2007

Computer arithmetic based on the Continuous Valued Number System

Mitra Mirhassani
University of Windsor

Computer Arithmetic Based on the Continuous Valued Number System

by

Mitra Mirhassani

A Dissertation
Submitted to the Faculty of Graduate Studies and Research
through 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
2007

Your file Votre référence
ISBN: 978-0-494-35105-5
Our file Notre référence
ISBN: 978-0-494-35105-5

Abstract

In this dissertation, the second generation design and implementation of a new type of computer arithmetic is explored. The approach used in this dissertation for binary computations is based on a novel number system called Continuous Valued Number System, where digits are continuous, and do not have a grid. Arithmetic operations in this number system are based on the modular reduction operations.

In this dissertation, actual implementation of adders and multipliers based on this new concept are developed. These systems are implemented by analog circuitry, and are based on the current-mode circuit design approach in CMOS technology. The current-mode approach provides efficient means for implementing low-power, and highly integrated designs based on this method of computation.

The almost carry-free nature of addition in the Continuous Valued Number System results in high performance analog adders, which becomes the main building block of more complex designs such as reconfigurable adders, tree and array multipliers. The implementations are compared with their binary counterparts, and results show that the Continuous Valued Number System can open up a new path for advanced signal processing.

The Continuous Valued Number System is an analog number system, and due to data overlap between the digits, the general operations in this number system are more precise compared to regular analog systems. Therefore, new models for analog neurons are proposed

which are based on the function evaluation properties of this number system. The stochastic modeling of the proposed neuron shows that our system can tolerate more implementation errors caused by noise or by nonlinearity in analog environments.

The work presented in this dissertation, paves the way for a new type of high performance computer arithmetic and advance signal processing.

To my best friend, Ali Tahmasebi

Acknowledgments

This research would have been impossible without the support and encouragements from my supervisors, peers and family.

I would like to thank Dr. M. Ahmadi, who has been more than just an academic advisor to me. In the past years, he has been my mentor and friend and his guidance has leaded me through my graduate studies. I am very grateful to him for his continual support of me and for providing a great environment for research. He is always looking out for students here at the University of Windsor and provides a collaborative atmosphere at the Research Centre for Integrated Microsystems (RCIM).

I also owe much gratitude to Dr. G. A. Jullien, who has influenced me with his insightful comments. Dr. Jullien has provided this young research with much needed support, and helped directing it toward its successful future. I have truly enjoyed our talks and discussions during his visits.

Further gratitude goes towards my committee members; Dr. A. Jaekel of computer science, Dr. H. Wu and Dr. C. Chen from ECE department, and my external examiner Dr. B. Parhami of the University of California at Santa Barbara for their interest in this work, and for their participating in my seminars and providing constructive feedbacks.

I would like to thank my best friend, Ali Tahmasebi, who has always stood by me. Without his encouragements I might have never started my graduate studies. He has been

with me in our ups and downs, and has always encouraged me to reach higher in my life and to believe in myself. He is the true friend that everyone looks for and I had the chance to find.

I would also like to thank my parents, Dr. A. Mirhassani and Ms. Simin Dorri, and my mother-in-law, Ms. Mahin Shakuiy, for their part. They are my source of inspiration; my father's perseverance, my mother's creativity, and my mother-in-law's patience, has given me motivation and encouraged me to go further.

I thank Mehran and Mahnaz, whom I consider my brother and sister. They are my source of hope here in Canada. I should also mention my brother and sister, Shiva and Ahmad Reza, and their families back home.

I would like to thank my friends here at the University of Windsor. I emphasis on my new friends who helped and have been with me during the most critical time of my research.

Contents

List of Figures

List of Tables

List of Abbreviations

CVNS	Continuous Valued Number System
RE	Reverse Evolution
MVL	Multiple Valued Logic
MID	Most Informed Digit
LID	Least Informed Digit
ADD_b	Basic CVNS Adder
ADD_g	Group CVNS Adder
ADD_{ts}	CVNS Addition Based on Sliding Truncation
ADD_{tu}	CVNS Addition Based on Uniform Truncation
TL	Threshold Logic
DCTA3	Double Carry Threshold Adder depth 3
DCTA2	Double Carry Threshold Adder depth 2
FO4	Fanout Of 4
BKCLA	Brent-Kung Carry Look Ahead Adder
CSA	Carry Skip Adder
RCSA	Reconfigurable Carry Skip Adder
SRACSA	Static Reduced Area Carry Select Adder
BBCSA	Branch Based Carry Select Adder
SOI	Silicon On Insulator
SoC	System on Chip
OSF	Overlap Sub-Function
NN	Neural Network
NSR	Noise to Signal Ratio

List of Symbols

Notation	Definition
$\lfloor \cdot \rfloor$	Floor function
β	CVNS radix
M	Maximum range of representable values
x_i	Digits of root in Positional Number System
$\hat{x}$	Recovered root value
$\hat{x}_i$	Recovered digits in Positional Number System digits
B	Positional Number System radix
m, t	Maximum and minimum indices in Positional Number System
$(\!(x)\!)$	CVNS representation of x
$(\!(x)\!)_i$	CVNS digit
$\lfloor (\!(x)\!)_i \rfloor$	Integer value of the CVNS digit
$(\!(x)\!)_i^{\bullet}$	Fixed point of the CVNS digit
$(\!(\tilde{x})\!)_i$	Errored CVNS digit
$(\!(\hat{x})\!)_i$	Corrected or recovered CVNS digit
β	CVNS radix
n, l	Maximum and minimum indices of the CVNS digits
ϵ	Error between the root and recovered values $(x - \hat{x})$
$\overline{\epsilon}_i$	Line style error
ϵ_i°	Modular style error
I_{ϵ_i}	Congruence style error
$\epsilon_{(\!(x)\!)_i}$	The additive form of all error types in the CVNS digit
$\check{\epsilon}_{(\!(x)\!)_i}$	Relative error term with respect to radix β

Notation	Definition
$\varepsilon'_{(\!(\mathrm{x})\!)_\mathrm{i}}$	Correction error sourced by the LID
ε_i	Difference between the recovered and correct integer $((\!(\hat{x})\!)_i - (\!(x)\!)_i)$
$\varepsilon_{\mathrm{th_i}}$	Error threshold of digit i
$\varepsilon_{\mathrm{IC_i}}$	Error from the imperfect correction
$[a_1, a_2)$	Bounded set of values which includes a_1 and excludes a_2
$(\!(=)\!)$	Modular CVNS equality
$\|\|a_1, a_2\|\|_\beta$	Shortest distance between two values in the CVNS
$\lfloor (\!(x)\!)_i \rfloor$	Rounded Floor Function
ψ	Group Size

Chapter 1

Introduction

The main focus of this thesis is on the design and implementation of a new type of computer arithmetic, which is based on a novel and unconventional number system. This recently introduced number system called *Continuous Valued Number System* (CVNS) [67], has new attributes and differs from conventional number systems. We have investigated the feasibility of applying this number system for realization of efficient arithmetic units. Due to its rather recent introduction, there is no recent research dedicated to this topic. This work serves as a continuation in exploring the implementation obstacles, and also the discovery of new applications and structures for this number system.

1.1 Computer Arithmetic

The field of the computer arithmetic is mostly digital, and there are many research publications contribute to the developments of this field. Traditional digital IC implementations of computer arithmetic are based on Boolean logic and two valued signals. Most modern arithmetic processors are built with architectures that have been well-established in the lit-

erature, with many of the latest innovations devoted to special logic circuits and the use of advanced technologies.

For arithmetic implemented in System-on-Chip (SoC) technologies, there are growing demands on increasing the speed, limiting the area, and also system and cross-talk noise.

Reducing the dynamic power consumption is a major research area and has achieved many great results so far. The system noise, on the other hand, is dependent on the rate of change of voltage between the wires (cross talk) and the rate of change of current (system noise). To some extent, the continual reduction of power supply voltage aids to reduce the system noise. However, the current and voltage slewing rates do not change, which means that the instantaneous noise level is not reduced. This property is disruptive to synchronous systems where incorrect values may be latched based on this dynamic noise.

To implement arithmetic architectures, there are other approaches available with new logic families such as *Multiple-Valued-Logic* (MVL) [77, 46], which increases the signal levels from two values to multiple values. Over the last two decades, MVL system design has received increasing attention for its potential to reduce interconnections and the number of functional units in VLSI designs. In the limit, Multiple-Valued-Logic becomes the *Continuous Valued Number System* (CVNS) where the implementation is analog with the goal of reducing noise, area and interconnection.

The CVNS and basic concepts of continuous digits were first introduced in [68], fundamental methods of addition and array multiplication were given in [69], and circuit design issues were discussed in [70] and [66]. The number system is analog in nature, and it has been shown that it can also be applied for implementing low-power low-noise arithmetic units [67]. Such number systems can open up an alternative path in developing new types of arithmetic and signal processing units. This method of computation has been used successfully in developing high performance arithmetic units [53, 50, 51]. Variations of this number system such as the *Redundant Analog Number System*, have also been proposed and recently used in developing arithmetic units [75].

1.2 Principal Idea

The inspiration for this unconventional number system is taken from the utility meter, in which the ensemble of digits are used to refine the reading and increase the precision. Figure 1.1 shows the radix-10 CVNS representation of $a = 3874$, where its CVNS digits are $a_3 = 3.874$, $a_2 = 8.74$, $a_1 = 7.4$, and $a_0 = 4$. Each dial is a continuous digit as opposed to two value digits in the binary world (zero and one), and is implemented in microsystems by continuous quantities such as current, voltage or charge. If the most left hand side leftmost dial value was not exact due to some implementation error, the reminder of the dials can be used to refine and adjust the representation.

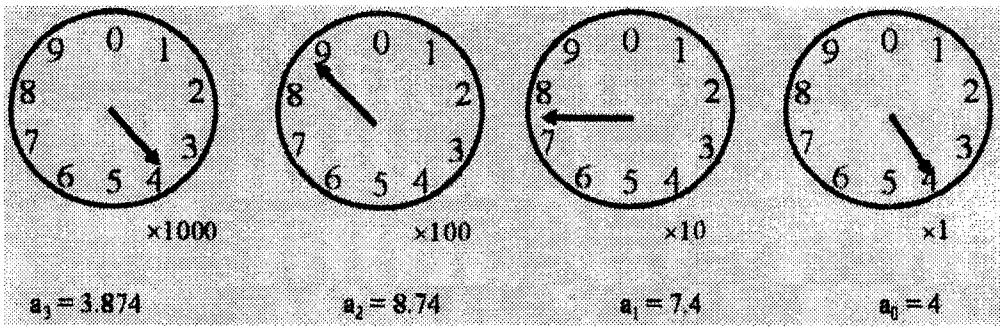


Figure 1.1: The radix-10 CVNS numerical example of $a = 3874$

The main difference between the CVNS and MVL representations is the definition of *continuous digits* or *analog digits*. The CVNS digits do not have a grid, and are realized by analog circuits. Some of the information presented by the leftmost dial is repeated in other dials. This redundancy ensures the accuracy of analog representation and operations in this number system.

1.3 Research Objective

In this thesis, the feasibility of using the CVNS for efficient, competitive, computer arithmetic implementations is investigated. The CVNS can open up a new paradigm in advanced signal processing and analog neural network implementations. Our aim is to develop a second generation of integrated computing units based on the CVNS. The proposed systems are proof of concept of the CVNS, and are compared with the conventional digital designs.

The motive behind this research has been our academic curiosity to search for novel and advanced signal processing alternatives. Previous work on this number system, explored the basic properties of this number system. The key properties of the CVNS indicate the advantages that this system may offer, however feasibility and actual implementation of arithmetic units based on this number system has not been investigated.

1.4 Thesis Organization

The key properties of the CVNS are presented in chapter 2. This chapter expresses some of the most important features of this number system, which are of further use for arithmetic development. The mathematical basis of the CVNS is first presented to provide background information, and numerical examples are used to clarify this theory.

Chapter 3 provides a systematic means for implementing CVNS adders. In this chapter, we deal with CVNS systems that have power-of-two radices in order to simplify the conversion technique between binary and the CVNS representation; yielding efficient and competitive arithmetic circuit designs. A new approach for addition of unsigned binary values using the CVNS is proposed, and two methods for generating the adder structures are introduced. Also, a new technique for converting the CVNS results back to binary representation is given.

Computational complexities for the CVNS adder are derived in this chapter, to provide a measure of the required area and delay. This complexity measure is used to compare the CVNS adder with the Threshold Logic implementations of binary adders.

This chapter also presents the implementation of a two operand 16-bit radix-4 CVNS

adder. This adder is a key unit for developing other complex structures such as reconfigurable adders, and array multipliers.

In chapter 4 an important new path for the CVNS is opened up, which can encourage further developments of the CVNS. Our CVNS adders require fewer interconnections, and therefore reconfigurable adders can be implemented using this approach.

Also in Chapter 4, the design and implementation of a mixed-signal 64-bit adder is developed, which is a cascade of four 16-bit adders. The 64-bit adder is implemented in a TSMC CMOS $0.18\mu m$ technology, and design and performance comparisons are given by at end of the chapter.

Chapter 5 presents a new type of CVNS multiplier by using CVNS compressors. These CVNS compressors reduce the partial product columns by evaluating the values down to one gate delay. At the final stage, two CVNS adders are used to generate the final result. The column compression and addition is based on the CVNS theory of digit wise addition, and the proposed multiplier is designed into the target $0.18\mu m$ CMOS technology and compared with standard binary multipliers in the same technology.

Chapter 6 focuses on the CVNS array multipliers. In this chapter again we take advantage of our novel and efficient high radix CVNS adders to implement CVNS array multiplier. The design and implementation of a 16×16 array multiplier is also developed, and two alternative methods of multiplications are given, which are very similar to the array multiplier, and can be easily implemented using the same style of circuit design.

In Chapter 7, the CVNS is used to interface and process analog signals (not quantized data; binary). The developed Analog Neural Network (Analog NN) is based on the function evaluation properties of the CVNS, and an Adaline neuron with a nonlinear activation function is proposed. In this preliminary work, it has been shown that the Continuous Valued Number System can increase the robustness of Analog Neural Networks considerably.

Finally, chapter 8, gives the summary of our contributions, and provides some suggestions for future research on this subject.

Appendix A is on the error correction properties of the CVNS. Error correction is a

fundamental and key property of the CVNS, and it requires further analysis. It should be pointed out that although this appendix expresses some of the perviously featured properties of the *Reverse Evolution* technique (which is for removing error and fine-tuning the CVNS representation), here we correct some of the previous work and we also use a simpler approach to describe the overall concept. Development of the Reverse Evolution technique is a key factor in applying the Continuous Valued Number System for future storage and memory applications.

Chapter 2

Continuous Valued Number System

The Continuous Valued Number System is a novel analog-digit representation and arithmetic system. This recently introduced number system performs arithmetic operations by applying digit-level modular reduction operations on continuous real values. There are no carries, in the accepted sense, and addition is performed without digits intercommunication. Digits share information and there is a digit-level overlap between them, which results in an arbitrary precision representation. In this section, the main properties of the CVNS including digit generation, addition and multiplication are given.

2.1 CVNS Analog Digits

Any real value, x, within a boundary $|x| \leq M$ represented in a positional number system with radix B, can be mapped to CVNS digits, $(\!(x)\!)_i$, in radix-β. The CVNS value, $(\!(x)\!)$, is the ensemble of the CVNS digits, $(\!(x)\!)_i$, and can be written as a vector as follows:

$$(\!(x)\!) \Rightarrow \{(\!(x)\!)_L, (\!(x)\!)_{L-1}, ..., (\!(x)\!)_0 | (\!(x)\!)_{-1}, ..., (\!(x)\!)_{-k}\} \tag{2.1}$$

where $(-k \leq i \leq L)$ represents the indices of the CVNS digits, and the bar (|) represents the radix point.

One of the fundamental aspects of the CVNS is that the digits with higher indices have higher information density compared to the digits with lower indices. In fact, the digit that has the highest index in the CVNS system, contains the whole information regarding the original value or root value that is being represented in the CVNS system. In other words, just by looking at the highest index CVNS digit, the root value can be obtained within the precision and noise constraints of the implementation. Therefore, the term "*Most Significant Digit*" can not be applied in the same context as any other conventional number systems. Hence in this dissertation, we refer to the digits either by their indices, or by the level of the information that they contain; for example, the digit with the highest index in the system is called the "*Most Informed Digit*" or MID in short. The index value of the MID is chosen such that:

$$\beta^{L+1} \leq M \tag{2.2}$$

There are two main methods for the CVNS digit generation. A series method, named *Cascade Digit Generation*, and a parallel method, which is called *Modulus Digit Generation*. These methods are discussed in the next sections.

2.2 Cascade Digit Generation

Any real value, x, within the boundary $|M|$ can be converted into its equivalent CVNS representation by applying this series method. This method generates the digits in series starting from the MID as shown below:

$$(\!(x)\!)_L = \frac{x}{M}.\beta \tag{2.3}$$

with an associated integer, $int((\!(x)\!)_L) = \lfloor (\!(x)\!)_L \rfloor$.

The next digit is as follows:

$$((x))_{L-1} = \Big(((x))_L - \lfloor((x))_L\rfloor\Big).\beta \tag{2.4}$$

All of the CVNS digits are obtained one by one; a series method. The other method, which is a parallel technique, is discussed later. The choice of which method to apply is determined by the mathematical convenience or by the hardware implementation requirements.

Example: We are going to find the radix-10 CVNS digits of an arbitrary value $x = 78.324 < 100$, using the cascade method. To satisfy expression (3.1), $L = 1$, and therefore the CVNS digits are obtained in series as follows:

$$((x))_1 = \frac{78.324}{100}.10 = 7.8324$$

The rest of the CVNS digits are obtained as follows:

$$((x))_0 = \big(((x))_1 - \lfloor((x))_1\rfloor\big).\beta = (7.8324 - 7).10 = 8.324$$

$$((x))_{-1} = \big(((x))_0 - \lfloor((x))_0\rfloor\big).\beta = (8.324 - 8).10 = 3.24$$

$$((x))_{-2} = \big(((x))_{-1} - \lfloor((x))_{-1}\rfloor\big).\beta = (3.24 - 3).10 = 2.4$$

$$((x))_{-3} = \big(((x))_{-2} - \lfloor((x))_{-2}\rfloor\big).\beta = (2.4 - 2).10 = 4$$

2.3 Modulus Digit Generation

The CVNS digits are obtained by applying a basic modular reduction operation as follows:

$$((x))_i = \Big(\frac{x}{M}.\beta^{L-i+1}\Big)\ mod\beta \tag{2.5}$$

where *mod* is the modulo operation on any real value such that $(a)mod\beta = a - I \times \beta$, $0 \leq (a)mod\beta < \beta$, and I is an integer.

Example: We are going to find the CVNS digits of the previous example, using the modulus reduction method as follows:

$$((x))_1 = \left(\frac{78.324}{100}.10\right) mod 10 = 7.8324$$

$$((x))_0 = \left(\frac{78.324}{100}100\right) mod 10 = 8.324$$

$$((x))_{-1} = \left(\frac{78.324}{100}.10^3\right) mod 10 = 3.24$$

$$((x))_{-2} = \left(\frac{78.324}{100}.10^4\right) mod 10 = 2.4$$

$$((x))_{-3} = \left(\frac{78.324}{100}.10^5\right) mod 10 = 4$$

2.4 CVNS Digit Ring

The CVNS digits are elements of a real ring, and are bounded by the radix of the CVNS representation. The CVNS ring is a bounded yet infinite set $[0, \beta)$ with an additive operation on it such that closure and inverse holds true. The name *ring* associates the CVNS with some form of circularity, which is represented best by a bounded set. The term real ring is in analogy to the integer ring. The integer ring, together with polynomial rings occur in the signal porcessing fields of Number Theoric Transforms and Digital Signal Processing. The common attributes of a mathematical ring applies to the CVNS real ring as well.

2.5 Canonic Digits

A set of the CVNS digits are in *Canonic Form* if all of the digits in the set satisfy:

$$\left.\begin{array}{l} 0 \leq (\!(x)\!)_i = \big((\!(x)\!)_i\big) mod\beta < \beta \quad \forall i \quad x \geq 0 \\ \\ -\beta < (\!(x)\!)_i = \big((\!(x)\!)_i\big) mod\beta \leq 0 \quad \forall i \quad x < 0 \end{array}\right\} \tag{2.6}$$

Digit values that violate 3.20 indicate that an error has occurred in that set, and should be corrected.

2.6 Vicinity of Digits

Arithmetic distance between two CVNS values is defined as the distance between the two points on a real ring which is similar to geometric distances on a circle. There are always two paths between any two points on a ring, namely clockwise and counter clockwise. Using this definition, equality in the CVNS is congruent, and is defined as follows:

$$x + I.\beta \equiv x \tag{2.7}$$

which holds for any integer I.

Definition of distance in the CVNS is used to define the vicinity of a point on the real ring. This definition becomes important in integer retrieval during error correction. The distance between two points is the shortest path between them:

$$\| x, y \|_\beta = min\{(x - y) mod\beta, (y - x) mod\beta\} \tag{2.8}$$

where $(x) mod\beta = x - I.\beta$ and $0 \leq (x) mod\beta < \beta$ and I is an integer.

The distance between two points is considered a positive value whether $x \leq y$ or vice versa, and $\| x, y \|_\beta \leq \frac{\beta}{2}$.

Vicinity of points on the real ring is also congruent. All the points y on the ring which adhere to the $\| x, y \|_\beta < v$ are in the vicinity of x, which means that their distance to x along the ring is less than v.

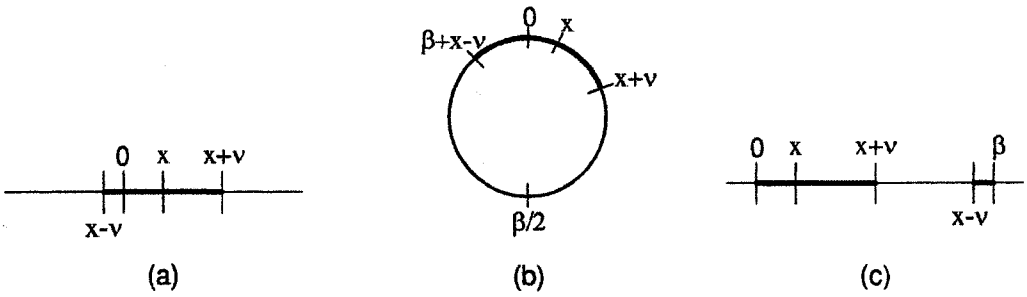


Figure 2.1: Vicinity on a line (a), on a ring (b) and on a ring that is peeled open at zero (c)

The definition is shown in Fig. 2.1 for a regular linear environment and for the CVNS ring. In (a), point x is placed on the line, and in (b) on the ring, and in both cases an arbitrary vicinity of the point is indicated by a thick line and thick arc segment respectively. If the ring is peeled open at zero as a line, shown by (c), it may not look like a geometric vicinity any more. This is an important issue when a ring is implemented by a line-style implementation medium.

2.7 Analog Digits Relationship

Any CVNS digit in general is composed of two parts; the integer part, $\lfloor ((x))_i \rfloor$, and the redundant part which is shared by its less informed digits. The relation between a digit and its next less informed digit is:

$$((x))_i = \lfloor ((x))_i \rfloor + \frac{((x))_{i-1}}{\beta} \tag{2.9}$$

By applying this relationship successively between the CVNS digits, the MID can be obtained as follows:

$$((x))_L = \frac{((x))_i}{\beta^{L-i}} + \sum_{n=i+1}^{L} \frac{\lfloor((x))_n\rfloor}{\beta^{L_n}} \quad ,i \leq L-1 \tag{2.10}$$

The above equation presents a flexible storage medium where a combination of low resolution cells can be employed for storing the high resolution CVNS values. This equation can be generalized for obtaining any digit from its corresponding less informed digits as follows:

$$((x))_j = \frac{((x))_i}{\beta^{j-i}} + \sum_{n=i+1}^{j} \frac{\lfloor((x))_n\rfloor}{\beta^{j-n}} \quad ,i \leq j \leq L \tag{2.11}$$

This equation is also used to evaluate the root around a fixed point. The fixed point is determined from a set of more informed associated integers, $\lfloor((x))_L\rfloor$, $\lfloor((x))_{L-1}\rfloor$, ..., $\lfloor((x))_{i+1}\rfloor$. The vicinity of the fixed point is scanned by $((x))_i$, and the fixed point is defined as:

$$((x))^{\bullet} = \frac{M}{\beta} \sum_{j=i+1}^{L+\ell} \frac{\lfloor((x))_j\rfloor}{\beta^{L_j}} \tag{2.12}$$

The CVNS representation is in fact a hierarchial approximation of a root value. The MID is at the highest level and it is only at this level that the associated integer of the digit is not required. Digit significance and its information of the root value decreases as we move down towards the LID.

2.8 Addition

The addition operation is performed independently on each of the CVNS digits. Considering two values x and y, where $x, y < |M|$, the CVNS digits of $((z)) = ((x))((+))((y))$, are obtained by digit wise addition of the CVNS digits as follows:

$$((z))_i = \Big(((x))_i + ((y))_i\Big) mod \beta \tag{2.13}$$

The addition of the CVNS values is by summation of the columns without intercommunication. The modular reduction operation after the addition prevents the digits exceeding the radix value in case of addition overflow.

Example: Radix-10 CVNS addition of two arbitrary values $x = 58.34$ and $y = 72.89$ is shown in Table 2.1. The number of required CVNS digits is $L = 1$, and $k = 2$. However, it can be noted that we have obtained a CVNS digit value for an indix higher than $L = 1$. The digits with indices higher than the MID are called *Excessively Evolved* Digits. These digits are used for overflow checking and and are only zero if the root value is zero. These digits are similar to the leading zeros in unsigned binary number system used for range extension, or to leading 1s for sign extension in 2's complement representation of binary values. However, in the CVNS these digits are the scaled version of the MID value and are zero only if the original value is zero.

Table 2.1: CVNS addition between two arbitrary values

i	2	1	0	−1	−2
$(\!(x)\!)_i$	0.5834	5.834	8.34	3.4	4
$(\!(y)\!)_i$	0.7289	7.289	2.89	8.9	9
$(\!(z)\!)_i$	1.3123	3.123	1.23	2.3	3

The CVNS digits of z are $\{1.3123, 3.123, 1.23\ 2.3, 3\}$, this CVNS digit set represents $z = 131.23$, which is the correct result.

2.9 Multiplication

Unlike addition, multiplication in the CVNS can not be performed digit wise. The CVNS multiplication is performed by a summation of partial products. It has been shown that multiplying two values by applying CVNS theory is as follows:

$$((z))_j = \Big(\sum_{\forall i} ((x))_{j-i} \times y_i\Big) mod\beta \tag{2.14}$$

where $0 < j \leq L$.

If the radix of the conventional number system is chosen as 2 (binary form), then y is represented in binary and each of its digits serve as an off-on switch for summing the CVNS partial products.

Example: The radix-10 CVNS multiplication of two arbitrary values, $x = 31.89$ and $y = 2.38$ are shown in Table 2.2. The MID index is $L = 1$ and $k = 1$ is chosen. Therefore, the CVNS digits of x are $((x)) \equiv (0.03189, 0.3189, 3.189, 1.89, 8.9 \mid 9)$, while y is kept in its original decimal form.

Table 2.2: CVNS multiplication between two arbitrary values

i	3	2	1	0	−1
$((x))_i$	0.03189	0.3189	3.189	1.89	8.9
$((x))_i \times y_0$	0.06378	0.6378	6.378	3.78	7.8
$((x))_i \times y_{-1}$	0.09567	0.9567	9.567	5.67	6.7
$((x))_i \times y_{-2}$	0.25512	2.5512	5.512	5.12	1.2

The CVNS digits of the result, z, are obtained from these partial products as follows:

$$((z))_{-1} = ((x))_{-1}.y_0 + ((x))_0.y_{-1} + ((x))_1.y_{-2} = (18.982) mod 10 = 8.982$$

$$((z))_0 = ((x))_0.y_0 + ((x))_1.y_{-1} + ((x))_2.y_{-2} = (15.8982) mod 10 = 5.8982$$

$$((z))_1 = ((x))_1.y_0 + ((x))_2.y_{-1} + ((x))_3.y_{-2} = 7.58982$$

Thus the CVNS digits of the product are $((z)) \equiv \{7.58982, 5.8982 \mid 8.982\}$, which is equivalent to $z = 75.8982$. This result is verified by the fact that the result of the multiplication is $x \times y = 31.89 \times 2.38 = 75.8982$.

2.10 CVNS Error Correction

CVNS is a modular number system and hence digit values are within a bounded ring $[0, \beta)$. When an error occurs in a CVNS digit which is close to the boundaries of the ring, that digit may wrap around the ring. Moreover, implementation of the CVNS using analog circuitry may cause such small errors in the CVNS digits, thus leading to this problem. These errors can be removed and corrected using a process called *Reverse Evolution* (RE) [67].

The RE process for correcting a CVNS digit that is in error, $((\tilde{x}))_i$, is to find the rounded floored function of the CVNS digits as follows:

$$\lfloor((\tilde{x}))_i\rfloor_R = \begin{cases} \left[((\tilde{x}))_i - \frac{((\tilde{x}))_{i-1}}{\beta}\right]_R mod^+\beta & x \geq 0 \\ \left[((\tilde{x}))_i - \frac{((\tilde{x}))_{i-1}}{\beta}\right]_R mod^-\beta & x < 0 \end{cases} \tag{2.15}$$

where mod^+ and mod^- are defined as follows:

$$(a)mod^+\beta = a + I.\beta; \quad 0 \leq (a)mod^+\beta < \beta$$

$$(a)mod^-\beta = a + I.\beta; \quad -\beta < (a)mod^+\beta \leq 0 \tag{2.16}$$

The corrected CVNS digits are obtained as follows:

$$((\hat{x}))_i = \begin{cases} \lfloor((\tilde{x}))_i\rfloor_R + \frac{((\hat{x}))_{i-1}}{\beta} & i > -k \\ ((\tilde{x}))_k & i = -k \end{cases} \tag{2.17}$$

Example: The radix-10 CVNS digits of an arbitrary value $x = 9.9587 < 10$ are given in the first row of the Table 2.3. A 5% error is applied to all the digits as shown in the second row. The third row shows the rounded floor function values, and corrected digits are in the last row. The only digit that remains unchanged is the LID. The error in the LID propagates upward and effects other CVNS digits.

Table 2.3: CVNS addition between two arbitrary values

i	2	1	0	-1	-2
$(\!(x)\!)_i$	9.9087	9.087	0.87	8.7	7
$(\!(\tilde{x})\!)_i$	10.404135	8.921	0.91	9.1	6.9
$\lfloor(\!(\tilde{x})\!)_i\rfloor_R$	$\lfloor 9.495445\rfloor = 9$	$\lfloor 9.102\rfloor = 9$	$\lfloor 0.041\rfloor = 0$	$\lfloor 8.41\rfloor = 8$	–
$(\!(\hat{x})\!)_i$	9.90869	9.0869	0.869	8.69	6.9

Error correction is an important feature in the CVNS, which allows for digits to be corrected, and to be restored to their original value. The RE process relaxes the target accuracy in the arithmetic operations. Therefore, the CVNS has the advantage of correcting faults over arithmetic operations. The error correction features of the CVNS are given in more detail in Appendix A.

2.11 Root Recovery

Any CVNS value can be converted back into its corresponding weighted magnitude representation, or root value, rather easily. From (6.8), it can be noted that the MID is the scaled version of the root value. The root of the CVNS values can be obtained from the MID as follows:

$$\hat{x} = (\!(x)\!)_L \frac{M}{\beta} \tag{2.18}$$

Since the MID requires high resolution, it may not be possible to implement it exactly using analog circuitry. If, however, the MID information is correct (without any error), the recovered value is not going to have any quantization or approximation error, and is clearly equal to the original value ($\hat{x} = x$).

The root value can also be recovered from the less informed digits which have lower resolution requirements. It can be obtained from any other particular digit as long as the associated integers of its more informed digits are known.

Moreover, to recover values from their CVNS format, and to obtain the result in the positional number system form, the non-iterative process used for error correction, Reverse Evolution (RE), can be used. This process is originally used for refining the CVNS digit values and removing errors from the CVNS digits.

In this approach, the final result is generated by checking the value of the less informed digits one by one. Thus this method generates the final outcome using a serial method.

2.12 CVNS Function Evaluation

The CVNS can increase the precision of analog values due to information overlap and the hierarchial level of information in its digits. Our goal is to obtain the digits of a function, which for example, can be the nonlinear activation function of an artificial neuron from the digits of the operand. Given $x, y < |M|$, and a function $b = f(a)$, we will find the CVNS function $f_j(.)$ for all of the CVNS digits of y such that:

$$(\!(y)\!)_i = f_j\big((\!(x)\!)_i\big) \tag{2.19}$$

The MID digit of $(\!(y)\!)$ is:

$$(\!(y)\!)_L = f_L\big((\!(x)\!)_L\big) = f\Big(\frac{M}{\beta}.(\!(x)\!)_L\Big).\frac{\beta}{M} \tag{2.20}$$

From the above equation we can compute other CVNS digits from the MID as follows:

$$(\!(y)\!)_i = ((\!(y)\!)_L.\beta^{L-i}) mod \beta \tag{2.21}$$

At this point we can compute any function from the input MID digit, since it describes the value of the input variable x. An example is given below to illustrate the CVNS function evaluation properties.

Example: Given the function $f(x) = 2x^2$, we wish to find digits of $f(3.24)$ in decimal CVNS ($\beta = 10$) with $M = 100$.

With $L = 1$ and choosing $k = 1$, the CVNS digit set of $x = 3.24$ is found as $(\!(x)\!) \equiv \{0.324, 3.24|2.4\}$.

The MID is $(\!(x)\!)_1 = 0.324$, hence, $f_1((\!(x)\!)_1) = 20(\!(x)\!)_1^2$. The output MID is $(\!(y)\!)_1 = 20(0.324)^2 = 2.09952$, and from that we find other required digits $(\!(y)\!)_0 = 0.9952$ and $(\!(y)\!)_{-1} = 9.952$.

From the output MID, we can obtain $y = 100 \times 2.09952/10 = 20.9952$. We can verify that $2.(3.24)^2 = 20.9952$.

As demonstrated by this example, the MID digit $(\!(y)\!)$ can be obtained from the MID digit of the input using $f(.)$, and then the remaining CVNS digits are obtained from the MID. It is also possible to obtain the MID from the lower informed digits if the associated digits of the input are known.

