UNIVERSITY OF ADELAIDE LIBRARY 18/10/07

Threshold Logic Based Implementation of High Performance VLSI Arithmetic Circuits

by

Peter Celinski

B.E. (Electrical & Electronic Engineering, with Honors),
The University of Adelaide, Australia, 1998

Thesis submitted for the degree of

Doctor of Philosophy

in

Electrical and Electronic Engineering
University of Adelaide

December, 2006

© 2006
Peter Celinski
All Rights Reserved

Contents

Abstract

This Thesis focuses on the area of high speed very large scale integration (VLSI) complementary metal oxide semiconductor (CMOS) circuit design using threshold logic (TL) techniques. The work described in this document contributes three major advances on high speed TL based CMOS circuit design: (i) the development and experimental verification of novel high speed TL gate circuit topologies; (ii) a method for delay modelling of sense-amplifier based TL gates and (iii) novel TL based networks for the implementation of high speed arithmetic circuits. In this Thesis, the basics and previous work in threshold logic are reviewed, including theoretical results for TL based networks used in arithmetic and TL gate circuit design techniques. Novel floating gate based TL circuit implementations based on precharged sense-amplifiers employing charge recycyling are described and experimentally verified. A new weight-shared circuit technique is proposed which significantly reduces the area cost. Based on the theory of Logical Effort, a model for the Charge Recycling Threshold Logic (CRTL) gate is developed and experimentally verified. This model is used to evaluate and compare a number of CRTL based circuits, demonstrating its significant reduced delay compared to conventional static and dynamic CMOS logic. New parallel counters are proposed and partial product reduction trees based on these counters for use in parallel multipliers are shown to be significantly faster than previously published schemes. The 64-bit prefix-8 adder presented here is the fastest 64-bit adder published to date. The contributions in this Thesis are an important step towards alleviating the issues faced in present day VLSI arithmetic design and demonstrate for the first time the significant benefits offered by TL compared to conventional logic circuit techniques. The methodologies introduced are shown to lead to increased circuit compactness and reduced power dissipation which are of particular interest for future smart sensor technology and will potentially impact on future portable electronics systems for a range of applications from mobile personal communications through to aerospace systems.

Statement of Originality

Declaration

NAME: PETER CELINSKI PROGRAM: PhD

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

SIGNATURE: ..... DATE: 27/09/2007

Acknowledgments

This work was carried out with the support of many people to whom I am indebted. Firstly I want to thank my supervisor, Professor Derek Abbott (Director of the Centre for Biomedical Engineering, School of Electrical and Electronic Engineering; The University of Adelaide) for encouraging me to work on my own ideas from the beginning and providing the funding and support for numerous conferences and collaborative research trips.

Thanks go to my friends and colleagues from the University of Adelaide School of Electrical and Electronic Engineering: Sam Mickan, Leonard Hall, Greg Harmer, Andrew Allison, John Salerno, Greg Sherman, Said Al-Sarawi, Kiet To, Troy Townsend and Tony Sarros. In particular I would like to acknowledge the collaboration of Tony and Said on the SEED based TL work presented in Chapter 7 and Troy on the multiplier work presented in Chapter 6.

I would like to thank my many colleagues and friends abroad—Jose F. López at the University of Las Palmas in Spain, Dmitry Cheresiz, Sorin Cotofana and Stamatis Vassiliadis at the Delft University of Technology in the Netherlands, Sunay Shah at Oxford, Bart Rylander at the University of Portland, and Vojin Oklobdzija at the University of California, Davis.

A special thanks to Shaghik Atakaramians for assisting with the Latex presentation for this thesis.

I would like to gratefully acknowledge the financial support of the University of Adelaide Postgraduate Scholarship, Research Abroad Scholarship, Australian Research Council, Delft University of Technology Research Fellowship, IEEE SA Section Travel Scholarship, SPIE Student Travel Grant, Sir Ross and Sir Keith Smith Fund Scholarship, and the D. R. Stranks Travelling Fellowship.

Lastly, I would like to sincerely thank Theresa, my parents and Tom for their tremendous support, encouragement and generous patience.

This thesis survived a few parties, a startup, two rounds of VC funding and the birth of my beautiful son Aydan.

Conventions

This thesis it typeset using the LaTeX2e software. WinEdt build 5.4 was used as an effective interface to LaTeX. Harvard style is used for referencing and citation in this thesis. Australian English spelling is adopted, as defined by the Macquarie English Dictionary.

