2005

Novel arithmetic implementations using cellular neural network arrays.

Youssef Ibrahim
University of Windsor

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Novel Arithmetic Implementations Using Cellular Neural Network Arrays

by

Youssef Ibrahim

A Dissertation
Submitted to the Faculty of Graduate Studies and Research through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the University of Windsor

Windsor, Ontario, Canada

2005

Abstract

The primary goal of this research is to explore the use of arrays of analog self-synchronized cells - the cellular neural network (CNN) paradigm - in the implementation of novel digital arithmetic architectures. In exploring this paradigm we also discover that the implementation of these CNN arrays produces very low system noise; that is, noise generated by the rapid switching of current through power supply die connections - so called di/dt noise. With the migration to sub 100 nanometer process technology, signal integrity is becoming a critical issue when integrating analog and digital components onto the same chip, and so the CNN architectural paradigm offers a potential solution to this problem. A typical example is the replacement of conventional digital circuitry adjacent to sensitive bio-sensors in a SoC Bio-Platform. The focus of this research is therefore to discover novel approaches to building low-noise digital arithmetic circuits using analog cellular neural networks, essentially implementing asynchronous digital logic but with the same circuit components as used in analog circuit design.

We address our exploration by first improving upon previous research into CNN binary arithmetic arrays. The second phase of our research introduces a logical extension of the binary arithmetic method to implement binary signed-digit (BSD) arithmetic. To this end, a new class of CNNs that has three stable states is introduced, and is used to implement arithmetic circuits that use binary inputs and outputs but internally uses the BSD number representation. Finally, we develop CNN arrays for a 2-dimensional number representation (the Double-base Number System - DBNS). A novel adder architecture is described in detail, that performs the addition as well as reducing the representation for further processing; the design incorporates an innovative self-programmable array. Extensive simulations have shown that our new architectures can reduce system noise by almost 70dB and crosstalk by more than 23dB over standard digital implementations.

To: **My parents,**

my wife, Eman,

and my daughter, Lauren

Acknowledgments

I would like to acknowledge all the people who assisted and supported me over the years of my graduate study at the University of Windsor. Although this thesis prominently bears my name, it would not have been possible without their contribution.

I would like to express my gratitude to my advisors, Dr. G.A. Jullien and Dr. W.C. Miller for providing guidance and support. Aside from keeping me on track, they have also allowed me the freedom to pursue research at my own pace. I would like to extend thanks to the members of my committee, Dr. N. Yazdi, Dr. A. Ngom, Dr. C. Chen, and Prof. M. Ahmadi whose insightful comments and helpful suggestions have improved this thesis. I am also grateful to Dr. R. Muscedere who helped me use system-level CAD tools.

I am indebted to the RCIM lab manager Mr. Till Kuendiger who was always there whenever I needed help. I appreciate my colleagues in the RCIM research group, my friends in all other research groups, and the entire Electrical and Computer Engineering staff. They were a constant source of encouragement and optimism over the years.

I would like to acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada, the Micronet Network of Centres of Excellence, and Gennum Corporation. Furthermore, I am obliged to CMC Microsystems (formerly The Canadian Microelectronics Corporation) for providing design tools, workstations and fabrication services.

Finally, I would like to thank my parents, Salah and Nadia, for their never-fading love and encouragement. I am also grateful to my surrogate parents, Larry and Doreen Cheshire, and to my surrogate parents-in-law, Joseph and Betty Tomc, for their love and assistance during my stay in Windsor. I am indebted to my church family at Campbell Baptist Church for their moral support throughout this research project. I am very grateful to my wife, Eman, and my daughter, Lauren, for their love, inspiration, and understanding through the very difficult moments in my studies.

Table of Contents

List of Figures

List of Tables

List of Symbols

Δ	Delta operator
ρ	Substrate resistivity
ε	Substrate permittivity
α	Self-feedback current coefficient
$\bar{1}$	Integer digit -1
$\bar{2}$	Integer digit -2
$\pi(i)$	Permutation i
$\bar{x}_i$	Two's complement of digit at position i of operand X
Γ	Number of full adder levels in binary reduction tree
μ	Micron
A_c	Output feedback template
$A_{ij;kl}$	Element kl of the output feedback template A for the CNN cell C(i,j)
B_c	Input control template
C	Carry out of an addition
$C(i,j)$	CNN cell at position (i,j)
$Ceff$	Effective parasitic capacitance
c_i	Digit at position i of the carry in/out C
C_x	Input capacitance of a CNN cell
I	Bias of a CNN cell
In_i	Input i to median extractor

i_{sub}	Substrate current
I_{unit}	Unit current
I_{VDD}	Instantaneous supply current
$I_x(i,j)$	DBNS representation of operand X
I_y	Output current of a CNN cell
$I_y(i,j)$	DBNS representation of operand Y
$I_z(i,j)$	DBNS representation of sum out Z
k	Average number of nonzero digits
L	Transistor length
L	Digit set of the binary signed digit
L_{eff}	Effective parasitic inductance
m	Milli
m	Meter
M	Number of rows in a CNN grid
m_i	Scaling factor of a current mirror
n	Nano
n	Number of digits in a number
N	Number of columns in a CNN grid
$N(i,j)$	Neighborhood of CNN cell C(i,j)
O_{minus}	Negative output current of reduction rule
O_{plus}	Positive output current of reduction rule
p	Pico
P	Product of multiplication
$p_{i,j}$	Partial product of digit i of operand X and digit j of operand Y
P_j	Product of operand X and digit at position j of operand Y
$q(r,t)$	Transient charge density
r	Radius of the CNN neighborhood
R_i	Resistor i
R_x	Input resistance of a CNN cell
R_y	Output resistance of a CNN cell
S	Sum out of an addition
s_i	Digit at position i of the sum out S
T	Time constant of a CNN cell
T_{delay}	Maximum delay of DBNS adder

t_i	Transfer digit at position i
u_{ij}	Input to CNN cell C(i,j)
$V(r,t)$	Transient voltage vector
V_{bias}	Bias voltage
V_{DD}	Power supply voltage
V_{eff}	Effective voltage on chip
V_{in}	Input voltage
V_{out}	Output voltage
V_{ref}	Reference voltage
V_{state}	State voltage of a CNN cell
W	Transistor width
w_i	Intermediate sum at position i
X	An operand to an arithmetic operation
x_i	Digit at position i of the operand X
$x_{i,j}$	Digit in position (i,j) in a DBNS number
x_{ij}	State voltage of CNN cell C(i,j)
Y	An operand to an arithmetic operation
y_i	Digit at position i of the operand Y
y_{ij}	Output voltage of CNN cell C(i,j)
Z	Instantaneous sum out
z_i	Digit at position i of the instantaneous sum out Z

List of Abbreviations

2D	Two dimensional
A/D	Analog-to-digital
ARDBNR	Addition-ready double-base number representation
BC	Before Christ
BSD	Binary signed-digit
BSDNS	Binary signed-digit number system
CDBNS	Canonic double-base number system
CD-ROM	Compact disk-Read only memory
CMOS	Complementary metal-oxide semiconductor
CNN	Cellular neural network
CNNBFA	CNN-based binary full adder
CNNBSDFA	CNN-based signed-digit full adder
CNNDBNSAU	CNN-based double-base adder unit
CNN-UM	Cellular neural network universal machine
CSD	Canonic signed-digit
CT-CNN	Continuous time CNN
DBNS	Double-base number system
DC	Direct current
DSN	Digital switching noise
DSP	Digital signal processing
DT-CNN	Discrete time CNN
ECC	Elliptic curve cryptography
FA	Full adder

FIR	Finite impulse response
FLP	Floating-point
IC	Integrated circuit
IDBNS	Index calculus double-base number system
IEEE	Institute of Electrical and Electronics Engineers
IIR	Infinite impulse response
JSF	Joint sparse form
LNS	Logarithmic number system
LSB	Least significant bit
MAC	Multiply-accumulate
MAF	Multiplication-add fused
MCM	Multiple constant multiplication
MDLNS	Multidimensional logarithmic number system
MSB	Most significant bit
NCDBNR	Near-canonic double-base number representation
NN	Neural network
PS	Porous silicon
RF	Radio frequency
SD	Signed-digit
SDFA	Signed-digit full adder
SDNS	Signed-digit number system
SoC	System-on-a-chip
SOI	Silicon-on-insulator
SSN	Simultaneous switching noise
TFSOI	Thin-film silicon-on-insulator
VCCS	Voltage controlled current source
VLSI	Very large scale integrated (or integration)

Chapter 1

Introduction

The objective of this research project is to develop novel arithmetic circuit structures using arrays of analog networks and the cellular neural network paradigm. The motivation behind this work contains both an exploration of this novel concept to a variety of arithmetic techniques and a more practical investigation into the use of these networks for implementing arithmetic circuits that produce very low di/dt noise as explained in Section 1.1. The digital switching noise and cross talk problems are defined in Section 1.2. A review of the research literature that addresses the noise problem is also presented in this section. Reasons to use the CNN paradigm are discussed in Section 1.3 and the state-of-the-art CNN-based binary adders are presented in Section 1.4. The design goals of the arithmetic circuits are introduced in Section 1.5. The thesis is outlined in Section 1.6 and Section 1.7.

1.1 Low-noise - motivation

The migration to sub 100 nanometer process technologies and the advances in fabrication processes have allowed the packing of ever increasing complex functionality into fewer chips on circuit boards by the use of massive integration of circuitry onto single silicon die using system-on-a-chip (SoC) tools and technologies. Despite the

impressive reductions in area and cost, signal integrity remains a critical issue when integrating analog and digital components onto the same chip; for example digital circuitry adjacent to sensitive bio-sensors in a SoC Bio-Platform as depicted in Figure 1.1. One of the parasitic effects that adversely influences signal integrity is digital switching noise (DSN) produced by the supply current drawn by fast switching digital components. This noise propagates across the common Si substrate and can easily corrupt sensitive analog signals. Noise can also couple to the substrate capacitively, a phenomenon known as crosstalk. The noise problem is amplified as operating frequencies increase and feature sizes decrease; being primarily responsible for inexplicable design failure and poor yields of mixed-signal SoC designs [1].

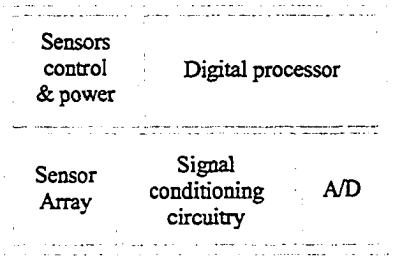


Figure 1.1 Simplified block diagram of a bio-sensor SoC chip.

The work presented in this thesis couples different digital number representations with low precision simple current-mode analog components in a novel way that combines the computational capability of analog circuits and noise immunity of digital components. In essence, we are building digital arithmetic circuits but using analog components to replace uncontrolled digital transitions (produced by the digital processor of Figure 1.1) with smooth analog transitions as illustrated in Figure 1.2. Digital arithmetic is converted into a problem of processing 2-D binary/ternary images and novel CNN structures are designed to manipulate these images to perform the required arithmetic task.

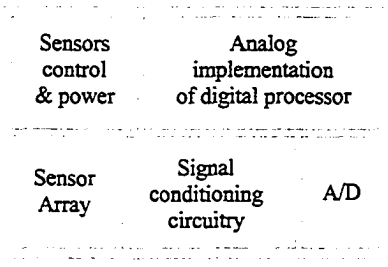


Figure 1.2 Reducing system noise in the bio-sensor with smooth analog transitions.

1.2 Substrate Noise

Digital switching noise is one of the major sources of trouble in a typical mixed-signal VLSI circuit design. When many static gates change state together, they draw a large cumulative current from the power supply. Due to the self-inductance of the off-chip bonding wires and package pins and the on-chip parasitic inductance inherent to the power supply rails, as shown in Figure 1.3, the fast current surges result in voltage fluctuations in the power distribution network [2]. The effective supply voltage on chip is given by the following equation:

$$V_{eff} = V_{DD} - L_{eff}\frac{dI_{V_{DD}}}{dt} \tag{1.1}$$

The second term on the right hand side of Eqn. (1.1) is referred to as digital switching noise (DSN), simultaneous switching noise (SSN), inductive—*Ldi/dt*—noise, or ΔI. A fraction of this noise is invariably injected into the substrate.

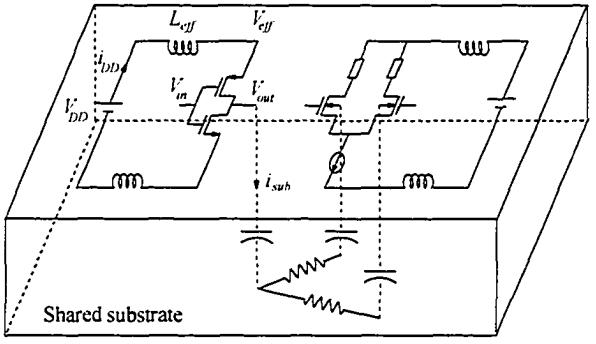


Figure 1.3 Lumped model of the substrate coupling.

The presence of parasitic capacitance between the transistors and the silicon substrate contributes significantly to the problem. When digital circuits switch, they inject current into the substrate via these capacitances. The amount of injected current is directly proportional to the slew rate of the switching voltage and the lumped parasitic capacitance according to Eqn. (1.2); this will be referred to as "cross-talk noise".

$$i_{sub} = C_{eff}\frac{dV_{out}}{dt} \tag{1.2}$$

Therefore, substrate noise increases as the operating frequency increases. Moreover, the scaling down of feature sizes increases the total capacitance associated with the internal circuitry [3]. With the number of transistors on a chip expected to reach over 600 million by 2009 [4], the amount of injected noise increases dramatically.

The common substrate on which both digital and analog circuits are embedded serves as a resistor network which can be modelled using a simplified form of Maxwell's equations [1]:

$$\frac{1}{\rho}\nabla^2 V(r,t) + \varepsilon\frac{\partial}{\partial t}(\nabla^2 V(r,t)) = -\frac{\partial}{\partial t}q(r,t) \tag{1.3}$$

where ρ is the resistivity and ε the permittivity of the uniformly-doped semiconductor. $V(r, t)$ is the transient voltage vector and $\frac{\partial}{\partial t}q(r, t)$ is the rate of generation of charge per unit volume at location $r = (x, y, z)$ on the substrate. Consequently, voltage variations around the injected points propagate in the substrate and also potential gradients arise due to the resistive nature of the substrate. Assuming a 3-D semi-infinite substrate that goes to infinity in all but one of the six spatial directions, the solution to Eqn. (1.3) in the Laplace domain for the voltage at any point on the substrate, V_2 due to a current, i_1 injected into the substrate a distance r away, is:

$$V_2(s) = \frac{\rho}{2\pi r} \cdot \frac{i_1(s)}{s(\rho \cdot \varepsilon) + 1} \tag{1.4}$$

The time variant substrate voltages are sensed by MOSFETs through the body effect and transferred to signal paths in consequence of current fluctuations or gain mismatches in analog circuits. On the other hand, sub-threshold current increase due to the body bias change may degrade digital signal integrity seriously and thus cause dynamic operation failures [5]. On-chip DSN can also create delay uncertainty since the power supply level temporally changes the local drive current [6]. Furthermore, logic malfunctions may be created and excess power may be dissipated due to faulty switching if the power supply fluctuations are sufficiently large [7][8]. Predicting how and when this will happen is a difficult problem, since it is highly dependent on the specific layout and process technology used. Therefore, it is crucial in today's sub 100 nanometer technology to reduce the peak value of dynamic current provided by the supply source i_{DD}, that is proportional to the carrier injection into the substrate [9]. Moreover, by monitoring the instantaneous power supply current, designers can determine a time window for the worst-case substrate current injection [1]. For example, Figure 1.4 shows the Hspice simulation of a standard digital inverter in 0.35μm CMOS technology during one period of excitation. The simulation also illustrates the instantaneous power supply current drawn by the inverter, digital switching noise, and cross talk noise.

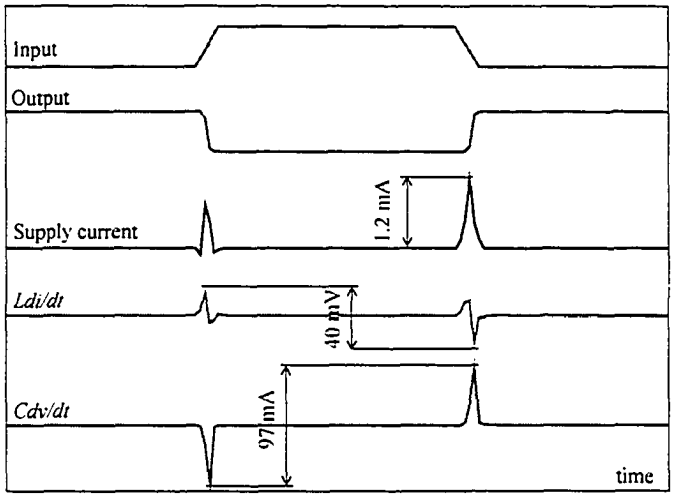


Figure 1.4 Hspice simulation of a standard digital inverter.

The rapidly growing SoC market presents an urgent need for highly effective solutions for the substrate noise problem. Research in this area can be broadly classified into three main categories: Process fabrication techniques, physical design and layout techniques, and innovative digital circuit design techniques.

Process fabrication techniques: The goal of process technologists, regarding the noise problem, is to prevent the substrate from working as a noise-coupling path by increasing its resistance, ultimately to infinity. They employ a number of expensive techniques to control the substrate noise problem. One technique employs a deep trench (through-the-wafer) of porous Si (PS) to provide radio frequency (RF) isolation in Si between noise generating and noise sensing circuits [10]. Traditional guard-rings [11][12] have very limited effectiveness in suppressing the underlying substrate noise due to the fact that they are very shallow structures on the wafer surface. However, a Faraday cage consisting of a ring of high-aspect ratio substrate vias encircling noisy or sensitive circuits results in improved performance [13]. A popular technique involves creating a deep N-well structure where active devices are insulated from the substrate by a buried implant layer [12]. Experimental results indicated that an improvement of 25-30dB can be achieved by applying a relatively low-fluence proton bombardment on the isolation-intended region

[14]. The expensive option based on silicon-on-insulator (SOI) wafers assures full DC isolation; however, it fails to maintain its advantage in the high frequency AC regime [15]. Still another technique is to deploy more costly thin-film silicon–on–insulator (TFSOI) technology where p+ substrate contact rings are used to improve the cross-talk isolation [16][17]. These techniques can help improve the intrinsic noise immunity of SoC devices, but have limited effectiveness at elevated frequencies. Thus, although the use of one or more of these fabrication techniques can reduce substrate noise, process remedies alone are insufficient to ensure a design's immunity to substrate noise coupling [18].

Physical design and layout techniques. Physical design and layout techniques concerning noise immunity primarily aim to reduce the parasitic inductance associated with the power supply network and package pins, minimize the parasitic capacitance between transistors and the substrate, and attenuate noise coupling from one area on a chip to another. It is common practise among analog designers to use separate supplies for digital and analog sections of the chip to isolate the sensitive analog components from noise introduced on the digital supplies [19]. The same technique is useful to isolate different sensitive blocks. Dividing a chip into sections with different substrate grounds will mitigate noise coupling [20]. Researchers have also found that using multiple digital and analog pins can achieve the largest noise reduction. Decreasing the value of the inductance of the bonding wire widens the bottleneck which reduces ground noise [2]. Proper choice of substrate contact geometry and placement plays a major rule in substrate noise distribution [21] and a careful design of power lines geometry and supply network distribution can greatly reduce parasitic inductance [22]. While relative placement of the logic and analog blocks affects the amount of noise coupling [19], analog layout techniques such as mirror symmetry and common-centroid geometries increases noise immunity of analog circuitry [9]. Adding a dedicated backplane substrate contact can substantially drain injected noise [23]. A first and excellent experimental study on the impact of physical design on substrate coupling noise is presented in [24] and a wealth of industry examples to highlight isolation impacts of technology can be found in [25].

Digital design techniques. Regardless of measures taken to minimize noise coupling from digital sections to analog sections on a chip, digital circuitry can still produce significant transient noise. For example, the Ldi/dt noise is estimated to reach 0.35 volt in 0.1μm CMOS technology with 1.2 volt power supply [26]. This peak noise seriously degrades signal integrity and can easily cause dynamic operation failure. Therefore minimizing on-chip noise is an important element in the effort to improve a design's noise immunity in high performance mixed-signal integrated circuits. Some techniques have been reported that can reduce the effect of DSN. Adding decoupling capacitance can reduce the amount of noise created by supplying local charge for nearby switching and thus lowering the peak current drawn across the package inductance [27]. Building a simple RC filter can leak out DSN with selected frequency roll-off [28]. A negative feedback loop can also be formed by sampling the noise and re-injecting it into the substrate with inverted phase. This inverted noise can reduce the substrate noise for low frequency operations [29]. Using divided switches with current control can also reduce switching noise by controlling the current slope [30]. A number of low-switching-noise digital CMOS families have also been reported: current steering logic [31], folded source-coupled logic [32], NMOS current-balanced logic [33], and cellular neural networks [34][35]. However, static power consumption is the main penalty of such structures. Furthermore, some actions at the system level can be taken to minimize switching activities, for instance alternative architectural allocation and scheduling [9], reducing switching activity by pin swapping [36], and the right choice of the clocking scheme [37].

1.3 CNN-based Arithmetic Circuits - Rationale

Among all of the above methods, the use of cellular neural networks is quite interesting for several reasons:

- uses analog circuit blocks with inherently lower system noise,
- additional noise reduction due to the asynchronous nature of the CNN arrays,
- noise reduction is independent of the traditional noise reduction methods (e.g., guard rings) and thus can be used in combination with them, and

- the regular structure of the arrays and locality of connections makes it an excellent choice for VLSI implementation.

CNN arrays inherently reduce system noise because their current-mode structures operate with almost constant supply current, thus reducing variation in supply current and, hence, switching noise. Moreover, the nodes are built with analog building blocks and effectively provide controlled slewing. In digital logic, by contrast, the output of logic gates switch rapidly between logic states; this switching rate of change being independent of the clock rate of the input logic signals. To illustrate this idea, Hspice simulations of instantaneous values of Ldi/dt and Cdv/dt from a standard CMOS static digital inverter are compared to those from a CNN cell (which can be used as an inverter by forcing its operation in the saturation mode and using the negative output of the cell) in Figure 1.5 and Figure 1.6 respectively. A parasitic inductance of 1nH is used to calculate the DSN while a parasitic capacitance of 2pF is used in Eqn. (1.2). The CNN circuit clearly significantly suppresses the noise in both cases. The advantage of using CNN arrays becomes greater as the circuit size increases.

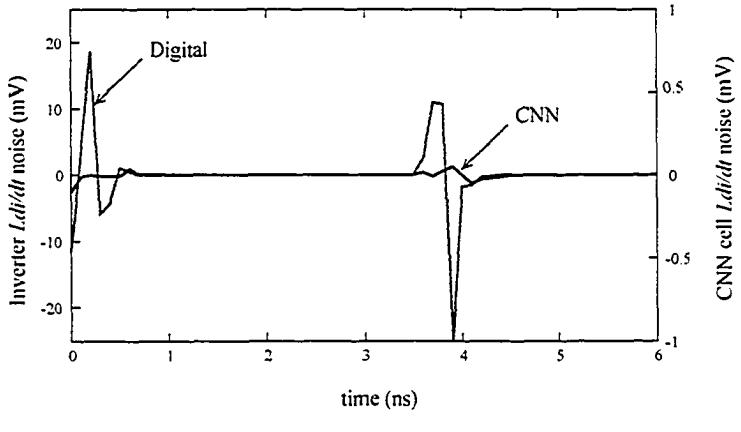


Figure 1.5 Hspice simulation of di/dt for a digital inverter and a CNN cell.

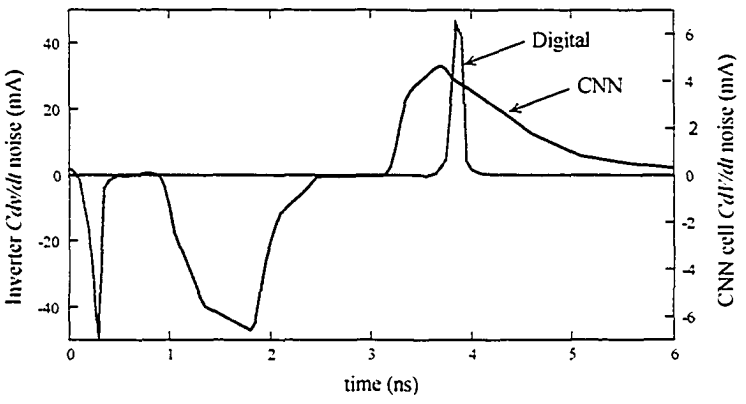


Figure 1.6 Hspice simulation of dv/dt for a digital inverter and a CNN cell.

In addition, CNN arrays permit a direct trade off between speed and cross talk (see Section 2.3 for details). By increasing the integration time constant, the slope of the output voltage is decreased and, hence, cross talk. Figure 1.7 shows Hspice simulations of the output voltage of a CNN for different time constants.

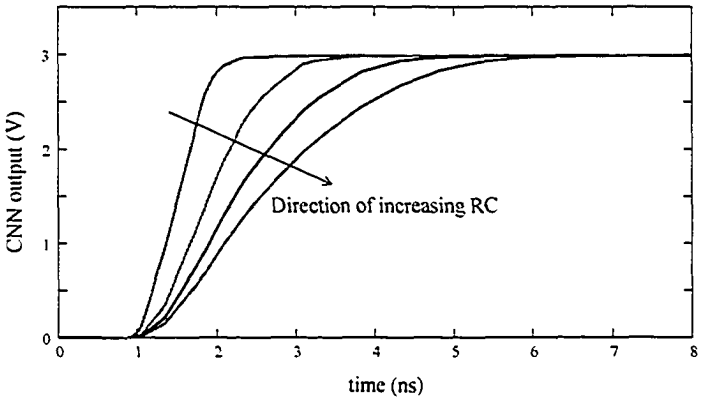


Figure 1.7 Hspice simulation of a CNN cell output voltage for different time constants.

1.4 Existing CNN-based Arithmetic Circuits

Two CNN-based binary adders, the flat adder and the recursive adder, were previously introduced in [34] and [35] respectively. In the flat structure, the addition of two N-bit binary numbers $A = a_N a_{N-1} \ldots a_2 a_1$ and $B = b_N b_{N-1} \ldots b_2 b_1$ is performed through successive conversion of the given addition operation to another equivalent addition using the rules:

$$c_i^1 = \begin{cases} 0 & i = 1 \\ a_{i-1} \wedge b_{i-1} & 2 \le i \le N+1 \end{cases} \tag{1.5}$$

$$d_i^1 = \begin{cases} a_i \oplus b_i & N \le i \le 1 \\ 0 & i = N+1 \end{cases} \tag{1.6}$$

As a result of each conversion step, one digit of the result is obtained and, in the worst case, the complete result is obtained in $N+1$ steps. Implementing this algorithm in CNN required assigning a row of $N+1$ cells for each original operand. Considering that in the worst case $N+1$ steps are needed to complete the addition, the network consisted of $2(N+1)$ rows. However, as addition proceeds, more digits of the first operand become zero and more digits of the second operand attain final value. Therefore, cells corresponding to these digits are not required in the CNN implementation. The optimized network structure consists of $N(N+1)$ cells as shown in Figure 1.8 for the addition of two 4-bit numbers. Two major drawbacks of this design are the huge silicon area required, that increases with $o(n^2)$, and consequently large power consumption.

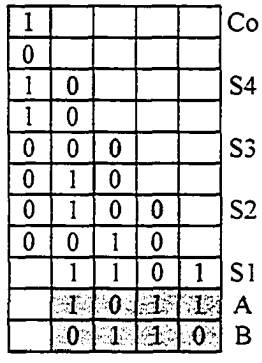


Figure 1.8 The flat binary adder.

To save on silicon area, the author implemented a recursive adder in [35] that requires four rows of $N+1$ cells. In this structure, the information is allowed to flow from the first two rows back to the last two rows. Unlike the flat structure which performs quite robustly for a wide range of template values, this parameter here should be chosen carefully to avoid divergence due to a potential race problem. The structure also lacks speed and stability due to the recursive operation as shown in the simulation of Figure 1.9. As addition proceeds, the height of the generated carries gradually falls. This indicates that the adder will eventually diverge from correct sum outputs for large operands. To alleviate this problem, the author suggested applying a positive velocity vector across the array from the LSB to the MSB. This was implemented by decreasing the value of the self-feedback factor α by 0.05 per bit position. This fine-tuning method can work with MATLAB simulation as variables can be decremented virtually by any small value. However, its realization is almost impossible because the analog designer is restricted by the physical constraints imposed by the technology being used.

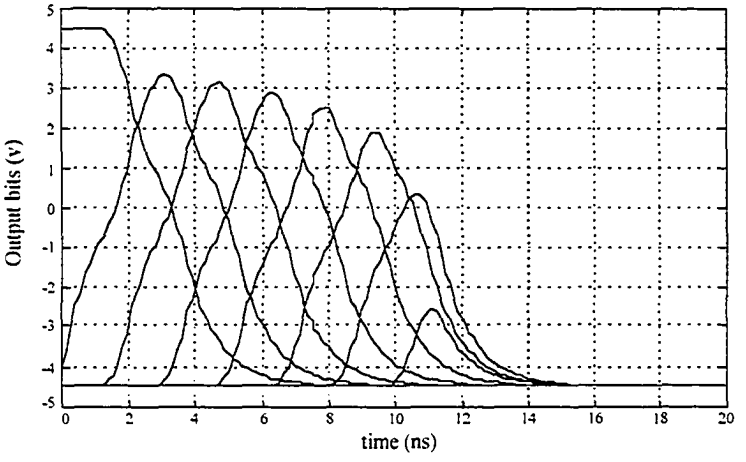


Figure 1.9 MATLAB simulation of the recursive binary adder.

Our research work builds on the previous work introduced in [34] and [35] by providing a novel systematic paradigm for implementing digital arithmetic circuits using analog CNN. The design methodology presented in the following chapters takes into consideration the pitfalls of the previous designs and ensures convergence of the network while optimizes its speed, silicon area, and power consumption.

1.5 CNN-based Arithmetic Circuits Design Goals

There are three major design goals that need to be fulfilled for a practical and successful design of a CNN-based arithmetic circuit: Convergence, scalability, and compatibility. The first design goal corresponds to the continuous output feedback nature of the CNN while the other two design goals come from the wide spectrum of applications using arithmetic circuits and a wealth of arithmetic circuit designs available in the literature.

1. Convergence: This requirement ensures that the developed CNN-based arithmetic circuits, after transient time, will always approach one of the stable equilibrium points, as will be discussed in Section 2.4.2.

2. Scalability: Precision required for arithmetic circuits varies by function. Consider multiplication as an example. At the low end, 8 bit words are used, as is the case in image compression algorithms, or 16 bits in more precise DSP tasks. At the high end, the word lengths in the IEEE double precision floating point standard are 53 bit and 64 bit. The scalability requirement ensures that arithmetic circuits with arbitrary sizes can be developed to meet the needs of specific applications.

3. Compatibility: The enormous collection of arithmetic circuit designs available in the literature places a stringent demand on new designs to provide backward compatibility. Instead of re-inventing the wheel, this requirement guarantees that the developed CNN-based arithmetic circuits can be used as embedded components in existing, more complex circuit structures without the need to re-design the whole circuit.

