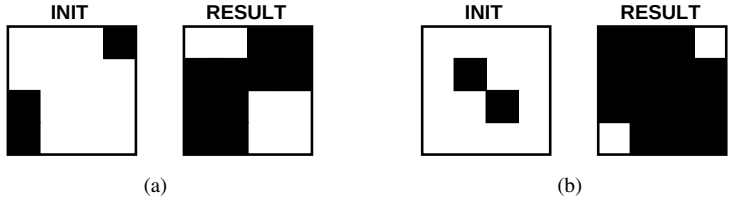


Figure 5.13 Examples of the Object Increase template evaluation.

$$AB = B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad z = 0.5 \tag{5.6}$$

This simple operation causes black objects in the image to grow by one pixel into every direction. As a consequence, every white pixel that has at least one black neighbor will also turn black. Of course, the object growth into only a selected direction can be easily obtained through a slight modification of the template matrix. For instance, if the growth is desired only on the right side of the object, all but $AB_{0,-1}$ and $AB_{0,0}$ terms should be changed to zeros.

To perform a B-template evaluation, signal A_B is set back to LO before the processing is initiated with signal $START \nearrow$. In this way, the asynchronous wave propagation in the network is disabled and stepped (discrete time) operation is obtained. Figure 5.13 shows the initial image and the measured result.

Operation	Execution Time
Write to memory	≤ 8 ns
Read from memory	≤ 8 ns
Write the transient mask	≤ 8 ns
Load image from data bus	61 ns per row
NOT	≤ 60 ns
XOR	≤ 80 ns
NAND	≤ 96 ns
NOR	≤ 160 ns
B-template	11 ns
A-template	4 ns per cell

Table 5.3 Execution Time of Basic Operations at $V_{DD1} = 1.2$ V

5.3.3 Speed of Operation

Due to the output frequency limitation of the pattern generator, the ultimate speed of some operations could not be determined. The measurements indicated that writing to and reading from a local memory, writing the transient mask as well as all the operations containing these tasks could be performed faster. Therefore, their execution times are marked in Table 5.3 as "less or equal to" the minimum measured value.

5.3.4 Power Consumption

The dynamic power consumption was measured as a current dragged from the power supply during looped evaluation of the B-template:

$$AB = B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad z = 0.5 \tag{5.7}$$

The borders were set black, the template terms were programmed, and the initial image, shown in Figure 5.14, was stored in one of the local memories. Then, the loop consisting of the following operations was executed at the speed of $25 \cdot 10^6$ cycles per second:

1. read out the initial image from memory,
2. run the template of Equation (5.7).

This loop comprises five lines of code. One line is spent for reading out the image from memory, two lines for the template evaluation, and additional two lines to assure the proper sequence of the control signals. With the initial image as shown in Figure 5.14, all but one of the cells in the array are forced to change the state at the same time, resulting in a power dissipation of 9.8 μW per cell at the supply voltage $V_{DD1} = 1.2$ V.

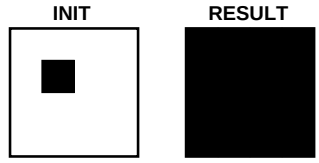


Figure 5.14 The initial image and the resulting one obtained in the power measurements.

Technology	6M-1P 0.18 μm CMOS
Supply Voltage	1.2 V
Control Signal Voltage	1.8 V
Array Size	4×4
No. of Transistors per PE	64
PE Area	155 μm^2
State Representation	1-bit
State Dynamics	Inverted Positive-Range High-Gain
I/O	Digital
Weight Programmability	1-bit
Dynamic Power per PE	9.8 μW

Table 5.4 Chip Characteristics

5.4 Discussion

The measurement results presented in the previous section confirm that the structure works properly in performing both the Boolean logic combinations and the template evaluation. If the supply voltage V_{DD1} is made variable for template-by-template evaluation, speed and robustness can be effectively traded for power consumption. The external buffers in the bidirectional data bus stopped working properly at 0.95 volts and this limited the minimum supply voltage to 0.55 volts in the measurements. Correct results with the Shadow template were obtained at V_{DD1} as low as 0.55 volts, while with the logic operations the limit was 0.6 volts. The correct and data-independent results of the more complex algorithms were achieved at a supply voltage of 1.2 volts. Thus, this value is listed in Table 5.4 as the supply voltage. Also, the given dynamic power consumption was measured with $V_{DD1} = 1.2$ V.

Although the presented cell has the bias term programmable to either 0.5 or 1.5, only operations requiring 0.5 bias were presented. That is because, in the layout drawing process the bias PMOS transistors were improperly scaled (M_{P1} with $W/L = 0.5/0.3$, M_{P2} with $W/L = 0.5/0.6$, and M_{P3} with $W/L = 0.5/0.3$) according to an older version of the schematic. Thus, the mirrored currents were made smaller (about 3 times smaller at 1.2 V supply voltage). For the case of 0.5 bias, having a lower value of the bias than half of the unit current can actually be seen as a means to improve the robustness [48]. However, for the case of 1.5 bias, the mirrored current should be close to 1.5 of the unit current. Nevertheless, the functionality of the 1.5 bias was

verified with the measurements at very low supply voltage (the mirroring of the bias current works better in the subthreshold region): $V_{DD1} = 0.3$ V was used for processing, while the programming and readout were conducted at $V_{DD1} = 0.8$ V. If the PMOS devices in the bias circuit were properly scaled as in the simulations of Section 5.2.1, the cell size would increase to about 170 μm^2. The obtained small cell dimensions and low power dissipation allow for the implementation of a very large arrays. Even one million processors on a single chip become feasible with nowadays CMOS processes.

Obviously, upscaling the array would contribute a noticeable power-consumption. However, it would not increase significantly the per-cell power dissipation. Issuing instructions to a very large array can impose certain time delays due to signal propagation. It also requires strong drivers and may cause a high instantaneous power dissipation if many signals change their states at the same time. The input/output (I/O) operations should not cause performance degradation either. If a large processing array is designed, it usually incorporates the photo detector in each cell (efficient data input) leading to the near-sensor image-processing (NSIP) concept [29]. At the same time, because of the pre-processing in the array, the processor outputs only a small fraction of initial information, e.g., the extracted features of some objects. In this way, the output bottleneck is avoided. Alternatively, this architecture can be used as co-processor supporting the general-purpose gray-scale chip. In this case, the image would need to be loaded through a data bus. However, one should , keep in mind that images are binary (each pixel value being represented by a single bit).

Table 5.5 provides a comparison of the presented design and other chips. However, the reported test-structure can fairly be compared with designs of [39] and [25] only, as they implement similar functionality. The chips of [6] and [31] can operate on gray-scale images and have built-in photosensors, and are included in Table 5.5 to give a broader context.

	This design	[6]	[39]	[31]	[25]
Technology	0.18 μm CMOS 1P-6M	0.35 μm CMOS 1P-5M	0.25 μm CMOS 1P-6M	0.8 μm CMOS 1P-2M	0.5 μm CMOS 1P-3M
Array size	4×4	128×128	176×144	32×32	48×48
PE density (PE/mm^2)	6451	180	3027	71	295
Supported images	B/W	Gray	B/W	Gray	B/W
Photosensors	No	Yes	No	Yes	No
♯ Memories per PE	4 B/W	2 B/W and 8 Gray	6 B/W	8 B/W	4 B/W
♯ Transistors per PE	64	198	73	97	N/A
Weight distribution	digital	analog	analog	digital	analog
Bits per term in A, B, z	1,1,1[a)]	8, 8, 8	6, 6, 9	1, 1, 0[a)b)]	6, 6, 6
τ	4 ns	135 ns	N/A	N/A	50 ns
Power per PE	9.8 μW	180 μW	N/A	N/A	81 μW

[a)] No separate bits for the A and B templates.
[b)] No threshold functionality, 4-connected to the neighborhood.

Table 5.5 Chip Comparison

This page is intentionally left blank.

Chapter 6

Binary CNN Based on Floating-Gate MOSFET

In this chapter, another approach to the hardware realization of the binary-programmable processing element is presented. The design is based on a floating-gate MOSFET (FG-MOSFET) structure and can be used to build a capacitively coupled cellular neural/nonlinear network (CNN) for processing black and white (B/W) images. The computation is performed by charge distribution at the input of a FG-MOSFET inverter, which determines the cell state. There is no current-flow through the interconnections after the network has settled, i.e. the processing has completed, thus a significant reduction in DC power consumption can be achieved. Although the coupling coefficients are basically implemented with capacitances, the network is a versatile processor due to the applied binary programming scheme. Along with the cell structure, the operation principles are described and the simulation results of the 8×8 network are presented.

6.1 Motivation

In the very simple structure of the FG-MOSFET, multiple input signals determine the potential of the floating gate of a MOS transistor by means of the Coulomb interactions. Therefore, the exploration of the FG-MOSFET structure applied to CNN is a good way to approach the capacitive interconnections between the cells. It is definitely easier to proceed with the conceptual and architectural development issues in the well-known MOS technology before facing them in some emerging nanotechnologies. Indeed, capacitive interconnections are inherent to many devices sized in the range of nanometers. Practically, all of the nanodevice concepts that rely on the electron charge transport, e.g. the single-electron tunneling (SET) transistor, feature such

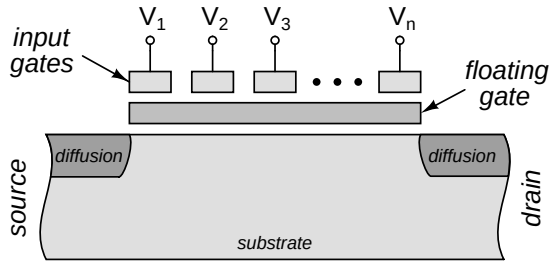


Figure 6.1 Conceptual drawing of the basic νMOS structure.

couplings. The main objective is to design a coefficient circuit that can be used to build a programmable network with capacitively interconnected cells. Then, similar strategies could be applied to the development process of a programmable CNN based on nanodevices.

6.2 Neuron MOSFET Structure

The neuron MOSFET structure has been proposed and described by Shibata and Ohmi [49]. This floating-gate transistor is capable of performing more advanced functions than simple current switching. The device can calculate a weighted sum of multiple input signals at the gate level. The result of the sum operation determines the output state of the transistor. Such a trigger, controlled by numerous inputs, resembles the behavior of a biological neuron, and thus the structure has been named "a neuron MOSFET" (abbreviated as νMOS).

The basic structure of the neuron MOSFET is sketched in Figure 6.1. The gate of a MOS transistor is an electrically floating node, to which multiple input gates are capacitively coupled. Such a capacitive network (modeled in Figure 6.2) is used to perform the weighted sum computation by means of charge redistribution. The potential ϕ_F of the floating gate is determined by the linear sum of the input voltages V_i weighted by the values of the coupling capacitances C_i as

$$\phi_F = \frac{\sum_{i=0}^{n} C_i V_i}{\sum_{i=0}^{n} C_i}, \tag{6.1}$$

where V_0 is the potential of the substrate, and C_0 is the capacitance between the floating node and the substrate. The neuron MOSFET structure has been further improved to increase the accuracy [50]. In addition to layout enhancement, calibration techniques have been proposed as means to reduce the circuit sensitivity to the variations of an IC fabrication process.

In terms of power dissipation, the voltage-mode summation performed at floating gate has a clear advantage over the wired sum of currents. Since there is no DC flow

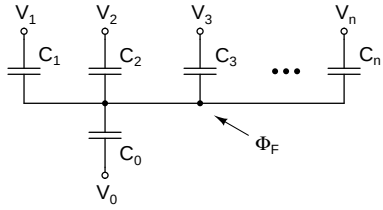


Figure 6.2 Capacitive network model of the floating gate structure.

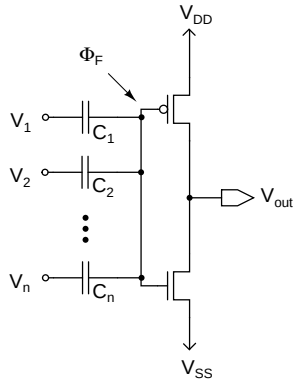


Figure 6.3 Schematic diagram of the neuron CMOS inverter.

in the capacitive network, it is important to keep the power dissipation low in other parts of a circuit as well. Therefore, the complementary CMOS-like configuration is essential. It can also enhance the speed performance. Schematic of the neuron CMOS inverter is shown in Figure 6.3.

A variety of possible applications have been projected for the νMOS structure. It is naturally suitable for (but not limited to) the hardware realizations of neural networks. A wide range of applications in binary and multivalued digital as well as analog circuits have been exploited. For instance, compact νMOS implementations of A/D and D/A converters have been proposed and tested [50].

6.3 Cell Structure

In the presented approach, the cell structure is based on the neuron CMOS inverter. The positive-range high-gain output nonlinearity [26] is applied, in which the dependence between the cell state and output takes the form of a threshold function. This nonlinearity can be mathematically expressed as

$$V_Y = f(V_X) = \begin{cases} V_{DD}, & V_X \geq V_{th} \\ 0, & V_X < V_{th} \end{cases}, \tag{6.2}$$

where voltages V_Y and V_X represent the cell output and state, respectively, and V_{th} is the threshold voltage of the comparator (inverter). As a consequence of the noninverting relationship between the cell state and output, black pixels of an image are represented in this approach by the HI logic value of the cell state V_X. This is opposite to the nomenclature used in the CMOS implementation presented in the previous chapter.

The cell state is determined by the potential of the floating gate ($V_X = \phi_F$). The switching threshold voltage of the inverter V_{th} is defined as the gate voltage at which the inverter output becomes $V_{DD}/2$. It can be assessed from the supply voltage and the n- and p-channel MOSFET threshold voltages (V_{Tn} and V_{Tp}), [50]:

$$V_{th} = \frac{V_{DD} + \sqrt{(W_n\mu_n L_p)/(W_p\mu_p L_n)}V_{Tn} + V_{Tp}}{\sqrt{(W_n\mu_n L_p)/(W_p\mu_p L_n)} + 1}, \tag{6.3}$$

where W_n (W_p) and L_n (L_p) are the channel width and length of the n-type (p-type) MOSFET, respectively, while μ_n (μ_p) stands for the electron (hole) mobility. Since the MOS devices of the inverter have the channel W/L ratios designed with respect to their carrier mobilities, Equation (6.3) reduces to

$$V_{th} = \frac{1}{2}(V_{DD} + V_{Tn} + V_{Tp}). \tag{6.4}$$

To make the CNN processor versatile, the binary programming scheme has been applied. Assuming that substrate is connected to the ground potential, the state equation of a cell can be written as

$$V_X(t) = \frac{\sum_{k=-1}^{1}\sum_{l=-1}^{1} C \cdot V_{Yk,l}(t) \cdot AB_{k,l}}{\sum_{k=-1}^{1}\sum_{l=-1}^{1} C \cdot AB_{k,l} + C_{sub}}, \tag{6.5}$$

where C is the unit coupling capacitance, $AB_{k,l}$ are the coefficients i.e., the elements of either A or B template matrix, and $V_{Yk,l}$ represents the output signals of the neighborhood. The capacitance between the floating gate and the substrate, denoted by C_{sub}, includes the parasitic component as well as the total bias capacitance.

The cell structure is depicted in Figure 6.4. Three major circuit blocks can be distinguished. Namely, the negative bias, the coefficient circuits (both on the left side in Figure 6.4), and the computing kernel (on the right side in Figure 6.4). Interestingly, an array consisting of such cells does not need to be surrounded by a ring of border cells. Instead, a dedicated global signal *BORDER* replaces the required neighborhood contribution $V_{Yk,l}$ and controls the corresponding switches in the coefficient circuits. If the black border is required, this signal is set to logical high state (HI) and cause the driven switches to conduct. Otherwise, it should be set to logical low state (LO).

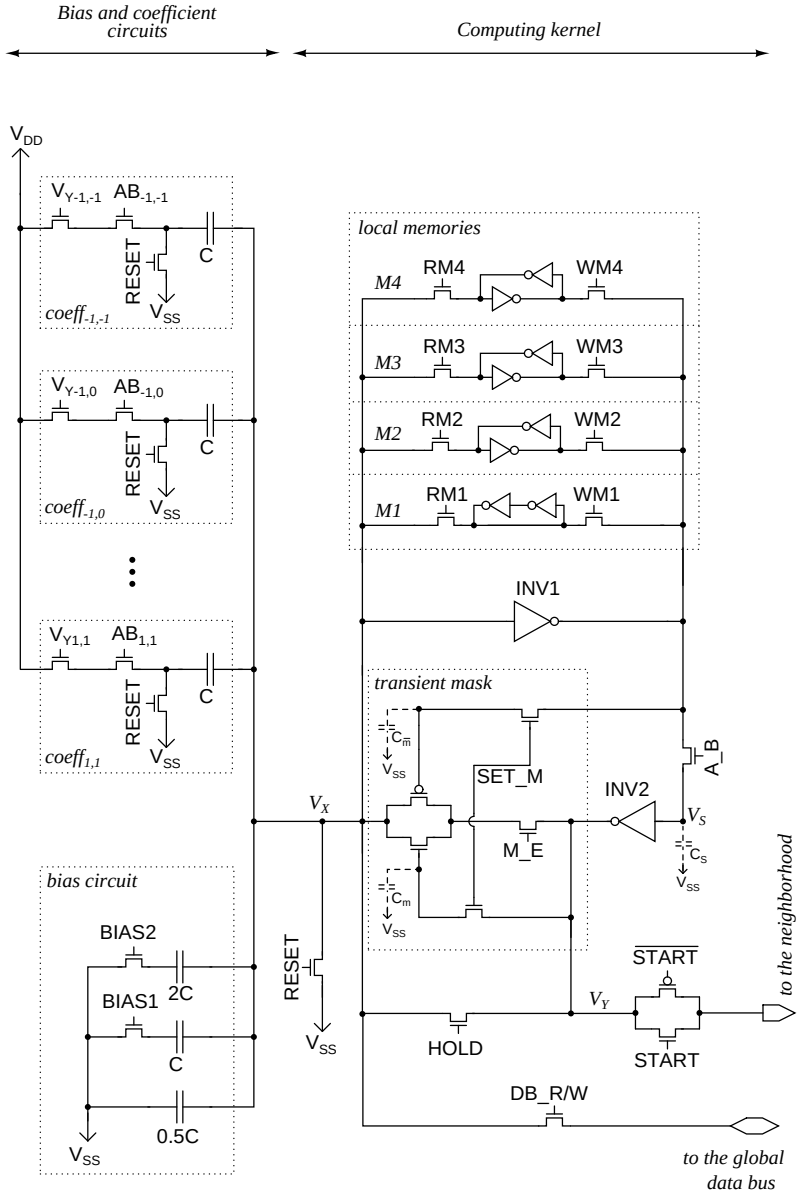


Figure 6.4 The structure of a CNN cell based on the νMOS structure.

6.3.1 Coefficient Circuits and Bias

The coefficient circuits and the bias are composed of capacitors and switches. Capacitances participate in the charge redistribution on the floating gate, thus changing its potential. Actually, as can be observed from Figure 6.4, it is a pseudo floating gate structure since a number of transistors are connected to it. Each coefficient circuit ($coeff_{-1,-1} \cdots coeff_{1,1}$) consists of a unit capacitor C and three control switches. The neighborhood contributions ($V_{Y-1,-1} \cdots V_{Y1,1}$, including self-feedback) together with template terms ($AB_{-1,-1} \cdots AB_{1,1}$) determine which of the coefficient circuits are active. When the activation requirements are fulfilled, i.e. both $V_{Yk,l}$ and $AB_{k,l}$ are high, the corresponding capacitor is connected to V_{DD}, and the potential of the floating gate increases. Otherwise, both terminals of the capacitor are electrically floating nodes, and thus the influence of such coupling can be neglected. The $RESET$-controlled switch is used to bring the circuit to the initial state, i.e. discharge the capacitors, before another template is evaluated.

The negative bias implementation also relies on the charge sharing and is designed in a very similar manner. The circuit consists of three paths comprising capacitances with values related to the unit capacitance: $0.5 \times C$, $1 \times C$ and $2 \times C$. Additionally, two of them contain switches that are controlled with the signals $BIAS1$ and $BIAS2$. Therefore, the overall strength of the bias can be programmed to four different values: 0.5, 1.5, 2.5, and 3.5. Since the capacitors are connected to V_{SS} instead of V_{DD}, an opposite influence on the floating gate potential is obtained.