2.12.1 Overlapped Sub-Function

There are other methods for computing the CVNS of output variable from the CVNS of the input variable. For example, the MID can be obtained around a fixed point, from a set of more informed associated integers, $\lfloor (\!(x)\!)_L \rfloor$, $\lfloor (\!(x)\!)_{L-1} \rfloor$, ..., $\lfloor (\!(x)\!)_{i+1} \rfloor$ as follows:

$$(\!(x)\!)_j = \frac{(\!(x)\!)_i}{\beta^{j-i}} + \sum_{t=i+1}^{j} \frac{\lfloor (\!(x)\!)_j \rfloor}{\beta^t} \tag{2.22}$$

for $i \leq j \leq m-1$.

By inserting Equation (2.21) in Equation (2.22), we can obtain:

$$((y))_j = \Big(f\big(\xi_L.\frac{((x))_i}{\beta^{L-i}} + \xi_L \sum_{t=0}^{L-i-1} \frac{\lfloor((x))_{L-t}\rfloor}{\beta^t}\big)\xi_L^{-1}.\beta^{L-j}\Big) mod\beta \tag{2.23}$$

where $\xi_i = M.\beta^{i-L-1}$, and $\xi_L = M/\beta$.

The above expression is the basis for finding the digit functions $f_i(.)$ from $f(x)$ for digits $i \leq L-1$. We will identify all the fixed points of a digit at level i by the tuple $T_i = [\lfloor((x))_L\rfloor, \ldots, \lfloor((x))_{i+1}\rfloor]$. The size of the tuple depends on the significance of the fixed point. For decreasing i, digit significance decreases toward the LID (Least Informed Digit), and the tuple increases in size. The MID does not require any tuple, since it is at the highest level of hierarchy.

We prepare a function $g_{T_i}(\rho)$ for each tuple T_i, and the domain of each of those functions equals the range of $((x))_i$, namely $[0, \beta)$ for positive values and and $(-\beta, 0]$ for negative values. Those functions are prepared for $\rho = ((x))_i$ as follows:

$$g_{T_i}(\rho) = f\Big(\frac{\rho.M}{\beta^{L-i-1}} + \sum_{t=0}^{L-i-1} \frac{M.\lfloor((x))_{L-i}\rfloor}{\beta^{t+1}}\Big) \tag{2.24}$$

A set of functions $G^i_{T_i}$ from the g_{T_i} as follows:

$$G^j_{T_i}(\rho) = \Big(\frac{g_{T_i}(\rho)\beta^{L-j+1}}{M}\Big) mod\beta \tag{2.25}$$

and conveniently we have:

$$((y))_j = G^j_{T_i}(\rho) \tag{2.26}$$

whereby $y = f(x)$ and specific permutation of T_j is obtained for x.

The function $G^j_{T_i}$ is termed the *Overlapped Sub-Function* (OSF), and is generally calculated for $i, j = L, L-1, \ldots, -k$ to prepare the function evaluations. With these, we are now able to find any CVNS digit of y, given any single CVNS digit of x and its associated permutation of the tuple T_j, using the corresponding function $G^j_{T_i}$. The redundancy among

the sub-functions is similar to redundancy among the digits. An example is given next to demonstrate these properties.

Example: Let us calculate $y = 2x^2$ for a radix-2 CVNS ($\beta = 2$), within the requirements $|x|, |y| < M$ with $M = 5$. With $M \leq 2^3$, we have $L = 3$ and choose $k = 1$ as our dimensions of the CVNS representation for this example.

The MID function is $f_L(\rho) = f(\frac{5\rho}{2}).\frac{2}{5} = 5\rho^2$. The OSF's for the next level of hierarchy, $i = 2$, are chosen to compute the output $(\!(2x^2)\!)_2$. The tuple for this level consists of one binary element, and hence there are two OSF's at this level; $G_0^2(\rho) = (5\rho^2/2) mod 2$ and $G_1^2(\rho) = (5(\rho+2)^2/2) mod 10$.

At the lowest level, $i = k = 1$, the tuple consists of two binary elements, and there are four OSF's, which are used to compute $(\!(2x^2)\!)_1$. These subfunctions are:

$$G_{0,0}^1(\rho) = (5\rho^2/4) mod 2,$$

$$G_{0,1}^1(\rho) = (5(\rho+2)^2/4) mod 2,$$

$$G_{1,0}^1(\rho) = (5(\rho+4)^2/4) mod 2,$$

$$G_{1,1}^1(\rho) = (5(\rho+6)^2/4) mod 2.$$

For an input $x = 1.3$, the CVNS digits are $(\!(x)\!) \equiv \{0.52, 1.04|0.08\}$. The integers for selecting an OSF is found by flooring the corresponding CVNS digits. The output CVNS digits are obtained from the MID and OSF's: $f_3(0.52) = 1.352, G_0^2(1.04) = 0.704$ and $G_{0,1}^1(0.08) = 1.408$. To confirma our results we compute the CVNS digits of $2(1.3)^2 = 3.38$ directly using the basic CVNS digit generation; $(\!(3.38)\!) \equiv \{1.352, 0.704|1.408\}$, which matches the results obtained by the OSF approach.

2.13 Summary

In this chapter, key properties of the CVNS have been given. The CVNS digits share information, and provide a chain style support for each other. The CVNS digits are continuous or analog, and do not have a grid. There are two general methods for digit generation, which are called Cascade digit generation and Modulus digit generation. The cascade method requires calculation of digits is sequence starting from the Most Informed Digit down to the Least Informed Digit (LID). The modulus method, allows the computation of any digit directly from the root value.

One of the main characteristics of the CVNS is the carry free addition between the CVNS digits. The chain of adders is particularly useful for developing other complex CVNS operations such as multiplication.

Protection of any CVNS digit is warranted by the redundancy among the digits. This redundancy allows for error correction in a CVNS digit set by a process called Reverse Evolution. Reverse Evolution reduces the error in the more informed digits.

In the following chapters, the design and implementation of the first generation of adders and multipliers based on the CVNS are given. The performance of these systems are compared with their digital counterparts, in terms of area, power and speed. The general approach in these designs is based on the CMOS current-mode analog circuit design, and we will find that the current-mode approach can provide low-power, and highly compact analog arithmetic designs.

Chapter 3

CVNS Addition

Traditional digital IC implementations are based on 2-level Boolean logic and perform the required arithmetic operation with two valued signals. As demand for handheld devices increases, low-power low-noise implementations of arithmetic and signal processing units becomes desireable. This is not only true for hand-held devices, but with any processor built with the latest very deep sub-micron technology. With very low power supply voltages and very fast switching, dynamic noise can be a major problem particularly in terms of substrate currents where SOI processing is now almost mandatory. Every day researchers look for alternative solutions in order to come up with more efficient implementations, and there are always new published results in the literature. However, most of these designs are based on Boolean algebra, and they essentially follow the same path and algorithms for performing the arithmetic operations.

Some of the highest-speed adder architectures, based on Boolean algebra include the carry lookahead adder [85] , carry skip adder [44] , carry select adder [7] , conditional-sum adder [72] , carry-increment adder [79, 93] , and carry-look ahead variations such as Brent-Kung [10] , Kogge-Stone adder [41] , Ladner-Fisher [42] , Han-Carlson [29] and Knowles [40]

. Each of these methods offers different advantages and tradeoffs in terms of delay of the operation, occupied area, power consumption, noise and wiring complexity.

3.1 Introduction

Addition is considered the most basic arithmetic operation, on which the other arithmetic functions are built, and any improvement in the adder performance will improve other operations that use addition such as multiplication. In the case of digital adder architectures, the basic structures, developed over the past few decades, are still employed. In this section, the design of CVNS adders is studied.

One specific class of the CVNS has proven very effective for radix-2 addition. We introduce this class, and discuss architectures and implementations in this dissertation. This class of the CVNS is able to interface with digital inputs directly, and the conversion process between binary and the CVNS is very simple and effective. Classical analog circuit blocks are used to construct adders with arbitrary precision. In this chapter, this class of the CVNS theory is discussed. We discuss the two operand addition of this number system, and the required expression are given in the CVNS.

In this chapter, a simplified conversion technique from binary to CVNS representation is introduced, and the general method for two operand binary addition is explained. Moreover, a new approach for addition using the CVNS representation is proposed which reduces the hardware cost of implementations for CVNS adders, and two methods for generating the adder configuration and structure are introduced.

Also, a new technique for converting the CVNS results back to binary representation is presented. Computational complexity for the CVNS adder is explained, to provide a measure in terms of the required area and delay. This complexity measure is used to compare the hardware cost of implementing the CVNS adder with Threshold Logic implementations of binary adders, which is the closest technology to the CVNS currently in use.

3.2 Addition of Two Radix-2 Operands in CVNS

The CVNS number representation is analog, therefore digits do not have a grid, and each digit takes on a continuous value. Although the CVNS is continuous, and does not introduce digitization errors, the original binary values are not continuous. As we will show, applying either of the CVNS digit generation methods results in a convenient method for converting discrete binary values into the CVNS representations by directly using the binary digits. The conversion between the two number systems is simple and effective and does not involve a computationally complicated process.

3.2.1 Binary to CVNS Conversion

Binary numbers can be converted into their CVNS representation rather easily under some assumed conditions. These assumptions do not affect the CVNS theory; however, they create a simple interface between the two number systems. The CVNS digit counterparts are, in fact, generated by applying either one of the given methods for the CVNS digit generation. In order to simplify the conversion, we can always assume that the maximum representable ranges in both number systems are equal, which leads to:

$$B^{m+1} = \beta^{L+1} = M \tag{3.1}$$

where m and L are the maximum index values of the binary and the CVNS number systems respectively, and $B = 2$ is the binary number system radix.

Although the CVNS radix can be equal to any integer, for a simple interface between the CVNS and binary representation, we focus on the CVNS radices:

$$B^{\nu} = \beta \tag{3.2}$$

with integer $\nu \geq 1$.

The last two conditions do not change the fundamentals of the CVNS and can be applied without loss of generality of the CVNS theory. These requirements only make the interface

between the two number systems more efficient and less complicated.

Any binary value represented can be shown as below:

$$x = \pm \sum_{i=0}^{m} x_i . B^i \tag{3.3}$$

where x_i represents the $m+1$ binary digits and in this case $B = 2$, is the binary number system radix.

By applying the given expressions for the CVNS digit generating, the MID of the corresponding CVNS number is equal to:

$$((x))_L = \sum_{i=0}^{m} x_i . B^{i+\nu-m-1} \tag{3.4}$$

It can be seen from the above equation that, in this case, the CVNS digit is obtained directly from the binary digits. The associated integer part of the MID is the partial sum of the digits with $B^{i+\nu-m-1} \geq 1$, which is satisfied for the index range $m+1-\nu \leq i \leq L$, and can be obtained as follows:

$$\lfloor ((x))_L \rfloor = \sum_{i=1-\nu}^{0} x_{m+i} . B^{i+\nu-1} \tag{3.5}$$

The next CVNS digit and its associated integer are equal to:

$$((x))_{L-1} = \sum_{i=0}^{m-\nu} x_i . B^{i+2\nu-m-1} \tag{3.6}$$

$$\lfloor ((x))_{L-1} \rfloor = \sum_{i=1-\nu}^{0} x_{m+i-\nu} . B^{i+\nu-1} \tag{3.7}$$

This process is used for obtaining all of the remaining CVNS digits. In general, any CVNS digit can be obtained from the expression given below:

$$((x))_{L-j} = \sum_{i=0}^{m-j\nu} x_i . B^{i+(1+j)\nu-m-1} \tag{3.8}$$

The conversion process to the CVNS representation directly applies to each digit value (in case of binary inputs, 0 or 1), which means less hardware and a faster conversion process. The required number of the CVNS digits under the condition in Equation (3.1) is:

$$L = \left\lceil (m+1)\frac{logB}{log\beta} \right\rceil - 1 = \left\lceil \frac{m+1}{\nu} \right\rceil - 1 \tag{3.9}$$

Equation (3.9) shows the upper bound of the required CVNS digits. If the implementation environment is able to distinguish between B^{ψ} different levels, the LID (Least Informed Digit) can be extracted using a quantizer generating ψ lower significant binary digits. In this case the minimum required number of CVNS digits is reduced to:

$$L = \left\lceil \frac{m+1-\psi}{\nu} \right\rceil - 1 \tag{3.10}$$

3.2.2 CVNS to Binary Conversion

The result of the CVNS addition should be converted back into the target binary format. The modular reduction operation has to be applied to the CVNS results after each two operand addition, and the binary output results from the integer modulo operation. If the CVNS radix is chosen higher than two ($\nu > 1$), each CVNS digit is translated into ν binary digits. For example, if the CVNS radix is chosen to be 4, then each CVNS digit is equivalent to 2 binary digits. Conversion of a CVNS digit, $(\!(z)\!)_{L-j}$, back into positional number system representation can be expressed as follows:

$$z_{L-j\nu+i} = \begin{cases} 1 & (\beta^i (\!(z)\!)_{L-j}) \, mod\beta > \beta/2 \\ 0 & (\beta^i (\!(z)\!)_{L-j}) \, mod\beta \leq \beta/2 \end{cases} \tag{3.11}$$

where $0 \leq j \leq L$ and $0 \leq i \leq \nu$.

3.2.3 Addition

The addition of two converted numbers can be easily performed in the CVNS representation by using (2.13). The digit wise addition of the two CVNS numbers given by the independent addition of each of the corresponding CVNS digits in the two numbers is:

$$((z))_{L-j} = \Big(((x))_{L-j} + ((y))_{L-j}\Big) mod\beta$$

$$= \Big(\sum_{i=0}^{m-\nu j} (x_i + y_i) B^{i+(1+\nu)j-m-1}\Big) mod\beta \tag{3.12}$$

The MID has theoretically all the information about the root value, and needs to communicant with all of the input binary digits as indicated by Equation (3.12). However, the CVNS is implemented using analog circuitry, and due to the limited precision of this implementation medium it may not be implemented precisely. To eliminate this problem CVNS correction can be used. This multiple digit CVNS precision enhancement procedure has previously been implemented using a mechanism referred to as Reverse Evolution. Reverse Evolution examines the digit values starting from the LID and is able to detect and remove multiple errors. However, it introduces unnecessary delays into the addition process and has a high hardware cost.

We now propose a new approach called *Truncated Addition* to overcome the limited precision of the analog circuits by adding CVNS values that are in fixed word lengths. This not only relaxes the target requirements on fan-in, but also reduces the system complexity. The original inputs to the CVNS system are digitized, enabling us to perform the addition with some imprecision in the analog circuits and still obtain the correct results without the need to apply the costly RE process.

3.3 Truncated Addition

In the CVNS digit ensemble, the MID circuitry interfaces with the circuitry of all of the input digits, in order to acquire the full information. In fact, the values of the original positional number system can be extracted from the MID, provided that the implementation medium has the appropriate precision to accurately represent the original binary representation.

However, the circuit complexity and the required precision for implementing the more informed digits increases with the binary input word length. To facilitate the implementation of such a system while reducing the analog circuitry, the more informed digits are truncated in the addition process. The CVNS digits can be truncated as long as the approximation does not affect the outcome of the final (integer) computation; hence, the usual need for an error correction process after each two additions is eliminated.

The conversion is performed over a fixed-size group of digits, and not on all of the digits at once. The input binary digits are divided into smaller groups, independent of the input word length. The group size is determined by the implementation requirements and the technology of implementation and is denoted by ψ with $\psi > 1$, a technology dependent parameter limited by the maximum reliable resolution of the implementation environment.

The summation term in (3.12) is split to create two summation terms; the first summation contains ψ terms, while the second term is truncated to provide a suitable approximation level for the addition. If the truncation method is applied, (3.12) is modified to:

$$(\!(z)\!)_{L-j} = \Big(\sum_{i=L_1}^{m-\nu j} (x_i + y_i) B^{i+(1+\nu)j-m-1} + C_{L-j} \Big) mod\beta \tag{3.13}$$

where C_{L-j} is the truncation signal. The value of the lower limit of the summation depends on the applied method.

If $\psi \leq L_1$, the remaining digits should also be grouped into smaller sizes. Methods for processing these binary digits within the limited resolution of the analog environment are explained in the following sections.

Example: Consider the addition of two 8-bit binary values $x_2 = 10100111$ and $y =$

00111011, using the CVNS approach. We will also evaluate the addition based on the proposed method of truncated addition. The results are shown in Table 3.1. The CVNS radix is chosen to be 4; therefore four CVNS digits are required.

Table 3.1: CVNS addition between two binary values

i	3	2	1	0
$(\!(x)\!)_i$	2.609375	2.4375	1.75	3
$(\!(y)\!)_i$	0.921875	3.6875	2.75	3
$(\!(z)\!)_i$	3.53125	2.125	0.5	2
$(\!(z)\!)_i$(Truncated addition)	3.25	2	0.5	2

Each of the CVNS digits of $(\!(z)\!)$ generates two adjacent binary digits. It can be noted from the third and forth row of the Table that the binary outcome for both approaches are the same; however, the required resolution for the last row is reduced.

3.3.1 Determining the Group Size

A large ψ will produce a hardware cost comparable to the original scheme, which defeats the purpose of applying this technique. A very small ψ removes the data overlap between the CVNS digits, where the more informed digits only receive enough information to adjust their value correctly and without receiving full information. The lower index digits may stay the same without receiving any information from higher informed digits.

If the error threshold of the implementation medium is ϵ, the value of ψ is:

$$\psi \leq \frac{log\epsilon}{logB} \tag{3.14}$$

If a quantizer is chosen for the LID digit conversion, this value changes to:

$$\psi \leq \frac{log(2\epsilon)}{logB} \tag{3.15}$$

To guarantee a correct binary result after the CVNS addition is performed, ψ should be chosen such that:

$$\psi \geq \nu + 1 \tag{3.16}$$

For $\psi = \nu$, there is no information overlap between the truncated CVNS digits, and for $\psi \leq \nu - 1$ the information is incomplete.

3.4 Truncation Scheme

The CVNS adder structure in general can be divided into three layers. In the first layer the binary digits are converted into the CVNS values in a fixed word length and the presence of the truncation signals for each digit is detected. These truncation signals are similar to the group carry signals in the binary adders such as carry look-ahead adders. The second layer of the CVNS adder determines the value of the truncation signal C. The last layer of the adder computes the truncated addition as shown by (3.13), and converts the results back to the binary number system.

Equation (3.12) shows the relation between the binary and the CVNS digits in the most redundant form. It should be noted that the CVNS digits that interface with more than ψ binary digits are split into several groups. Therefore, the less informed CVNS digits may not contain more than one group of data, hence are not altered by truncation and stay the same for different adder configurations.

The CVNS digits with indices from 0 up to $\lceil \frac{\psi}{\nu} \rceil$ are left unchanged, and truncation starts for digits with higher indices. Each CVNS digit, is divided into $\lceil \frac{m+1-j\nu}{\psi} \rceil$ groups. The outcome of each of the groups is used only to determine the truncation signal, C, within that digit. The total number of all of the groups, redundant and non-redundant, in the adder is:

$$\sum_{j=0}^{L} \left\lceil \frac{m+1-j\nu}{\psi} \right\rceil \tag{3.17}$$

The above expression reflects the upper boundary of the required gates for full redundancy. However, to save area, some of the gates are used for generating the data for multiple CVNS digits. There are two main grouping methods, whose outcome changes the structural format of the CVNS adder based on the different connectivity requirements. These methods are called *Sliding Groups* and *Uniform Groups*, and differ from each other by the method with which the binary inputs are grouped within each digit, as explained below.

3.4.1 Sliding Groups

This partitioning technique is close to the partitioning scheme used for the only previously reported CVNS adder design [4]. The difference between the two technique is that the data from the lower significant binary digits are not ignored in this design. The starting point for generating groups is always from the most significant binary digit. The details of this scheme are shown in Figure 3.1.

The first layer of the CVNS adder interfaces only with the binary digits which are distributed in the lower groups of the CVNS digits. For example, to generate the MID at the first layer, binary digits with indices from 0 to $L - \psi$ are processed in fixed length groups. The outcome of each group are two signals that are analogous to the carry-propagate and carry-generate signals of fast binary adders. By partitioning the binary input into groups of length ψ, the input data is processed in fixed lengths, and the maximum resolution of each group is B^{ψ}.

Hardware redundancy can be reduced by choosing ψ carefully, thus information generated in the groups of the lower informed digits is also reused for the more informed digits. These requirements can be met when:

$$\psi = i\nu \tag{3.18}$$

where i is an integer value larger than 2.

To reduce the area and to optimally reuse the generated data, (3.18) provides the best results. As can be seen from Figure 3.1, the first repetition starts when $m - 2\psi + 1 =$

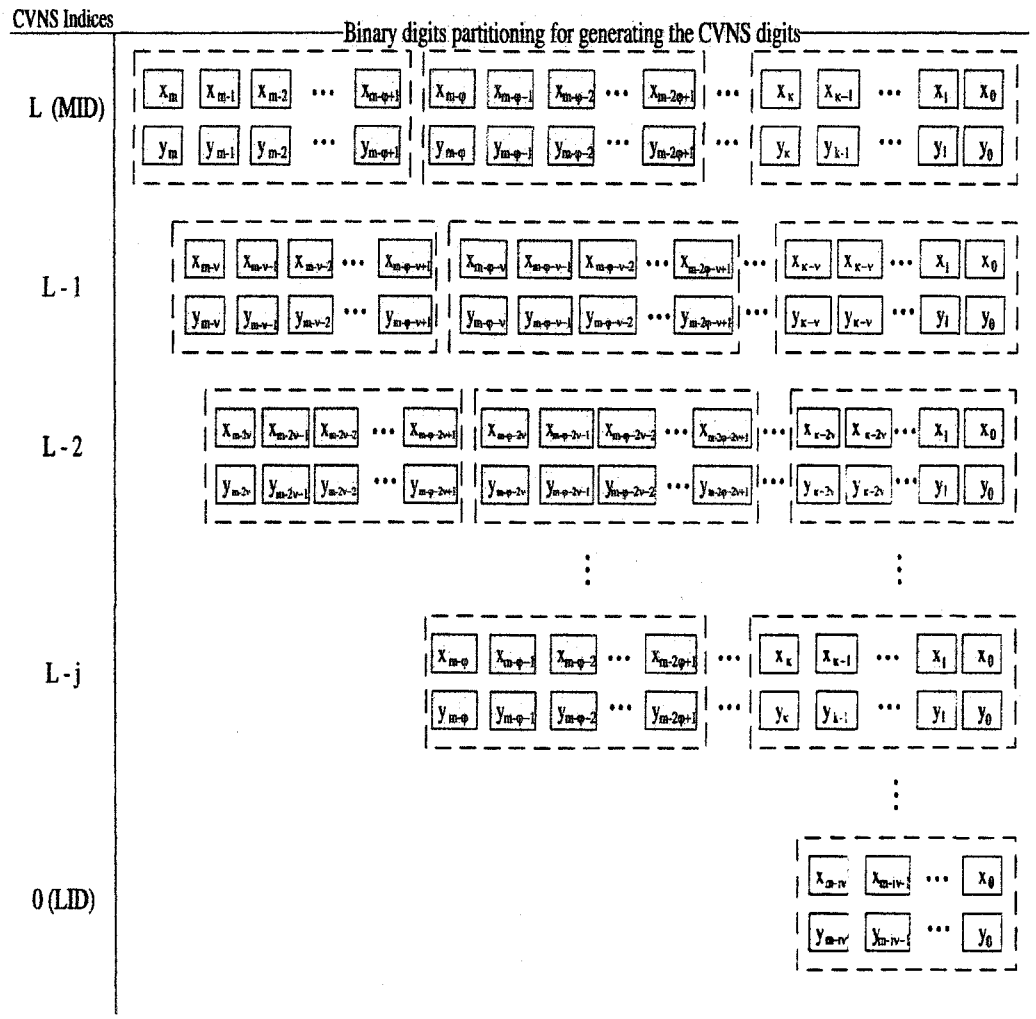


Figure 3.1: Binary input bits partitioning in the CVNS digits based on "sliding groups", showing the redundant and non-redundant groups, where $k = m - \left\lceil \frac{m+1-\psi}{\psi} \right\rceil \psi$ and $0 \leq j \leq L$

$m - j\nu - \psi + 1$, which can be simplified into $\psi = j\nu$. If $\psi = i\nu$, then we ensure that data from some of the groups can be reused. This condition is highly effective in reducing the total number of required gates. The results and expressions in this section are for the general case, and can be simplified if the above condition is satisfied.

The formation of the CVNS digits with indices from the $(L - \lceil \frac{\psi}{\nu} \rceil + 1)$th digit up to the $(L+1)$th digit, generates the full information, which means that all of the other digits can reuse the data which is generated within those digits. The group method reduces the digit-level redundancy among the digits which also results in reduced gate-level redundancy at the implementation level. However, full information of the input binary bits required for proper addition is kept the same as the original CVNS addition. The total number of redundant groups which can be reduced from the total gates given in (3.17), is:

$$G = \sum_{i=\lceil \frac{\psi}{\nu} \rceil}^{L-\lceil \frac{\psi}{\nu} \rceil} \left\lceil \frac{m - (L-i)\nu - \psi + 1}{\psi} \right\rceil \tag{3.19}$$

Figure (3.1) can be used as a map to properly distribute the input binary information within the CVNS digits, and to decide on which groups to generate. The outcomes of each group are:

$$gt_{(p)(q)} = \begin{cases} 1 & \sum_{i=u_{ts}(j)-\psi+1}^{u_{ts}(j)} f_{ts}(i) \geq B \\ 0 & otherwise \end{cases} \tag{3.20}$$

$$rt_{(p)(q)} = \begin{cases} 1 & \sum_{i=u_{ts}(j)-\psi+1}^{u_{ts}(j)} f_{ts}(i) \geq B - B^{-\psi} \\ 0 & otherwise \end{cases} \tag{3.21}$$

where

$$f_{ts}(i) = (x_i + y_i).B^{i-m+(L+1-j)\nu+q\psi-1} \tag{3.22}$$

$$u_{ts}(j) = m - (L-j)\nu - q\psi \tag{3.23}$$

and p, q and j are integers in the range of $1 \leq p \leq \lceil \frac{\psi}{\nu} \rceil$, $(m - \lceil \frac{\psi}{\nu} \rceil + 1) \leq j \leq L$, and $1 \leq q \leq \lceil \frac{m-(L-j)\nu-\psi+1}{\psi} \rceil$ correspondingly; p is dependent on j as follows:

$$\begin{aligned} L - j &= t\left\lceil \frac{\psi}{\nu} \right\rceil + o \\ p &= \left\lceil \frac{\psi}{\nu} \right\rceil - o \end{aligned} \tag{3.24}$$

where integers t and o are the modules and remainders values respectively.

The signals gt and rt, which are the outputs of the first layer of the adder, appear in the CVNS digits periodically, with period $\lceil \frac{\psi}{\nu} \rceil$. The second layer of the CVNS adder combines the above information to generate the truncation signal, C_{L-j}, for the third layer. Therefore, the truncation signal C_{L-j} from (3.13) is obtained recursively from the groups and is equal to:

$$\begin{aligned} C_{L-j} = \beta^{\psi/\nu-1}.(&gt_{(p)(q)} + rt_{(p)(q)}.gt_{(p)(q-1)} + rt_{(p)(q)}.rt_{(p)(q-1)}.gt_{(p)(q-2)} + \cdots \\ &+ rt_{(p)(q)}.rt_{(p)(q-1)}...rt_{(p)(1)}.C_{in}) \end{aligned} \tag{3.25}$$

where $0 \leq j \leq L$, $1 \leq k \leq \left\lceil \frac{m-(L-j)\nu-\psi+1}{\psi} \right\rceil$ and C_{in} is the carry to the adder and p is found from (3.24).

It can be noted from (3.25), that there are at most $\lceil \frac{m+1}{\psi} \rceil - 1$ addition terms, while the last term requires $\lceil \frac{m}{\psi} \rceil$ multiplications. The maximum fan-in of the implementation medium may require that C_{L-j} be computed in two stages, which is the only information that is passed to the last layer. CVNS addition does not require the computation of carry-propagate or carry-ready for individual bits.

At the last layer, final required groups within each CVNS digit are computed and the addition is performed based on (3.13). The value of the lower limit, L_1, in Equation 3.13 for this method is equal to $i = m - \nu j - (\psi - 1)$.

An alternative method for partitioning the binary input digits is given next, which is more efficient in terms of hardware costs.

3.4.2 Uniform Groups

In this method, as shown in Figure (3.2), the binary digit partitioning starts from the least significant binary digit, ensuring that all of the generated information is used efficiently. The number of redundant groups is higher in this format, which means that the required number of groups is lower as follows:

$$G = -\left\lceil \frac{m+1}{\psi} \right\rceil + \sum_{i=\lceil \frac{\psi}{\nu} \rceil}^{L} \left\lceil \frac{m-(L-i)\nu-\psi}{\psi} \right\rceil \tag{3.26}$$

The values of the gt and rt signals are:

$$gt_{(p)} = \begin{cases} 1 & \sum_{i=u_{tu}(p)-\psi+1}^{u_{tu}(p)} f_{tu}(i) \geq B \\ 0 & otherwise \end{cases} \tag{3.27}$$

$$rt_{(p)} = \begin{cases} 1 & \sum_{i=u_{tu}(p)-\psi+1}^{u_{tu}(p)} f_{tu}(i) \geq B - B^{-\psi} \\ 0 & otherwise \end{cases} \tag{3.28}$$

where

$$f_{tu}(i) = (x_i + y_i)B^{i-p\psi-1} \tag{3.29}$$

$$u_{tu}(p) = p\psi - 1 \tag{3.30}$$

and p is an integer in the range of $1 \leq p \leq \lceil \frac{m-\psi+1}{\psi} \rceil$.

The binary outputs of these units are used in the second layer of the CVNS adder for generating the truncation signal. The fact that the truncation signals are binary provides

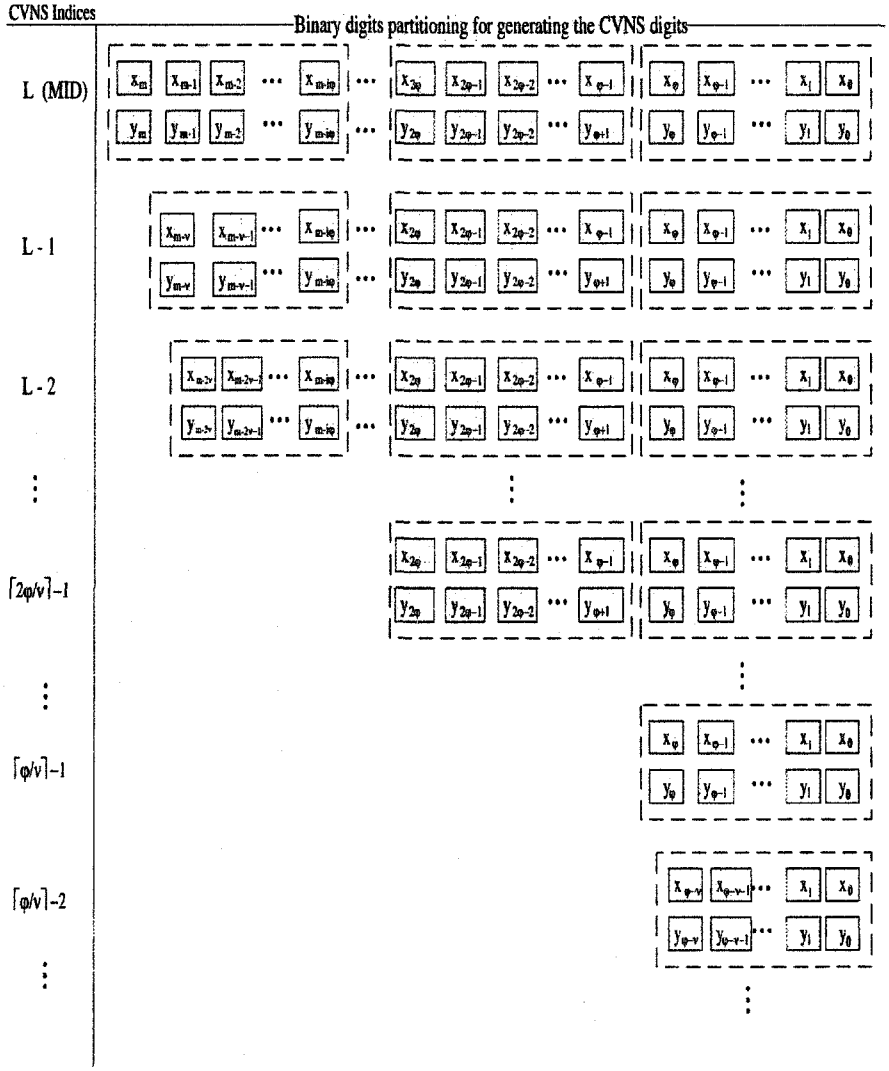


Figure 3.2: Binary input bits partitioning in the CVNS digits based on the "uniform groups", showing the redundant and non-redundant groups.

some extra flexibility in the choice of circuitry (analog or digital). The last group in each of the CVNS digits contains $(m - j\nu)$ down to the $((\lceil \frac{m-j\nu}{\psi} \rceil - 1)\psi)$th binary input digits. The truncated addition result is equal to:

$$((z))_{L-j} = \Big(\sum_{i=(\lceil \frac{m-j\nu}{\psi} \rceil - 1)\psi}^{m-\nu j} (x_i + y_i) B^{i+(1+\nu)j-m-1} + C_{n-j} \Big) mod\beta \tag{3.31}$$

where

$$C_{L-j} = B^{k+j\nu-m-1}.(gt_{(i)} + rt_{(i)}.gt_{(i-1)} + rt_{(i)}.rt_{(i-1)}.gt_{(i-2)} + \ldots + rt_{(i)}.rt_{(i-1)}\ldots rt_{(2)}.rt_{(1)}.C_{in}) \tag{3.32}$$

and $k = (\lceil \frac{m-j\nu+1}{\psi} \rceil - 1)\psi$, and $1 \leq i \leq \lceil \frac{m-j\nu-\psi+1}{\psi} \rceil$.

Both partitioning methods result in a multi-layer CVNS adder which consists mainly of modular reduction gates. For high numbers of input bits, the CVNS adder may require one or more layers for computing the truncation signal, due to fan-in limitations based on the implementation techniques used to compute Equation (3.32). In fact, when $m \leq 2\psi^2 - 1$, the CVNS adder is implemented in only three layers. The second layer of the CVNS adder can be implemented with analog modulus blocks, as well as regular digital blocks. The choice depends on the designer criteria and appropriate technology parameters. For example, low noise applications may take advantage of analog blocks for their inherent low-noise properties.

If the number of gt and rt signals to the second layer are higher than the maximum fan-in of the gates, the term C_{L-j} has to be split into smaller units, each with 2ψ terms. The first and last layers of the CVNS adder may stay the same. Adding more layers increases the delay, which is the same for all forms of two operand adders. The results of Equation (3.13) are now ready to be converted back into binary form.

With these equations a practical CVNS addition process in the CVNS, and a method for interfacing between the conventional positional number system and the CVNS system is developed. In the next section the processing element count and system complexity of the CVNS adder are computed and compared with the previously published CVNS adders [67, 4] and also Threshold Logic (TL) adders.