1.6 Thesis Overview

This research work explores the implementation of arithmetic circuits using arrays of analog circuits and the CNN computing paradigm, and also addresses mixed-signal applications where the presence of digital switching noise is a major problem. We thus describe a general technique for building low-noise digital arithmetic circuits using analog cellular neural networks, essentially implementing asynchronous digital logic with analog circuits. Each node in our asynchronous architectures uses controlled current sources driving into capacitors; providing both low current and voltage time derivatives ($\delta i/\delta t$ and $\delta v/\delta t$) and, as a result, reducing both instantaneous and time-averaged system and cross talk noise. In our approach, nonlinear templates are employed to perform the required arithmetic task without decomposing the arithmetic operation into primitive linear templates. Utilizing nonlinear templates facilitates performing medium complexity arithmetic operations with three major advantages. 1) Considerable reduction in processing time; since the arithmetic task is performed using one nonlinear template, the time needed to load/unload different templates with their inputs and initial conditions is eliminated. 2) Simplification of the circuit; this is because the control logic traditionally required to control template operations is no longer needed. 3) Decrease in power consumption; this result comes straightforwardly from the fact that power consumption is

directly related to processing time. Therefore, reducing processing time translates into lower power consumption. Moreover, removing the control logic from the circuit structure reduces power consumption further by the amount needed by the control logic.

To demonstrate the effectiveness of our methodology, we have designed and simulated CNN arrays for arithmetic operations using binary, binary signed-digit, and double-base number systems. First, we re-defined the arithmetic task in the given number system using continuous functions which are mapped into nonlinear templates. We then designed and simulated a CMOS circuit implementation of the arithmetic operation. We have demonstrated that the designed structures, regardless of the number system being used, are quite modular which enables the accurate evaluation of the performance of larger networks. We have also presented other novel contributions including the introduction of a new class of CNN featuring a 3-state transfer characteristics and an innovative self-programmable array using a novel feedback connection between groups of cells. Finally, we have analyzed the performance of the designed circuits in terms of power consumption, delay, and area. We have finally illustrated the efficiency of our designs to suppress noise by comparing them to standard digital implementations.

1.7 Thesis Organization

The thesis is organized as follows. Chapter 2 provides the basic theory of CNN arrays required to understand the work presented in subsequent chapters. Chapter 3 to Chapter 5 introduce general procedures to develop arithmetic circuits for the three number systems described in Section 1.6. Each chapter analyzes the corresponding arithmetic operations, defines the required templates, and provides Hspice simulations to demonstrate convergence of the designed circuits. We also present the designs of multi-bit adders and multipliers to illustrate the scalability and compatibility of each algorithm in more useful arithmetic tasks. Each chapter also examines the impact of the corresponding design on system noise using extensive Hspice simulations. Finally, Chapter 6 summarizes the work and provides a detailed comparison of the performance of each design in terms of noise, area, delay, and power consumption. Chapter 6 also presents the final conclusions.

Chapter 2

Cellular Neural Networks: An Overview

Cellular Neural Network (CNN) arrays represent a massively parallel asynchronous computing paradigm that is a hybrid of Cellular Automata (CA) and Artificial Neural Networks (ANNs). CNN arrays take advantage of both worlds: their local connectedness makes the arrays well suited for VLSI implementation, and, similar to ANNs, they provide a natural parallel processing paradigm. The CNN regular grid–like structure also makes it a good candidate for online solutions of systems of first order non–linear differential equations. CNNs represent an analog nonlinear dynamic system operating in continuous or discrete time. When considered as a system, a CNN is characterized by the fact that information is directly exchanged just between neighboring neurons. Of course, this characteristic does not prevent the capability of obtaining global processing. Cells that are not in the immediate neighborhood have an indirect effect because of the propagation effects of the dynamics of the network. By exploiting locality of connections, electronic IC and optical or electro-optical implementations become feasible, even for large nets, which is the main advantage of CNNs over ANNs. Since the research work presented in this thesis is built on CNN arrays, the purpose of this chapter is to acquaint the reader with basic CNN theory needed for subsequent chapters. A brief history of CNNs and the scope of

applications is given in Section 2.1. The spatial layout and restrictions on connections between neighboring cells are described in Section 2.2. In Section 2.3, mathematical equations defining the behavior of a CNN cell are reviewed and a general circuit architecture of a CNN is given. The dynamics of CNNs is discussed in Section 2.4, and modes of operation and notes on CNN stability are also presented.

2.1 CNN History and Applications

Since their introduction in 1988 by Chua and Yang [38][39], Cellular Neural Networks have attracted considerable attention. They are well suited for image processing applications, because of their two-dimensional structure and local interconnections, which are typical characteristics of many image processing algorithms [40]-[42]. Enormous advances have been made by many researchers in this field [43]. While software prototypes prove the potential of CNN [44]-[46], a great deal of research has also been reported in hardware implementations which can be used for real–time applications. These implementations include transconductance-mode based processing elements [47], switched-current signal processing elements [48], discrete-time implementations [49][50], current-mode implementation [50][51] and more [52]-[54]. The first CNN realizations were designed to perform one specific function in image processing or classification, such as edge detection [40], connected component detection [55], noise removal [39], or hole filling [56]. More complex image processing functions are also reported including image and video compression [57]-[60], image rotation [61]-[62], nonlinear image filtering [63]-[66], image enhancement [67]-[69], image restoration and reconstruction [70]-[72], image segmentation [73]-[75], pattern matching and classification [76]-[78], and character/face recognition [79]-[81]. CNN arrays have also been applied to a wide variety of important tasks in robot navigation [82]-[84], motion detection and estimation [85]-[88], defect inspection [89]-[92], satellite communication and secure transmission systems [93]-[97], analysis of brain electrical activity in epilepsy [98]-[100], cryptography [101], bionic eyeglasses [102]-[103], and solving partial differential equations and optimization problems [104]-[107], just to mention a few. More recently, researchers investigated programmable CNNs to provide flexibility in implementing analog parallel array

processors [53],[108]. In other work, the slope and the threshold of the activation function has also been made tunable [109]-[110]. Because of the thresholded activation function at the output of the CNN structure, many image processing applications have traditionally been based on black and white images even though CNN arrays, by their nature, are analog, continuous processing systems. However, grey/color based applications have recently started to emerge [61], [66], [111]-[113]. From a hardware point of view, the local connectivity of the CNN array lends itself to practical VLSI implementations. The addition of logic functions results in a programmable analog/logic array computer capable of performing algorithms that combine the strengths of analog template processing and logic operations [41],[114]. In fact, CNN arrays sets the platform for a new algorithmic style based on the spatio–temporal properties of the array. The key elementary instruction is a spatio–temporal transient generated by a two dimensional nonlinear dynamic processor array. This basic instruction resembles the typical convolutional operator used in image processing applications.

2.2 CNN Structures

The CNN is intrinsically defined spatially; generally only 1- or 2-dimensional space is considered, so that the CNN can be realized physically. The most common types of CNN can be characterized as a 2-D planar array of dynamic cells (neurons) with rectangular, triangular or hexagonal geometry. Any cell on the i^{th} row and j^{th} column, $C(i,j)$, is connected only to cells within a small neighborhood, denoted as $N(i,j)$. For hardware implementations, and due to the wiring complexity involved, most often neighborhoods are of radius 1; although, for software simulations, a radius of 3 or more has been reported. As an example, Figure 2.1 shows two different CNN grids with a neighborhood of radius 1. Note that in this figure, each cell in the rectangular grid is connected to the inputs and outputs of 9 cells, including to itself, while cells in the hexagonal grid are connected to the inputs and outputs of 7 cells for the same radius.

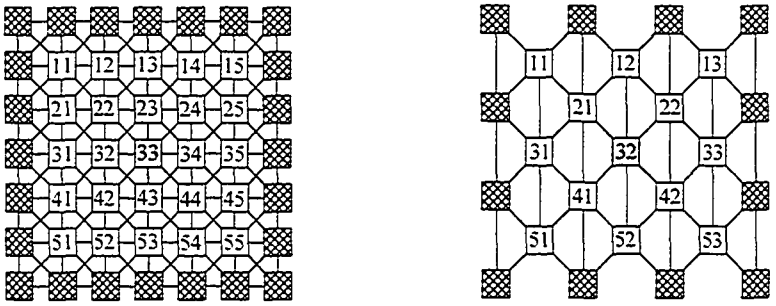


Figure 2.1 Examples of rectangular and hexagonal CNN grids with neighborhood of size 1. Light grey cells belong to the neighborhood of the dark grey cell.

In a CNN, cells may be all identical or they can belong to a few different types as is the case for biological neurons. The interconnection strengths or connection weights are usually spatially invariant. However, more than one connection network may also be present, with a different neighborhood size to permit short range interactions and subsystem connections. To ensure that the cells on the perimeter of the CNN grid achieve proper convergence, dummy border cells (hatched cells in Figure 2.1) are added on the border of the processing array to simulate interaction with imaginary cells outside the CNN grid. The size of the dummy border depends on the neighborhood radius. For example, for a rectangular array with a neighborhood of radius 1, the width of the dummy border would be 1 cell as depicted in Figure 2.1. A dummy cell outputs a constant voltage that a properly converged computing cell would produce if it were in its place. A dummy cell would also receive an input signal voltage as if it were a member of the array. Therefore, a cell on the perimeter of the CNN array uses the input signal voltage and dynamic output voltage of neighboring cells as well as the static output and signal voltages of the dummy cells to arrive at the proper final state. The border cells are treated as members of the array for initialization purposes and template implementation, but are not considered in the final state analysis.

All cells in the CNN operate in parallel and when one computing cell is allocated to each pixel in the 2D input signal, the CNN achieves very high signal processing speeds. Although the cells are only locally connected, the network is able to perform global

operations on the 2-D inputs. This is possible because the missing global connections are replaced by a time-multiplexing of the connections and the time-propagation of the information through the network from cell to cell.

2.3 CNN Cell Architecture

CNN cells are multiple input - single output nonlinear processors that consist of linear and nonlinear circuit elements. Each cell is characterized by an internal state variable x_{ij}, that is bounded for all time $t > 0$. Every cell also has a constant external input u_{ij} and output y_{ij}. The evolution and dynamics of the state of cell ij is described by the first order nonlinear differential equation:

$$C_x \dot{x}_{ij}(t) = -\frac{1}{R_x} x_{ij}(t) + \sum_{c \in N_{ij}} A_c \cdot y_c(t) + \sum_{c \in N_{ij}} B_c \cdot u_c(t) + I \tag{2.1}$$

and output function:

$$y_{ij}(t) = \frac{1}{2}(|x_{ij}(t) + 1| - |x_{ij}(t) - 1|) \tag{2.2}$$

where I is a local value called the bias, and N_{ij} is the r-neighborhood of the cell $C(i,j)$ which contains all cells within a radius r. The output nonlinearity $f(x_{ij})$ is a piecewise linear function; f is linear in the unit range [-1,1], and outside the unit range the output saturates to +1 for positive state values and to -1 for negative state values, as shown in Figure 2.2. A_c and B_c are two generic parametric functionals. The A_c template connections represents the inter-cell connection weights and provides an output feedback mechanism. The B_c template connections in turn represents connections to the input and serves as an input control mechanism. Specific entry values of the bias term and the feedback and control templates are application dependent and, most often, are identical for all cells (so called cloning templates). The constant bias, I, and the cloning templates determine the transient behavior of the cellular nonlinear network.

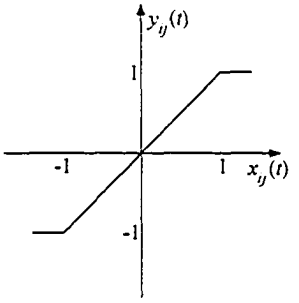


Figure 2.2 CNN cell activation function.

The solution to this system of equations is the classical exponential function of a first order system. The maximum convergence rate of a cell is determined by the integration time constant C_xR_x. Therefore the speed of the CNN array can be controlled by adjusting this value. This property is crucial in controlling the cross talk (see Section 1.3).

In Figure 2.3, a block diagram that implements Eqn. (2.1) is represented. The cell sums the incoming signal from the neighbors, itself, and the constant bias of the I template and integrates them to compute its internal state. The cell also sends two signals to each of its neighbors: one signal is its output multiplied by a weight from the A_c template; the second signal is its input multiplied by a weight from the B_c template.

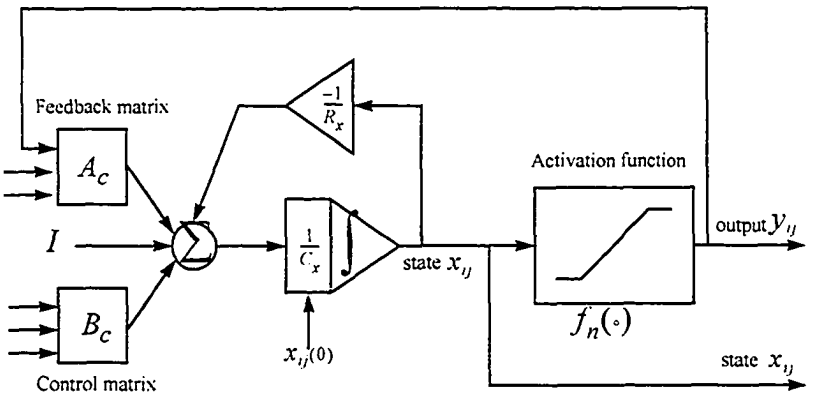


Figure 2.3 A block diagram representation of a CNN cell.

The schematic of an electrical implementation of a CNN cell is shown in Figure 2.4. The linear capacitor C_x and the linear resistor R_x constitute a lossy current integrator. The output y_{ij} is generated by a nonlinear voltage controlled current source (VCCS) I_y across the output resistor R_y. The VCCS I_y is controlled by the state voltage x_{ij}. The linear VCCS, $A_c y_c$ and $B_c u_c$, generate current signals that are sent to the states of the neighboring cells (and itself). The I template is realized by the independent current source I.

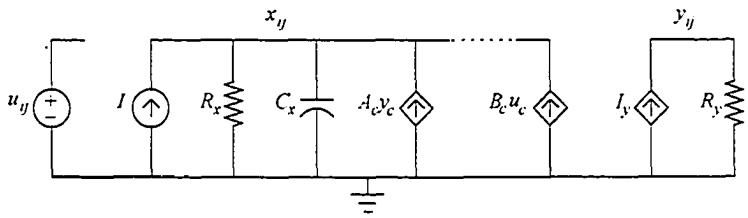


Figure 2.4 Schematic of an electrical implementation of a CNN cell.

2.4 CNN Dynamics

Each cell in a CNN is a non-linear dynamic system capable of processing continuous signals in either continuous–time or discrete–time modes. Continuous time (CT-CNN) are nonlinear dynamic systems described by differential equations. Discrete time (DT-CNN), with advantages similar to CT-CNN in terms of local activity etc., is described by nonlinear finite difference equations [115]. In most cases, the network is non-Markovian; i.e., the future internal state depends also on the past history of the system [116]. In the special case of a time-variant CNN, all the templates, neighborhoods, and parameters can also be a function of time. This complex dynamic phenomena of CNN arrays has been studied by many researchers [117]-[119].

2.4.1 Modes of Operation

The CNN can operate in two modes: Input-driven mode and autonomous mode.

In input-driven mode, the signal to be processed is applied to the inputs of the CNN cells and the states of the CNN cells are set to the initial conditions. After transient time, the results of the computation can be taken from the steady state equilibrium values and the outputs of the cells.

In autonomous mode, the B_c template is set to zero and the inputs of the CNN network are not used. The signal to be processed is applied as the initial conditions of the network. The result of the processing operation is contained in the steady state equilibrium values and the outputs of the network.

2.4.2 Network Convergence

It is well known that the stability of CNN arrays is a critical characteristic for most applications [120]- [123]. The stability criteria requires that the state of each cell should be bounded for all time $t > 0$ and, after the transient has settled down, a cellular neural network must always approaches one of its stable equilibrium points. This last fact is relevant because it implies that the circuit will not diverge or oscillate. The rate of convergence is determined by many factors including the amount of current that is flowing into the cell, the capacitor C_x, and the effecvtive resistance R_x in parallel with C_x. The stability of CNN networks is analyzed in [38] and [124] by defining a Lyapunov's function which expresses the generalized energy present in the system. The properties of this function imply that the states of the network will always evolve towards a constant DC equilibrium value. Moreover, if the self-feedback term $A_{ii} = \alpha \cdot y_{ii}(t)$ satisfies the condition: $\alpha > 1/R_x$, each cell state settles at a stable equilibrium point with a magnitude greater than 1. This means that the outputs of the network will be binary because of the nonlinear activation function. This characteristic makes the CNN very attractive for some pattern extracting applications, such as edge detection and connected component detection, where a binary output image is acceptable. In such cases, the issues of linearity, precision, and offsets of the output values are not relevant because the state variables are not of critical importance [51],[114]. However, in some cases, a CNN with linear continuous observable outputs is required, for example, to build a real time control system

or to obtain an output image with multiple grey or color levels. Since the cells outputs are limited by the activation function, state variables can be used as continuous outputs [111].

2.5 Summary

In this chapter, the basic theory of cellular neural networks is presented. A brief history of CNN and the horizon of applications available in the literature is also given. The most common types of CNN can be defined spatially as a 2-D planar array of dynamic cells with rectangular, triangular or hexagonal geometry. Dummy border cells are usually added on the border of the processing array to ensure that the cells on the perimeter of the CNN grid achieve proper convergence. The behavior of the CNN cell is expressed as a first-order differential equation and a functional schematic with continuous feedback is analyzed. The dynamics of the nonlinear systems are discussed and differences between input-driven mode and autonomous mode are explained. The stability criteria of these continuous feedback systems requires that the network should always evolve towards a constant DC equilibrium value.

Chapter 3

Binary Arithmetic Using CNNs

This chapter presents an intriguing technique for building binary arithmetic circuits using analog cellular neural networks. The developed circuits also exhibit very low system noise because of the smooth dynamics of the interconnected nonlinear analog cells. Moreover, cross-talk can be controlled by adjusting the R_xC_x integration time constant of the cell, as discussed in Section 1.3. The new designs reduce the peak system noise by up to 57dB and cross talk by about 20dB when compared to traditional CMOS digital counterparts, developed in the same 0.35μm CMOS technology. The chapter is organized as follows. In Section 3.1, the binary number system is briefly reviewed and the binary addition algorithm is explained. A systematic procedure to implement a 1-bit binary full adder using an array of analog CNN cells is introduced in Section 3.2. The binary addition algorithm is first examined. The sum and carry functions are defined using new continuous functions. This facilitates implementing the sum and carry functions using simple basic analog circuits which is discussed subsequently. CMOS implementation of the full adder is then presented. The convergence of the full adder for all possible inputs is shown using comprehensive Hspice simulations. This last property guarantees the stability of larger arithmetic networks. The scalability of the full adder is addressed by designing a 32-bit

CNN-based binary adder in Section 3.3. The compatibility of the full adder design with existing complex structures is discussed in Section 3.4. The design of a 32x32-bit binary multiplier is presented to demonstrate that the new full adder can be embedded in existing architectures without modifying the original circuit. Noise performance of the new designs is also addressed. The chapter is summarized in Section 3.5.

3.1 The Binary Number System: Overview

3.1.1 Definition

The binary number system is a weighted number system in which any algebraic value X can be represented by an n-bit vector as:

$$X = \sum_{i=0}^{n-1} x_i \times 2^i \tag{3.1}$$

where $x_i \in [0,1]$ and the algebraic value X is bound by $0 \leq X \leq (2^n - 1)$.

3.1.2 Binary Addition

Binary addition is the most basic function of binary arithmetic because other binary arithmetic operations can be decomposed into primitive operations performed using binary addition. For example, binary subtraction can be calculated by adding the minuend to the 2's complement of the subtrahend and multiplication can be obtained by adding the multiplicand to itself a number of times equal to the multiplier. As with decimal addition, when the result of adding two binary bits at position i exceeds the value of the binary radix (2), a *carry out* is produced and added to the next place i+1 as a *carry in*. This process is shown in the binary addition example of Figure 3.1. The example shows the addition of two 5-bit numbers X=13 and Y=23.

	2^5	2^4	2^3	2^2	2^1	2^0	
							weight
	1	1	1	1	1		carry
	0	0	1	1	0	1	X
+	0	1	0	1	1	1	Y
=	1	0	0	1	0	0	S

Figure 3.1 An example of binary addition.

The addition process described above can be made modular by using a functional unit called a full adder (FA). The FA shown in Figure 3.2-a accepts three binary inputs x_i, y_i, and c_i, and generates two binary outputs s_i and c_{i+1}. The operation of the FA can be described using the following equation:

$$x_i + y_i + c_i = 2c_{i+1} + s_i \tag{3.2}$$

The solution to Eqn. (3.2) is given by:

$$\begin{aligned} s_i &= (x_i + y_i + c_i) mod\ 2 \\ c_{i+1} &= \lfloor (x_i + y_i + c_i)/2 \rfloor \end{aligned} \tag{3.3}$$

where $\lfloor a \rfloor$ represents the largest integer value such that $\lfloor a \rfloor \leq a$.

Since each FA exchanges binary values with its immediate neighbor full adders only, one can create any arbitrary size n-bit parallel binary adder by cascading n full adders together as shown in Figure 3.2-b.

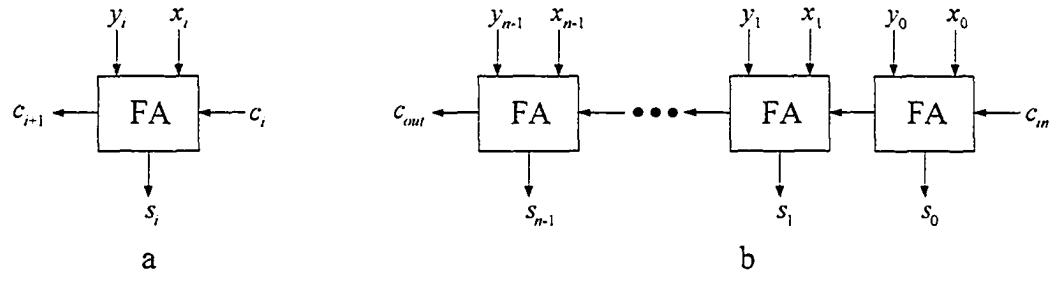


Figure 3.2 Block diagram of a binary adder: (a) 1-bit full adder, (b) *n*-bit binary adder.

3.2 Designing a 1-bit Binary Full Adder Using CNN (CNNBFA)

In the CNN paradigm, developing a certain function or operation means to design template connections, both output feedback template and input control template, as well as the bias to each cell so that, when applying these templates to the CNN network, the output of the network corresponds to the desired function applied to the input. Binary arithmetic can be implemented on CNN arrays by decomposing the arithmetic operation into a set of primitive Boolean functions. Each Boolean function can be mapped into a set of linear templates as described by Galias [126]. These linear templates would be applied in sequence, each for a finite time, to obtain the final output of the arithmetic operation. Another method to implement binary addition using recursive nonlinear templates is described in [34]. However, the network proposed is sensitive to template values and diverges for operands larger than 8 bits. In this section, a novel method to implement binary addition using nonlinear templates is introduced. The algorithm guarantees stability of the operation and the scalability of the CNN network to perform addition of operands of arbitrary size is discussed in Section 3.3. Binary multiplication using this technique is addressed in Section 3.4.

3.2.1 CNNBFA Templates Design

The objective here is to describe the sum and carry outputs of the binary addition operation described by Eqn. (3.3) as continuous (analog) functions of the values of the input bits x_i, y_i, and c_i. Thereupon, one can use the new continuous functions as the templates connecting cells in a CNN network. The binary sum function given in Eqn. (3.3) can be re-written using the *XOR* logic function as:

$$s_i = x_i \oplus y_i \oplus c_i \tag{3.4}$$

The logic *XOR* function can be defined in the continuous analog domain as the absolute of the difference between the two inputs:

$$x_i \oplus y_i = |x_i - y_i| \tag{3.5}$$

Then the binary sum function can be directly mapped into the continuous analog domain by substituting Eqn. (3.5) into Eqn. (3.4) twice yielding:

$$s_i = \left| |x_i - y_i| - c_i \right| \tag{3.6}$$

Now consider a CNN implementation of the 1-bit binary full adder shown in Figure 3.2-a. Each of the output variables, s_i and c_{i+1}, can be implemented using one CNN cell. The input variables, x_i, y_i, and c_i, can be applied as input signals to the CNN network. The input signals must be binary voltages for design compatibility with standard digital circuits. However, employing current-mode circuits can substantially reduce circuit complexity and interconnects. For example, adding two signals in current-mode can be performed with a wired sum without active devices. A current-mode CNN cell can act as an input buffer that accepts voltage inputs and produces current signals for internal processing. Observe from Figure 3.2-b that the carry output from weight 2^i is the same as the carry input to weight 2^{i+1} and there is no need to use a CNN cell for the carry input. The final CNN grid is shown in Figure 3.3-a. With this mapping in mind, the template connections for the binary sum function of Eqn. (3.6) can be written as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot \left| |y_{i+2,j}(t) - y_{i+3,j}(t)| - y_{i+1,j}(t) \right| \quad (3.7)$$

where β is a constant chosen to speed up the transition.

The binary carry function of Eqn. (3.3) can be mapped into the continuous analog domain using a simple summing node where all the currents from the involved cells are summed together. The thresholded output nonlinearity of the CNN cell can be utilized to implement the floor function. However, this requires forcing the CNN cell state voltage to settle at a value outside the linear range [-1,1]. This can be achieved by subtracting a unit current instead of dividing by two in Eqn. (3.3). The new equation describing template connections to the carry cell can then be written as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot ((y_{i+1,j-1} + y_{i+2,j-1} + y_{i,j-1}) - 1) \quad (3.8)$$

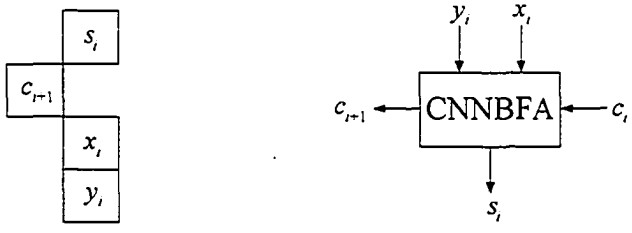


Figure 3.3 Representation of the CNN-based 1-bit full adder: (a) CNN grid, (b) block diagram.

3.2.2 CNNBFA CMOS Basic Building Blocks

The CNNBFA shown in Figure 3.3 consists of four CNN cells that are connected together using the templates described by Eqn. (3.7) and Eqn. (3.8). The templates can be synthesized using several primitive current-mode components. Then the CNNBFA can be constructed by connecting the four basic CNN cells using these primitive current-mode circuits. The basic building blocks of the CNNBFA are:

Wired-sum: The most attractive feature of current-mode logic circuits is that arithmetic summation, including polarity, of analog currents can be performed by means of a simple wired connection without active devices. From Kirchhoff's current law, the current I in Figure 3.4 is given by:

$$I = I_1 + I_2 \tag{3.9}$$

This property reduces interconnection complexity and the goal in the design of the current-mode circuits is to maximize the usage of this operation so that the resulting arithmetic circuits become quite simple.

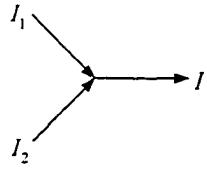


Figure 3.4 Schematic of a current-mode summing node.

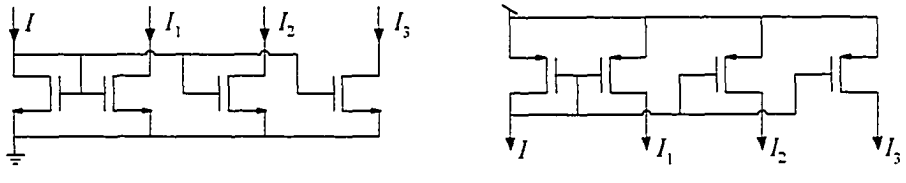
Current source: A current source delivers one given level of current (unit current). Current sources are designed by a pMOS or an nMOS transistor as shown in Figure 3.5. The current level is adjusted by the transistor ratio W/L and the gate reference voltage. In the event that different current levels are required in other sections of the circuit, current mirrors may be adopted as will be discussed next.

Figure 3.5 Schematic of current sources: (a) nMOS current source, (b) pMOS current source.

Current mirror: Current mirrors constitute the main interconnections between cells and the speed of the circuit depends largely on the speed of the current mirrors. They are used for signal distribution by generating replicas of the input current. A replica can be scaled by using an appropriate output and input W/L ratio. Current mirrors are also used for inverting the current direction. The simple nMOS and pMOS current mirrors shown in Figure 3.6 are used in the design of the CNNBFA. The nMOS current mirror is used for producing replicas while the pMOS type is used for scaling and inverting direction. The replicas are given by:

$$I_i = m_i I \tag{3.10}$$

where m_i is the scaling factor and is determined mainly by the W/L ratio of the input and output transistors.



a b

Figure 3.6 Schematic of simple current mirrors: (a) nMOS current mirror, pMOS current mirror

Subtractor: Subtraction can be viewed mathematically as addition of the minuend and the negative of the subtrahend; $I = I_1 + (-I_2)$. The realization of a subtractor can be achieved using a wired-sum and a simple current mirror (to invert the direction of the subtrahend) as shown in Figure 3.7.

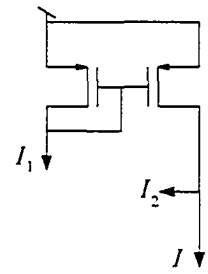


Figure 3.7 Schematic of a subtractor.

Absolute function: For correct operation of a uni-directional current-mode circuit, the input current should flow in a certain direction. The current output of a subtractor depends on the values of the two inputs. If the minuend is less than the subtrahend, the output current will be negative; i.e., flows in the opposite direction. The design of the CNNBFA uses the absolute function presented in [127], and depicted in Figure 3.8 for convenience, to force the current into the correct direction. The current output of the absolute function is given by:

$$I = \begin{cases} I_{in} & I_{in} \geq 0 \\ -I_{in} & I_{in} < 0 \end{cases} \tag{3.11}$$

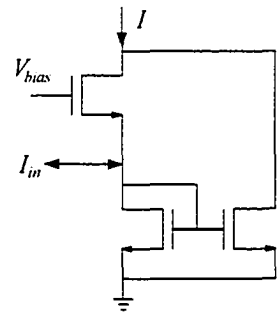


Figure 3.8 Schematic of absolute function.

Activation function: The central building block in CNN circuits is the cell and the core of the cell is the activation function. The state-to-output converter with a current output presented in [128] is used in the design of the CNNBFA. Note in this implementation, shown again in Figure 3.9, that one side of the differential pair is used for the implementation of the nonlinear activation function and the other side is used for signal distribution to the neighbor cells. The output of the circuit is given by Eqn. (2.2).

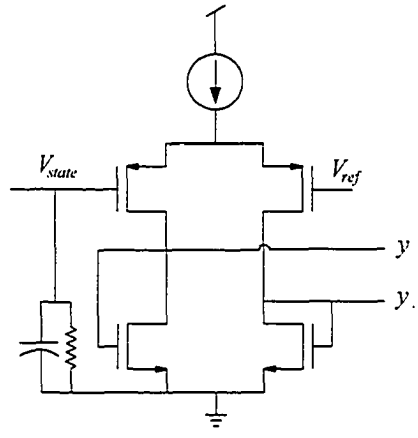


Figure 3.9 Schematic of basic CNN cell.

3.2.3 CNNBFA CMOS Implementation

The CNNBFA consists of four CNN cells as shown in Figure 3.3-a. The two input rows function as input buffers to convert binary input voltages into currents for internal processing by the other two rows. The template connections for the sum function in Eqn. (3.7) can be realized using basic building blocks presented in the previous section. The complete CNN sum cell schematic with connections to neighbor cells is shown in Figure 3.10.

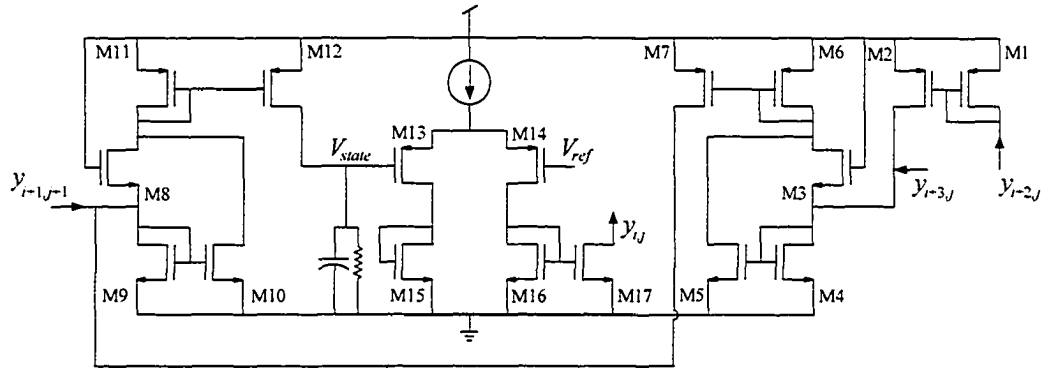


Figure 3.10 Schematic of the CNNBFA sum cell with connections to neighbors.