6.3.2 Computing Kernel

The structure of the computing kernel of a cell is derived from the design proposed in Section 5.1.2.1. It contains two inverters $INV1$ (vMOS) and $INV2$, a transient mask based on the programmable transmission gate, four SRAMs collected in the local memory block, and a number of switches. Additionally, three parasitic gate capacitances C_s, C_m, and $C_{\overline{m}}$, utilized as the dynamic memories, are displayed with the small dashed-line symbols in Figure 6.4. The FG-MOSFET implementation extorts a number of modifications of the computing kernel structure. Namely, since the coupling capacitances are an integral part of $INV1$, the $START$ switch had to be placed at the cell output. At the same time the A_B key is moved into the position between the inverters. As a result, the $INV2$ constantly drives the switches in coefficient circuits during the processing (regardless of the evaluated template type), hence minimum size transistors can be used for the keys $V_{Yk,l}$. Moreover, the $START$-controlled switch, which connects the output to the corresponding coefficient circuits of each cell within the neighborhood, is implemented with a transmission gate to assure that full logic levels are conveyed (image data and control signals have the same voltage swing).

When the charge distribution in the floating gate is initiated with $START \nearrow$, the weighted sum of the neighborhood contributions and the bias determine the potential of the floating gate, and thus the cell state. When this potential exceeds the threshold voltage of the $INV1$, it is considered as HI logic value and the inverter output flips. The cell state can be latched in the SRAM-like loop formed by inverters $INV1$ and $INV2$ and switches controlled with global signals $HOLD$ and A_B. Of course, the A_B signal is also used to choose whether an A- or B-template is run. For the A-template it is kept HI (the switch is conducting) during the processing so that the cell output follows the changes of the cell state. For the B-tamplate evaluation the A_B must be brought back to LO before the processing is initialized (the parasitic gate capacitance C_S stores the desired value). The $RESET$-controlled switch plays the same role as in the coefficient circuits, i.e. it brings the floating gate potential to the ground level (V_{SS}) before another computation begins. Otherwise, the charge accumulating at the floating node during the algorithm processing would cause the circuit malfunction. Additionally, it can be used to set the initial state value, thus performing the tasks of the signal $ZERO$ introduced in the basic cell structure (see Section 4.3.2). The DB_R/W signal serves as a read/write enable for the image data transfers to and from a global bus. The input image data and/or intermediate processing results can be stored in the local SRAMs. One of these memories, namely $M1$, is connected so that the cell state can be inverted in a fast and robust way by means of the write-and-read operation sequence. Due to the fast inversion ability and the placement of A_B switch between the inverters $INV1$ and $INV2$, the IM_E signal is not used as the input of $INV2$ can be a different value than the actual output of $INV1$ (e.g., its inversion). The structure of the transient mask is based on the programmable transmission gate and utilized to implement a fixed state map as well as to set the conditions in computing the Boolean combinations.

6.4 Simulation Results

In this section, a selection of operations is presented. The logic functions computed locally within each cell as well as examples of template evaluations are described to show the cell versatility. The operations are visualized by the simulation results of an 8×8 network. For the sake of clarity, the more complex algorithms are accompanied by the corresponding sequences of the global control signals.

The cell was designed for a 0.18 μm CMOS process to operate with the supply voltage $V_{DD} = 1.2$ V. The simulations of the single cell as well as the network were performed with Eldo, and the level 59 parameters were used for all transistors. Coupling capacitors were modeled with ideal capacitances. The value of the unit capacitance C was set to 50 fF, as such a value has been implemented within a neuron MOS structure using two polysilicon layers for the gates [50].

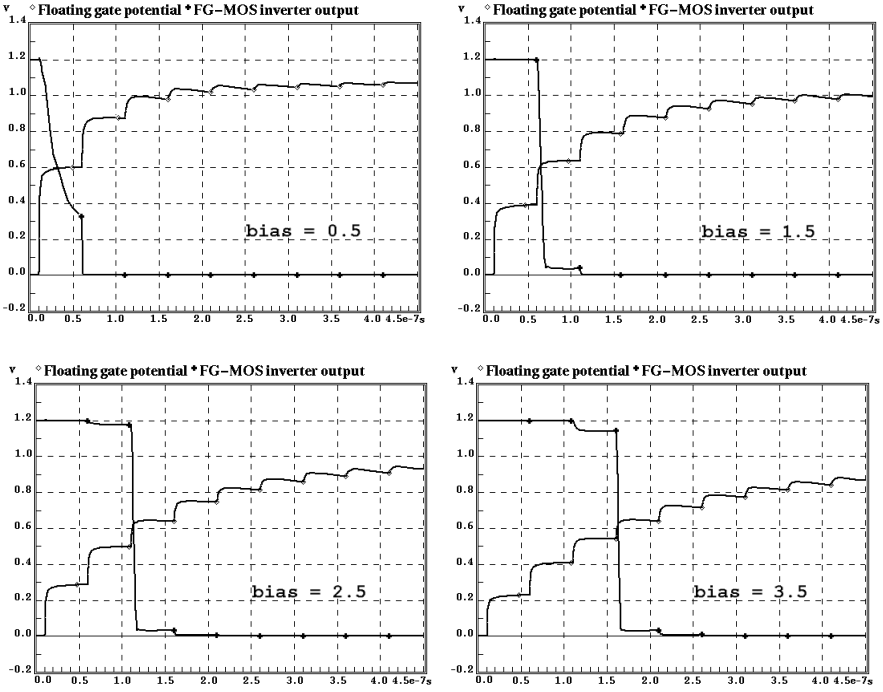


Figure 6.5 Cell triggering at different bias values.

6.4.1 Cell State vs. Neighborhood and Bias

To test the operation of charge redistribution, the activation of $INV1$ was simulated with the positive contributions from the neighborhood successively turned on, at different values of the negative bias. The results are presented in Figure 6.5. The cell state is properly detected as HI each time the number of active coefficient circuits exceeds the bias set. The visible compression effect in the curve representing the floating gate potential is not a critical issue, as it gets weaker with the increasing bias. As can be deducted from the waveforms in Figure 6.5, when $|D-z| = 0.5$, it may happen that the $INV1$ is driven by a voltage near its threshold value, for which the transistors are not fully saturated/cut-off. Such a situation concerns the robustness, the speed of operation and/or the power consumption. Therefore, the minimum size of a unit capacitance is limited. It is also worth noting that every switch connected to the (pseudo) floating node introduces parasitics that degrade the performance. In addition to a carefull selection of the unit capacitor size, the result of each template evaluation should be latched in the SRAM-like loop ($INV1$, A_B, $INV2$, $HOLD$) for a short moment. In this way, a full logic level at the floating gate is restored, reducing the power consumption at a small cost in the speed of operation.

Op1	Op2	AND	OR	NAND	NOR	XOR
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0
V_m		$Op1$	$\overline{Op1}$	$Op1$	$\overline{Op1}$	$Op1$
V_S		$\overline{Op2}$	$\overline{Op2}$	$Op2$	$Op2$	$Op2$
$V_X(0)$		0	1	1	0	$Op2$

Table 6.1 Two-operand logic functions.

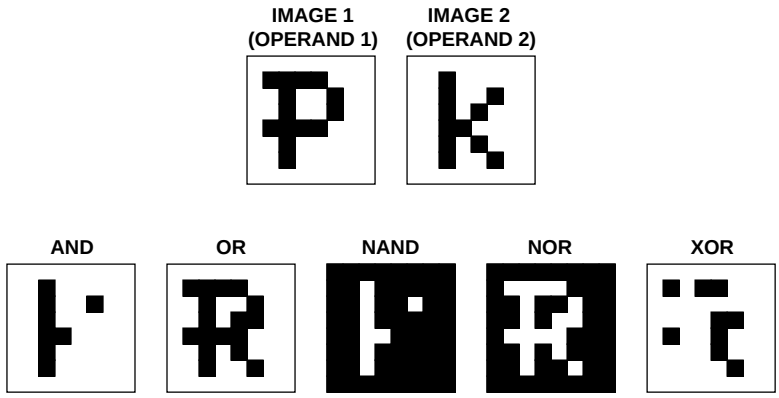


Figure 6.6 The operands ($IMAGE1$ and $IMAGE2$) and the results of the logic operations.

6.4.2 Logic Operations

Also in this implementation, basic logic functions are computed as a conditional pass-through operation (or inversion) with the output of an operation $V_X(0)$ initialized (pre-programmed) to a certain desired value. The way these operations are performed follows the guidelines of Section 4.5 and is analogous to the description of Section 5.3.1. However, this time, $INV2$ and M_E instead of $INV1$ and IM_E are used for conditional driving. For clarity, Table 6.1 also contains the values stored at the corresponding nodes of the cell structure for each logic operation. V_m is the voltage stored at C_m, V_S is the value stored at C_S, i.e. the input of $INV2$, and $V_X(0)$ is the predefined output value stored at node V_X. The $RESET$ signal is used to initialize the result of a function as LO. In case the outcome should be preset to the high logic level, an additional inversion is performed (writing to and reading from local memory $M1$).

The images corresponding to the operands used in the simulations and the obtained results of logic functions are shown in Figure 6.6.

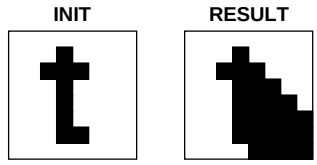


Figure 6.7 The initial image and the result of SE-Shadow template.

6.4.3 A-Templates

6.4.3.1 Simple A-Template: Shadow

Shadow is one of the simplest A-template operations, forcing black pixels to reproduce and propagate into a given direction. Therefore, a shadow-like pattern originating from the black objects of an image is being created. Since this template is robust and it allows to estimate in a quite convenient way the speed of a wave asynchronously propagating throughout the network, it is usefull to test such an operation during the design process.

Shadowing into South-East (SE) direction was chosen for the simulations. In the presented design, it translates to the form of:

$$AB = A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad z = 0.5 \tag{6.6}$$

Each cell corresponding to a white pixel of an image checks whether the nearest neighbor at the North-West location is black. If so, the cell changes its state to represent a black pixel as well. Otherwise, it remains white. Due to the non-zero self-feedback term in the template, the cells corresponding to the black pixels keep their state regardless of the neighborhood condition. As a consequence, each black object of an image will have a shadow into SE direction. The input image used in the simulations and the obtained outcome image are shown in Figure 6.7. According to the simulation results (see waveforms in Figure 6.8), it takes only 1.8 nanoseconds for the black wave to travel a distance of one pixel. In a real implementation, unavoidable parasitics would slow down the operation.

6.4.3.2 A-Template With a Fixed State Map: Hole Filler

The Hole Filler is another example of an A-template. As a result of this operation, all white areas of an image, which are enclosed by black pixels, turn black. Similarly as in Section 5.3.2.2, it serves as a test of the fixed state map implementation. In the binary programming scheme, this template takes a form of Equation 5.4, repeated here for convenience:

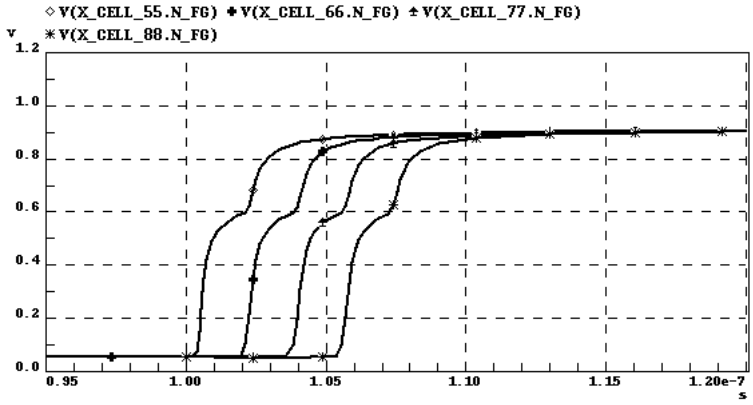


Figure 6.8 Successively switching states of the consecutive cells due to asynchronous wave propagation.

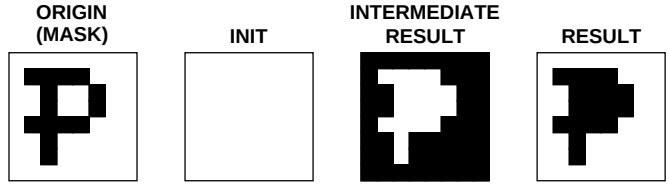


Figure 6.9 The initial image and the result of Hole Filler template.

$$AB = A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad z = 0.5 \tag{6.7}$$

The algorithm used to perform this operation with a FG-MOSFET CNN is similar to that used with the CMOS implementation (see Section 5.3.2.2). Small differences appear mainly due to the different output nonlinearity and interconnection structures.

1. Load an image: $HOLD \searrow$, $DB\_R/W \nearrow\searrow$
2. Set the transient mask condition: $SET\_M \nearrow\searrow$
3. Initialize the network to contain white pixels only: $RESET \nearrow\searrow$
4. Begin the conditional driving of the floating-gate node by $INV2$: $M\_E \nearrow$
5. Set the border black and run the template: $BORDER \nearrow$, $START \nearrow$ ... $\searrow$
6. Invert the result: $WM1 \nearrow\searrow$, $RM1 \nearrow\searrow$.

The input image and the result of this algorithm evaluation are shown in Figure 6.9.

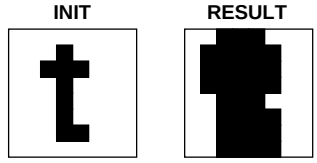


Figure 6.10 The initial image and the outcome of the Object Increase evaluation.

6.4.4 B-Templates

6.4.4.1 Simple B-Template: Object Increase

As in case of CMOS implementation, the Object Increase template was chosen to test the B-template evaluation. For the ease of reading, its cloning template from Equation 5.6 is repeated:

$$AB = B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad z = 0.5 \tag{6.8}$$

The initial image used for simulations and the obtained result of this operation are presented in Figure 6.10. As can be observed, every white pixel of an image that have at least one black neighbor turns black.

6.4.4.2 Higher Bias Template: Junction Extraction

As it is very common in the binary programming scheme, the templates presented above use the smallest possible bias of 0.5. However, a number of applications exist, in which a higher value of the bias is required. As an example of such a case, the Junction Extraction template was simulated. The outcome of this operation is the image that contains only the pixels corresponding to the junction points of a skeleton in the original image. To be interpreted as a junction, a black pixel needs to have more than two black neighbors. In the presented approach, this template is expressed as:

$$AB = B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad z = 2.5 \tag{6.9}$$

The algorithm for computing the Junction Extraction template is following:

1. Load an image: $HOLD \searrow$, $DB\_R/W \nearrow\searrow$
2. Invert it: $WM1 \nearrow\searrow$, $RM1 \nearrow\searrow$
3. Set the transient mask to disable the state change during template evaluation in the cells corresponding to the white pixels of an image: $SET\_M \nearrow\searrow$

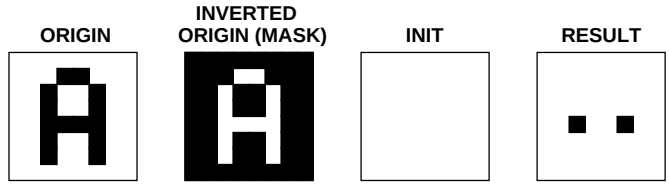


Figure 6.11 Evaluation of the Junction Extraction template.

4. Select the B-template operation: $A\_B \searrow$
5. Set the empty initial image (white pixels only): $RESET \nearrow\searrow$
6. Start the conditional driving of the floating-gate node by $INV2$: $M\_E \nearrow$
7. Evaluate the template: $START \nearrow ... \searrow$

This evaluation procedure is visualized by the original, the mask, the initial and the result images presented in Figure 6.11.

6.5 Discussion

Since the computing power of a parallel array processor such as a CNN strongly depends on the array size, a hardware realization featuring a large spatial resolution is desirable. However, such a design target requires consideration of many implementation issues. Power consumption, layout dimensions, mismatch and robustness become as important as system functionality. Unfortunately, these parameters are not independent of each other, and thus the final layout is a tradeoff. The influence of the fabrication process variation should be taken into account, especially when designing an analog or a mixed-signal system. Obviously, the array architecture requires the cell layout to be as compact as possible, but the component sizes are lower bounded by the mismatch and/or parasitics. Too small devices can cause a lower robustness and result in circuit malfunction.

In the proposed cell structure, most of the transistors operate as simple switches. Therefore, they are not critically sensitive to the mismatch and can be made very small. The group of components affected by the fabrication process imperfections comprise the coupling capacitors and the floating gate comparator ($INV1$). They need to be sized with respect to the robustness requirements to assure a proper network operation. It should be noticed that a systematic mismatch of the coupling capacitors can occur due to the parasitic capacitances. However, this effect can be minimized by parasitic extraction, post layout simulations and appropriate tuning of the design. Additionally, the implementation of the coupling capacitors within the improved neuron CMOS in-

verter structure with a simple calibration [50], helps to keep the layout dense. However, a fabrication process with two polysilicon layers is required.

Another very important aspect in an array processor realization is the power consumption. In the presented design, this issue is addressed in multiple ways. The voltage-mode summation (computed through a charge redistribution in a floating gate) minimizes the static dissipation, since no current flows after the processing is completed. Moreover, complementary structures are used for the state detection and the output function shaping blocks. On the other hand, the low supply voltage limits the currents, and thus the dynamic power consumption remains low.

Chapter 7

Binary CNN Designed for SET Technology

This chapter presents a novel design of a binary CNN in which the cell structure is suitable for implementation with single-electron tunneling (SET) technology. With the binary programming scheme applied, the proposed cell is versatile and can be used to build an ultra-dense CNN processor operating on B/W images.

This chapter is organized so that Section 7.1 gives the motives for choosing SET technology. Section 7.2 describes the concept and features of SET technology. The basics of neural networks are introduced in Section 7.3. Section 7.4 presents a selection of interesting SET-implementations of neural hardware. The SET-based design of the binary-programmable CNN cell is depicted in Section 7.5. The simulation results are shown in Section 7.6 and the implementation issues are discussed in Section 7.7.

7.1 Motivation

Through recent decades, the CMOS technology has flourished. Due to the undergoing development, the device feature size has shrunk by orders of magnitude and the trend still continues. Such an enormous progress in integration capabilities has contributed to crowning CMOS as the leading semiconductor fabrication technology. With no doubt, CMOS will remain a dominant technology for years to come, although the shortcomings of further device shrinking are already in the sight. Nevertheless, the predicted limitations of the CMOS downscaling have initiated a growing research activity in the field of emerging nanoscale devices. One of the most promising and relatively well investigated concepts is the SET technology [51]. It has the appealing potential of downscaling by orders of magnitude beyond CMOS. Thus, SET circuits can achieve

an ultra-high integration-density and a very-high-speed operation, while an ultra-low power-consumption is maintained. Furthermore, a new fabrication technique is not necessary, and SET junctions can be integrated with CMOS logic on the same chip.

7.2 Single-Electron Tunneling Technology

7.2.1 SET Junction

A basic component in SET technology is a tunnel junction. It is composed of two conductors (or semiconductors) separated by a very thin insulator as sketched in Figure 7.1. Such a junction is equivalent to a capacitor as long as no tunneling event takes place. However, due to the very thin insulation layer, tunneling of electrons through the potential barrier is possible at certain circumstances. Figure 7.2 presents energy levels in the junction. When no external excitation is applied, the energy Fermi levels, E_{F1} and E_{F2}, at both sides of the tunnel barrier are equal. In this case, no electron tunneling occurs. The junction is in the state called "Coulomb blockade". However, if an additional energy is supplied e.g., the junction is connected to a voltage source, the energy levels will be shifted by

$$\Delta E = E_{F1} - E_{F2} = e \cdot V, \tag{7.1}$$

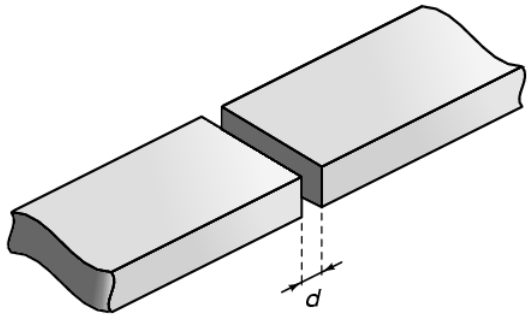
where ΔE is the energy shift resulting from the applied voltage V, and e represents an electron charge. If the supplied energy exceeds the Coulomb energy of

$$E_C = e^2/(2C_T), \tag{7.2}$$

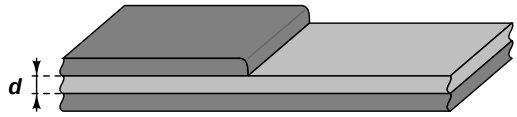
a tunneling event can occur.
There are two main model parameters characterizing the junction:

- tunnel capacitance, C_T, is defined by the size of the junction and materials used for fabrication.
- tunnel resistance, R_T, is defined by the voltage across the junction divided by the current flowing through the junction.