3.5 Processing Element Count

The CVNS adder can reduce the number of interconnections and achieve area and power savings. In this section, the basic CVNS adder (ADD_b)[67], the Group adder (ADD_g)[4], and our two proposed methods, namely truncation based on the sliding method (ADD_{ts}) and truncation based on the uniform method (ADD_{tu}), are compared. Comparisons are also made between the system complexities in terms of number of required gates, fan-in and fan-out of the system, system resolution and finally gate and system sizes.

The original CVNS method of addition, ADD_b, does not introduce any approximations into the system. This means that the MID requires a fan-in of $2m$ and resolution of $r_b = B^{-m}$. For high numbers of input bits, such a high fan-in and resolution in an analog implementation medium is not feasible. Consequently, the group method, ADD_g, offered a trade-off in terms of reducing the fan-in of the gates and maximum required resolution at the cost of greater operation delay. The ADD_g reduced the maximum fan-in to 2ψ and the required resolution to $r_g = B^{-\psi}$. In both truncation methods, ADD_{ts} and ADD_{tu}, a maximum fan-in of $2\psi + 1$ is required, and the required resolution is equal to $r_{ts,tu} = B^{-\psi}$. Therefore, the group adder and the truncated adder have similar implementation requirements.

The ADD_b adds two radix-2 values in four stages of binary to CVNS conversion, modulus addition, CVNS to digital conversion, and reverse evolution to remove the uncertainty imposed by the implementation medium. The size of the gates directly increases with the gate fan-in and maximum weight in the adder. The conversion unit in the CVNS adder is the most area-consuming unit. The number of required conversion units is equal to $G_{(conv-b)} = \lceil \frac{m+1}{\nu} \rceil$. In case of ADD_g, conversion process requires the same number of gates:

$$G_{(conv-g)} = \left\lceil \frac{m+1}{\nu} \right\rceil \tag{3.33}$$

The ADD_{ts} and ADD_{tu} require:

$$G_{(conv-ts)} = \left\lceil \frac{m+1}{\nu} \right\rceil \times \left\lceil \frac{m+1}{\psi} \right\rceil \tag{3.34}$$

$$G_{(conv-tu)} = \left\lceil \frac{m+1}{\nu} \right\rceil + \left\lceil \frac{m+1}{\psi} \right\rceil \tag{3.35}$$

Complexity theory states that the number of gates indicates the real size that a system may occupy [84]. However, for this assumption to be valid, transistor sizing should be rather uniform, and gate size is assumed to dominate interconnect wiring in a close neighborhood. The theory is valid when the circuit is made up of uniform basic gates such as AND, OR and Inverters with low fan-in. The theory fails when applied to the CVNS, since gate sizing is directly affected by the number of inputs and weight values for binary to CVNS conversion. Therefore, the total number of gates in the CVNS adder may not be a proper measure for the area estimation.

Beame et al. [6] proposed a new method for comparing Threshold Logic gates. In this method the size of a Threshold Logic gate is evaluated as a function of the total number of connections of the gate which is the sum of the number of inputs of all gates in the circuit. However, this method also fails to consider the non-uniform sizing of the transistors. Subsequently, Fernández et al. [64] proposed a new function for the area estimation, which takes into account the transistor sizing. To estimate the area in the CVNS systems, a new parameter is defined here which is close to the suggestion in [64], but much simpler, since the CVNS theory requires positive weight values. The value of this parameter, s, is:

$$s = 2\sum_{i} B^i \tag{3.36}$$

In the CVNS adder, conversion units are the most area-consuming ones, the value of the parameter, s, of each conversion gate for each of the CVNS digits is:

$$s_{(conv-b_j)} = 2\sum_{i=0}^{m-j\nu} B^{m-i-(1+j)\nu+1} \tag{3.37}$$

The total for all of the $(L+1)$ CVNS digits is:

$$s_{(conv-b)} = 2\sum_{j=0}^{L}\sum_{i=0}^{m-j\nu} B^{m-i-(1+j)\nu+1} \tag{3.38}$$

For the group adder, ADD_g this parameter is:

$$s_{(conv-g)} = 2\sum_{j=0}^{\lceil\frac{\psi}{\nu}\rceil-1}\sum_{i=0}^{m-(L-j)\nu} B^i + 2\big(L-\lceil\frac{\psi}{\nu}\rceil+1\big)\sum_{i=0}^{\psi-1} B^i \tag{3.39}$$

The estimated area of the conversion units for the truncated-sliding group adder may be higher than that of the group adder, since in this method the full information redundancy is obtained. The size parameters of conversion gate area of the ADD_{ts}, and ADD_{tu} are:

$$\begin{aligned} s_{(conv-ts)} = {} & 2\sum_{j=L-\lceil\frac{\psi}{\nu}\rceil+1}^{L}\Big(\lceil\frac{m-(L-j)\nu-\psi+1}{\psi}\rceil-1\Big)\sum_{i=0}^{\psi-1} B^i \\ & + 4\Big(\sum_{j=0}^{\lceil\frac{\psi}{\nu}\rceil-1}\sum_{i=0}^{m-(L-j)\nu} B^i\Big) + 2\Big(L-\lceil\frac{\psi}{\nu}\rceil+1\Big)\sum_{i=0}^{\psi-1} B^i \end{aligned} \tag{3.40}$$

$$\begin{aligned} s_{(conv-tu)} = {} & 2\sum_{j=0}^{\lceil\frac{\psi}{\nu}\rceil-1}\sum_{i=0}^{m-(L-j)\nu} B^i + 2\Big(\lceil\frac{m+1}{\psi}\rceil-1\Big)\sum_{i=0}^{\psi-1} B^i \\ & + 2\sum_{j=\lceil\frac{\psi}{\nu}\rceil}^{L}\sum_{i=(l-1)\psi}^{m-(L-j)\nu} B^{i-(l-1)\psi} \end{aligned} \tag{3.41}$$

where l is defined as $\lceil\frac{m-(L-j)\nu+1}{\psi}\rceil$.

At first glance, the sliding group adder area is higher in comparison to the group adder, ADD_g. However, ADD_g also needs to adjust the final results after each two operand addition, which requires additional units thus increasing the area and the delay proportional to the number of the CVNS digits. Based on the given expression in [67] each correction unit requires to compare two neighboring digits and perform the correction by scaling the more

informed digit and comparing it with the less informed digit. Therefore, in total there are ℓ correction units and the size parameter, s_{RE}, which is used to evaluate the size of the reverse evolution process is:

$$G_{(RE-b,g)} = L \tag{3.42}$$

$$s_{RE} = L(B+1) \tag{3.43}$$

This increases the total area required for the ADD_b and ADD_g implementations proportional to the number of CVNS digits. All three types of the CVNS adders require A/D conversion units, hence it is assumed that in comparing the area these parts cancel each other out. Therefore, for area comparisons these parts are not included considering that similar techniques are used for all of them. The main difference between the $\text{ADD}_{b,g}$ and $\text{ADD}_{ts,tu}$ structures is the reverse evolution process.

Delay is another important issue for two operand addition. ADD_b is not feasible for analog implementation due to its high fan-in and resolution requirements. However, if it was implemented, it sums up two binary values in parallel. The original CVNS adder, ADD_b, and the group adder, AAD_g, perform addition in four stages; digital to analog conversion, modulus operations, analog to digital conversion and reverse evolution, where each stage adds its own delay into the signal path. Nevertheless, the final results are required to be adjusted. The main source of error for the AAD_b is the implementation medium, whereas the error in AAD_g is generated by removing the data overlap between the CVNS digits.

$$T_{b,g} = T_{D/A} + T_{mod} + T_{A/D} + (L-1).T_{RE} \tag{3.44}$$

where $T_{D/A}$ is the delay caused by the binary to the CVNS conversion, T_{mod} is the modulus gate delay, $T_{A/D}$ is the CVNS to binary conversion delay, and T_{RE} is the delay from the error correction process (reverse evolution delay).

On the other hand, ADD_{ts} and ADD_{tu} forms do not need to correct the results after each two additions, which drastically reduces the delay of the adder. The delay, D, is independent of the number of input bits into the system, and is equal to:

$$T_{ts,tu} = T_{D/A} + T_{mod} + T_{A/D} \tag{3.45}$$

A case study for a 32-bit, two-operand addition is presented in Table 3.2, using the expressions in this section as an example for all four type of the CVNS adders.

Table 3.2: Comparison between the four different types of the CVNS adder for a 32-bit radix-2 ($\beta = 2$ and $\nu = 1$) addition. ψ is considered 4 and formulas developed in this section are used.

Adder Type	ADD_b[67]	ADD_g[4]	ADD_{ts}	ADD_{tu}
$fan-in$	64	8	9	9
r	2^{32}	2^4	2^4	2^4
G_{conv}	32	32	60	39
s_{conv}	$4(2^{33}-1)$	892	1664	444
G_{RE}	32	32	N/A	N/A
s_{RE}	96	96	N/A	N/A
$s_T = s_{conv} + s_{RE}$	34,359,738,460	988	1,664	444
T	$\propto 34$	$\propto 34$	$\propto 3$	$\propto 3$

The size parameter, s, indicates that the size of the basic and group CVNS adders are higher, and both have more delay which is proportional to the number of input digits. For large word length, these approaches may not be feasible. In fact these approaches do not offer any potential advantages over the truncated adders.

Truncation addition is based on the comparators for detecting carry information within each CVNS digit. Interestingly, Threshold Logic (TL) principles are also based on comparators for detecting individual bit-level and group-level carry-ready and carry-generate signals. Therefore, while CVNS works with continuous values, it is closely related to Threshold Logic. Therefore, in the next section, TL adders are compared with our new CVNS adder designs.

3.5.1 Comparisons Between the TL and CVNS Techniques

Threshold Logic has been proven to be a powerful method for implementing complex functions with far less complexity than conventional digital systems. The fundamental theories of the CVNS are somewhat different to those of Threshold Logic family. However, our earlier assumption, (3.1), for developing a simple interface between the binary and the CVNS systems makes this special class of CVNS numbers similar to Threshold Logic. However, there is one fundamental difference between the two methods; CVNS represents and computes continuous values while Threshold logic uses binary numbers. For adding two binary numbers based on Threshold Logic all of the bit-level and group-level carry-ready and carry-generate signals are computed; in contrast the CVNS adder requires only a few gt and rt signals.

The CVNS adders are compared with the TL adders [65, 83] for 32-bit, two operand adders. In [83] a depth-three adder with polynomially bounded weights (EW) and in [65] optimized two- and three-layer Threshold Logic adders (DCTA2_{TH} and DCTA3_{TH}) designs are given. The maximum fan-in and resolution reflect the limitations imposed by the implementation medium. Table 3.3 shows that within the same environment, CVNS requires less area with lower complexity and interconnection costs.

Table 3.3: Comparison between CVNS and Threshold logic adders for a 32-bit radix-2 ($\beta = 2$ and $\nu = 1$) CVNS adder, with $\psi = 4$.

Adder Type	DCTA2_{TH} [65]	DCTA3_{TH}[65]	EW [83]	ADD_{ts}	ADD_{tu}
$fan-in$	65	8	15	9	9
r	2^{32}	2^4	2^6	2^4	2^4
s_T	$1,717^{10}$	8,642	7556	1664	444
$I(interconnections)$	1,216	896	1504	88	46

These results are extended in Table 3.4, to compare the area of our CVNS adders with the Threshold Logic adders for various input word lengths. In this table the area required

for A/D conversion is added onto the total area of the CVNS adders.

Table 3.4: Comparison between the Threshold logic and the CVNS adders in terms of gate sizing for various word lengths.

	size (s)					
m	DCTA2$_{TH}$[65]	DCTA3$_{TH}$[65]	EW [83]	ADD$_{ts(\nu=1)}$	ADD$_{tu(\nu=1)}$	ADD$_{ts(\nu=2)}$
8	1068	254	416	352	142	114
16	262236	722	1,196	630	314	258
32	$1.7E10$	4538	7556	1696	604	546
64	$7.3E+19$	24482	40796	3648	1294	1122
128	$1.3E+39$	376694	737048	7672	2670	2274

The number of interconnections in the CVNS adder is double the sum of the number of required gates at the first and second layers. For the sliding partitioning method, this value is:

$$I_{ts} = 2\Big(\lceil\frac{m+1}{\nu}\rceil \times \lceil\frac{m+1}{\psi}\rceil\Big) + (L+1) \tag{3.46}$$

and for uniform partitioning this is equal to:

$$I_{tu} = 2\Big(\lceil\frac{m+1}{\nu}\rceil + \lceil\frac{m+1}{\psi}\rceil\Big) + (L+1) \tag{3.47}$$

Table 3.5 shows the theoretical measures of the required interconnections of the main types of Threshold Logic adders, as well as the two CVNS adders. It shows that the uniform group CVNS adder requires fewer interconnections. By increasing the radix of the CVNS, fewer continuous digits are required, hence interconnection cost is reduced. In the next section, the design of a 16-bit radix-4 CVNS adder based on the Uniform Group truncated addition method is demonstrated.

Table 3.5: Number of Interconnections required for the Threshold logic adder and the CVNS adder ($\psi = 4$) for different word lengths.

	Interconnection (I)					
m	DCTA2$_{TH}$ [65]	DCTA3$_{TH}$[65]	EW[83]	ADD$_{ts(\nu=1)}$	ADD$_{tu(\nu=1)}$	ADD$_{ts(\nu=2)}$
8	112	136	256	40	28	16
16	352	320	592	144	56	32
32	1,216	832	1,504	544	112	64
64	4,480	2,048	3,648	2,112	224	128
128	17,152	5,376	9,600	8,320	448	256
256	67,072	14,336	24,832	33,024	896	512

3.6 CVNS 16-bit Radix-4 Adder

In this section, the actual design of a current-mode radix-4 CVNS adder for 16-bit binary addition is developed. The CVNS adder consists of three main layers:

- Binary to CVNS conversion
- Truncation signal generation
- Addition of the CVNS digits and conversion form the CVNS back to binary

The Uniform Group method is used to reduce the hardware redundancy, while full information redundancy based on the CVNS theory is still obtained. The radix of the CVNS is chosen to be 4, in order to decrease the number of the CVNS digits. The optimum value of ψ is chosen based on the implementation technology, proper partitioning scheme, and the CVNS radix value. The implementation technology chosen is a $0.18\mu m$ TSMC CMOS process with $1.8V$ power supply; and $\psi = 4$ is an appropriate choice for this target technology. Since CVNS radix is 4 ($\nu = 2$), there are only 8 CVNS digits ($L + 1 = 8$).

The truncation signals within the groups are generated first. The CVNS adder operates in current mode, hence the two CVNS digits are simply added through a summing node. The general configuration of a radix-4 CVNS adder is shown in Figure 3.3, where the gray rectangles represent the blocks that are used for computing the truncation signals, $C = Tr(.)$. The modular reduction circuit is the main block in the design of the CVNS adder, and is explained in the next section.

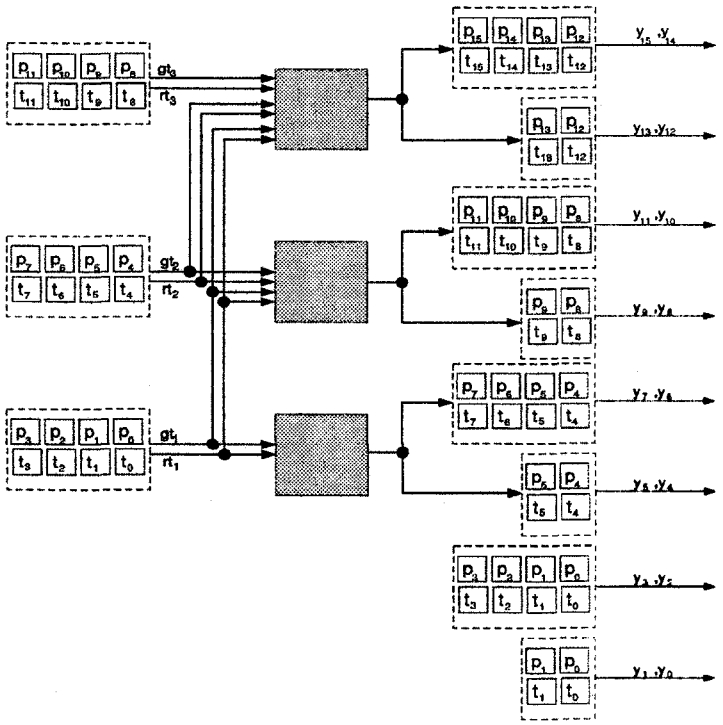


Figure 3.3: General configuration of the 16-bit Radix-4 CVNS adder.

3.7 Modular Reduction Circuit

The central circuit of a CVNS adder is the modular reduction circuit, which is essentially a current comparator. In this design, a differential current comparator is applied which increases the noise immunity of the adder. Generally, the input to one of the transistors of

the input pair is the input current, while the other input transistor is always ON and set to represent the threshold value. This method requires large transistors to always conduct current, which increases the static power consumption.

In our approach, the input current i is compared with its complement value $i' = (1 - 2^{-\psi})\beta - i$, where $\psi = 4$. This not only reduces the power requirements, but also decreases the area by eliminating the large transistors required at the input pair. This also leads to a better noise margin and a higher gate speed due to the fact that inputs have opposite transitions [8]. Figure 3.4 shows the CVNS current mode modular reduction operation, which generates two binary outputs for each CVNS digit.

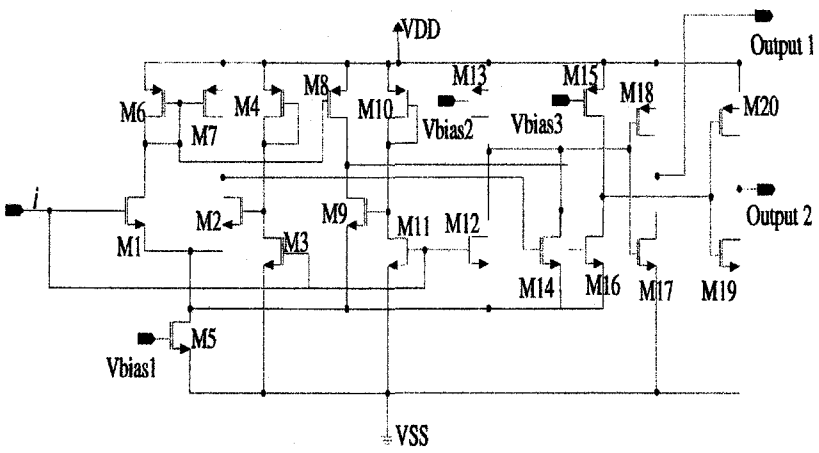


Figure 3.4: Current mode modular reduction and A/D conversion circuit.

The current, i, is the equivalent analog value of the binary inputs, which is compared with its complement value at the differential pair, through transistors (M1-M7). The voltage at the M2-M7 branch drops for values less than 2 and rises for values higher or equal to 2. This branch also turns transistor M14 ON or OFF. There are two comparators in this gate, the second one, formed by transistors M8 through M11, is used for generating the less significant bit of the CVNS adder output.

Transistors M12-M20 are used for binary output generation. The biasings of M13 and M15 cause the gates of M17-M18 to go HIGH for a logic 1 and LOW for a logic 0. The

output of this inverter is the less significant binary output digit. The more significant binary output digit is generated at the output of transistors M19-M20. The layout of the adder was dynamically tested and simulated with alternating inputs and taking into account all layout parasitics. The layout occupies an area of $2,300\mu m^2$ and is shown in Figure 3.5. The CVNS adder has a maximum delay of 460ps for adding two binary values of 0000, 0000, 0000, 0001 and 1111, 1111, 1111, 1111 where the carry has to propagate through all the digits.

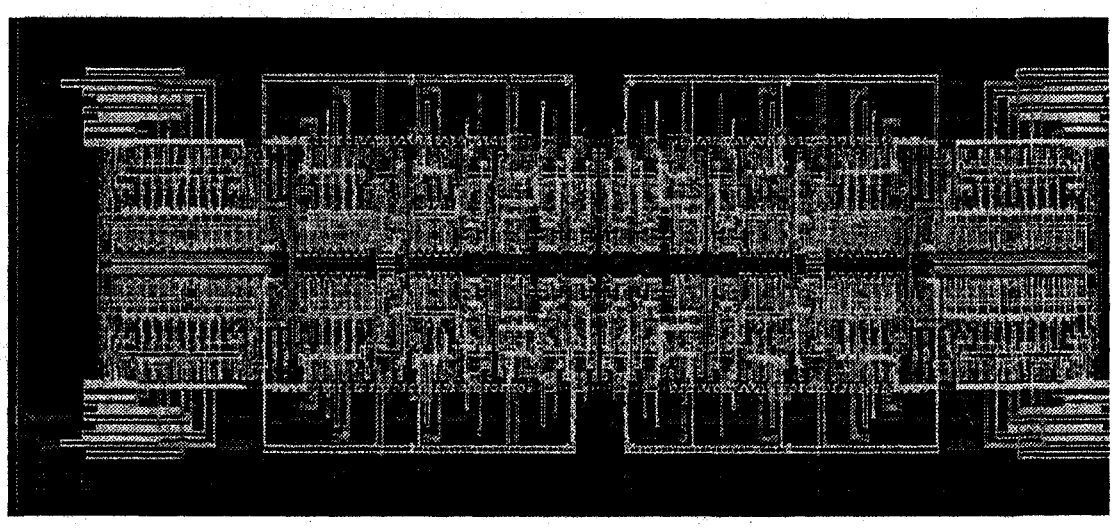


Figure 3.5: Layout of 16 bit radix-4 CVNS adder in TSMC $0.18\mu m$ CMOS process.

Analytical delay models are used to compare different styles. It has been shown that the logical effort model [30] can provide a rapid comparison between a variety of adders. For comparison purposes, the delay of several standard adders in terms of minimum-sized fanout-of-four (FO4) inverter delays are given in Table 3.6. The fanout-of-four delay in TSMC $0.18\mu m$ process is around 90 ps, which means that the worst case delay of our design is roughly around 5.12 FO4. This result shows that the continuous CVNS adder can achieve a speed comparable to the other high speed adder designs. A comparison against the only other published CVNS adder design [4] is shown in Table 3.7. The work in [4] was implemented with a $0.35\mu m$ CMOS process, therefore the results are scaled down for comparison purposes.

This may not be a fair comparison, since both designs are analog and analog area does not scale as uniformly as digital designs, but it still shows strong advantages in applying the truncation method for the CVNS addition.

Table 3.6: Delay comparison of several 16-bit adders in terms of fanout-of-four

Adder Architecture	Number of Fo4 delays
Ripple Carry	26.6
Conditional Sum	13
Carry Increment [79, 93]	15.7
Knowles [40]	9.9
Helper 1a [30]	9.7
Helper 1b [30]	10.1
Helper 1.5 [30]	9.7
Helper 2 [30]	9.7
Radix-4 CVNS	5.12

Table 3.7: Comparison between two CVNS adders

	CVNS Adder [67]	Our design
Area	10,050 μm^2	2,300 μm^2
Delay	49,885.7 ps	460 ps
Word length	8 bits	16 bits

The adder presented here is the first practical design of adders based on the CVNS theory. By comparing to standard binary designs we have shown that the CVNS can open up new design approaches for efficient arithmetic units. The work reported in this dissertation takes advantage of the maximum medium resolution for implementing the adder, which contrasts

with the previous suggestion to use lower resolution for the addition and arithmetic, and use higher resolutions in the error correction process. By avoiding using error correction during the two operand addition, the area and worst-case delay has been decreased.

3.8 System and Cross Talk Noise

Switching noise is one of the major problems in VLSI designs. In a typical CMOS design, when many gates change state, a large cumulative current is drawn from the power supply. This in turn results in voltage fluctuations in the power supply distribution network in the chip. Moreover, parasitic capacitances between the substrate and transistors, inject current into the substrate due to the digital switching activity. The amount of this current is dependent to the slew rate of the switching voltage and total parasitic capacitances, which is referred to as the cross talk noise. As the operating frequency increases, the substrate noise increases as well. Therefore, a low-noise system should have minimum current sparks and slew rate on the supply rails.

The CVNS reduces the system noise because of the its current mode operations which have almost constant supply current. Current mode operations reduce the supply current variations and, hence, reduce switching noise. The supply current stays almost constant regardless of the binary inputs into the system. Moreover, in the CVNS adder, switching between different states is controlled and occurs with minimum voltage drop. In digital logic, by contrast, the output of logic gates switches rapidly from rail to rail. The idea of using analog circuitry for reducing system and cross talk noise is not new, and there have been previous attempts to implement digital arithmetic with analog circuity [36, 35] .

Two 1-bit full adders based on the CVNS methodology and regular complementary CMOS with pMOS pull-up and nMOS pull-down transistors were designed and simulated and the power supply variations for both systems were monitored. By taking the worst case time domain values of the transient analysis, we can judge the relative performance of the CVNS full adder against a complementary CMOS full adder. The CVNS full adder exhibits

an almost constant current as expected, while the binary full adder has high current sparks which contributes to its high switching and cross talk noise. The maximum current slops for the CVNS full adder is 1.468A/ps and for the binary adder is 3.72A/ps.

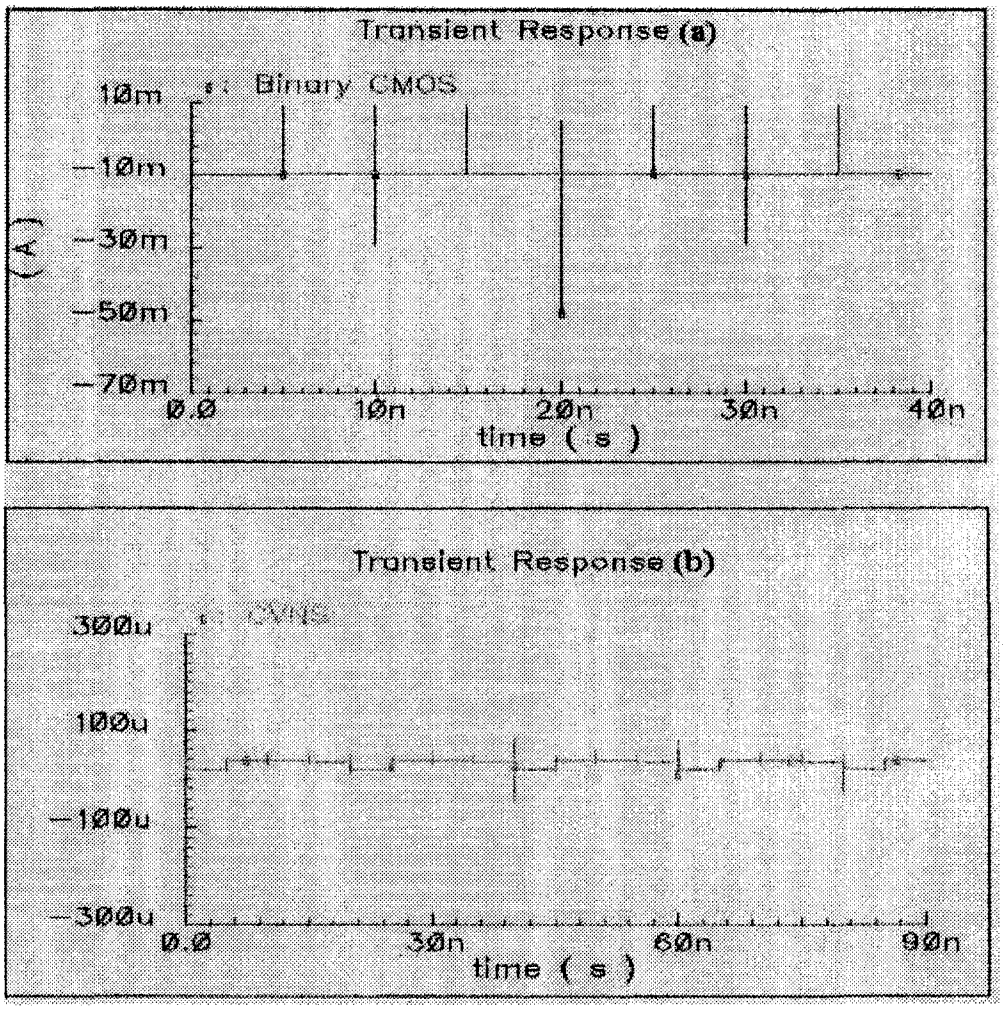


Figure 3.6: Instantaneous current drawn from the power supply (a) Complementary CMOS full adder (b) CVNS full adder.

The adder presented here is among the first designs based on the CVNS theory. By comparing to standard binary designs we have shown that the CVNS can open up new design approaches for efficient arithmetic units. The work reported here has taken advantage of

the maximum resolution to implement the adder, while in previous CVNS adder designs the maximum resolution was used in the error correction process. By avoiding the need for error correction methods during the two operand addition, the area and worst-case delay has been decreased.

3.9 Summary

The CVNS theory has been explored for two operand binary addition. The CVNS theory offers carry-free addition; however, its implementation faces challenges due to the implementation medium. These factors are in contrast with the advantages that the CVNS can offer (analog implementation). Other previously reported methods such as the group method, also result in more complex systems, with higher delay.

The two new approaches of Sliding Groups and Uniform Groups have been introduced and evaluated in this chapter. These approaches resolve the implementation problems of the CVNS adder for large word length in analog circuitry. Moreover, the complexity of these methods are compared with earlier versions of the CVNS adder, and we have shown that both methods result in better structures. Specifically, the uniform group method not only has fewer interconnections and requires less area, but also achieves full CVNS information redundancy.

This method has been applied for a two operand binary addition fully designed in a target CMOS technology, and its performance has been compared with other adder forms. New circuits were proposed for the CVNS adder, which offer higher performance. Based on comparing the performance of the implementation, the CVNS was shown to offer a computational speed comparable to high speed adders but built with analog building blocks, while offering other potential advantages such as reduced system and cross-talk noise.

These methods are proposed for unsigned addition of two binary values; but they can easily be modified to facilitate addition between two binary signed or fractional values.

The adder proposed in this chapter can successfully be used as the basic building block

for implementing reconfigurable adders, tree multipliers, and array multipliers.

Chapter 4

CVNS Reconfigurable Adder

4.1 Introduction

To design efficient signal processing units for multimedia signal processing, reconfigurable adders are required, capable of processing data with varying word lengths without adding too much to the design complexity and overhead circuitry. An efficient adder design is critical for the development of the reconfigurable architectures, and it usually can add one 64-bit, two 32-bit, four 16-bit or eight 8-bit operations [24, 23, 61, 60, 20, 78, 14, 57, 86, 45]. Generally, adding the reconfigurability property into the adder, increases the cost of implementation in terms of worst case delay, and power consumption [24, 60, 20].

Typically, to speed up the addition operation, summation terms are computed for both input carries of zero and one, and once the correct carry is determined, the proper signal then propagates through multiplexers in the adder.

The CVNS offers an efficient method for reducing the cost of reconfigurable adder implementation, where summation is performed digit wise with fewer number of interconnections. The adder described here has a modular architecture, and is created by replicas of the same

block in a hierarchical design level. The 64-bit adder is implemented by cascading four similar 16-bit radix-2 CVNS adders. Performance of the CVNS adder is evaluated in terms of worst-case delay, energy dissipation and area.

4.2 System Level design of the Reconfigurable 64-Bit Adder

CVNS is a continuous number system, therefore it is implemented by VLSI analog circuits. To reduce the design time of the adder, a regular architecture has been employed throughout the 64-bit adder. The 64-bit reconfigurable adder is divided into four 16-bit adders, and each adder is divided to 4 uniform blocks. Inputs and outputs of the system are in binary form. In regular mode of operation, the system performs four parallel 16-bit additions.

To change the mode of operation of the 64-bit adder two signals are used namely; *part1* and *part2* as shown in Table 4.1.

Table 4.1: Adder operation for different word lengths, controlled by *part*1 and *part*2 signals

*part*1	*part*2	Adder Configuration
0	0	Byte (8-bit)
0	1	Half-Word (16-bit)
1	0	Word (32-bit)
1	1	Double word (64-bit)

To partition the 16-bit adder whenever 8-bit operation is required, control signal breaks down the size of each of the 16-bit adders to 8 bits. This mode of operation is controlled by signal *ctrl*8 which is generated as follows:

$$ctrl8 \equiv (part1 \vee part2) \tag{4.1}$$

A pair of the 16-bit adders is combined to perform 32-bit addition. The information is exchanged between the two adders, from the less informed adder to the more informed adder, not only in the 32-bit but also in the 64-bit mode of operation. Therefore, control signal can be changed from $ctrl32 \equiv part1 \wedge part2'$ to simplify only $part1$. By examining the Table 4.1 the $ctrl32$ is equivalent to:

$$ctrl32 \equiv part2 \tag{4.2}$$

All four 16-bit adders are combined to work as a 64-bit adder when both $part1$ and $part2$ signals are one. Therefore, control signal for this configuration is setup as follows:

$$ctrl64 \equiv (part1 \wedge part2) \tag{4.3}$$

These control signal are used to adjust the size and resolution of the adder on-demand. In the following sections a mathematical analysis of each 16 bit adder is given, followed by an explanation of how cascading these adders results in a reconfigurable 64-bit adder.

4.3 Mathematical Analysis

In this section a mathematical analysis of the 64-bit adder is given. We first start by looking at the structure of each 16-bit adder, and drive the mathematician expressions for truncation signals outside each adder. Then, input carry signal to each of the adders is generated from these signals. In general, the CVNS reconfigurable adder has a simple design with minimal interconnection for such systems.

4.3.1 Radix-2 16-Bit Addition

A 16-bit CVNS adder converts binary data to CVNS, performs addition on the CVNS digits, and converts the final results back to binary. Binary numbers are converted to the CVNS

representation very easily if conditions in (3.1) and (3.2) are satisfied, which are repeated here for convenience.

$$B^{m+1} = \beta^{L+1} = M$$

and

$$B^{\nu} = \beta$$

For illustration purposes, the CVNS radix is chosen to be $\beta = 2$. The reason for choosing a low radix for this design will be explained later. The relationship between 16-bit binary digits, inputs of each adder, x_{i+t}, and the real value they represent, x^t, is:

$$x^t = \sum_{i=0}^{15} x_{i+t}.B^i \tag{4.4}$$

where $B = 2$ is the binary number system radix, and $t = 0, 16, 32, 48$.

Based on the method given in chapter 3, a direct relationship between the binary and the corresponding CVNS digits is obtained as follows:

$$((x))_{15-j+t} = \sum_{i=0}^{15-j} x_{i+t}.B^{i+j-15} \quad 0 \leq j \leq 15 \tag{4.5}$$

The above equation is pre-congruent, meaning that the modular reduction operator, which is required for the CVNS digit generation as expressed by (2.5), is not included here.