The operation of the circuit can be directly described using Eqn. (3.7) as follows. The input signal representing y_i is subtracted from the input signal representing x_i using the current subtractor circuit of M1-M2. The absolute of the output of the subtraction is obtained using the absolute function generator of M3-M5 and mirrored using the current mirror of M6-M7. This last operation inverts the direction of the signal so that the second subtraction operation in Eqn. (3.7) reduces to merely addition of the output of the current mirror M6-M7 and the input signal representing c_i. The absolute of the sum signal is first obtained using the second absolute function generator of M8-M10 and then mirrored using the current mirror of M11-M12. The direction of the output current of the current mirror M11-M12 forces the output of the CNN cell to take one of the two binary values 0 and 1.

The synthesis of the carry function given in Eqn. (3.8) is simpler than the sum function because it only includes wired addition of current signals. Therefore, there is no need to use absolute function generators. The carry output is also expected to settle down to the final value before the sum output because the number of devices in the critical path is less. The carry cell with template connections to neighbor cells is shown in Figure 3.11.

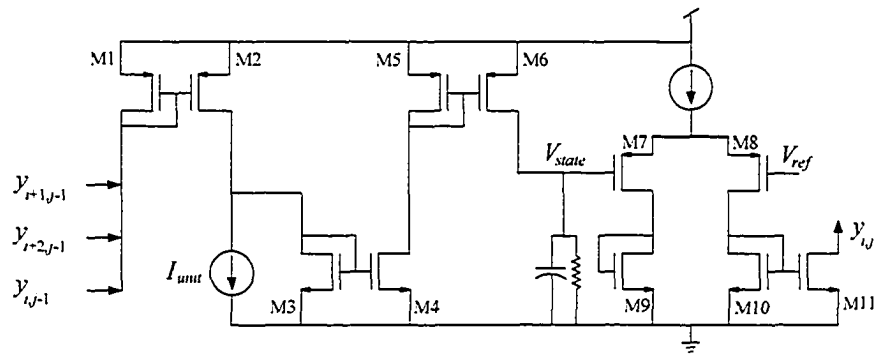


Figure 3.11 Schematic of the CNNBFA carry cell with connections to neighbors.

The operation of the circuit comes directly from the carry function of Eqn. (3.8). The three input signals x_i, y_i, and c_i are summed together. The sum of the signals is mirrored using the current mirror M1-M2. A unit current, implemented using an nMOS current source, is subtracted from the sum of the three input signals. The rest of the current is drained by transistor M3 and mirrored again using the current mirror M5-M6 so that the output of the CNN cell takes one of the two binary values 0 or 1.

3.2.4 CNNBFA Hspice Simulation

The CNNBFA shown in Figure 3.3 has three binary inputs (x_i, y_i, and c_i) and two binary outputs (s_i, and c_{i+1}). To test the design for correct network operation, a truth table of the functionality of a 1-bit binary full adder is constructed as shown in Table 3.1. The truth table is divided into four sections according to the number of logic "1s" in the input patterns. A test circuit is designed using Hspice and parameters from 0.35μm CMOS process technology. The inputs from the truth table are applied as a test bench to the CNNBFA and the output signals are probed for plotting. The worst case delay of the outputs of Hspice simulations of each section in the truth table is plotted in Figure 3.12. The squares on the left side of the graph represent the input signals while the squares on the right side represent the output signals. Logic "1" is represented by a filled square and logic "0" by an empty square. The delay for the sum and carry signals was measured as the time it takes the output signal to rise from 10% to 90% of its final value. The worst case

delay was measured as 2.85*ns* and 1.65*ns* for the sum and carry signals respectively. Hspice simulations also show that the network is stable and the outputs monotonically approach their correct steady state values. This also guarantees that multi-bit adders, discussed in the next section, will also converge since the CNNBFA units are connected to each other through the carry signal which is stable in itself.

Table 3.1 Truth table of a 1-bit binary full adder.

x_i	y_i	c_i	s_i	x_{i-1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

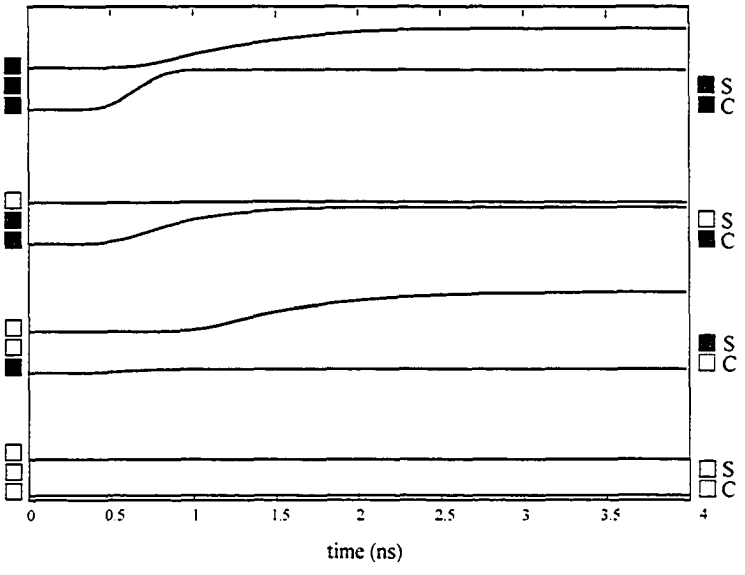


Figure 3.12 Hspice simulation of the CNNBFA.

3.3 CNNBFA Design Scalability

The CNNBFA can be used as an enabling building block to design binary adders with arbitrary sizes. An n-bit binary adder can be obtained by cascading n CNNBFA units as shown in Figure 3.13. Similar to the traditional 1-bit digital full adder, the carry output c_{i+1} from the CNNBFA in position i is connected to the carry input c_i of the CNNBFA in position i+1. The number of CNN cells required by the n-bit binary adder is $4n$ because each CNNBFA uses four CNN cells.

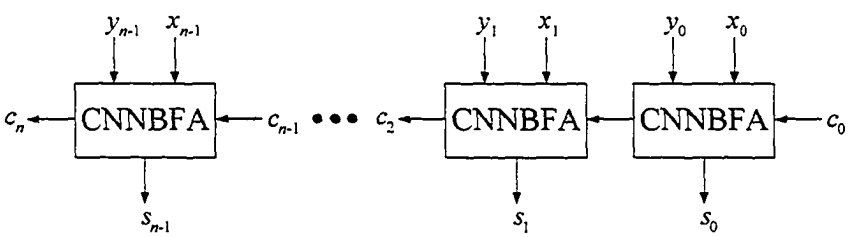


Figure 3.13 Block diagram of an n-bit CNN-based binary adder.

3.3.1 A 32-bit CNN-based Binary Adder

To prove the scalability of the CNNBFA, a 32-bit CNN-based binary adder is designed. Hspice simulations using random patterns of input signals were carried out and the outputs of the designed adder were verified. It is known, however, that the sum and carry outputs at position i depend on the carry input from position i-1 which in turn depends on the carry input from position i-2 etc. This means that the worst case for the CNN network convergence (and maximum delay) would be to add two binary numbers that force the carry to propagate all the way from the least significant bit to the most significant bit. Hspice simulations that reflects this situation are shown in Figure 3.14. The figure shows the addition process of a small part of the 32-bit adder to make the figure readable. The corresponding operands are X=01111111 and Y=00000001. The simulations demonstrate that the 32-bit CNN-based binary adder always converges to the correct sum outputs even for maximum carry propagation.

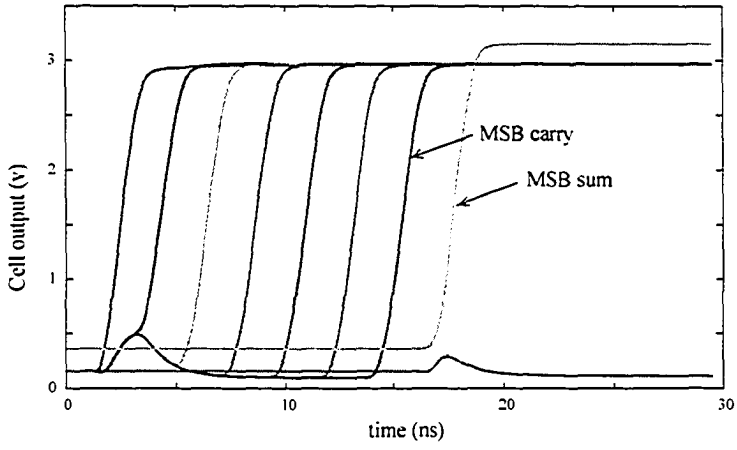


Figure 3.14 Hspice simulation of an 8-bit section of the 32-bit CNN-based binary adder.

3.3.2 Impact of the CNN-based Binary Adder on Substrate Noise

The amount of switching noise is calculated as the product of the effective parasitic inductance and the rate of change of the instantaneous power supply current (i.e., the current drawn from the power supply by the circuit under test). The parasitic inductance is mainly determined by the specific layout of the circuit and the process technology used to implement the circuit. In order to make suitable comparisons, we make the assumption that the parasitic inductance is identical for both the CNN and CMOS circuits and so the instantaneous noise voltages are given by Eqn. (3.12):

$$\begin{aligned} v_{CNN} &= L\delta i_{CNN}/\delta t \\ v_{CMOS} &= L\delta i_{CMOS}/\delta t \end{aligned} \tag{3.12}$$

If we use a measure of noise power as the power dissipated in a given resistor, R, then we can compute the ratio of this noise power for both the CNN and CMOS noise "sources". Expressing this in decibels (dB) we find the effective power ratio is given by Eqn. (3.13):

$$P_{CMOS/CNN} = 20 \cdot \log_{10}\left(\frac{\delta i_{CNN}/\delta t}{\delta i_{CMOS}/\delta t}\right) \text{dB}$$
$$= (20 \cdot \log_{10}(\delta i_{CNN}/\delta t)) - (20 \cdot \log_{10}(\delta i_{CMOS}/\delta t)) \quad (3.13)$$

In order to better demonstrate the noise differences as a function of time, we will provide separate graphs of the two computations shown in the lower expression of Eqn. (3.13). The effective power ratio will be understood to be the difference between the two graphs. As an aside, we note that although most calculations of noise are statistically based (where there is often an assumption of stationarity), in our case we are very interested in the time domain behaviour of what is non-stationary noise, since the switching noise may be responsible for circuit errors at specific times (such as the incorrect latching of values in registers at high switching noise events such as clock rise and fall). By taking worst case time domain values we can make judgments on the relative merits of CNN circuits versus CMOS circuits at these specific worst-case times.

In this section, the switching noise of the designed CNN-based 32-bit binary adder is compared to the switching noise of a 32-bit standard digital binary adder by monitoring the instantaneous power supply current [1]. Both adders are designed using the same 0.35μm CMOS process technology and are operated at the same speed, the speed determined by the slower CNN-based adder. Several Hspice simulations were performed on random operands and the switching noise of each simulation was recorded. The worst case switching noise of the CNN-based binary adder is plotted against the worst case switching noise of the standard digital binary adder in Figure 3.15.

Notice from the figure that the switching noise of the CNN-based adder is less than that of the standard digital adder at all times during the addition process. This is because the CNN-based adder tends to smooth out the transitions while the standard digital adder changes states abruptly. The worst-case scenario for both adders is when the binary inputs (representing the operands) are first applied. During the first few nano seconds, switching noise increases drastically and then drops gradually as the addition proceeds and more bits

settle to the final value. In these simulations, the CNN-based binary adder achieves an improvement of 50dB in switching noise over the standard digital binary adder.

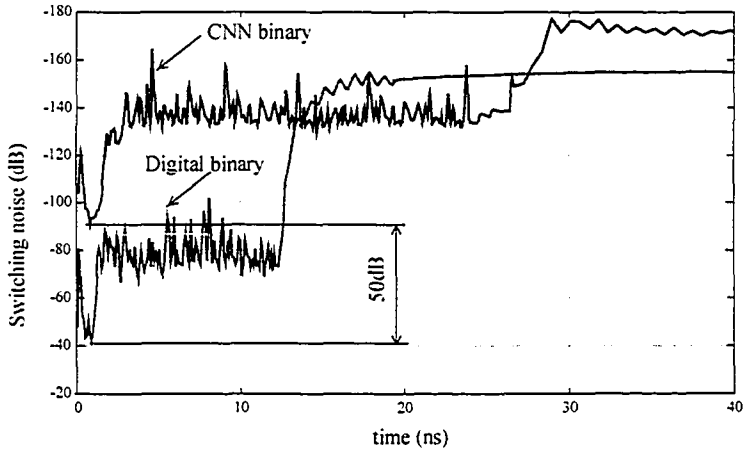


Figure 3.15 Switching noise of the CNN-based and standard digital 32-bit binary adders.

The presence of parasitic capacitance between the switching transistors and the silicon substrate also increases the substrate noise. The amount of noise injected into the common substrate is given by Eqn. (1.2) as the product of the lumped parasitic capacitance and the rate of change of the switching node voltage. The parasitic capacitance is also a function of the specific circuit layout and the process technology used to fabricate the circuit. The cross talk of the CNN-based 32-bit binary adder is plotted against that of the standard digital 32-bit binary adder in Figure 3.16. The CNN-based adder suppresses cross talk by about 20dB over of that of the corresponding digital adder.

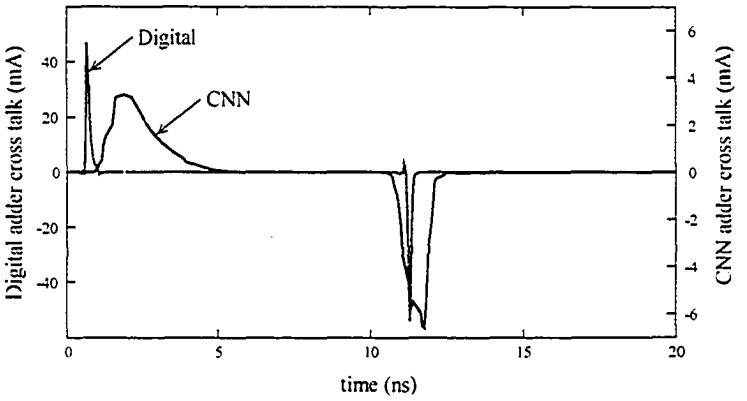


Figure 3.16 Cross talk noise of the CNN-based and standard digital 32-bit binary adders.

3.4 CNNBFA Design Compatibility

The developed CNNBFA has standard inputs (operands and carry in) and standard outputs (sum and carry out). This property facilitates using the CNNBFA in more complex circuit structures without the need to change the design of the existing circuit. As an example, consider designing a standard $n \times n$-bit carry-save tree multiplier. The structure of such a multiplier is shown in Figure 3.17.

The multiplier uses a carry-save reduction tree of binary full adders to reduce the n binary numbers of partial products into two binary numbers. The number of levels of the binary full adders used in the reduction tree is $O(\log n)$. The final stage of the multiplier is a binary adder that adds together the two binary numbers produced by the carry-save reduction tree. This specific application, demonstrates that the CNNBFA can be used as an embedded component in the carry-save reduction tree. It also illustrates that the CNN-based multi-bit binary adder designed in Section 3.3 can also be embedded in the final stage of the carry-save tree multiplier.

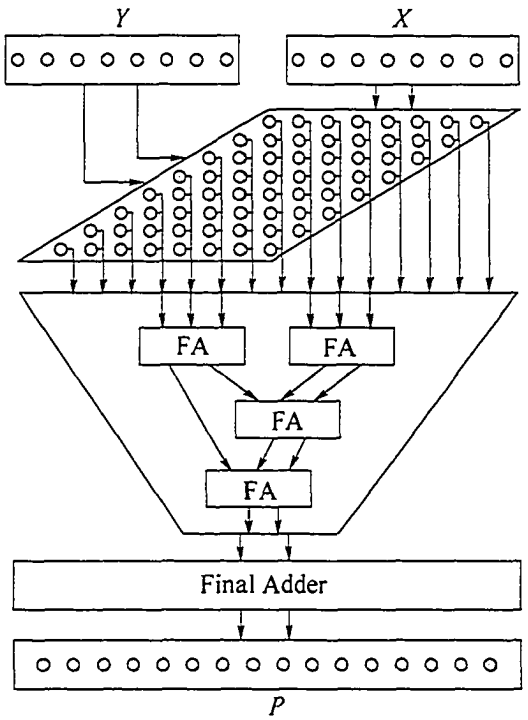


Figure 3.17 Block diagram of a carry-save tree multiplier.

3.4.1 A 32x32-bit CNN-based Binary Multiplier

Using the CNNBFA and the CNN-based multi-bit adder, a 32x32-bit carry-save binary multiplier was developed. Extensive Hspice simulations were performed using random 32-bit binary operands. The multiplier outputs of one of the simulations is shown in Figure 3.18 where, for the sake of clarity, only the first 16-bits are shown. The CNN-based binary multiplier converged for all simulations and all outputs monotonically approached their final state values.

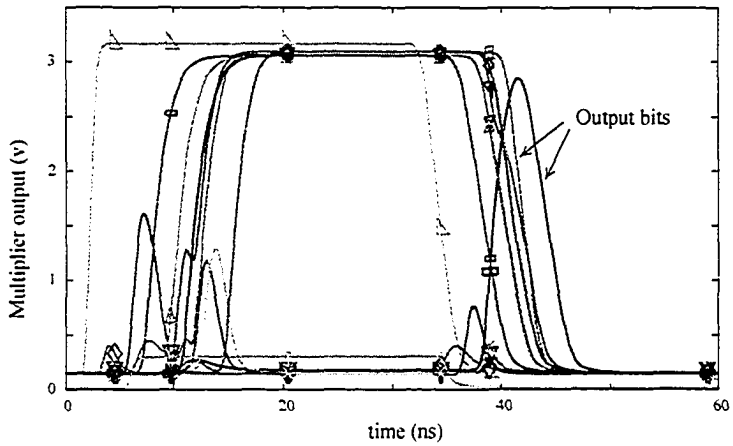


Figure 3.18 Hspice simulation of a section of the 32x32-bit CNN-based binary multiplier.

3.4.2 Impact of the CNN-based Binary Multiplier on Substrate Noise

The $n \times n$-bit carry-save binary multiplier presented in Section 3.4 consists mainly of 1-bit binary full adders, connected in a carry-save reduction tree, and a multi-bit binary adder in the final stage. Since the CNN-based addition process improves switching noise, as discussed in Section 3.3.2, one can conjecture that the CNN-based multiplier, as a collection of CNN-based addition processes, will also improve switching noise. The 32x32-bit CNN-based carry-save binary multiplier developed in the previous section has binary inputs and outputs and the multiplication process itself is carried out using parallel CNN-based addition. To compare switching noise, a 32x32-bit standard digital binary multiplier was designed using the carry-save structure of Figure 3.17. Numerous Hspice simulations were performed on random operands and switching noise of each simulation was recorded. The worst case switching noise of the CNN-based binary multiplier is plotted against the worst case switching noise of the standard digital binary multiplier in Figure 3.19.

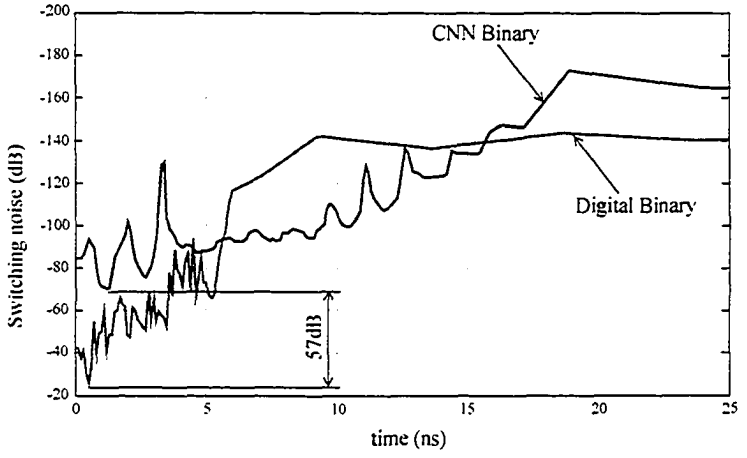


Figure 3.19 Switching noise of the CNN-based and standard digital 32x32-bit binary multipliers.

The CNN-based binary multiplier achieves 57dB improvement in switching noise over the standard digital binary multiplier. This measurement was taken as the difference between the maximum values of the switching noise of the CNN-based multiplier and the digital multiplier during one multiplication process. As in the case of binary addition, the maximum values of switching noise occur at the beginning of the multiplication process when the binary operands are first applied. Then switching noise decreases as the multiplication proceeds toward the final product. The CNN-based multiplier smooths out the transitions over a longer period of time which helps reduce peak switching noise.

3.5 Summary of CNN-based Binary Arithmetic

In this chapter, a general procedure to perform binary arithmetic using analog CNNs was developed. New equations were derived to perform the sum and carry functions in binary addition. The new equations were defined as continuous functions to facilitate mapping them into the analog domain. The equations were synthesized using simple analog circuits including: summing nodes, current mirrors, and absolute function generators. Following the general procedure, a 1-bit CNN-based binary full adder (CNNBFA) was developed. The CNNBFA uses the derived sum and carry continuous functions to describe template

connections to neighboring cells. Hspice simulations of the CNNBFA proved that the CNN network will converge for all possible binary inputs.

Similar to the standard digital 1-bit full adder, the CNNBFA accepts binary inputs (operands and carry in) and produces binary outputs (sum and carry out). This property gives circuit designers the ability to use the CNNBFA as an enabling building block to develop multi-bit CNN-based binary adders with arbitrary sizes. It also provides circuit designers with a full adder unit that can be embedded in existing, more complex, circuit architectures without the need to re-design the circuit blocks.

To illustrates the scalability of the CNNBFA, a 32-bit CNN-based binary adder was developed by cascading 32 CNNBFA units. The CNN-based adder converged for all test patterns applied to it. The smooth transitions of the CNN nodes achieved improvement of 50dB in switching noise and 20dB in cross talk over standard digital adder operating at the same speed. A 32x32-bit CNN-based carry-save binary multiplier was also developed to demonstrate the compatibility of CNNBFA with standard circuit designs. Extensive Hspice simulations show that the CNN-based multiplier improves switching noise by 57dB compared to a 32x32-bit carry-save binary multiplier implemented using standard digital logic. All circuits were developed using the same 0.35μm CMOS process technology.

Chapter 4

Binary Signed-Digit Arithmetic Using CNNs

A novel methodology for building CNN binary signed-digit (BSD) arithmetic circuits using analog cellular neural networks is described in this chapter. The work extends the concepts developed in Chapter 3 to introduce an original application of CNNs to binary signed digit arithmetic. In these architectures, the signed-digit radix-2 number representation with symmetrical digit set $\{\bar{1}, 0, 1\}$ is coupled with low-precision bi-directional current-mode analog components in a novel way that combines the computational capability of analog circuits and noise immunity of digital components. The structures use a new class of current-mode CNN that has three stable states to match the three values of the digit set. Although switching noise is the primary concern, the designs incorporate all the advantages of signed-digit arithmetic such as reduced circuit complexity and reduced routing area. The chapter is organized as follows. In Section 4.2, an overview of the binary signed-digit number system is given and the addition algorithm is explained. In Section 4.3, a practical technique to implement a BSD adder unit in the CNN framework is presented. First, a new class of CNN featuring a fundamental 3-state cell is introduced. This facilitates mapping the 3-valued number system naturally into the new class of CNN. Subsequently, the BSD addition algorithm is analyzed and new functions that govern connections to neighbor

cells are defined. The design of a BSD adder unit is then presented and convergence is illustrated using Hspice simulations. The designs of a 32-digit CNN-based BSD adder and a 32x32-digit CNN-based BSD multiplier are presented in Section 4.4 and Section 4.5 respectively. The impact of the new designs on DSN and cross talk is also examined. A summary of the work done in BSD is given in Section 4.6

4.1 Introduction

The design of high speed adders and multipliers has always been a challenging topic in computer arithmetic. The signed digit number system (SDNS) is a redundant number system that can be employed to further enhance the performance of the LNS computation [129], floating-point (FLP) multiplication-add fused (MAF) operation [130][131], complex multiplication [132], trigonometric function calculation [133], division [134], square rooting [135], online multiply-accumulate (MAC) operation [136], residue arithmetic [137][138], and integer multiplication [139]. The representation that has fewest nonzero digits is known as the canonic signed-digit (CSD) representation. It was shown in [140] that, on average, CSD uses 33% fewer nonzero digits than the binary number. This property justifies its adoption in high speed digital signal processing applications [141]. Canonic signed digit IIR/FIR filter coefficients result in a much smaller number of nonzero digits [142]-[144]. By combining subexpressions occurring often in coefficients, the CSD representation can be used in design automation algorithms leading to quality solutions to Multiple Constant Multiplication (MCM) problems and efficient digital filter design [145]-[148]. Several types of programmable filter architectures have also been developed and implemented using CSD [149]-[151]. Solinas and Proos introduced one kind of SD representation with minimum joint weight, named the Joint Sparse Form (JSF) [152][153]. Such a representation is useful in simplifying the circuits for the implementation of elliptic curve cryptosystems (ECC) [154]. The binary signed-digit representation (BSD) can be used in adder circuits to limit carry propagations to one position to the left by eliminating the dependency of the carry output function on the carry input signal [155]-[157]. With limited carry propagation, operations can be performed in parallel for all digits of two arbitrary size numbers [137],[157][158]. This means that

addition (and subtraction) of two BSD numbers can be performed in a constant time independent of the length of the operands. Consequently, fast computations can be performed in a parallel system. On the other hand, in a conventional ripple-carry adder the worst-case propagation delay is proportional to the size of the adder. This is because the carry may propagate from the least significant bit to the most significant bit. This advantage of BSDNS becomes more noteworthy in applications requiring arithmetic operations with large operand sizes [159]. Another important property of the BSDNS is that individual digits carry their own sign and separate sign information is not necessary. If joined with bi-directional current-mode circuits, this property can potentially reduce the amount of interconnect and routing complexities since multivalued signals convey more information than binary signals, thus requiring less amount of interconnects to transmit a similar bandwidth of information [160]-[162]. Efficient sign detection of the operands can further improve the performance of applications [163][164]. The choice of BSDNS in this work is made because a sufficient noise margin can be obtained using radix-2 SD arithmetic circuits compared to higher radix SD arithmetic circuits. Also, using high-radix number representations require complex analog components such as threshold detectors and comparators whose complexity approaches that of A/D converters. In addition, using analog CNNs permits a direct trade-off between circuit speed and power consumption, and therefore presents a design alternative to the pure digital circuit with fast-clocked pipelined registers with high instantaneous power consumption. Intuitively, reducing the instantaneous power consumption reduces switching noise; which is the ultimate goal of this research initiative. Obviously, analog circuits are not expected to equal the efficiency of power consumption of standard digital logic solutions [165], but in noise sensitive applications this should be an acceptable trade-off.

4.2 The Binary Signed-Digit Number System: Overview

4.2.1 Definition

The binary signed-digit number system, first introduced by Avizienis [157], is a weighted number system in which any algebraic value X can be represented by an n-digit vector as:

$$X = \sum_{i=0}^{n-1} x_i \times 2^i \tag{4.1}$$

where each digit, x_i, can assume one of the values in the symmetrical digit set $L = \{\bar{1}, 0, 1\}$, and $\bar{1} \equiv -1$.

From the definition above, BSD is a ternary number where each digit x_i carries its own sign, and there is no extra sign bit assigned to the number as a whole. An n-digit binary signed-digit number $X = [x_{n-1}, x_{n-2}, \ldots, x_1, x_0]$ has the value:

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0 \tag{4.2}$$

where X is bound by $-(2^n - 1) \leq X \leq (2^n - 1)$ and the sign of X is the sign of the most significant non-zero digit.

This sign symmetry is advantageous for arithmetic operations in that:

1. The representation for -X of a BSD number X can be obtained directly by changing the signs of all digits in X. For example, using primes to denote complementation, we have $(\bar{1})' = 1$, $1' = \bar{1}$, and $0' = 0$.
2. Various signed arithmetic operations can be performed without special conversion techniques.

The BSDNS is called a redundant number system because a given number may have more than one signed-digit representation. For example, the integer 3 can be represented in BSDNS by any of the 4-digit vectors: $0011, 010\bar{1}, 01\bar{1}0, 1\bar{1}0\bar{1}, 1\bar{1}\bar{1}1$. This property is valid except for the value 0 which has a unique representation; all digits equal to zero. The inherent redundancy of the BSD number representation allows limited carry addition and (by changing all the digit signs in the subtrahend) limited borrow subtraction. This

facilitates totally parallel operations, with an $O(1)$ time complexity of addition and subtraction of any length operands (i.e., independent of the word length, n).

4.2.2 BSD Addition

The objective of using the BSD addition algorithm is to reduce the addition time by reducing the length of the maximum carry propagation chain. The goal is to eliminate the carry propagation altogether. Given two BSD numbers, $X = [x_{n-1}, x_{n-2}, \ldots, x_1, x_0]$ and $Y = [y_{n-1}, y_{n-2}, \ldots, y_1, y_0]$, where X and Y are described by Eqn. (4.1) and $x_i, y_i \in L$, the addition of X and Y can be performed in parallel in three successive steps [165]. First, each digit x_i is added linearly to the corresponding digit y_i to form the instantaneous sum digit z_i. Second, the instantaneous sum digit z_i is used to form an intermediate sum w_i and a transfer digit t_i. Finally, the sum digit s_i is obtained by linearly adding the intermediate sum digit w_i and the previous transfer digit t_{i-1}. That is, the transfer digit acts as a form of carry to the next position. These three steps can be summarized in the following set of equations:

$$z_i = x_i + y_i \tag{4.3}$$

$$2t_i + w_i = z_i \tag{4.4}$$

$$s_i = w_i + t_{i-1} \tag{4.5}$$

where $z_i \in \{\bar{2}, \bar{1}, 0, 1, 2\}$, $w_i \in \{\bar{1}, 0, 1\}$, and $t_i \in \{\bar{1}, 0, 1\}$ are the linear sum, the intermediate sum, and the transfer digit, respectively.

To achieve parallel addition without carry propagation in BSD arithmetic, the last step in the algorithm should be performed without producing a carry. This means that final sum digit s_i has to be retained within the digit set L. This can be obtained by imposing restrictions on values of w_i in Eqn. (4.4). A problem arises only if $z_i = \bar{1}$ and t_{i-1} is

negative or $z_i = 1$ and t_{i-1} is positive. In this case, Eqn. (4.5) will generate a final sum digit $s_i = \pm 2$ $(s_i \notin L)$. To solve this problem, the restrictions on values of w_i in Eqn. (4.4) are such that $w_i \in \{0, 1\}$ when $z_{i-1} < 1$ and $w_i \in \{\bar{1}, 0\}$ when $z_{i-1} \geq 1$. These restrictions on w_i are summarized in Eqn. (4.6):

$$t_i = \begin{cases} 1 & (z_i = 2) OR ((z_i = 1) AND (z_{i-1} \geq 0)) \\ 0 & (z_i = 0) OR ((z_i = 1) AND (z_{i-1} < 0)) OR ((z_i = \bar{1}) AND (z_{i-1} > 0)) \\ \bar{1} & (z_i = \bar{2}) OR ((z_i = \bar{1}) AND (z_{i-1} \leq 0)) \end{cases} \quad (4.6)$$

From Eqn. (4.6), the final sum s_i is determined by z_i and z_{i-1} independent of the other linear sum inputs. Therefore, the carry propagation is always limited to one position to the left. This property of the BSDNS allows fast parallel operation, and the addition time is independent of the length of operands n, as discussed earlier. An example of this BSD addition process is shown in Figure 4.1. The example illustrates the addition of two BSD numbers $X = -5$ and $Y = -19$.