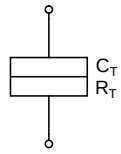
$C_T \propto 1/d$, and $R_T \propto d^2$, where d is the thickness of the barrier insulator. R_T should be much larger than the resistance quantum R_Q to minimize the effect of the quantum fluctuation noise ($E_C \gg E_Q = \hbar\omega/2 = h/2\tau = h/2R_TC_T \Leftrightarrow R_T \gg R_Q \equiv h/e^2 \approx 26\,\mathrm{k}\Omega$). A practical choice of the R_T would be in the order of $10^5\Omega$ [52]. C_T should be small to minimize the effect of the thermal noise ($E_C \gg k_BT$). These requirements for the values of R_T and C_T suggest a wider insulation layer. However, a too thick barrier would completely block the tunneling events and the structure would become useless.



(a)

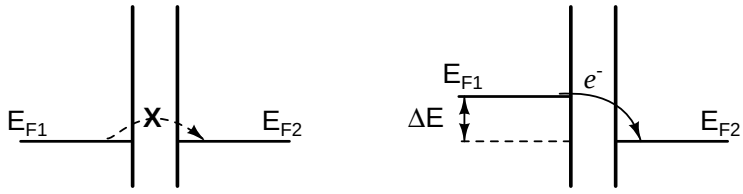


(b)



(c)

Figure 7.1 SET junction: a) in-line structure b) overlapping structure and c) symbol representation for schematics. Metallic SET junction is sometimes called a metal-insulator-metal (MIM) structure.



(a) (b)

Figure 7.2 Potential barrier in SET junction a) without excitation and b) with an external voltage applied.

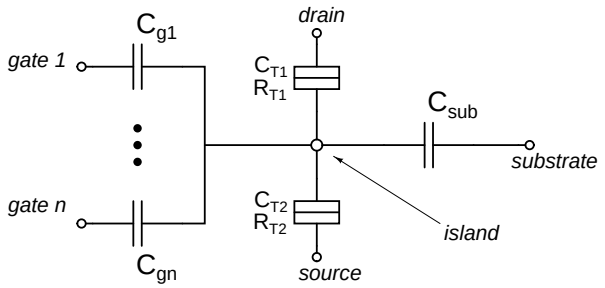


Figure 7.3 The model of a SET transistor.

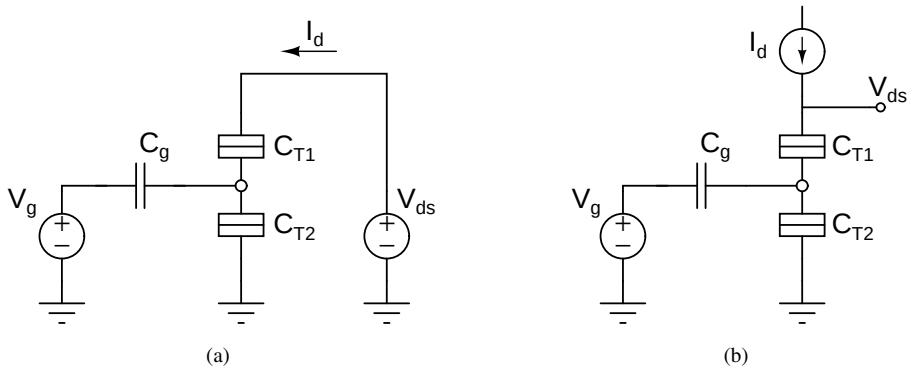


Figure 7.4 SET transistor with a) voltage bias and b) current bias.

7.2.2 SET Transistor

A single-electron tunneling transistor was invented in 1985 and first described in 1986 by Averin and Likharev [51]. Figure 7.3 illustrates the model of a SET transistor structure. It consists of two tunnel junctions connected in series by a very small island (e.g., a quantum dot). A drain-source bias is applied across the junctions while the island is capacitively coupled to a gate bias or inputs.

The output can be either a current or a voltage, depending on how the transistor is biased. The voltage- and the current-biased transistors are shown in Figure 7.4. If the substrate under the device is connected to either supply voltage or ground potential, the stray capacitance C_{sub} starts to function as a bias gate. In this way, n- and p-type devices are obtained as shown in Figure 7.5. Hence, CMOS-like complementary circuits can be built. However, in contrast to MOS technology, the substrate of the n-type SET switch it is connected to the positive supply voltage, while the substrate of the p-type SET device is grounded [53].

An important physical parameter of a SET transistor is the total island capacitance, given as:

$$C_\Sigma = C_{T1} + C_{T2} + C_g + C_{sub}. \tag{7.3}$$

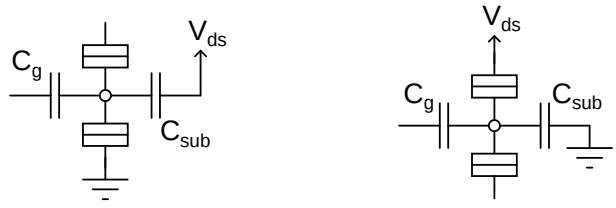


Figure 7.5 SET transistor as: a) n-type switch and b) p-type switch.

From a designer's point of view, the device characteristics are very meaningful. For instance, the I-V curves shown in Figure 7.6, present a Coulomb blockade phenomenon in a SET transistor. As can be seen, when no gate voltage is present, the current flow is possible only above a certain threshold value of the drain-source voltage, V_{ds-th}. This threshold can be lowered (and eventually eliminated) with a voltage applied to the gate. Also, the state diagram of a SET transistor, presented in Figure 7.7, is a useful tool. It demonstrates how the threshold drain-source voltage depends on the transistor parameters and the applied gate voltage, V_g. The characteristic diamond shapes show that the value at which the transistor begins to conduct is a periodic function of V_g. Moreover, it can be observed that at low values of a drain-source bias, the voltage gain and transconductance of a SET transistor may change sign, upon the gate voltage. Additionally, the diamond diagram shows that the voltage gain is limited by the intrinsic capacitance values. For instance, the negative impact of an unavoidable stray capacitance C_{sub} has been simulated[1] and is presented in Figure 7.8, where the transistor's current is plotted on the $[V_{ds}, V_g]$ plane. At sufficiently low temperatures, where:

$$k_B T \ll E_C, \qquad E_C \equiv e^2/C_\Sigma, \tag{7.4}$$

the transistor's current within the diamond-shaped regions is exponentially low (due to the Coulomb blockade phenomenon) and increases rapidly outside these regions. Therefore, to avoid excessive currents in the system under design, SET circuits should be biased in a way that the voltage across each transistor is near the threshold value.

Another very interesting feature of a SET transistor can be observed in its transfer I-V characteristic shown in Figure 7.9, where the drain current and the charge of the island are plotted versus the gate voltage. When the applied gate voltage spans a range many times wider than the drain-source bias voltage, periodic current peaks appear with respect to the electrons transferred to and from the island. This phenomenon is called Coulomb oscillations. Such a periodic function can be used to implement a de-

[1] All simulation results shown in this chapter for the SET circuit were obtained with SIMON (SIMulation Of Nano-structures) software [54].

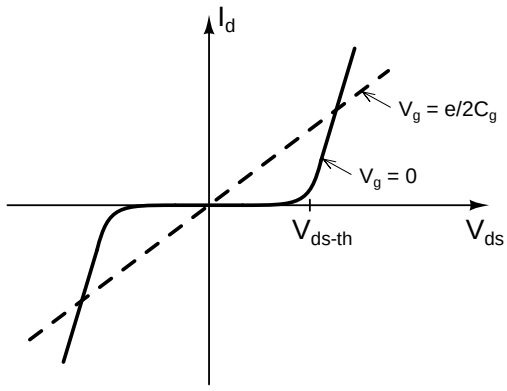


Figure 7.6 Coulomb blockade in a SET transistor. I-V curves show the average current flow at two extreme cases of $V_g = 0$ and $V_g = e/2C_g$.

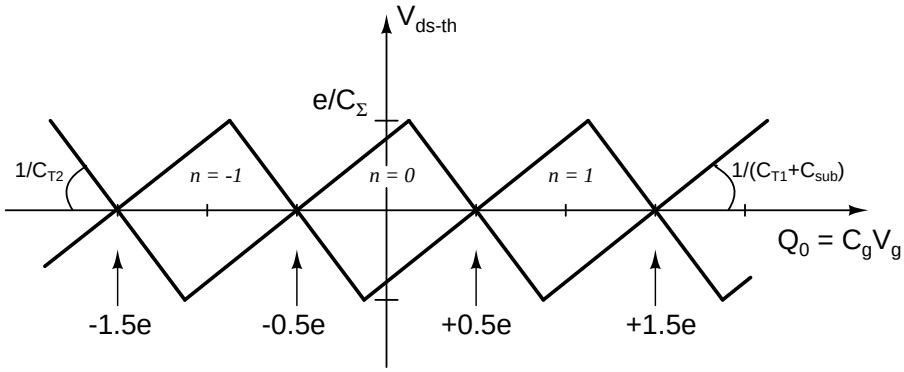
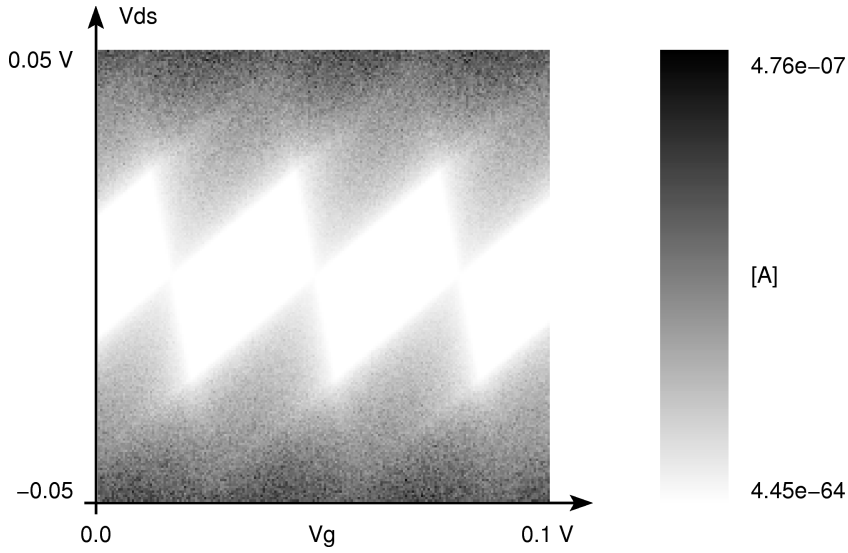
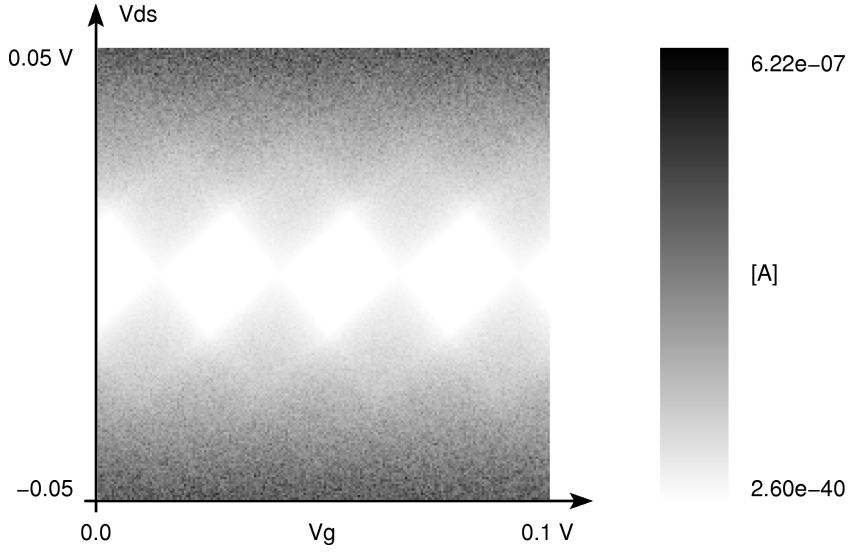


Figure 7.7 State diagram of a SET transistor. e is the electron charge, n is the number of electrons residing on the island, $Q_0 \equiv C_g V_g$ is so-called "external charge" (as a continuous variable, it may be a fraction of the elementary charge e), and C_Σ is the total island-capacitance.



(a)



(b)

Figure 7.8 Stability plots of a SET transistor a) without any substrate capacitance and b) with the substrate capacitance $C_{sub} = 4.5$ aF. The following parameters were used in the simulations: $C_T = 1$ aF, $R_T = 100$ kΩ, $C_g = 6$ aF, and $T = 1$ K.

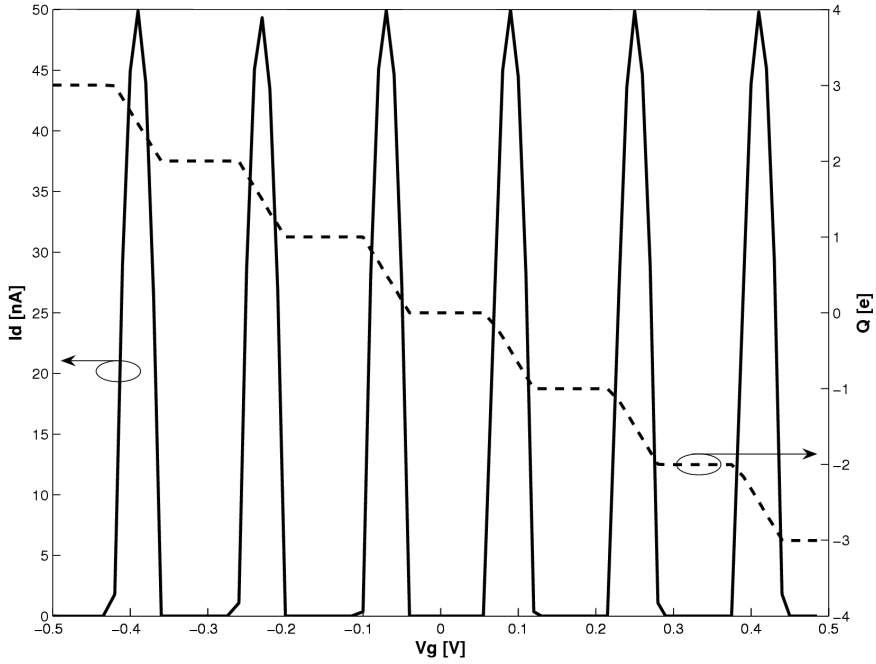


Figure 7.9 The drain current of a SET transistor and the charge of the island (expressed with number of electrons) as a function of a gate voltage. Simulation parameters: $T = 0$ K, $C_T = 1$ aF, $R_T = 100$ kΩ, $C_g = 1$ aF, and $V_{DD} = 10$ mV.

sired circuit behavior (e.g., XOR-based logic). It also can be utilized for an alternative coding of information as a means to make the circuit insensitive to a random background charge (RBC – explained in Section 7.2.4), thus improving its reliability [55].

7.2.3 Operation Regimes

Depending on the value of the applied bias, a SET circuit can operate in two different regimes [56]. One possible operation regime is the single-electron transport, where the information is coded with one or just a few electrons. A SET circuit operating in this mode can achieve the shortest transition times, the lowest power consumption, and the highest possible voltage gain at the expense of driving capabilities and limited robustness. The other, so-called "high-current regime", occurs for relatively high voltages across junctions, and it uses a large number of electrons to charge and discharge the load capacitance of a circuit. Usually, this operational mode appears for circuits built with SET transistors working as current switches. The maximal switching speed of a SET device operating in this regime is determined by its RC time constant (set by R_T

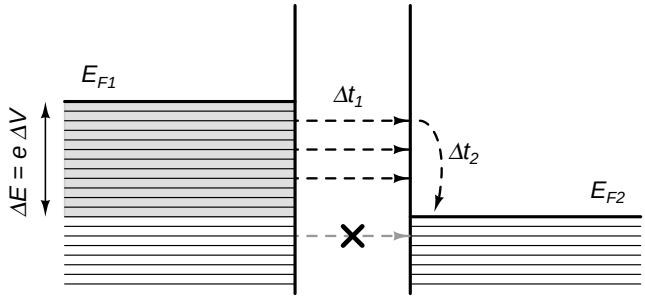


Figure 7.10 The "hot-electron model" applied to a metallic SET junction (diagram adapted from [56]).

and C_T). This regime allows for more robust operation and higher fan-out at the cost of power consumption and speed.

To explain the physical phenomena causing the difference in performance between the SET circuits operating in these two distinct regimes, a so-called "hot-electron model" of a metallic SET junction is shown in Figure 7.10. A voltage applied across the junction causes an energy shift. As a result, an electron from any level within the energy range marked with a gray shaded area of the metal-region on the left hand side may tunnel through the barrier towards the metal-region on the right hand side, in which energy vacancies are available to that electron. As can be deducted, for higher voltages, a substantial amount of electrons will be able to tunnel. However, no electron from below the gray-shaded range is allowed to cross the barrier, as all energy levels below E_{F2} are filled. A tunneling electron travels through the barrier within a time period of Δt_1, which is in the order of 10^{-15} s. After crossing the barrier, it contains a large kinetic energy, thus it is dubbed a "hot-electron". From this point, the electron will drop towards the Fermi level, dissipating a certain amount of energy. The time spent for that fall, Δt_2, depends on the energy shift, i.e. the value of applied voltage. For a significant energy shift, this time will be as large as 10^{-13} s, while for the case of energy levels almost aligned, Δt_2 may also be in the order of 10^{-15} s [56]. This explains the speed and power benefits of the single-electron transport regime.

7.2.4 Random Background Charge Effect

Unfortunately, single-electron tunneling devices suffer from a very serious problem of random background charge (RBC) effect[2]. To understand its influence on the operation of a SET circuit, a following scenario could be considered. A single charged impurity is trapped on the substrate surface, at a distance r from the island of a SET transistor.

[2]In fact, this is an issue common to all nanoscale devices.

If that distance is comparable with the island diameter a, then the island will be polarized with a charge of the order of $e \cdot \frac{a}{r}$ [52]. This charge affects all characteristics of single-electron device. For instance, in the case of a SET transistor, it will determine the Coulomb blockade threshold V_{ds-th}, which for $r \approx a$ may become zero. Even assuming an extremely low concentration of charged impurities, a certain part of chip area would be influenced and a set of devices would be unusable. Moreover, it has been observed that at low frequencies the background (offset) charge is random in size and slowly fluctuates step-wise. Typically, it remains constant for about one minute to one hour, and then changes by a few tenth of e. Although the RBC fluctuations resemble to a certain extent the $1/f$ (aka flicker) noise, the exact cause of the RBC appearance is still not quite clear and different options are considered[3]. Furthermore, the intensity of these fluctuations varies from sample to sample, and between the measurement sessions [57]. This makes it even more difficult to apply any counteraction.

7.2.4.1 Circumventing the RBC Problem

A move from capacitively coupled (C-SET) to resistively coupled (R-SET) devices as they are RBC-insensitive has been suggested [58] to overcome this problem. However, the R-SET approach is impractical for integration, since large (> 1 MΩ) resistance with quasi-continuous charge-transfer is required to be able to compensate for RBC [59]. Such resistors are theoretically feasible, but for the room-temperature operation they become lengthy[4] thus lowering the achievable device density. Moreover, their stray capacitance is likely to become larger than the total capacitance of a SET island, C_Σ, and disable the room-temperature operation.

The RBC problem remains unsolved at the device level. However, a number of alternative solutions have been proposed for other levels of the system design. For instance, due to the periodicity in the transfer characteristic, the information may be put in the amplitude or frequency component of the signal wave as proposed in [55], [60]. In this way, the information is not affected by RBC as long as the output signal carrier (either voltage or current) is strong enough to be detected and processed. However, at certain value of RBC the output signal might be completely blocked (lost). Such a case could be avoided by adding to the SET circuit a variable capacitor in the atto-Farad range [60]. Since such small variable capacitors are not available, this approach is not yet feasible. Another way to conquer the RBC effect is to use calibration circuits to control the island charges [61]. A tunable voltage source can be coupled via additional capacitor to the island in order to compensate for the initial island charge as well as

[3] Taking into account that impurities can migrate and trapped electrons might get released, the presented scenario can be used to explain the RBC occurrence and fluctuations.

[4] It should be much longer than the electron-phonon interaction length, which for most materials is well above 10nm (see [59] and references therein).

RBC fluctuations. This means that each and every island needs to be supported by its own calibration gate (capacitor) and voltage source. Obviously, it is impractical for SET systems with a large number of islands, but may be a good choice for low complexity circuits. Finally, one can surmount the influence of RBC by means of hardware redundancy [56]. This approach does not attempt to eliminate the source of error, but instead, a false output of a particular processing element is tolerated. A neural network is a possible way to achieve a fault-tolerant system. Recently, a SET-based spiking-neuron design has been proposed [62], in which the architecture consists of four layers (the input layer, the logic layer with redundant units, the averaging layer, and the decision layer) and data is processed in a feed-forward manner.