The digit-wise summation of two CVNS numbers, $((x))$ and $((y))$ is:

$$\begin{aligned} ((z))_{15-j+t} &= ((x))_{15-j+t} + ((y))_{15-j+t} \\ &= \Big(\sum_{i=0}^{15-j} (x_{i+t} + y_{i+t}) B^{i+j-15} \Big) mod\beta \end{aligned} \tag{4.6}$$

A CVNS digit as shown by Equation (2.9) is composed of two parts; an integer part and a non-integer part, which is shared with its less informed digits. By applying Equation (2.9) in the above expression, the summation term can be expressed as follows:

$$((z))_{15-j+t} = \left(x_{15-j+t} + y_{15-j+t} + 2^{-1} \sum_{i=0}^{14-j+t} (x_{i+t} + y_{i+t}) B^{i+j-15}\right) mod\beta \tag{4.7}$$

or

$$((z))_{15-j+t} = \left(x_{15-j+t} + y_{15-j+t} + 2^{-1}((z))_{14-j+t}\right) mod\beta \tag{4.8}$$

The term $((z))_{15-j+t}$ is in the CVNS form and has to be converted back into binary form. For values of $((z))_{15-j+t} \in [0,1)$, the binary outcome digit, z_{15-j+t}, is equal to 0, and for $((z))_{15-j+t} \in [1,2)$ it is equal to 1. At this stage, the low radix of the CVNS allows us to remove the modular reduction operation ($mod2$) and replace the CVNS to binary conversion with a simple XOR operation, and to generate the binary outcome of the adder as follows:

$$z_{15-j+t} = x_{15-j+t} \oplus y_{15-j+t} \oplus p_{15-j+t} \tag{4.9}$$

where $\oplus$ denotes the logical XOR function, and

$$p_{15-j+t} = \begin{cases} 1 & \text{if } 0 \leq ((z))_{14-j+t} < 1 \\ 0 & \text{if } 1 \leq ((z))_{14-j+t} < 2 \end{cases} \tag{4.10}$$

To implement the term $((z))_{14-j+t}$ a high resolution analog environment, equivalent to 14 bits is required. If the implementation environment distinguishes only B^{ψ} different levels, the above equation has to be changed to perform the summation over a fixed sized group of bits of length ψ with $\psi > 1$. This parameter is technology dependent, limited by the maximum reliable resolution of the environment. We have chosen it to be equal to 4, which can not only be easily implemented by current-mode analog circuits in our target technology, but also simplifies the partitioning scheme. Truncated addition over a group of digits for 16-bit summation with $\psi = 4$ is:

$$((z))_{14-j+t} = \Big(\sum_{i=4\lceil \frac{14-j+t}{4} \rceil}^{14-j+t} (x_{i+t} + y_{i+t}) B^{i+j-14} + B^{4\lceil \frac{15-j}{4} \rceil - 19+j} Tr_{14-j+t} \Big) mod\beta \qquad (4.11)$$

where Tr_{14-j+t} is called the truncation signal and is equal to:

$$Tr_{14-j+t} = \Big(gt_{(k+t)} + rt_{(k+t)} gt_{(k+t-1)} + \cdots + rt_{(k+t)} rt_{(k+t-1)} \cdots rt_{(2+t)} rt_{(1+t)} C_{in_t} \Big) \qquad (4.12)$$

where gt and rt signals are:

$$gt_{(k+t)} = \begin{cases} 1 & \text{if } \sum_{i=4k-4}^{4k-1} (x_{i+t} + y_{i+t}) B^{i-4k+1} \geq 2 \\ 0 & \text{otherwise} \end{cases} \qquad (4.13)$$

and

$$rt_{(k+t)} = \begin{cases} 1 & \text{if } \sum_{i=4k-4}^{4k-1} (x_{i+t} + y_{i+t}) B^{i-4k+1} \geq 1.875 \\ 0 & \text{otherwise} \end{cases} \qquad (4.14)$$

and k is an integer equal to $\lceil \frac{11-j}{4} \rceil$ and $t = 0, 16, 32, 48$.

The 64-bit adder is not entirely CVNS and rather it is a mix of both classical binary circuits, such as XOR for generating the output, and CVNS style circuits for evaluating terms such as (6.11) and (6.12). Analog circuits in this design are the front circuits and are used for processing a group of input bits with higher speed and fewer interconnections. These analog blocks detect the existence of the truncation signals within a group of binary inputs. Digital circuits are at the output stage of each adder and provide the required driving capability for various interconnection loadings in different adder configurations.

Equation (4.12) indicates that only a limited number of truncation signals are required within the 16-bit adder. Because of the low number of interconnections, control and partitioning of the adder is performed with lower complexity. The number of truncation signals is dependent on the chosen length of the groups. Based on the chosen group length, in this design ($\psi = 4$), in each of the 16-bit adders, three truncation signals are generated as follows:

$$Tr_{t+4:t+7} = gt_{(1+t)} + rt_{(1+t)}Cin_t \tag{4.15}$$

$$Tr_{t+8:t+11} = ctrl8.in_{8+t} + ctrl8'\left(gt_{(2+t)} + rt_{(2+t)}gt_{(1+t)} + rt_{(2+t)}rt_{(1+t)}Cin_t\right) \tag{4.16}$$

$$Tr_{t+12:t+15} = gt_{(3+t)} + rt_{(3+t)}\Big(ctrl8.in_{8+t} + ctrl8'\left(gt_{(2+t)} + rt_{(2+t)}gt_{(1+t)} + rt_{(2+t)}rt_{(1+t)}Cin_t\right)\Big) \tag{4.17}$$

or

$$Tr_{t+12:t+15} = gt_{(3+t)} + rt_{(3+t)}Tr_{t+8:t+11} \tag{4.18}$$

where C_{in_t} is the carry input to each of the adders. The expression for this signal is derived in the next section.

The three truncation signals, given by the above equations, are the only signals that are passed from the second layer of the adder to the third layer. The carry out of the adder is generated in the same style. Figure 4.1 shows the gate level design of a 16-bit radix-2 CVNS adder.

CVNS addition in this chapter is performed in three main stages which are: evaluating *gt* and *rt* signals from each group of input bits, as shown by gray blocks; combining the information for generating the truncation signals, as shown by the dashed blocks; and the final addition.

4.3.2 Radix-2 64-Bit Addition

The reconfigurable 64-bit adder is generated by cascading four 16-bit radix-2 CVNS adders. This CVNS adder has a uniform design, making it suitable for reconfigurable media processing and SoC applications. In this adder, information is generated locally for addition, hence the number of required interconnections is reduced. Between the four adders, there are eight truncation signals, which are similar to the *gt* and *rt* signals inside each adder, and are as follows:

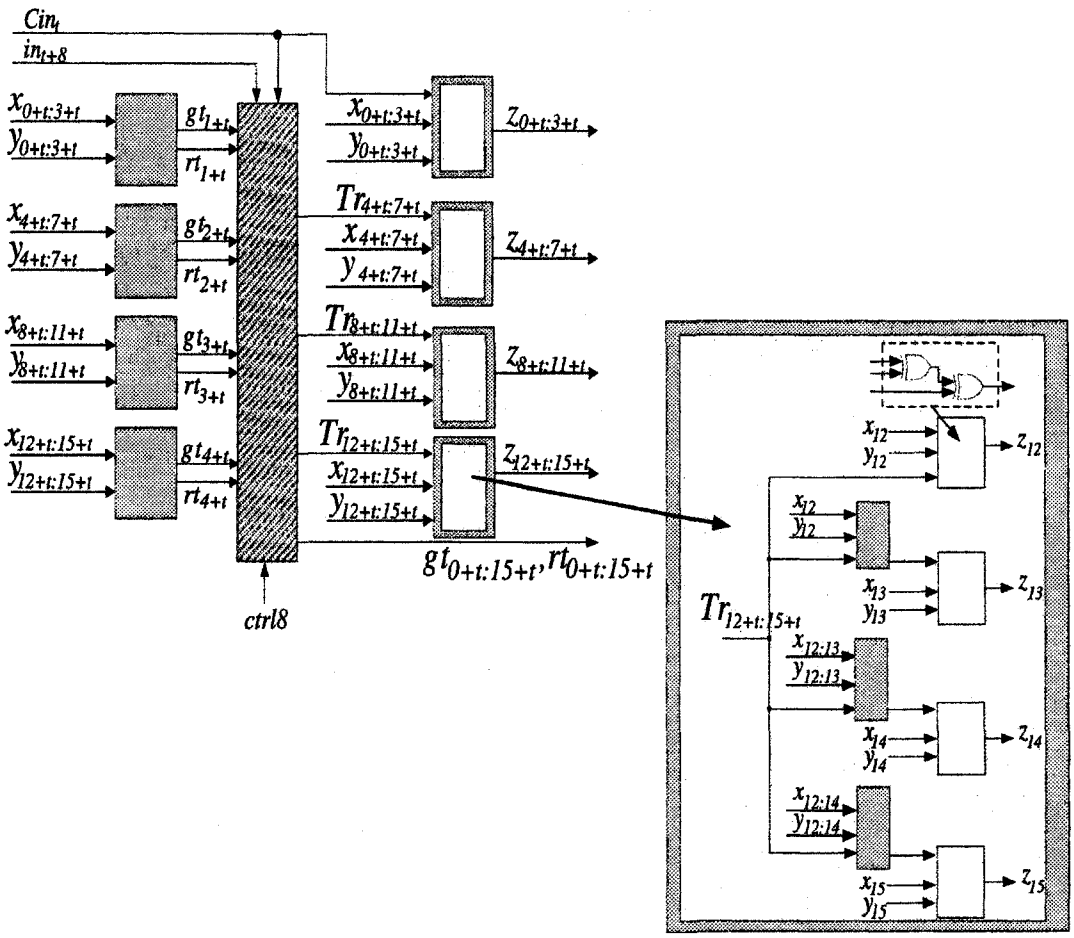


Figure 4.1: Gate level design of each 16-bit adder

$$gt_{(t:t+15)} = gt_{(t+12:t+15)} + rt_{(t+12:t+15)}gt_{(t+8:t+11)} + rt_{(t+12:t+15)}rt_{(t+8:t+11)}gt_{(t+4:t+7)} + rt_{(t+12:t+15)}rt_{(t+8:t+11)}rt_{(t+4:t+7)}.gt_{t:t+3} \quad (4.19)$$

$$rt_{(t:t+15)} = rt_{(t+12:t+15)}rt_{(t+8:t+11)}rt_{(t+4:t+7)}rt_{(t:t+3)} \quad (4.20)$$

where $t = 0, 16, 32, 48$.

The propagation of these signals is controlled by $ctrl32$ and $ctrl64$. The input carry into each of the 16-bit adders, which has been shown in (4.11) as Cin_t, is as follows:

$$Cin_{16} = ctrl32'(gt_{(0:15)} + rt_{(0:15)}.in_0) + ctrl32.in_{16} \quad (4.21)$$

$$Cin_{32} = ctrl64'.in_{32} + ctrl64\big(gt_{(16:31)} + rt_{(6:31)}gt_{(0:15)} + rt_{(16:31)}rt_{(0:15)}.in_0\big) \quad (4.22)$$

$$Cin_{48} = ctrl32'.in_{48} + ctrl32\Big(gt_{(32:47)} + rt_{(32:47)}(ctrl64'.in_{32} + ctrl64(gt_{(16:31)} + rt_{(16:31)}gt_{(0:15)} + rt_{(16:31)}rt_{(0:15)}.in_0)\Big) \quad (4.23)$$

Figure 4.2 shows the general configuration of the 64-bit adder. Expressions for defining the general configuration of the system were derived in this section. In this figure, the numbers within the dashed blocks indicate the equations used, and configuration of the white blocks (indicating a 16-bit adder) have been presented in Figure 4.1.

4.4 Current-Mode Circuits

Differential style analog circuits were used for implementing the CVNS adder. These types of circuits increase the noise immunity of the analog circuits in the mixed signal design. Each gate was designed and sized to reduce power requirements.

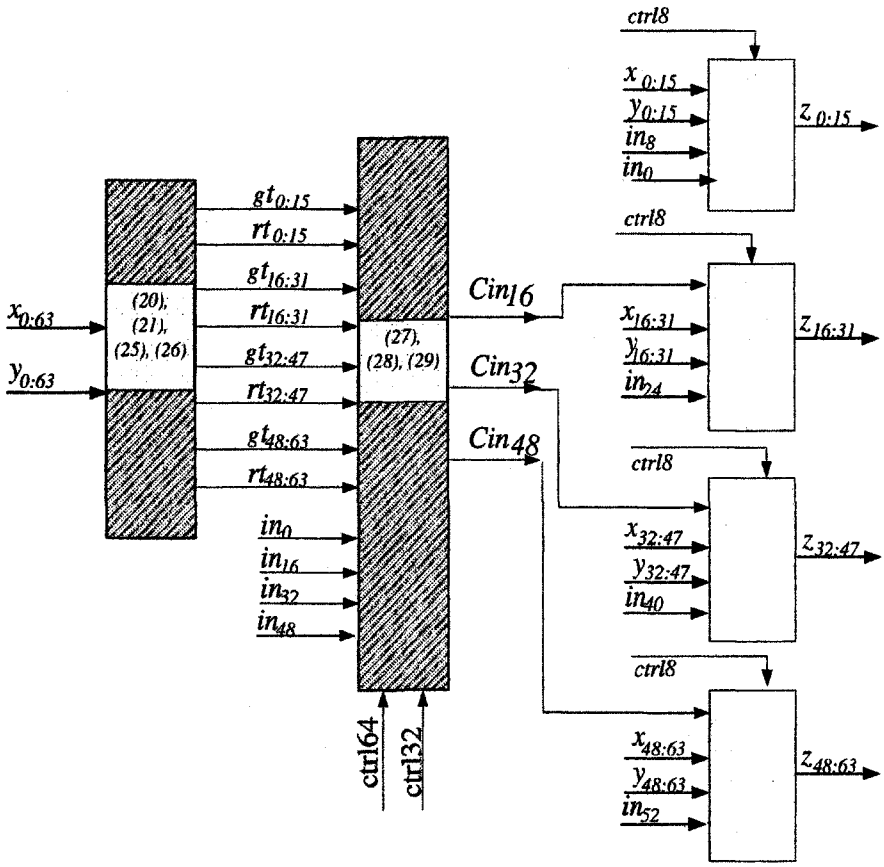


Figure 4.2: General configuration of the 64-bit adder

Circuit operations are in current mode, in order to achieve a low noise, low power and high speed adder. For example, the addition of the CVNS digits are performed by hardwiring the two values in a summation node. One example is shown in Figure 4.3. Binary digits are converted into the CVNS using a current mode DAC (Digital-to-Analog-Converter) and are added at the input of a PMOS current comparator (M20-M21). The comparator threshold is data dependent, meaning that data is compared with its complement value $(1.875 - i)$. The complement value of the current is generated by the $(M22 - M23)$ transistor pair. The output of the comparator goes high when the added current is higher than the radix. These gates are used in the first and last stage of the adder. If it is at the first stage of the adder, the outcome is a *gr* or *rt* signal of that group of bits. Outputs of the gate at the third stage are sent to the XOR gates to generate the final results.

In this type of circuit, the major source of power consumption is static, and is directly related to the biasing current [31]. In this configuration the required biasing voltage is low, hence the constant current source generated by (M1-M3) is also low.

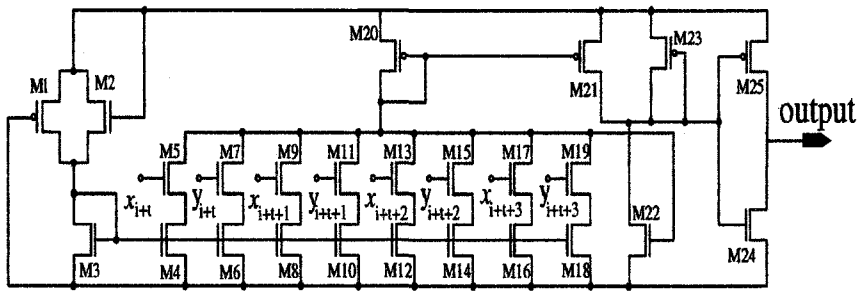


Figure 4.3: The modular reduction circuit, and additional operations including the binary to CVNS conversion and current mode CVNS digits addition

At the truncation signal path, the second layer of the each of the 16-bit adders, signals are digital. Therefore, digital style circuits can be used in the implementation. Figure 4.4 shows the path for generating the truncation signal of the most informed group within the 16-bit adder along with the control signal, *ctrl*8. When this signal is LOW, the adder is in normal mode. A HIGH signal indicates that the adder needs to be reduces to 8-bit, which is

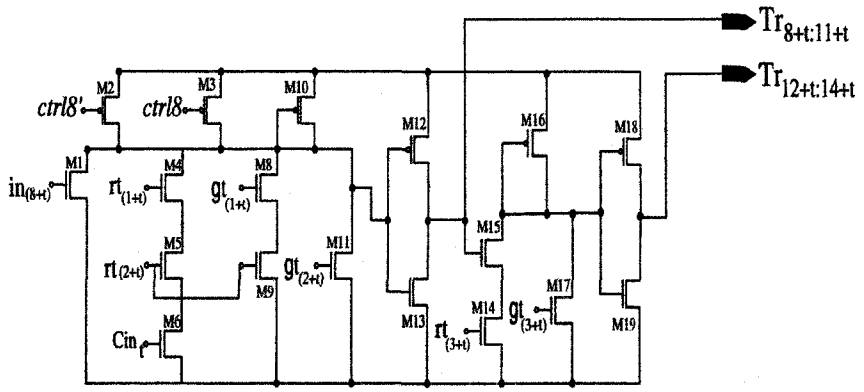


Figure 4.4: Module for generating the truncation signal to the more informed group, controlled by the ctrl8 for 16- or 8-bit operations

performed by a large PMOS transistor which takes the input of the inverter HIGH; hence, a 0 is generated and prevents passing of the lower order group data.

4.5 Results

The adder was designed and simulated using process parameters from in the TSMC CMOS $0.18\mu m$ with 1.8V supply power, and was compared with several non-reconfigurable and reconfigurable digital adders [24, 61, 60, 20, 45, 55] as shown in Table 4.2. Figure 4.5 shows the delay of the adder of several outputs. The efficiency of the adders is not only evaluated in terms of the worst case delay, but also by power consumption and area. It should be noted than the efficiency of the arithmetic units is measured in terms of power-delay product, or energy-delay product [26, 27].

In [24] a Brent-Kung Carry Look Ahead (BKCLA) adder is proposed. The input digits are partitioned into groups of length 4, and configuration is controlled by two control signals. In [61] a Reconfigurable Carry Skip Adder (RCSA) and in [60] a Static Reduced Area Carry Select Adder (SRACSA) are presented. The adder in [55] is an optimized 64-bit adder, which a has a modular structure based on the Branch-Based Carry Select Adder (BBCSA) proposed for the SOI technology (Silicon On Insulator). This technology results in systems

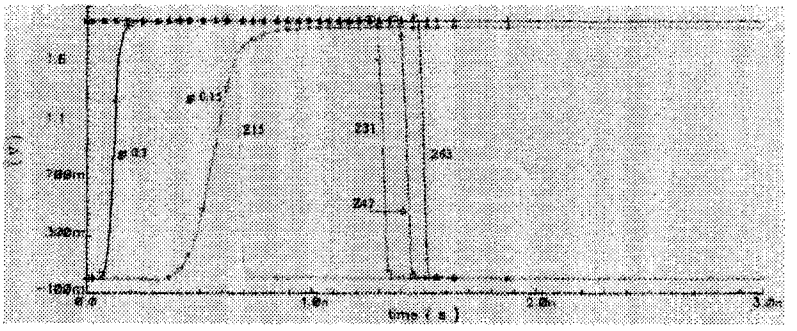


Figure 4.5: Delay measurement for bit positions $z_{16} = 679$ ps, $z_{31} = 1.3ns$, $z_{47} = 1.4$ns and $z_{63} = 1.5$n

Table 4.2: Comparison results

	Delay (ns)	Energy (pJ)	ED product	Area (μm^2)
BKCLA [24]	1.4	23	32.2	N/A
RCSA [61]	3	21	63	17,017
SRACSA [60]	1.7	18	30.6	N/A
BBCSA (SOI) (1.5V) [55]	0.72	96	69.1	205,800
AMCGG [20]	0.85	60	50	56,400
AFG [20]	0.85	52	43	53,976
HCLCS [45]	1.38	89.9	124	N/A
CVNS	1.5	14	21	13,250

which are in general 35% faster, and 33% more compact compared to the bulk technology. Therefore, to compare the systems these effects has to be considered. However, the BBCSA still has a high Energy-Delay product and area, making it unsuitable and costly for this type of application.

The values for the power and energy given is this table are based on operating frequency of 1MHz, and it will increase as the frequency of operation increases. The CVNS power consumption stays almost the same regardless of the fan-in and fan-out and the frequency of the adder.

The reduced area of the CVNS is due to the fact that operations are in current-mode, therefore addition of group of input digits becomes low-cost in terms of hardware implementation of the system. Moreover, summation over a group of input digits does not require global carry information from all of the input digits, and this information is passed to appropriate blocks efficiently, requiring only a limited number of interconnections in the system.

4.6 Summary

In this chapter, the design and implementation of a mixed-signal 64-bit adder, based on the Continuous Valued Number System, has been demonstrated. The 64-bit adder is a cascade of 4 16-bit CVNS adders. Carry information is generated locally for each block, hence the number of required nets in the system is small. This property also reduces the design complexity and allows easy partitioning of the adder on demand. This property along with the compact and low-power low-noise properties, makes this adder suitable for media signal processing applications. The 64-bit adder designed in the TSMC CMOS $0.18\mu m$ technology resulted in an adder with a worst case delay of 1.5 ns and static energy dissipation of about 14 pJ with a core area of 13250 μm^2.

Chapter 5

CVNS Column-Compression Multiplier

5.1 Introduction

Efficient logic circuit design is a fundamental task in the design of high performance devices. The design trend for the arithmetic system implementation in System-on-Chip (SoC) technologies, is to reduce the area, power dissipation and also system and cross-talk noise. Most modern arithmetic processors are built with architectures that have been well-established in the literature, with many of the latest innovations devoted to special logic circuits and the use of advanced technologies. Specifically, the design of multipliers is critical in digital signal processing applications, where a high number of multiplications is required. There are a wide variety of methods for multiplication with complexity order n and $log(n)$ gate delay [22, 34, 11, 76].

The Continuous Valued Number System can be used to reduce the interconnection complexity and the number of active devices in the system. The CVNS is an appropriate for

developing new types of arithmetic and signal processing units. Despite its analog nature, the CVNS theory, in fact, produces familiar arithmetic structures. While addition in the CVNS is digitwise, the general method of multiplication is based on the array multiplication approach.

Here, a new type of the CVNS multiplier is developed using CVNS compressors that can reduce the partial product columns by evaluating the values down to one gate delay. The final result is generated using a high-speed digitwise CVNS adder in radices other than the conventional binary radix. Such an approach is used to develop the digital multiplier discussed in this chapter, where the partial products are generated in the familiar digital form. The column compression and addition is based on the CVNS theory of digitwise addition. The proposed multiplier is designed into the target $0.18\mu m$ CMOS technology and favourably compared with standard binary multipliers in the same technology.

5.2 CVNS Multiplier

If the radix of the conventional number system is chosen as 2 (binary form), then y is represented in binary and each of its digits serve as an off-on switch for summing the CVNS partial products. Moreover, regular signed binary digits can also be used to reduce the summations.

There are, however, other methods for multiplying binary numbers using the CVNS theory. In particular regular arrays of binary partial products can be generated and then reduced using the CVNS compressors; finally the summation is performed on these columns, rather than summing the CVNS partial products, using CVNS compressors. Compressors are implemented using Boolean logic [12, 33] or by applying threshold logic gates [63, 91, 82].

We define a CVNS compressor as a unit which receives N binary inputs, and generates $\lfloor \log_2 N \rfloor + 1$ outputs. The outputs of a gate are equal to:

$$out_1 = (N) mod 2 \tag{5.1}$$

and

$$out_j = \begin{cases} 1 & (N) mod\ 2^j \geq 2^{j-1} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where $2 < j < \lfloor \log_2 N \rfloor + 1$.

Signal out_1 is also equivalent to the binary sum of N binary inputs. However, to sum N bits in a binary medium, binary full adders are required. In the CVNS this is obtained using just one gate, as will be shown later. For example, 5 binary inputs generate $\lfloor 2.3 \rfloor + 1 = 3$ outputs, where the outputs are $out_1 = (5) mod 2 = 1$, $out_2 = 0$ and $out_3 = 1$.

5.3 CVNS Binary Column Multiplier

A CVNS compressor gate has the capability of processing more than the usual values of 4 or 7 binary inputs each time, hence increasing the speed of partial product reduction with lower complexity. Figure 5.1 shows an 8×8 binary multiplication. Each column of partial products is sent to a CVNS compressor, the compressor outputs are passed to the CVNS adders to obtain the final results. For this purpose high-radix two operand CVNS adders have been used. The adder radix is chosen higher than two, in order to reduce the number of required gates and interconnections. In this example, two high radix CVNS adders are used in series. A 9-bit and then another 13-bit radix-4 compute the final result. The addition of the two operand binary inputs is based on the truncation of inputs, and is repeated here as follows:

$$((z))_{L-j} = [\sum_{i=L_1}^{m-\nu j} (x_i + y_i) B^{i+(1+\nu)j-L-1} + C_{L-j}] mod^{+} \beta$$

where C_{L-j} is the truncation signal which is used for removing the uncertainty form the MID digits, $L_1 = L - \nu j - (\psi - 1)$, and ψ is a technology dependent parameter limited by the maximum reliable resolution of the implementation environment.

The general configuration of the 8-bit multiplier is shown in Figure 5.2. The columns are processed in parallel and the layout of the final gates is very uniform with low complexity.

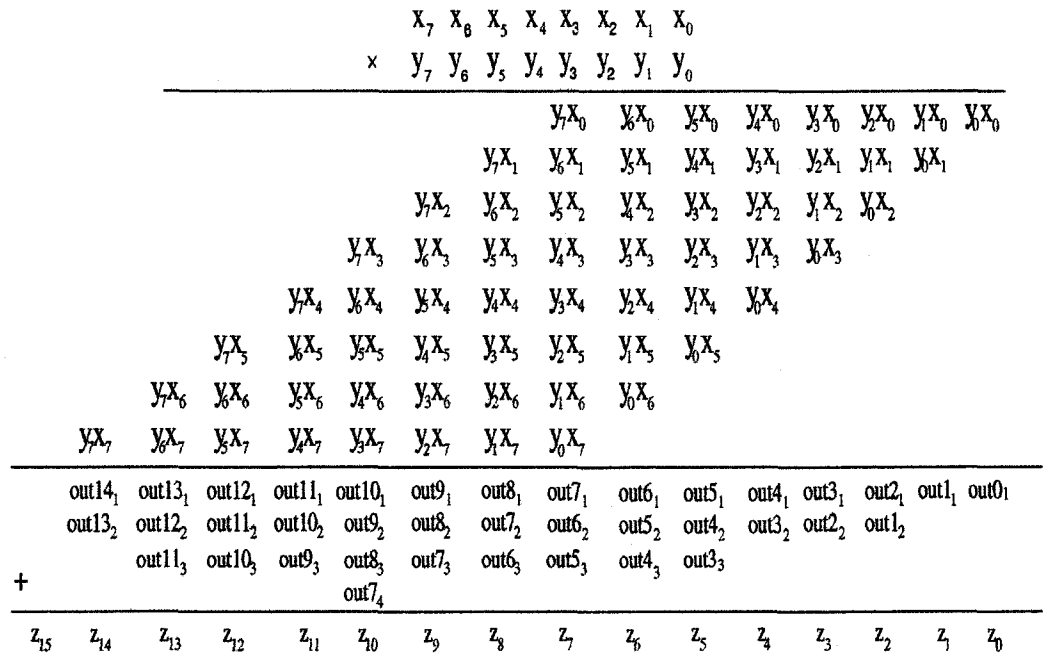


Figure 5.1: All of the required partial products are generated as required, and each column is compressed using the CVNS compressors. The final results are then added using high radix CVNS adders.

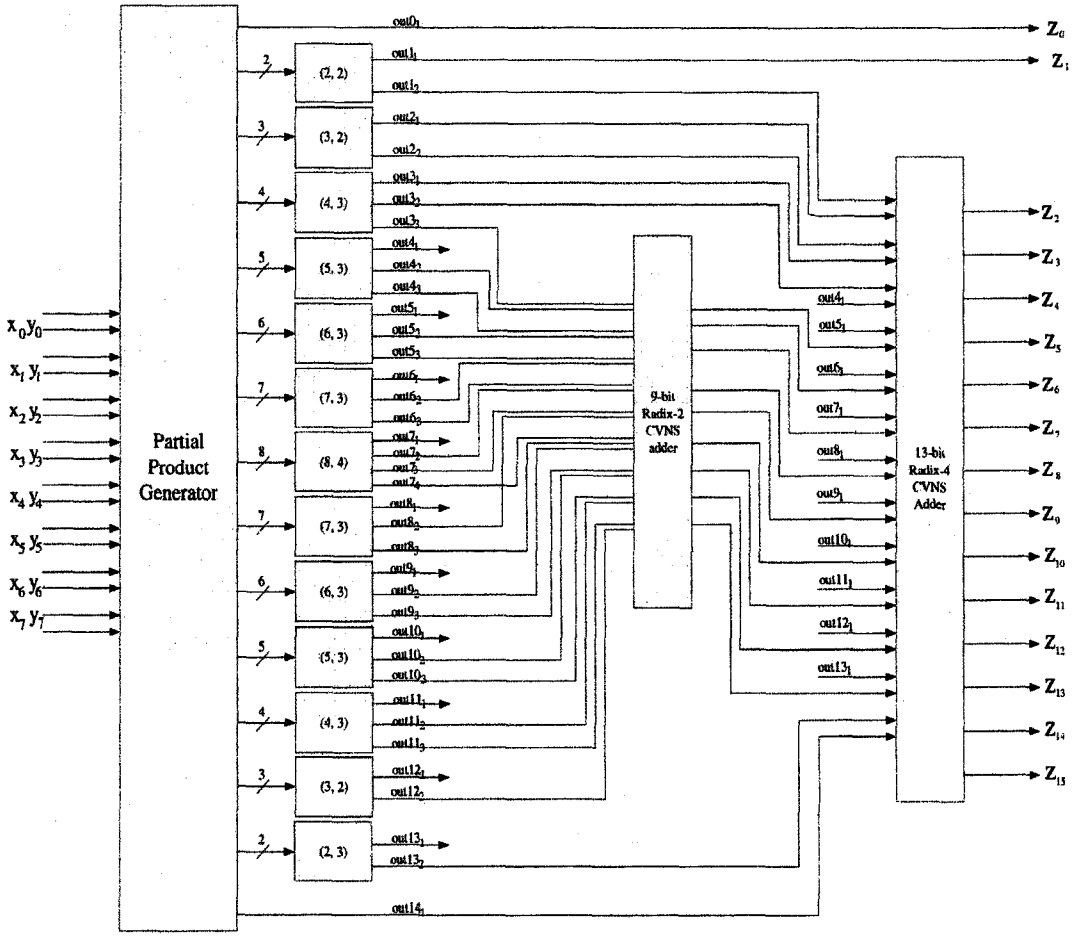


Figure 5.2: The 8×8 multiplier configuration, where partial products are compressed using CVNS compressors, and final results are generated two high radix CVNS adders.

The partial product generator block is the standard array of binary AND gates. To reduce these partial products, signed binary digits or Booth recoding can be used. The physical implementation and simulation results are given in the following section.

5.4 CVNS Column Multiplier Implementation

The central circuit of a CVNS adder is the modulus operation circuit, which is essentially a current comparator. In this design a differential current comparator is used, which provides higher noise immunity of the analog units.

We use the same style of circuit design as that of truncated adders. In our approach, the input current, i, is compared through a data dependent comparator, which results in lower power requirements, and decreases the area; it also provides improved better noise margin and higher gate speed.

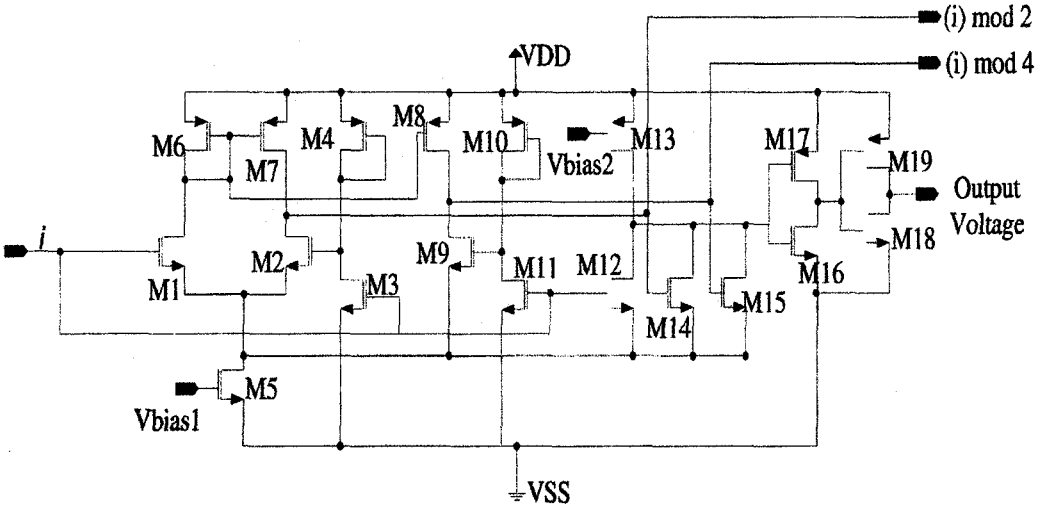


Figure 5.3: Current mode CVNS compressor.

The current, i, is the equivalent analog value of the binary inputs, which is compared with its complement value at the differential pair, through transistors M1-M7. The voltage in the secondary pair of the current comparator drops for values less than 2 and rises for values higher than or equal to 2. This branch also turns ON or OFF transistor M14 of the

second comparator, which is formed by transistors M8 through M11. This branch is the indicator for voltages higher than $2^2 = 4$.

Transistors M12-M19 are used for the output signal generation. The gate of transistor M14 is connected to the $(i)mod2$ branch, and the gate of M15 is connected to the $(i)mod4$ branch. The biasing of M13 causes the gates of M16-M17 to go HIGH for a logic 1 and go LOW for a logic 0. Accordingly, if either of the $(i)mod2^{1,2}$ branches go high, M14 and M15 are turned ON and cause a voltage drop at M16-M17 gates. The output of a (6, 3) CVNS compression gate is shown in Figure 5.4, where the input values are changing.