	2^5	2^4	2^3	2^2	2^1	2^0	weight
		0	$\bar{1}$	0	1	1	X
+		$\bar{1}$	$\bar{1}$	1	0	1	Y
		$\bar{1}$	$\bar{2}$	1	1	2	Z
		1	0	$\bar{1}$	$\bar{1}$	0	W
	$\bar{1}$	$\bar{1}$	1	1	1		T
=	$\bar{1}$	0	1	0	0	0	S

Figure 4.1 An example of BSD addition.

Similar to binary addition, BSD addition can be made quite modular because the final sum at position i depends only on the inputs at position i and i-1. Therefore, given the set of equations describing BSD addition (Eqn. (4.3)-Eqn. (4.5)) and the restrictions on the transfer digit summarized in Eqn. (4.6), one can create a functional BSD adder unit that accepts four inputs (x_i, y_i, z_{i-1}, and t_{i-1}) and generates three outputs (w_i, t_i, s_i). However, the linear additions of Eqn. (4.3) and Eqn. (4.5) can be realized in bi-directional current-mode circuits using summing nodes without active devices. A block diagram of the bi-directional current-mode BSD adder unit is shown in Figure 4.2-a where the block marked SDFA represents the restrictions on the transfer digit t_i of Eqn. (4.6). Now, designing a multi-digit BSD adder is made easy. To design an n-digit BSD adder, one can simply connect n BSD adder units as shown in Figure 4.2-b.

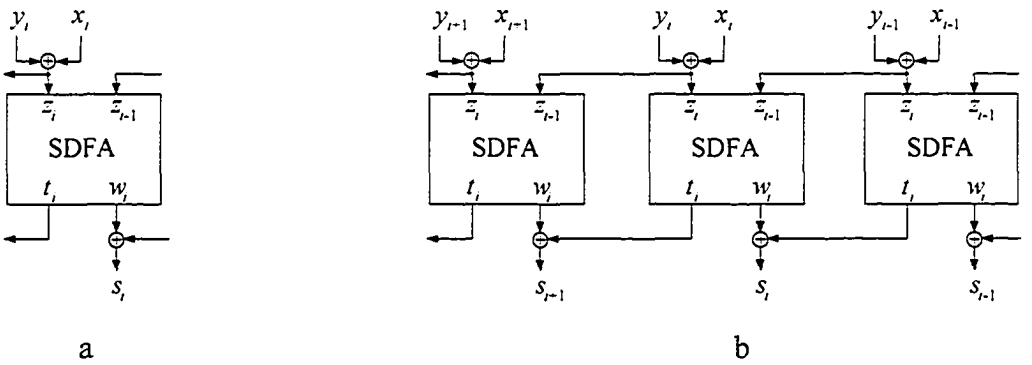


Figure 4.2 Block diagram of a BSD adder: (a) 1-digit BSD adder, (b) *n*-digit BSD adder.

It is worth mentioning that subtraction in the BSD number system is performed by a similar procedure. Noting that subtraction is essentially addition of the minuend and the negative of the subtrahend:

$$X - Y = X + (-Y) \tag{4.7}$$

Therefore, subtraction can be performed by changing the sign of all the digits of the subtrahend and adding the two operands together.

4.3 Designing a 1-digit BSD Full Adder Using CNN (CNNBSDFA)

As was mentioned in Section 2.1, the nonlinear controlled current source in a CNN cell provides two stable states as shown in Figure 2.2. This property makes CNN a natural choice for applications characterized by 2-D binary outputs. The BSDNS, on the other hand, is a ternary number system and it is extremely challenging to naturally map BSD addition into the CNN framework. Although CNN arrays have been successfully used to process images with many grey/color levels, the BSDNS is different because it requires three distinctive stable states with sufficient noise margins between the states. This restriction guarantees stable and correct arithmetic operations. The two stable states of the traditional CNN can be used to represent the digits $\bar{1}$ and 1 in the symmetrical digit set $\{\bar{1}, 0, 1\}$. Coding the digit 0 requires creating a third stable state at the center of the linear range of the original activation function. The ideal required 3-state transfer function is illustrated in Figure 4.3.

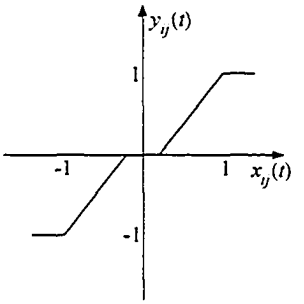


Figure 4.3 The required CNN cell activation function.

4.3.1 A 3-State CNN Cell

Current-mode circuits are analog in nature. There is no naturally available stable state because the currents flowing can take on any value. This kind of logic circuit is non-restoring, and it is often necessary to introduce some correcting circuits that will quantify the amount of current at any stage. In this section, a novel current-mode circuit design is

introduced that provides three stable states using continuous feedback signals. The design utilizes a 3-input median selector which orders the input signals based on the instantaneous magnitude and then finds the value in the middle of the sorted list. In this section a brief analysis of the median extractor is given. A detailed analysis can be found in [166].

Given n real input values In_i, $i = 1$ to n, there is always a permutation $\pi(i)$ of the indices such that the outputs, defined as $Out_i = In_{\pi(i)}$, are sorted by value:

$$Out_1 \geq Out_2 \geq \ldots \geq Out_n \tag{4.8}$$

If the number of input signals is odd, $n = 2m + 1$, the median value is defined as the value in the middle of the sorted list:

$$Out_{m+1} = med(In_1, In_2, \ldots, In_{2m+1}) \tag{4.9}$$

such that the same number of input values are greater than or equal to the median as the number of input values less than or equal to the median. The 3-input median extractor, shown in Figure 4.4, is implemented with the pMOS series transistors M21-M31, M22-M32, and M23-M33. These three groups of transistors provide bias for the matched pMOS transistors M11-M13 and the bleeding transistor M41. The equilibrium condition requires that at least one of these bias paths be "on", therefore, any two of the nodes A_1-A_3 have a low potential.

Assuming, without loss of generality, that $V_{in1} < V_{in2} < V_{in3}$, the corresponding drain currents will be $I_1 < I_2 < I_3$. In equilibrium, node A_1 will be saturated high, node A_3 will be saturated low, and $I_0 = I_2$ to maintain a suitable value for the node A_2 potential. In this case, the M22-M32 path provides the common bias for M11-M13 and M40-41. It is important to note that M3 and M11 are in the linear region. Therefore, even in the case of a large input voltage difference $V_{in2} \ll V_{in3}$, the I_3 drain current will not become dominant

and "steal" all the bias current I_{bias}. The DC characteristic curve of the 3-input median circuit of Figure 4.4 is shown in Figure 4.5 where V_{in2} and V_{in3} are held constant at 1 and 2 volts respectively while V_{in1} is swept from 0 to 3 volts. The median transfer characteristic has a unit gain in the range between V_{in2} and V_{in3} (1V-2V in the graph).

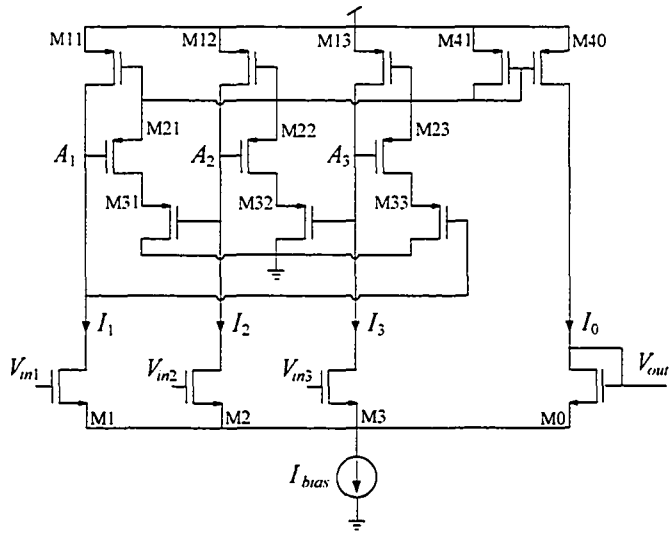


Figure 4.4 Schematic of the 3-input median extractor.

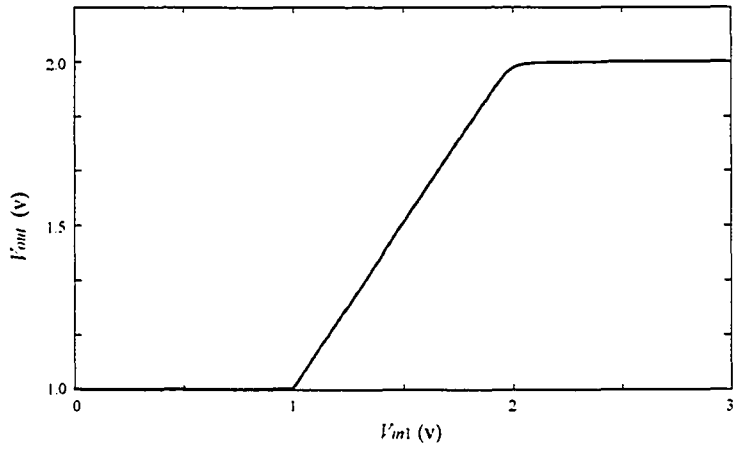


Figure 4.5 Transfer characteristics of the 3-input median extractor.

Now, consider changing the function of the diode connected transistor M0 to subtract a mirror of the input current, produced by V_{in1}, in a feedback loop as shown in Figure 4.6. The gain will be zero in the range between V_{in2} and V_{in3}. The gain outside this range will be an inverted ratio of the input voltages. The equilibrium equation of this circuit can be written as:

$$\frac{R_1 V_{in1} + R_2 V_{out}}{R_1 + R_2} = med\left\{\frac{R_1 V_{in1} + R_2 V_{ref}}{R_1 + R_2}, V_{in2}, V_{in3}\right\} \tag{4.10}$$

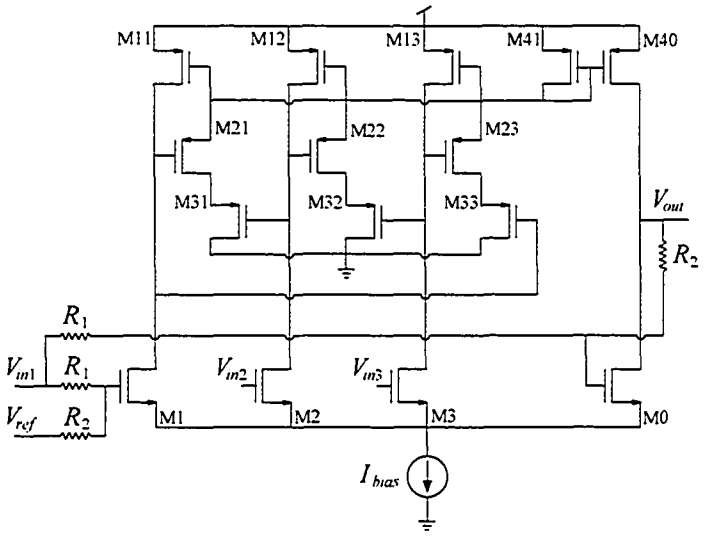


Figure 4.6 Schematic of the 3-state circuit.

This circuit has three different operating zones that depend on the input voltage V_{in1}. Assuming, without loss of generality, that $V_{in2}<V_{in3}$, the circuit operation can be described by Eqn. (4.11).

$$V_{out} = \begin{cases} \left(1+\frac{R_1}{R_2}\right)V_{in2} - \frac{R_1}{R_2}V_{in1} & V_{in1} < \left(1+\frac{R_2}{R_1}\right)V_{in2} - \frac{R_2}{R_1}V_{ref} \\ V_{ref} & \left(1+\frac{R_2}{R_1}\right)V_{in2} - \frac{R_2}{R_1}V_{ref} < V_{in1} < \left(1+\frac{R_2}{R_1}\right)V_{in3} - \frac{R_2}{R_1}V_{ref} \\ \left(1+\frac{R_1}{R_2}\right)V_{in3} - \frac{R_1}{R_2}V_{in1} & V_{in1} > \left(1+\frac{R_2}{R_1}\right)V_{in3} - \frac{R_2}{R_1}V_{ref} \end{cases} \quad (4.11)$$

The width of the null gain range is determined by the input voltages V_{in2}, V_{in3}, V_{ref} and the ratios of the resistors. The slope of the curve is determined by the feedback resistors ratio. To compensate for the negative slope, the negative output of the CNN cell is used instead. A Hspice simulation of the 3-state CNN cell using this solution is shown in Figure 4.7.

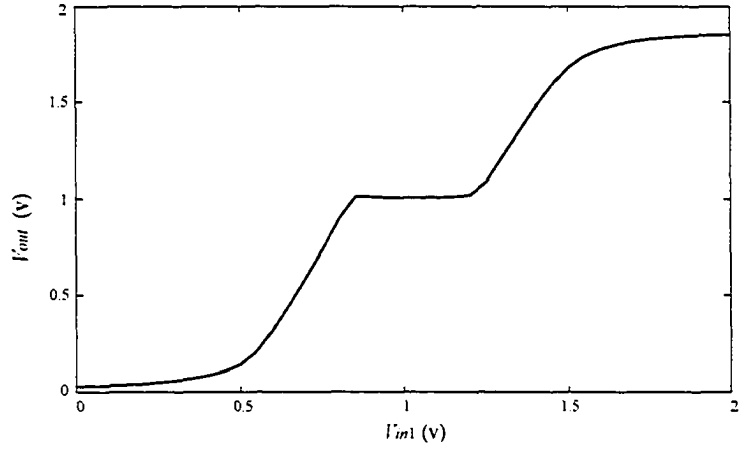


Figure 4.7 Transfer characteristics of the 3-state CNN cell.

4.3.2 CNNBSDFA Templates Design

The restrictions on the transfer digit t_i in Eqn. (4.6) can be broken into several analog primitive functions that can be then processed using the 3-state CNN cells described in the previous section. Consider the CNN structure shown in Figure 4.8-a to implement the SDFA of Figure 4.2. This structure consists of four 3-state CNN cells. The nonlinearity of the CNN cell output in column position $j = 4$ can be used to restrict the value of the

instantaneous sum z_i to one of the digits of the BSDNS. A new intermediate analog signal can be defined for the CNN cell in column position $j = 3$ as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot (y_{i+1,j} + y_{i+1,j+1}) \tag{4.12}$$

where the input signals $y_{i+1,j}$ and $y_{i+1,j+1}$ represent the thresholded outputs of the instantaneous sum at row positions i and $i-1$ respectively. Template connections for the transfer digit t_i can then be expressed as a continuous function of the outputs of neighbor CNN cells:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot (y_{i+2,j} + y_{i+2,j+1} - y_{i+1,j}) \tag{4.13}$$

Given the value of the transfer digit t_i in Eqn. (4.13), one can use Eqn. (4.4) to solve for the intermediate sum w_i. Template connections to the intermediate sum w_i can be directly mapped as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot (y_{i+3,j} - 2y_{i+1,j}) \tag{4.14}$$

A block diagram of the CNN implementation of the BSDFA is shown in Figure 4.8-b.

Figure 4.8 Representation of the CNN-based 1-digit SD adder: (a) CNN grid, (b) block diagram.

4.3.3 CNNBSDFA Hspice Simulation

The inputs to the CNNBSDFA, shown in Figure 4.8-b, z_i and z_{i-1}, can take any value from the digit set $\{\bar{2}, \bar{1}, 0, 1, 2\}$ while the outputs, t_i and w_i, can only take values from the BSD digit set $\{\bar{1}, 0, 1\}$. However, for correct BSD addition operation, the value of the transfer digit t_i is governed by the rules given in Eqn. (4.6). These rules determine the value of t_i based on the values of both z_i and z_{i-1}. The value of the intermediate sum digit w_i can then be obtained by substituting the values of z_i and t_i in Eqn. (4.4). The functional operation of the CNNBSDFA in Figure 4.8-b can be summarized in the truth table of Table 4.1.

Table 4.1 Truth table of BSD addition (* represents don't care).

z_{i-1}	z_i	w_i	t_i
*	2	0	1
2	1	$\bar{1}$	1
1	1	$\bar{1}$	1
0	1	$\bar{1}$	1
$\bar{1}$	1	1	0
$\bar{2}$	1	1	0
*	0	0	0
2	$\bar{1}$	$\bar{1}$	0
1	$\bar{1}$	$\bar{1}$	0
0	$\bar{1}$	1	$\bar{1}$
$\bar{1}$	$\bar{1}$	1	$\bar{1}$
$\bar{2}$	$\bar{1}$	1	$\bar{1}$
*	$\bar{2}$	0	$\bar{1}$

The truth table was used as a guide to test the CNNBSDFA for correct operation. All 25 sets of inputs (including the don't cares) were applied in sequence and the outputs were probed for plotting. The outputs of each of the Hspice simulations are shown in Figure 4.9. The maximum delay for the output signal to reach 90% of its final value was measured as 6.12*ns*. This delay is comparable to the delay of a voltage-mode BSD adder reported using 0.18μm CMOS technology in [159] and one-eighth the delay reported using a negative differential resistance method in [161]. Although the delay is larger than

that of the corresponding CNN-based 1-bit binary adder, nevertheless a multi-digit BSD adder is much faster than a binary adder of the same size. This is because the delay is constant for the case of the BSD adder. On the other hand, the delay of the binary adder increases linearly with the size of the operands. It is clear from the simulations that the CNNBSDFA converges to the correct output values for every possible input set. This property guarantees the convergence of more complex circuits as will be shown next.

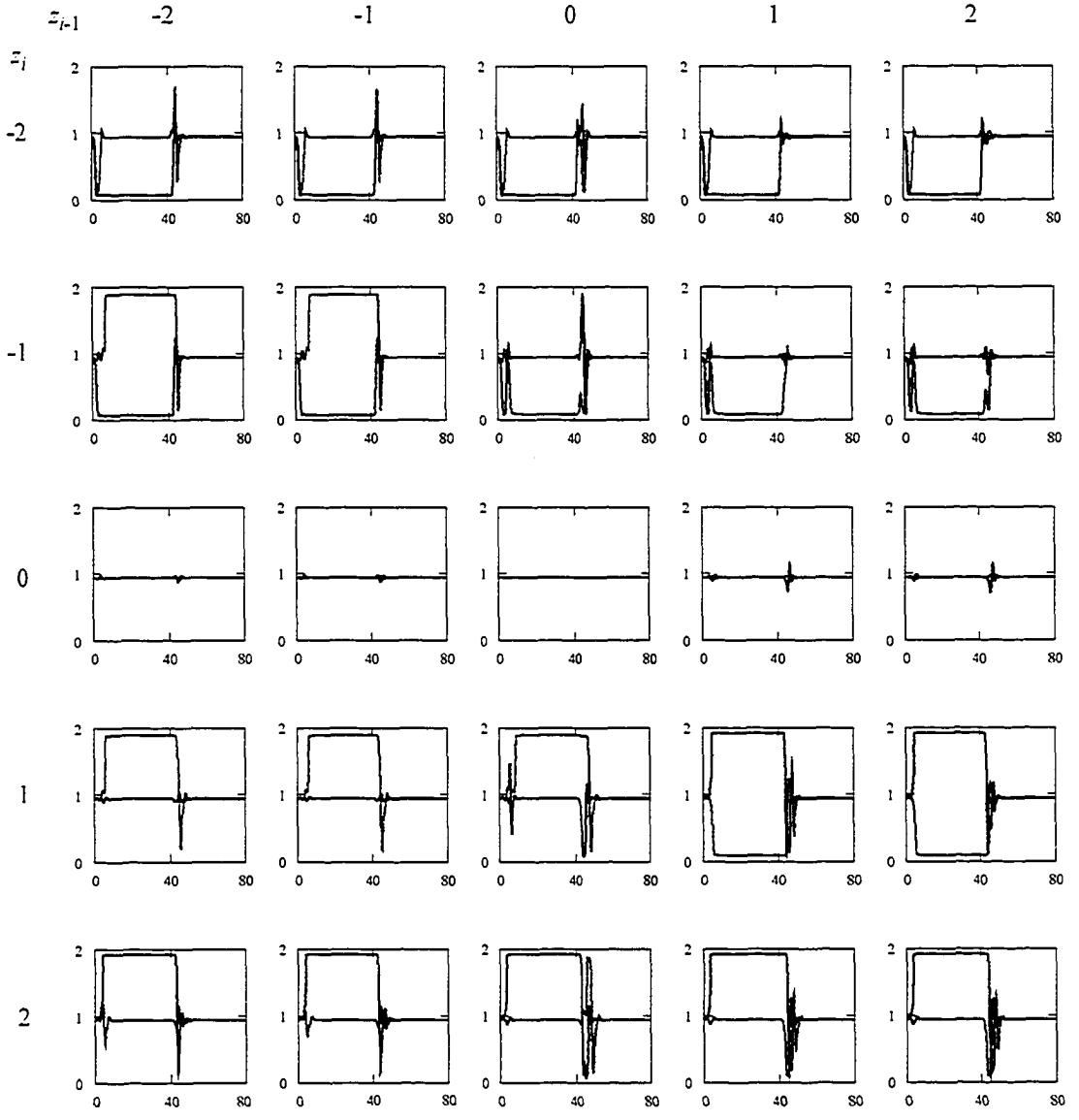


Figure 4.9 Hspice simulation of the CNNBSDFA.

4.4 CNNBSDFA Design Scalability

The CNNBSDFA can be used in a similar way to the traditional SDFA in Figure 4.2-b to develop multi-digit BSD adders of arbitrary sizes. The operation is also similar where the operands at position i are summed linearly using a summing node to form z_i. The instantaneous sums z_i and z_{i-1} are used to develop the transfer digit t_i and the intermediate sum w_i in the CNNBSDFA. The transfer digit t_{i-1} is linearly summed to the intermediate sum w_i using a summing node to form the sum signal s_i. The connections between CNNBSDFA units to construct an n-digit CNN-based BSD adder is shown in Figure 4.10. The complexity of the adder is $O(n)$ because the adder uses n units of the CNNBSDFA.

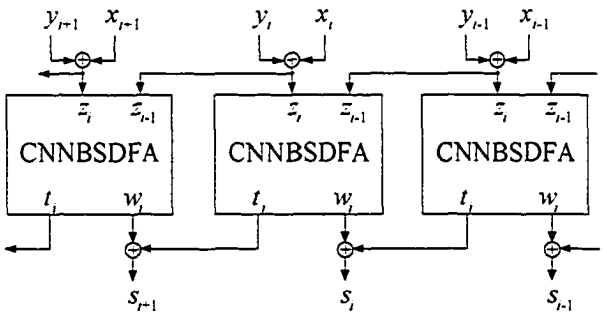


Figure 4.10 Block diagram of an *n*-digit CNN-based BSD adder.

4.4.1 A 32-digit CNN-based BSD Adder

To illustrate the scalability of the CNNBSDFA, a 32-digit CNN-based BSD adder was developed. The adder is constructed by cascading 32 units of the CNNBSDFA and connecting each CNNBSDFA to its immediate neighbors as shown in Figure 4.10. The inputs z_{i-1} and t_i to the first CNNBSDFA are set to zero and the first sum digit s_0 is equal to the value of the intermediate sum digit w_0. A random number generator was used to generate test patterns for the adder where each operand is bound by $-(2^{32}-1) \le X, Y \le (2^{32}-1)$. The adder was simulated using Hspice and the outputs of

each simulation were verified against the correct sum values of the test pattern. A small part of one of the simulations is shown in Figure 4.11. In this simulation, the instantaneous sum input pattern is $\bar{2}\bar{1}\bar{2}0\bar{2}1\bar{2}\bar{2}\bar{1}\bar{1}0\bar{1}\bar{1}20010211\bar{2}\bar{2}$ and the corresponding sum output is $\bar{1}00\bar{1}00\bar{1}\bar{1}0\bar{1}0\bar{1}\bar{1}000\bar{1}\bar{1}\bar{1}00\bar{1}\bar{1}0$. Here logic "1" is mapped as +1.8v, logic "0" as +1v, while logic "-1" is mapped as +0.2v. The 32-digit adder converged for all applied test patterns. This result is not surprising since, from Section 4.3.3, the 1-digit CNNBSDFA converges for all possible inputs.

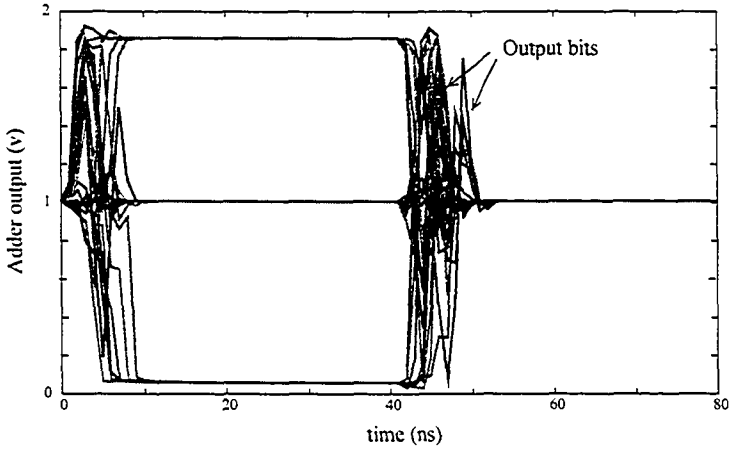


Figure 4.11 Hspice simulation of a section of the 32-digit CNN-based BSD adder.

4.4.2 Impact of the CNN-based BSD Adder on Substrate Noise

Switching noise was recorded for each of the Hspice simulations performed in the previous section. The worst case switching noise is plotted in Figure 4.12 against the worst case switching noise for a 32-bit standard digital adder. Both adders operate at the same speed, the speed determined by the CNN-based BSD adder. The CNN-based BSD adder reduces switching noise by 61dB over the 32-bit traditional digital binary adder. This is an improvement of 11dB over the CNN-based binary adder presented in Chapter 3.

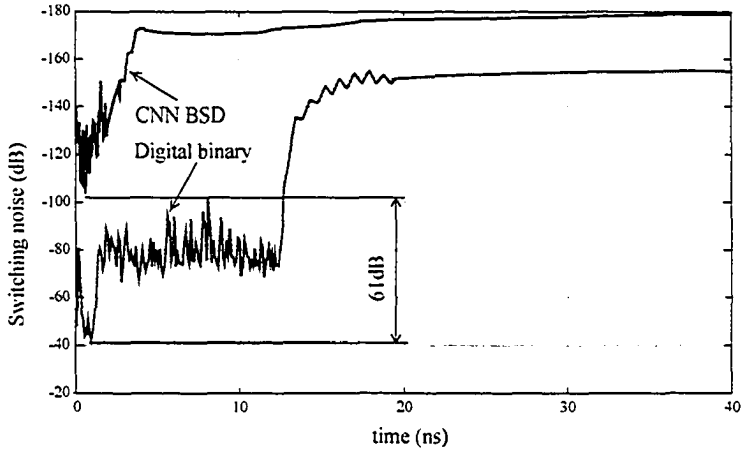


Figure 4.12 Switching noise of the CNN-based 32-digit BSD adder and 32-bit standard digital binary adder.

Cross talk is plotted in Figure 4.13 for both adders. The CNN-based BSD adder reduces cross talk by more than 23dB compared to that of the digital adder. This is an improvement of more than 3dB over the corresponding CNN-based binary architecture.

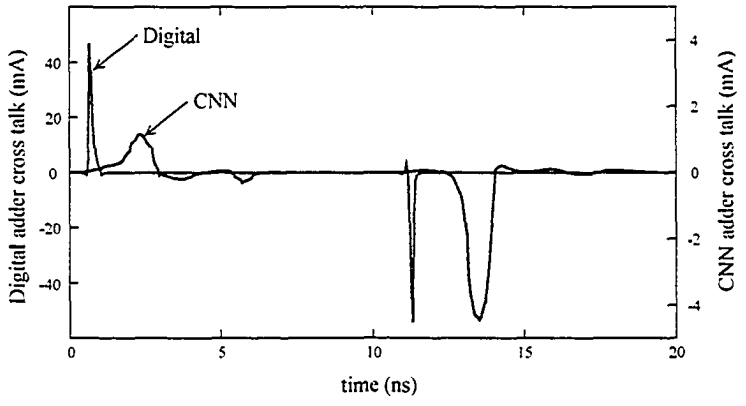


Figure 4.13 Cross talk of the CNN-based 32-digit BSD adder and 32-bit standard digital binary adder.

4.5 CNNBSDFA Design Compatibility

To the circuit designer, the CNNBSDFA unit has the same functionality and input and output pins as the traditional SDFA unit (compare Figure 4.2-b and Figure 4.10). This property allows the circuit designer to replace SDFA units in existing complex circuits with CNNBSDFA units and obtain the same functionality of the complex structure, but with the added advantage of reduced switching noise and cross talk. To illustrate this concept, consider the BSD multiplier design by Kawahito et el [167] for four-input addition of partial products in the first level of the binary tree. This structure is chosen because it speeds up the multiplication and reduces the number of full-adder modules and interconnections. The algorithm is repeated here for convenience.

Since the inputs are two's complement binary number representations, the multiplicand $X = [x_{m-1}, x_{m-2}, \ldots, x_1, x_0]$ and the multiplier $Y = [y_{n-1}, y_{n-2}, \ldots, y_1, y_0]$, are expressed as:

$$X = x_{m-1} \times 2^{m-1} + \sum_{i=0}^{m-2} x_i \times 2^i \tag{4.15}$$

$$Y = y_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} y_i \times 2^i \tag{4.16}$$

where n and m are even integers. The following expression of X is obtained by using the digit $\bar{x}_i = 1 - x_i$ and substituting $x_i = 1 - \bar{x}_i$ into Eqn. (4.15):

$$X = \bar{x}_{m-1} \times 2^{m-1} - \sum_{i=0}^{m-2} \bar{x}_i \times 2^i - 1 \tag{4.17}$$

The product P_j of X and an arbitrary digit y_i of Y is given by:

$$P_j = y_j X = \sum_{i=0}^{n-1} p_{i,j} \times 2^i \tag{4.18}$$

where $p_{i,j}$ is a partial product. When j is even, Eqn. (4.15) is used for the generation of product P_j, and when j is odd, Eqn. (4.17) is used. Thus, the partial products $p_{i,j} \in \{0, 1\}$ when j is even and the partial products $p_{i,j} \in \{\bar{1}, 0\}$ when j is odd except for the most significant digits. Therefore, the linear sum $z_{i,j}$ of four successive partial products:

$$z_{i,j} = p_{i,4j} + p_{i-1,4j+1} + p_{i-2,4j+2} + p_{i-3,4j+3} \tag{4.19}$$

obviously belongs to the set $\{\bar{2}, \bar{1}, 0, 1, 2\}$. This means that four product operands, P_{4j}, P_{4j+1}, P_{4j+2}, and P_{4j+3}, can be added in parallel by using the CNNBSDFA adder designed in Section 4.3.

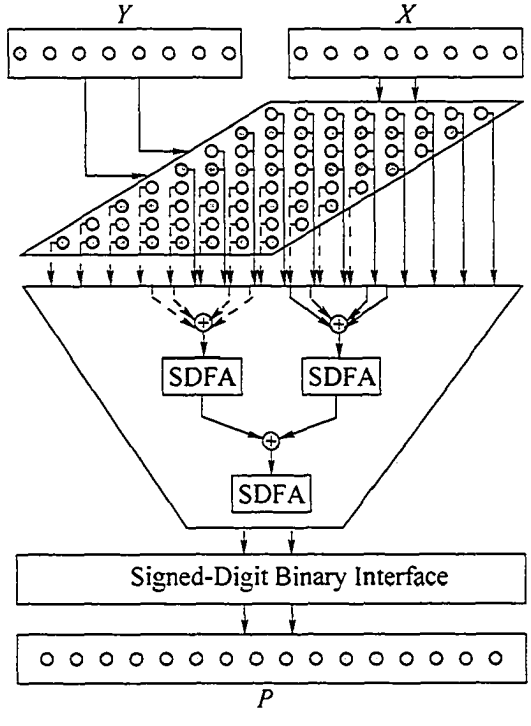


Figure 4.14 Block diagram of the BSD multiplier.