Publications

[1] **Peter Celinski**, Derek Abbott, and Jose F. López, "Novel Extension of neu-MOS techniques to neu-GaAs," *Proc. of SPIE MICRO/MEMS 99*, **vol 3893** Gold Coast, Australia, pp. 169-175, December 1999.

[2] **Peter Celinski**, Said Al-Sarawi, and Derek Abbott, "A Delay Model for neuron-MOS and Capacitive Threshold Logic," *Proc. 7th IEEE International Conference on Electronics, Circuits & Systems*, Lebanon, pp. 932-935, December 2000.

[3] **Peter Celinski**, Gregory D. Sherman, and Derek Abbott, "Implementation of Arbitrary Boolean Functions in Threshold Logic," *Proc. SPIE International Symposium on Smart Electronics and MEMS*, **vol. 4236**, Melbourne, Australia, December 2000.

[4] **Peter Celinski**, Jose F. López, and Derek Abbott, "Novel Extension of neu-MOS techniques to neu-GaAs," *Microelectronics Journal* (Elsevier), **vol 31**, no 7, pp. 577-582, 2000.

[5] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "Complementary neu-GaAs structure," *IEE Electronics Letters*, **vol 36**, no 5, pp. 424-425, March 2000.

[6] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "A Low Power, High Speed Threshold Logic and its Application to the Design of Novel Carry Lookahead Adders," *Proc. SPIE International Symposium on Smart Electronics and MEMS*, pp. 258-265, Adelaide, Australia, December 2001.

[7] **Peter Celinski**, Gregory D. Sherman, Jose F. López, and Derek Abbott, "A Mapping Technique For the Synthesis of Linear Threshold Networks to Implement Boolean Functions," *Advances in Neural Networks and Applications*, Nikos Mastorakis, editor, World Scientific Engineering Society, pp. 224-228 2001.

[8] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott,"Low Power, High Speed Charge Recycling Threshold Logic Gate," *IEE Electronics Letters*, **vol 37**, no 17, pp. 1067-1069, August 2001.

[9] **Peter Celinski**, Troy Townsend, Jose F. López, Said Al-Sarawi, and Derek Abbott, "A Compact Parallel Multiplication Scheme Based on (7,3) and (15,4) Self-Timed Threshold Logic Counters," *Proc. 2nd WSEAS International Conference on Instrumentation, Measurement, Control, Circuits and Systems*, pp. 2641-2645, Mexico, May 2002.

[10] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "A Compact (m,n) Parallel Counter Circuit Based on Self Timed Threshold Logic," *Proc. 2nd WSEAS International Conference on Instrumentation, Measurement, Control, Circuits and Systems*, pp. 2601-2605, Mexico, May 2002.

[11] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "A Family of Low Depth, Threshold Logic, Carry Lookahead Adders," *Proc. 2nd WSEAS International Conference on Instrumentation, Measurement, Control, Circuits and Systems*, pp. 1981-1983, Mexico, May 2002.

[12] **Peter Celinski**, Sorin D. Cotofana, and Derek Abbott, "Generalized, Compact (m,n) Counters for High Speed Multipliers," *Proc. SPIE International Symposium on Smart Materials, Nano-, and Micro-Smart Systems*, pp. 205-213, Melbourne, Australia, December 2002.

[13] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "Low Depth Carry Lookahead Addition Using Charge Recycling Threshold Logic," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 469-472, Phoenix, USA, May 2002.

[14] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "Compact Parallel (m,n) Counters Based on Self Timed Threshold Logic," *IEE Electronics Letters*, **vol 38**, no 13, pp. 633-635, June 2002.

[15] **Peter Celinski**, Jose F. López, Said Al-Sarawi, and Derek Abbott, "Low depth, low power carry lookahead adders using threshold logic," *Microelectronics Journal* (Elsevier), **vol. 33**, No. 12, pp. 1071-1077, 2002.

[16] **Peter Celinski**, Derek Abbott, and Said Al-Sarawi, "Level sensitive latch," US Patent number **6,542,016**, April 2003.

[17] **Peter Celinski**, Sorin D. Cotofana, Jose F. López, Said Al-Sarawi, and Derek Abbott, "State-of-the-Art in CMOS Threshold Logic VLSI Gate Implementations and Applications," *Proc. SPIE VLSI Circuits and Systems Conference*, **vol 5117**, pp.53-63 Spain, May 2003 (**invited**).