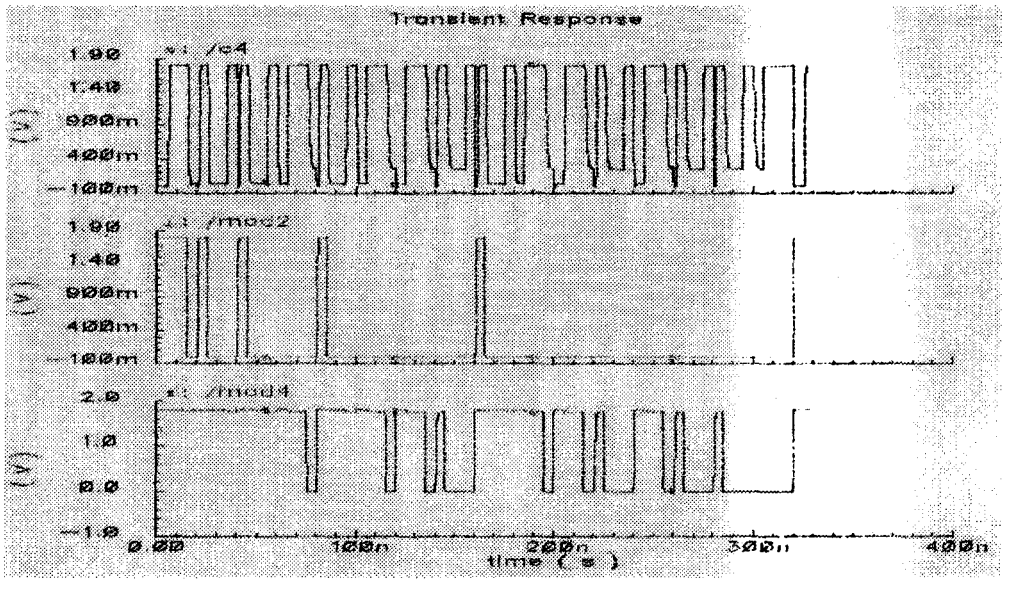


Figure 5.4: A sample output of a (6, 3) CVNS compressor.

The differential pairs are biased statically, hence the multiplier block has static power consumption. The dynamic power consumption due to the partial product generator unit is not significant, and can be ignored. Total system noise is also reduced due to the limited voltage variations and analog nature of the CVNS blocks. The multiplier is realized in TSMC CMOS $0.18\mu m$ technology, with a maximum delay of $900ps$ and a core area of $11200\mu m^2$.

Table 5.1: Comparison between the 8-bit CVNS multiplier and several standard digital multipliers of the same width in the same technology.

Multiplier Type	Delay
Wallace Multiplier	1.82 ns
MCML [73]	3 ns
Tree-RB [22]	2 ns
Reduced Tree Multiplier [88]	1.05 ns
Signed Multiplier [54]	1.6 ns
Pipelined Multiplier [39]	1.7 ns
Reconfigurable PLA and Multiplier [33]	1.1 ns
CVNS Multiplier	0.9 ns

The example demonstrates that CVNS designs can yield fast arithmetic circuits using low noise analog circuitry.

The functionality of the multiplier has been tested with more than 400 test vectors. In this configuration Column C12 has the longest path to the output and has to pass all three stages of compression and both CVNS adders. Therefore, it requires the computation of all of the truncation signals, and hence is the critical path of the multiplier. Table 6.1 shows the delay comparison between the CVNS adder and other digital multipliers.

5.5 Summary

The CVNS compressor function definition is introduced in this chapter, and a new CMOS current mode circuit is developed. The compressor design has been designed and simulated and its functionality verified. CVNS compressors are used for the partial product reduction in a digital multiplier, and the final addition is performed by two high-radix CVNS adders. The CVNS multiplier has high speed with a relatively small layout area. The design will

generate lower system noise than its CMOS logic counterparts, because of the smooth nature of the analog transitions.

Chapter 6

CVNS Array Multiplier

6.1 Introduction

Array multipliers offer a regular layout, and are generated by a regular array of full adders and half adders [34, 71, 81, 80, 74, 13, 25].

The CVNS has the potential advantage of reducing the wiring complexity and the number of active devices required in arithmetic circuits. The number system can provide an alternative path in the development of new types of arithmetic and signal processing units. Classical analog circuit blocks are used to construct the multiplier with arbitrary precision. While addition in the CVNS is digitwise, multiplication , in this chapter, is based on the classical array multiplication structure.

The CVNS array multiplier consists of a regular arrangement of addition operations and exploits the CVNS redundancy. The term *Array Multiplier* is derived from the array structure of a collection of adders, where they are laid out on a two-dimensional plane. The basis of the array multiplication are integer residue scaling and residue addition. Such an approach is used in developing the digital multiplier discussed in this chapter. The proposed

multiplier is designed into the target $0.18\mu m$ CMOS technology and favourably compared with standard binary multipliers in the same technology.

6.2 CVNS multiplication

Unlike addition, multiplication in the CVNS can not be performed digit wise. Therefore, CVNS multiplication is performed by summation of partial products, as in standard binary multiplication, using Equation 6.1 [67].

$$((z))_j = \Big(\sum_{\forall i} ((x))_{j-i} \times y_i\Big) mod\beta \tag{6.1}$$

where $0 < j \leq L$.

In general, by increasing the radix of the CVNS, fewer CVNS digits are required. Therefore, in the addition process discussed earlier, radices higher than 2 are used to reduce the hardware cost of implementation. However, this trend does not apply to CVNS multiplication. Thus the general expression given for the CVNS multiplication, generates redundant terms that are needed to be removed by modular reduction operations in higher radices ($\beta > 2$). The general expression for CVNS multiplication is less efficient in higher radixes.

As an example, radix-4 CVNS digits of a 4-bit multiplier are obtained as below. Based on the given expression for the CVNS multiplication, the 4-digit CVNS representation of x, $\{((x))_3, ((x))_2, ((x))_1, ((x))_0\}$, is as follows:

$$((x))_0 = 2x_1 + x_0 \tag{6.2}$$

$$((x))_1 = 2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0 \tag{6.3}$$

$$((x))_3 = 2^{-2}((x))_2 = 2^{-2}\big(2^{-2}((x))_1\big) \tag{6.4}$$

Also, y is converted into its radix-4 positional number system representation. This multiplication scheme generates 4 output terms, as follows:

$$((z))_0 \equiv \big((2x_1 + x_0).(2y_1 + y_0)\big) mod 4 = \Big(4x_1y_1 + 2x_1y_0 + 2x_0y_1 + x_0y_0\Big) mod 4 \tag{6.5}$$

$$\begin{aligned}((z))_1 &\equiv \Big((2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0).(2y_1 + y_0) + (2x_1 + x_0).(2y_3 + y_2)\Big) mod 4 \\ &= \Big(4(y_3x_1 + y_1x_3) + 2(y_3x_0 + y_2x_1 + y_1x_2 + y_0x_3) + (y_2x_0 + y_1x_1 + y_0x_2) \\ &+ 2^{-1}(y_1x_0 + y_0x_1) + 2^{-2}y_0x_0\Big) mod 4\end{aligned} \tag{6.6}$$

$$\begin{aligned}((z))_2 \equiv \Big(&2^{-2}(2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0).(2y_1 + y_0) \\ &+ (2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0)(2y_3 + y_2)\Big)\end{aligned} \tag{6.7}$$

$$\begin{aligned}((z))_3 \equiv \Big(&2^{-4}(2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0).(2y_1 + y_0) \\ &+ 2^{-2}(2x_3 + x_2 + 2^{-1}x_1 + 2^{-2}x_0)(2y_3 + y_2)\Big)\end{aligned} \tag{6.8}$$

In each term, the first partial product summation is canceled out by the modular reduction operator. This scheme yields accurate results; however, in terms of efficiency it is not the most efficient method of multiplication, due to the redundant and unused terms. The number of these unused terms increases in the CVNS representations with higher radices, and for larger input word lengths. These redundant terms have no effect on the output and are not useful for fine tuning the results either. Moreover, the design of the modular reduction circuit becomes more complex, in order to be able to detect additional levels and remove these terms, so as to form the canonic CVNS digits.

The effect of redundant terms may not seem drastic for a small multiplier, with low word length. However, for a higher number of input digits, the overhead cost of implementation increases. This effect does not happen for radix-2 CVNS multiplication based on the general expression for the multiplication.

If the radix is chosen equal to 2 (binary form), then y is represented in binary and each of its digits serve as an on-off switch for summing the CVNS partial products. Figure 6.1 shows the general configuration of a 4-bit, radix-2 CVNS multiplier.

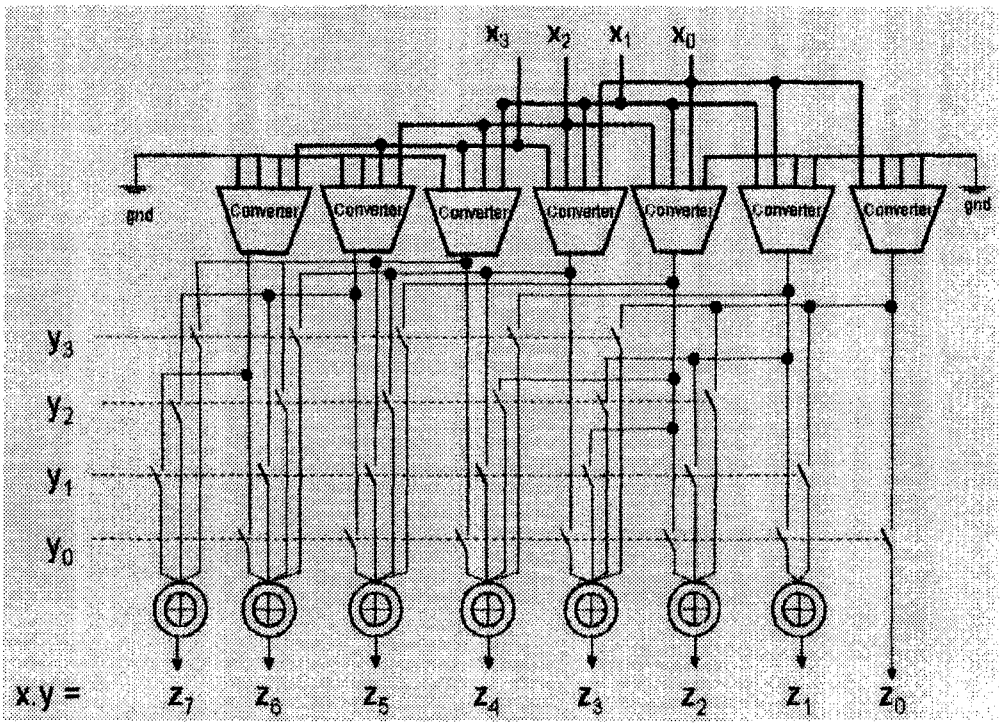


Figure 6.1: The CVNS 4-bit multiplier

It should be noted that this configuration is also suitable only for small word lengths. This method has two drawbacks; a low CVNS radix necessitates a larger number of digits, and multiplication based on Figure 6.1 requires error correction for higher word lengths. Error correction increases the delay of the operation, which is not desirable.

An approach that can perform the multiplication in higher radixes, with lower complexity and delay is more desirable. There are, other methods for multiplying binary numbers using the CVNS theory. In particular regular arrays of the CVNS adders can be generated and partial product terms are reduced by a series of high-radix CVNS adders. This approach is explained in the next section, followed by a design example of a 16×16 multiplier.

Moreover, two alternative methods of multiplication are given which can be easily implemented using the same circuit design style. We call these methods *MID Switching* and *Comparator Multiplier*. The general approach of multiplication in these suggested methods is similar to the array multiplier.

6.3 CVNS Array Multiplier

In this approach, the CVNS digits of x are generated, and are added together according to the digit values of y. Digits of y are used as enable signals. This will ensure that the multiplication scheme remains efficient. Moreover, in this approach the complexity of the multiplier is reduced, while the information redundancy is kept at its maximum. Therefore, general expression for the CVNS multiplication is not used, instead the CVNS multiplier structure is generated by an array of high radix CVNS adders. The proposed method takes advantage of high performance CVNS adders, where CVNS digits can be represented in higher radices.

At the first layer of the array multiplier for $m \times m$-bit multiplication, $m/2$ L-bit CVNS adders are required. At each layer, the number of terms is reduced by half, considering the fact that two operand CVNS adders are used. The number of layers, at this configuration, is dependent only on input word length. We demonstrate this approach through an example;

the design and implementation of an 16 × 16 CVNS array multiplier.

6.3.1 16 × 16 Array Multiplier

In this section, a design example for a 16 × 16 array multiplier, based on the CVNS approach, is demonstrated. The Proposed CVNS array multiplier is structured considering the required binary partial products. In this design, we will take advantage of our previously designed 16-bit adder. The CVNS adders use a radix-4 truncated addition method which requires minor modification at the first layer.

At the first layer of the CVNS multiplier, multiplier digits act as activation signals for the CVNS adders. The general configuration of the multiplier is shown in Figure (6.2). There are in total $log_2(m)$ layers, where each layer reduces the partial products in half. For 16-bit multiplication, 4 layers are required.

6.3.2 Analog Circuits

To design the first layer of the multiplier, a differential style circuit design is used for their obvious advantages in increasing the noise margin, and providing higher operational performance.

To increase the symmetry of the differential pair and to reduce the mismatch effect at the input pair, the input signal to the differential pair is split into two equal sets of signals.

The input signal i is the result of adding a group of length ψ digits. We apply the truncated addition method, and we need to detect the gt and rt signals, which are generated by the differential open-loop comparators. The inequality at this stage produces the following values of the gt and rt signals at the first layer of:

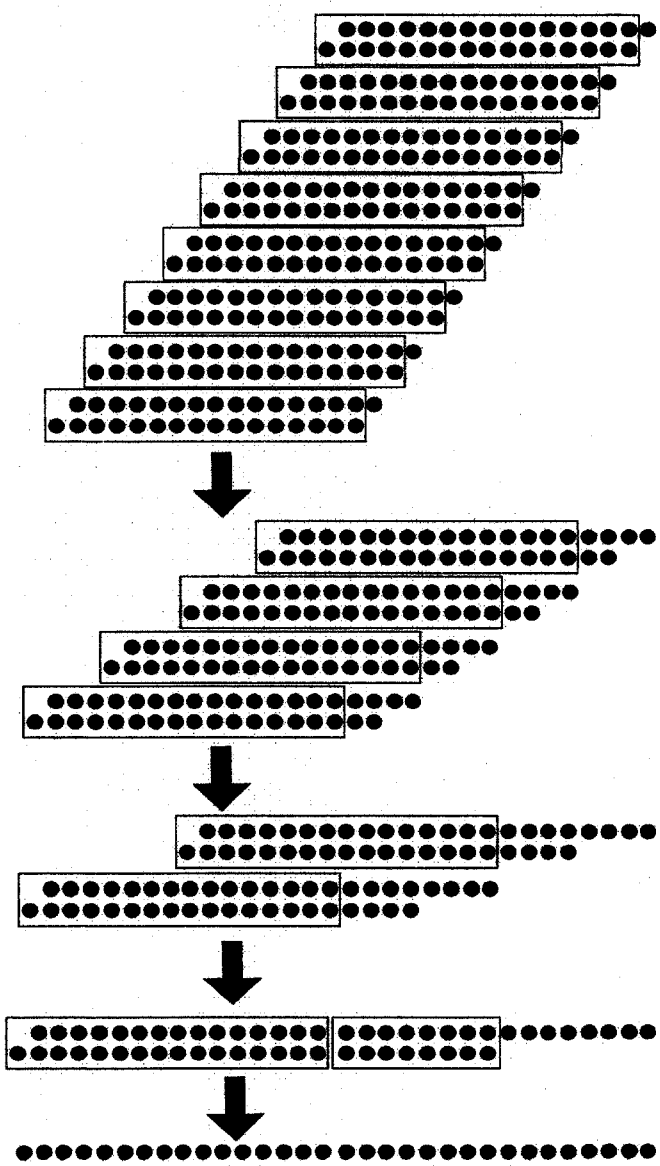


Figure 6.2: 16 × 16 bit binary multiplication is performed by 4 layers of high radix CVNS adders.

$$gt_{(p)} = \begin{cases} 1 & \sum_{i=4p-4}^{4p-1} (y_{j-1}x_i + y_j x_{i-1}) 2^{i-4p-1} \geq 2 \\ 0 & otherwise \end{cases} \tag{6.9}$$

$$rt_{(p)} = \begin{cases} 1 & \sum_{i=4p-4}^{4p-1} (y_{j-1}x_i + y_j x_{i-1}) 2^{i-4p-1} \geq 1.875 \\ 0 & otherwise \end{cases} \tag{6.10}$$

where $1 \leq p \leq 3$ is an integer, and $0 \leq j \leq 15$ is the index value of y digits.

In our improved version of the comparator, we change the above inequalities to

$$gt_{(p)} = \begin{cases} 1 & \sum_{i=4p-4}^{4p-1} y_j x_{i-1} 2^{i-4p-1} \geq 2 - y_{j-1}x_i 2^{i-4p-1} \\ 0 & otherwise \end{cases} \tag{6.11}$$

$$rt_{(p)} = \begin{cases} 1 & \sum_{i=4p-4}^{4p-1} y_j x_{i-1} 2^{i-4p-1} \geq 1.875 - y_{j-1}x_i 2^{i-4p-1} \\ 0 & otherwise \end{cases} \tag{6.12}$$

The same principle for the addition and computation of the gt and rt signals is used; however, this time the differential pair circuit has a more symmetrical structure, and each input transistor is connected to a more balanced configuration of parasitic capacitors.

Moreover, for computing the gt signals, the maximum value of each of the signals $y_j x_{i-1} 2^{i-4p-1}$ and $y_{j-1}x_i 2^{i-4p-1}$ is $\beta - 2^{-\psi}$, which means that no carry is generated within

the groups, if either one of y_j or y_{j-1} signals are equal to zero. Therefore, the differential pair circuit operation becomes data dependent, with the enabling signal $switch \equiv y_j \wedge y_{j-1}$. The new circuit, shown in Figure (6.3), exhibits low gate delay, and its driving capability can be increased by adding high drive buffers at the output.

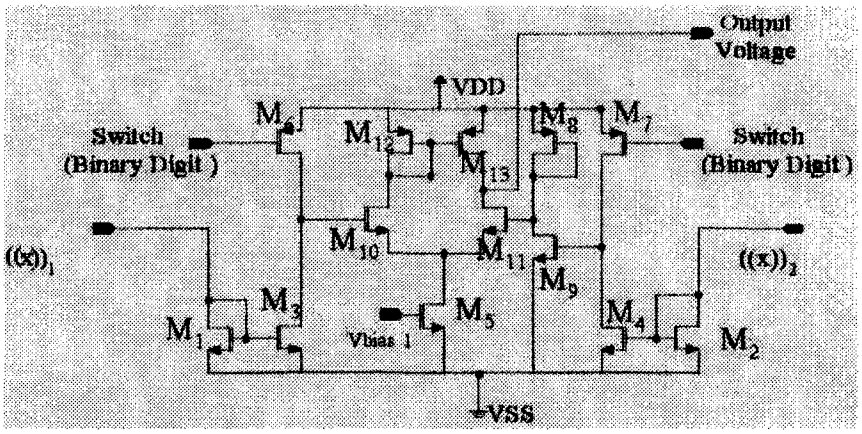


Figure 6.3: New MDAC style current comparator for detecting the carry information in a group of CVNS digit of length ψ.

The modular reduction circuit, shown in Figure (6.4), can not take advantage of this configuration. Therefore, the input current which is the equivalent analog value of the binary inputs, is compared with its complement value at the current comparator. The output voltage of the current comparator drops for values less than 2 and rises for values higher or equal to 2, which turns ON or OFF transistor M19. Transistors M15-M19 are used for the binary output signal generation. The biasing of M18 causes the gates of the buffers to go HIGH for a logic 1 and go LOW for a logic 0. Accordingly, if the $(i) mod 2$ branch goes high, M19 is turned ON and causes a voltage drop.

The CVNS multiplier is designed and simulated in the TSMC CMOS 0.18μm technology, with a maximum delay of 3.2 ns, and a core area of 42137μm^2. The example demonstrates that CVNS designs can yield fast arithmetic circuits using low noise analog circuitry. For comparison purposes, the performance of the analog CVNS multiplier is compared with various types of digital array multipliers, and the results are shown in Table 6.1.

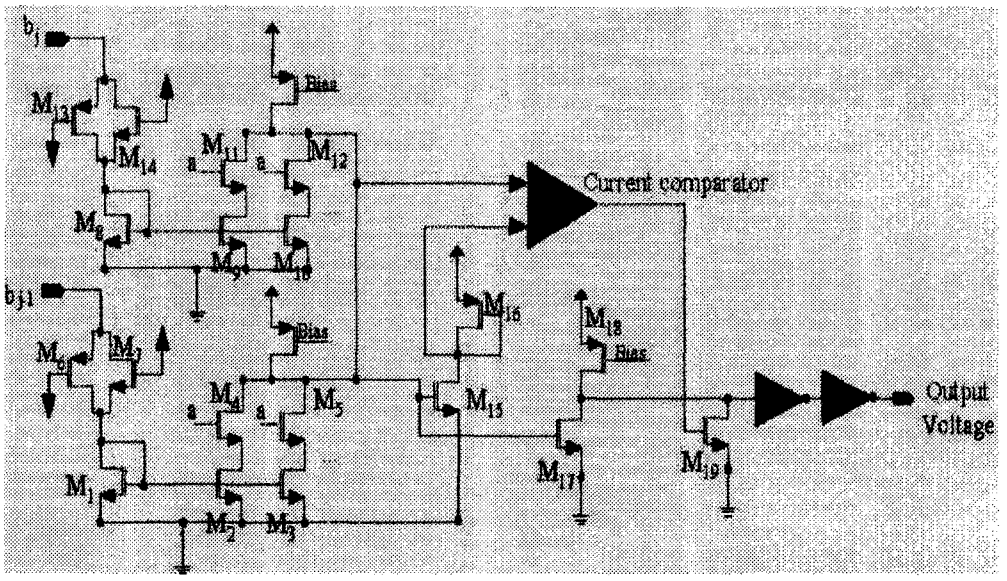


Figure 6.4: Current mode modular reduction circuit

Comparisons between this example as the first implemented CVNS multiplier showed that it can achieve performance comparable to mature digital designs. The total area of the CVNS multiplier can be reduced even more, considering that the first layer requires copies of the CVNS digits of x. This means that the continuous value of this digit has to be distributed in the chip area, which requires a more carefully drawn layout. In this design we generated the MID digit locally for each adder, that contributed to an overhead in area; however, the robustness of the design against noise has been increased.

A reduced size multiplier has been sent for fabrication as a proof of concept design. A full custom 8×8 CVNS array multiplier layout has also been generated, and is ready for fabrication in a 40-pin package. In total, seven radix-2 8-bit and two radix-2 2-bit adders were designed. There are approximately 4000 transistors, individually sized and laid out. The layout is based on a common centroid method, and transistors are fingered in the same orientation for better matching. The minimum size for NMOS is 350/200, and for PMOS is 500/200. We have chosen a lower radix for this design, in order to decrease the circuit complexity and to increase the robustness of our prototype chip. Figures 6.5, 6.6 and 6.9

Table 6.1: Comparing the CVNS multiplier and several standard digital multipliers.

Multiplier	Delay (ns)	Area (μm^2)
Low Switching Activity ($0.25\mu m$) [15]	9.8	381,238
Split Linear Array Multiplier (24-bit) [34]	5.91	44680
Linear Array Multiplier (24-bit) [34]	5.88 ns	43217
Modified Linear Array Multiplier [25]	4.39	126000
Multiplexer Based Array Multiplier (Truncated)[13]	4.34	5980
Multiplexer Based Array Multiplier (Truncated)[13]	4.42	6094
Pipelined Array Multiplier [71]	3.4	N/A
Simple High Speed Multiplier ($0.13\mu m$) [38]	4.42	14589
Array Multiplier using 12-T Adders (4×4) [81]	3.97	N/A
MCML Architecture (8×8) [74]	3.8	N/A
Optimal Bit-Level Multiplier [80]	3.52	36797
CVNS Array Multiplier ($0.18\mu m$)	3.2	42137

show parts of the layout and extracted view and the routed chip, respectively.

CVNS multiplication methods are not limited to only the above two approaches. There are a wide variety of different types of circuits and topologies, that can be applied for the CVNS multiplication. In this section, we have proved that the CVNS multiplication is feasible, and the end results are comparable to conventional digital designs. In the next sections, we briefly mention two alternative methods for the CVNS multiplication, followed by the summary.

6.4 Comparator Multiplier

This scheme is based on the array multiplier proposed in the previous section. Although in this technique, digital information is converted into the CVNS form, it does not require

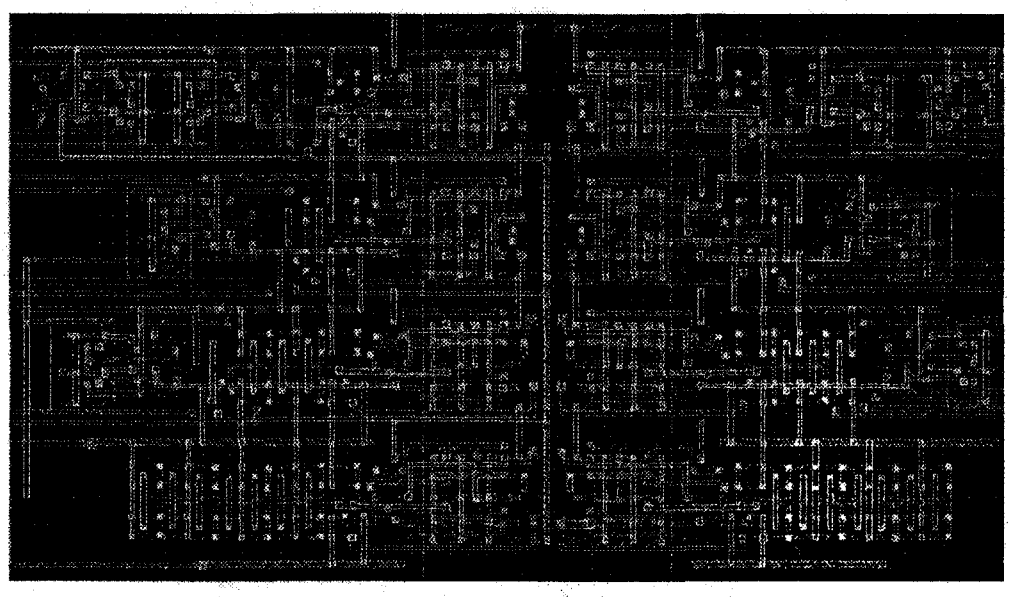


Figure 6.5: Layout view of some parts of the 8 × 8 array multiplier

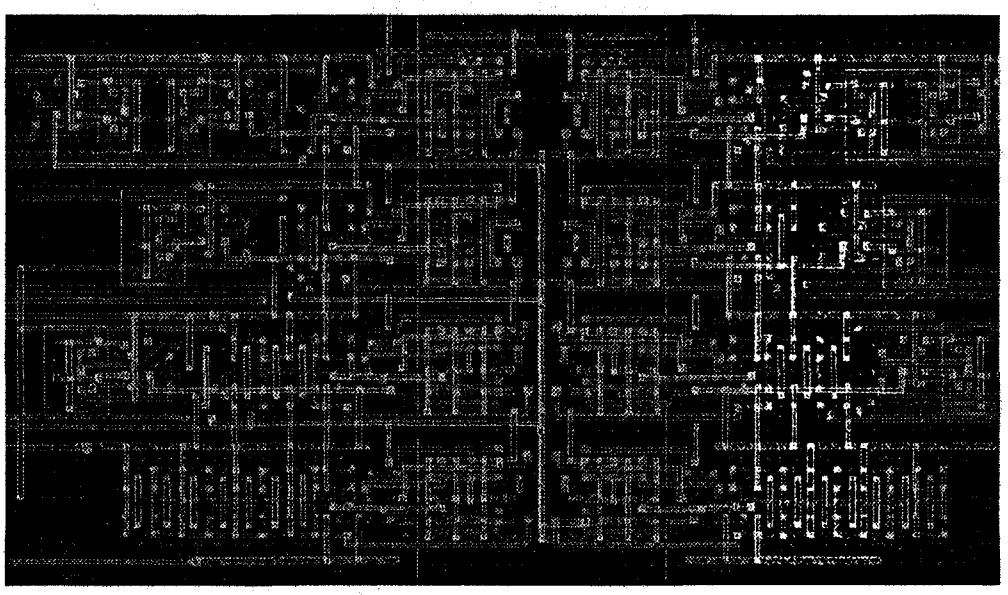


Figure 6.6: Extracted view of some parts of the 8 × 8 array multiplier

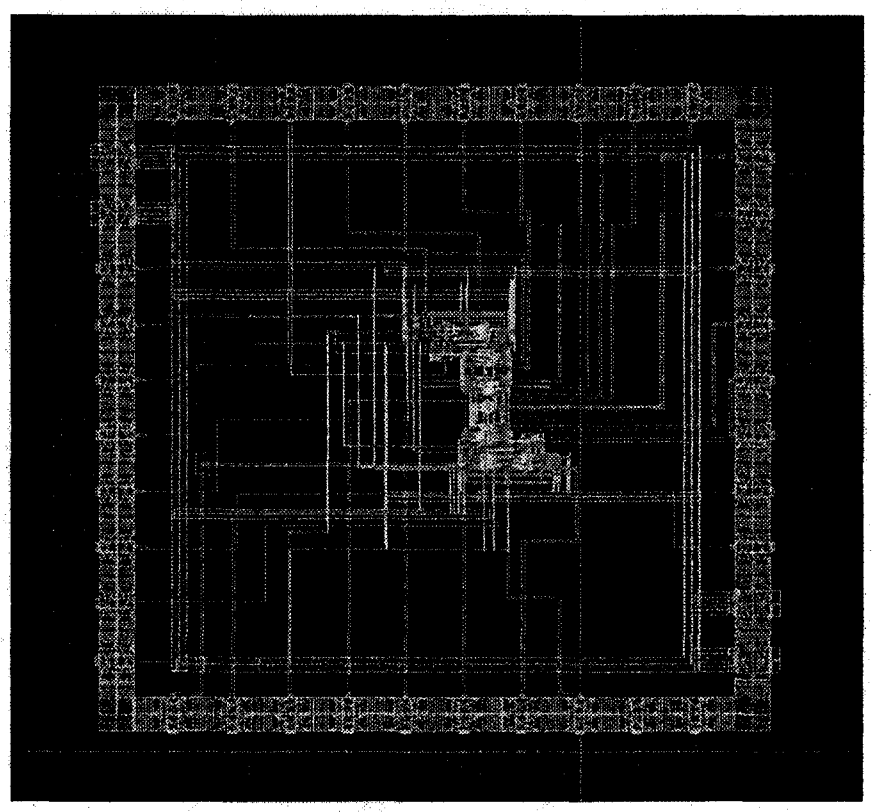


Figure 6.7: 8 × 8 CVNS array multiplier

conversion to binary after each two in put addition.

The CVNS digit set of x and $2x$ are formed, and are multiplied with proper digits of y through MDACs, the same as the array mulitplier. At the second stage, proper CVNS digits can be added together in current mode, and therefore the speed of reducing the partial products increases. At the output, each column is compared with its adjacent column from the right. Comparison is based on the rounded floor function, which is used in the RE process. The operation speed of this multiplier is a factor of m (number of input digits) higher compared to the basic CVNS multiplication architecture (which requires frequent error correction), because it does not require any correction.

Although this method offers higher speed, the modular reduction circuit design becomes rather complex.

6.5 MID Switching Multiplier

MID switching operations are mostly in the analog domain, and unlike the array multiplier it does not require frequent conversions to and from binary. The MID principal of operation is very close to a reduced area array multiplier architecture. The MID multiplier relies on the maximum reliable resolution of the analog domain, and generates only the MID digit. The MID digit has all of the required information of the lower informed digits, therefore required information can be extracted from it. Figure 6.8 shows the general configuration for a 4×4 multiplication scheme.

For low-resolution multiplication, the MID switching multiplier shows great savings in terms of area and power. MID Switching can also be used for the truncated multiplications. By cascading the MID switching multipliers, the range of the multiplication can be extended. The speed of this multiplier is dependent on the A/D converter speed. This multiplier is completely analog and summations in current mode result in a high speed of operation and area savings. However, it requires high speed A/D converters in order to generate all of the binary digits from a single analog CVNS digit.

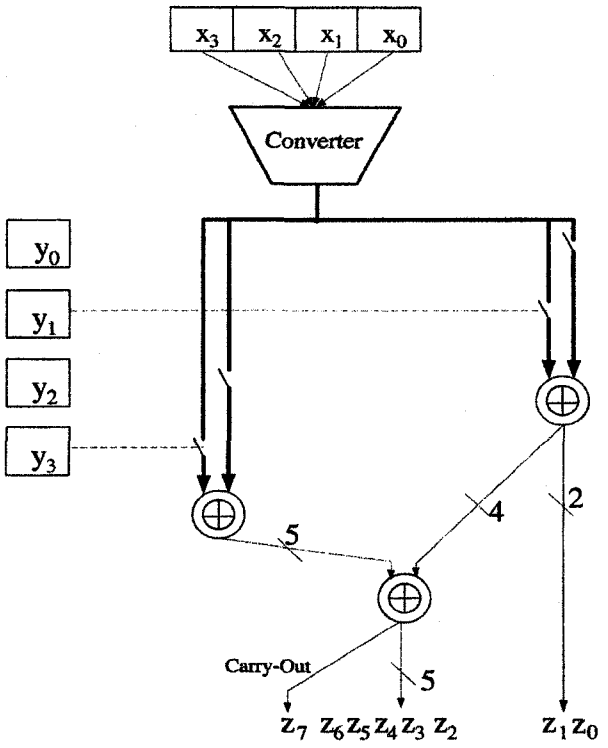


Figure 6.8: MID Switching CVNS multiplier

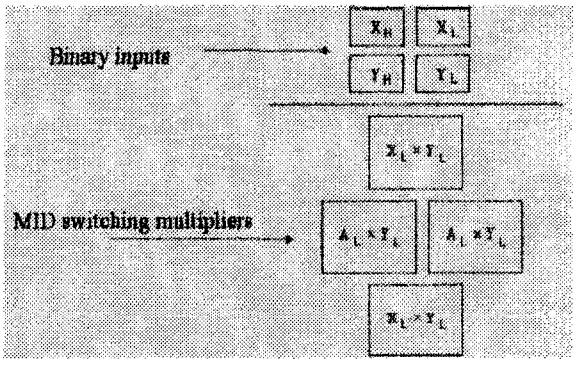


Figure 6.9: Extending the range of MID multiplier

6.6 Summary

In this chapter, we have explored the possibility of applying the CVNS theory for digital multiplication. The CVNS array multiplier is composed of a regular array of high radix CVNS adders. The CVNS multiplier has high speed and with a relatively small layout area. By applying this method, higher radixes can be used, which reduces the area and increases the efficiency of the operations. However critical path delay of operation may not decrease, as, in our scheme, it is dependent on the number of input digits.