Accordingly, the binary tree was modified as shown in the 8x8-digit multiplier structure in Figure 4.14. Since every four partial product operands are added in parallel, the number

of operands is reduced by one fourth at the first level of the binary tree. consequently, the number of SD full adder levels is reduced to

$$\Gamma = \lceil \log_2(n/4) \rceil \tag{4.20}$$

where $\lceil a \rceil$ denotes the smallest integer such that $\lceil a \rceil \geq a$.

4.5.1 A 32x32-digit CNN-based BSD Multiplier

In this section, a 32x32 multiplier, that has the structure as Figure 4.14, is designed. Similar to the binary multiplier of Figure 3.17, the BSD multiplier consists of three main parts: a partial product generator matrix that forms the 32x32 partial products, a binary tree that reduces the partial products into two operands in parallel, and an adder that adds the final two operands. However, the multiplier uses binary inputs and outputs and internally uses BSD redundant binary numbers with the digit set $\{\bar{1}, 0, 1\}$. A BSD signal between tree levels is represented as a bi-directional current on a single wire. From Eqn. (4.20), the 32x32-digit multiplier has a delay of three levels of full adders. The bi-directional wired summations are fully used, greatly reducing the number of active devices and the complexity of the interconnections. The final stage of the multiplier is a BSD-to-binary converter where the 64-digit BSD number is converted into a 65-bit binary number. The conversion is performed by a two-input binary adder such as the CNN-based binary adder presented in Section 3.3.1. The Hspice simulation of a small section of the multiplier output in BSD representation is shown in Figure 4.15. The operands are X = 10101011 and Y = 10001010 and the BSD product is P= $1\bar{1}1\bar{1}1000\bar{1}1\bar{0}0\bar{1}0$. The multiplier output in binary representation is shown in Figure 4.16 where P = 101110000101110.

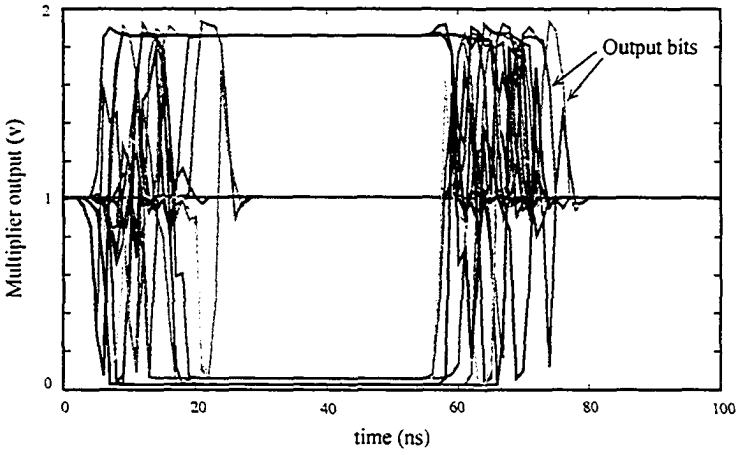


Figure 4.15 Hspice simulation of an 8-digit section of the 32x32-digit CNN-based BSD multiplier. The output is in BSD representation.

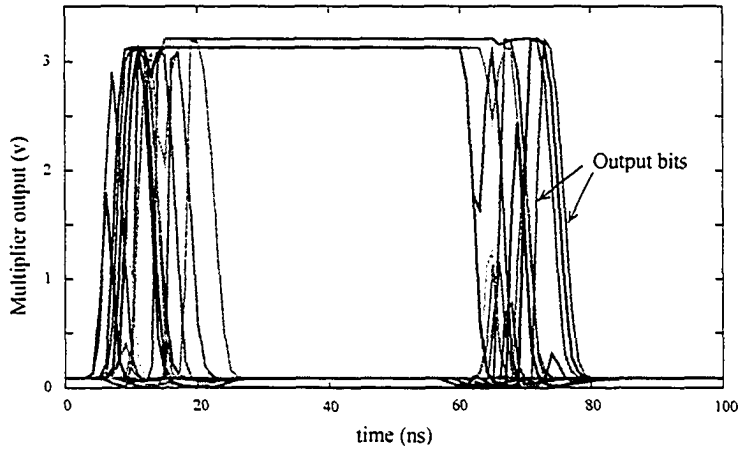


Figure 4.16 Hspice simulation of an 8-digit section of the 32x32-digit CNN-based BSD multiplier. The output is in binary representation.

4.5.2 Impact of the CNN-based BSD Multiplier on Substrate Noise

It has been shown in Section 4.4.2 that the CNN-based BSD adder improves switching noise by 11dB over the CNN-based binary adder presented in Section 3.3.2. Therefore, one can conclude that the CNN-based BSD multiplier will also show switching noise improvement over the CNN-based binary multiplier. In addition, the BSD multiplier structure used in Figure 4.14 employs fewer full adder units in the reduction tree than the

binary multiplier structure of Figure 3.17. This property will certainly show an advantage in switching noise of the BSD multiplier. Hspice simulation of switching noise of the CNN-based 32x32-digit BSD multiplier developed in the previous section and the corresponding 32x32-bit standard binary multiplier is shown in Figure 4.17. Both multipliers are designed using the same 0.35μm CMOS process technology.

The CNN-based BSD multiplier achieved 69.2dB improvement in switching noise over the traditional digital multiplier. This measurement was taken as the difference between the maximum values of switching noise of the CNN-based multiplier and the digital multiplier during one multiplication process. The CNN-based BSD multiplier also improves switching noise by 12.2dB over the corresponding CNN-based binary multiplier. This result is not surprising since, as was mentioned above, the BSD multiplier utilizes fewer full adder units.

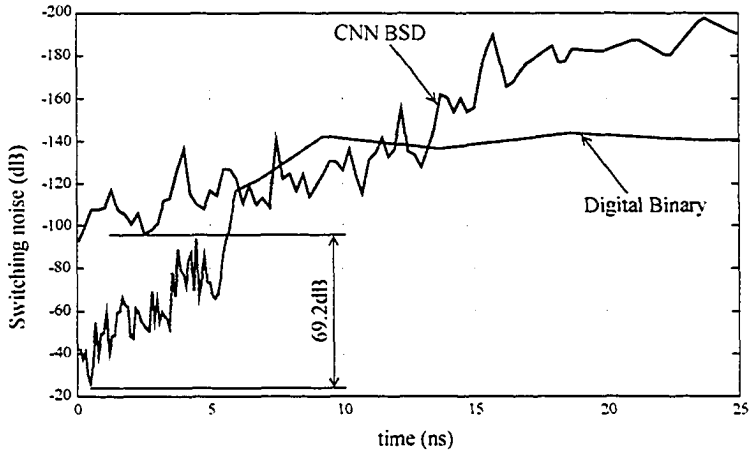


Figure 4.17 Switching noise of the CNN-based BSD and standard digital 32x32-bit binary multipliers.

4.6 Summary of CNN-based BSD Arithmetic

A general methodology to implement BSD arithmetic using the analog CNN paradigm is presented in this chapter. The greatest challenge was to discover a way to represent the 3-valued BSD number system naturally into the CNN framework. This led to the design of a new class of CNN characterized by a fundamental 3-state CNN cell. The 3-state CNN cell

was used to design a fundamental 1-digit BSD adder unit. Addition in BSDNS requires some restrictions on values of the transfer digit to ensure that the final sum will always be in BSD representation. These restrictions were defined as primitive analog functions to facilitate implementing them in the analog CNN framework. The rest of the BSD addition algorithm is implemented using summing nodes without active devices. Hspice simulations depict that the 1-digit BSD adder converges for all possible inputs to the adder. This property guarantees convergence of any complex circuit that uses the 1-digit BSD adder as an embedded block.

To demonstrate the scalability of the 1-digit BSD adder, a 32-digit BSD adder was designed by cascading 32 units of the 1-digit BSD adder. The BSD adder is tested using Hspice for different random inputs. Switching noise and cross talk are also simulated and compared to that of a 32-bit traditional binary adder operating at the same speed. The use of bi-directional current summing nodes contributed to the reduction of switching noise by a large degree. Although the delay of the 1-digit BSD adder is larger than that of the CNN-based 1-bit full adder, the multi-digit BSD adder is much faster than the same size multi-bit binary adder. The key difference is that the delay for a BSD adder is always constant regardless of the number of digits being added. On the other hand, in the binary case, the delay increases linearly with the size of the operands.

A CNN-based 32x32-digit BSD multiplier is also designed to illustrate the compatibility of the 1-digit BSD adder with existing structures. The key feature of the multiplier is the addition of four operands in the first level of the reduction tree. This feature reduces the number of BSD full adders by half in the first level of the reduction tree. It also reduces the interconnections considerably and renders the circuit more compact. Moreover, reducing the number of full adders has a desirable effect on switching noise since the instantaneous supply current is reduced. Hspice simulations show that the CNN-based BSD multiplier reduces switching noise to unprecedented levels. Not surprisingly, the switching noise of the BSD multiplier is 12.2dB lower than that of the corresponding CNN-based binary multiplier.

Chapter 5

Double-Base Number System Arithmetic Using CNNs

This chapter introduces a new architecture and circuitry for implementing addition and non-zero digit reduction for the double-base number system (DBNS), a recently introduced highly redundant number system with a 2-dimensional representation. This chapter builds on previous work to implement non-zero digit reduction using an analog cellular neural network approach, which naturally maps the 2-D DBNS representation to a 2-D analog CNN architecture. The new design exploits some of the properties of the DBNS to provide limited-carry addition with reduced complexity. The chapter is organized as follows. In Section 5.2, the DBNS and its graphical representation as 2-D maps is briefly reviewed. The addition algorithm and non-zero digit reduction to addition-ready representation are also explained. In Section 5.3, a general methodology to implement a DBNS adder unit using the CNN paradigm is presented. First, mapping DBNS addition and non-zero digit reduction algorithms into CNN image manipulation is discussed. The problems associated with the non-zero digit reduction is addressed and, subsequently, a novel and efficient non-zero digit reduction technique is introduced. Finally, the CMOS design of a CNN-based DBNS adder unit is presented which uses simple current-mode circuits and linear templates without hardware overhead. The design of a 20x20 CNN-based DBNS adder is

presented in Section 5.4. In addition, Hspice simulation results that show the effectiveness of the design are also presented and the advantage of reducing system-noise is demonstrated. A summary of the work may be found in Section 5.5.

5.1 Introduction

In the two previous chapters we have used both the standard binary representation and also the redundant signed digit representation to implement arithmetic using CNN arrays. For binary representations, the carry propagation delay in the addition operation accounts for the largest portion of the delay. The signed digit representation is redundant, but has the advantage of reducing the carry propagation to a fixed delay, independent of the word length. More information on classical redundant number representations can be found in [168]. Of course, the goal is to achieve both high speed and regularity of layout; as a result, making the arithmetic units suitable for very-large-scale-integration (VLSI) implementation. The double-base number system (DBNS), introduced by Dimitrov et al. [171], has some interesting properties related to reducing the carry propagation, and the DBNS has similar properties to the classical logarithmic number system (LNS) if an index calculus is used [171][172]. The DBNS provides more degrees of freedom than the LNS and promises efficient arithmetic implementation over the LNS for applications such as modular exponentiation in cryptography [173]. The index calculus double-base number system (IDBNS) has already been used to reduce hardware complexity in digital signal processing [174][175]. The number system has been extended to more than 2 bases and a logarithmic version of this extension, which is referred to as the multidimensional logarithmic number system (MDLNS), has also proved useful for implementing digital filters [172][176]. The canonic version of the DBNS promises carry-free addition operations by exploiting the redundancy in the representation. This property of the number system is useful not only for addition but also for multiplication. Without carry propagation, operations can be performed in parallel for all digits of a number. Consequently, the computation can be completed much faster in massively parallel systems.

The use of CNN architectures for implementing DBNS arithmetic is not as tenuous as it at first seems. The use of CNN arrays for implementing image morphology operations is well established [43] and, as will be shown, the manipulation of DBNS 2-D representations is quite similar to such image processing operations. For example, addition operations consist of simple overlays of number "images" followed by a reduction of the number system to a form suitable for further additions (addition-ready form). In [34], an initial attempt to implement non-zero digit reduction to an addition-ready form using the CNN paradigm was reported. However, this implementation has the disadvantage of considerable hardware overhead by the requirement for hysteresis in the CNN cells and also the necessity to use discrete digital logic circuits to control the switching of templates during the computation, which increases the potential of injected noise. In the following sections, a new method is presented for implementing digital arithmetic in the DBNS that eliminates these disadvantages. The new circuit uses a novel self-programmable CNN architecture where the switching of templates is performed based on the state values of the cells. After performing the addition of two addition-ready representation maps, the CNN reduces the sum back to an addition-ready representation using a simple reduction rule. This results in the most efficient implementation reported for multiple additions in the DBNS.

5.2 The Double-Base Number System: Overview

5.2.1 Definition

The double-base number system is a weighted number system that uses two bases and allows 0 and 1 as its digits. The discussion here will be limited to the original definition of the DBNS with orthogonal bases 2 and 3. This assumption will not affect the generalization of the techniques presented below.

Any integer X can be represented in the DBNS as in Eqn. (5.1) [171]:

$$X = \sum_{i,j} x_{i,j} 2^j 3^i \tag{5.1}$$

where digits $x_{i,j} \in \{0, 1\}$. It can be seen that the DBNS reduces to the binary number system for $i = 0$ and to the ternary number system for $j = 0$.

The DBNS can be represented graphically using a 2-dimensional grid by using the base 2 as the x-axis and the base 3 as the y-axis. Consider a 2-D grid with $\lceil \log_2 X \rceil$ columns and $\lceil \log_3 X \rceil$ rows where every cell (i,j) corresponds to the 2-integer value $2^j 3^i$. Any arbitrary integer $X \leq 2^k 3^m$ can be represented as a sum of 2-integers, which appears in the first $k + \lceil \log_2 3m \rceil$ columns and $m + \lceil \log_3 2k \rceil$ rows. For example, the integer number 108 can be represented in the DBNS in different ways as: 108, 72+36, 54+36+18, 54+24+18+12, 36+27+24+18+3. The geometric interpretation of the integer number 108 into DBNS-maps is shown in Figure 5.1. DBNS-maps are obtained by representing each non-zero term in Eqn. (5.1) as a black pixel, referred to as an active cell, and each zero term as a white pixel.

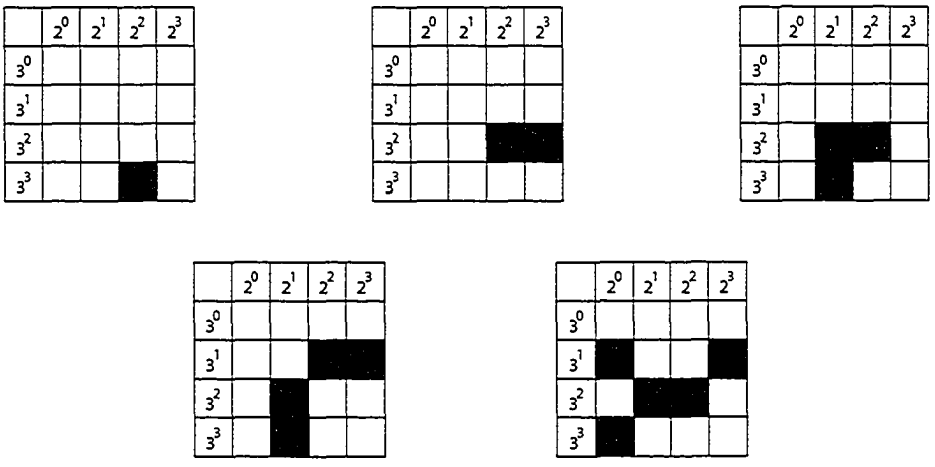


Figure 5.1 Different representations of 108 in the DBNS

It is obvious from the above example that the DBNS is a highly redundant number system where, in general, each integer X can have several representations. It is also clear that the DBNS-map allows extremely sparse representations of the integers. It is this property of sparseness that promises fast arithmetic algorithms with reduced complexity. The

representation of an arbitrary integer in the DBNS-map, using the minimum number of active cells, is called the canonic double-base number representation (CDBNR). This canonic form is not, in general, unique.

The average number of nonzero digits required to represent any integer, X, in the form of Eqn. (5.1) has a complexity with respect to the number given by Eqn. (5.2) [174]:

$$k = O\left(\frac{\log X}{\log\log X}\right) \tag{5.2}$$

The complexity is a weak function of X so one expects that the implementations of arithmetic operations will be quite efficient using the canonic form of the DBNS. The major drawback with using the canonic form is that it is a hard problem to compute the minimal representation. However, a near canonic form, referred to as the addition-ready double base number representation (ARDBNR), can be relatively easily computed with a simple greedy algorithm while retaining most of the efficiencies of computing with the canonic form [172]. In the following discussion, the ARDBNR form is always used prior to implementing any addition operation.

5.2.2 DBNS Addition

Let $I_x(i,j)$ and $I_y(i,j)$ be the DBNS maps of two integers X and Y, represented in the ARDBNR. It is known from the definition of the ARDBNR [174] that if both X and Y have an active cell in the position 2^j3^i, then the cells in the position $2^{j+1}3^i$ are non-active in both maps. Therefore the image $I_z(i,j)$ of the DBNS map of the number $Z = X + Y$ can be computed by overlaying the DBNS maps corresponding to X and Y. Any collisions (black squares coincident) are resolved using the following identity:

$$2^j3^i + 2^j3^i = 2^{j+1}3^i \tag{5.3}$$

which can be represented graphically as shown in Figure 5.2. This process is similar to the traditional carry propagation in the binary number system and the two terms *overlaying rule* and *carry propagation rule* will be used interchangeably. By definition,

the ARDBNR has no adjacent active cells in a row; this means that the carry propagation in DBNS addition is limited to one cell position to the right.

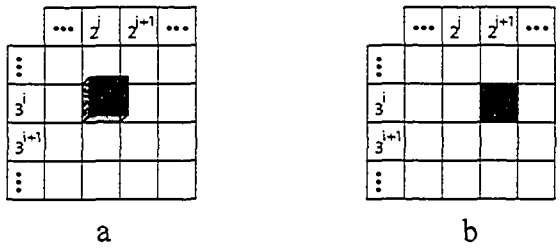


Figure 5.2 Graphical representation of the overlaying rule: (a) initial map, (b) final map.

An example of the addition process described above is shown in Figure 5.3. The image $I_Z(i,j)$ of the DBNS map of the number $Z = X + Y$ is obtained by overlying the two ARDBNR maps $I_X(i,j)$ and $I_Y(i,j)$. Note that there is a collision at position (2,1) of the overlaid image in Figure 5.3-c. The collision is solved by applying Eqn. (5.3). The two operands, $X = 91$ and $Y = 23$, are represented in the ARDBNR, but, as generally expected, the sum, Z, is not in the ARDBNR form. In order to prepare the final sum for another addition, the DBNS map of Z needs to be mapped into an ARDBNR form as will be discussed in the following section.

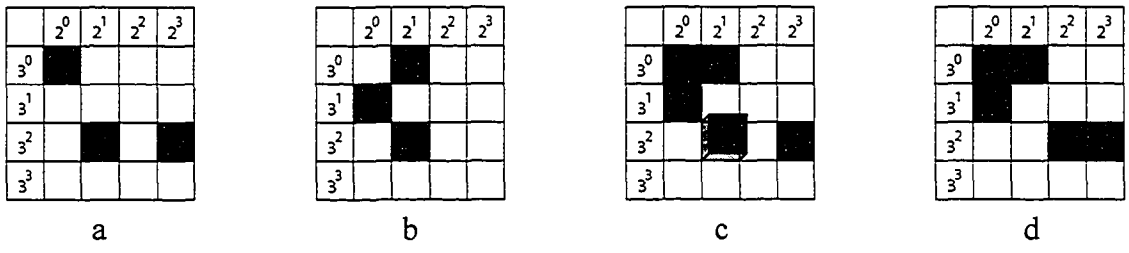


Figure 5.3 Addition in DBNS: (a) X, (b) Y, (c) map obtained by overlaying, (d) Z after applying the overlaying rule.

5.2.3 Reduction to Addition-Ready Representation

In many applications, the complexity of the computational algorithms depends on the average number of zeros representing the data; the greater the number of zeros the more efficient the algorithm [165][179]. This means that one should seek the smallest number of 2-integers (a CDBNR) to represent the operands before performing any arithmetic operation. Finding the canonical form for a DBNS representation is computationally complex; however, an efficient greedy algorithm has been introduced in [174] that can produce a near-canonic representation (NCDBNR) with a logarithmic complexity. The NCDBNR produced by the greedy algorithm is close enough to the canonic form to implement efficient arithmetic algorithms. In this section, it will be shown that using a very simple reduction rule one can prepare arbitrary DBNS-maps for a limited-carry addition process; it is only necessarily to transform the map into an addition-ready DBNR (ARDBNR) form that has no consecutive active cells lying in a row. Thus, when adding two DBNS representations, carries from any overlapping active cells will have a non-active cell to the right which can hold the doubled weight of the carry (see Section 5.2.2). It can also be seen that using ARDBNR representations eliminates carry ripple, similar to the limitation on carry ripple available by using signed digit redundant binary representations [180]. We state the following definition:

Definition 5.1 We will call two adjacent active cells in positions (i,j) and $(i,j+1)$ an active group in position (i,j).

The reduction rule that transforms an arbitrary DBNS-map into an ARDBNR map is to replace any active group (two consecutive active cells lying in a row) with a single active cell using the following identity:

$$2^j3^i + 2^{j+1}3^i = 2^j3^{i+1} \tag{5.4}$$

which has the geometric representation shown in Figure 5.4.

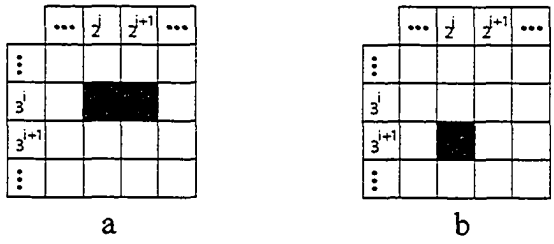


Figure 5.4 Graphical representation of the reduction rule: (a) initial map, (b) final map.

To perform non-zero digit reduction on any DBNS-map to obtain an ARDBNR-map, one needs to eliminate all adjacent active cells lying in a row in the DBNS-map. Therefore, Eqn. (5.4) needs to be applied successively to replace any active group in position (i,j) with one active cell in position $(i+1,j)$. If the cell in position $(i+1,j)$ is already active, the carry propagation rule described by Eqn. (5.3) is applied so that the cell becomes non-active and the cell in position $(i+1,j+1)$ becomes active. The non-zero digit reduction process of the image $I_Z(i,j)$ obtained in the addition example of Section 5.2.2 is shown in Figure 5.5.

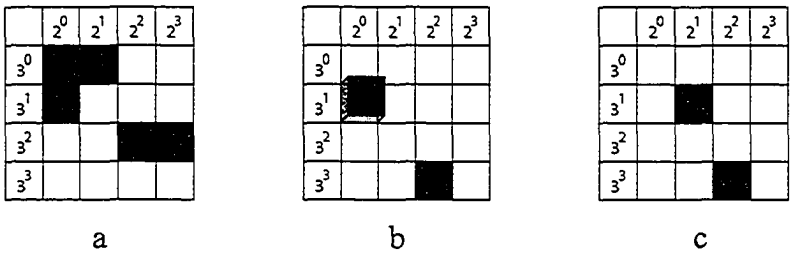


Figure 5.5 Non-zero digit reduction of Z: (a) initial map, (b) intermediate map, (c) final map.

The maximum size of the ARDBNR map obtained using the overlaying and row reduction rules can be calculated using the following theorem:

Theorem 5.1 Any $M \times N$ DBNR-map can be transformed into an ARDBNR-map that is $(M+2) \times N$ pixels at most.

Proof. If there is an active group in position (M,j), in the original DBNR-map or as a result of addition-ready transformation, the application of the reduction rule to that group will generate an active cell in position $(M+1,j)$. In the special case where an additional active cell exists in position $(M,j-1)$, the application of the reduction rule to an active group in position $(M-1,j)$ will generate an active cell in position (M,j). This new active cell together with the active cell $(M,j-1)$ can be replaced by an active cell in position $(M+1,j-1)$ which in turn, with the active cell in position $(M+1,j)$, can be replaced by an active cell in position $(M+2,j-1)$. This completes the proof.

5.3 Designing a 1-bit DBNS Adder Unit Using CNN (CNNDBNSAU)

The final sum output in DBNS addition as described in Section 5.2 is obtained in two steps: performing addition of the two operands represented in ARDBNR-maps using the overlaying rule given by Eqn. (5.3) and transforming the sum output to an ARDBNR-map using both the overlaying rule and the row reduction rule given by Eqn. (5.4). In the CNN framework, this requires the design of two templates: A template for the overlaying rule and another template for the row reduction rule. The operation of the templates is usually handled using an external control unit that is programmed to load different templates in specific order for certain periods of time [41], [181][182]. Another method was reported in [34] that used discrete logic gates attached to each cell to control template operation. In the following sections, a novel self-programmable analog CNN array that performs DBNS addition as well as reduction to addition-ready representation is presented. The network switches between the overlaying rule and the row reduction rule based on the output voltages of the involved cells without the need to use external control logic.

5.3.1 CNNDBNSAU Templates Design

Consider first designing the template for the row reduction rule described by Eqn. (5.4). Examining the graphical representation of the reduction rule in Figure 5.4 reveals that the cell at position $(i+1,j)$ should be activated if and only if the two cells at positions (i,j) and $(i,j+1)$ are active. The operation is similar to the two-input logical *AND* function with the

output voltages of the cells (i,j) and (i,j+1) representing the inputs to the *AND* function and the output voltage of the cell (i+1,j) representing the output of the *AND* function. Therefore, using a superscript notation in order to separate prior and post values on the image (array) cell nodes, the output voltage of the cell (i+1,j) can be described using the equivalent logic equation on the digits in the DBNS representation:

$$I_{\_}^{2}(i+1,j) = I_{\_}^{1}(i,j) \wedge I_{\_}^{1}(i,j+1) \tag{5.5}$$

Using the CNN notation, the continuous function of the output voltage of cell (i,j) can be written as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot (y_{i-1,j} + y_{i-1,j+1} - 1) \tag{5.6}$$

The row reduction rule replaces the active group at position (i,j) with one active cell at position (i+1,j). Therefore, once the cell at position (i+1,j) is activated, the cells at positions (i,j) and (i,j+1) should be de-activated. The output voltages of the input cells to the reduction rule can then be described using the following logic equations:

$$I_{\_}^{2}(i,j) = I_{\_}^{1}(i,j) \oplus (I_{\_}^{1}(i,j) \wedge I_{\_}^{1}(i,j+1)) \tag{5.7}$$

$$I_{\_}^{2}(i,j+1) = I_{\_}^{1}(i,j+1) \oplus (I_{\_}^{1}(i,j) \wedge I_{\_}^{1}(i,j+1)) \tag{5.8}$$

A simple and cost-effective way to map the logic equations given by Eqn. (5.7) and Eqn. (5.8) to CNN continuous functions is to use a negative mirror of the current given by Eqn. (5.6) (that is used to activate cell (i+1,j)) to de-activate the two input cells (i,j) and (i,j+1). This method ensures that the current given by Eqn. (5.6) will set cells (i,j) and (i,j+1) non-active only if a reduction rule is detected. Otherwise, the two input cells will keep their original state.

The overlaying rule given by Eqn. (5.3) can be realized in a similar way. Notice, however, from the graphical representation in Figure 5.2 that one of the two input cells is on a third dimension. This can be realized as a 3-D CNN architecture, or by applying one of the operands as initial conditions to a 2-D CNN architecture and applying the second operand

as inputs to the network. Using the second approach, the overlaying rule can be described using the following logic equation on the digits of the operands:

$$I_z(i,j+1) = I_x(i,j) \wedge I_y(i,j) \tag{5.9}$$

which can be written in continuous form as:

$$A_{ij:kl}(y_{kl}(t)) = \beta \cdot (y_{i,j-1} + u_{i,j-1} - 1) \tag{5.10}$$

As in the case of the row reduction rule, the two input cells to the overlaying rule should be de-activated when cell $(i,j+1)$ is activated. The logic function that describes this condition can be written as:

$$I_z(i,j) = I_x(i,j) \oplus I_y(i,j) \tag{5.11}$$

Here one can also use a negative mirror of the output current given by Eqn. (5.10) to de-activate the input cells.

5.3.2 Dealing with Special Cases of DBNS-maps

The previous discussion about the reduction of a DBNS-map into an ARDBNR-map (see Section 5.2.3), dealt only with one application of the reduction rule in the map. Even in the case of parallel applications of the reduction rules across the DBNS-map, the initial assumption was that the groups of active cells that participate in the reduction rules are far apart with no interaction (and interference) among the active cells. However, if we now include the possibility of such interactions, then if an active cell participates in more than one reduction rule at the same time, the network might become unstable and, even if this does not happen, may settle to a wrong output. Figure 5.6 shows a special case where an active cell in position (i,j) that can participate in two different occurrences of the row reduction rule given by Eqn. (5.4) at the same time. Cell (i,j) can participate with the cell $(i,j+1)$ to activate cell $(i+1,j)$ as shown in Figure 5.6-b. Another possibility is that cell (i,j) and cell $(i,j-1)$ are replaced by cell $(i+1,j-1)$ as shown in Figure 5.6-c. Both solutions are correct even though they give different DBNS-maps. This is due to the redundancy inherent in the DBNS. Now assume that cell (i,j) participated in the two occurrences at the

same time. This means that both cell (i+1,j) and cell (i+1,j-1) will become active as shown in Figure 5.6-d. Now one can apply the row reduction rule again on cells (i+1,j) and (i+1,j-1) and replace them with cell (i+2,j-1) as shown in Figure 5.6-e. This final value is not correct and clearly it is important to prevent the simultaneous participation of a cell in more than one network activity.

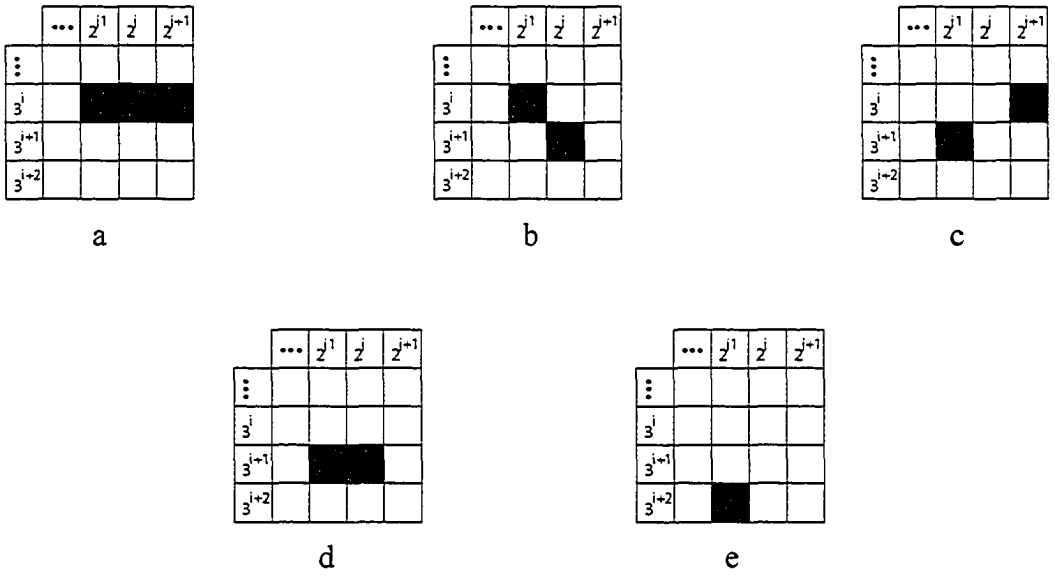


Figure 5.6 An example of simultaneous reductions: (a) initial map, (b) and (c) correct solutions, (d) and (e) wrong output.