[18] **Peter Celinski**, Sorin D. Cotofana, and Derek Abbott, "A-DELTA: A 64-bit High Speed, Compact, Hybrid Dynamic-CMOS Threshold-Logic Adder," *Proc. 7th International Work Conference on Artificial and Natural Neural Networks*, (IWANN), pp. 73-80, Spain, 2003.

[19] **Peter Celinski**, Sorin D. Cotofana, and Derek Abbott, "Area Efficient, High Speed Parallel Counter Circuits Using Charge Recycling Threshold Logic," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 233-236 Bangkok, Thailand, May 2003.

[20] **Peter Celinski**, Sorin D. Cotofana, and Derek Abbott, "A Logical Effort Based Delay Model of Charge Recycling Threshold Logic Gates," *Proc. ProRISC Workshop on Circuits, Systems and Signal Processing*, pp. 43-48, Veldhoven, Netherlands, November 2003 (**keynote presentation**).

[21] Troy Townsend, **Peter Celinski**, Said Al-Sarawi, and Michael J. Liebelt, "Hybrid Parallel Counters - Domino and Threshold Logic," *Proc. IEEE Computer Society Annual Symposium on VLSI — Emerging Trends in VLSI Systems Design* (ISVLSI'04), pp. 275-276, Lafayette, Louisiana, February, 2004.

[22] **Peter Celinski**, Said Al-Sarawi, Derek Abbott, Sorin D. Cotofana, and Stamatis Vassiliadis, "Logical Effort Based Design Exploration of 64-bit Adders Using a Mixed Dynamic-CMOS/Threshold-Logic Approach," *Proc. IEEE Computer Society Annual Symposium on VLSI — Emerging Trends in VLSI Systems Design* (ISVLSI'04), pp. 127-132, Lafayette, Louisiana, February, 2004.

[23] **Peter Celinski**, Sorin D. Cotofana, and Derek Abbott, "Delay Evaluation of High Speed Data-path Circuits Based on Threshold Logic," *Proc. of 14th International Workshop on Power and Timing Modeling, Optimization and Simulation* (PATMOS); Santorini, Greece, pp. 899-906, September 2004.

[24] Said Al-Sarawi, **Peter Celinski**, and Tony Sarros, "Data Converters," Australian Patent Office, Application Number **2004906742**, November 2004.

[25] Tony Sarros, Said Al-Sarawi, **Peter Celinski**, and Kerry A. Corbett, "Optical Threshold Logic Analog to Digital Converters Using Self Electro Optic Effect Devices," *Proc. of SPIE MSN & MSS*, **vol. 5649**, Sydney, Australia, pp. 227-236 December 2004.

List of Figures

List of Tables

Chapter 1

Introduction

"It is clear to me that we will develop silicon neural systems, and that learning how to design them is one of the greatest intellectual quests of all time."
— Carver Mead

THIS chapter introduces Threshold Logic and outlines the motivation for the work in this Thesis. The structure of the Thesis and the contents of the Chapters are outlined, and the original contributions are summarized.

1.1 Context and Technology

Arithmetic operations, including addition and multiplication, are critical operations performed by microprocessors and digital signal processors. Consequently, the design of high speed and low power adder and multiplier circuits are key requirements in building a high performance computer. The computer designer's task is to find a good solution in the vast design space which spans from the process technology, circuits, architecture, software up to the algorithm level. In addition to computation speed, increasingly important is the design for low power dissipation and low cost.

Complementary metal-oxide silicon (CMOS) has now been the dominant underlying technology for building processors and related components for almost three decades (Pucknell and Eshraghian 1988, Weste and Eshraghian 1995). Rapid improvements in the speed and density of CMOS transistors has seen the clock frequency of microprocessors double every two years and they are expected to reach around 12 GHz by 2010 (SIA 2001). The clock frequency of a clocked microprocessor chip is the most important design parameter in the quest for higher performance. The minimum clock cycle time is often set by the adder delay in a processor. Adders are also used in floating-point arithmetic units, arithmetic logic units, memory addressing and program counter updates. Multipliers are important in many computational problems and complete multiplication units have been integrated in state-of-the-art digital signal processors and microprocessors.