The area of this multiplier can be reduced further using a more in trade off to a more careful layout. In our designs we have chosen a conservative approach, where we generate all of the required CVNS digits.

Two alternative methods of multiplication are also suggested. The CVNS field is new, and alternative methods offered here are not the only possibilities for the CVNS multiplication.

Chapter 7

CVNS Based Neural Networks

7.1 Introduction

The field of neural networks takes a special place in analog and digital signal processing. Analog Neural Networks (Analog NN) [2, 3, 17, 28, 32, 37, 43, 58, 59] provide an efficient means for implementing neurons with simple and elegant nonlinear analog circuits and with only a few transistors. However, the low precision of analog circuits is a limiting factor for realizing large size Analog NNs. The size and complexity of such networks is affected by reduced noise margin, power supply and low resolution (6 to 7 bits) of analog circuits.

Digital Neural Networks [9, 92, 5], on the other hand, can extend the resolution, and offer a practical solution for storing the synaptic weights on the chip. However, such networks face the difficulty of generating a smooth neural activation function which is impeded by digital quantization and by increasing circuit complexity required for implementing the nonlinear function. Proper network synaptic convergence depends on the differentiability of the nonliner neural function, which makes requires the function to be analytic (or smooth).

The CVNS approach can extend the dynamic range of a continuous signal beyond the

limitations that are imposed by the analog medium. In the CVNS, its smoothness of the nonlinear function is maintained in the digit with the highest index, and only a few additional digits are required to increase the accuracy. Moreover, the ability to keep the number of digits low, reduces the synaptic interconnection complexity.

In this chapter, a model of an Adaline neuron based on the function evaluation properties of the CVNS is proposed. We explore improvements in the CVNS implementation of Analog NN in order to improve the robustness of a network, built by such elements, to weight errors. The effect of weight inaccuracies are studied in this chapter through stochastic modeling of the proposed network. Section 7.2 presents our proposed CVNS Adaline, and section 7.3 covers the stochastic model of a network of our proposed Adaline, followed by a summary.

7.2 CVNS Madaline Network

The Adaline (*Ada*ptive *lin*ear *e*lement) is widely used in filter and signal processing. A network of such elements, called a *Madaline*, is a feed-forward multi-layer network of Adaline elements. Such network is trained by the LMS or Widrow-Hoff algorithm, also known as delta rule [87].

A CVNS Adaline consists of a main function and a few sub-functions. Sub-functions have information overlap in the same way as the analog-digits. Therefore, CVNS operations have higher precision compared to purely analog implementations. Figure (7.1) shows the general configuration of our proposed single CVNS Adaline. Synaptic weights are stored on multilevel analog memories on the chip in the CVNS format. Inputs to the network are not required to be converted into the CVNS form and are kept in their original format (positional Number System). Multiplication in the CVNS does not require both values to be converted into the CVNS, and is based on scaling the modular CVNS digits.

The output of each multiplier is in the CVNS form, and is as follows:

$$((u))_{i|j,n} = x_j.((w))_{i|j,n} \tag{7.1}$$

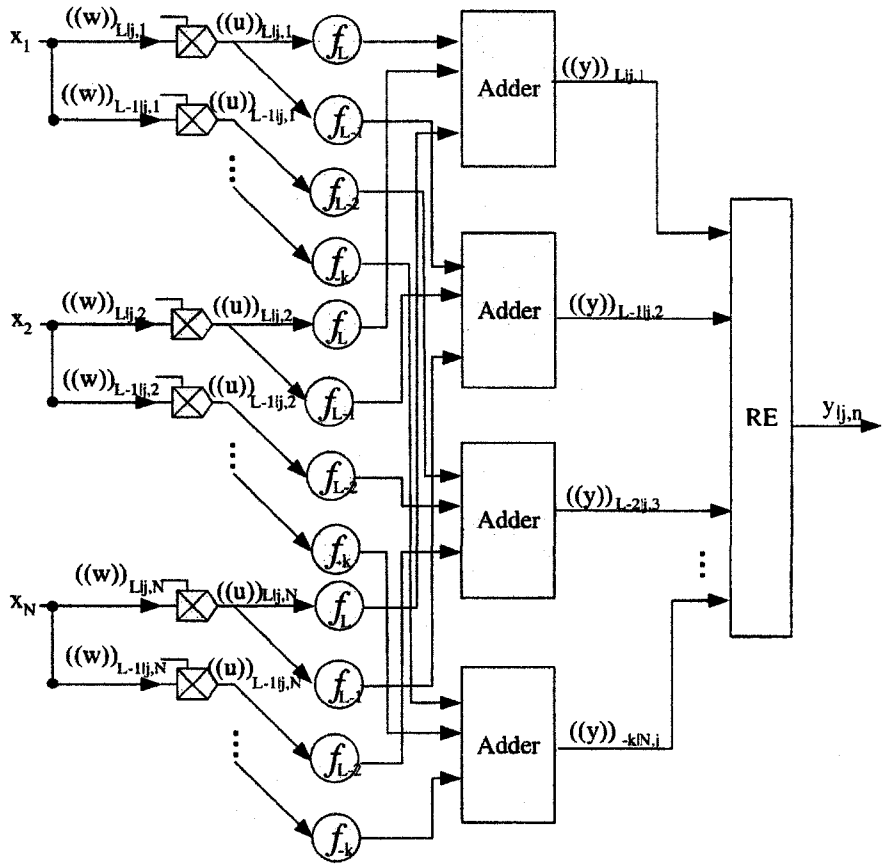


Figure 7.1: CVNS Adaline configuration, where inputs are multiplied by the CVNS synaptic weights, and nonlinear activation function, f, is developed from function evaluation properties of the CVNS.

where $-k \leq i \leq L$ is the number of CVNS digits, and $((u))_{i|j,n}$, indicates a CVNS digit with index i, at layer j and node n.

Each CVNS Adaline nonlinear function consists of a main nonlinear function, and, depending on the number of digits, there are a few nonlinear sub-functions which are used for refining the output. Before the RE process, each CVNS Adaline output has $L+k$ analog-digits, where each element, denoted as $((y))_i$, is given by:

$$((y))_{i|j,n} = \sum_{t=1}^{N} f_i\Big(((u))_{i|j,n}\Big) \tag{7.2}$$

where $-k \leq i \leq L$.

The output of a CVNS Adaline, $y_{j,n}$, is generated from multiple analog digits in a series method, called Reverse Evolution, and are repeated here for convenience.

$$\lfloor((y))_i\rfloor_R \equiv [((y))_i - y''_{i-1}/\beta]_R mod\beta \tag{7.3}$$

and

$$y''_i = \begin{cases} \lfloor((y))_i\rfloor_R + y''_{i-1}/\beta & i > -k \\ ((y))_{-k} & i = -k \end{cases} \tag{7.4}$$

where $\lfloor a_i\rfloor_R \equiv [a_i - a_{i-1}/\beta]_R$ and $[\,.\,]_R$ is the rounded floor function operation.

In the CVNS with radix β and $L+k$ digits, the main function, f_L, has the highest level among the sub-functions. The nonlinear function used here is based on a sigmoidal function commonly used in multi-layer networks. The sigmoidal function is expressed as follows:

$$f(a) = \frac{1-e^{-2a}}{1+e^{-2a}} = tansig(a) \tag{7.5}$$

The highest level function, and the next sub-function are obtained as follows:

$$f_L = \frac{\beta}{M} tansig\Big(\frac{M}{\beta}((u))_{L|j,n}\Big) \tag{7.6}$$

$$f_{L-1} = \left(\frac{\beta^2}{M}.tansig\Big(\frac{M}{\beta}((u))_{L|j,n}\Big)\right) mod\beta \tag{7.7}$$

It can be noted from the function evaluation properties of the CVNS that the sub-functions can be obtained either from the lower or higher hierarchial functions. If we choose to find the sub-functions from lower hierarchial function, the precision increases; however, hardware cost of implementation also increases. On the other hand, if sub-functions are obtained from the higher hierarchial function, the cost decreases, but it may require more accurate circuits. Based on these trade offs, a choice for selecting different levels of hierarchy has to be made. In this chapter, we have selected a midway point, where each subfunction is obtained from its next higher hierarchial level. Therefore, the complexity of the network is low, and demand on the accuracy is not high. Moreover, one less CVNS digit is required for storage.

If a network is built by transconductance synapses, and current-mode resistive-type neurons, summation and nonlinear functions in the network can be easily performed by hard-wiring the outputs of the neurons to distributed resistive-type neurons. In this form neurons are divided into parallel sub-neurons, where the required number of such elements is equal to the number of inputs to the synapses multiplied by the number of CVNS digits. Such a network is shown in Figure 7.2, where neurons are indicated as resistors.

Resistive-type neurons have an adjustable activation function to allow increase of fan-in to the network. Moreover, distributing the neurons increases the signal to noise ratio of the network, and reduces network sensitivity to mismatch and non-idealities of the analog environment [21].

The required quality of the implementation, dictates the CVNS dimensions; namely the radix, and the number of digits. To increase the accuracy of the analog operations and increase the precision, more CVNS digits can be used. The number of digits is chosen based on the desired output Noise-to-Signal-Ratio (NSR). These relations are derived from the stochastic modeling of the CVNS analog neural network in the next sections.

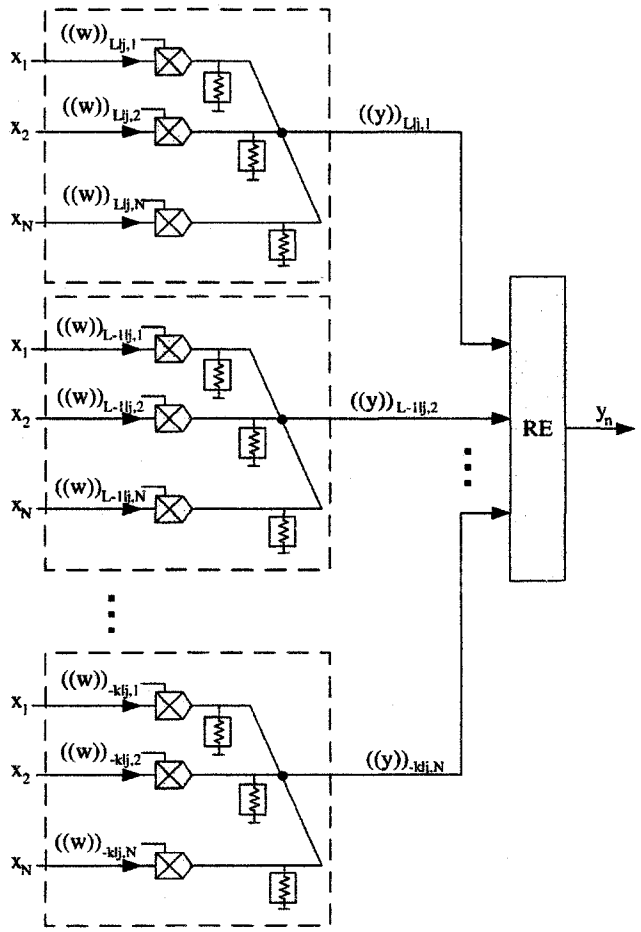


Figure 7.2: CVNS Adaline with distributed neurons

7.3 Neural Network Sensitivity to Errors

The required weight accuracy in a neural network hardware realization has been studied extensively [62, 90, 1, 89, 16, 18, 56, 19]. The sensitivity studies in general are based on two methods; the analytical method, which defines the sensitivity as the partial derivative of the output to input; the statistical approach, which measures the sensitivity as the ratio of the standard deviation of output to the standard deviation of the synaptic weights or inputs. In the next section, the statistical approach is used to study the effect of implementation errors on a large network. The study considers the effect of error at the recall state or effects of implementation error on a trained network, and does not directly address these effects on the learning algorithms.

Stochastic modeling of the design allows us to compare the accuracy of our CVNS based neural network with regular networks in terms of NSR (Noise-to-Signal-ratio). The output Noise-to-Signal-Ratio (NSR) is defined as the ratio of the variance of the output error to the variance of the ideal output (without any error which may have been caused by the quantization effect, implementation issues, etc.). We are interested to explore the use of the CVNS to improve the accuracy of Analog NN implementations.

Stochastic modeling is used to identify the relationship between input and output of the network. In this way, the effect of an imperfect implementation is derived, which in turn allows us to decide on the complexity of the system. Based on the evaluated accuracy of the network and specifications of the environment, dimensions of the CVNS system are chosen.

The nonlinear function is an odd function, and all of the weights in the network have the same variance. Moreover, it is assumed that the mean value of each weight in the network is zero. The same assumptions are made for the inputs, errored weights and errored inputs.

The stochastic model provided in [62] defines the output and corresponding error of an Adaline in an arbitrary layer of the network as follows (given in CVNS terms):

$$((\mathbf{Y}))_{|j,n} = f\Big(\mathbf{X}^T.((\mathbf{W}))_{|j,n}\Big) \tag{7.8}$$

$$\Delta(\langle\langle \mathbf{Y} \rangle\rangle_{|j,n} = f\Big((\mathbf{X} + \Delta\mathbf{X})^T.\big(\langle\langle \mathbf{W} \rangle\rangle_{|j,n} + \Delta(\langle\langle \mathbf{W} \rangle\rangle_{|j,n}\big)\Big) \tag{7.9}$$

Since weight and input matrices are considered zero-mean independent identically distributed variables, therefore the output variance of all CVNS outputs before RE in any layer are the same, and the NSR of the output is:

$$NSR = \frac{\sigma^2_{\Delta\langle\langle y \rangle\rangle_{i|j,n}}}{\sigma^2_{\langle\langle y \rangle\rangle_{i|j,n}}} \tag{7.10}$$

The output NSR ratio is a linear combination of input NSR, $\sigma^2_{\Delta x_j}/\sigma^2_{x_j}$, and weight NSR, $\sigma^2_{\Delta\langle\langle w \rangle\rangle_{i|j,n}}/\sigma^2_{\langle\langle w \rangle\rangle_{i|j,n}}$, amplified by a gain function $g(.)$, expressed as follows:

$$NSR = g(\sqrt{N}\sigma_x\sigma_{\langle\langle w \rangle\rangle}) \times \Big(\frac{\sigma^2_{\Delta x}}{\sigma^2_x} + \frac{\sigma^2_{\Delta\langle\langle w \rangle\rangle}}{\sigma^2_{\langle\langle w \rangle\rangle}}\Big) \tag{7.11}$$

where N is the number of inputs to the Adaline, σ_x and $\sigma_{\langle\langle w \rangle\rangle}$ are the standard deviations of inputs and synaptic weights respectively, and, for convenience, the subscripts are removed.

The Analog NN implemented by the CVNS approach increases the precision of analog circuits using multiple analog digits. In our proposed neural network model, the final output of neuron is refined to increase the accuracy. There is, however, a small approximation error in the final value which is propagated upward from the unchecked Least Informed Digit. The approximation error is the implementation error of the LID and causes an error in the output as follows:

$$\Delta y = \frac{\Delta\langle\langle y \rangle\rangle_{-k|j,n}}{\beta^{L+k}} \tag{7.12}$$

where $\Delta\langle\langle y \rangle\rangle_{-k|j,n}$ is the error in the Least Informed Digit.

The variance of the output error is:

$$\sigma^2_{\Delta y} = \frac{\sigma_{\Delta\langle\langle y \rangle\rangle_{-k|j,n}}}{\beta^{2(L+k)}} \tag{7.13}$$

Therefore, the final Noise-to-Signal-Ratio, NSR_{CV}, is:

$$SNR_{CV} = \frac{1}{\beta^{2(L+k)}} g(\sqrt{N}\sigma_x\sigma_{((w))}) \times \left(\frac{\sigma^2_{\Delta x}}{\sigma^2_x} + \frac{\sigma^2_{\Delta((w))}}{\sigma^2_{((w))}}\right) \tag{7.14}$$

As an example a case study is presented below to show the improvements.

Example: Improvements in error tolerance We are interested in implementing a three layer Madaline with 256 neurons in each layer. The acceptable, NSR is $-20dB$, and weights and inputs are limited, by the hardware implementation, to $[-0.3, 0.3]$ and $[-1, 1]$ respectively. Therefore, the estimated variance of weights and inputs is $\sigma^2_{((w))} = 0.03$ and $\sigma^2_x = 1$. The gain function is approximately equal to:

$$g\left(\sqrt{N}\sigma_x\sigma_{((w))}\right) \approx 0.5 + 0.53\sqrt{N}\sigma_{((w))}\sigma_x \tag{7.15}$$

The output NSR is:

$$NSR_{CV} = \frac{1}{\beta^{2(m+k)}}(g^3 + g^2 + g)\frac{\sigma^2_{\Delta((w))}}{\sigma^2_{((w))}} \tag{7.16}$$

The accuracy of the weights and number of digits is:

$$\frac{1}{\beta^{m+k}} \times \frac{\sigma_{\Delta((w))}}{\sigma_{((w))}} = 0.027 \tag{7.17}$$

Assuming that the dimensions of the CVNS are $\beta = 2$ and $m + k = 2$, we have:

$$\frac{\sigma_{\Delta((w))}}{\sigma_{((w))}} = 0.108 \tag{7.18}$$

This means that the magnitude of the weight errors must be less than 0.108 of the magnitude of the weight. Without the CVNS, the magnitude of the weight error should be 0.027 of the weight magnitude, and there is an improvement by a factor of 4 in the results for the CVNS network. The CVNS has improved the accuracy of the implementation medium, and makes the analog design more robust compared to regular analog neural networks.

Figure 7.3 shows the allowable error in the synaptic weight implementation using the CVNS approach for various NSR values as a function of number of layers of the Madaline.

The dimensions of the CVNS are chosen as $\beta = 2$ and $L = 2$. For comparison purposes, the NSR of the regular Analog NN Madaline implementation is also shown. It can be noted that the CVNS offers a more relaxed implementation medium, by adding only one extra digit.

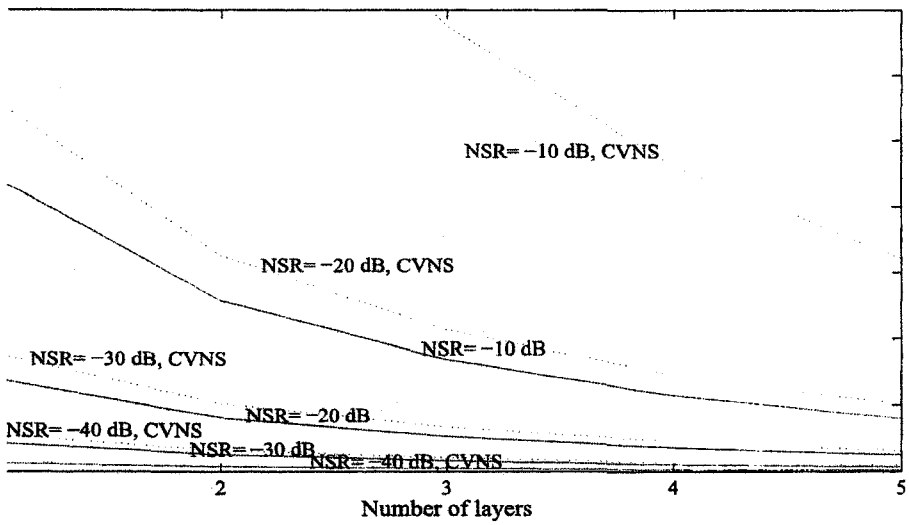


Figure 7.3: Allowable synaptic weight implementation error as a function of number of layers. In this case at each layer there are 256 neurons, weights are limited in the interval [-0.3, 0.3] and first layer inputs are bounded by [-1,1].

Figure 7.4 shows the required equivalent resolution for implementing the Analog NN for various output NSR values are shown. Again, the same function is evaluated for regular Analog NN, and results indicate that the CVNS implementation medium can tolerate a low resolution environment better with only one extra digits.

There is a tradeoff between CVNS radix, and the number of CVNS digits. A high radix increases the system and circuit design complexity. Therefore, it is preferable to choose the radix low, while satisfying the implementation requirements. However, a low radix value requires more digits, hence increasing the implementation cost.

The choice of radix dictates the complexity of the system, a high radix requires more accurate circuits, and minimum value that radix can take is $\beta = 2$. A low radix, decreases

the complexity of the overall design. Number of required CVNS digits is decided from the desirable NSR of the network output from expression (7.14). For example, if network implementation requires equivalent of ℓ-bit resolution, while the implementation medium can offer $t < \ell$ bits resolution, CVNS digits are required to refine the accuracy. The number of required CVNS digits is:

$$L + k = \left\lceil \frac{log(2^{\ell - t})}{log(\beta)} \right\rceil \tag{7.19}$$

Distributing the neurons in the network decreases the sensitivity and NSR of the outputs to the weight errors [21]. This means that implementing large size Analog NN on the analog platform can tolerate more non-ideal environment such as mismatches and nonlinearity in a fabricated chip. Increasing the inputs of the Adaline by a factor S increases the NSR ratio, which means that the effect of weight quantization in a large size network increases.

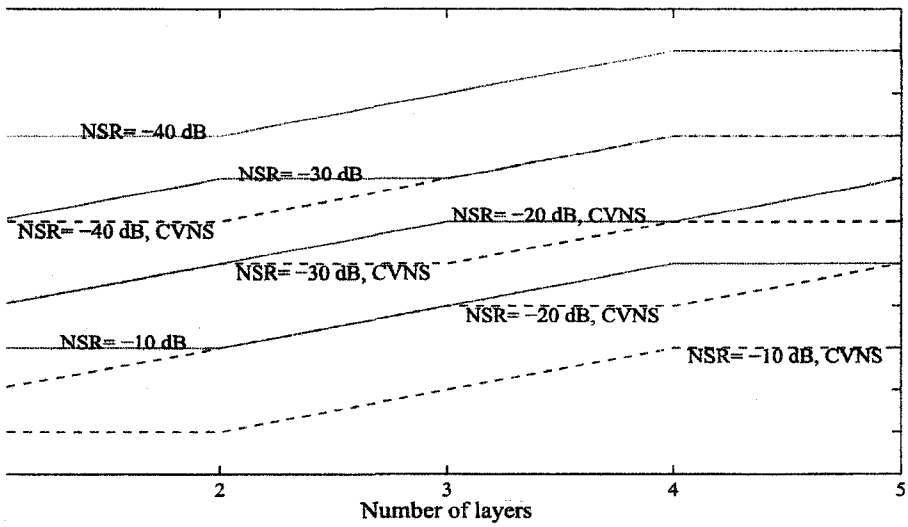


Figure 7.4: Required resolution of the implementation medium as a function of number of layers. In this case at each layer there are 256 neurons, weights are limited in the interval [-0.3, 0.3] and first layer inputs are bounded by [-1,1].

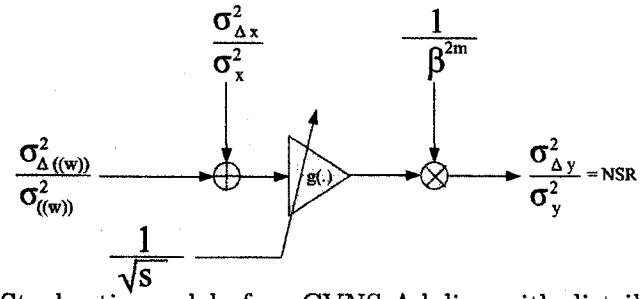


Figure 7.5: Stochastic model of an CVNS Adaline with distributed neurons

By distributing neurons in the network this unwanted effect can be reduced. The NSR of scaled Madaline, NSR_l, and NSR of an scaled CVNS Madaline, NSR_{CV_s} are defined as:

$$NSR_s = g(\sqrt{S.N}\sigma_x\sigma_{((w))}) \times \left(\frac{\sigma^2_{\Delta x}}{\sigma^2_x} + \frac{\sigma^2_{\Delta((w))}}{\sigma^2_{((w))}}\right) \tag{7.20}$$

$$NSR_{CV_s} = \frac{1}{\beta^{2(L+k)}} g\left(\frac{\sqrt{N}\sigma_x\sigma_{((w))}}{\sqrt{S}}\right) \times \left(\frac{\sigma^2_{\Delta x}}{\sigma^2_x} + \frac{\sigma^2_{\Delta((w))}}{\sigma^2_{((w))}}\right) \tag{7.21}$$

where S is the scaling factor.

Figure 7.5 shows the stochastic modeling of distributed neurons. A case study is demonstrated next to show the improvements.

Example:NSR of scaled Madaline In a Madaline suppose the inputs and synaptic weights are uniformly distributed over the range $[-2, 2]$; therefore, $\sigma^2_x = \sigma^2_{((w))} = 4/3$. Assuming a 6-bit quantization scheme, weights are quantized by $q = 1/64$; therefore, weight values are equally spaced from each other by $1/16$, and weight error variance is $\sigma^2_{\Delta((w))} \approx 3.255 \times 10^{-4}$. Output NSR of the lumped neuron of 25-input is:

$$\begin{aligned} NSR &= (0.5 + 0.53\sqrt{25} \times 1.34^2) \times 2.44125 \times 10^{-4} \\ &= 1.2836 \times 10^{-3} \end{aligned}$$

When the number of inputs to the network increases to 100, the NSR of all three approaches are:

$$NSR_s = 2.573 \times 10^{-3} = -26dB$$

$$NSR_{CV_s} = \frac{1}{\beta^{2(L+k)}} 1.283 \times 10^{-3}$$

$$= -20(L+k)log\beta - 29dB$$

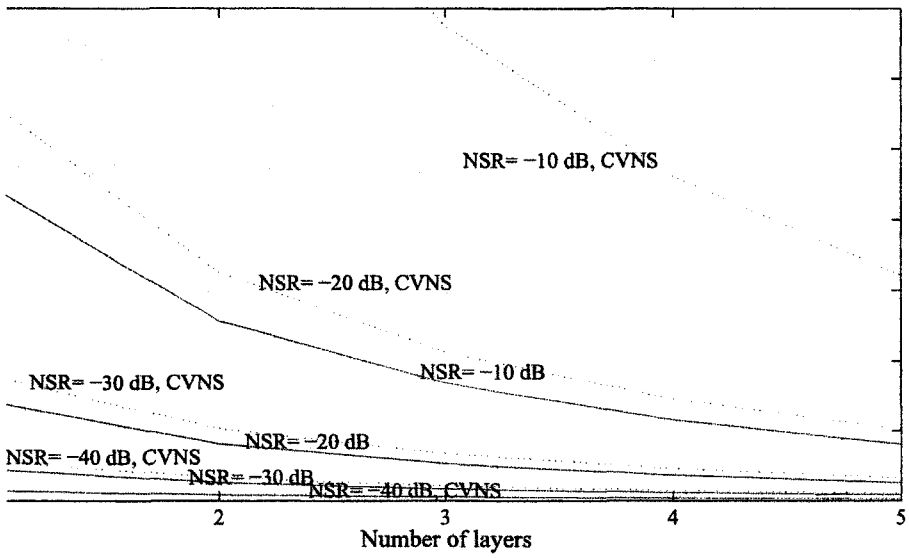


Figure 7.6: Improvements for a CVNS based Madaline for various scaling factors

The CVNS network improvements depend on the number system dimensions. The difference between the two value is:

$$-2(L+k)log\beta - 3 \ dB$$

Figure 7.6 shows the improvements for a CVND Madaline with two analog digits, for various number of neurons based on the scaling factor S. As the number of inputs to the network increases, the noise improvements in the CVNS Madaline becomes more considerable.

7.4 Conclusion

The CVNS representation can increase the precision of analog operations beyond the limitations imposed by the analog environment. We have defined the function evaluation in the CVNS theory, which consists of a look-up in a set of continuous Overlapped Sub Functions (OSF). In particular the MID OSF is proportionate to the original function, and is able to maintain smoothness of the nonlinear function in the MID, and few extra functions increase the accuracy of the continuous MID representation. The function evaluation properties of the CVNS were applied for developing a new kind of neural networks. In this chapter, we also showed that our model for implementing Analog NN using the CVNS is more reliable than classical analog circuits can be used for implementation, however the eventual result shows that this network has less sensitivity to analog imperfections.

Chapter 8

Contributions, Conclusions and Future Work

The idea of the CVNS is taken from a very simple common household item, where operations in this system resemble the operations in reading a utility meter. The nature of the dials in the utility meter is to not only expand the dynamic range, but also to increase the precision of the defined value. The Continuous Valued Number System was developed from this idea to enable its replication in integrated analog circuits, and represented a new paradigm for the implementation of signal processing systems.

Values in the CVNS are represented by continuous-valued digits or "analog-digits", and operations in this number system resemble the general operations of a utility meter. In the CVNS, the ensemble of digits that represents a number, allows for tolerance in the CVNS representation and in the implementation circuitry. Protection against noise and other impairments is warranted through the redundancy among the CVNS digits. This redundancy allows for correction and restoration of the digits to their original value.

Moreover, addition between the CVNS values is almost "carry-free", which means that

the general CVNS operations can be targeted to high speed implementations.

8.1 Contributions

In this dissertation, we have proposed simple and elegant solutions as the second generation designs based on this number system.

In Chapter 3, a simplified conversion technique from binary to CVNS representation, and a new technique for converting the CVNS results back to binary representation has been developed.. A general method for two operand binary addition was also developed, which has resulted in a new VLSI architecture and implementation.. Two alternative methods for generating the adder structure were introduced.

The computational complexity of CVNS adders has been derived, to provide a measure of performance in terms of the VLSI implementation area and delay. This complexity measure has been used to compare different CVNS adder architectures with each other and with also Threshold Logic implementations of binary adders. Finally, we have demonstrated the design of a 16-bit two operand CVNS adder with controlled precision, targeted to the TSMC CMOS $0.18\mu m$ technology. Comparisons between the adders show that the CVNS adder is superior in terms of area, power and reduction of the system and cross talk noise.

Chapter 4 has presented a new and important path for the CVNS. In this chapter, the design and implementation of a mixed-signal 64-bit CVNS adder was developed. The 64-bit CVNS adder is a cascade of four 16-bit adders. The reduced area implementation of the CVNS, along its the compact and low-power properties, makes this adder suitable for multi-media signal processing applications. The 64-bit adder, implemented in TSMC CMOS $0.18\mu m$ technology, provides a worst case delay of 1.5 ns and a static energy dissipation of about 14 pJ with a core area of 13250 μm^2.

Chapter 5, proposed a new type of the CVNS multiplier using the new concept of the CVNS compressors to reduce the partial product columns where values are evaluated in only one gate delay. The final output is generated by using high-speed high-radix CVNS

adders. The new multiplier was designed with the target $0.18\mu m$ CMOS technology and again, favourably compares with standard binary multipliers in the same technology, with the reduced noise advantages.

Chapter 6, presented our new CVNS array multiplier. Instead of using general configuration for CVNS multiplication, arrays of high radix CVNS adders were used to increase the speed of operation, and to reduce the cost of hardware implementation. Two full custom designs have been carried out in CMOS target technology. A 16×16 and a reduced area 8×8, from which the 8×8 array multiplier is currently in fabrication. Moreover, some suggestions were also given for alternative forms of multipliers in the CVSN theory.

In Chapter 7 we have developed an analog neural network implementation that uses direct analog input signals. The function evaluation properties of the CVNS were also used implementing a smooth (differentiable) non-linear function for each neuron. An CVNS Adaline neuron, was proposed, and we have used stochastic models of a CVNS Madaline to obtain suitable levels of robustness for the network. The error analysis indicates that the network is able to better handel implementation errors better in comparison to a classical analog neural networks.

In summary, the contributions of this dissertation are listed below:

- Novel addition approaches are proposed for feasible implementation of CVNS. [53]
- System complexity equations are derived, and compared with similar architectures. [48]
- New and more efficient methods for radix conversion and CVNS to binary conversion methods have been developed. [53, 48]
- Efficient circuits have been developed, to increase the operational speed and to reduce the power consumption. [53, 48]
- A full custom design example adder was designed and simulated to demonstrate the proposed approach. [53, 48]

- A new reconfigurable adder, with on-demand resolution. [52]
- Four general classes of novel CVNS multiplication schemes are proposed.
- An array multiplier with a regular architecture has been implemented in a CMOS $0.18\mu m$ technology and compared with standard digital designs. [50]
- A prototype of an 8×8 array multiplier has been design, simulated and successfully sent for fabrication.
- New CVNS compressors based on the modular reduction arithmetic are proposed, and applied to the implementation of an efficient CVNS Tree Multiplier. [51]
- Additional multiplication schemes including the MID switching for truncated multiplication and a more adaptable CVNS multiplier have been proposed.
- The effect of implementation errors in analog environment with limited precision has been studied. [49]
- The effect of an imperfect analog environment on CVNS digits has been explored. [49]
- The correction of digits in the CVNS has been compared with MVL correction. [49]
- Analog complex function evaluation based on the CVNS is studied, which allows the future implementation of completely analog arithmetic units [47]
- A CVNS Adaline is proposed to be used in a CVNS Neural Network.
- Based on stochastic modeling of the CVNS Analog NN, we have demonstrated that the CVNS offers a more robust implementation. [47]

8.2 Future Research

The field of the CVNS theory is very new, and is able to offer a wide host of new solutions for various applications. In this section we offer a few suggestions for the future research on

this number system.

Today, signal processing and arithmetic operations rely almost exclusively on digital processors and computers. On the other hand, almost all forms of data from the outside world are in analog form (music, image, voice and so on). To process these data in regular digital signal processing systems, these analog signals have to be digitized, and after the processing, results are converted back into the analog form again. The CVNS can offer an elegant solution to the implementation of such a processor.

Most of this dissertation, with the exception of Chapter 7, only deals with systems that have quantized input and output data. A proper starting point for future research is to deal directly with analog signals instead of utilizing quantized information. A good starting point for a fully analog design a continuation of the work we have started on the Madaline network.

Yet another important feature of the CVNS is the error correction method, which requires further attention. Error correction plays an important role in the CVNS operations. It is important to remember that the role of error correction should always be to fine-tune the outcome of the operations, and to be optional. Error correction should not be replaced with the actual arithmetic operations. This fact been demonstrated in the design of the CVNS adder, where the delay and area of the adder has been increased for the group adder.

Alternative methods for error correction are specially useful for storage and memory applications. Storage of the CVNS values is an important area of research, and needs to be addressed for fully analog CVNS systems implementation.

To implement denser and more reliable multi-level memories, non traditional and novel solutions are needed. The CVNS which is built on analog platforms uses the redundancy among the digits to detect and correct multiple errors. The Reverse Evolution in the Continuous Valued Number System, which is used for detecting and correcting multiple errors in the system may provide a novel solution to the reliability of multi-level memory systems.