There are two other special cases where the application of a reduction rule on an active group can affect another active cell that is participating in a different reduction rule. These two cases are illustrated in Figure 5.7. In Figure 5.7-a, the application of the reduction rule to the active group in position (i-1,j) will try to activate the cell in position (i,j). However, since cell (i,j) is already active, it will try to participate in the application of the overlaying rule. Moreover, cell (i,j) can participate in the application of the row reduction rule with the cell in position (i,j-1).

Figure 5.7-b shows a similar case where the application of the row reduction rule to the active group (i-1,j) will invoke the application of the carry propagation rule on cell (i,j).

Also, cell (i,j) can participate in the application of the row reduction rule with the cell in position $(i,j+1)$.

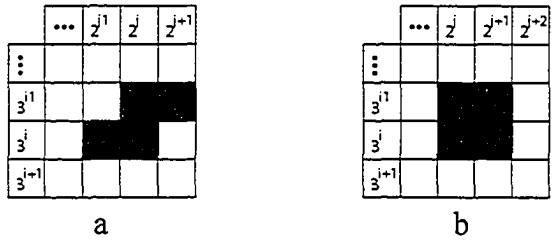


Figure 5.7 Two possible simultaneous applications of Eqn. (5.4) to the same cell.

It is clear that for correct network operation, the simultaneous participation of an active cell in more than one reduction rule should be prevented. The above situations can be summarized in the following two restrictions:

1. Any active cell (i,j) can not, at any moment of time, participate in more than one reduction rule.
2. The application of a reduction rule on any group of active cells cannot affect, at any moment of time, an active cell that is a candidate for another reduction rule.

The goal of applying the reduction rules is to generate an ARDBNR map. Therefore, giving preference for a candidate cell (i,j), which is violating the ARDBNR rule, to participate in a certain reduction rule over another, should lead to a more sparse representation. Since applying the row reduction rule replaces two active cells in row i with an active cell in row $i+1$ and applying the overlaying rule replaces two overlaying active cells in column j with an active cell in column $j+1$, it becomes natural to try to clear cells in the last row and last column first. The following theorem proves that using this method results in a DBNS-map with fewer active cells.

Theorem 5.2 Given any DBNR-map, applying the reduction rules to active groups in positions (i,j) in descending order of i and j results in a DBNR-map with fewer active cells.

Proof. Let us consider the order of scanning rows first. For any $M \times N$ DBNR-map, if there is an active group in position (i,j) $(0 \le i < M)$, there is also the possibility that another active group will exist in position $(i+1,j)$. The active group in position (i,j) cannot be chosen first because this will violate restriction 2. In this case, cell $(i+1,j)$, a member of the active group in position $(i+1,j)$, will be affected by the application of the reduction rule to the active group in position (i,j). For the case $i=M$, there will always be an empty row $M+1$ according to Theorem 5.1. Therefor the reduction rule must be applied to active groups in every row i starting from $i=M$ in descending order of i.

Now let us look at the order of scanning columns. Assume the DBNR-map has k adjacent active cells in a row. We consider two cases:

Case 1: k is even. Since the application of the reduction rule takes two active cells at a time, all the k active cells will eventually participate in a reduction rule and the order is not critical.

Case 2: k is odd. In this case, k-1 active cells will participate in $(k-1)/2$ applications of the reduction rule. The remaining active cell, might be able to participate later if an active group in row i-1 generates an active cell adjacent to it. The remaining active cell will be in position (i,j) where $0 \le j \le N-1$ if we apply the reduction rule in descending order of j, and $1 \le j \le N$ if the application were in ascending order of j. For $1 \le j \le N-1$, any active group in positions $(i\text{-}1,j\text{-}1)$, $(i\text{-}1,j)$, or $(i\text{-}1,j+1)$ can force the remaining active cell to participate in a reduction rule. For the case $j=N$, only active group in position $(i\text{-}1,j\text{-}1)$ is allowed. For the case $j=0$, active groups in positions $(i\text{-}1,j)$ and $(i\text{-}1,j+1)$ are allowed which gives an additional 33% possibility for the remaining active cell to participate in a reduction rule over the case $j=N$. Since the application of a reduction rule results in fewer active cells (two active cells are replaced by a single active cell), this completes the proof.

Figure 5.8 illustrates Case 2 where k is odd. The rows were scanned for candidates in descending order of i while the columns were scanned in ascending order of j in Figure 5.8-b and in descending order of j in Figure 5.8-c. Note that Figure 5.8-b can not be

further processed because the final DBNS-map contains no consecutive active cells in one row.

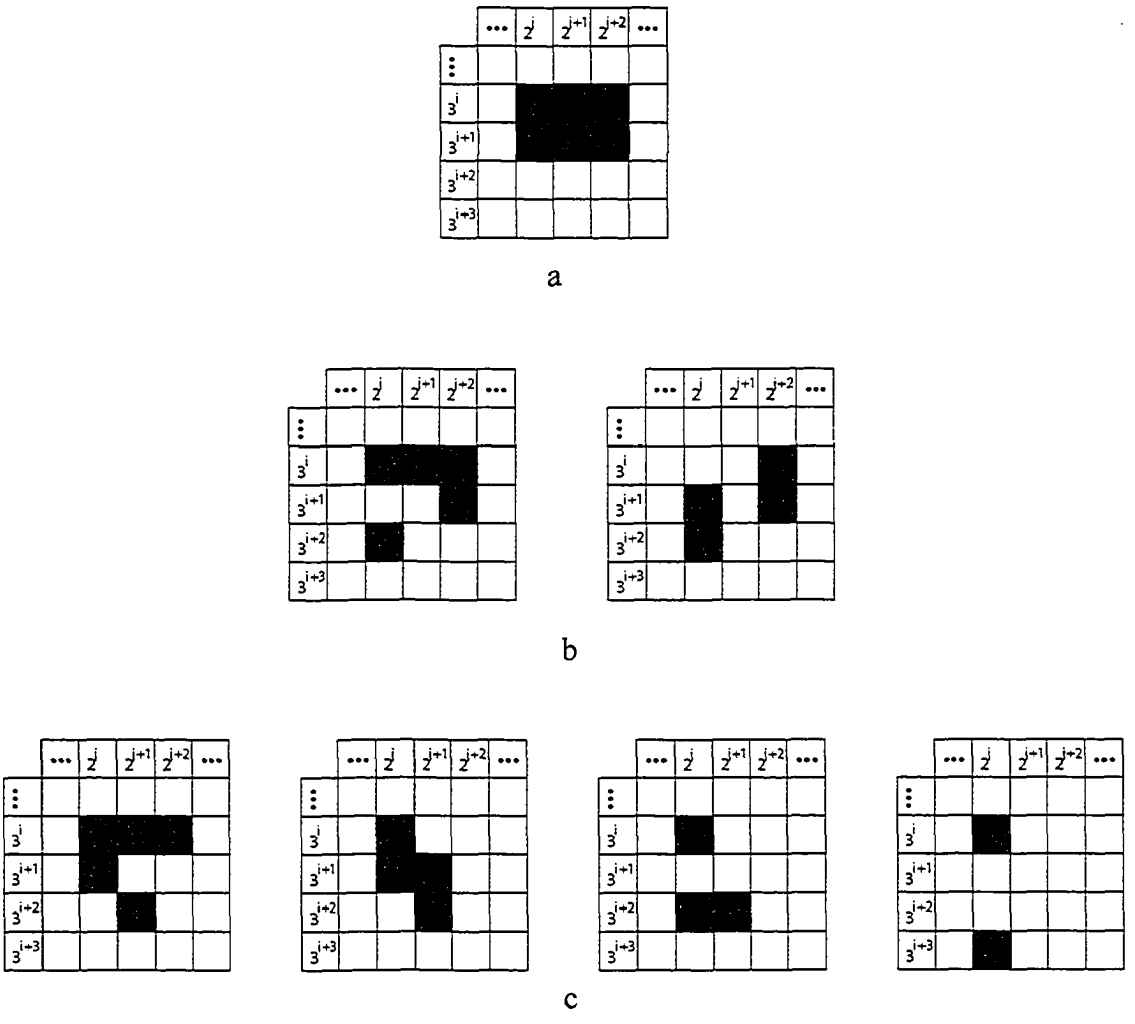


Figure 5.8 An example of the order of reduction: (a) initial map, (b) *j* in ascending order, (c) *j* in descending order.

The following theorem calculates the time required to transform any DBNR-map to ARDBNR-map using the order of reductions described above.

Theorem 5.3 Any $M \times N$ DBNR-map can be transformed to ARDBNR-map in time less than or equal to $(M + \lceil N/2 \rceil) \cdot T$, where T is the time needed to perform a single reduction.

Proof: Assume for simplicity that N is even. For any row i in the DBNR-map, any active group at position $(i,0)$ ($(i,1)$ if N is odd) will not participate in a reduction rule unless all adjacent active groups in the same row have already participated. If all cells in row i are active, completing the reductions requires time equal to $(N/2)\cdot T$ ($(\lceil N/2\rceil - 1)\cdot T$ if N is odd) since the reduction rule takes two active cells at a time. Also, for any column j in the DBNR-map, the active group in position $(0,j)$ will not participate unless all other adjacent active groups in the same column have already participated. If the map has active groups in positions (i,j) $\forall i$, completing the reductions requires time equal to $M\cdot T$. The worst case scenario would be a map with active groups in positions $(i,0)$ $\forall i$ ($(i,1)$ if N is odd) and also all cells in row M active. The time required to complete the reductions will then be $(M+N/2)\cdot T$ ($(M+\lceil N/2\rceil - 1)\cdot T$ if N is odd). For the special case where N is odd and the map has an additional active cell in position (1,0), the application of the reduction rule to the active group (0,1) will generate an active cell in position (1,1). This last active cell together with the active cell (1,0) form a new active group that needs one more time constant T to be reduced. This completes the proof.

5.3.3 CNNDBNSAU CMOS Implementation

The row reduction rule can be realized using the current-mode circuit shown in Figure 5.9. This circuit can detect the occurrence of an active group. The input current from cells (i,j) and $(i,j+1)$ is mirrored by transistor M2. A unit current is subtracted by the current source I_{sink} and the rest of the current is drained by transistor M3. The circuit outputs high current if both inputs from cells (i,j) and $(i,j+1)$ are high and zero current otherwise. When a reduction rule is detected at cells (i,j) and $(i,j+1)$, a positive current is used to activate cell $(i+1,j)$ through the current mirror M6-M7. At the same time, a negative feedback current is used to deactivate the two input cells (i,j) and $(i,j+1)$ through the current mirror M3-M5. Figure 5.10 shows how the feedback template is connected to the cells participating in the row reduction rule of Eqn. (5.4). As was mentioned in Section 5.3.1, the functions of the row reduction rule and the overlaying rule are essentially the same: replace two active cells with one active cell. The only difference is that the row reduction rule is mapped in a 2-D architecture while the overlaying rule takes on a third dimension.

Therefore, the same circuit in Figure 5.9 can be used to implement the carry propagation rule of Eqn. (5.3) with appropriate changes in the input/output connections.

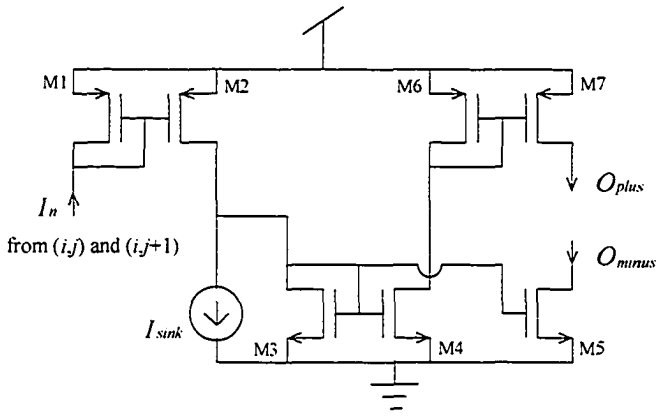


Figure 5.9 Schematic of the reduction rule.

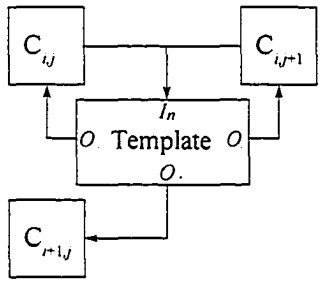


Figure 5.10 Connection to participating cells.

The circuit shown in Figure 5.9 does not prevent a cell from participating in more than one reduction rule at the same time, and neither does it enforce the order of applying the reduction rules to adjacent active cells, as described in Section 5.3.2. Consider again the situation depicted in Figure 5.11. Four groups of active cells that can take part in the row reduction rule described by Eqn. (5.4) can be identified. These four groups are outlined and marked as G1, G2, G3, and G4.

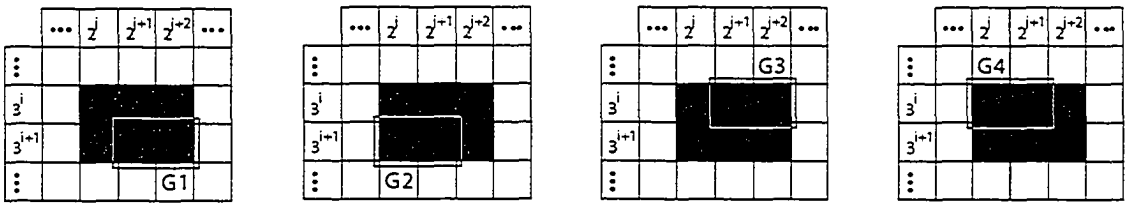


Figure 5.11 A situation where the row reduction rule can be applied to four different groups of cells.

If the order of applying the reduction rules, as described in Section 5.3.2, can be enforced, that will take care of the problem of a cell participating in more than one reduction rule at the same time. From Section 5.3.2, the row reduction rule should not be applied to group G4 unless it is not applicable to groups G1, G2, or G3 respectively. This restriction can be realized using a negative feedback mirror of the output of groups G1, G2, and G3 (O_{minus} in Figure 5.9) to the input of group G4 as shown in Figure 5.12. This will ensure that none of the other groups will be active during the application of row reduction rule to group G4.

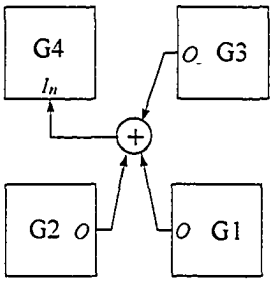


Figure 5.12 Connection between groups of cells.

The circuit schematic for a DBNS adder cell in position (i,j) with the feedback templates representing Eqn. (5.6) and Eqn. (5.10) is shown in Figure 5.13. It is important to stress that the operation of the CNN-DBNS adder is radically different to that of the CNN universal machine (CNN-UM), designed in [41] and implemented in [181], or the work presented in [182]. While the first two use a stored program to control the use of templates and the last one uses temporal superimposition of signals, the CNN-DBNS adder

architecture is self-programmable in the sense that the network switches, as needed, between the two templates described by the row reduction rule of Eqn. (5.4) and the overlaying rule of Eqn. (5.3) based on the state voltages of the involved cells.

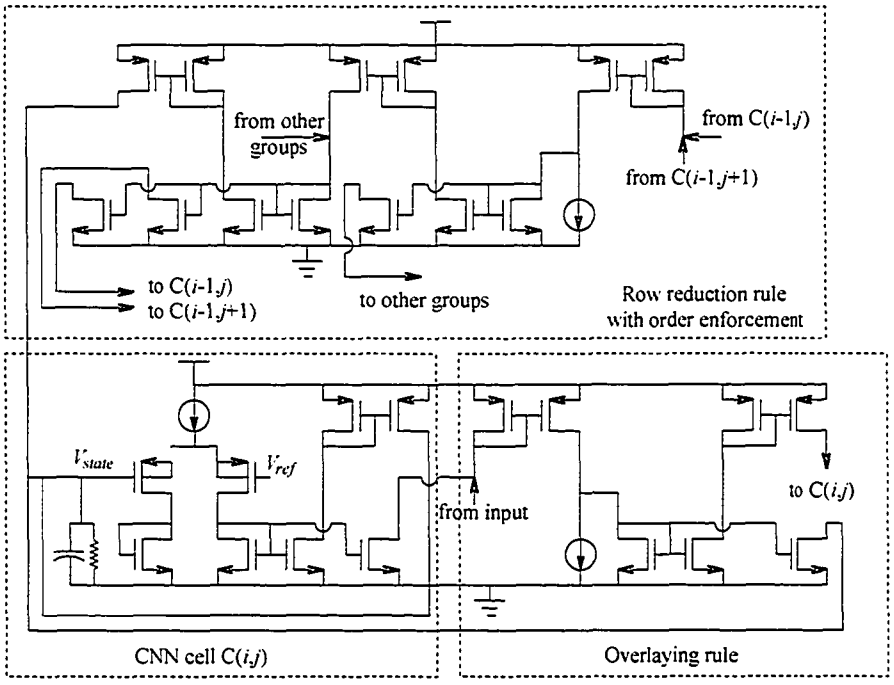


Figure 5.13 CNNDBNS adder cell schematic.

5.3.4 CNNDBNSAU Hspice Simulation

The application of the row reduction rule to the two active cells at positions (i,j) and $(i,j+1)$ will force the cell at position $(i+1,j)$ to change to the active state. However, if the cell at position $(i+1,j)$ was active already, the overlaying rule will take place at the doubled weight cell $(i+1,j)$ and will activate the cell at position $(i+1,j+1)$. Therefore, the neighborhood of the CNN network required is of radius 1. The truth table in Table 5.1 summarizes all possible conditions of the cell $(i+1,j)$ and neighbor cells prior and post the processing of the DBNS map. Notice that from all the 8 possible initial conditions, the final DBNS map will be different to the initial map (CNN cells change states) in only the last two entries of the truth table. These two cases correspond to the application of the overlaying or row reduction rules described above. Hspice simulations of these two cases

are shown in Figure 5.14. Figure 5.14-a shows the Hspice simulation of the DBNS adder cells where only the row reduction rule is applicable while Figure 5.14-b shows the Hspice simulation of the DBNS adder cells when both the overlaying and row reduction rule are applicable.

Table 5.1 Truth table of DBNS adder unit.

z_{ij}	z_{ij-1}	z_{i-1j}	z_{ij}	z_{ij-1}	z_{i-1j}	z_{i-1j-1}
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	0
1	0	0	1	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

The worst case delay for the network to settle after applying the overlaying or row reduction rules is measured as 5.1*ns*. This delay is 60% less than the delay reported in [34] using discrete logic gates to control template operation.

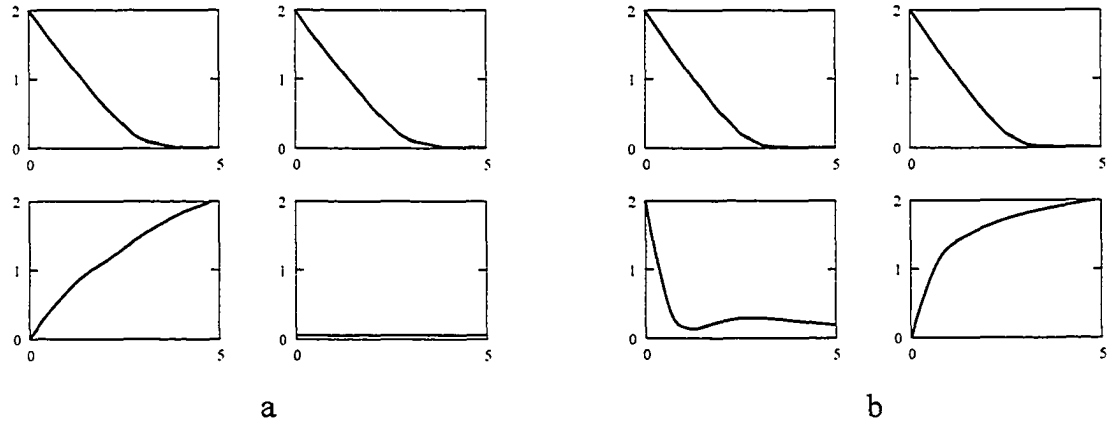


Figure 5.14 Hspice simulation of the CNNDBNSAU: (a) only the row reduction rule is applicable, (b) both the row and overlaying reduction rules are applicable.

5.4 CNNDBNSAU Design Scalability

The CNN implementation of any $M \times N$ DBNS adder consists of a 2-dimensional grid of CNNDBNS adder units with each adder unit corresponding to an entry in the DBNS-map. At the beginning of the addition process, the ARDBNR-map representing X is loaded as the initial conditions on the nodes of the array and the ARDBNR-map representing Y is applied as inputs for a time unit T equal to the CNN cell time constant. The adder units are connected to the input Y using Eqn. (5.3). After a time period T, the input Y has no effect and the adder units use Eqn. (5.3) and Eqn. (5.4) to connect to each other to reduce the sum, Z, to ARDBNR.

5.4.1 A 20x20 CNN-based DBNS Adder

A 20x20 CNN-based DBNS adder is developed using a 20x20 grid of CNNDBNSAU. This DBNS adder has the same dynamic range as a 32-bit binary adder [171]. The CNNDBNSAU are connected to each other using the outputs from the overlaying and row reduction rules presented in Section 5.3.3. The CNN-based DBNS adder has a very regular structure as shown in the schematic of a 4x4 section in Figure 5.15. A number of Hspice simulations, using parameters from 0.35μm CMOS technology process, were performed using random DBNS operands represented in the addition-ready form. To save on space, the simulation of only a small 4x4 section of the DBNS adder is presented here using the two DBNS-maps shown in Figure 5.16. These maps are not in the CDBNR form but they comply with the definition of the ARDBNR. The maps were chosen to illustrate the worst case situation where the sum output has to go through successions of applications of the overlaying and row reduction rules to be represented in an ARDBNR form. The example also illustrates the maximum delay where the row reduction rule has to be applied to each row.

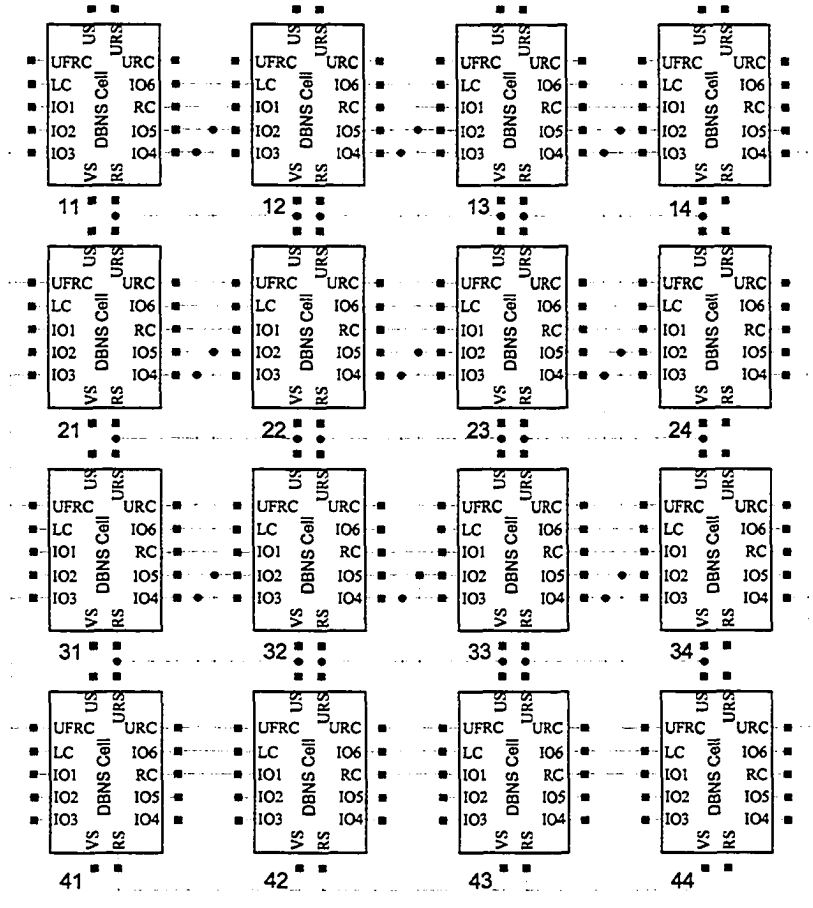


Figure 5.15 Schematic of a 4x4 section of the CNN-based DBNS adder.

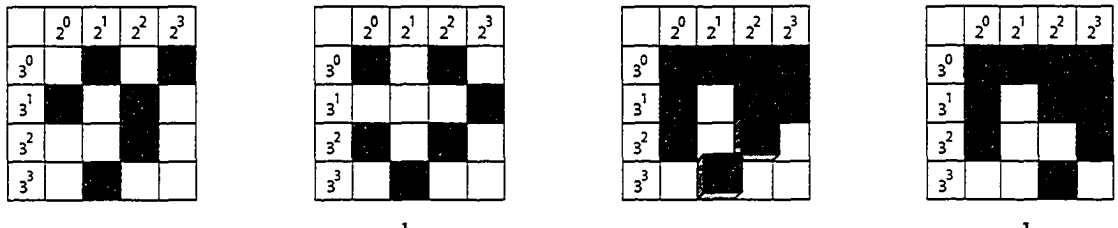


Figure 5.16 An example of addition using the CNN-DBNS adder: (a) *X*, (b) *Y*, (c) Overlaying *X* and *Y*, (d) *Z* after time *T*.

A step by step application of the reduction rules to transform the DBNS-map of the sum output in Figure 5.16-d to addition-ready representation is shown Figure 5.17. In this figure, the grey color represents a cell participating in a reduction rule at that particular instant of time. The output of Hspice simulation of the addition process and the non-zero digit reduction to ARDBNR is shown in Figure 5.18. In the worst case, any two $M \times N$ ARDBNR-maps can be added and the sum converted to ARDBNR-map in a time delay given by:

$$T_{delay} = (M + \lceil N/2 \rceil) \cdot T \tag{5.12}$$

where $\lceil a \rceil$ denotes the smallest integer such that $\lceil a \rceil \geq a$.

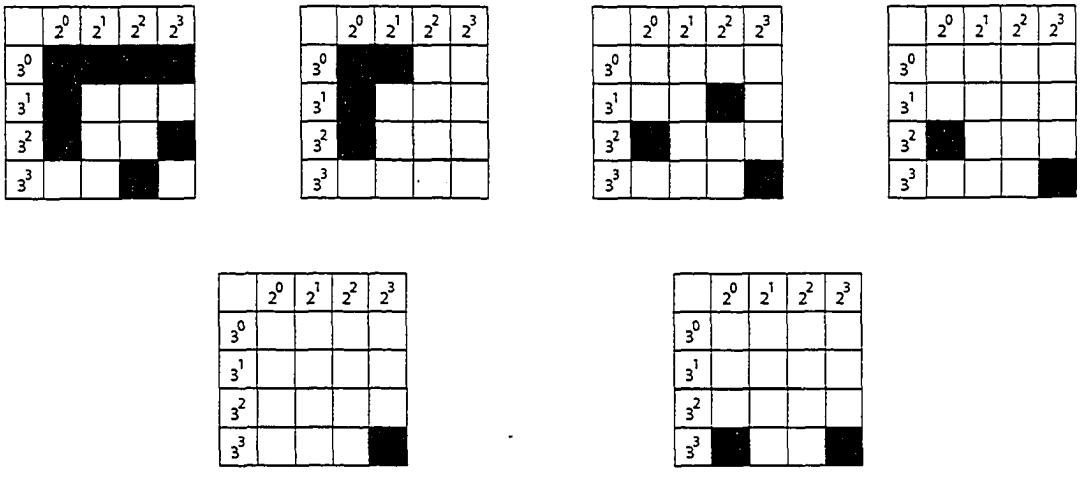


Figure 5.17 Non-zero digit reduction of *Z*. Starting from left to right, each map is obtained from the previous map after a time *T*.

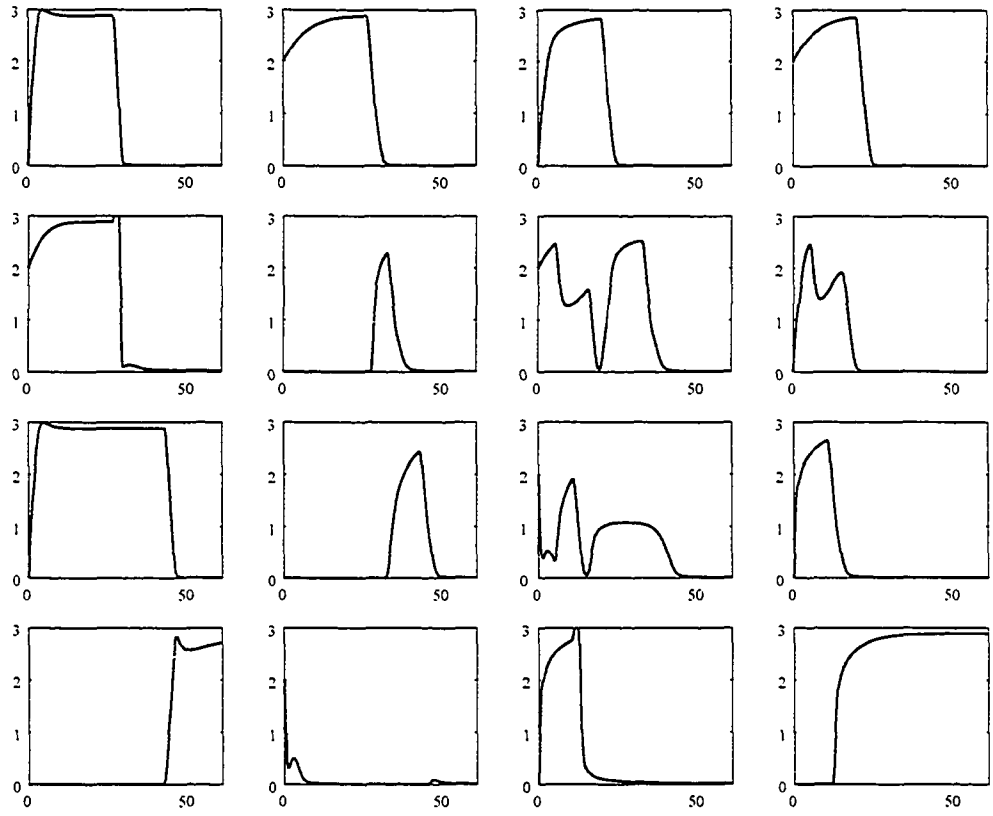


Figure 5.18 Hspice simulation of a 4x4 section of the CNN-based DBNS adder.

5.4.2 Constraints on the CNN-based DBNS Adder to be Self-Programmable

The negative output current of the circuit in Figure 5.9 can be described using the following equation:

$$O_{minus} = m_1(I_n - I_{\sin k}) \tag{5.13}$$

where m_1 is the ratio of the current mirror M3-M5 and I_n is the sum of the output currents from the cells in positions (i,j) and $(i,j+1)$. The positive output current is given by the following equation:

$$O_{plus} = m_2 m_3(I_n - I_{\sin k}) \tag{5.14}$$

where m_2 is the ratio of the current mirror M3-M4 and m_3 is the ratio of the current mirror M6-M7. For simplicity of discussion, we will refer to the multiplying factor (m_1 in Eqn. (5.13) and m_2m_3 in Eqn. (5.14)) as the mirror ratio m.

The choice of the mirror ratio m and the value of the current source I_{sink} plays a crucial role in the performance of the CNN network. An improper choice of either of these parameters might cause instability or force the cell output to settle to an incorrect stable point. There are two critical network transitions: A row reduction rule followed by a row reduction rule and a carry propagation rule followed by a row reduction rule.

An example of a row reduction rule followed by a row reduction rule is depicted in Figure 5.19.

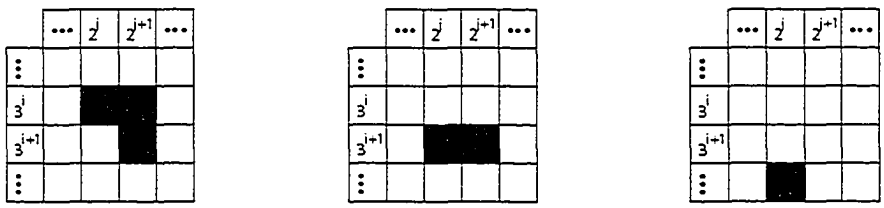


Figure 5.19 An example of a reduction rule followed by a reduction rule: (a) initial map, (b) map after the first reduction rule, (c) map after the second reduction rule.