Gordon Moore observed in 1965 that the number of transistors per unit of chip area had doubled every year since the integrated circuit was invented and he predicted that this trend would continue (Moore 1965). This rate has slowed down in processors, however memory density has continued to approximately double every 18 months and provides the current definition of Moore's Law. As the technology nears physical scaling limits, alternative fabrication technologies and techniques will need to be used in order for this trend to continue.

As process technology feature sizes continue to shrink, high speed digital VLSI circuit designers are increasingly faced with the challenging issues of, among others, power dissipation, signal integrity and design complexity. Threshold Logic based circuits are a potential candidate for alleviating these design issues.

1.2 Threshold Logic

The realization that human brains are far superior to computers in solving many problems including combinatorial optimization and image and speech processing, despite the building blocks being several orders of magnitude slower, has led to significant interest in the field of artificial neural networks. This Thesis considers the Boolean model of an artificial neuron, namely the linear Threshold Logic (TL) gate, that computes a neural-like Boolean function of binary inputs. The main issues in the study of networks (circuits) of TL gates include the estimation of their computational capabilities and limitations and the comparioson of their properties with those of traditional Boolean logic circuits based on AND, OR and NOT (AON) gates. There is strong evidence that TL circuits are more efficient than AON circuits in implementing a number of important functions including integer addition, multiplication and division (Beiu *et al.* 2003).

Threshold Logic has been studied since the 1960's (Minnick 1961, Winder 1963, Sheng 1965, Dertouzos 1965, Hu 1965, Lewis and Coates 1967, Sheng 1969). It is a possible alternative or complementary design technique to traditional Boolean logic due to the greater computational capability of TL gates. There has been a resurgence in TL research in recent years in the VLSI design community. Whereas the theoretical development of TL is mature, satisfactory CMOS implementations of TL gates and circuits based on these gates have only recently been developed.

Significant questions remain, concerning factors such as performance improvement and silicon area reduction, which have not been solved by such theoretical studies. For decades, TL gates have offered the promise of higher functional (computational) density over traditional logic using AON gates, leading to reduced gate count and therefore smaller logic networks and faster circuits, however, few practical designs have been demonstrated.

The computational power of the threshold logic network design style lies in the intrinsically more complex functions implemented by such gates when compared to conventional Boolean logic gates. This potentially allows circuit realizations, which require fewer threshold gates than standard logic gates (Muroga 1971). It is well known that an arbitrary Boolean function can be realized by a network of threshold gates and there are many examples where the number of gates required is considerably less than when conventional logic is used. In contrast, there are also functions where there are no savings.

More recently, a number of theoretical results have shown that TL networks with the number of levels polynomially bounded can implement functions that require networks with unbounded levels of standard logic gates (Impagliazzo *et al.* 1997). In particular, important functions like multiple-addition, multiplication, division and sorting can be implemented by polynomial-size threshold circuits of small constant depth.

Threshold gates operate on the so-called majority or threshold decision principle, where the gate's output value depends on whether the weighted arithmetic sum of its input values exceeds a threshold. Conventional simple logic gates, such as AND and OR gates, are special cases of threshold gates, and Boolean logic may be considered a sub-set of threshold logic. However, the usefulness of threshold logic as a design alternative is determined by the availability of efficient threshold gate implementations, where efficiency is measured in terms of delay, power dissipation and silicon area, or a combination of these.

With regard to silicon implementations, of particular importance is that threshold gates are more sensitive than conventional gates to transistor mismatch and noise. The sensitivities depend on the particular circuit implementation, and there is frequently a trade-off between increased logic capability and robustness.

A threshold logic gate is functionally similar to a hard limiting neuron without learning capability. The gate has n binary inputs $x_1, x_2, \ldots, x_n$, a set of corresponding n real number weights $w_1, w_2, \ldots, w_n$ and a single binary output y, as shown in Fig. 1.1.

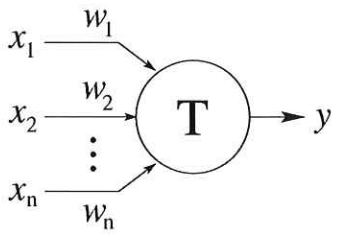


Figure 1.1. Model of the Threshold Logic Gate. The gate shows the outputs y after a thresholding operation T that sums inputs x_1 to x_n after multiplication by weights w_n.