7.2.5 Fabrication

To the advantage over other nanotechnology concepts, the implementation of SET junctions does not impose a necessity for a new fabrication technique. Indeed, SET devices can be defined by a lithography, and even integrated together with a CMOS logic on the same chip [63]-[66]. On the other hand, for the room-temperature operation, very small junction capacitances ($\leq$ 1 aF) are needed, and thus a very-high fabrication-accuracy is required. Currently, it is a problem to produce such tiny devices in big volumes, however, it should be solved by continuous improvement in processing technology.

Traditionally, SET junctions are made of a metal or semiconductor with a thin insulator (e.g., oxide) forming a tunnel barrier. The first metallic SET junctions were produced by evaporating aluminum from two angles through a hanging resist mask (typically created by e-beam lithography on a double-layer resist). This technique, called "shadow mask evaporation" was developed by Fulton and Dolan [67]. Additionally, a number of alternative fabrication approaches have been reported. An interesting proposal relies on the extremely narrow Si wire connecting the source and drain terminals [68]. In fabrication, e-beam lithography and reactive ion etching (RIE) are used to pattern a narrow channel. The noise in e-beam lithography causes variations in the wire width, which is further enhanced during the oxidation. This effect is known as a line width roughness (LWR). After complete fabrication process, such a narrow channel becomes a series of quantum dots, and the smallest dot in that series determines the behavior of the transistor [68]. Another interesting approach is the SET transistor with electrically induced barriers that can be fabricated by the conventional Si integration technologies [69]. Such a technique allows for an ultra-low-power operation, however, it does not exhibit a potential to surpass CMOS in terms of integration density. The device structure is based on a dual-gate MOSFET with two polysilicon layers forming the gates. A narrow inversion layer is induced by a positive voltage of the lower gate,

and the formation of a quantum dot is extorted by a negative voltage of the upper gate (which has a form of two parallel closely-spaced tiny lines). In this way, a certain reproducibility is obtained [69]. A similar concept has recently been used to develop a SET transistor with tunable barriers fabricated in standard Si MOS technology. Such a transistor consists of a silicon nanowire channel and polysilicon fine-gates placed above to induce the electrostatic barriers. Since the barrier parameters depend on the applied gate voltages, this approach gives a high degree of controllability [70].

7.2.6 Applications

The application range suitable for SET devices is quite restricted[5]. The traditional analog approach requires excellent device-matching properties that are beyond the capabilities of SET fabrication techniques. On the other hand, the traditional digital approach demands reliable devices with high ratio of the currents in ON and OFF states. Therefore, the applications should involve architectures with regular structures that allow for relatively easy elimination of malfunctioning sub-circuits. Nevertheless, development of applications for SET devices is an active research field. Especially, multiple-valued logic and memories have gathered a lot of attention [63], [64], [71]-[74]. Other proposed applications include adders [75] and artificial neural network implementations [56], [57], [76]-[78]. Yet, not many SET circuits have found their way into products. For example, the SET electron pump circuit is utilized in metrology as the world's most precise current standard and the SET transistors are used as acute charge sensors, see [52] and references therein.

7.2.7 Discussion

According to the projections in the 2005 Edition Report of the International Technology Roadmap for Semiconductors (ITRS) [9], the ultimate CMOS devices will reach the switching speed of 12 THz and CMOS circuits will operate at 1 THz. For the SET case, devices are predicted to reach the switching speed of 10 THz, while circuits would operate at 1 GHz. That means the SET technology could not surpass the CMOS technology, and thus it is unlikely to become the next leading platform. However, the SET transistor operating in high-current regime is assumed in [9] as the SET device. On the other hand, Reference [56] argues that circuits based on SET junctions operating in the single-electron transport regime can be two orders of magnitude faster and at the same time more energy-efficient than those biased to the high-current regime, as was explained in Section 7.2.3. Therefore, a SET circuit needs to operate in the single-electron transport regime in order to outperform the CMOS counterpart in terms of power consumption and speed.

[5] Generally, it applies to most (if not all) of the nanoscale devices

However, operating with a countable number of electrons limits the circuit fan-out, i.e. driving capabilities, and the problem of interconnections arises. Hence, the system architectures with mainly local information exchange (e.g., CNN) are recommended. Also, a dedicated interface circuit can be useful. Another problem appears in test measurements. It originates from both the small fan-out and the sensitivity of a SET structure to the disturbance induced by an external probe. This could also be surmounted with a special interface, e.g., based on SET-FET hybrid circuits. An example of such a structure has been presented in [66], where a two-stage output buffer is proposed. The first stage consists of a SET transistor with a FET playing the role of a load resistor. The second stage has the same topology but is built with two FETs. In this way, the output signal of a SET circuit is amplified and the driving capabilities are increased.

The appealing features of SET technology as well as the obstacles against its wide employment are collected below to summarize this introduction [56].
Advantages of SET technology:

- No necessity for a new fabrication technology
- Extremely-low power-consumption
- The switching time in the order of femtoseconds
- Small devices with great down-scaling possibility

Problems existing in SET technology:

- Random background charge fluctuations
- Low operating temperature
- Fabrication accuracy
- Interconnections
- Measurements/testing

Due to the device reliability issues harnessing the SET technology[6], fault tolerant architectures are in favor. As indicated in Sections 7.2.4.1 and 7.2.6, implementations of artificial neural networks (ANN) become attractive due to the inherent ability of generating hardware redundancy at the system level. In this way, a robust system based on SET devices can be designed. Since the idea of artificial neural networks and the CNN paradigm are two distinct concepts, the next section will briefly introduce the basics of ANN.

[6] A limited device reliability is generally assumed for all nanotechnologies.

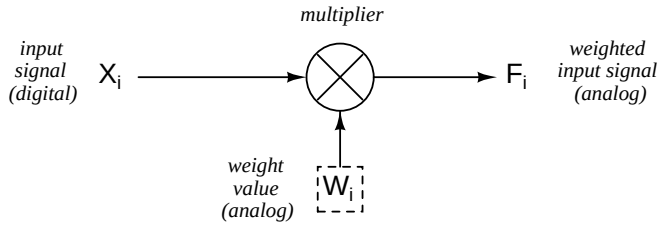


Figure 7.11 The basic synapse model with weight storage and multiplication.

7.3 Artificial Neural Networks

With the progress in studies of the biological neural systems, engineers attempt to apply the discovered strategies to electronics. The aim of these efforts is to achieve a processing throughput, robustness, and power efficiency comparable with those found in the nervous systems. The basic principles, structures and even name conventions were transferred from biology to engineering forming the concept of artificial neural networks (ANN) [56], [57].

7.3.1 Components of a Neural Network

There are two fundamental elements used to compose a neural network: the synapse and the neuron. Their model structures and functionality are briefly described below. A combination of a neuron with multiple synapses is called a perceptron.

7.3.1.1 Synapse

A synapse is used to define the connection strength between two neurons or between a network input and a neuron. With varying synaptic weights, a different network behavior is obtained. The operation of applying a weight can be conveniently modeled with multiplication. Therefore, two functions are assigned to a synapse:

- Storage of the weight value
- Multiplication of the input signal by the weight value

A basic structure of a synapse is schematically presented in Figure 7.11. The digital input signal X_i is multiplied by the corresponding analog weight value W_i. The resulting output of the synapse F_i is an analog signal that is fed as an input signal to the neuron.

7.3.1.2 Neuron

For the development of an ANN, a simple neuron model, implementing basic functionality, is often sufficient. A suitable structure proposed by McCulloch and Pitts [79] is presented in Figure 7.12.

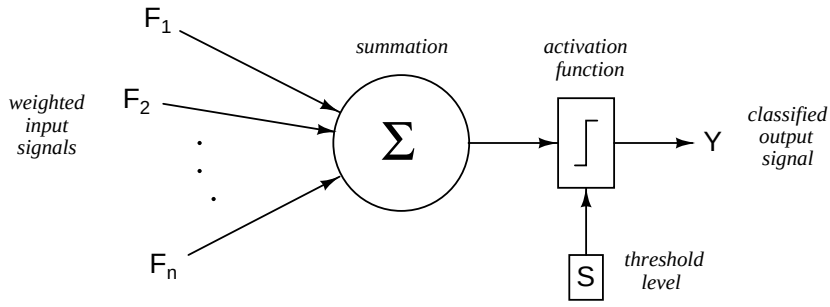


Figure 7.12 The McCulloch and Pitts neuron model.

Conventionally, a neuron performs two operations:

- Summation of all weighted input signals
- Classification performed via an activation function.

The weighted input signals F_i are summed and the result is fed to the classifying stage. The step or sigmoid functions are most often used as the activation function. However, a number of different nonlinearities may serve the purpose of classification. The provided threshold value S defines the border between the classes.

7.3.2 Learning Algorithm

A neural network can be taught to perform a particular operation by means of a learning algorithm. This algorithm is capable of adjusting the weights of the synapses and the activation thresholds of the neurons in order to obtain the desired behavior. There are two types of learning:

- supervised
- unsupervised

In the process of supervised learning, a group of input data sets are loaded into the network. For each data set, the outcome of the neural network is read-out and compared with the desired output, and the error value is generated. Next, the weights and the thresholds are modified by the external supervisor in order to reduce the difference between the desired and the actual output. When the error value is below a certain edge level for all data sets, the training is completed.

In the process of unsupervised learning, the error signal is produced within the neural network. The learning algorithm defines the way this error is generated as well as the procedure for adjusting the weights and the thresholds.

7.4 ANN in SET Technology

When looking at the properties and requirements of bio-inspired computing paradigms, the SET implementation of an artificial neural network comes as a natural solution with multiple advantages. The computing power of a neural network as a parallel processing platform depends on the network size. The larger the number of processing elements (PEs) the more powerful the system. This serves as a driving force for building a network with a huge amount of cells. However, the circuit implementation implies a number of practical constraints. Each PE should be characterized by small physical dimensions and a very low power dissipation. Both requirements can be met if cells are built using SET devices. At the same time, the inherent robustness of a neural network enables using faulty nanodevices to build large systems with acceptable error rate. Especially, the persistent problem of RBC could be overcome with either learning algorithm or hardware redundancy.

7.4.1 Boltzmann Machine Neuron

The Boltzmann machine can be considered as a feedback-type neural network, useful in solving problems related to combinatorial optimization, classification, and association [76]. It consists of many identical PEs bidirectionally interconnected to form a large network. The neurons have binary outputs and the connections have various strengths. The stochastic state transition and annealing algorithms are used to obtain the globally optimal solution without falling into a local optimum. The only difference between the Boltzmann machine neuron and the neuron model described in Section 7.3.1.2 is that the output of the activation function is fed to the stochastic response unit to generate the output bitstream with the state probabilities controlled by the input. The stochastic response unit is an integral part of each neuron.

For practical use, the implementation of a Boltzmann machine must integrate thousands of neurons on a chip. Therefore, the efficient implementation of the random bit generator is a crucial issue. In conventional electronics, it would consist of many devices, and thus a large network would not be feasible. To overcome this problem, Akazawa and Amemiya [76] proposed the idea of utilizing the stochastic character of the tunneling process, inherent for SET circuits operating in high-current regime. Thus, no extra hardware is needed for the random bit generator. The proposed neuron has a structure of a complementary SET inverter [53] tuned to operate under unstable conditions, for which state transitions occur frequently. The required bias levels are obtained from the state diagram. Transient simulations performed by Akazawa and Amemiya show that the neuron outputs a random bitstream with the probability of HI and LO levels depending on the input signal.

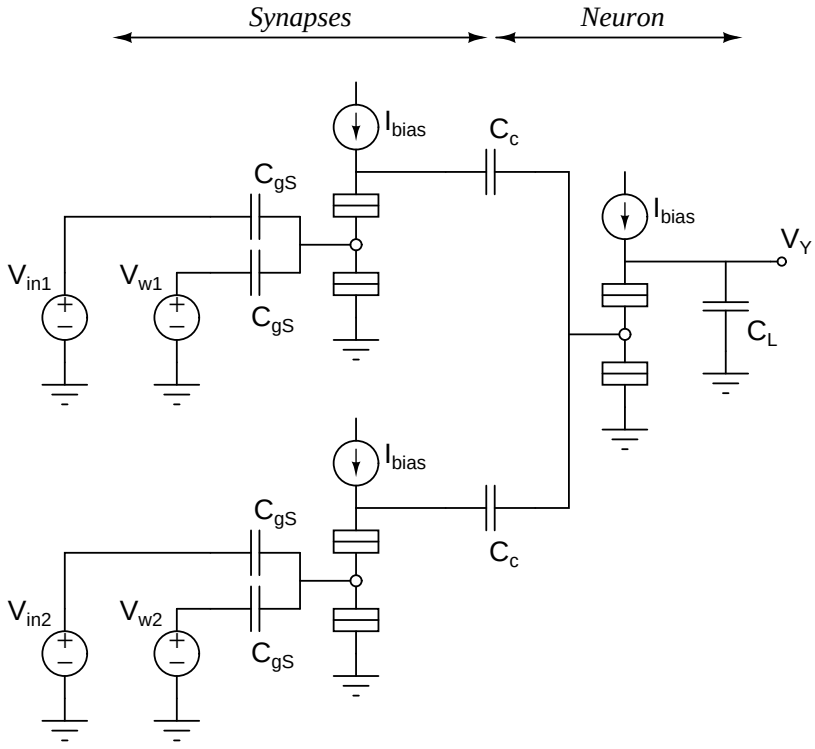


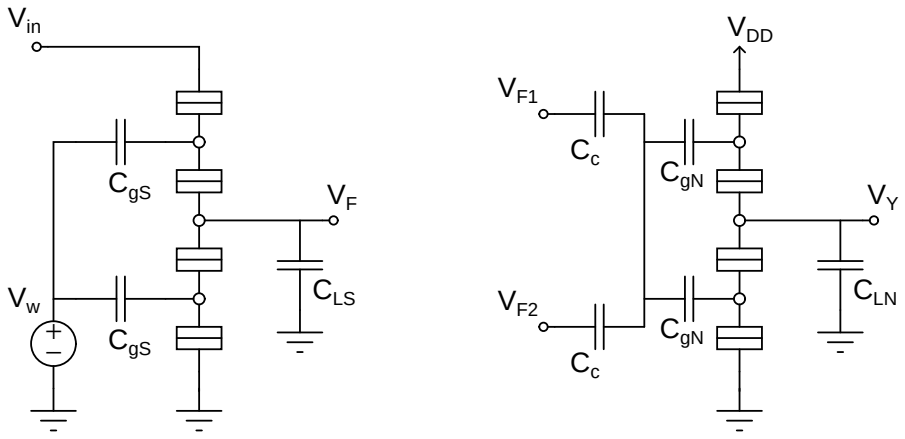
Figure 7.13 Two-input perceptron implemented with just three current-biased SET transistors.

7.4.2 SET Transistor as a Neural Hardware

The neural hardware design in the SET technology proposed by Goossens, Verhoeven and van Roermund is a fully analog implementation [77],[57]. One current-biased SET transistor serves as a synapse circuit. Also, the neuron structure is reduced to one current-biased transistor as shown in Figure 7.13.

The synapse structure applies the weight value stored on voltage source V_w to the input signal V_{in}. The sum of these voltages define the effective gate potential of the synapse transistor. Since the transistor output voltage depends on this gate potential, the V_w can be considered as an offset determining the voltage gain of the SET transistor. The contributions from all synapses are summed by the coupling capacitors C_c in between the two stages. The activation function is a sigmoid with a positive slope, which results from a cascade of the synapse and the neuron transistors.

A clear advantage of this approach is the compactness. Such a simple structure enables extremely dense implementations, and the minimal number of islands reduces the circuit sensitivity to disturbances like RBC fluctuations. Furthermore, a learning algorithm can easily be applied due to a fully analog operation. However, this circuit works in a high-current regime, and thus consumes a significant amount of power.



(a) (b)

Figure 7.14 SET-inverter implementation of a) the synapse and b) the two-input neuron. The SET inverter structure composed of four tunnel junctions connected in series via three quantum dots (islands) is sometimes called a three island structure (3IS).

7.4.3 Neural Hardware Based on SET Inverter

Van de Haar and Hoekstra proposed a neural hardware, in which each neuron and synapse are implemented with a SET inverter working in the single-electron transport regime [56], [78].

The synapse structure is shown in Figure 7.14(a). The digital input signal, represented by the voltage V_{in}, can be set either by the supervisor of the neural system or by the other neuron outputs. The analog weight value, represented by the voltage V_w, is set by the supervisor. This value is discretized to a number of levels defined by the circuit parameters. The choice of the parameter values is a tradeoff between the accuracy of analog signal representation and the speed of operation (and power consumption). The multiplication of the input signal and the weight is obtained with the input signal used as the supply voltage of the synapse inverter structure.

Figure 7.14(b) shows a two-input neuron structure. The weighted input signals, represented by the voltages V_F, are summed by the coupling capacitors C_c. The activation function, in the form of a hard-limiter, is performed by the SET inverter. In order to achieve output signals in the same voltage range, circuit parameters of the neuron are scaled with a factor of 0.8 with respect to the synapse.

Since this implementation does not handle signals in a fully analog way (V_{in} is a digital signal), the convergence properties may be uncertain for some neural applications, and thus should be further investigated.

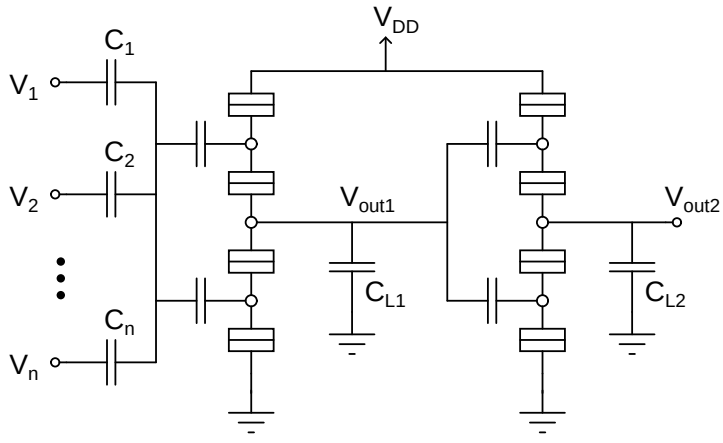


Figure 7.15 CNN cell based on cascaded SET inverters.

7.4.4 3IS Cascade as a CNN Cell

Figure 7.15 presents the CNN cell proposed by Gerousis and Goodnick for implementation with SET technology [80]-[84]. It is based on a cascade of SET inverters. Multiple input signals V_i are weighted by the corresponding coupling capacitors C_i. Therefore, these capacitors implement the functionality of synapses. A capacitive network computes the sum in the voltage domain by means of charge redistribution. The resulting potential of the floating node is fed to the SET inverter cascade, which performs the output classification via a step activation function. Therefore, this structure resembles a neuron. If it is required by a specific application, the inverted output of a neuron V_{out1} is also available for network interconnections.

The proposed cell was used to build networks dedicated to simple B/W-image processing, in order to demonstrate the CNN-like behavior. The presented designs, including a 1×3 network for line detection, 1×3 and 1×5 networks for shadowing, and a 3×3 network for checkboard-like pattern creation, were probably the first CNN applications with SET circuits. However, this approach suffers from a number of drawbacks. First of all, these designs are application specific implementations, in which the network operation is set by the values (ratio) of coupling capacitances, and thus fixed. Due to the lack of programming capabilities, such networks are not versatile. Moreover, since the function is specified by the ratio of capacitive couplings, templates cannot contain negative terms along with positive ones. Therefore, a very limited group of tasks can be implemented. Alternatively, some template modifications are necessary, but no template design rules are given. Also, a fabrication of a large number of coupling capacitors with the required accuracy in the range of zeptofarads would be extremely difficult. Even with the cutting-edge technologies, it is very challenging and rises the issue of robustness.

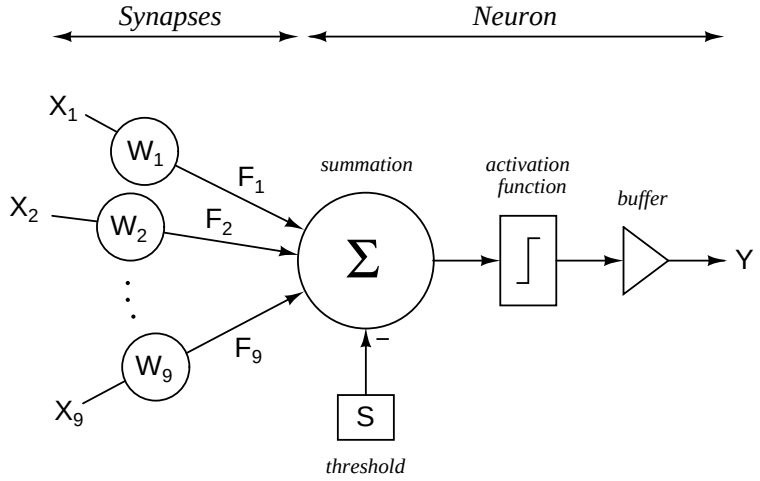


Figure 7.16 Model of a complete cell.