8.3 Final Conclusion

In this dissertation, various arithmetic units based on the CVNS have been proposed. We have compared all of these structures with the state of the art digital designs. The CVNS is able to offer great savings in terms of area and power consumption. The reduced area of the CVNS implementations in this dissertation are due to the the use of current mode circuitry, where summation is performed by hardwiring the nodes. Differential style circuit design has resulted in circuits with higher speed of operation and increased noise margin of the analog circuits. The CVNS implementations delay is comparable to the static digital designs.

With no guarantee on the end results and without any existing background information on the potentials that this approach may offer, the current research was initially motivated by academic curiosity.

Our results have finally demonstrated the potential superiority of the CVNS compared to conventional digital systems, where the reduced noise advantages of analog implementations can offer an attractive alternative for realizing advanced signal processing.

CVNS theory is still quite new and there are still many unknown possibilities, which require further investigation. Our results show that signal processing based on the CVNS is able to offer implementation savings, and may be a future alternative for some integrated microsystems that require low noise computer arithmetic.

In our research, we have explored and generalized the CVNS theory and proposed several high performance implementations for computer arithmetic and signal processing.

This CVNS project has been a part of an NSERC-CRD research grant between Gennum Corporation, Research Centre for Integrated Microsystems at the University of Windsor and the ATIPS Laboratories at the University of Calgary. The development of the CVNS along with other projects in our research centre were reported on a regular basis. This research has also been featured in the 2004 issue of the University of Windsor Alumni Magazine, the 2005-2006 Ontario Research Council Success Stories, and the annual iCORE (Alberta) Research Reports.

References

[1] A. Alippi and L. Brizzo. Accuracy vs. precision in digital vlsi architectures for signal processing. *IEEE Transaction on Neural Networks*, 47(4):472–477, 1998.

[2] A. G. Andreou, K. A. Boahen, and P. O. Pouliquen. Current mode subthreshold MOS circuits for analog vlsi neural systems. *IEEE Transactions on Neural Networks*, 2(2):205–213, 1991.

[3] Y. Arima, M. Muraski, T. Yamada, A. Maeda, and H. Shinohara. A refreshable analog VLSI neural network chip with 400 neurons and 40k synapses. *IEEE Journal of Solid State Circuits*, 27(12):1854–1861, 1992.

[4] R. Aroca, M. Ahmadi, R. Hashemian, and G. A. Jullien. A B-compliment continuous valued digit adder. *Proc. IEEE 9th Int. Conf. Elec. Cir. and Sys.*, 2:433–436, 2002.

[5] K. Basterretxea, J. M. Tarela, and I. del Campo. Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons. *IEE Proceedings on Circuits, Devices and Systems*, 151(1):8–24, 2004.

[6] P. Beame, E. Brisson, and R. Ladner. The complexity of computing symmetric functions using threshold circuits. *Theoretical Computer Science*, 100:63–71, 1992.

[7] O. Bedrij. Carry-select adder. *IRE Trans. Electron Computers*, EC-11:340–346, June, 1962.

[8] V. Beiu. Low-power differential conductance-based logic gate and method of operation thereof. *U.S patent 6 580 296*, June, 2003.

[9] S. Bettola and V. Piuri. High performance fault-tolerant digital neural networks. *IEEE Transaction on Computers*, 5(23):230–233, 1997.

[10] R. Brent and H. Kung. A regular layout for parallel adders. *IEEE Trans. Computers*, 31-3:260–264, March 1982.

[11] T. K. Callaway and Earl E. Swartzlander. Power delay characteristics of cmos multipliers. *IEEE 13th Asilomar Conf. on Sig., Sys. and Comp.*, pages 26–32, July 6-9, 1997.

[12] C-H. Chang, J. Gu, and M. Zhang. Ultra low-voltage low-power cmos 4-2 and 5-2 compressors for fast arithmetic circuits. *IEEE Trans. on Cir. and Sys. I: Regular Papers*, 51(10):1985–1997, Oct. 2004.

[13] Chip-Hong Chang, Ravi K. Satzoda, and S. Sekar. A novel multiplexer based truncated array multiplier. *IEEE Int. Symp. on Cir. and Sys. (ISCAS)*, 1:85–88, 2006.

[14] L-H. Chen, Oscal T.C-Chen, and R-L. Ma. A high efficiency reconfigurable digital signal processor for multimedia computing. *IEEE Int. Symp. on Cir. and Sys.(ISCAS)*, 2:768–771, 2003.

[15] Oscal T. C. Chen, S. Wang, and Yi-Wen Wu. Minimization of switching activities of partial products for designing low-power multipliers. *IEEE Trans. on VLSI Systems*, 11(3):418–433, 2003.

[16] A. Y. Cheng and D. S. Yeung. Sensitivity analysis of neocognitron. *IEEE Trans. Syst., Man, Cybern. C*, 29:238249, 1999.

[17] J. Choi and B. J. Sheu. A high precision VLSI winner take all circuit for self-organizing neural networks. *IEEE Journal of Solid State Circuits*, 28(2):576–583, 1993.

[18] J. Y. Choi and C.-H. Choi. Sensitivity analysis of multilayer perceptron with differentiable activation functions. *IEEE Trans. Neural Networks*, 3:101107, 1992.

[19] M.-Y. Chow and S. O. Tee. A measure of relative robustness for feedforward neural networks subject to small input perturbations. *Int. J. Neural Syst.*, 3(3):291299, 1992.

[20] P. Corsonella, S. Perri, and M. Margala. Efficient addition circuits for modular design of processors-in-memory. *IEEE Trans. on Cir. and Sys.-I Regular papers*, 52(8):1557–1567, 2005.

[21] H. Djahanshahi, M. Ahmadi, G. A. Jullien, and W. C. Miller. Quantization noise improvement in a hybrid distributed-neuron ANN architecture. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(9):842 – 846, 2001.

[22] M. D. Ercegovac and T. Lang. Digital arithmetic. *Morgan Kaufmann Publishers, Elsevier Science Ltd*, 2004.

[23] A. A. Farooqui, V .G. Oklobdzija, and F. Chechrazi. 64-bit media adder. *IEEE Int. Symp. on Cir. and Sys.(ISCAS)*, pages 100–103, 1999.

[24] A. A. Farooqui, V. G. Oklobdzija, and F. Chechrazi. Multiplexer based adder for media signal processing. *IEEE Int. Symp. on VLSI Tech., Sys., and App.*, pages 100–103, 1999.

[25] A. Goldovsky, B. Patel, M. Schultet, R. Kolagotlao, H. Srinivas, and G. Burns. Design and implementation of a 16 by 16 low-power twos complement multiplier. *IEEE Int. Symp. on Cir. and Sys.*, 5:345–348, 2000.

[26] R. Gonzales, B. M. Gorodn, and M. Horwitz. Supply and threshold voltage scaling for low power cmos. *IEEE J. Solid State Circuits*, 32:1210–1216, 1997.

[27] R. Gonzales and M. Horwitz. Energy consumption in general purpose microprocessors. *IEEE J. Solid State Circuits*, 31:1277–1284, 1996.

[28] S. M. Gowda, B. J. Sheu, and J. Choi. Design and characterization of analog neural network modules. *IEEE Journal of Solid State Circuits*, 28(3):301–303, 1992.

[29] T. Han and D. Carlson. Fast area efficient vlsi adders. *Proc. 8th Symp. Comp. Arith.*, pages 49–56, Sept. 1987.

[30] D. Harris and I. Sutherland. Logical effort of carry propagation adders. *IEEE Proc. of the Thirty-Seventh Asilomar Conf. on sig. sys. and comp.*, 1:873–878, Nov. 2003.

[31] H. Hassan, M. Anis, and M. Elmesry. Mos current mode circuits: analysis, design and variability. *IEEE Trans. on Very Large Scale Integration (VLSI) systems*, 13(8):885–898, 2005.

[32] Y. He and U. Cilinhiroglu. A charge-based on-chip adaptation kohonen neural network. *IEEE Transaction on Neural Networks*, 4(3):462–469, 1993.

[33] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Oklobdzija, R. K. Krishnamurthy, and S. Y. Borkar. A 110 gops/w 16-bit multiplier and reconfigurable pla loop in 90-nm cmos. *IEEE Journal of Solid-State Cir.*, 41-1:256 – 264, 2005.

[34] Zh. Huang and M. D. Ercegovac. High performance low-power left-to-right array multiplier design. *IEEE Trans. on comp.*, 54(3):272–283, 2005.

[35] Y. Ibrahim, G. A. Jullien, and W. C. Miller. Ultra low noise signed digit arithmetic using cellular neural networks. *Proc. IEEE 4ht Int. workshop on System-on-Chip for Real-Time Applications*, pages 136–142, July, 2004.

[36] Y. Ibrahim, W. C. Miller, G. A. Jullien, and V. S. Dimitrov. DBNS addition using cellular neural networks. *IEEE Int. Symp. on Cir. and Sys.*, 4:3914–3917, May, 2005.

[37] M. Jabri and B. Flower. Weight perturbation: An optimal architecture and learning technique for analog feed forward and recurrent multilayer networks. *IEEE Transaction on Neural Networks*, 3(1), 1992.

[38] Jung-Yup Kang and Jean-Luc Gaudiot. A simple high-speed multiplier design. *IEEE Trans. On Computers*, 55(10):1253–1258, 2006.

[39] A. Khatibzadeh and K. Raahemifar. A novel design of a 6-ghz 8 × 8-b pipelined multiplier. *Proceedings. Fifth International Workshop on System-on-Chip for Real-Time Applications*, pages 387– 391, July 2005.

[40] S. Knowles. A family of adders. *Proc. 15th IEEE Symp. Comp.*, pages 277–281, June, 2001.

[41] P. Kogge and H. Stone. A parallel algorithm for the efficient solution of a general class of recurrence relations. *IEEE Trans. Computers*, C22-8:786–793, Aug. 1973.

[42] R. Ladner and M. Fisher. Parallel prefix computation. *J. ACM*, 27-4:831–838, Oct. 1980.

[43] B. W. Lee and B. J. Sheu. General purpose neural chips with electrically programmable synapses and gain adjustable neurons. *IEEE Journal of Solid State Circuits*, 27(9):1299–1302, 1992.

[44] M. Lehman and B. Burla. Skip techniques for high speed carry propagation in binary arithmetic units. *IRE Trans. Electron Computers*, EC-10:226–231, Dec. 1962.

[45] J-F. Li, J-D. Yu, and Y-J. Huang. A design methodology for hybrid carry-lookahead/carry select adders with reconfigurability. *IEEE Int. Symp. on Cir. and Sys.(ISCAS)*, 1:77–80, 2005.

[46] M. C. Mekhallalati and M. K. Ibrahim. A new high radix maximally redundant signed digit adder. *Proc. IEEE Int. Symp. on cir. and sys.*, 1:459–462, May, 1999.

[47] M. Mirhassani, M. Ahmadi, and G. A. Jullien. Robust low-sensitivity adaline neuron based on continuous valued number system. *Submitted to Analog Integrated Circuits and Signal Processing*, , 14 pages, submission date: May, 2007.

[48] M. Mirhassani, M. Ahmadi, and G. A. Jullien. Continuous valued digit adders for two operand binary addition. *Submitted to IEEE Transaction on Computers*, 14 pages, submission date: January, 2007.

[49] M. Mirhassani, M. Ahmadi, and G. A. Jullien. Fault tolerant arithmetic of continuous valued number system. *Submitted to IEEE Transaction on Computers*, 14 pages, submission date: January, 2007.

[50] M. Mirhassani, M. Ahmadi, and G. A. Jullien. 16-bit binary mutliplication using high radix analog digits. *IEEE Asilomar Conf. on Sig., Sys., and Comp.*, 9:4 pages, 2006.

[51] M. Mirhassani, M. Ahmadi, and G. A. Jullien. Digital multiplication using continuous valued digits. *IEEE Int. Symp. on Cir. and Sys. (ISCAS)*, pages 363–366, 2007.

[52] M. Mirhassani, M. Ahmadi, and G. A. Jullien. Reconfigurable 64-bit binary adder based on continuous digits. *Submitted to IEEE Transaction on Very Large Scale Integration (VLSI) System*, 9 pages, submission date: April, 2007.

[53] M. Mirhassani, M. Ahmadi, and G. A. Jullien. 16-bit radix-4 continuous valued digit adder. *Proc. SPIE, Advanced Sig. Process. Alg., Arch., and Imp. XVI*, 6313-03:12 pages, Aug. , 2006.

[54] R. Mudassir, H. El-Razouk, and Z. Abid. New designs of signed multipliers. *The 3rd International IEEE-NEWCAS Conference.*

[55] A. Neve, H. Schettler, T. Ludwig, and D. Flandere. Power-dealy minimization in high performance 64-bit adder carry-select adders. *IEEE Trans. on Very Large Scale Integration (VLSI) systems*, 12(3):235–244, 2004.

[56] S.-H. Oh and Y. Lee. Sensitivity analysis of a single hidden-layer neural networks with threshold function. *IEEE Trans. Neural Networks*, 6:10051007, 1995.

[57] A. Peleg and U. Weiser. Mmx technology. *IEEE Micro*, pages 42–50, 1996.

[58] R. Perfetti and E. Ricci. Anoher k-winners-take-all analog neural network. *IEEE Journal of Solid State Circuits*, 11(4):357–363, 1998.

[59] R. Perfetti and E. Ricci. Analog neural network for support vector machine learning. *IEEE Journal of Solid State Circuits*, 17(4):829–836, 2000.

[60] S. Perri, P. Corsonella, and G. Cocorullo. A 64-bit reconfigurable adder for low power media processing. *Electron. Lett.*, 38(9):397 – 399, 2002.

[61] S. Perri, P. Corsonella, and G. Cocorullo. A high speed energy efficient 64-bit reconfigurable binary adder. *IEEE Trans. on Comp.*, 11(5):939–943, 2003.

[62] S. W. Piche'. The selection of weight accuracies for madaline. *IEEE Transaction on Neural Network*, 6(2):432–445, 1995.

[63] J. M. Quintana, M. J. Avedillo, E. Rodrigues-Villegas, and A. Rueda. Threshold-logic based design of compressors. *9th Int. Con. on Elec., Cir. and Sys.*, 2:661 – 664, 2002.

[64] J. F. Ramos and A. Gago. Two operand binary adders with threshold logic. *IEEE Trans. on Comp.*, 48(12):1324 – 1337, 1999.

[65] J. F. Ramos, J. Hidalgo, M. J. Martin, J. C. Terjero, and A. Gago. A threshold logic gate based on clocked-coupled inverters. *Int. J. on Electronics*, 89(4):253–265, 1992.

[66] A. Saed, M. Ahmadi, and G. A. Jullien. Arithmetic circuit analog digits. *IEEE Int. Symp. on Multiple valued logic*, pages 186–191, 1999.

[67] A. Saed, M. Ahmadi, and G. A. Jullien. A number system with continuous valued digits and modulo arithmetic. *IEEE Trans. on Comp.*, 51-11:1294–1304, NOv. 2002.

[68] A. Saed, M. Ahmadi, G. A. Jullien, and W. C. Miller. Overlap resolution: Arithmetic with continuous valued digits for hybrid architectures. *Proc. IEEE Midwest Symp. Cir. and Sys.*, 1:377–380, 1997.

[69] A. Saed, M. Ahmadi, G. A. Jullien, and W. C. Miller. Overlap resolution: Arithmetic with continuous valued digits for hybrid architectures. *IEEE Asilomar Conf. on Sig., Sys., and Comp.*, 2:1188–1191, 1997.

[70] A. Saed, M. Ahmadi, G. A. Jullien, and W. C. Miller. Circuit tolerances and word lengths in overlap resolution. *IEEE Int. Symp. on Cir. and Sys.*, 1:197–200, 1998.

[71] Tze-Yee Sin, EMC. Wong, and I. C. C. Jong. A 1.6-ghz 16x16b asynchronous pipelined multiplier. *Proc. IEEE Midwest Symp. Cir. and Sys.*, 1:336–339, 2001.

[72] J. Sklansky. Conditional-sum addition logic. *IRE. Trans. Electron Computers*, EC-9:226–231, June 1960.

[73] V. Srinivasan, S. H. Dong, and J. B. Sulistyo. Gigahertz-range mcml multiplier architectures. *Proceedings of the 2004 International Symposium on Circuits and Systems*, 2:785–8, May, 2004.

[74] Venkat Srinivasan, Dong S. Ha, and Jos B. Sulistyo. Gigahertz-range mcml multiplier architectures. *IEEE Int. Symp. on Cir. and Sys.*, 2:785–788, 2004.

[75] S. Srivanasont and A. Surarerks. Redundant analog number system. *TENCON 2004. 2004 IEEE Region 10 Conference*, 2:170–182, NOv. 2004.

[76] Paul F. Stelling and Vojin G. Oklobdzija. Implementing multiply-accumulate operation in multiplication time. *Poc. 13th IEEE Symp. on Comp. Arith.*, pages 99–106, 1997.

[77] I. M. Thoidis, D. Soudris, J. M. Fernandez, and A. Thanailakis. The circuit design of multiple-valued logic voltage-mode adders. *Proc. IEEE Int. Symp. on cir. and sys.*, 4:162–165, May, 2001.

[78] M. Tremblay, J. M. O'Conner, V. Narayanan, and H. Liang. Vls speeds new media processing. *IEEE Micro*, 16:10–20, 1996.

[79] A. Tyagi. A reduced area scheme for carry-select adders. *IEEE Trans. Computers*, 42-10:1162–1170, Oct. 1993.

[80] J. Um and T. Kim. Optimal bit-level arithmetic optimization for high-speed circuits. *IEE Electronic Letters*, 36(5):405–407, 2000.

[81] F. Vasefi and Z. Abid. Low power n-bit adders and multiplier using lowest-number-of-transistor 1-bit adders. *IEEE Proc. Canadian Conf. on Elec. and Comp. Eng.*, pages 1731–1734, 2005.

[82] S. Vassiliadis and S. Cotofana. 7—2 counters and multiplication with threshold logic. *IEEE Thirtieth Asilomar Conf. on Sig., Sys. and Comp.*, 1:192–196, 1996.

[83] S. Vassiliadis, S. Cotofana, and K. Bertels. 2-1 addition and related arithmetic operations with threshold logic. *IEEE Trans. on Comp.*, 45(9):62–67, 1996.

[84] I. Wegener. The complexity of boolean functions. *Chapter 1*, pages 1–19, 1998.

[85] A. Weinberger and J. L. Smith. A logic for high speed addition. *Nat. Bur. Stand. Circ.*, 591:3–12, 1958.

[86] C-L. Wey and J-F. Li. Design of reconfigurable array multipliers and multiplier-accumulator. *The 2004 IEEE Asia-Pacific Conf. on Cir. and Sys.*, pages 37–40, 2004.

[87] B. Widrow and M E Hoff. Adaptive switching circuits. *IRE WESCON enConvetion Record*, IRE Part 4:96–104, 1960.

[88] A. Wroblewski, M. Wroblewski, C. Saas, and Josef A. Nossek. Reduced binary tree fir filters. *Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS)*, 2.

[89] L. Yang, D. Hu, Y. Luo, and X. Zhang. Robustness analysis of feedforward neural networks composed of threshold neurons. *Proc. IEEE Int. Conf. Intell. Processing Syst.*, 1:502506, 1997.

[90] X. Zeng and D. S. Yeung. Sensitivity analysis of multiplayer perceptron to input and weight perturbation. *IEEE Transaction on Neural Networks*, 12(6):1358–1366, 2001.

[91] D. Zhang and M. I. Elmasry. Vlsi compressor design with applications to digital neural networks. *IEEE Trans. on VLSI sys.*, 5-2:533–535, 1997.

[92] D. Zhang and M. I. Elmasry. VLSI compressor design with applications to digital neural networks. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, 47(3):1085–1091, 2006.

[93] R. Zimmermann. Non-heuristic optimization and synthesis of parallel-prefix adders. *Proc. Intl. Workshop on Logic and Architecture Synthesis*, pages 123–132, Dec. 1996.

Appendix A

CVNS Error Correction

The CVNS error correction allows for digit correction and restoration. The error correction of the CVNS digits, allows for more tolerance in the analog representation and implementation. In this appendix, the error correction features of the CVNS are presented in more details.

A.1 Introduction

Although it has been mentioned that the CVNS is able to recover from possible errors caused by the environment in which it has been implemented [67], its nature has not been explored in details yet. In this dissertation, the error recovery mechanism of this number system is examined comprehensively. The error threshold for implementing digits, and the effect of the CVNS dimensions (radix and number of digits) on error recovery are studied. This study is able to provide a better understanding of implementation medium requirements. Error correction is studied here, in the context of its ability to enhance the CVNS representation due to reduced noise margin. The error correction presented in this chapter is an essential part of the CVNS representation, and it is required to develop mathematical and theoretical

foundations for it. Therefore, we start to look at the integrity of the CVNS representation considering various types of errors. The sources of such errors could be from nonlinear circuits, fabrication errors and system noise.

Section A.2 deals with different types of error that may affect the digits. The error correction concept starts by looking at linearly additive style errors in Section A.3, and in Section A.10 a more robust method for removing the errors is given.

A.2 Error

A CVNS digit can be affected by three form of errors:

- A digit error which is added linearly into the analog digit which results in an erroneous digit value of

$$(\!(\tilde{x})\!)_i = (\!(x)\!)_i + \bar{\varepsilon}_i \tag{A.1}$$

  As a result, the digit value may go beyond the range of $\pm\beta$. This type of error is called *linearly additive-style* error, which is usually caused by the implementation medium. The error source can be linear or nonlinear source of error. However, in the CVNS all of these errors are called linearly additive due to their effect on the CVNS digits. In this section we focus on removing this type of error from any CVNS digit, and develop a non-iterative method which is able to enhance the representation.

- A digit error may be added to the digit values of modulus β which results in an errored digit as follows:

$$(\!(\tilde{x})\!)_i = \big((\!(x)\!)_i + \varepsilon_i^{\circ}\big) mod \beta \tag{A.2}$$

  In this case, the digit value never reaches the $\pm\beta$. *modularly additive-style* errors are the result of imperfect CVNS operations. Many of the linearly additive-style error correction properties can be applied to this type of error as well.

- A digit error of precisely an integer multiple of the radix is added to the digit which results in an errored CVNS digit as below:

$$(\!(\tilde{x})\!)_i = (\!(x)\!)_i + I_{\varepsilon_i}\beta \tag{A.3}$$

  Errors from an imperfect modulus operation may introduce such *congruence* errors into the digit.

An additive error which is composed of all three type of errors and corresponding relative error are as follows:

$$\varepsilon_{(\!(x)\!)_i} = \bar{\varepsilon}_i + (\varepsilon_i^\circ) mod\beta + I_{\varepsilon_i}\beta \tag{A.4}$$

$$\check{\varepsilon}_{(\!(x)\!)_i} = \frac{\varepsilon_{(\!(x)\!)_i}}{\beta} \tag{A.5}$$

As a result, the digit value may go beyond $\pm\beta$ range. The three errors are defined in Fig. A.1, for $0 < \bar{\varepsilon}_i \ll \beta$, $0 < \varepsilon_i^\circ \ll \beta$, and $I_{\varepsilon_i} = -1$. From this figure, it is evident that a specific error can be decomposed into various combinations of these error styles, and hence is not unique. It is not important, however, to find the composition of the errors, and they can be removed. The CVNS digit error and the relative error which is defined as error related to the radix, are as follows:

The effectiveness of the proposed error correction will be optimized by measuring the *Probability Density Function* (PDF) of error, to obtain an efficient method for reducing the error, and increasing the probability of success. The three main properties of the error PDF are as follows:

PDF

In any PDF, a value ε_0 can always be found such that the occurrence of values less than ε_0 are more probable than occurrence of values greater than ε_0.

$$\int p(\varepsilon)d\varepsilon = 1 \Rightarrow \forall p(\varepsilon) : \exists \varepsilon_0 > 0 : \int_{-\varepsilon_0}^{\varepsilon_0} p(\varepsilon)d\varepsilon > \left(\int_{\infty}^{-\varepsilon_0} p(\varepsilon)d\varepsilon + \int_{\varepsilon_0}^{\infty} p(\varepsilon)d\varepsilon\right) \tag{A.6}$$

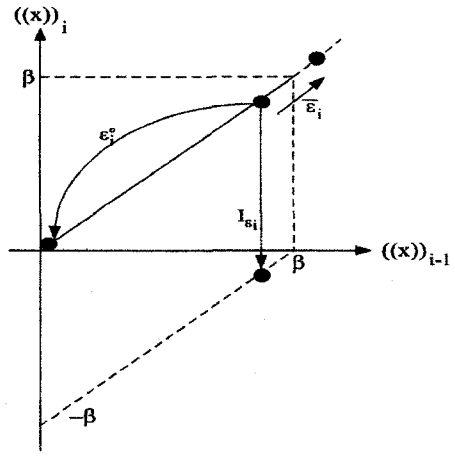


Figure A.1: CVNS digit errors: Linearly additive-Style, Modularly additive-Style and Congruence errors

In practical electronic impl$\int_{|\varepsilon|<\varepsilon_0} p(\varepsilon)d\varepsilon \gg \int_{\varepsilon_0<|\varepsilon|<Q} p(\varepsilon)d\varepsilon$rrors often occur more than large error values:

$$\int_{|\varepsilon|<\varepsilon_0} p(\varepsilon)d\varepsilon \gg \int_{\varepsilon_0<|\varepsilon|<Q} p(\varepsilon)d\varepsilon \tag{A.7}$$

where Q is the maximum value of representation quantity in which the CVNS is implemented. If most of the errors happen between the 0 and ε_0, then $\int_{|\varepsilon|<\varepsilon_0} p(\varepsilon)d\varepsilon \approx 1$.

It is not required to have $p(\varepsilon)$ equal to zero for any value or range of ε, as occurs for instance outside of the boundaries of a uniform distribution, or that PDF be clock shaped as in the case of Gaussian distribution. It is only required that $\varepsilon_0 \ll Q$.

Descriptive Point

The PDF of the error probability, $p(\varepsilon)$, can be compared by what is called a Descriptive Point (DP), ε_0, of a specific PDF function. For a given ε_0, summation $P_{\varepsilon_0} = \int_{\varepsilon_0<|\varepsilon|} p(\varepsilon)d\varepsilon$ is computed for each PDF, and results are compared. The PDF with the smallest P_{ε_0} is more desirable. Similarly, we can obtain the corresponding DP for any PDF. In general, it is desirable to have P_{ε_0} as small as possible, in which case, ε_0 can be taken as a multiple of

the standard deviation of error.

Narrowing the PDF

An important feature is that if a PDF is narrowed by a factor $\lambda > 1$ towards the vertical axis such that $p'(\varepsilon) = \lambda.p(\varepsilon.\lambda)$, then $\int_{-\infty}^{\infty} p'(\varepsilon)d\varepsilon = 1$ is maintained. The PDF of the $p'(\varepsilon)$ has an equal $P_{\varepsilon_0'}$ for a stricter DP which is equal to $\varepsilon_0' = \varepsilon_0/\lambda$. This means that for error PDF any operation that narrows the function, is a more favorable PDF, with a much smaller standard deviation of $\sigma' = \sigma/\lambda$. A widened PDF of $p'(\varepsilon) = \lambda^{-1}.p(\varepsilon)$, has an equal P_{ε_0} for $\varepsilon_0' = \varepsilon_0\lambda$ with a larger standard deviation of $\sigma' = \lambda\sigma$.

These properties are used to evaluate the effectiveness of error correction in the CVNS. In the next section, a method is proposed for linearly additive-style error correction and modularly additive-style error is explained later.

A.3 Linearly Additive-Style Error

Correction of the digits starts from the less informed digits, and then applied to more informed digits. The starting point is by checking the relation between any two adjacent digits, which was given in (2.9). The linearly additive-style CVNS errored version of two neighboring digits are $(\!(\tilde{x})\!)_i = (\!(x)\!)_i + \bar{\varepsilon}_i$ for the more informed digit, and $(\!(\tilde{x})\!)_{i-1} = (\!(x)\!)_{i-1} + \bar{\varepsilon}_{i-1}$ for the less informed digit. A successful error correction is able to recover the associated integer correctly or in other word $\lfloor (\!(\hat{x})\!)_i \rfloor = \lfloor (\!(x)\!)_i \rfloor$.

As an initial attempt, we use the floor function to obtain the integer from the errored CVNS digits or in other words $\lfloor (\!(\hat{x})\!)_i \rfloor = \lfloor (\!(\tilde{x})\!)_i \rfloor$. In order to obtain the recovered digit, $(\!(\hat{x})\!)$, errored digits are placed in (2.9) as follows:

$$(\!(\hat{x})\!)_i = \lfloor (\!(\tilde{x})\!)_i \rfloor + \frac{(\!(\tilde{x})\!)_{i-1}}{\beta} \tag{A.8}$$

Successful error recovery depends on both correct integer extraction and existing error in the less informed digit. It is required that the result of the floor function to be a close

approximation of the original CVNS integer. Only when the original CVNS value lies closely to an integer, the symmetric error $\overline{\varepsilon}_i$ is small enough, and floor function results in correct extraction of the integer. The error which comes from the lower less informed digit denoted by ε'_i, is the correction error and is equal to:

$$\varepsilon'_i = \frac{\overline{\varepsilon}_{i-1}}{\beta} \tag{A.9}$$

Only for error values such that $\overline{\varepsilon}_{i-1} < \overline{\varepsilon}_i\beta$, the digit error has decreased. Properties of error PDF are used to evaluate the effect of error correction using the regular floor function, and to compute the error PDF of the new CVNS digit. The PDF of the error of the corrected digit, $p(\varepsilon'_i)$, is now a narrower version of the PDF of the less informed digit, which is the result of multiplication by the radix ($\beta \geq 2$). The error PDF of the recovered digit, using the the floor function is:

$$p(\varepsilon'_i) = \beta.p(\overline{\varepsilon}_{i-1}.\beta) \tag{A.10}$$

The probability of success is:

$$P\big(\lfloor((\hat{x}))_i\rfloor = \lfloor((x))_i\rfloor\big) = \int_{\lfloor((x))_i\rfloor-((x))_i}^{1+\lfloor((x))_i\rfloor-((x))_i} p(\varepsilon'_i).d\varepsilon'_i \tag{A.11}$$

For a mid-integer CVNS digit value of $((\tilde{x}))_i = I + \frac{1}{2}$, where $I = 0, \ldots, \beta - 1$, we conclude that $P\big(\lfloor((\hat{x}))_i\rfloor = \lfloor((x))_i\rfloor\big) = 1$. However, for integer CVNS values of $((\tilde{x}))_i = 0, \ldots, \beta$, this probability is reduced to $\frac{1}{2}$. To enhance the results, we have to find an approach that delivers $P\big(\lfloor((\hat{x}))_i\rfloor = \lfloor((x))_i\rfloor\big) = 1$ for all CVNS values. The relation between the CVNS digits can be used to develop a more accurate approach in reducing the error.

The difference between the two values of $((x))_i$ and $((x))_{i-1}/\beta$ is ideally the integer value of $\lfloor((x))_i\rfloor = ((x))_i - \frac{((x))_{i-1}}{\beta}$. This can be used as the starting point to find a more accurate approximation of the associated integer by using a rounded floor function. The operation of rounding is to find the closest integer to a real value. On the ring this means finding the closest integer on the range to the point in question. The rounding on the real ring for any positive or negative value of a is defined as:

$$\lfloor((a))_i\rfloor_R = \left[((a))_i - \frac{((a))_{i-1}}{\beta}\right]_R \tag{A.12}$$

Rounding can be defined for real values outside the set $[0, \beta)$, however, the integer outcome has to be within the ring range for the CVNS application. The CVNS digits are placed into above equation to obtain a better approximation of the associated integer value, as follows:

$$\begin{aligned}\lfloor((\hat{x}))_i\rfloor_R &= \left[((\tilde{x}))_i - \frac{((\tilde{x}))_{i-1}}{\beta}\right]_R \\ &= \left[((x))_i - \frac{((x))_{i-1}}{\beta} + \bar{\varepsilon}_i - \frac{\bar{\varepsilon}_{i-1}}{\beta}\right]_R\end{aligned} \tag{A.13}$$

For bounded error values, it can be concluded that:

$$\left|\bar{\varepsilon}_i - \frac{\bar{\varepsilon}_{i-1}}{\beta}\right| < \frac{1}{2} \Rightarrow \lfloor((\hat{x}))_i\rfloor = \lfloor((x))_i\rfloor = \left(((x))_i - \frac{((x))_{i-1}}{\beta}\right) \tag{A.14}$$

Under the condition in (A.14), this process will always yield to a close approximation of the integer value $\lfloor((\hat{x}))_i\rfloor \leq \beta$. This condition is always true regardless of whether either of the errored digits $((\tilde{x}))_i$ or $((\tilde{x}))_{i-1}$ exceed $\pm\beta$.

After the correction it is necessary to check for the digits for saturation. It is possible for digits to overflow and pass the boundaries, if implementation medium allows it. This will happen more for digits that are close to the lower or upper boundary set. This issues will be solved later by a more robust error correction in Section A.10

A.4 Linearly Additive-Style Error Correction

By using the rounded floor function instead of the regular floor function, a more robust error recovery process is derived, which is required to be applied to all of the CVNS digits. The developed error correction is a sequential process, which starts from the Least Informed Digit up to the Most Informed Digit. The LID remains uncorrected, since there is no reference digit

for its correction. To increase the efficiency of the process, corrected lower informed digits are used for the recovery, which means that every digit requires to wait for the correction of all of it's less informed digits.

Linearly additive-Style Error Correction Steps:

The error correction method is a non-iterative approach developed for linearly additive-style errors and is summarized as follows:

- For the linearly additive-style error correction condition $\left|\overline{\varepsilon}_i - \frac{\overline{\varepsilon}_{i-1}}{\beta}\right| < \frac{1}{2}$ should be satisfied for any two adjacent digits to have a successful correction.
- Correction starts from $((\tilde{x}))_{-k+1}$, which is the digit next to the LID. It is assumed that $((x))_{-k} = ((\tilde{x}))_{-k} = ((\hat{x}))_{-k}$, which means that the value of the LID is considered un-errored.
- For each pair of digits, $((\tilde{x}))_i$ and $((\tilde{x}))_{i-1}$, find integer $\lfloor((\hat{x}))_i\rfloor$ from (A.13) which is repeated here again. It should be noted that the corrected version of the less informed digit is used.

$$\lfloor((\hat{x}))_i\rfloor_R = \left[((\tilde{x}))_i - \frac{((\hat{x}))_{i-1}}{\beta}\right]_R \qquad i > -k$$

- After finding the integer, the current digit value can be corrected using the recovered version of the less informed digit as follows:

$$((\hat{x}))_i = \lfloor((\hat{x}))_i\rfloor_R + \frac{((\hat{x}))_{i-1}}{\beta} \qquad i > -k \tag{A.15}$$

- Correction proceeds to the next higher index digits up to the MID.