The output y is given by (all operators algebraic):

$$y = \begin{cases} 1, & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq T \\ 0, & \text{otherwise.} \end{cases} \tag{1.1}$$

A device that implements this theoretical model must compute the linear weighted sum of the binary inputs, store the threshold value and compare the weighted sum

to this threshold. The various gate implementations proposed differ in the way they implement the weights, threshold and comparison. They rely on representing each distinct weighted sum of inputs and the threshold level by an analogue voltage or current and will be explored in Chapter 2.

A TL gate can be programmed, either statically (hard wired at design time) or dynamically, to realize many distinct Boolean functions by adjusting the threshold T and/or the weights w_i. For example, an n-input TL gate with unit weights, $w_i = 1$ and threshold $T = n$ will realize an n-input AND gate. The output of this gate is logic 1 if and only if all inputs are 1. By setting $T = n/2$ and leaving the weights unchanged, the gate computes a majority function. This versatility means that TL offers a significantly increased computational capability over conventional AND-OR-NOT logic.

Significantly reduced area and increased circuit speed can therefore *potentially* be obtained, especially in applications requiring a large number of input variables, such as computer arithmetic. For example, to implement the 8-input MAJORITY function (5-or-more-out-of-8) using conventional 2-input AND and OR gates requires a network of 47 gates of depth 6. The same function is readily implemented using a single 8-input threshold gate with unit weights and threshold equal to 5 ($T = 5$).

A small number of practical circuit examples have emerged since the mid 1990's, illustrating the potential of TL to reduce circuit area, interconnect (wiring) and to lower power dissipation when compared to conventional logic. The most advanced circuits presented to date consist of simple arithmetic components (Celinski *et al.* 2003c), including full-adders, compressors, parallel population counters, single high fan-in majority/OR/AND gates, Muller-C elements, 4-bit adders and embedded TL flip-flops.

Despite the resurgence of research interest in threshold logic, there remain a number of unresolved issues in the approaches proposed to date. This Thesis aims to address these issues:

- The TL gate designs proposed previously have had a high power dissipation relative to conventional CMOS logic.
- The TL gate designs proposed previously have a high delay relative to conventional CMOS logic.
- The TL gate designs occupy a large area relative to conventional CMOS logic.

- No large scale TL based circuit designs (complete arithmetic blocks) have been demonstrated that offer advantages over conventional CMOS logic.

- The silicon area savings and performance improvement, which result from the use of threshold gates for realizing a large scale logic design, have not previously been determined.

- The extent to which the theoretical advantages of TL (i.e. the exponential to polynomial reduction in gate count for common functions, such as PARITY), offset by practical implementation issues, including limited precision of weights.

The main aim of this Thesis is to demonstrate, using novel TL gate implementations and based on a large scale design, the advantages of the threshold logic circuit design paradigm over conventional CMOS logic in terms of improved power dissipation, speed performance and area efficiency.

1.3 Thesis Overview and Organization

Chapter 1 introduces threshold logic. Chapter 2 reviews the literature on threshold logic gate circuit implementations, and theoretical results related to the implementation of important arithmetic functions using threshold gate networks.

Novel contributions to threshold logic circuit design are presented in Chapter 4, including two new threshold gate designs, their performance evaluation and test chip measurement results. The application of these gates to the design of adder circuits is described in Chapter 5. In Chapter 6, the focus is on parallel counters and their application to the design of multiplier circuits. Chapter 7 shows how threshold logic techniques may be applied to technologies other than CMOS, including Gallium Arsenide and Self-Electro-Optical (SEED) devices. Chapter 8 includes a discussion of a mapping technique developed to aid in the implementation of logic functions using threshold gates.

Chapter 8 concludes the Thesis with a summary of the outcomes and conclusions, and recommendations for future work in threshold logic.

1.4 Original Contributions

For many years, research in linear threshold logic has focussed on two areas, namely computational circuit complexity and hardware implementation, with surprisingly little overlap between the two. This Thesis aims to establish a connection between the theory and implementation of threshold logic circuits. Establishing this connection will lead to solutions for practical problems and inspire new theoretical questions raised as a result of implementation issues.

The original contributions in this Thesis in the area of threshold logic technology lie in (i) threshold gate circuit techniques and (ii) threshold logic network design for important arithmetic operations, including addition and multiplication. The originality of these contributions is evidenced by the list of patents, journal and conference publications on Pages xix to xxii.