7.5 Binary-Programmable SET-Based CNN Cell

The CNN cell structure proposed in this section is an attempt to adapt the same strategy for programming capacitive couplings as the one utilized in the FG-MOSFET CNN (see Chapter 6), in order to create a versatile binary CNN processor in SET technology. This design has been partially inspired by the works presented in Sections 7.4.3 and 7.4.4. The cell model shown in Figure 7.16 is a perceptron representing the complete processing node of a neural network. Multiple synapses are combined with a neuron, which is an extended version of the basic McCulloch and Pitts model shown in Figure 7.12. The added buffer functionality is to assure the signal levels restoration (voltage gain) in order to avoid information diminishing.

Such a processing element requires a vector of binary input signals, X, a vector of weight values, W, and a threshold value, S. In this approach the weights are limited to binary values. The input signals are multiplied by the weights, summed, and fed into a nonlinear activation function block, which performs the classification. The level of the signal Y depends on whether or not the weighted sum of the input signals exceeds the threshold set. Therefore, it is possible to treat the S signal as another input signal X_i with (typically) an opposite polarization as sketched in Figure 7.16. The buffer assures that the classified output signal, Y, can be used to drive the subsequent neurons.

The complete cell structure implemented with SET transistors is presented in Figure 7.17. The backbone of this circuit is the SET inverter operating in the single-electron transport regime. It performs all of the neuron tasks. The capacitive network at its input acts as an adder, while the inverter provides the output nonlinearity and the required signal gain. As seen from the preceding sections, it is quite a standard

approach to the implementation of neuron functionality. The novelty of this design lies in the synapse circuitry. Each synapse consists of a unit capacitor, C, used for proportional charge distribution at the floating node (input of the SET inverter), and two switches. Neighborhood output signals ($V_{Y1,1} - V_{Y3,3}$) together with template terms ($AB_{1,1} - AB_{3,3}$) determine which of the coupling capacitors are connected to V_{DD}. When both of the synapse switches are conducting, the capacitor is connected to V_{DD}. As a result, the potential of the floating node increases. Otherwise, i.e., at least one of the keys is not conducting, both the capacitor terminals are floating and the influence of such a coupling can be neglected. The threshold level is set in the same manner, except the connection is to the V_{SS} instead of V_{DD}. In this way, the potential changes caused by the bias and coefficient circuits are opposite. There are three branches in the bias structure with capacitances scaled with respect to the unit capacitor to $0.5C$, C, and $2C$. Therefore, the cell can have a bias programmed to four different values: 0.5, 1.5, 2.5 or 3.5. The potential of the floating node is probed and thresholded by the inverter. If this potential exceeds the value of $V_{DD}/2$, the neuron is activated and its output goes LO. Since the neighborhood contribution signals drive p-type SET switches, a second inversion is not needed. The additional *reset*-controlled switch is used to remove the charge remaining at the floating node after the computation. In this way, the initial conditions are restored for each consecutive operation.

7.6 Simulation Results

The neuron circuit has been simulated with SIMON software [54]. Starting with the component values from the van de Haar's design [56], the following set of parameters was established and used in the simulations. Each tunnel junction has a capacitance of $C_T = 1$ aF and a resistance of $R_T = 100$ kΩ. Every SET transistor has a gate capacitance of $C_g = 6$ aF and a substrate capacitance of $C_{sub} = 4.5$ aF. The amplitude of the global control signals and the supply voltage $V_{DD} = 10$ mV. The unit capacitance was $C = 80$ aF, the load capacitance was $C_L = 35$ aF, and the temperature was up to $T = 400$ mK. The selected values do not allow for room-temperature operation, but are feasible with currently available technologies.

The operation principle is visualized in Figure 7.18. The first waveform shows the successively activated neighborhood contributions. As a result, the potential of the floating node rises gradually as can be seen in the second waveform. The visible compression effect is not a critical issue. When the floating node potential exceeds the value of $V_{DD}/2$, the neuron is triggered and the output voltage V_Y goes LO as shown in the last waveform. The presented case with the bias set to 3.5 is the least robust (as explained in Section 4.4), and the fact that no template would require threshold higher than 3.5 gives confidence about the reliable operation. The waveforms in Figure 7.19

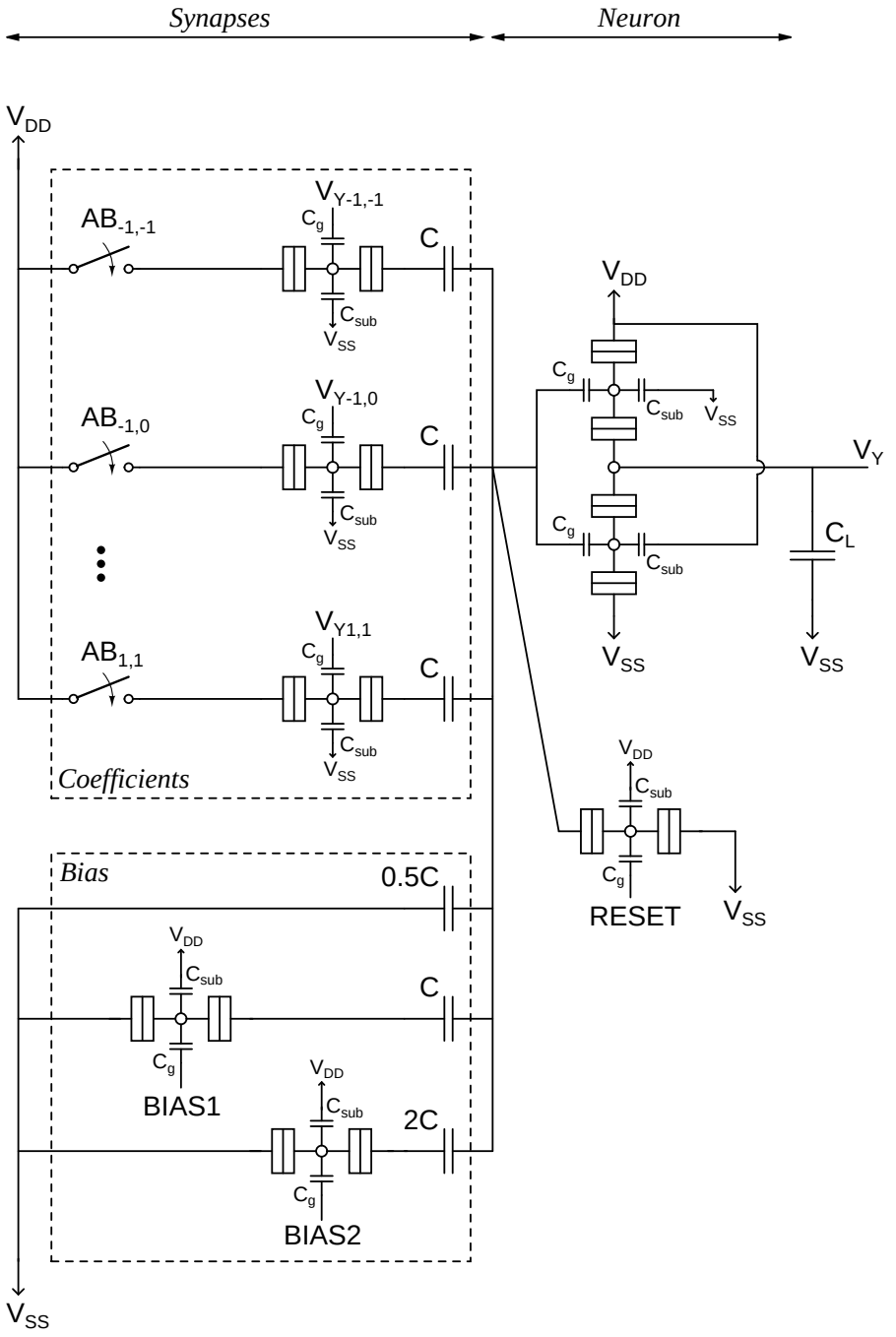


Figure 7.17 The CNN cell structure designed for implementation with SET technology.

show the potential of the floating node and the cell output at each bias condition.

The waveforms in Figure 7.20 present the neuron output voltage and the current dragged by the inverter from the power supply at different bias conditions. The current peaks appearing at the moment the output flips are in the range of 10^{-17} A for the lower two threshold values, and of 10^{-13} A for the higher thresholds. These results show how extremely low power is sufficient for operation of SET circuits.

7.7 Discussion

With the use of SET devices, the area occupied by a cell is kept small and bears the possibility for further downscaling. The power dissipation is kept low as well. With coupling capacitors of 80 aF, the floating node can store up to about 60 electrons. In addition to the energy-efficient voltage-mode summation, the complementary structure of the inverter minimizes the static power consumption. Moreover, the extremely low supply voltage of only 10 mV keeps the dynamic power consumption low. Many portable applications that require lots of computational power would benefit from exploiting such area-speed-power advancements.

Generally, SET transistors feature a low power gain. However, with the use of an inverter structure to shape the output nonlinearity and short (local) interconnections, the voltage gain is provided and the signal restoration is ensured. A remaining problem is the sensitivity to RBC. As the fabrication technologies improve and the SET devices get even smaller, a certain degree of hardware redundancy can be employed as a means to deal with faulty devices. How much smaller than CMOS counterparts the nanodevices have to be to incorporate this redundancy without the penalty in the chip area, remains to be seen.

The presented neuron design is suitable for both pure SET or SET-FET hybrid implementations. The *AB*-controlled switches can be implemented using p-type SET transistors. That would lead to a SET-only implementation of an entire system. However, NMOS transistors can also be used for these switches. In contrast to V_Y-controlled devices, these are driven by the global signals (template terms), and thus do not require a SET implementation. However, it would inevitably increase the chip area and would require a much larger amplitude of the corresponding signals. On the other hand, the *reset* and the *bias* switches, though also driven by the global signals, cannot be replaced with NMOS devices for a practical reason. Namely, a long line (for connection to the outside of the SET array) and a MOS switch as well introduce a relatively large parasitic capacitance, the influence of which cannot be disregarded. When the MOS switch is OFF, this large capacitance would determine the potential of the coupling capacitor terminal, and thus disable it from following the potential of the floating node. As a result, the NMOS switch would be seen by the SET circuit as always conducting.

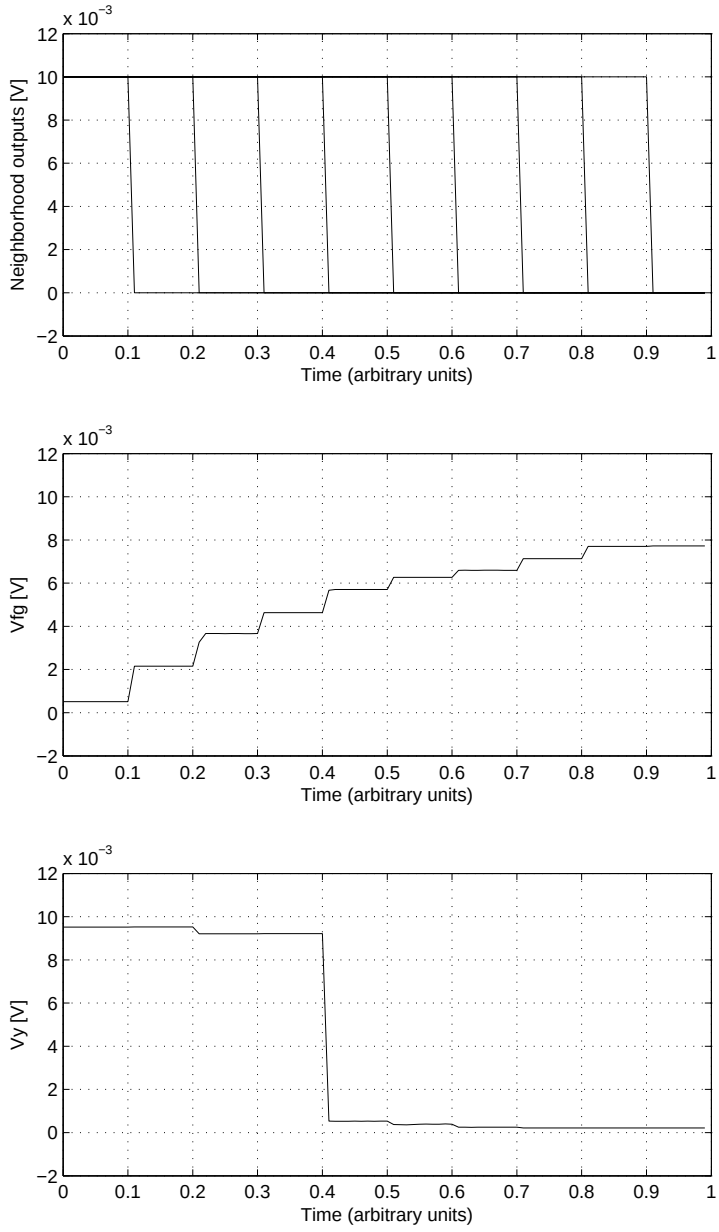


Figure 7.18 Cell operating principle. Neighborhood outputs activate the coefficient circuits causing the potential of the floating node to rise. For number of active coefficient circuits larger than bias (set to 3.5 in this case), the floating node potential exceeds the comparator threshold of $V_{DD}/2$ and neuron output triggers.

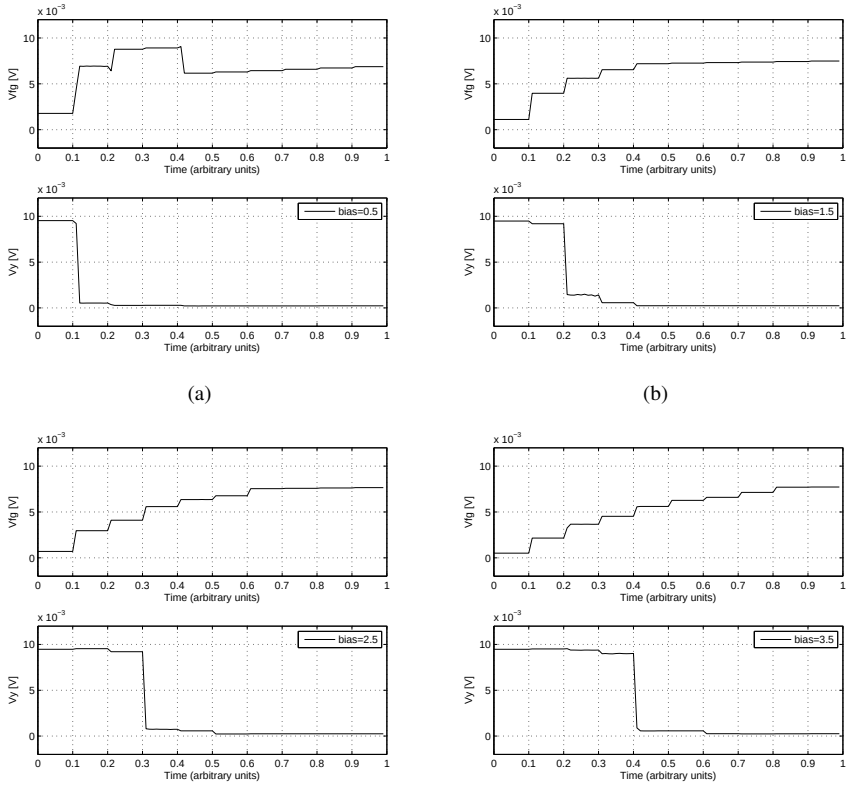


Figure 7.19 Cell output activation at bias programmed to a) 0.5, b) 1.5, c) 2.5, d) 3.5

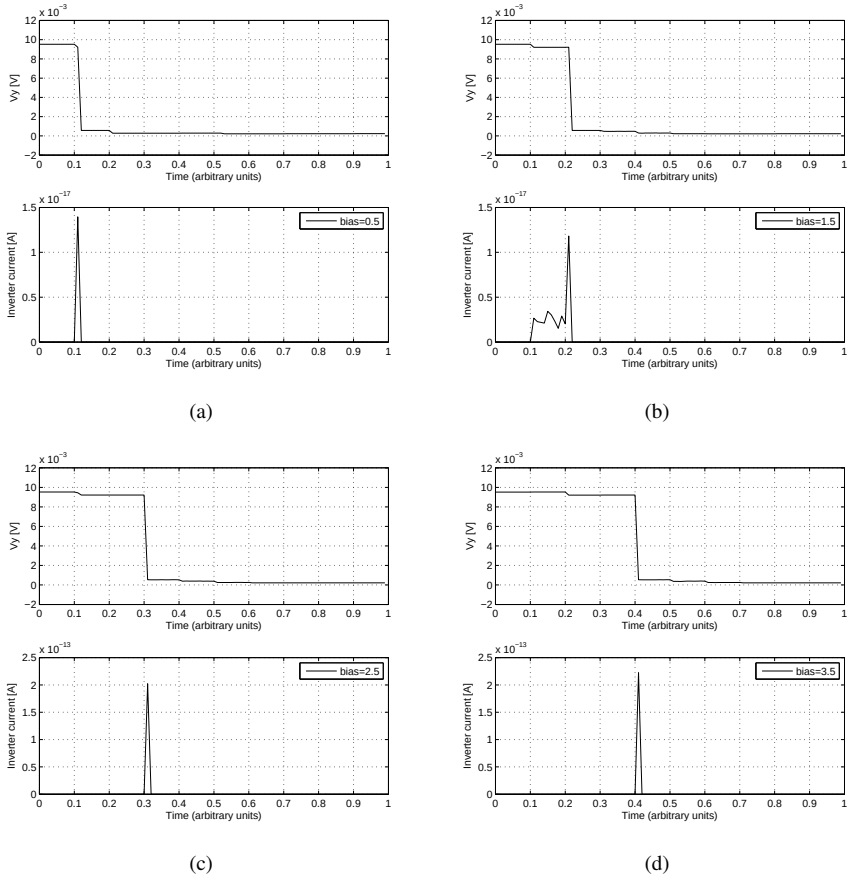


Figure 7.20 The output voltage and the supply current of the inverter at bias programmed to a) 0.5, b) 1.5, c) 2.5, d) 3.5.

Since the SIMON software [54] does not support MOS devices, the influence of these parasitics was simulated with a model 1 pF capacitor. According to the simulation results, this effect is not an issue for the NMOS switches within the coefficient circuits since they connect the synapse coupling capacitors to V_{DD}.

Not many CNN designs suitable for implementation with SET technology have been proposed so far. The most popular is the CNN cell designed by Gerousis and Goodnick. However, it has multiple disadvantages as depicted in Section 7.4.4. The SET realization of a CNN cell proposed in Section 7.5 overcomes some of these shortcomings. This design is probably the first programmable CNN implementation in SET technology. The adopted binary programming scheme is fast and robust, and certainly makes the cell one of the most versatile nanoimplementation of CNN. Most of the templates handling B/W images can be computed with the presented structure. However, to take a full advantage of the applied programming scheme, the cell structure needs to be extended. Namely, the transient mask is needed for computing local logic functions as well as for fixed state map implementation. Additionally, local memories are required for the algorithmic evaluation of the more complex operations. After providing the cell with these extensions, evaluation of all B/W operations would be possible. On the implementation side, a coupling capacitor of 80 aF and junctions with 1 aF tunnel capacitances can be fabricated with the desired accuracy. However, room temperature operation is not achieved. The proposed cell structure was partially inspired by the SET implementation of the neural hardware presented in Section 7.4.3, which can operate on binary images and implements adjustable analog weights. However, since the process of the weight adjustment remains unspecified, the programming strategy would need to be developed. Even if the weights were reduced to binary values, it is not possible to apply the binary programming scheme to that structure. That is due to the lack of ability to block the influence of a neighbor marked with “0” in the template matrix. The synapse multiplier in the form of a SET inverter always contributes to the adder (floating node). As an immediate solution, a tri-state buffer would be applicable, but seems an unnecessarily large structure.

This page is intentionally left blank.