Examining Figure 5.19-a, one can see that there is only one group of active cells that can participate in the row reduction rule of Eqn. (5.4). The active cells in positions (i,j) and $(i,j+1)$ will be replaced by an active cell in position $(i+1,j)$ as shown in Figure 5.19-b. This means that the cells in positions (i,j) and $(i,j+1)$ will switch from logic 1 to logic 0 while the cell in position $(i+1,j)$ will switch from logic 0 to logic 1. However, once the transition starts and the cell in location $(i+1,j)$ generates an output current, cells at positions $(i+1,j)$ and $(i+1,j+1)$ will constitute another candidate group of active cells for the row reduction rule. Consequently, while the row reduction rule for the active group in position (i,j) tries to force the state voltage of the cell in position $(i+1,j)$ to rise, the row

reduction rule of the active group in position $(i+1,j)$ tries to force the state voltage of the same cell to fall which leads to undetermined final state. The solution to this problem is to ensure that the row reduction rule (for the active group in position $(i+1,j)$ in this case) will not start unless the sum of the output current is more than a certain large threshold. This can easily be implemented by setting the current sink in Figure 5.9 to the threshold value and then using a high mirror ratio to adjust the level of the current flowing into transistor M5 or transistor M7 to the unit current.

The case where the carry propagation rule is followed by the row reduction rule is similar. It is known from the definition of ARDBNR that there will always be a non-active cell to the right of the doubled weight cell as depicted in Figure 5.20. Therefore, the application of the carry propagation rule is straight forward. However, once transition starts and the cell in position (i,j) generates an output current, the row reduction rule of Eqn. (5.4) will be applicable to the cells (i,j) and $(i,j+1)$. This case reduces to the above case where the application of one rule forces the state voltage of the cell to rise while the application of the other rule forces the state voltage of the cell to fall. The solution is already implemented in the solution of the above case; ensuring that the reduction rule will not start unless the sum of the output current is more than a certain large threshold.

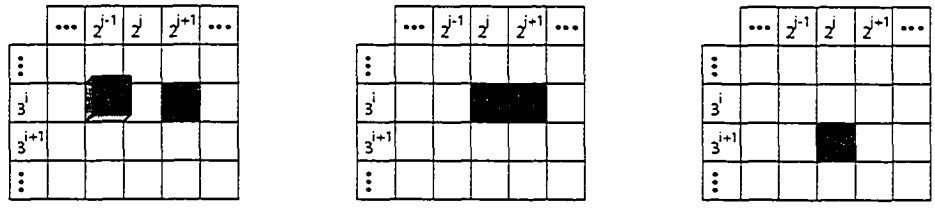


Figure 5.20 An example of a carry propagation rule followed by a reduction rule: (a) initial map, (b) map after applying the overlaying rule, (c) map after applying the row reduction rule.

5.4.3 Impact of the CNN-based DBNS Adder on Substrate Noise

To get an estimate of the amount of switching noise and cross talk reduction, the CNN-based adder is compared to a 32-bit standard digital binary adder since, to the knowledge of the authors, there is no published DBNS design that uses standard logic gates. The 32-bit digital binary adder has the same dynamic range as the 20x20 DBNS adder and is designed using standard library cells in the same 0.35μm CMOS technology. Different random Hspice simulations were performed on the adders and switching noise was recorded. Switching noise for the CNN-based DBNS adder is plotted in Figure 5.21 against switching noise for the digital design. In all Hspice simulations, the CNN-based DBNS adder showed advantages in switching noise with average improvement of 49dB over the digital adder.

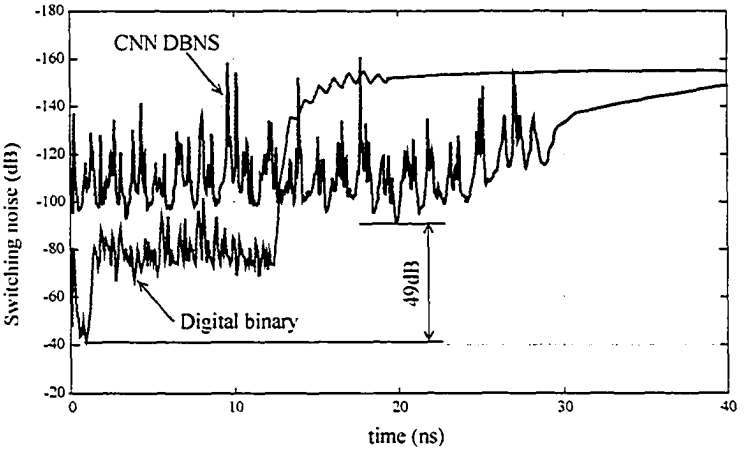


Figure 5.21 Switching noise of the CNN-based 20x20 DBNS adder and 32-bit standard digital binary adder.

Cross talk is also simulated using Hspice and plotted in Figure 5.22. The CNN-based DBNS adder reduced cross talk by more than 20dB compared to that of the digital adder.

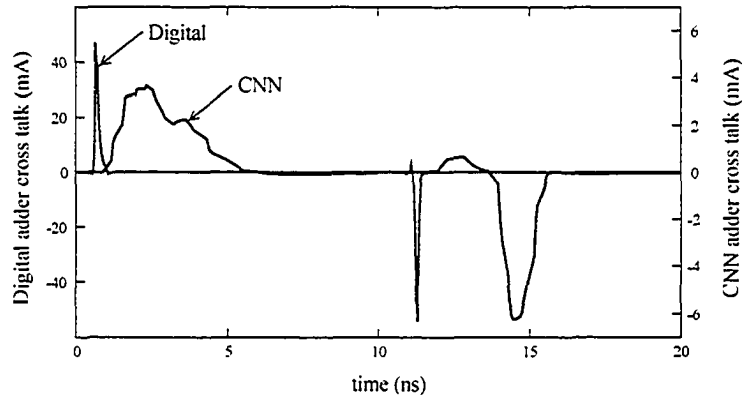


Figure 5.22 Cross talk of the CNN-based 20x20 DBNS adder and 32-bit standard digital binary adder.

5.5 Summary of CNN-based DBNS Arithmetic

This chapter has presented a practical methodology to implementing DBNS arithmetic using analog CNN arrays. An interesting property of the DBNS is that numbers can be represented naturally as 2-D grids. This property facilitates loading DBNS numbers either as 2-D initial conditions or as 2-D inputs into CNN architectures. Therefore, arithmetic operations in the double-base number system become a problem of CNN image morphology. Subsequently, the challenge is to design proper CNN templates that perform the required arithmetic task.

Addition in DBNS is reduced into two consecutive image manipulation steps: Finding the immediate sum of the operands maps and transforming the immediate sum into addition-ready representation for further processing. These two steps are implemented using the overlaying and row reduction rules. First, the row reduction rule is synthesized using a simple current-mode circuit. A key advantage of this circuit is that it can be used to perform the overlaying rule as well. The only difference is where the inputs come from and where the outputs go. A CNN-based DBNS adder unit is then designed by employing the overlaying and row reduction circuits as template connections to neighbor cells. The functionality of the CNNDBNSAU is proved using extensive Hspice simulations.

The scalability of the CNNDBNSAU is illustrated by designing a 20x20 DBNS adder. Special cases of the DBNS-maps are identified and restrictions are defined to control the application of the reduction rules. The restrictions are developed using analog feedback connections between groups of cells. This property renders the network self-programmable in the sense that the adder switches between the overlaying and row reduction rules based on the outputs of the involved cells. Hspice simulations show that the CNN-based DBNS adder achieved 49dB improvement in switching noise and more than 20dB reduction in cross talk over a standard digital adder that possesses the same dynamic range and operates at the same speed.

Chapter 6

Conclusions

6.1 Summary and Contributions

This thesis reports a rather intriguing research project involving the development of analog cellular neural networks for applications in implementing arithmetic with special applications in mixed-signal systems that require the protection of very sensitive analog signals. Noise issues represent a new domain and will become important as integrated circuits, for example, start to include the packaging of bio-sensors in wireless devices for applications in the health sciences. The novelty associated with this work is based on the use of arrays of non-linear analog circuits, to perform digital processing. Current-mode CNN arrays work with the supply current being almost constant providing minimum instantaneous supply current variations. This property drastically reduces switching noise. In addition, CNN arrays provide smooth output transitions with an *RC* mechanism to control the slew rate. This feature allows circuit designers to reduce cross talk at the expense of circuit speed. The major contributions in this thesis focus on the development of practical methodologies to implement arithmetic operations using analog CNN arrays. The research covered arithmetic operations using three different number systems: The binary number system, the binary signed-digit number system, and

the double-base number system. CNN realizations for each number system uncover certain advantages and disadvantages that will be summarized below. Circuit designers then have to weigh the pros and cons and decide which specific number system will best suit their application.

Binary number system arithmetic: To tackle the noise issue, the research commenced in the arithmetic domain focusing on the binary number system. The main challenge was to transform binary arithmetic into 2-D image morphology; to be processed using analog CNN arrays. This included defining new continuous functions representing the sum and carry functions, synthesizing the new functions using simple current-mode circuits such as summing nodes and current mirrors, and designing network neighborhood and template connections between CNN cells. A 1-bit full adder was then designed using four CNN cells and tested to converge for all possible inputs. This property is of paramount importance because it guarantees the stability of larger networks. The 1-bit full adder constitutes a basic building block that was used to develop more complex circuits such as a multi-bit adder and a multiplier. The designed binary circuits have the following advantages:

- Exhibits low switching noise. Hspice simulations show that the circuits improve switching noise by 57dB compared to standard digital designs operating at the same speed.
- Offer low cross talk. Based on Hspice simulations, cross talk diminishes by more than 20dB compared to its digital counterpart.
- Provide standard binary inputs and outputs. This property facilitates using the new circuits as black boxes in more complex structures with no changes to the circuits being designed.

Nevertheless, the developed circuits have some disadvantages that are common to current-mode circuits as listed below:

- Consumes more power. Since these current-mode structures are driven by current, they will always consume power even when the circuits are not processing any data.

- Longer critical paths. This is a feature of current-driven circuits when optimized for low power consumption. The small current requires more time to charge/discharge the load capacitances.

Binary signed-digit number system: The exceptional results, regarding noise suppression, obtained from the CNN-based binary designs drove the extension of the design methodology to a redundant number system which provides benefits of reduced delay and interconnects. The binary signed-digit representation was chosen because it offers superior noise margins compared to higher radix systems. Nonetheless, the main challenge was to discover a means to represent the 3-valued BSD number system naturally in the CNN framework. This led to the design of a new class of CNN characterized by a fundamental 3-state activation function. The addition algorithm of BSD was analyzed and decomposed into several steps. Four 3-state CNN cells, together with new equations that define the BSD addition algorithm, were used to develop a 1-digit BSD full adder. As was the case with the binary circuits, the 1-digit BSD full adder was tested to converge for all possible inputs. This 1-digit BSD full adder forms a key element used in building a multi-digit adder and a multiplier. The multiplier features four-input addition of the partial products in the first level of the binary tree. This property reduces the number of full adders needed and, hence, instantaneous supply current. The developed BSD circuits has several advantages:

- Exhibits very low switching noise. The use of bi-directional current-mode summing nodes reduced switching noise to unprecedented levels. The BSD circuits achieved almost 70dB improvement in switching noise over standard digital circuits. This is an improvement of more than 12dB over the corresponding CNN-based binary circuits.

- Offer reduced cross talk. Cross talk is reduced by more than 23dB of that of the digital circuits. This is an improvement of more than 3dB over the CNN-based binary design.

- Allows constant delay operation regardless of the word length. This is the main advantage for using the binary signed-digit number representation. On the other hand, the delay in the binary system increases linearly with the length of the operands. Consequently, the advantage of using BSD becomes prominent as operands sizes increase.

- Provides standard binary inputs and outputs even though they internally use BSD representation. This property facilitates using the new circuits as embedded components in existing structures without any changes to the circuits.

The developed BSD circuits also have some disadvantages such as:

- Increased power consumption. These structures work with three levels of current compared to two levels in the case of binary circuits. Therefore, to keep the noise margin the same as for the binary circuits, the current provided by current sources has to be doubled.

- Require more transistors and larger silicon area. The BSD algorithm is more complex than the binary algorithm. Realizing the restrictions on the transfer digit requires complex circuits that use a large number of transistors which usually translates into larger silicon area.

Double-base number system: A fascinating feature of the DBNS is that numbers can be represented naturally as 2-D grids. This property facilitates mapping DBNS numbers into CNN architectures. The DBNS is also a relatively new number system and no practical implementations had previously been published when this research work was started. This peaked our interest in developing a practical methodology to convert arithmetic operations in the DBNS into a problem of CNN image morphology. In order to take full advantage of the limited-carry property promised by the DBNS, operands have to be represented in addition-ready maps. These addition-ready representations can be obtained by applying two reduction rules to the original DBNS-maps: the overlaying rule and the row reduction rule. As an initial step toward the goal, an innovative circuit design to implement the row reduction rule was developed. The key feature of this circuit is that it can also be used to implement the overlaying rule using appropriate inputs and outputs. Special cases of DBNR-maps were also identified and a unique feedback connection between groups of cells was defined to deal with them. This method guaranteed correct network operation as well as obtaining a DBNS-map with the smallest possible number of active cells. The use of feedback between groups of cells renders the CNN-based DBNS adder self-programmable where the network decides on which reduction rule to use next based on the outputs of the involved cells. The main advantages of the DBNS circuit can be summarized as follows:

- Exhibits low switching noise. Since the adder incorporates a pure analog control method that governs the operation of the reduction rules, the network embraces the common feature of the binary and BSD designs. A 20x20 DBNS adder achieved 49dB improvement in switching noise over the corresponding traditional digital adder.

- Offers low cross talk. The smooth transitions of the CNN nodes reduce cross talk to more than 20dB compared to that of the fast digital nodes in a digital implementation.

- Promises carry-free addition if the canonic representation is used. Even so, since the canonic representation is difficult to obtain, the addition-ready representation can be used to provide limited-carry addition.

The main disadvantage of the DBNS design when compared to digital designs is power consumption. This seems to be the bottleneck that faces analog designers and little appears to be able to be done to improve it particularly when speed has to be maximized.

A final point to be made for the DBNS representation, is that it appears to be useful for cryptographic applications [174] because of its sparseness. One important property of crypto hardware systems is their resistance to side attacks. These attacks can take several forms in terms of hardware implementations, including measuring circuit power consumption *changes* as certain crypto calculations are being performed. A crypto system based on CNN arrays promises to be more resistance to such attacks than a computational processor based on standard logic implementations, because of the very low noise levels (power consumption changes) inherent in the CNN array operation.

A summary of the design specifications of the novel CNNBFA, CNNBSDFA, and CNNDBNSAU designs is shown in Table 6.1. The table also shows design specifications for the state-of-the-art CNN-based 1-bit binary full adders, 1-digit BSD adders, and CNN-based DBNS structure to perform reduction to addition-ready representation. The new CNNBFA consumes about 43% of the power required by the recursive structure reported in [35] and much less when compared to the power required by the flat structure reported in [34] where the power increases with $O(n^2)$. The speed of the new CNNBFA is about 8 times the speed of the recursive structure and about 19 times the speed of the flat structure. The new CNNBFA also requires less than half the transistors of existing structures and

this maps into smaller silicon area. In addition, the new CNNBFA is scalable and compatible with the well-known circuit designs.

The delay of the new CNNBSDFA is comparable to the *5ns* delay of the voltage-mode signed-digit adder circuit reported in 0.18μm CMOS technology [159]. In addition, the CNNBSDFA uses 27% fewer transistors than the voltage-mode design. When compared to the recently introduced negative-differential-resistance (NDR) signed digit adder [161][162], the CNNBSDFA is 8 times faster and consumes 82% less power. At first glance, the BSD adder appears to be slower than the binary adder. However, an operand size of just two digits is required to produce similar worst-case propagation delay values for a ripple-carry adder built with binary full adder cells. This indicates that the BSD can provide significant speedup of addition for multi-digit adders since addition of operands of any length can be accomplished in the time required for a 2-bit binary addition.

The new CNNDBNSAU design performs both addition and reduction to AR representation while the existing structure performs reduction only [34]. The new CNNDBNSAU design is more than two times faster and uses fewer transistors which means smaller silicon area and less power consumption. In addition, reduction rules equations and operations are synthesized using simple current mirrors without hysteresis or digital logic which leads to improved switching noise.

Table 6.1 Summary of design specifications.

NS	Design	Transistor count	Active Area (μm^2)	Delay (ns)	Power (μW)
Binary	Flat [34]	≈108x(1+N/4)	—	≈60	279.5x(1+N/4)
	Recursive [35]	≈101+	—	≈25	≈392
	New CNNBFA	43	21.83	3.22	169.98
BSD	Digital [159]	140	—	5	—
	NDR [161]	—	4789	17	2300
	New CNNBSD	102	72.44	6.12	412.51
DBNS	DBNS [34]	≈78+	—	13	—
	New CNNDBNS	56	28.35	5.1	197.11

The increased power and silicon area of the CNNBSDFA design, compared to the CNNBFA and CNNDBNSAU, can be justified by its ultra-low switching noise performance for very sensitive applications where reducing switching noise is of crucial importance. The CNN-based BSD design reduces switching noise to unprecedented levels as can be seen in Figure 6.1 and Figure 6.2. Figure 6.1 shows switching noise comparison for different adder designs as a function of adder size while Figure 6.2 shows switching noise comparison for different multiplier designs as a function of multiplier size. It is clear that the BSD design offers the lowest switching noise for all operand sizes in both cases.

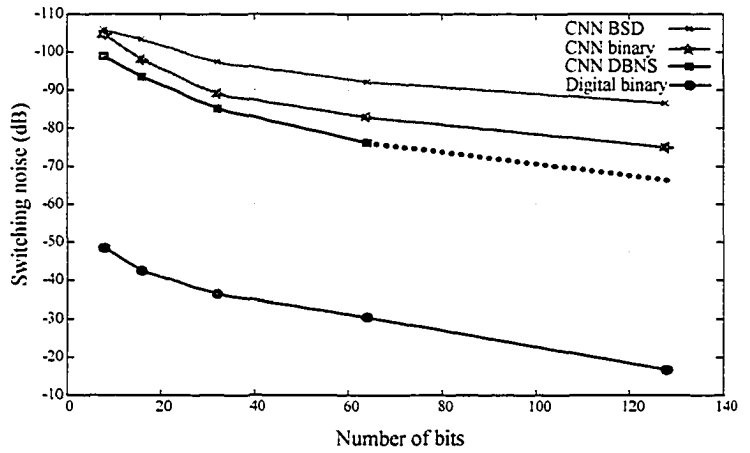


Figure 6.1 Switching noise of different adders vs. adder size.

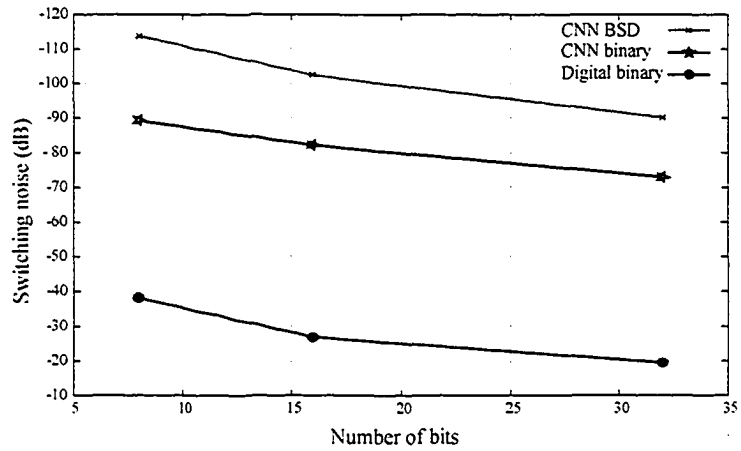


Figure 6.2 Switching noise of different multipliers vs. multiplier size.

The CNN-based BSD design also offers reduced cross talk compared to all other designs as can be seen in Figure 6.3.

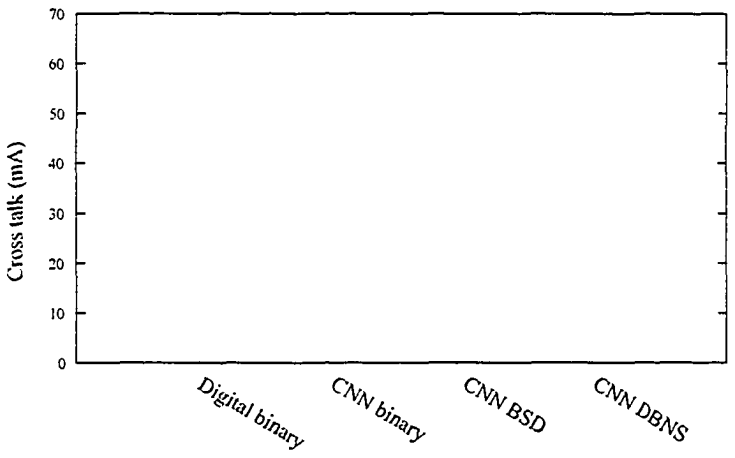


Figure 6.3 Cross talk of different adders designs.

6.2 Conclusions

In this research work, a practical methodology to develop ultra-low noise arithmetic circuits using analog cellular neural networks has been presented. The technique has been used to design arithmetic circuits for three different number systems: The binary number system, the binary signed-digit number system, and the double-base number system. First, for each number system, the addition algorithm has been re-defined using continuous functions that can be realized as CNN templates. Next, the templates have been utilized to develop a 1-bit adder unit that converges for all possible inputs. This property guarantees the stability of larger networks that use the adder unit as an embedded component. For the BSD number system, a new class of CNN featuring a 3-state activation function has been developed. This enables mapping the 3-valued BSD number system naturally into the 3-states of the new CNN cell. The DBNS adder unit uses a novel synthesis of the reduction rules to perform addition as well as reduction to addition-ready representation for further processing. Using the different adder units as enabling building blocks, complex circuit structures such as multi-bit adders and large multipliers have been developed. The CNN-

based BSD arithmetic circuits are the first to be published in the literature. The DBNS design employs a novel self-programmable structure that switches between different templates based on the output voltages of the involved cells. This renders the DBNS adder circuit the first complete CNN-based adder circuit in the literature. The BSD and DBNS designs also incorporate the traditional advantages of such number systems including reduced delay and interconnects for large operands. Moreover, all the CNN structures developed also exhibit the advantage of the ultra-low noise property. Hspice simulations, using parameters from 0.35μm CMOS technology, show that switching noise and cross talk have been reduced by up to 70dB and more than 23dB, respectively, when compared to traditional digital circuits operating at the same speed.

6.3 Suggestions for Future Work

The following are some directions for future research:

1. When we compare the noise performance of the designed circuits and the digital implementations, we assume that all circuits have equivalent parasitic inductance and capacitance. This is not always true because the parasitic elements depend greatly on the specific layout of the circuit. A more accurate comparison can be performed by fabricating the noise source (the design being tested) and a noise sensing element (e.g., a sensitive amplifier) on the same chip a certain distance apart. Then random inputs can be applied to the noise source and the effect on the noise sensing element can be measured using accurate testing equipment.
2. In Chapter 5, we discussed addition in the double-base number system using the CNN paradigm. We also developed an adder that performs addition as well as reduction to the addition-ready representation. However, we assumed that the inputs are in the addition-ready representation. For a practical implementation that is compatible with the digital convention, the adder should accept binary inputs and produce binary outputs. Therefore, the implementation of a binary to double-base and a double-base to binary converters needs to be addressed.

3. We did not discuss the implementation of multiplication in Chapter 5. This is because, the traditional method of implementing a multiplier (using shift and add) seems inefficient when considering the required silicon area. A thorough study needs to be done in this area.

4. All the measurements presented in this thesis are based on Hspice simulation at room temperature. For precise performance evaluation, second-order effects (such as thermal noise and device mismatch) need to be taken into account.

REFERENCES

[1] "Substrate noise analysis of mixed-signal ICs," Cadence Design Systems, 2001.

[2] P. Larsson, "Power supply noise in future IC's: A crystal ball reading," in Proc. IEEE Custom Integrated Circuits Conf., pp. 467-474, 1999.

[3] P. E. Gronowski et al., "High-performance microprocessor design," IEEE J. Solid-State Circuits, vol. SC-33, no. 5, pp. 676-686, May 1998.

[4] International Technology Roadmap for Semiconductors. http://www.itrs.net/Common/2004Update/2004Update.htm

[5] M. Nagata and A. Iwata, "Substrate crosstalk analysis in mixed signal CMOS integrated circuits," Proc. Asia and South Pacific - Design Automation Conf., ASP-DAC 2000, pp. 623-629, 2000.

[6] S. R. Vemuru, "Effects of simultaneous switching noise on the tapered buffer design," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 5, no. 3, pp. 290-300, September 1997.

[7] J. J. Becerra and E. G. Friedman, Analog Design Issues in Digital VLSI Circuits and Systems. Norwell, Massachusetts, Kluwer Academic Publishers, 1997.

[8] E. G. Friedman, High Performance Clock Distribution Networks, Kluwer Academic Publishers, 1997.

[9] X. Aragonès, J. L. Gonzalez and A. Rubio, Analysis and Solutions for Switching Noise Coupling in Mixed-Signal ICs. Kluwer Academic Publishers, 1999.

[10] Han-Su Kim, Jenkins, K.A., and Ya-Hong Xie, "Effective crosstalk isolation through p+ Si substrates with semi-insulating porous Si," IEEE Electron Device Letters, vo. 23, no. 3, pp.160-162, 2002.

[11] D. K. Su, M. J. Loinaz, S. Masui, and B. A. Wooley, "Experimental results and modelling techniques for substrate noise in mixed-signal integrated circuits," IEEE J. Solid State Circuits, 28, 420, 1993.

[12] K. Joardar, "Substrate crosstalk in BiCMOS mixed mode integrated circuits," Solid-State Electron, vol. 39, pp. 511–516, April 1996.

[13] J. H. Wu, J. Scholvin, J. A. del Alamo, and K. A. Jenkins, "A Faraday cage isolation structure for substrate crosstalk suppression," IEEE Trans. Microwave Wireless Compon. Lett. vol. 11, pp. 410-412, 2001.

[14] Liao, C.P., Juang, K.C., Huang, T.H., Duh, D.S., Yang, T.T., and Liu, M.N. "A new isolation technology for mixed-mode and general mixed-technology SOC chips," Semiconductor Manufacturing Technology Workshop, pp. 124-132, 2000.

[15] J. P. Raskin, A. Viviani, D. Flandre, and J. P. Colinge, "Substrate cross-talk reduction using SOI technology," IEEE Trans. Electron Devices, vol. 44, pp. 2252-2261, 1997.

[16] M. Kumar, Y. Tan, and J. K. O. Sin, "A simple, high performance complementary TFSOI BiCMOS technology with excellent cross-talk isolation and high-Q inductors for low power wireless applications," in Proc. IEEE Int. SOI Conf., Wakefield, MA, pp. 142-143, Oct., 2000.

[17] Kumar, M., Tan, Y., and Sin, J.K.O., "Excellent cross-talk isolation, high-Q inductors, and reduced self-heating in a TFSOI technology for system-on-a-chip applications," IEEE Trans. Electron Devices, vol. 49, no. 4, pp. 584-589, 2002.

[18] "High integration brings noise to a new level," Technology Update, Wireless System Design, november 1999.

[19] Y. Tsividis, Mixed Analog-Digital VLSI Design and Technology, McGraw-Hill, 1995.

[20] A. Cathelin, D. Saias, D. Belot, Y. Leclercq, and F. J. R. Clement, "Substrate parasitic extraction for RF integrated circuits," Design Automation & Test in Europe, poster 4C-2, March 2002.

[21] R.M. Secareanu, S. Warner, S. Seabridge, C. Burke, T.E. Watrobski, C. Morton, W. Staub, T. Tellier, and E.G. Friedman, "Placement of substrate contacts to alleviate substrate noise in epi and non-epi technologies," Proc. of the 43rd IEEE Midwest Symposium on Circuits and Systems, vol. 3 , pp. 1314-1318, 2000.

[22] L.-R. Zheng and H. Tenhunen, "Effective power and ground distribution scheme for deep submicron high speed VLSI circuits," Proc. of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS '99, vol. 1, pp. 537-540, 1999.

[23] T. Blalack, "Design techniques to reduce substrate noise," Analog Circuit Design: Volt Electronics; Mixed-Mode Systems; Low-Noise and RF Power Amplifiers for Telecommunication, pp. 193-217, J. Huijsing-Editor, Kluwer, 1999.

[24] R.M. Secareanu, S. Warner, S. Seabridge, C. Burke, T.E. Watrobski, C. Morton, W. Staub, T. Teilier, and E.G. Friendman, "Physical design to improve the noise immunity of digital circuits in a mixed-signal smart-power system," Proc. of the 2000 IEEE International Symposium on Circuits and Systems, ISCAS 2000, vo. 4, pp. 277-280, 2000.

[25] T. Blalack, Y. Leclercq, and C.P. Yue, "On-chip RF isolation techniques," Proc. of the Bipolar/BiCMOS Circuits and Technology Meeting, pp. 205-211, 2002.

[26] K.T. Tang and E.G Friedman, "On-chip delta-I noise in the power distribution networks of high speed CMOS integrated circuits," Proc. 13th Annual IEEE International ASIC/SOC Conf., pp. 53-57, 2000.

[27] L. Connell, N. Hollenbeck, M. Bushman, D. McCarthy, S. Bergstedt, R. Cieslak, J. Caldwell, "A CMOS broadband tuner IC," IEEE International Solid-State Circuits Conf., vol. 45, pp. 400-401, February 2002.

[28] L. Forbes, B. Ficq and S. Savage, "Resonant forward-biased guard-ring diodes for suppression of substrate noise in mixed-mode CMOS circuits," Electronics Letters, vol. 31, no. 9, pp. 720-721, April 1995.

[29] T. Liu, J.D. Carothers, and W.T. Holman, "A negative feedback based substrate coupling noise reduction method," Proc. Twelfth Annual IEEE International ASIC/SOC Conf., pp. 49-53, 1999.

[30] S. Sakiyama, J. Kajiwara, M. Kinoshita, K. Satomi, K. Ohtani, and A. Matsuzawa, "An on-chip high-efficiency and low-noise DC/DC converter using divided switches with current control technique," IEEE International Solid-State Circuits Conf., ISSCC '99, Digest of Technical Papers, pp. 156-157, 1999.

[31] H-T. Ng and D.J. Allstot, "CMOS current steering logic for low-voltage mixed-signal integrated circuit," IEEE Trans. VLSI Systems, 5, pp. 301-308, Sept. 1997.

[32] D.J. Allstot, S-H. Chee and M. Shrivastawa, "Folded source-coupled logic vs. CMOS static logic for low-noise mixed-signal ICs," IEEE Trans. Circuits and Systems I, 40, pp. 553- 563, Sept. 1993.

[33] E. Albuquerque, J. Fernandes and M. Silva, "NMOS current-balanced logic," Electronics Letters, 32, pp. 997-998, May 1996.

[34] S. Sadeghi-Emamchaie, Novel Cellular Neural Networks for Low-noise Digital Arithmetic Architectures, Ph.D. Thesis, University of Windsor, 1999.

[35] S. Sadeghi-Emamchaie, G. A. Jullien, and W. C. Miller, "Very low-noise (switching free) CNN-based adder," SPIE Proc. Advanced Signal Processing Algorithms, Architectures, and Implementations IX, 3807, (7), 1999.

[36] A.J. Acosta, P. Parra, J. Juan, M. Valencia, and R. Jiménez, "Reduction of switching noise in low-power CMOS digital circuits by gate-level optimization," International Workshop on Logic and Synthesis, pp.101-106. June 2001.