The work in this Thesis proposes two original CMOS threshold gate circuit topologies, Charge Recycling Threshold Logic (CRTL) and Self-Timed Threshold Logic (STTL). These gates exhibit superior delay performance and lower power dissipation compared to any previously proposed threshold gate. The development of CRTL and STTL was carried out in collaboration with Dr. José F. López at the Research Institute for Applied Microelectronics at the University of Las Palmas de G.C., Spain, and test chips were designed in cooperation with the Microelectronic Circuits and Analogue Devices Research Group in the Department of Engineering Science at the University of Oxford. The CRTL and STTL gates were experimentally verified, and a new weight-sharing circuit technique was proposed, which greatly reduces the circuit area cost of CRTL and STTL based circuits and may be applied to other TL gate designs.

The work presented here has also led to the development of a new, patented level sensitive latch for use in the design of high-speed, compact flip-flop circuits with low internal power dissipation and clock load as well as new, patented analog-to-digital converters based on optical Self-Electro-Optic Devices (SEEDs).

Following the development of CRTL and STTL, a delay modelling technique is proposed, based on the principles of Logical Effort (Sutherland and Sproull 1991), for the systematic evaluation and design for minimal delay of CRTL based circuits. The method developed is applicable to other sense-amplifier based TL gates.

To demonstrate the advantages of the threshold logic design paradigm on a non-trivial circuit problem, two new hybrid CRTL/CMOS-domino adder designs are developed.

The prefix-8 adder design is shown to be over 1 FO4 delay faster than any other 64-bit CMOS adder design proposed to date that does not require multiple non-overlapping clock phases. The adders were developed in collaboration with Assoc. Prof. Sorin Cotofana from the Computer Engineering Group at the Delft University of Technology in the Netherlands.

New parallel counters networks and multiplier circuits are developed, and the proposed partial product reduction schemes are shown to be significantly faster than previously published implementations.

The adder and multiplier circuits shown in this Thesis are the first published large-scale designs based on threshold logic which demonstrate conclusively the advantage of threshold logic over conventional CMOS in arithmetic application.

The original contributions of this Thesis offer benefits to high performance and low power VLSI processor design and to potential applications of threshold logic in compact low power, portable, wireless devices. Portability is important for aerospace applications, smart sensors and personal communication systems.

Chapter 2

Threshold Logic Arithmetic

THE focus of this chapter is the review of significant, recent developments in threshold logic arithmetic. The theoretical results in threshold logic addition and multiplication are reviewed and summarized, providing a context for the original contributions in these areas developed later in the Thesis.

2.1 Neural Computation

It is believed that the computational power of neural systems is related to their adaptive behaviour, and for this reason the majority of research in the field of neuromorphic analog VLSI is related to implementing neurons that learn or adapt (Holler *et al.* 1989, Hasler *et al.* 1995). It has also been demonstrated that the elementary function performed by neurons, namely the sum of weighted inputs followed by thresholding, is in itself, without learning, highly useful. The capability of such building blocks has been extensively studied and the computational complexity theory of circuits is well established. However, there has been little work reported in the literature on the link between theoretical results in threshold logic and work on improved silicon implementations particularly relating to the polynomial network size using TL versus the exponential circuit size using conventional AON logic required to compute useful Boolean functions.

Threshold networks have been shown to be particularly efficient in implementing various important arithmetic functions. For example, integer multiplication can be implemented by a polynomial size threshold circuit of constant depth. This means that to implement the threshold circuit to compute the multiplication of two n-bit integers, polynomially many (in n) threshold gates are required. In contrast, using conventional AON logic gates requires exponentially many gates.

2.2 Threshold Logic vs. Conventional Logic

The model of the threshold gate introduced in Chapter 1, is repeated for convenience in Fig. 2.1.

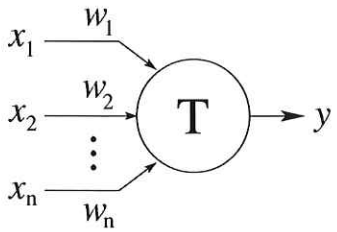


Figure 2.1. Model of the Threshold Logic gate. The gate shows the outputs y after a thresholding operation T that sums inputs x_1 to x_n after multiplication by weights w_n.

The Boolean function computed by such a gate is called a threshold function and it is specified by the gate threshold T and the weights $w_1, w_2, \ldots, w_n$, where w_i is the weight

associated with the i^{th} input variable x_i. It has been shown (Muroga 1971) that any linear threshold function can be implemented with integer weights. The threshold function implemented by the model shown in Fig. 9.1 can be written in a more compact form using the sgn notation:

$$y = \text{sgn}\left\{\sum_{i=1}^{n} w_i x_i - T\right\}, \tag{2.1}$$

where the sgn function is defined as follows, $\text{sgn}(x) = 1$ if $x \geq 0$ and $\text{sgn}(x) = 0$ if $x < 0$.