Chapter 8

Other Prospective Nanodevices and Architectures

Over recent years, many concepts of nanometer-scale devices have emerged and suitable architectures have been investigated. One of the most promising nanotechnologies – single-electron tunneling technology – was described in Section 7.2. On the architecture side, Chapter 2 presented the paradigm of cellular neural/nonlinear network (CNN), which is commonly believed to be among the most applicable architectures for use with nanodevices. Partially similar strategies are utilized in the bio-inspired architectures e.g., the artificial neural networks (ANN) introduced in Section 7.3. However, a broad spectrum of different approaches to information processing co-exist. In this chapter, a selection of other interesting proposals are collected and briefly described to give a broader perspective view on the research field of nanotechnology[1]. Although most of these concepts have not yet been used in the CNN hardware realizations, they offer attractive and unique features, which may be exploited in future CNN designs.

This chapter is organized into two sections – one focused on the device concepts, and the other dedicated to alternative architectures.

8.1 Emerging Nanodevices

To overcome the obstacles of further miniaturization of FETs, a number of ideas for nanometer-scale replacements have been suggested[2]. Usually, solid-state nanodevices attempt to take an advantage of the effects that occur at such small sizes due to quantum mechanics.

[1] Yet, the important scientific area of material development is entirely neglected in this chapter, as it goes far beyond the scope of this thesis. Curious reader is directed to the ITRS Report [9] and references therein.

[2] Also, the CMOS technology diverges into many non-classical approaches (e.g., FinFET).

"Island" is a region or layer different from the surrounding material that can be composed of metal or semiconductor. A small island is usually the structural backbone of solid state nanodevices. Proper sizing of the island in each dimension leads to a distinctive behavior. In this way, a variety of devices can be obtained.

8.1.1 Quantum Dots

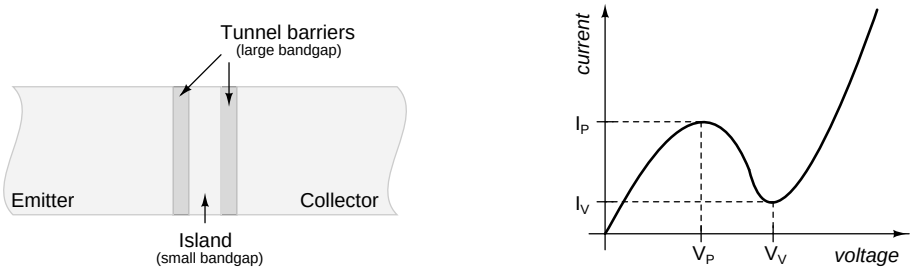
Quantum Dot (QD) is a component, in which the island is short in all three dimensions, confining the electrons with zero classical degree of freedom. Electronic states are quantized in all three dimensions, and the quantum energy levels for an electron are widely spaced. This means that the number of electrons residing on the island is very restricted, and the current flow through the island in any dimension is strictly defined by externally supplied energy.

The QD device group obviously includes an individual dot, sometimes referred to as an "artificial atom", as well as coupled dots, also known as "QD molecule", and a bit more complex structure called "QD cell", in which four or five dots constitute a single binary device. The QD cell structure has led to a development of computing paradigm called quantum cellular automata (QCA) to be described in Section 8.2.1.

Recently, an interesting self-assembly technique for fabricating uniform arrays of individual QDs has led to a development of image processor [85], [86]. The fabrication process consists of the following steps: evaporation of an aluminum layer on silicon substrate, complete anodization to produce a nanoporous alumina film on the surface of silicon, electrodeposition of a semiconductor within the pores, electrodeposition of a metal above the semiconductor, and cotrolled etching of the alumina to expose some of the metal dots to the surface. The pore diameter and the alumina thickness between the neighboring pores depend on the acid used in anodization process. For the sulphuric acid anodization, the pore diameter and the alumina barrier between pores are both 10 nm, while for the oxalic acid anodization, their sizes are 50 nm and 20 nm, respectively. The simulations based on the experimentally extracted structure-parameters, show that the device is able to perform operations like vertical or horizontal line detection. However, these operations were obtained with a manual modification of the interdot conductances. As proposed in [86], the modification could be realized on chip by e.g., damaging horizontal or vertical row of alumina between the dots with a scanning ion beam. However, that would lead to a fixed-function processor.

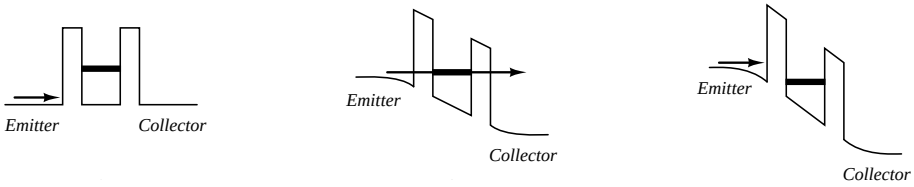
8.1.2 Resonant Tunneling Devices

Resonant tunneling diode (RTD) is a double-barrier quantum-well structure with dimensions of a few nanometers [87]. The device has two contacts (emitter and collector) made of a semiconductor with a small bandgap. Between these contacts are two



(a) (b)

Figure 8.1 Resonant tunneling diode: a) structure and b) $I-V$ characteristics.



(a) (b) (c)

Figure 8.2 Energy diagrams of RTD structure with various voltages applied: a) $V = 0$, b) $V = V_P$, and c) $V_P < V < V_V$.

quantum barriers (semiconductor with a large bandgap) separated by a quantum well (small bandgap semiconductor) as shown in Figure 8.1. Since the well structure is extremely narrow, it can contain only a single resonant energy level and quantum phenomena like tunneling can occur. Electrons can travel from the emitter to the collector only if their energy is in line with this resonant level. The most appealing features of RTDs are the ultra-fast switching capabilities and the region of negative differential resistance (NDR) in their $I-V$ characteristics shown in Figure 8.1.

With the increase of voltage applied, the energy levels at the emitter and the collector side are shifted upwards and downwards, respectively, as shown in Figure 8.2. That shift enables more and more electrons to tunnel through the barriers. At certain voltage value, the conduction band on the emitter side is in-line with the resonant energy level of the well and a current peak can be observed. Further voltage increase pushes electrons past the resonant energy level attenuating the tunneling. This can be observed as a drop in the current forming the valley. For higher voltages, more and more electrons are able to flow over the top of the quantum barriers and a rise in the $I-V$ curve occurs.

Similar conducting properties can also be achieved through the adjustment of the energy levels with a variable voltage applied to a gate, which controls the quantum well potential. In this way, a resonant tunneling transistor (RTT) is obtained [88]-[90].

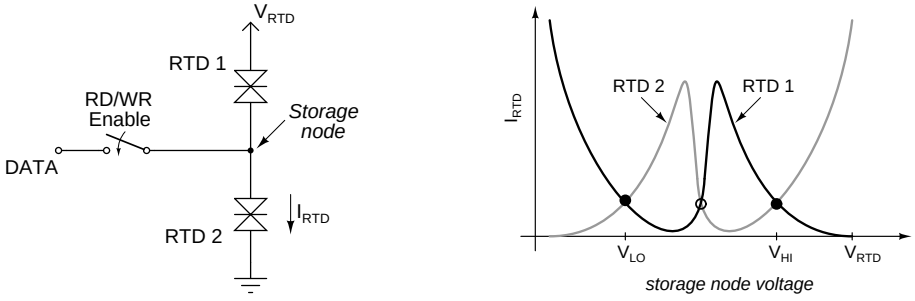


Figure 8.3 Series connection of two resonant tunneling diodes: a) schematics and b) $I-V$ characteristics. The stable operation points are marked with filled circles.

The intrinsic very-high switching-speed and the NDR region in the $I-V$ plot makes the RTDs potentially attractive as high-speed switching devices. Two RTDs connected in series have two stable operating points (and one unstable) as shown in Figure 8.3, and can rapidly switch between the stable points with the aid of gate control. This property has attracted research activities e.g., in the field of memory implementation [91]-[93]. Unfortunately, the peak current of an RTD depends exponentially on the thickness of the tunneling barrier, which is difficult to fabricate with high uniformity. Therefore, to improve the device matching, the peak current should also be gate-controlled. Usually, it is obtained with a transistor integrated together with the series-connected RTDs [94]. However, this approach has a few drawbacks. First, the resulting structure is complex, and thus the dimension scaling is limited. Moreover, the inherent high speed becomes upper-bounded by the RC delays related to charging and discharging of the transistor gate capacitance. Additionally, a low I_{ON}/I_{OFF} ratio impose a challenge as it is orders of magnitude below the requirements for use in digital circuits.

These issues limit the potential use of RTDs to applications, in which a high speed is required, while a low dynamic range is acceptable. Nonetheless, a number of attempts to utilize RTDs in the CNN cell design can be found in the literature [95]-[99].

8.1.3 Carbon Nanotubes

An interesting class of nanodevices are the one-dimensional (1-D) structures. This group includes carbon nanotubes and nanowires. Carbon nanotube (CNT) resembles a graphite sheet rolled into a seamless and hollow cylinder with the diameter in the range of a single to a few tens of nanometers made of carbon atoms only, as shown in Figure 8.4. Nanotubes can be categorized into two groups [100]:

- single-wall nanotubes (SWCNT) made up of a single graphite sheet
- multi-wall nanotubes (MWCNT) consisting of multiple shells

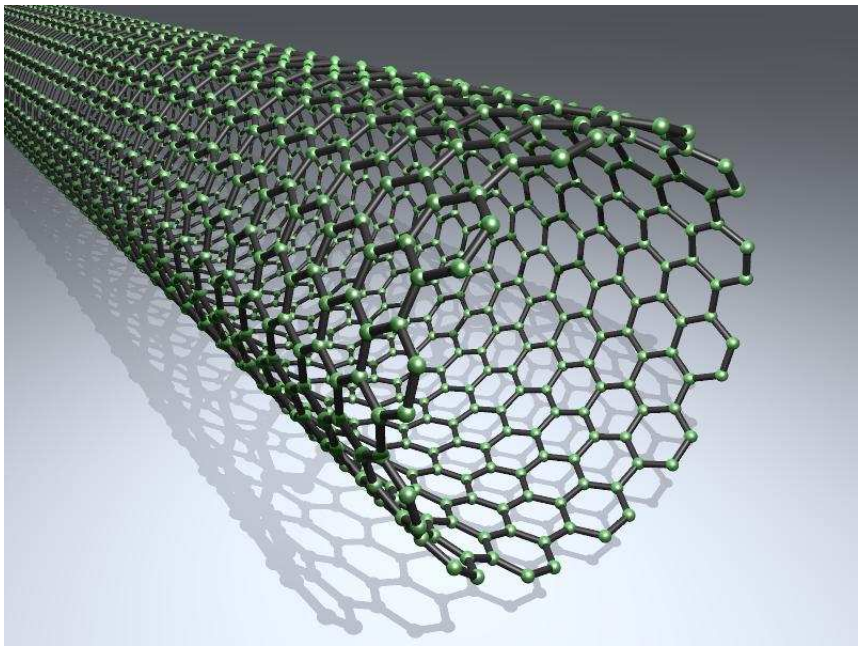


Figure 8.4 A single-wall carbon nanotube.

Such structures exhibit unique physical properties, which make them useful as structures and composites in a wide range of scientific fields including electronics, biotechnology and medicine among others. CNTs are lightweight, highly elastic and are among the strongest fibers. They can exhibit metallic, semimetallic or semiconducting properties depending on chirality and diameter. Electron transport along the nanotubes reveals quantum properties. Moreover, an ability to self-heal monatomic defects has been observed, and its underlying mechanisms and controllability are under investigation [101], [102].

CNTs become attractive for applications like electron field emitters, probes in scanning-type microscopes, supersensitive sensors, gas (e.g., hydrogen) storage or electrode materials (see [103], [104] and references therein). 1-D devices have also been applied to a FET structure, replacing the channel. Such a carbon nanotube FET (CNTFET) can exhibit a superior characteristics [105]. Despite a significant progress in the synthesis of nanotubes with controllable properties, existing challenges in their assembly, processing and fabrication disable their high volume manufacturing.

8.1.4 Molecular Devices

The idea of molecular electronics explore the potential of individual molecules to perform logic operations by means of electron transport and controlled switching. A large number of two- (wires [106], [107], resistive switches [108], diodes [109]) and three-

terminal (transistor-like [110]) devices could be composed into molecular circuits [9], [111]. The envisioned potential drives the research activity in this field. A wide range of the electronic properties of organic molecules may be designed with perfect reproducibility. Chemically induced self-assembly processes (e.g., [112]) could enable fabrication of molecular circuits with very high device densities. Extremely low power consumption may approach the thermodynamic limit, due to low voltages and small number of electrons involved in molecular switching processes. However, challenges that need to be faced include the realization of contacts to the molecular building blocks, the interconnects and the interface to the outside world as well as the stability of the materials through many cycles. Additionally, the circuit design will be very difficult as the contacts and interconnections affect the functionality in the molecule.

8.1.5 Ferromagnetic Logic Devices

Ferromagnetic logic devices use the local orientation of a ferromagnetic material to store the logic state. These devices exhibit the property of being non-volatile and can operate at room temperatures. An example of such a device is the moving domain wall (MDW) [113]. It relies on the segmentation of a ferromagnetic strip into local nanodomains separated by domain walls. Different field vector orientations induce local minima and maxima of the field at the domain walls, which can be used to store bits of information. Logic gates can be formed by applying a rotating magnetic field to structures patterned to perform certain Boolean operations. However, rather low operation-speed is projected for this type of a device [114].

8.1.6 Spin Logic Devices

A great interest in logic implementations with devices that utilize the spin degree of freedom originates from the success of spin-based transport in magnetic storage media. This class of emerging nanodevices includes (among others) different types of spin transistors. For instance, a current modulator, proposed in [115], relies on spin-orbit coupling in narrow-gap semiconductors with ferromagnetic contacts for injection and detection of specific spin orientations. Another interesting concept is the spin-torque transistor, in which the drain-source current is modulated by the magnetization direction of a ferromagnetic base as it affects the spin accumulation in the conducting channel [116]. A spin MOSFET, consisting of a MOS structure and half-metallic-ferromagnets for drain and source contacts, combines the electrostatic and spin-dependent control of the drain current [117]. It allows to achieve a large magnetocurrent ratio, high transconductance and amplification, small power-delay product and OFF-current, while its simple structure would help to obtain high integration density and process yield. The operation of a spin-gain transistor [118] is not based on

spin-dependent current filtering to increase the spin-polarization degree of the current. Instead, the conditions for ferromagnetic transition are created by injecting enough carriers and then switching a small spin-polarized control current to break the isotropy and to induce spontaneous magnetization in the same direction as the control current. In this way, the spin gain is achieved without an external magnetic field.

The spintronics is a vital scientific field and many attractive concepts have already been proposed. However, no practical logic device has been demonstrated so far.

8.2 Novel Architectures

8.2.1 Quantum Cellular Automata

The concept of quantum cellular automata was proposed and developed by the group from University of Notre Dame [119]. Though initially meant as a computing paradigm with semiconductor quantum dots, the idea evolved into a broader architectural concept, where devices can also be in the form of molecules [120] or nanomagnets [121].

A QCA consists of an array of quantum dot cells that are locally interconnected. The connectivity is done by means of magnetic or electrostatic field couplings. Traditional QCA cell has a form of closely spaced quantum dots, where two electrons occupy the antipodal sites on the cell diagonal in one of two possible configurations as shown in Figure 8.5. Due to the quantum confinement, Coulomb interaction between electrons, quantum mechanics and discreteness of electronic charge, such a cell is always in one of these two possible states. Electrostatic perturbation caused by the surrounding environment (e.g., from neighboring cells) extorts a rapid nonlinear switch between the states. That enables the encoding of information bit in the cell. Since closely spaced components can interact with each other, a chain of cells can form a wire to propagate the logic state, as seen in Figure 8.6. Figure 8.7 shows how just a simple geometrical offset in such a chain can perform an inversion. The Boolean logic operations can be computed with compact arrangements of these cells. For instance, the majority gate takes the form of a cross (intersection of two lines), with the arms working as three inputs and an output, while a center cell constitutes a computing device, see Figure 8.8.

The architecture of QCA offers the potential for an extremely low-power operation and ultra high density. Unfortunately, as the information is transferred from one stage to another, signal energy is lost to the environment due to unavoidable dissipation processes. For that reason, QCA circuits are so far impractical for implementation. However, the investigated possibilities to clock the magnetic QCA with a globally applied magnetic field and the molecular QCA with an electric field may result in working solutions.

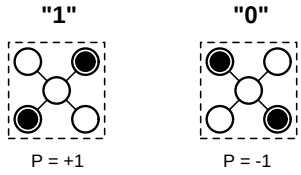


Figure 8.5 Bit storage by means of electron positions within bistable quantum cell.

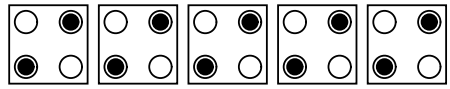


Figure 8.6 Chain of QD cells behaves as a wire.

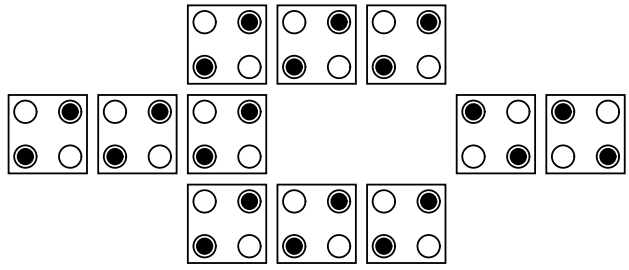
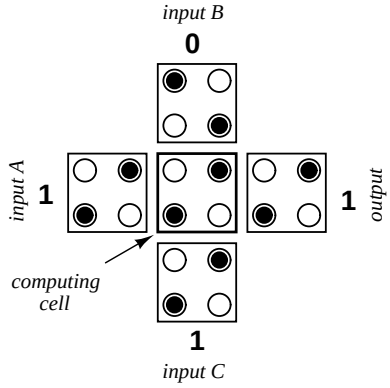


Figure 8.7 Simple geometrical offset can be used to perform inversion.



A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 8.8 The truth table and QCA implementation of the majority gate.

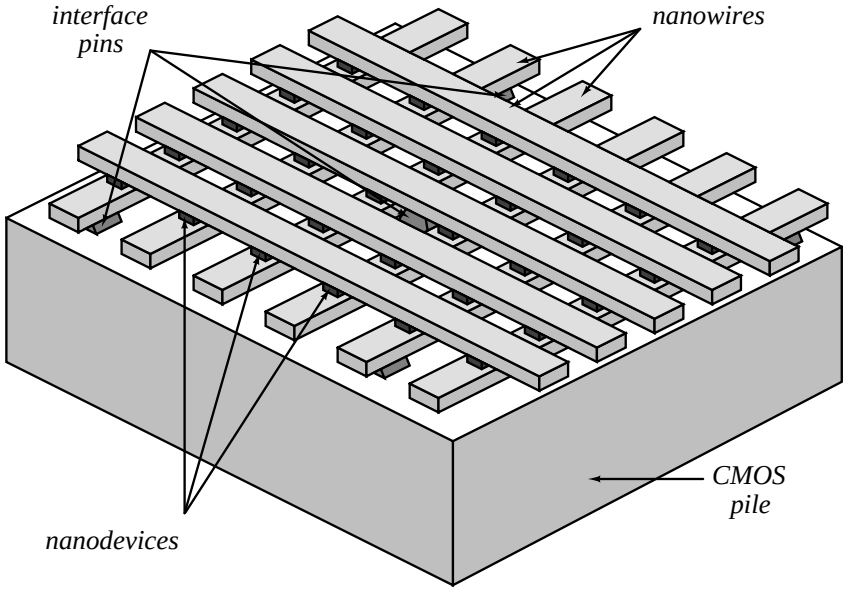


Figure 8.9 Conceptual sketch of CMOL structure

8.2.2 CMOS-Nanodevice Hybrids

Architectures incorporating both nanodevices and CMOS circuits are increasingly important. It is not only a need for an interface between the nanostructures and the outside world. Neither, should it be considered as just a step towards architectures based entirely on nanodevices. As the research has progressed, it has become evident that such hybrid structures have numerous advantages. Indeed, they may turn out to be a necessity due to the shortcomings of nanodevice such as low gain (≈ 1) and small fan-out, which may impose a limitation on system functionality. These obstacles could be overcome by complementing nanocircuits with MOSFET or hybrid structures.

An example of a relatively well developed hybrid approach is the concept called CMOL (as an abbreviation of CMOS/nanowire/MOLecular) [122], [123]. Its structure is schematically presented in Figure 8.9. It consists of an advanced CMOS circuitry, on top of which there are two orthogonal layers of parallel nanowires. At each nanowire crosspoint a two-terminal molecular device e.g., a single-electron latching switch [124], is self-assembled.