Example: Given the value $x = 98.792$ within the range $M = 1000$ the decimal CVNS ($\beta = 10$) requires $n = 2$, and $l = -2$ is chosen. The first two rows in Table A.1 show CVNS digits and their associated integers. The next two rows contain altered CVNS digits with their associated integers. The last two rows are the results of error recovery using both floor function, and rounded floor function. Although floor function is mathematically simpler than the rounded floor function, it is not able to retrieve the integer values correctly and

results are not acceptable. The last row shows the results of the correction based on the algorithm given for linearly additive-style error correction. The only error in the digits is the result of un-corrected LID error, which has propagated into the more informed digits.

i	2	1	0	−1	−2
$(\!(x)\!)_i$	0.97792	9.7792	7.792	7.92	9.2
$\lfloor(\!(x)\!)_i\rfloor$	0	9	7	7	9
$(\!(\tilde{x})\!)_i$	1.12792	9.9292	7.942	8.07	9.35
$\lfloor(\!(\tilde{x})\!)_i\rfloor$	1	9	7	8	9
$(\!(\hat{x})\!)_i$ by Floor Function	1.978935	9.78935	7.8935	8.935	9.35
$(\!(\hat{x})\!)_i$ by Rounded Floor Function	0.977935	9.77935	7.7935	7.935	9.35

Table A.1: Error correction example for $x = 97.792$ with $\beta = 10$ and Maximum Range of $M = 1000$

A.5 Linearly Additive-Style Correction Error

After applying the sequential error correction, the remaining error is due to possible errors in the LID that during the correction propagates upward. Digit errors after correction are equal to:

$$\varepsilon_i' = \begin{cases} \bar{\varepsilon}_{-k} & i = -k \\ \frac{\bar{\varepsilon}_{-l}}{\beta^{i+l}} & i \geq -k+1 \end{cases} \tag{A.16}$$

and the error PDF of the digits is:

$$p(\varepsilon_i') = \beta^{i+l}.p(\bar{\varepsilon}_{-k}) \tag{A.17}$$

The MID error coming from the LID, after correction is:

$$\varepsilon'_n = \frac{\overline{\varepsilon}_{-k}}{\beta^{L+k}} \tag{A.18}$$

This error causes a small difference between the root and recovered value. The difference between the two, $\varepsilon = \hat{x} - x$, is now proportional to the MID error ($\varepsilon = \frac{M}{\beta}\varepsilon'_n$). Since $L + k$ indicates the total number of CVNS digits, this error can be reduced by increasing the number of digits. In other words, precision of the final results is controlled by the number of digits, the same as the Positional Number System. The relative error is found to be equal to:

$$\check{\varepsilon} = \frac{\varepsilon}{M} = \frac{\hat{x} - x}{M} = \frac{\overline{\varepsilon}_{-k}}{\beta^{L+k+1}} \tag{A.19}$$

The PDF of the relative error $p(\check{\varepsilon})$, is a narrowed version of the PDF of the LID error, and improvements are exponential with increasing the number of digits.

$$p(\check{\varepsilon}) = p(\overline{\varepsilon}_{-k}.\beta^{L+k+1}).\beta^{L+k+1} \tag{A.20}$$

The standard deviation of the $p(\check{\varepsilon})$ is:

$$\sigma_{\check{\varepsilon}} = \frac{\sigma_{(\overline{\varepsilon}_{-k})}}{\beta^{L+k+1}} \tag{A.21}$$

A.6 Error Threshold

It is required to find the largest value of the tolerable error as the threshold of error for a pair of digits such that all of the error combinations can be removed successfully. From the statistical point of view, and to obtain the maximum value of the DP of error PDF has to be obtained such that (A.14) is satisfied for all error values less than that. The quality of the implementation medium should be able to accommodate the required error threshold. For a radix-β the error threshold is always higher than the error Descriptive Point.

Fig. A.2 shows the map of error digits. The slanted lines indicate the digit errors, with parameter β. The top line indicates $\bar{\varepsilon}_{i-1} = \left(\bar{\varepsilon}_i + \frac{1}{2}\right)\beta$, and the bottom line indicates $\bar{\varepsilon}_{i-1} = \left(\bar{\varepsilon}_i - \frac{1}{2}\right)\beta$. The dashed area indicates the points were digit errors satisfy (A.14). It is assumed that ε_{th_i} is the error threshold for digit $(\!(x)\!)_i$, and $\varepsilon_{th_{i-1}}$ is the error threshold of digit $(\!(x)\!)_{i-1}$. The error bound of two adjacent digits is $|\bar{\varepsilon}_i| < \varepsilon_{th_i}$ and $|\bar{\varepsilon}_{i-1}| < \varepsilon_{th_{i-1}}$, and points on the map that satisfy these conditions lie within a rectangle defined as Θ, which is centered around the origin and is the largest rectangle that fits within the dashed area. The larger the rectangle, the higher the error threshold for a pair of digits.

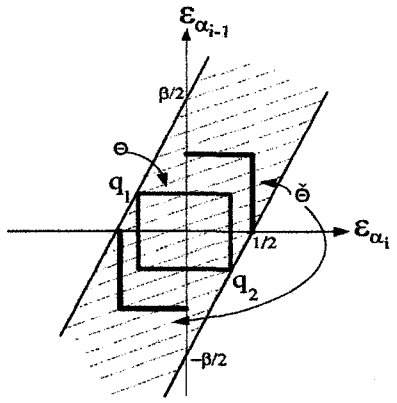


Figure A.2: Error Map of two adjacent CVNS digits

This rectangle touches the upper line at point $q_1 : \left(\varepsilon_{th_i}, (\frac{1}{2} - \varepsilon_{th_i})\beta\right)$, and the lower line at $q_2 : \left(-\varepsilon_{th_i}, (-\frac{1}{2} + \varepsilon_{th_i})\beta\right)$. Therefore, within this area, threshold pair ε_{th_i} and $\varepsilon_{th_{i-1}}$ are related as follows:

$$\varepsilon_{th_i} = \frac{1}{2} - \frac{\varepsilon_{th_{i-1}}}{\beta} \tag{A.22}$$

Before correction, all of the CVNS digits have the same chance of error, and the error PDF of all of the CVNS digits is essentially the same ($p(\bar{\varepsilon}_i) = p(\bar{\varepsilon}_{i-1})$). Since this assumption holds true for DP of two adjacent CVNS digits, the same threshold for both digits is expected $\left(\varepsilon_{th_i} = \varepsilon_{th_{i+1}} = \varepsilon_{th}\right)$. Therefore, error threshold of the CVNS digits obtained from (A.22) is:

$$\varepsilon_{th} = \frac{\beta}{2(\beta+1)} \tag{A.23}$$

The area associated with the relative error threshold ($\check{\varepsilon}_{th} = \varepsilon_{th}/\beta$) is shown in Fig. A.2 by rectangle $\check{\Theta}$, and defined as:

$$\check{\varepsilon}_{th} = \frac{\varepsilon_{th}}{\beta} = \frac{1}{2(\beta+1)} \tag{A.24}$$

The LID error affects the overall accuracy of the CVNS representation, after applying error correction. If errors were bounded by the error threshold associated with the radix of choice, the error in the the LID is bounded well ($|\overline{\varepsilon}_{-k}| \leq \varepsilon_{th}$). As expected, this condition reduces the relative approximation error of the CVNS representation given in (A.19) to:

$$\check{\varepsilon} \leq \frac{\check{\varepsilon}_{th}}{\beta^{L+k}} = \frac{1}{2(\beta+1)\beta^{L+k}} \tag{A.25}$$

Equations (A.19) and (A.25) show that the CVNS error representation reduces exponentially with increasing the number of digits. An important conclusion can be drawn here; although redundancy in the CVNS is similar to the basic form of adding redundancy in positional number systems which is used to increase the accuracy (repetition of binary digits), it improves the result of representation more and is more robust to errors. In the case of positional numbers and repetition of data, the value is simply extracted by averaging the multiple replicas, in which the approximation error is linearly proportional to the number of replicas, hence it is linearly proportional to the number of digits.

At this point the error threshold of the CVNS and MVL systems can be compared. In a typical MVL system, the radix is chosen such that the error threshold allows for an acceptable low error probability, that can be considered practically zero. This threshold is often termed noise margin, since it relates to the physical distance between two subsequent digit integer values. In this condition, by increasing the radix of the system, the threshold of the system decreases. The relative error threshold of the MVL system in this case is:

$$\check{\epsilon}_{th} = \frac{1}{2(B-1)} \tag{A.26}$$

The maximum radix which can be chosen in either CVNS or MVL systems, should be chosen considering the error threshold. The upper bound of the allowable radix for both case can be obtained from (A.24) and (A.26) as follows:

$$\beta_{max} = \frac{1}{2\,\check{\varepsilon}_{th}} - 1 \quad CVNS \tag{A.27}$$

$$B_{max} = \frac{1}{2\,\check{\epsilon}_{th}} + 1 \quad MVL \tag{A.28}$$

For the same value of relative error threshold in both environments, $\check{\varepsilon}_{th} = \check{\epsilon}_{th}$, CVNS can take advantage of lower radix values compared to the MVL, which means less implementation complexity.

A.7 Probability of Success of the Proposed Linearly Additive-Style Error Correction

In a typical environment $p(\overline{\varepsilon}_i)$ with DP $\varepsilon_0 \ll 1$, practically all error values are within the threshold limit, and hence $P\big(\lfloor((\hat{x}))_i\rfloor_R = \lfloor((x))_i\rfloor\big) \approx 1$, which can be computed precisely for a given environment. A successful error recovery requires to a close approximation of the associate integers $\big(\lfloor((\hat{x}))_i\rfloor_R - \lfloor((x))_i\rfloor = \varepsilon_i\big)$. The ideal case is when there is no error ($\varepsilon_i = 0$). The total probability of occurrence of ε_i is:

$$P\big(\lfloor((\hat{x}))_i\rfloor_R - \lfloor((x))_i\rfloor = \varepsilon_i\big) = \int_{\overline{\varepsilon}_i=-\infty}^{\overline{\varepsilon}_i=\infty} p(\overline{\varepsilon}_i) \cdot \int_{\beta(\overline{\varepsilon}_i-\frac{1}{2}-\varepsilon_i)}^{\beta(\overline{\varepsilon}_i+\frac{1}{2}-\varepsilon_i)} p(\overline{\varepsilon}_{i-1})d\overline{\varepsilon}_{i-1} \tag{A.29}$$

This equation does not take into account that the error values may be limited to a maximum or minimum value, which is imposed from the implementation bounds. Therefore, (A.29) provides a lower bound for the probability. The general properties of the PDF

are, that the total probability is equal to 1, and that a positive variance exists such that $\int_{-\infty}^{\infty} \epsilon^2.p(\epsilon)d\epsilon = \sigma^2$. Moreover, the sample PDF should be twice integrable in order to find an analytical solution. For this reason, we consider a negative absolute exponential Laplace PDF as shown:

$$p(\varepsilon) = \frac{e^{-\sqrt{2}|\epsilon|/\sigma}}{\sqrt{2}\sigma} \tag{A.30}$$

Analytical solutions of (A.29) do not exist for other forms of the error of type e^{-x^2}, bell shaped curves of type $cos(2atan(e)+1)$, the Cauchy distribution $\frac{1}{1+e^2}$, and $1-tanh^2(e)$. The reason is that the variable $\bar{\varepsilon}_i$ of the outer integral occurs in the limits of the inner integral. By substituting the Laplace PDF of (A.30) for both $p(\bar{\varepsilon}_i)$ and $p(\bar{\varepsilon}_{i-1})$ in (A.29), for the particular case of interest $\varepsilon_i = 0$, using the analytical software tools we obtain:

$$P\big(\lfloor((\hat{x}))_i\rfloor_R = \lfloor((x))_i\rfloor\big) = 1 - \frac{\beta^2 e^{\frac{-1}{\sqrt{2}\sigma}} - e^{\frac{\beta}{\sqrt{2}\sigma}}}{\beta^2 - 1} \tag{A.31}$$

Considering a similar case for the MVL, the probability of successful digit retrieval by rounding is:

$$P(\hat{x}_i = x_i) = \begin{cases} \int_{-1/2}^{1/2} p(\epsilon)\, d\epsilon & 0 < x_i < B-1 \\ \int_{-1/2}^{\infty} p(\epsilon)\, d\epsilon & x_i = B-1 \\ \int_{-\infty}^{1/2} p(\epsilon)\, d\epsilon & x_i = 0 \end{cases} \tag{A.32}$$

By replacing the same PDF given by (A.30) in the above equation, the probability of success for MVL systems is:

$$P(\hat{x}_i = x_i) = \begin{cases} 1 - e^{\frac{-1}{\sqrt{2}\sigma}} & 0 < x_i < B-1 \\ 1 - \frac{1}{2}e^{\frac{-1}{\sqrt{2}\sigma}} & x_i = 0,\ B-1 \end{cases} \tag{A.33}$$

The probability of success in the MVL system is independent of the radix, however for the CVNS it directly affects the CVNS dimension, namely the radix value. The equality $P\left(\lfloor((\hat{x}))_i\rfloor_R = \lfloor((x))_i\rfloor\right) = 1$ is true for error that is at most bounded by the error thresholds of each environment. The importance of these equations is that in any environment with a required confidence which is represented by function P, the dimensions of the CVNS namely the radix can be set to meet these requirements.

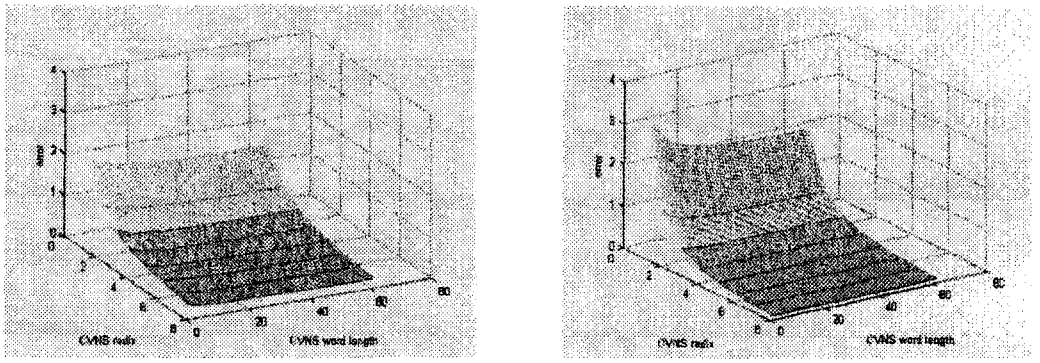
A.8 Selecting the CVNS Dimensions

The dimensions of the CVNS are determined by the number of digits and the radix value. These parameters determine the eventual accuracy of the representation, depending on the implementation medium. In general, the implementation parameters are fixed and it is required to specify the eventual representation from a system perspective. Dimensions of the CVNS are decided as follows:

- Quality of the medium determines the highest selectable radix. If error values were bounded to the error threshold value, radix β is the free variable and can be set to increase the threshold. In this case, the CVNS radix does not require to be high. Other factors such as system complexity as a function of the CVNS radix may restrict the selection range.

- Demand on the accuracy of the CVNS approximation sets the number of digits. The previously chosen radix of the CVNS is now used to decide on the required number of digits.

The accuracy of presenting x in the CVNS is compared with the quantization error of presenting it in MVL. In comparison, the relative quantization error of a real value depends on the represented value, and not on the implementation medium. We assume that the value x is uniformly distributed within a boundary $\pm M$, therefore, the relative error is also distributed evenly, and is bounded by:



(a) Bounded Digit Error (b) Unbounded Digit Error

Figure A.3: Error functions of two cases of bounded and unbounded error in the CVNS with respect to the quantization error in positional number system

$$|\check{\epsilon}| \leq \frac{1}{2(B^{m+t+1}-1)} \tag{A.34}$$

The standard deviation of the quantized error with uniform distribution is:

$$\sigma_{\check{\epsilon}} = \frac{\sqrt{3}}{6(B^{m+t+1}-1)} \tag{A.35}$$

The CVNS error in both cases of bounded and unbounded LID error is compared to the quantization error of the positional number system as follow:

$$\frac{\check{\varepsilon}}{\check{\epsilon}} = \begin{cases} \frac{2(B^{m+t+1}-1)}{\beta^{L+k}}.\bar{\varepsilon}_{-k} & \textit{Bounded error} \\ \\ \frac{(B^{m+t+1}-1)}{(\beta+1)\beta^{L+k}} & \textit{Unbounded error} \end{cases} \tag{A.36}$$

These functions are shown in the Fig. A.3 as a function of the CVNS radix, and the number of the CVNS digits. Both show the same trend, where by increasing either one of the parameters, CVNS implementation error becomes less than the quantization error.

A.9 Imprecise Correction

In the previous section, it was assumed that the correction is performed with infinite accuracy. This assumption helped to understand the theoretical bounds of the probability of success for the CVNS digit recovery. In general, the CVNS accuracy can be increased by increasing the number of digits, as long as the digit error remains within the limit. It was assumed, in studying the confidence of the error corrections, that the error PDF is practically zero for the error values higher than the threshold. In this section, we discuss the limitations of the overall accuracy imposed by an environment with imprecise correction.

Correction is essentially a two step algorithm; the first step uses an approximation of the associated integer, and second step corrects the digit. In this process, division, subtraction and rounding operations are performed, where each may introduce more errors. Moreover, the result of the rounding operation may not be an exact integer value. These errors, called correction error (ε_{IC_i}), now exist in the corrected CVNS digit. Therefore, the error threshold requirement becomes more restrict and (A.14) is changed to:

$$\left|\overline{\varepsilon}_i - \frac{\overline{\varepsilon}_{i-1}}{\beta} + \varepsilon_{IC_i}\right| < \frac{1}{2} \tag{A.37}$$

The addition of the error term, ε_{IC_i}, tightens the threshold requirements, but Reverse Evolution process should be able to restore the digits correctly. The total errors from an imperfect addition, subtraction, and rounding during the correction process result in a corrected digit as:

$$((\hat{x}))_i = \lfloor((\hat{x}))_i\rfloor_R + \frac{((\hat{x}))_{i-1}}{\beta} + \varepsilon'_i \tag{A.38}$$

The error in the digit is:

$$\varepsilon'_i = \varepsilon_{IC_i} + \frac{\overline{\varepsilon}_{i-1}}{\beta} \tag{A.39}$$

The relative error, $\check{\varepsilon}$ has to be estimated, in order to provide the requirements for relative error threshold of the physical medium. During error correction, the correction error

accumulates in the eventual outcome, and the relative error is as follows:

$$\check{\varepsilon} = \frac{\bar{\varepsilon}_{-k}}{\beta^{L+k}} + \sum_{i=-l+1}^{L} \frac{\varepsilon_{IC_i}}{\beta^{L-i}} \tag{A.40}$$

The standard deviation which was given in (A.21) for perfect error correction, under the assumptions:

- The correction errors have Gaussian PDFs,
- The standard deviation of all the relative correction errors are equal,
- The tightened error threshold applies to all of the digits errors,

is now changed to:

$$\sigma_{\check{\varepsilon}} = \frac{\sigma_{(\bar{\varepsilon}_{-k}/\beta)}}{\beta^{L+k}} + \sigma_{\varepsilon_{RE_i}} \sum_{i=0}^{L+k-1} \beta^{-i} \tag{A.41}$$

For the correction error to become less significant compared to the LID error, it is important to have

$$\sigma_{\varepsilon_{RE_i}} \ll \frac{\sigma_{(\bar{\varepsilon}_{-k}/\beta)}}{\beta^{L+k}} \tag{A.42}$$

This means that the environment in which the correction is performed must have higher precision than the environment from which digits are generated. This confirms to the general notions of the requirements for a mixed-signal environment, where the analog domain should have a higher quality compared to the digital environment.

Basic theory of the error correction in the CVNS along with equations were derived in this section. In the next section, modularly additive-style errors are considered as well. A novel method is proposed, which removes all the line- as well as modularly additive- and congruence- style errors.

A.10 Modularly Additive-Style Error

CVNS is a modular number system and hence digit values are within a bounded ring $[0, \beta)$. When an error occurs in a CVNS digit which is close to the boundaries of the ring, that digit would wrap around the ring. Even if this event happens for only one of the CVNS digits, the linearly additive-style error correction can not correct it and will result in more errors during the correction.

In the previous sections, linearly additive-style errors were considered, which are typical for practical linearity additive-style quantities of hardware implementations of the CVNS. The general form of an errored digit was given in (A.4), and it is required to remove the modularly additive-style error from the CVNS digit as well. Many of the properties of the linearly additive-style errors can be easily expanded for the modularly additive-style errors.

A CVNS digit which contains three forms of errors is repeated here:

$$(\!(\tilde{x})\!)_i = ((\!(x)\!)_i + \varepsilon_i^{\circ}) \, mod\beta + \overline{\varepsilon}_i + I_{\varepsilon_i}\beta$$

The error in the digit is equal to:

$$\varepsilon_i = ((\!(x)\!)_i + \varepsilon_i^{\circ}) \, mod\beta + \overline{\varepsilon}_i + I_{\varepsilon_i}\beta - (\!(x)\!)_i \tag{A.43}$$

Error can be decomposed into a congruent term, $I_{\varepsilon_i}\beta$ and a remaining term, $\underline{\varepsilon}_i$. If linearly additive-style error correction presented in the previous section is applied to a CVNS digit with an error the same as the above, the term $\underline{\varepsilon}_i$ can be removed from the digit, but the congruent term stays and it is required to remove all forms of error from the digit. A robust modularly additive-style error correction method, therefore is able to remove any type of error that may be caused by the noise, nonlinearity or non perfect modulus operation. A Linearly additive-style error value $\underline{\epsilon}_i < \frac{\beta}{2}$ is always correctible. Two adjacent errored digits are:

$$(\!(\tilde{x})\!)_i = (\!(x)\!)_i + \underline{\varepsilon}_i + \underline{I}_{\varepsilon_i}\beta \tag{A.44}$$

$$(\!(\tilde{x})\!)_{i-1} = (\!(x)\!)_{i-1} + \underline{\varepsilon}_{i-1} + \underline{I}_{\epsilon_{i-1}}\beta \tag{A.45}$$

The CVNS digits lie on a ring which is a bounded set of real values $[0, \beta)$, and an error may cause the digits to warp around this ring, and generate completely wrong results. The implementation of this number system with quantities such as current, voltage or charge is also bounded, can be considered as a ring. To retrieve a close and accurate integer value of the associated CVNS digit a new modulus rounding function is used for restoring the digits as:

$$\lfloor (\!(\hat{x})\!)_i \rfloor_\beta^\pm = \begin{cases} \left((\!(\tilde{x})\!)_i - \frac{(\!(\tilde{x})\!)_{i-1}}{\beta} \right) mod\beta^+ & x \geq 0 \\ \\ \left((\!(\tilde{x})\!)_i - \frac{(\!(\tilde{x})\!)_{i-1}}{\beta} \right) mod\beta^- & x < 0 \end{cases} \tag{A.46}$$

In cases where the CVNS digit is close to the ring boundaries, the new function is more efficient. Notation $[.]_\beta^\pm$ is used to combine the top and bottom terms as:

$$\lfloor (\!(\hat{x})\!)_i \rfloor_\beta^\pm = \left[(\!(\tilde{x})\!)_i - \frac{(\!(\tilde{x})\!)_{i-1}}{\beta} \right]_\beta^\pm \tag{A.47}$$

Substituting the errored digit, results in:

$$\lfloor (\!(\hat{x})\!)_i \rfloor_\beta^\pm = \left[(\!(x)\!)_i + \underline{\varepsilon}_i + \underline{I}_{\epsilon_i}\beta - \frac{(\!(x)\!)_{i-1} + \underline{\varepsilon}_{i-1} + \underline{I}_{\epsilon_{i-1}}\beta}{\beta} \right]_\beta^\pm \tag{A.48}$$

The rounding operation has the property $[x + I]_\beta = [x]_\beta + I$ for any real value of x and integer I. An expression for the associated integer is divided into an integer part, and a group of integer and non-integer error terms as follows:

$$\lfloor (\!(\hat{x})\!)_i \rfloor_\beta^\pm = \lfloor (\!(x)\!)_i \rfloor + \left[\underline{\epsilon}_i - \frac{\underline{\epsilon}_{i-1}}{\beta} \right]_\beta^\pm + \underline{I}_{\epsilon_i}\beta + \underline{I}_{\epsilon_{i-1}} \tag{A.49}$$

Any two CVNS digits positioned on the real ring are related to each other as follows:

$$((x))_i = \begin{cases} \left(\lfloor ((x))_i \rfloor_\beta^+ + \frac{((x))_{i-1}}{\beta} \right) mod^+ \beta & ((x))_i \geq 0 \\ \\ \left(\lfloor ((x))_i \rfloor_\beta^- + \frac{((x))_{i-1}}{\beta} \right) mod^- \beta & ((x))_i < 0 \end{cases} \tag{A.50}$$

The restored CVNS digit which contains error using the above equation is equal to:

$$((\hat{x}))_i = \begin{cases} \left(\lfloor ((x))_i \rfloor_\beta^+ + \frac{((x))_{i-1}}{\beta} + \frac{\varepsilon_{i-1}}{\beta} + \left[\underline{\varepsilon}_i - \frac{\varepsilon_{i-1}}{\beta} \right]_\beta^\pm + \underline{I}_{\varepsilon_i} \beta - \underline{I}_{\varepsilon_{i-1}} \right) mod^+ \beta & ((x))_i \geq 0 \\ \\ \left(\lfloor ((x))_i \rfloor_\beta^- + \frac{((x))_{i-1}}{\beta} + \frac{\varepsilon_{i-1}}{\beta} + \left[\underline{\varepsilon}_i - \frac{\varepsilon_{i-1}}{\beta} \right]_\beta + \underline{I}_{\varepsilon_i} \beta - \underline{I}_{\varepsilon_{i-1}} \right) mod^- \beta & ((x))_i < 0 \end{cases} \tag{A.51}$$

This step eliminates congruent error terms $\underline{I}_{\varepsilon_{\alpha_{i-1}}}$, and $\underline{I}_{\varepsilon_{i-1}}$, due to the positive and negative modulus operation. By substituting, $((x))_i = \lfloor ((x))_i \rfloor_\beta^\pm + \frac{((x))_{i-1}}{\beta}$, and with the notation $mod^\pm$ to combine the top and bottom terms, the recovered digit is:

$$((\hat{x}))_i = \left(((x))_i + \frac{\varepsilon_{i-1}}{\beta} + \left[\underline{\varepsilon}_i - \frac{\varepsilon_{i-1}}{\beta} \right]_\beta^\pm \right) mod^\pm \beta \tag{A.52}$$

we can formulate a new boundary for successful modularly additive-style error correction.

$$\left| (\underline{\varepsilon}_i) mod \beta - \frac{(\underline{\varepsilon}_{i-1}) mod \beta}{\beta} \right| < \frac{1}{2} \tag{A.53}$$

This term is very similar to linearly additive-style error condition given in the previous section. The only remaining error is a fraction $\frac{1}{\beta}$ of the error $\underline{\varepsilon}_{i-1}$ of the less informed digit, which has propagated to the current digit.

$$((\hat{x}))_i = \left(((x))_i + \frac{\varepsilon_{i-1}}{\beta} \right) mod^\pm \beta \tag{A.54}$$

The error correction starts from the LID, and the error that propagates into the digit under correction is in fact a fraction $\frac{1}{\beta}$ of the LID error, and possibly an integer multiple of

the radix. Multiple integer of the radix error does not propagate into the next more informed digit due to the modulus operator. Error should be sufficiently small to warrant that the operand of the modulus term in (A.54) is less than β.

The ring implementation of the CVNS is the only true representation of the analog-digits used in this number system. Line implementation is an approximation, which was used for simplifying the mathematical equations. Modularly additive errors are characteristics of implementation the CVNS on the real ring. Moreover, it should be noted that while linearly additive-style error correction method is limited to errors that maintain $((\hat{x}))_i = ((x))_i$, the modularly additive error correction allows larger range. On the ring for two values to be equal, they should reside on the vicinity of each other. Digit errors in the amount of $0 \pm I.\beta$ are considered without any error.

A.11 Modularly additive-Style Error Correction

Similar to linearly additive-style error removal, steps for the modularly additive-style error removal are summarized. In comparison, modularly additive errors require one more step, which is decomposition to congruent error term, and the remaining linearly additive errors.

Iterative Modularly additive-Style Error Correction Steps:

- Decompose the error, which provides an insight into the correctibility of the CVNS digit.
- If condition stated in(A.53) is satisfied, digits can be recovered correctly.
- The LID remains the same, and is assumed to be unerroed. Correction starts from the more informed digit of the LID.
- For each pair of the CVNS digits, the integer part is found from (A.46) which is repeated here. It should be noted that during the correction, the recovered version of the less informed digits is used.

$$\lfloor((\hat{x}))_i\rfloor_\beta^{\pm} = \begin{cases} \left((\tilde{x}))_i - \frac{((\hat{x}))_{i-1}}{\beta}\right) mod\beta^{+} & x \geq 0 \\ \\ \left((\tilde{x}))_i - \frac{((\hat{x}))_{i-1}}{\beta}\right) mod\beta^{-} & x < 0 \end{cases}$$

- By using this integer, the digit is corrected through (A.50) which is repeated here

$$((x))_i = \begin{cases} \left(\lfloor((\hat{x}))_i\rfloor_\beta^{+} + \frac{((\hat{x}))_{i-1}}{\beta}\right) mod^{+}\beta & ((x))_i \geq 0 \\ \\ \left(\lfloor((\hat{x}))_i\rfloor_\beta^{-} + \frac{((\hat{x}))_{i-1}}{\beta}\right) mod^{-}\beta & ((x))_i < 0 \end{cases}$$

- This process is repeated for all of the CVNS digits up to the MID.

If a congruent error is in the CVNS digit, a wrap around the ring happens, and in the CVNS arithmetic this implies an overflow. The above scheme warrants that the corrected digits stay canonic, and are placed on the ring $[0, \beta)$. If the ring is implemented by a linearly additive-style implementation medium, and if all of the operations are performed by the linearly additive-style implementation medium and with infinite precision, then the digits end up to be canonic again.

Example: Given the value $x = 98.792$ within the range $M = 1000$ the decimal CVNS, $\beta = 10$ requires $L = 2$, and $-k = -2$ is chosen. Table A.2 shows the CVNS digits without any error, and also errored CVNS digits. Although floor function is mathematically simpler than the rounded floor function, it is not able to retrieve the integer values correctly and results are not correct. The last row shows the result of the correction based on the algorithm given for linearly additive-style error correction. The only error in the digits is the result of un-corrected LID error, which has propagated into the more informed digits.

This example shows that the rounded floor function is not able to recover errors in a modular environment, since it is designed and optimized for linearly additive-style errors.

The relative error is:

i	2	1	0	−1	−2
$((x))_i$	0.98792	9.8792	8.792	7.92	9.2
$((\tilde{x}))_i mod 10$	1.13792	0.0292	8.792	8.07	9.35
$\lfloor((\hat{x}))_i\rfloor_R$	1	−1	8	7	9
$\lfloor((\hat{x}))_i\rfloor_\beta$	0	9	8	7	9

Table A.2: Error correction example for $x = 98.792$ with $\beta = 10$ and Maximum Range of $M = 1000$

$$\breve{\varepsilon} = \frac{x - \hat{x}}{M} = \frac{\underline{\varepsilon}_{-k}}{\beta^{L+k}} \tag{A.55}$$

This equation can be written as the previous error threshold developed for the linearly additive-style errors and can be applied to the errors on the ring as follows:

$$|(\underline{\varepsilon}_i) mod^{\pm} \beta| = \|\underline{\varepsilon}_i\| < \varepsilon_{th} \tag{A.56}$$

The condition $\|\underline{\varepsilon}_i\| < \varepsilon_{th}$ guarantees that if $|\underline{\varepsilon}_i| < \varepsilon_{th}$, errors of $\underline{\varepsilon}_i \pm I.\beta$ are tolerated. The important consequence is that new correction method is more robust, and can accommodate congruent digit values as well. The condition given in the linearly additive-style error correction, is more strict and can not remove congruent error terms. When the CVNS is implemented in an environment which is not modular by nature, artificial congruence should have been applied. This method adds or subtracts integer multiples of the radix to or from the CVNS digit which might have been resulted from an addition or other types of arithmetic operations. This additional step, warrants that the CVNS digits are canonic and stay within the ring range. It should also be noted that the results developed for linearly additive-style error in terms of error statistics and chances of success are applicable to the modularly additive-style errors as well.

These correction schemes lay the foundations for digital architectures that internally

employ CVNS analog processing. In this chapter, we showed that despite the uncertainty in the CVNS digits due to analog nature of the implementation medium, the CVNS can be restored. A set of digits which do not follow the general relation between the two adjacent digits, indicate error. The error correction returns the digits into their normal canonic form, and corrects the relation. These schemes can detect and correct multiple errors, if errors were bounded by the conditions which are defined in this dissertation. Within the CVNS context, digits are considered correctible if their associated digits were found with the aid of the less informed digits. A digit value is statistically enhanced if the absolute value of the less informed digit error is less than β times the absolute value of the error of the digit under correction.

A.12 Conclusion

The CVNS digits do not have noise margin, unlike conventional number systems such as Binary or Multiple-Valued-Logic. Protection of the CVNS digits against noise and other implementation factors is warranted by the redundancy among the digits. The redundancy is systematic among the digits, and allows for digit errors during a non-iterative process. Error in the digits is detected when digits do not conform to their normal form. Error correction narrows the probability density of error in CVNS digits.

The CVNS error threshold is a digit error threshold boundary with associated statistical confidence which is a function of the digit error PDF. If all digit errors occur within the error threshold, then it is possible to restore the digits, regardless of which specific error combination might have occurred. Therefore, CVNS digits are recovered or at least enhanced from an error statistics point of view. The digit correction has to be performed in an environment which has comparable accuracy to the digital to analog conversion.

CVNS targets situations where the implementation cost function dictates low radices, despite the theoretical feasibly of high radices. Compared to Multiple-Valued-Logic number systems, CVNS does not require increasing the radix to increase the integrity of the CVNS

digits. For a given choice of radix, any improvement in the error statistics immediately results in an improvement of the error variance compared to the quantized and Multiple-Valued-Logic implementations.

VITA AUCTORIS

Mitra Mirhassani obtained her Master's degree from the University of Windsor, Windsor, Ontario in 2003. During her master's work she was involved with the design and implementation of mixed signal Neural Networks. For her research, she won the National MICRONET Best Graduate Student Paper Award on paper entitled: "VLSI Implementation of Feed-Forward neural Network with Learning On-Chip Using $0.18\mu m$ CMOS", System Division at 2003.

She started her Ph.D. research on the area of computer arithmetic, neural network implementation and Continuous Valued Number System. Mitra has won many scholarships such as Ontario Graduate Scholarship (OGS), Ontario Graduate Scholarship in Science and Technology (OGSST) and NSERC Postgraduate scholarship (PGS D).