[37] A.J. Acosta, R. Jiménez, J., M. J. Juan, Bellido, and M. Valencia, "Influence of clocking strategies on the design of low switching-noise digital and mixed-signal VLSI circuits," in 10th PATMOS, pp. 316-326, Sept. 2000.

[38] L. O. Chua and L. Yang, "Cellular neural networks: Theory," IEEE Trans. Circuits & Systems, vol. CAS-35, pp. 1257-1272, Oct. 1988.

[39] L. O. Chua and L. Yang, "Cellular neural networks: Applications," IEEE Trans. Circuits & Systems, vol.CAS-35, pp. 1273-1289, Oct. 1988.

[40] L. O. Chua and T. Roska, "The CNN paradigm," IEEE Trans. Circuits & Systems, vol. CAS-40, pp.147-155, March 1993.

[41] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," IEEE Trans. Circuits & Systems, vol. CAS-40, pp. 163-172, March 1993.

[42] T. Roska and A.Rodríguez-Vázquez (Editors), Towards the Visual Microprocessor, John Wiley & Sons Ltd., 2000.

[43] L.O. Chua and T. Roska, Cellular Neural Networks and Visual Computing: Foundations and Applications, Cambridge University Press, 2002, ISBN 0 521 65247 2.

[44] R. Kunz, R. Tetzlaff and D. Wolf SCNN, "A universal simulator for cellular neural networks," Proc. IEEE CNNA 96, Sevilla, pp. 255-260, 1996.

[45] L. Bertucco and N& G., "A multi-layer cellular neural network simulator for image processing applications", Proc. the Third Int. ICSC Symposia on Intelligent Industrial Automation IN99 and Soft Computing SOC0'99, pp. 147-151, 1999.

[46] G. De Sandre and A. Premoli, "Piecewise-exponential approximation for fast time-domain simulation of 2-D cellular neural networks," IEEE Trans. Circuits and Systems II: Express Briefs, vol. 51, no. 8 , pp. 400-405, 2004.

[47] T. Kacprzak and K. Slot, "Multiple-input OTA based circuit for cellular neural network implementation in VLSI CMOS technology," Proc. IEEE CNNA-92, pp. 157-162, 1992.

[48] S. Espejo, A. Rodriguez-Vazquez, R. Dominguez-Castro, and J. L. Huertas, "Switched-current techniques for image processing cellular neural networks in MOS VLSI," Proc. IEEE Int. Symp. Circuits & Systems, pp. 1537-1540, 1992.

[49] M. Sindhwani, T. Srikanthan, and K.V. Asari, "VLSI efficient discrete-time cellular neural network processor," IEE Proc. Circuits, Devices and Systems, vol. 149, no. 3, pp. 167-171, 2002.

[50] A. Rodriguez-Vazquez, S. Espejo, R. Dominguez–Castro, J. L. Huertas, and E. Sanchez-Sinencio, "Current-mode techniques for the implementation of continuous and discrete-time cellular neural networks," IEEE Trans. Circuits and Systems II, vol. 40, no. 3, pp. 132–146, 1993.

[51] J. E. Varrientos, E. Sanchez-Sinencio, and J. Ramirez-Angulo, "A current-mode cellular neural network implementation," IEEE Trans. Circuits & Systems, Pt. II, vol. CAS-40, pp. 147-155, March 1993.

[52] M. Gilli, F. Corinto, and P.P. Civalleri, "Design and synthesis methods for cellular neural networks," Proc. Int. Joint Conf. on Neural Networks, vol. 2, pp. 1486-1491, 2003.

[53] P. Kinget and M. Steyaert, "A programmable analog cellular neural network CMOS chip for high speed image processing," IEEE Journal of Solid-State Circuits, vol. 30, no. 3, pp. 235-243, March 1995.

[54] Wen-Cheng Yen, Rong-Jian Chen, and Jui-Lin Lai "Design of MIN/MAX cellular neural networks (MMCNNS) in CMOS technology," Proc. of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, CNNA '02, pp. 339-346, 2002.

[55] T. Matsumoto, L. O. Chua, and H. Suzuki, "CNN cloning template: Connected component detector," IEEE Trans. Circuits and Systems, vol. 37, no. 5, pp. 633-635, May 1990.

[56] T. Matsumoto, L. O. Chua, and R. Furukawa, "CNN cloning template: Hole-filler," IEEE Trans. Circuits and Systems, vol. 37, no. 5, pp. 635-638, May 1990.

[57] H. Aomori, T. Otake, N. Takahashi, and M. Tanaka, "Lossless image coding based on lifting wavelet using discrete-time cellular neural network with multi-templates," Proc. the 2004 Int. Symposium on Circuits and Systems, ISCAS '04, vol. 3, pp. III-101-4, 2004.

[58] D. Feiden and R. Tetzlaff, "Binary image coding using cellular neural networks," Proc. the Int. Joint Conf. on Neural Networks, vol. 2 , pp. 1149- 1152, 2003.

[59] M. Tanaka, Y. Tanji, M. Onishi, and N. Takahashi "Lossless image compression and reconstruction by cellular neural networks," Proc. the 2000 6th IEEE Int.Workshop on Cellular Neural Networks and Their Applications, CNNA '00, pp.57-62, 2000.

[60] Chang Wen Chen, Lulin Chen, and Jiebo Luo, "A cellular neural network for clustering-based adaptive quantization in sub-band video compression," IEEE Trans. Circuits and Systems for Video Technology, vol. 6 , no. 6, pp. 688-692, 1996.

[61] G. Costantini, D. Casali, and R. Perfetti, "Cellular neural network template for rotation of grey-scale images," Electronics Letters, vol. 39, no. 25, pp. 1803-1805, 2003.

[62] Q. Gao, P. Messmer, and G.S Moschytz, "Binary image rotation using cellular neural networks," IEEE Int. Symp. on Circuits and Systems, ISCAS '02, vol. 3, pp. III-113–III-116, 2002.

[63] J. Kowalski, "0.8 μm CMOS implementation of weighted-order statistic image filter based on cellular neural network architecture," IEEE Trans. Neural Networks, vol. 14, no 5, pp. 1366-1374, 2003.

[64] V.V. Khryasshyov, A.L. Priorov, E.Yu. Sautov, and E.A. Sokolenko, "Digital image filtration on cellular neural network," IEEE Int. Conf. on Artificial Intelligence Systems, ICAIS '02, pp 248-251, 2002.

[65] J. Kowalski and T. Kacprzak, "Cellular neural network based weighted median filter for real time image processing," Proc. Int. Conf. on Image Processing, vol. 1, pp. 545-548, 2001.

[66] M.A.J. Moran and J.A.F. Munoz, "Applications of cellular neural networks (CNN) to grey scale image filtering," Ninth Int. Conf. on Artificial Neural Networks, ICANN '99, vol. 1, pp. 449-454, 1999.

[67] P. Ecimovic and J. Wu, "Delay-driven contrast enhancement using a cellular neural network with state-dependent delay," Proc. 7th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '02, pp. 202-208, 2002.

[68] T. Hammadou and A. Bouzerdoum, "Novel image enhancement technique using shunting inhibitory cellular neural networks," IEEE Trans. Consumer Electronics, vol. 47, no. 4, pp. 934-940, 2001.

[69] M. Brendel and T. Roska, "Adaptive image sensing and enhancement using the adaptive cellular neural network Universal Machine," Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '00, pp. 93-98, 2000.

[70] S. Itakura, Y. Tanji, T. Otake, and M. Tanaka, "Progressive image reconstruction via cellular neural networks," IEEE Int. Symp. on Circuits and Systems, ISCAS '02, vol. 1, pp. I-233 - I-236 , 2002.

[71] M.E. Celebi and C. Guzelis, "Image restoration using cellular neural network," Electronics Letters, vol. 33, no. 1, pp. 43-45, 1997.

[72] P.A. Stubberud and A.R. Stubberud, "Text image restoration using cellular neural networks," Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '97, vol. 1, pp. 749-752, 1997.

[73] P.R. Giaccone, D. Tsaptsinos, and G.A. Jones, "Foreground-background segmentation by cellular neural networks," Proc. 15th Int. Conf. on Pattern Recognition, vol. 2 , pp. 438-441, 2000.

[74] D.L. Vilarino, D. Cabello, M. Balsi, and V.M. Brea, "Image segmentation based on active contours using discrete time cellular neural networks," Proc. Fifth IEEE Int. Workshop on Cellular Neural Networks and Their Applications, pp. 331-336, 1998.

[75] T. Sziranyi and J. Zerubia, "Markov random field image segmentation using cellular neural network," IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications, vol. 44, no. 1, pp. 86-89, 1997.

[76] Chin-Teng Lin, Shi-An Chen, Chao-Hui Huang, and Jen-Feng Chung, "Cellular neural networks and PCA neural networks based rotation/scale invariant texture classification," Proc. IEEE Int. Joint Conf. on Neural Networks, vol. 1, pp. 153-158, 2004.

[77] R. Schonmeyer, D.F. Feiden, and R. Tetzlaff, "On-chip template training for pattern matching by cellular neural network universal machines (CNN-UM)," Proc. Int. Symp. on Circuits and Systems, ISCAS '03, vol. 3, pp. III-514 - III-517, 2003.

[78] G. Grassi and E. Di Sciascio, "A new learning algorithm for pattern classification using cellular neural networks," IEEE Int. Symp. on Circuits and Systems, ISCAS '01, vol. 3, pp. 652-655, 2001.

[79] N. Stanic, M. Potrebic, D. Durdevic, D. Dujkovic, and P. Kostic, "Character recognition using a cellular neural network," 2002 6th Seminar on Neural Network Applications in Electrical Engineering, NEUREL '02, pp. 135-138, 2002.

[80] V. Tavsanoglu and E. Saatci, "Feature extraction for character recognition using Gabor-type filters implemented by cellular neural networks," Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '00, pp. 63-68, 2000.

[81] R.H. Tsai, B.J. Sheu, M.Y. Wang, and S.H. Jen, "Two-dimensional cellular neural networks for pre-processing in face recognition and digital library search," Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '97, vol. 1, pp. 733-736, 1997.

[82] D. Feiden and R. Tetzlaff, "Obstacle detection in planar worlds using cellular neural networks," Proc. 7th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '02, pp. 383-390, 2002.

[83] A. Zanela and S. Taraglio, "A cellular neural network stereo vision system for autonomous robot navigation," Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '00, pp. 117-122, 2000.

[84] M. Kanaya and M. Tanaka, "Robot multi-driving controls by cellular neural networks," Proc. Third IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA '94, pp. 481-486, 1994.

[85] G. Grassi and L.A. Grieco, "Object-oriented image analysis via analog CNN algorithms – part I: Motion estimation," Proc. 7th IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA '02, 2002.

[86] M. Balsi, "Regularization-based continuous-time motion detection by single-layer cellular neural networks," Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA '00, pp. 135-140, 2000.

[87] M.G. Milanova, A.C. Campilho, and M.V. Correia, "Cellular neural networks for motion estimation," Proc. 15th Int. Conf. on Pattern Recognition, vol. 3, pp. 819-822, 2000.

[88] K. Kondo, H. Morishita, Y. Konishi, and H. Ishigaki, "Design of two-stage cellular neural network filter for detecting particular moving objects," Proc. Fifth Int. Symp. on Signal Processing and Its Applications, ISSPA '99, vol. 2, pp. 665-668 , 1999.

[89] V. Preciado, D. Guinea, R. Montufar, and J. Vicente, "Real-time inspection of metal laminates by means of CNNs," Proc. SPIE, vol. 4301, no. 39, pp. 260-270, 2001.

[90] D. Guinea, A. Gordaliza, J. Vicente, and M. C. Garca-Alegre, "CNN based visual processing for industrial inspection," Proc. SPIE, vol. 3966, no. 45, pp. 315-322, 2000.

[91] C.L. Chang and C.T. Lin, "CNN-based defect inspection in images with regular pattern," Proc. 16th Eur. Conf. Circuit Theory and Design, ECCTD'03, pp. I221-I224, 2003.

[92] R. Perfetti and L. Terzoli, "Analogic CNN algorithms for textile applications," Int. J. Circuit Theory Applications, no. 28, pp. 77-85, 2000.

[93] R. Fantacci, R. Gubellini, D. Tarchi, and T. Pecorella, "DiffServ on-board satellite switching based on cellular neural networks," IEEE Int. Conf. on Communications, vol. 7, pp. 3953-3957, 2004.

[94] R. Fantacci, M. Forti, M. Marini, and L. Pancani, "Cellular neural network approach to a class of communication problems," IEEE Trans. Circuits Syst. I, vol. 46, pp. 1457-1467, 1999.

[95] A. Fancsali and J. Levendovszky, "CNN based real-time call admission control in packet-switched networks," in Proc. Polish-Czech-Hungarian Workshop on Circuit Theory, Signal Processing, and Telecommunication Networks, Budapest, Hungary, 2001.

[96] J. Levendovszky and A. Fancsali, "Real-time call admission control for packet-switched networking by cellular neural networks," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 51, no. 6, pp. 1172-1183, 2004.

[97] Zhang Yifeng and He Zhengya, "A secure communication scheme based on cellular neural network," IEEE Int. Conf. on Intelligent Processing Systems, ICIPS '97, vol. 1, pp. 521-524, 1997.

[98] F. Gollas, C. Niederhofer, and R. Tetzlaff, "Prediction of brain electrical activity in epilepsy using a higher-dimensional prediction algorithm for discrete time cellular neural networks (DTCNN)," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol. 5, pp. V-720 - V-723 , May 2004.

[99] M. Laiho, A. Paasio, A. Kananen, and K. Halonen, "A mixed mode polynomial-type CNN for analyzing brain electrical activity in epilepsy," in International Journal of Circuit Theory and Applications, pp. 165-180, 2002.

[100] R. Tetzlaff, C. Niederhofer, and P. Fischer, "Feature extraction in epilepsy using a cellular neural network based device - first results," Proc. Int. Symp. on Circuits and Systems, ISCAS '03, vol. 3, pp.III-850 - III-853, 2003.

[101] J. Vandewalle, B. Preneel, and M. Csapodi, "Data security issues, cryptographic protection methods, and the use of cellular neural networks and cellular automata," Proc. Fifth IEEE Int. Workshop on Cellular Neural Networks and Their Applications, pp. 39-44, 1998.

[102] F. Werblin, A. Jacobes, and J. Teeters, "The computational eye," IEEE Spectrum, pp. 30-37, May 1996.

[103] F.S. Werblin and A. Jacobs, "The cellular neural network as a retinal camera for visual prosthesis," Int. Conf. on Neural Networks, vol. 4 , pp. 2327-2332, 1997.

[104] I. Krstic, B. Reljin, and P. Kostic, "Cellular neural network to model and solve direct non-linear problems of steady-state heat transfer," Int. Conf. on Trends in Communications, EUROCON '01, Vol. 2, pp. 420-423, 2001.

[105] T. Nakaguchi, K. Omiya, and M. Tanaka, "Hysteresis cellular neural networks for solving combinatorial optimization problems," Proc. 7th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '02, pp. 539-546, 2002.

[106] N.K. Al-Ani and T. Kacprzak, "Application of time-varying cellular neural network for optimal solutions," Proc. 6th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, CNNA '00, pp. 235-240, 2000.

[107] A. Rasmussen and M.E. Zaghloul, "CMOS analog implementation of cellular neural network to solve partial differential equations with a microelectromechanical thermal interface," Proc. 40th Midwest Symp. on Circuits and Systems, vol. 2 , pp. 1326-1329, 1997.

[108] M. Leong, P. Vasconcelos, J.R. Fernandes, and L. Sousa, "A programmable cellular neural network circuit," 17th Symp. on Integrated Circuits and Systems Design SBCCI 2004, pp. 186-191, 2004.

[109] B.J. Sheu, S.H. Bang, and W.C. Fang, "Optimal solutions of selected cellular neural network applications by the hardware annealing method," Proc. IEEE CNNA-94, pp. 279-284, 1994.

[110] B.J. Sheu, S.H. Bang, and W.C. Fang, "Analog VLSI design of cellular neural networks with annealing ability," Proc. IEEE CNNA-94, pp. 387-391, 1994.

[111] C.C. Lee and J. Pineda de Gyvez, "Color image processing in a cellular neural network environment," IEEE Trans. Neural Networks, vol. 7, no. 5, pp. 1086-1098, Sept. 1996.

[112] L. Wang, J. Gyvez, and E. Sanchez-Sinencio, "Time multiplexed color image processing based on a CNN with cell-state outputs," IEEE Trans. VLSI Systems, vol. 6, no. 2, pp. 314-322, June 1998.

[113] H.N. Cheung, A. Bouzerdoum, and W. Newland, "Properties of shunting inhibitory cellular neural networks for color image enhancement," Proc. 6th Int. Conf. on Neural Information Processing, ICONIP '99, vol. 3 , pp. 1219-1223, 1999.

[114] L.O. Chua and T. Roska, "The CNN Universal Machine part 1: The architecture," Int. Workshop on Cellular Neural Networks and their Applications (CNNA), pp. 1-10, 1992.

[115] M. Gilli, T. Roska, L.O. Chua and P.P. Civalleri, "CNN dynamics represents a broader class than PDEs," Int. Journal of Bifurcation and Chaos, vol. 12, pp. 2051-2068, 2002.

[116] M. Gilli, P. Checco, and F. Corinto,"A spectral technique for the analysis of nonlinear dynamic arrays," IEEE Eleventh Int. Workshop on Nonlinear Dynamics of Electronics Systems, pp. 85-88, May 2003.

[117] M. Di Marco, M. Forti, and A. Tesi, "Rich dynamics in weakly-coupled full-range cellular neural networks," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol 3, pp. III - 41-4, May 2004.

[118] M. Gilli and F. Corinto, "On dynamic behavior of weakly connected cellular neural networks," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol. 5, pp. V-489 - V-492, May 2004.

[119] M. Biey, M. Gilli and P. Checco, "Complex dynamic phenomena in space-invariant cellular neural networks," IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications, vol. 49, no. 3, pp. 340-345, March 2002.

[120] Zhigang Zeng, Jun Wang, and Xiaoxin Liao, "Stability analysis of delayed cellular neural networks described using cloning templates," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 51, no. 11, pp. 2313-2324, Nov. 2004.

[121] H. Jiang, Z. Li, and Z. Teng, "Boundedness and stability for non autonomous cellular neural networks with delay," Phys. Lett. A, vol. 306, pp. 313-325, 2003.

[122] V. Singh, "Robust stability of cellular neural networks with delay: linear matrix inequality approach," IEE Proc. Control Theory and Applications, vol. 151, no. 1, pp. 125-129, 2004.

[123] H. Hara, N. Takahashi, and T. Nishi, "Necessary and sufficient conditions for one-dimensional discrete-time binary cellular neural network with non symmetric connections to be stable," The 47th Midwest Symp. on Circuits and Systems, MWSCAS '04 , vol. 1, pp. I-389 - I-392, 2004.

[124] Xuemei Li, Lihong Huang, and Jianhong Wu, "A new method of Lyapunov functionals for delayed cellular neural networks," IEEE Trans. Circuits and Systems I: Regular Papers, vol. 51, no. 11, pp. 2263-2270, Nov. 2004.

[125] J. J. Yeboah Jr., Y. Ibrahim, G. A. Jullien, J. W. Haslett, "Ultra low-noise arithmetic using cellular neural networks," Micronet R & D Wkshp, pp. 105-106, 2004.

[126] Z.Galias, "Designing cellular neural networks for the evaluation of local Boolean functions," IEEE Trans. Circuits and Systems-II, vol. 40, pp. 219-223, 1993.

[127] K. Bult and H. Wallinga, "A class of analog CMOS circuits based on the square-law characteristic of a MOS transistor in saturation," IEEE J. Solid-State Circuits, vol. 22, no. 3, pp. 357-365, March 1995.

[128] P. Kinget, and M. Steyaert, Analog VLSI Integration of Massive Parallel Processing Systems, Kluwer Academic Publishers, 1997.

[129] Chichyang Chen and Rui-Lin Chen, "Performance-improved computation of very large word-length LNS addition/subtraction using signed-digit arithmetic," Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors, pp. 337-347, 2003.

[130] C. Chen, L.A. Chen, and J.R. Cheng, "Architectural design of a fast floating-point multiplication-add fused unit using signed-digit addition," IEE Proc. Computers and Digital Techniques, vol. 149, no. 4, pp. 113-120, 2002.

[131] Wen-Chang Yeh and Chein-Wei Jen, "A high performance carry-save to signed-digit recoder for fused addition-multiplication," Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP '00, vol. 6, pp. 3259-3262, 2000.

[132] K. W. Shin and Bang-Sup Song, "A complex multiplier architecture based on redundant binary arithmetic," 1997 IEEE ISCS, Jun. 1997.

[133] A.K. Cherri and M. Hamad, "Algorithms for optoelectronics implementation of trigonometric functions based on modified signed-digit numbers," The 14th Int. Conf. on Microelectronics, ICM '02, pp. 109-113, 2002.

[134] M.D. Ercegovac and T. Lang, "Simple radix-4 division with operands scaling," IEEE Trans. Computers, vol. 39, no. 9, pp. 1,204-1,208, Sept. 1990.

[135] L. Ciminiera and P. Montuschi, "High radix square rooting," IEEE Trans. Computers, vol. 39, no. 10, pp. 1,220-1,231, Oct. 1990.

[136] W.G. Natter and B. Nowrouzian, "A novel algorithm for signed-digit online multiply-accumulate operation and its purely signed-binary hardware implementation," Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '00, vol. 5, pp. 329-332, 2000.

[137] M. Kameyama, T. Seikibe, and T. Higuchi, "Highly parallel residue arithmetic chip based on multiple-valued bidirectional current-mode logic," IEEE J. Solid State Circuits, vol. 24, pp. 1,404-1,411, Oct. 1989.

[138] Shugang Wei and K. Shimizu, "Residue arithmetic circuits using a signed-digit number representation," Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS '00, vol. 1, pp. 24-27, 2000.

[139] A.G. Dempster and M.D. Macleod, "Using all signed-digit representations to design single integer multipliers using subexpression elimination," Proc. Int.Symp. on Circuits and Systems, ISCAS '04, vol. 3, pp. III-165-8, 2004.

[140] Wu Huapeng and M.A. Hasan, "Closed-form expression for the average weight of signed-digit representations," IEEE Trans. Computers, vol. 48, no. 8, pp. 848-851, 1999.

[141] D. Lo Iacono and M. Ronchi, "Binary canonic signed digit multiplier for high-speed digital signal processing," The 47th Midwest Symposium on Circuits and Systems, MWSCAS '04, vol. 2, pp. II-205-8, 2004.

[142] Li Liang, M. Ahmadi, M. Sid-Ahmed, and K. Wallus, "Design of canonical signed digit IIR filters using genetic algorithm," The Thrity-Seventh Asilomar Conf. on Signals, Systems & Computers, vol. 2, pp. 2043-2047, 2003.

[143] M. Tanaka, and A. Nishihara, "Design of signal word decomposed filters with canonical-signed digit coefficients," Proc. TENCON 2000, vol. 1, pp. 482-486, 2000.

[144] Y.M. Hasan, L.J. Karam, M. Falkinburg, A. Helwig, and M. Ronning, "Canonic signed digit Chebyshev FIR filter design," IEEE Signal Processing Letters, vol. 8, no. 6, pp. 167-169, 2001.

[145] M. Martínez-peiró, E. I. Boemo, and L. Wanhammar, "Design of high-Speed multiplierless filters using a nonrecursive signed common subexpression algorithm," IEEE Trans. Circuit Syst., vol. 49, no. 3, pp. 196-203, 2002.

[146] Fei Xu, Chip-Hong Chang, and Ching-Chuen Jong, "HWP: a new insight into canonical signed digit," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol. 5, pp. V-201-204, 2004.

[147] In-Cheol Park and Hyeong-Ju Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," IEEE Trans. CAD of ICs, vol. 12, no. 12, pp. 1525-1529, 2002.

[148] A.G. Dempster and M.D. Macleod, "Digital filter design using subexpression elimination and all signed-digit representations," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol. 3, pp. III-169-72, 2004.

[149] M. Kosunen, and K. Halonen, "A programmable FIR filter using serial-in-time multiplication and canonic signed digit coefficients," The 7th IEEE Int. Conf. on Electronics, Circuits and Systems, ICECS '00, vol. 1, pp. 563-566, 2000.

[150] K. Khoo, A. Kwentus, and A. Jr. Willson, "Programmable FIR digital filter using CSD coefficients," IEEE JSSC, vol. 31, no. 6, pp. 869-874, 1996.

[151] W. Oh, and Y. Lee, "Implementation of programmable multiplierless FIR filters with powers-of-two coefficients," IEEE Trans. CAS-11, vol. 42, no. 8, pp. 553-556, 1995.

[152] J. A. Solinas, "Low-weight binary representations for pairs of integers," Technical Report CORR 2001-41, Center for Applied Cryptographic Research, University of Waterloo, Canada, 2001.

[153] J. Proos, "Joint sparse forms and generating zero columns when combing," Technical Report CORR 2003-23, Center for Applied Cryptographic Research, University of Waterloo, Canada, 2003.

[154] R. Katti, and Xiaoyu Ruan, "Left-to-right binary signed-digit recoding for elliptic curve cryptography," Proc. Int. Symp. on Circuits and Systems, ISCAS '04, vol. 2, pp. II-365-8, 2004.

[155] N. Takagi, "Multiple-valued-digit number representations in arithmetic circuit algorithms," in Proc. 32th IEEE Int. Symp. Multiple-Valued Logic, pp. 224-235, May 2002.

[156] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford, U.K., and New York: Oxford Univ. Press, 2000.

[157] A. Azivienis, "Signed-digit number representations for fast parallel arithmetic", in IRE Trans. Elect. Comp., EC- 10, pp. 389-400, 1961.

[158] I. Choo and R.G. Deshmukh, "A novel fast parallel signed-digit hybrid multiplication scheme for digital systems," Canadian Conf. on Electrical and Computer Engineering, vol. 2, pp. 630-635, 2000.

[159] H. Fukuda, "Signed digit CMOS (SD-CMOS) logic circuits with static operation," Proc. 34th Int. Symp. on Multiple-Valued Logic, ISMVL '04, pp.128-34, 2004.

[160] T. Hanyu, A. Mochizuki, and M. Kameyama, "Multiple-valued dynamic source-coupled logic," in Proc. 33th Int. Symp. Multi-Valued Logic, pp. 207-212, May 2003.

[161] A.F. Gonzalez, M. Bhattacharya, S. Kulkarni, and P. Mazumder, "Standard CMOS implementation of a multiple-valued logic signed-digit adder based on negative differential-resistance devices," Proc. 30th IEEE Int. Symp. on Multiple-Valued Logic ISMVL '00, pp. 323-328, 2000.

[162] A.F. Gonzalez and P. Mazumder, "Multiple-valued signed digit adder using negative differential resistance devices," IEEE Trans. on Computers, vol. 47, no. 9, pp. 947-959, 1998.

[163] T. Lang and J.D. Bruguera, "Multilevel reverse-carry computation for comparison and for sign and overflow detection in addition," Proc. Int'l Conf. Computer Design ICCD '99, pp. 73-79, 1999.

[164] T. Srikanthan, S.K. Lam, and Mishra Suman, "Area-time efficient sign detection technique for binary signed-digit number system," IEEE Trans. on Computers, vol. 53, no. 1, pp.69-72, 2004.

[165] K. Navi, A. Kazeminejad, D. Etiemble, "Performance of CMOS current mode full adders", Proc. 24th Int'l. Symp. Multiple Valued Logic, pp. 27-34, 1994.

[166] I. Opris, G. Kovacs, "Analogue median circuit," Electronics Letters, 30, 17, pp. 1369-1370, 1994.

[167] S. Kawahito, M. Kameyama, T. Higuchi, "Multiple-valued radix-2 signed-digit arithmetic circuits for high-performance VLSI systems," IEEE J. Solid State Circuits, 25, 1, pp.125-131, 1990.

[168] A.A. Azivienis, A Study of Redundant Number Representations for Parallel Digital Computers, Ph.D. Thesis, University of Illinois, 1960.

[169] Y. Ibrahim, G. A. Jullien, and W. C. Miller, "Ultra low noise signed digit arithmetic using cellular neural networks," Proc. 4th Int'l Wkshp, pp, 136-142, IWSOC 2004.

[170] Y. Ibrahim, G. A. Jullien, and W. C. Miller, "Arithmetic implementation techniques using analog cellular neural networks," 17th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, IMACS '05, 2005.

[171] V.S.Dimitrov, S.Sadeghi-Emamchaie, G.A.Jullien and W.C.Miller, "A near canonic double-base number system with applications in DSP," SPIE Conference on Signal Processing Algorithms, vol. 2846, pp.14-25. 1996.

[172] V.S. Dimitrov and G.A. Jullien, "Loading the bases: a new number representation with applications", invited article in IEEE Circuits and Systems Magazine, 2nd Quarter, pp. 6-23, 2003.

[173] V.S.Dimitrov, G.A.Jullien and W.C.Miller, "An algorithm for modular exponentiation," Information Processing Letters, vol. 36, No. 5, pp. 155-159, May 1998

[174] V.S.Dimitrov, G.A.Jullien and W.C.Miller, "Theory and applications of the double-base number system," IEEE Trans. on Computers, vol. 48, No. 10, pp. 1098-1106, Oct. 1999

[175] G. A. Jullien, V. S. Dimitrov, B. Li, W. C. Miller, A. Lee, and M. Ahmadi, "A hybrid DBNS processor for DSP computation," Proc. Int. IEEE Symp. Circuits and Systems, Orlando, FL, vol. 1, pp. 5-8, 1999.

[176] V. S. Dimitrov, J. Eskritt, L. Imbert, G. A. Jullien and W.C. Miller, "The use of the multi-dimensional logarithmic number system in DSP applications", Proc. 15th IEEE Symp. on Computer Arithmetic, June, pp. 247-254, 2001.

[177] Y. Ibrahim, G. A. Jullien, W. C. Miller, and V. S. Dimitrov, "DBNS arithmetic using cellular neural networks," IEEE Int'l Symp. on Circuits and Systems, ISCAS '05, pp. 3914 - 3917, 2005.

[178] Y. Ibrahim, G. A. Jullien, and W. C. Miller, "Double-base arithmetic using cellular neural networks," submitted to the IEEE Journal on Circuits and Systems, 2005.

[179] P. Kornerup, "Computer arithmetic: exploiting redundancy in number representations," ASAP95, Strasbourg.

[180] M. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann Publishers, 2003, ISBN 1 558 60798 6

[181] S. Espejo, R. Domínguez-Castro, G. Liñán, and Á. Rodríguez-Vázquez, "A 64x64 CNN universal chip with analog and digital I/O," in Proc. 5th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS '98, Lisbon, Portugal, pp. 203-206, 1998.

[182] Hyongsuk Kim, T. Roska, H. Son, and I. Petras, "Analog addition/subtraction on the CNN-UM chip with short-time superimposition of input signals," IEEE Trans. Circuits Syst. I, vol. 50, no. 3, pp. 429-432, Mar. 2003.

Vita Auctoris

Youssef Ibrahim received the B.Sc. degree in Computer & Control Engineering from Ain Shams University, Egypt, in 1993, the M. Eng. degree in Computer Engineering from Cairo University, Egypt, in 1999 and the Ph. D. degree in Electrical & Computer Engineering from The University of Windsor, Ontario, Canada, in 2005.

In 1994, he joined the Electronics Research Institute, Egypt, as a Research Assistant. He became a Research Associate in 1999 at the same institute. From 1995 to 2000, he held an instructor position at the American University in Cairo.

His research interests areas are computer arithmetic, digital signal/image processing, circuits and systems theory and design. In particular, he is interested in practical and theoretical aspects of computation structures, including parallel systems such as cellular neural networks, with a special emphasis in the reduction of system noise. He has published several papers in these areas

Dr. Ibrahim is the recipient of many awards and scholarships including the 2003 Ontario Graduate Scholarship, Ontario, Canada, the 1999 Award of Distinction in Higher Studies, Egypt, and the 1989 Ministry of Education Medal, Egypt.