The main appeal of CMOL circuits is that they would not require any alignment between nanowire arrays or with the interface pins connecting to CMOS. That opens the door for different fabrication techniques. For instance, nanoimprint and interference patterning have a potential for scaling into a-few-nanometer range, bearing a promise of an enormous device density. However, they come with poor alignment accuracy. To be free of the alignment restrictions, CMOL requires only a special placement of

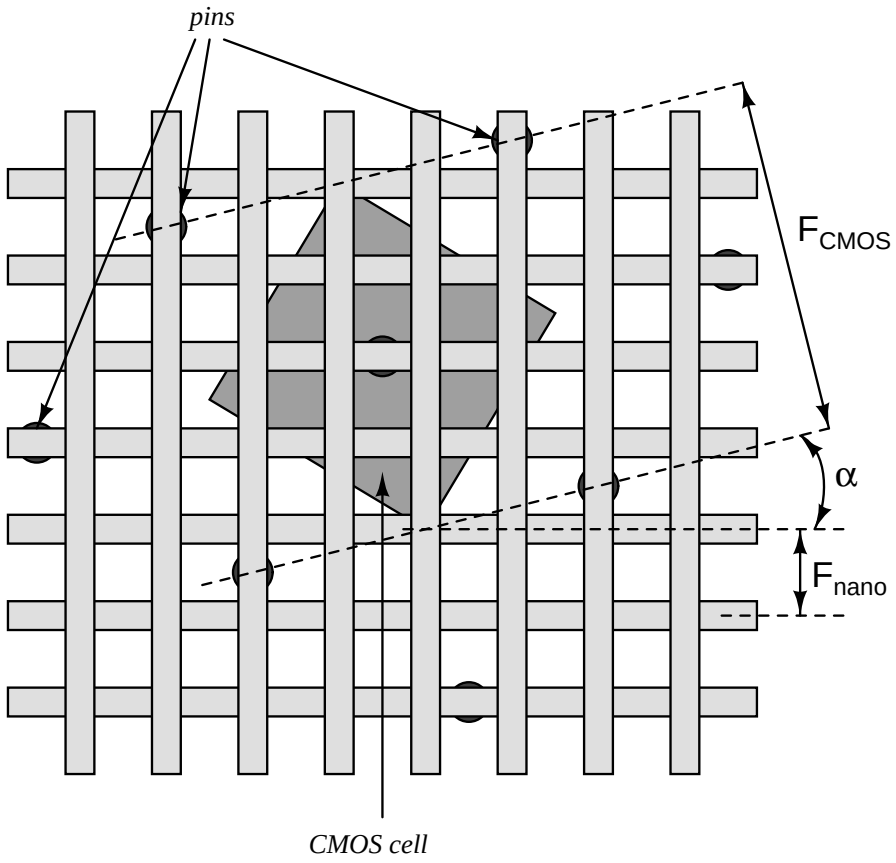


Figure 8.10 Top view of the CMOL structure showing locations of the interface pins. For clarity, only one CMOS cell is marked.

the pins. As seen in Figure 8.10 they are arranged in rectangular grid that is slightly inclined versus the nanowires. In this way, each nanowire can be connected to an individual CMOS pin [3]. Since molecular devices could be formed by chemically-induced self-assembly at the nanowire crosspoints the alignment would be obtained automatically. Moreover, since the nanowires and the molecular devices are identical, a random shift by an integer multiplication of nanowire spacing, F_{nano}, can fully be neglected. Also, a shift by a fraction of nanowire spacing does not necessarily damage the connections.

Possible applications of the CMOL concept are in line with those projected for other nanotechnology structures. The targeted architectures need to be suitable for grid implementations. The restrictions are imposed by the need of the inherent fault

[3]However, to obtain this when the distance between nanowires is many orders of magnitude smaller than the size of the underlying CMOS logic cells, the system would require a large array of CMOS cells. Therefore, the area of a CMOS cell becomes upper bounded, and thus the circuit complexity is limited.

tolerance, which is required to cope with finite yield of molecular devices.
For instance, memories seem to be a natural CMOL applications, since the regular matrix structure ease the implementation of defect tolerance [125], [126]. For the small nanodevice defect ratio (below 10%), the CMOL memories with molecular devices as storage cells and CMOS subsystem used for all other functions (coding, decoding. I/O, etc.) would provide a higher effective bit density than the pure CMOS implementation. With the improvement in fabrication yield, CMOL systems could reach terabit sizes in a single chip with a reasonable die area due to the density limited only by the quantum tunneling between the nanowires and yield in molecule production.
Another potential is in the artificial neural networks. A new family of bio-inspired neuromorphic CMOL networks – Distributed Crosspoint Networks (abbreviated as "CrossNets") have been proposed [127]-[130]. Despite their hardware-induced limitations, the CrossNets can perform a majority of operations typical for neural networks, e.g., pattern classification. Reaching the cell density of the cerebral cortex (10^7 cells per square centimeter) while operating at much higher speed would enable a large group of new applications. Complex image classification, for instance, recognizing a face of a person in a crowd – a task that mammal's brain performs in tens of milliseconds – would only take a few microseconds. Such a great performance could naturally be applied to security systems, production quality control, etc.

This page is intentionally left blank.

Chapter 9

Conclusions and Future Research

In this thesis, hardware implementations of binary cellular neural networks targeting a high spatial resolution were investigated. Processing cells optimized for operations on binary images were designed for CMOS, floating gate MOS and SET technology. Since a high cell density was targeted, an effort was put to make the processing elements compact and power-efficient, and thus to make large array implementations feasible.

The research in this thesis was focused on two aspects. First, the programmability needs of the binary CNN were examined. As images are binary and so is the cell state, it seemed natural to investigate whether the template coefficients could be limited to binary values as well. As a result, a binary programming scheme has been developed. With the proposed binary programming scheme, coefficient circuits are very simple, and thus the cell structure is compact. The template write time becomes very short since the (re-)programming is done digitally. Moreover, these improvements come at no penalty in processor versatility. All B/W operations can be performed with binary templates as more complex tasks are performed algorithmically. This programming scheme also proved to be applicable to structures designed for implementation with SET devices. Most probably, this is the first such a complete and robust programming scheme for architectures based on nanodevices.

The second aspect of this thesis was to develop architectures that utilize the binary programming scheme and are suitable for different technologies, as to demonstrate their feasibility and possible performance. A hardware realization of a binary-programmable CNN was designed for CMOS technology, fabricated and tested. Measurements show that besides the limited programmability this very compact imple-

mention can perform a wide variety of low-level image operations efficiently in terms of both power consumption and speed of operation. A compact structure of programmable transmission gate proved useful in the implementation of a fixed state map as well as in the evaluation of logic functions. The small cell dimensions and low power consumption enable a very large array (even with 10^6 cells) to be implemented on the same chip.

As a step towards nanoimplementations, a binary programmable CNN with cells based on the neuron MOSFET structure was developed. The floating gate of a neuron CMOS inverter is naturally suitable for realization of the voltage-mode weighted summation. The input signals connected via coupling capacitors participate in charge redistribution at the floating node. The resulting potential is detected and thresholded. In this way, a high-gain output nonlinearity is achieved. Since capacitive interconnections consume power only during the computation process, a low static power dissipation is obtained. The implementation of coupling capacitors within a νMOS complementary structure as well as transistors working as switches lead to a compact layout. Again, the binary programming scheme was applied to make the processor versatile, and simulation results confirmed the proper network operation. This approach served as a basis for the development of the effectively programmable CNN implementation based on SET transistors.

The SET technology is among the most thoroughly investigated nanotechnologies. Due to its scaling abilities far beyond CMOS projections and very high energy efficiency, it offers a potential for building ultra dense systems operating at high speed with extremely low power consumption. However, not many designs for SET technology exist. A versatile design of binary CNN cell presented in this thesis is suitable for either pure SET or a hybrid SET-FET implementation. Its accuracy requirements can be achieved with current fabrication technologies and, with further downscaling, room temperature operation can be obtained. This design is probably the first programmable SET CNN and one of the most versatile CNN implementations with nanodevices.

The future work could be focused on the further improvement of the SET CNN design. To take the full advantage of the binary programming scheme, the cell structure should be completed with local RAM and transient mask. Also, a number of other nanotechnology concepts are worth investigating as this research field is very vital and no dominant approach for the post-CMOS era has been selected.

Bibliography

[1] L.O. Chua and L. Yang. Cellular neural networks: theory. *IEEE Transactions on Circuits and Systems – I*, 35(10):1257–1272, 1988.

[2] L.O. Chua and L. Yang. Cellular neural networks: applications. *IEEE Transactions on Circuits and Systems – I*, 35(10):1273–1290, 1988.

[3] L.O. Chua and T. Roska. The CNN paradigm. *IEEE Transactions on Circuits and Systems – I*, 40(3):147–156, 1993.

[4] R. Domínguez-Castro et al. A 0.8-μm CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instruction storage. *IEEE Journal of Solid-State Circuits*, 32(7):1013–1026, 1997.

[5] G. Liñán, S. Espejo, R. Domínguez-Castro, and A. Rodríguez-Vázquez. ACE4k: An analog I/O 64×64 visual microprocessor chip with 7-bit analog accuracy. *International Journal of Circuit Theory and Applications*, 30(2/3):89–116, 2002.

[6] A. Rodríguez-Vázquez et al. ACE16k: The Third Generation of Mixed-Signal SIMD-CNN ACE Chips Toward VSoCs. *IEEE Transactions on Circuits and Systems – I*, 51(5):851–863, 2004.

[7] Á. Zarándy and C. Rekeczky. Bi-i: a standalone ultra high speed cellular vision system. *IEEE Circuits and Systems Magazine*, 5(2):36–45, 2005.

[8] International Technology Roadmap for Semiconductors: 2004 Update. On-line: http://www.itrs.net/Links/2004Update/2004Update.htm, 2004.

[9] International Technology Roadmap for Semiconductors: 2005 Edition. On-line: http://www.itrs.net/Links/2005ITRS/Home2005.htm, 2005.

[10] A. Paasio, M. Laiho, A. Kananen, and K. Halonen. An Analog Array Processor Hardware Realization with Multiple New Features. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1952–1955, 2002.

[11] A. Paasio, J. Flak, M. Laiho, and K. Halonen. High Density VLSI Implementation of a Bipolar CNN with Reduced Programmability. In *Proceedings of 2004 IEEE International Symposium on Circuits and Systems*, volume III, pages 21–24, Vancouver, Canada, 2004.

[12] J. Flak, M. Laiho, A. Paasio, and K. Halonen. VLSI Implementation of a Binary CNN: First Measurement Results. In *Proceedings of the 8th IEEE International Workshop on Cellular Neural Networks and their Applications CNNA 2004*, pages 129–132, Budapest, Hungary, 2004.

[13] J. Flak, M. Laiho, A. Paasio, and K. Halonen. Cell Structure for a Capacitively Coupled Binary CNN. In *Proceedings of the 9th Biennial Baltic Electronics Conference*, pages 103–104, Tallinn, Estonia, 2004.

[14] M. Laiho, A. Paasio, J. Flak, and K. Halonen. Template Design for Binary-Programmable Cellular Nonlinear Networks. In *Proceedings of 2005 IEEE International Symposium on Circuits and Systems*, volume IV, pages 3981–3984, Kobe, Japan, 2005.

[15] J. Flak, M. Laiho, and K. Halonen. Binary Cellular Neural/Nonlinear Network with Programmable Floating-Gate Neurons. In *Proceedings of the 9th IEEE International Workshop on Cellular Neural Networks and their Applications CNNA 2005*, pages 270–273, Hsin-chu, Taiwan, 2005.

[16] J. Flak, M. Laiho, and K. Halonen. FG-MOS neuron for binary CNN. In *Bioengineered and Bioinspired Systems II*, volume 5839 of *Proceedings of SPIE*, pages 314–322, Seville, Spain, 2005.

[17] J. Flak, M. Laiho, A. Paasio, and K. Halonen. Dense CMOS implementation of a binary-programmable cellular neural network. *International Journal of Circuit Theory and Applications*, 34(4):429–443, 2006.

[18] J. Flak, M. Laiho, and K. Halonen. Programmable CNN cell based on SET transistors. In *Proceedings of the 10th IEEE International Workshop on Cellular Neural Networks and their Applications CNNA 2006*, pages 182–185, Istanbul, Turkey, 2006.

[19] J. Flak, M. Laiho, and K. Halonen. On emerging nanodevices and architectures. In *Proceedings of the 10th Biennial Baltic Electronics Conference*, pages 67–70, Tallinn, Estonia, 2006.

[20] M. Laiho, A. Paasio, J. Flak, and K. Halonen. Template design for cellular nonlinear networks with one-bit weights. Accepted to *IEEE Transactions on Circuits and Systems – I*.

[21] Lauri Koskinen. *Analog parallel processor solutions for video encoding*. PhD thesis, Helsinki University of Technology, 2004. Available: http://lib.tkk.fi/Diss/2005/isbn9512279592/.

[22] T. Roska et al. CNN Software Library, Version 1.1. Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Science. On-line: http://lab.analogic.sztaki.hu/Candy/csl.html, 2003.

[23] T. Roska and L.O. Chua. The CNN universal machine: an analogic array computer. *IEEE Transactions on Circuits and Systems – II*, 40(3):163–173, 1993.

[24] S. Espejo, R. Carmona, R. Domínguez-Castro, and A. Rodríguez-Vázquez. A VLSI oriented continuous-time CNN model. *International Journal of Circuit Theory and Applications*, 24:341–356, 1996.

[25] Ari Paasio. *Integration of cellular nonlinear network universal machine*. PhD thesis, Helsinki University of Technology, 1999.

[26] A. Paasio and K. Halonen. A new cell output nonlinearity for dense cellular nonlinear network integration. *IEEE Transactions on Circuits and Systems – I*, 48(3):272–280, 2001.

[27] H. Harrer and J.A. Nossek. Discrete-time cellular neural networks. *International Journal of Circuit Theory and Applications*, 20:453–467, 1992.

[28] K.A. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems – II*, 47(5):416–434, 2000.

[29] R. Forchheimer and A. Åström. Near-sensor image processing: a new paradigm. *IEEE Transactions on Image Processing*, 3(6):736–746, 1994.

[30] A. Åström, R. Forchheimer, and J-E. Eklund. Global feature extraction operations for near-sensor image processing. *IEEE Transactions on Image Processing*, 5(1):102–110, 1996.

[31] J-E. Eklund, C. Svensson, and A. Åström. VLSI implementation of a focal plane image processor – a realization of the near-sensor image processing concept. *IEEE Transactions on Very Large Scale Integration Systems*, 4(3):322–335, 1996.

[32] P. Dudek and P.J. Hicks. A general-purpose processor-per-pixel analog SIMD vision chip. *IEEE Transactions on Circuits and Systems – I*, 52(1):13–20, 2005.

[33] P. Dudek. A 39×48 general-purpose focal-plane processor array integrated circuit. In *Proceedings of 2004 IEEE International Symposium on Circuits and Systems*, volume V, pages 449–452, Vancouver, Canada, 2004.

[34] P. Dudek. Implemntation of SIMD vision chip with 128×128 array of analogue processing elements. In *Proceedings of 2005 IEEE International Symposium on Circuits and Systems*, volume VI, pages 5806–5809, Kobe, Japan, 2005.

[35] A. Lopich and P. Dudek. Architecture of a VLSI cellular processor array for synchnronous/asynchronous image processing. In *Proceedings of 2006 IEEE International Symposium on Circuits and Systems*, pages 3618–3621, Island of Kos, Greece, 2006.

[36] P. Dudek. A flexible global readout architecture for an analogue SIMD vision chip. In *Proceedings of 2003 IEEE International Symposium on Circuits and Systems*, volume III, pages III–782–III–785, Bangkok, Thailand, 2003.

[37] S. Kagami, T. Komuro, and M. Ishikawa. A high-speed vision system with in-pixel programmable ADCs and PEs for real-time visual sensing. In *Proceedings of The 8th IEEE International Workshop on Advanved Motion Control*, pages 439–443, Kawasaki, Japan, 2004.

[38] T. Komuro, S. Kagami, and M. Ishikawa. A dynamically reconfigurable SIMD processor for a vision chip. *IEEE Journal of Solid-State Circuits*, 39(1):265–268, 2004.

[39] A. Paasio, A. Kananen, K. Halonen, and V. Porra. A QCIF Resolution Binary I/O CNN-UM Chip. *Journal of VLSI Signal Processing – Systems for Signal, Image and Video Technology*, 23:281–290, 1999.

[40] A. Stoffels, T. Roska, and L.O. Chua. Object-oriented image analysis for very-low-bitrate video-coding systems using the CNN universal machine. *International Journal of Circuit Theory and Applications*, 25:235–258, 1997.

[41] K. Pagiamtzis and A. Sheikholeslami. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41(3):712–727, 2006.

[42] T. Ikenaga and T. Ogura. A fully parallel 1-Mb CAM LSI for real-time pixel-parallel image processing. *IEEE Journal of Solid-State Circuits*, 35(4):536–544, 2000.

[43] D.G. Elliot, M. Stumm, W.M. Snelgrove, C. Cojocaru, and R. McKenzie. Computational RAM: implementing processors in memory. *IEEE Design & Test of Computers*, 16(1):32–41, 1999.

[44] H. Ai, N. Li, T. Li, M.K. Mandal, and B.F. Cockburn. Efficient parallel implementation of motion estimation on the computational RAM architecture. In *Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering*, pages 609–613, 2002.

[45] M. Sayed and W. Badawy. A new class of computational RAM architectures for real-time MPEG-4 applications. In *Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, pages 328–332, 2003.

[46] P.E. Allen and D.R. Holberg. *CMOS Analog Circuit Design*. Oxford University Press, second edition, 2002.

[47] M. Laiho, A. Paasio, and V. Brea. Effect of mismatch on the reliability of binary-programmable CNNs. In *Proceedings of 2006 IEEE International Symposium on Circuits and Systems*, pages 3622–3625, Island of Kos, Greece, 2006.

[48] V. Brea, M. Laiho, and A. Paasio. Robustness improvement in binary cellular non-linear network architectures. In *Proceedings of the 2005 European Conference on Circuit Theory and Design*, pages I–149–I–152, Cork, Ireland, 2005.

[49] T. Shibata and T. Ohmi. A functional MOS transistor featuring gate-level weighted sum and threshold operations. *IEEE Transactions on Electron Devices*, 39(6):1444–1455, 1992.

[50] A. Rantala, S. Franssila, K. Kaski, J. Lampinen, M. Åberg, and P. Kuivalainen. Improved neuron mos-transistor structures for integrated neural network circuits. *IEE Proceedings on Circuits Devices and Systems*, 148(1):25–34, 2001.

[51] D.V. Averin and K.K. Likharev. Coulomb blockade of single-electron tunneling, and coherent oscillations in small junctions. *Journal of Low Temperature Physics*, 62(3-4):345–373, 1986.

[52] K.K. Likharev. SET: Coulomb blockade devices. *Nano et Micro Technologies*, 3(1-2):71–114, 2003.

[53] J.R. Tucker. Complementary digital logic based on the Coulomb blockade. *Journal on Applied Physics*, 72(9):4399–4413, 1992.

[54] C. Wasshuber, H. Kosina, and S. Selberherr. SIMON – A Simulator for Single-Electron Tunnel Devices and Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(9):937–944, 1997.

[55] R.H. Klunder and J. Hoekstra. Circuit architecture solution for background charge fluctuations in single electron tunneling transistors. In *Proceedings of*

European Conference on Circuit Theory and Design, volume III, pages 213–216, Espoo, Finland, 2001.

[56] Rudie van de Haar. *Simulation of single-electron tunnelling circuits using SPICE*. PhD thesis, Delft University of Technology, 2004.

[57] Martijn Goossens. *Analog Neural Networks in Single-Electron Technology*. PhD thesis, Delft University of Technology, 1998.

[58] K.K. Likharev. Single-electron transistors: Electrostatic analogs of the DC SQUIDS. *IEEE Transactions on Magnetics*, 23(2):1142–1145, 1987.

[59] K.K. Likharev. *Nano and Giga Challenges in Microelectronics*, chapter Electronics Below 10nm, pages 27–68. Elsevier, Amsterdam, Netherlands, 2003.

[60] Roelof Harm Klunder. *Circuit Design with Metallic Single-Electron Tunnel Junctions*. PhD thesis, Delft University of Technology, 2002.

[61] C.P. Heij, P. Hadley, and J.E. Mooij. A single-electron inverter. *Applied Physics Letters*, 78(8):1140–1142, 2001.

[62] T. Oya, T. Asai, Y. Amemiya, A. Schmid, and Y. Leblebici. Single-electron circuit for inhibitory spiking neural network with fault-tolerant architecture. In *Proceedings of 2005 IEEE International Symposium on Circuits and Systems*, volume III, pages 2535–2538, Kobe, Japan, 2005.

[63] H. Inokawa, A. Fujiwara, and Y. Takahashi. A multiple-valued logic with merged single-electron and MOS transistors. In *Technical Digest of International Electron Devices Meeting*, pages 7.2.1–7.2.4, 2001.

[64] H. Inokawa, A. Fujiwara, and Y. Takahashi. A multiple-valued logic and memory with combined single-electron and metal-oxide-semiconductor transistors. *IEEE Transactions on Electron Devices*, 50(2):462–470, 2003.

[65] K. Uchida, J. Koga, R. Ohba, and A. Toriumi. Programmable single-electron transistor logic for low-power intelligent Si LSI. In *2002 IEEE International Solid-State Circuit Conference Digest of Technical Papers*, volume 1, pages 264–266, San Francisco, USA, 2002.

[66] Kyu-Sul Park et al. SOI single-electron transistor with low RC delay for logic cells and SET/FET hybrid ICs. *IEEE Transactions on Nanotechnology*, 4(2):242–248, 2005.

[67] T.A. Fulton and G.J. Dolan. Observation of single-electron charging effects in small tunnel junctions. *Physical Review Letters*, 59(1):109–112, 1987.

[68] L. Zhuang, L. Guo, and S.Y. Cho. Silicon single-electron quantum-dot transistor switch operating at room temperature. *Applied Physics Letters*, 72(10):1205–1207, 1998.

[69] D.H. Kim, J.D. Lee, and B-G. Park. Room temperature coulomb oscillation of a single electron switch with an electrically formed quantum dot and its modeling. *Japan Journal of Applied Physics*, 39(4B):2329–2333, 2000.

[70] A. Fujiwara et al. Single electron tunneling transistor with tunable barriers using silicon nanowire metal-oxide-semiconductor field-effect transistor. *Applied Physics Letters*, 88(053121), 2006.

[71] K. Yano et al. Single-electron memory for giga-to-tera bit storage. *Proceedings of IEEE*, 87(4):633–651, 1999.

[72] H. Inokawa, A. Fujiwara, and Y. Takahashi. A multiple-valued single-electron SRAM by the PADOX process. In *Proceedings of 6th International Conference on Solid-State and Integrated-Circuit Technology*, volume 1, pages 205–208, 2001.

[73] R. Ohba, N. Sugiyama, K. Uchida, J. Koga, and A. Toriumi. Nonvolatile Si quantum memory with self-aligned doubly-stacked dots. *IEEE Transactions on Electron Devices*, 49(2):1392–1398, 2002.

[74] K. Uchida, J. Koga, R. Ohba, and A. Toriumi. Programmable single-electron transistor logic for future low-power intelligent LSI: Proposal and room-temperature operation. *IEEE Transactions on Electron Devices*, 50(7):1623–1630, 2003.

[75] Y. Ono, H. Inokawa, and Y. Takahashi. Binary adders of multigate single-electron transistors: Design using pass-transistor logic. *IEEE Transactions on Nanotechnology*, 1(2):93–99, 2002.

[76] M. Akazawa and Y. Amemiya. Boltzmann machine neuron circuit using single-electron tunneling. *Applied Physics Letters*, 70(5):670–672, 1997.

[77] M.J. Goossens, C.J.M. Verhoeven, and A.H.M. van Roermund. Concepts for ultra-low power and very-high-density single-electron neural networks. In *Proceedings of the International Symposium on Nonlinear Theory and its Applications*, volume 7B-7, pages 679–682, Las Vegas, USA, 1995.

[78] R. van de Haar and J. Hoekstra. Simulation of a neural node using SET technology. In *Proceedings of Fifth International Conference on Evolvable Systems*,

Lecture Notes in Computer Science 2606, pages 377–386, Trondheim, Norway, 2003.

[79] W.S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of mathematical biophysics*, 5:115–133, 1943.

[80] C. Gerousis, S.M. Goodnick, and W. Porod. Toward nanoelectronic cellular neural networks. *International Journal of Circuit Theory and Applications*, 28(6):523–535, 2000.

[81] C. Gerousis, S.M. Goodnick, W. Porod, and A.I. Csurgay. High-speed and low-power cellular non-linear networks using single-electron tunneling technology. In *Proceedings of 2002 IEEE International Symposium on Circuits and Systems*, volume II, pages 45–48, Scottsdale, USA, 2002.

[82] Costa Gerousis. *Design and simulation of cellular non-linear networks using single-electron tunneling transistor technology*. PhD thesis, Arizona State University, 2002.

[83] S.M. Goodnick, C.P. Gerousis, and W. Porod. Simulations of single-electron transistor (SET) CNN cells. In *Proceedings of European Conference on Circuit Theory and Design*, volume III, pages 53–56, Krakow, Poland, 2003.

[84] C.P. Gerousis, S.M. Goodnick, and W. Porod. Nanoelectronic single-electron transistor circuits and architectures. *International Journal of Circuit Theory and Applications*, 32(5):323–338, 2004.

[85] K. Karahaliloğlu, S. Balkır, S. Pramanik, and S. Bandyopadhyay. A quantum dot image processor. *IEEE Transactions on Electron Devices*, 50(7):1610–1616, 2003.

[86] S. Bandyopadhyay, K. Karahaliloğlu, S. Balkır, and S. Pramanik. Computational paradigm for nanoelectronics: self-assembled quantum dot cellular neural networks. *IEE Proceedings on Circuits, Devices and Systems*, 152(2):85–92, 2005.

[87] J.P. Sun, G.I. Haddad, P. Mazumder, and J.N. Schulman. Resonant tunneling diodes: models and properties. *Proceedings of IEEE*, 86(4):641–661, 1998.

[88] M.A. Reed, W.R. Frensley, R.J. Matyi, J.N. Randall, and A.C. Seabaugh. Realization of a three-terminal resonant tunneling device: The bipolar quantum resonant tunneling transistor. *Applied Physics Letters*, 54:1034–1036, 1989.

[89] C.H. Mikkelson, A.C. Seabaugh, E.A. Beam III, J.H. Luscombe, and G.A. Frazier. Coupled-quantum-well field-effect resonant tunneling transistor for multi-valued logic/memory application. *IEEE Transactions on Electron Devices*, 41(2):132–137, 1994.

[90] J. Stock, J. Malindretos, K.M. Indlekofer, M. Pöttgens, A. Förster, and H. Lüth. A vertical resonant tunneling transistor for application in digital logic circuits. *IEEE Transactions on Electron Devices*, 48(6):1028–1032, 2001.

[91] R.J. Aggarwal, K.V. Shenoy, and C.G. Fonstad Jr. A technology for monolithic integration of high-indium-fraction resonant-tunneling diodes with commercial MESFET VLSI electronics. In *Proceedings of the Seventh International Conference on Indium Phosphide and Related Materials*, pages 85–88, 1995.

[92] J.P.A. van der Wagt. Tunneling-based SRAM. *Proceedings of IEEE*, 87(4):571–595, 1999.

[93] J.P.A. van der Wagt, H. Tang, T.P.E. Broekaert, A.C. Seabaugh, and Y.-C. Kao. Multibit resonant tunneling diode SRAM cell based on slew-rate addressing. *IEEE Transactions on Electron Devices*, 46(1):55–62, 1999.

[94] P. Fau et al. Fabrication of monolithically-integrated InAlAs/InGaAs/InP HEMTs and InAs/AlSb/GaSb resonant interband tunneling diodes. *IEEE Transactions on Electron Devices*, 48(6):1282–1284, 2001.

[95] M. Hänggi, R. Dogaru, and L.O. Chua. Physical modeling of RTD-based CNN cells. In *Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications CNNA 2000*, pages 177–182, Catania, Italy, 2000.

[96] R. Dogaru, M. Hänggi, and L.O. Chua. A compact and universal cellular neural network cell based on resonant tunneling diodes: Circuit, model, and functional capabilities. In *Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and their Applications CNNA 2000*, pages 183–188, Catania, Italy, 2000.

[97] M. Hänggi and L.O. Chua. Cellular neural networks based on resonant tunnelling diodes. *International Journal of Circuit Theory and Applications*, 29(5):487–504, 2001.

[98] P. Julián, R. Dogaru, M. Itoh, M. Hänggi, and L.O. Chua. Simplicial RTD-based cellular nonlinear networks. *IEEE Transactions on Circuits and Systems – I*, 50(4):500–509, 2003.

[99] A. Khitun and K.L. Wang. Cellular Nonlinear Network Based on Semiconductor Tunneling Nanostructure. *IEEE Transactions on Electron Devices*, 52(2):183–189, 2005.

[100] S. Iijima. Helical microtubules of graphitic carbon. *Nature*, 354:56–58, 1991.

[101] Miyamoto Y, S. Berber, M. Yoon, A. Rubio, and D. Tománek. Can photo excitation heal defects in carbon nanotubes? *Chemical Physics Letters*, 392:209–213, 2004.

[102] F. Ding, K. Jiao, M. Wu, and B.I. Yakobson. Pseudoclimb and dislocation dynamics in superplastic nanotubes. *Physical Review Letters*, 98(7):075503–1–075503–4, 2007.

[103] Y. Saito. Preparation and properties of carbon nanotubes. In *Proceedings of 1999 International Symposium on Micromechatronics and Human Science MHS '99*, pages 43–49, 1999.

[104] P.G. Collins and P. Avouris. Nanotubes for electronics. *Scientific American*, pages 62–69, December 2000.

[105] D. Rondoni and J. Hoekstra. Toward models for CNT devices. In *Proceedings of 16th Annual Workshop on Circuits, Systems and Signal Processing ProRISC '05*, pages 272–278, 2005.

[106] L.A. Bumm et al. Are single molecular wires conducting? *Science*, 271(5256):1705–1707, 1996.

[107] A. Nitzan and M.A. Ratner. Electron transport in molecular wire junctions. *Science*, 300(5624):1384–1389, 2003.

[108] J. Chen, M.A. Reed, A.M. Rawlett, and J.M. Tour. Large on-off ratios and negative differential resistance in a molecular electronic device. *Science*, 286(5444):1550–1552, 1999.

[109] M. Elbing et al. A single-molecule diode. *Proceedings of National Academy of Sciences of the United States of America*, 102(25):8815–8820, 2005.

[110] D.M. Cardamone, C.A. Stafford, and S. Mazumdar. Molecular transistors based on quantum interference. *SPIE Newsroom*, 10.1117/2.1200701.0587, 2007.

[111] D. Goldhaber-Gordon, M.S. Montemerlo, J.C. Love, G.J. Opiteck, and J.C. Ellenbogen. Overview of nanoelectronic devices. *Proceedings of the IEEE*, 85(4):521–540, 1997.

[112] G.M. Whitesides, J.P. Mathias, and C.T. Seto. Molecular self-assembly and nanochemistry: a chemical strategy for the synthesis of nanostructures. *Science*, 254(5036):1312–1319, 1991.

[113] D.A. Allwood et al. Submicrometer ferromagnetic NOT gate and shift register. *Science*, 296:2003–2006, 2002.

[114] M.C.B. Parish and M. Forshaw. Physical constraints on magnetic quantum cellular automata. *Applied Physics Letters*, 83(10):2046–2048, 2003.

[115] S. Datta and B. Das. Electronic analog of the eletro-optic modulator. *Applied Physics Letters*, 56(7):665–667, 1990.

[116] G.E.W. Bauer, A. Bratas, Y. Tserkonyak, and B.J. van Wees. Spin-torque transistor. *Applied Physics Letters*, 82(22):3928–3930, 2003.

[117] S. Sugahara and M. Tanaka. A spin metal-oxide-semiconductor field-effect transistor using half-metallic-ferromagnet contacts for the source and drain. *Applied Physics Letters*, 84(13):2307–2309, 2004.

[118] D.E. Nikonov and G.I. Bourianoff. Spin gain transistor in ferromagnetic semiconductors – the semiconductor bloch-equations approach. *IEEE Transactions on Nanotechnology*, 4(2):206–214, 2005.

[119] C.S. Lent, P.D. Tougaw, W. Porod, and G.H. Bernstein. Quantum cellular automata. *Nanotechnology*, 4:49–57, 1993.

[120] C.S. Lent and B. Isaksen. Clocked molecular quantum-dot cellular automata. *IEEE Transactions on Electron Devices*, 50(9):1890–1896, 2003.

[121] R.P. Cowburn and M.E. Welland. Room temperature magnetic quantum cellular automata. *Science*, 287(5457):1466–1468, 2000.

[122] K.K. Likharev. Neuromorphic CMOL circuits. In *Proceedings of the Third IEEE Conference on Nanotechnology*, pages 339–342, San Francisco, USA, 2003.

[123] K.K. Likharev. CMOL: a new concept for nanoelectronics. In *Proceedings of the 12th International Symposium "Nanostructures: Physics and Technology"*, pages 339–342, San Francisco, USA, 2004.

[124] S. Fölling, Ö. Türel, and K.K. Likharev. Single-electron latching switches as nanoscale synapses. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 216–221, Washington, USA, 2001.

[125] D. B. Strukov and K. K. Likharev. Prospects for terabit-scale nanoelectronic memories. *Nanotechnology*, 16(1):137–148, 2005.

[126] D. B. Strukov and K. K. Likharev. Defect-tolerant architectures for nanoelectronic crossbar memories. *Journal of Nanoscience and Nanotechnology*, 7(1):151–167, 2007.

[127] Ö. Türel, I. Muckra, and K.K. Likharev. Possible nanoelectronic implementation of neuromorphic networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 365–370, Portland, USA, 2003.

[128] Ö. Türel and K. K. Likharev. CrossNets: Possible neuromorphic networks based on nanoscale components. *International Journal of Circuit Theory and Applications*, 31(1):37–53, 2003.

[129] Ö. Türel, J.H. Lee, X. Ma, and K.K. Likharev. Nanoelectronic neuromorphic networks (CrossNets): New reults. In *Proceedings of the International Joint Conference on Neural Networks*, pages 389–394, Budapest, Hungary, 2004.

[130] Ö. Türel, J.H. Lee, X. Ma, and K.K. Likharev. Neuromorphic architectures for nanoelectronic circuits. *International Journal of Circuit Theory and Applications*, 32(5):277–302, 2004.

Appendix A

Binary Template Library

A.1 B-Templates Performing Neighborhood Logic OR

Object Increase

$$increase = \varphi(AB * IN - 0.5) \tag{A.1}$$

$$AB = B_{increase} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.2}$$

Dilation

$$dilation = \varphi(AB * IN - 0.5) \tag{A.3}$$

where nonzero entries in $AB = B_{dilation}$ determine structuring element of dilation.

Erosion

$$erosion = \overline{\varphi(AB * \overline{IN} - 0.5)} \tag{A.4}$$

where nonzero entries in $AB = B_{erosion}$ determine structuring element of erosion.

Peel

$$peel = \overline{\varphi(AB * \overline{IN} - 0.5)} \tag{A.5}$$

where nonzero entries in $AB = B_{peel}$ determine how the object should be peeled. For instance:

Left Peeler

$$AB = B_{peelleft} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.6}$$

A.2 B-Templates Using Transient Mask

Point Removal

$$pointrem = Y_e(AB, IN, IN, \overline{IN}, 0.5) \tag{A.7}$$

$$AB = B_{pointrem} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.8}$$

Point Extraction

$$pointextr = \overline{Y_{\bar{e}}(AB, IN, IN, \overline{IN}, 0.5)} \tag{A.9}$$

$$AB = B_{pointextr} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.10}$$

Edge Detection

$$edge = Y_e(AB, \overline{IN}, 0, \overline{IN}, 0.5) \tag{A.11}$$

$$AB = B_{edge} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.12}$$

Line Removal

$$linrem = IN \oplus Y_e(AB, IN, IN, \overline{IN}, 0.5) \tag{A.13}$$

where $\oplus$ denotes the XOR operator, border is white, and the locations of nonzero terms in AB define the direction of lines to be removed. For instance:

Diagonal Line Removal

$$AB = B_{deldiag} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{A.14}$$

Vertical Line Removal

$$AB = B_{delvert} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.15}$$

Horizontal Line Removal

$$AB = B_{delhoriz} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.16}$$

A.3 B-Templates Using Pattern Matching

General Pattern Matching

$$match = \overline{Y(AB^{(w)}, IN, 0, \varphi(AB^{(b)} * \overline{IN} - 0.5), 0.5)} \tag{A.17}$$

where the nonzero entries in $AB^{(b)}$ ($AB^{(w)}$) detect black (white) pattern pixels.

Diagonal Detection

$$detdiag = match \tag{A.18}$$

$$\begin{aligned} AB^{(b)} = B^{(b)}_{detdiag} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ AB^{(w)} = B^{(w)}_{detdiag} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{A.19}$$

Right Edge Detection

$$rightedge = match \tag{A.20}$$

$$\begin{aligned} AB^{(b)} = B^{(b)}_{rightedge} &= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ AB^{(w)} = B^{(w)}_{rightedge} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{A.21}$$

Local Concave Place Detection

$$lcp = Y_e(AB1, IN, 0, \overline{match}) \tag{A.22}$$

$$\begin{aligned} AB^{(b)} = B^{(b)}_{lcp} &= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ AB^{(w)} = B^{(w)}_{lcp} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ AB1 = B1_{lcp} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{aligned} \tag{A.23}$$

Skeleton NE

$$skeleton_{NE} = IN \oplus match \tag{A.24}$$

$$\begin{aligned} AB^{(b)} = B^{(b)}_{skeleton\_NE} &= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ AB^{(w)} = B^{(w)}_{skeleton\_NE} &= \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{A.25}$$

Skeleton N

$$skeleton_N = IN \oplus Y_e(AB2, IN, 0, \overline{match}) \tag{A.26}$$

$$\begin{aligned} AB^{(b)} = B^{(b)}_{skeleton\_N} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ AB^{(w)} = B^{(w)}_{skeleton\_N} &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ AB2 = B2_{skeleton\_N} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \end{aligned} \tag{A.27}$$

A.4 B-Templates Using Threshold Logic

Junction Extraction

$$junction = Y_e(AB, IN, IN, \overline{IN}, 2.5) \tag{A.28}$$

$$AB = B_{junction} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.29}$$

Corner Detection

$$corner = \overline{Y_{\bar{e}}(AB, IN, IN, \overline{IN}, 3.5)} \tag{A.30}$$

$$AB = B_{corner} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.31}$$

A.5 A-Templates Computing with Propagating Wave

Selected Object Extraction, aka **Figure Reconstruction**

$$figrec_{t+1} = Y_{t+1}(AB, IN2, \overline{IN1}, 0.5) \tag{A.32}$$

$$AB = A_{figrec} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.33}$$

Hole Filler

$$holefil_{t+1} = \overline{Y_{t+1}(AB, 0, IN, 0.5)} \tag{A.34}$$

$$AB = A_{holefil} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.35}$$

Concave Location Filler, aka **Hollow**

$$hollow_{t+1} = Y_{t+1}(AB, IN, IN, 3.5) \tag{A.36}$$

$$AB = A_{hollow} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.37}$$

Global Connectivity Detection

$$connectivity_{t+1} = IN1 \oplus Y_{t+1}(AB, IN1 \oplus IN2, \overline{IN1}, 0.5) \tag{A.38}$$

$$AB = A_{connectivity} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{A.39}$$

A.6 A-Templates with Positive and Negative Coefficients

Concentric Contour Detection

$$\begin{array}{rl} 1^{\text{st}} \text{ contour} & \begin{cases} B1 & = \varphi(AB, \overline{IN}, 0.5) \\ M1 & = \overline{IN} \oplus B1 \end{cases} \\ 2^{\text{nd}} \text{ and following contours} & \begin{cases} B2 & = \varphi(AB, B1, 0.5) \\ B3 & = \varphi(AB, B2, 0.5) \\ M1 & = M1 \vee (B2 \oplus B3) \end{cases} \end{array} \tag{A.40}$$

$$AB = B_{conccont} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.41}$$

Connected Component Detection, aka **Horizontal Hole Detection** (one iteration)

$$\begin{array}{rl} B1 & = Y_{\overline{e}}(AB1, IN, IN, \overline{IN}, 0.5) \\ B2 & = Y_{e}(AB2, \overline{B1}, \overline{B1}, \overline{B1}, 0.5) \\ B3 & = Y_{\overline{e}}(AB3, \overline{B1}, B2, B2, 0.5) \\ OUT & = Y_{e}(AB4, \overline{B1}, B3, B3, 0.5) \end{array} \tag{A.42}$$

$$\begin{array}{rl} AB1 = & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ AB2 = & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ AB3 = & \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ AB4 = & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \tag{A.43}$$

ISBN 978-951-22-8853-3
ISBN 978-951-22-8854-0 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)