A threshold logic *network* will be considered an acyclic graph, in which each connection is associated with a weight and in which each node calculates the step activation function defined by Equation 2.1. A threshold logic *circuit* will be considered to be the transistor level CMOS implementation of a network, which consists of one or more threshold gates.

As an example, using the notation of Equation 2.1, the 3-input AND, 4-input MAJORITY and 2-input XOR functions may be written as in Equations 2.2-2.4 and Figs. 2.2(a)-(c) show the networks that implement the corresponding functions,

$$\text{AND}(x_1, x_2, x_3) = \text{sgn}(x_1 + x_2 + x_3 - 3) \tag{2.2}$$

$$\text{MAJ}(x_1, x_2, x_3, x_4) = \text{sgn}(x_1 + x_2 + x_3 + x_4 - 2) \tag{2.3}$$

$$\text{XOR}(x_1, x_2) = \text{sgn}\left\{x_1 + x_2 - 2 \cdot \text{sgn}(x_1 + x_2 - 2) - 1\right\}. \tag{2.4}$$

A device that implements the theoretical model in Fig. 9.1 must compute the linear weighted sum of the binary inputs, store the threshold value and compare the weighted sum to this threshold. The gates discussed in later sections follow this paradigm, but they differ in the way they implement the weights, threshold and comparison.

A fair question to ask is whether there is a justifiable reason for using threshold gates, given that any Boolean function can be systematically implemented by AND, OR and NOT (AON) gates. The reason is that for many important functions, including XOR, the number of AON gates grows exponentially with the number of inputs to the function (Wegener 1991). On the other hand, using threshold gates the number of gates is linear in the number of input bits. In general, a depth-2, AON circuit implementation of the n-bit XOR function requires at least $2^{n-1} + 1$ gates, and only $n + 1$ TL gates. The

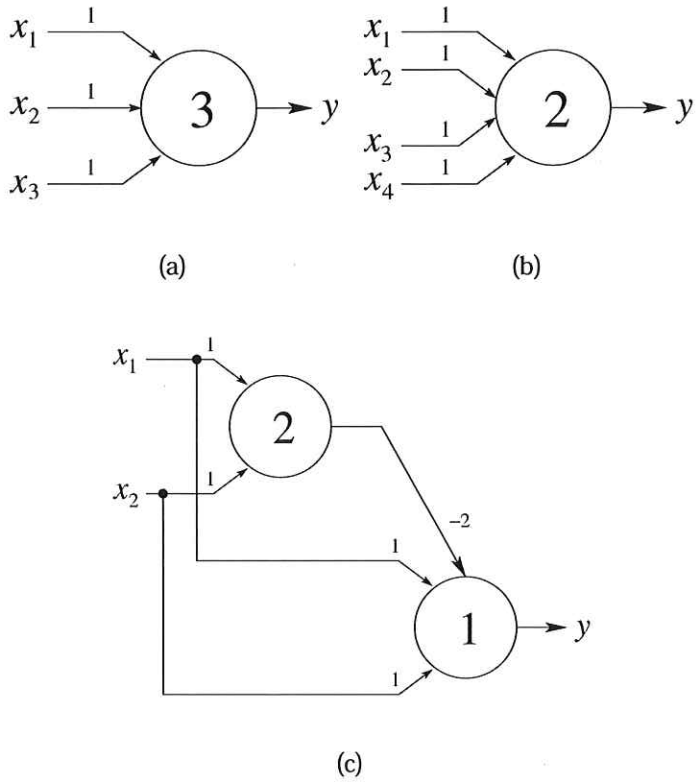


Figure 2.2. Example networks that implement (a) 3-input AND, (b) 4-input MAJORITY and (c) 2-input XOR. The binary logic inputs are denoted by x_i and the binary output of each gate is denoted by y. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The XOR gate is an example of a two-level threshold logic network.

power of TL based circuits is evident when we observe that for any single AON gate there is an equivalent TL gate, but the reverse does not hold.

Even though TL circuits are computationally more powerful, the gate implementations, as will be discussed in the following sections, are more complex and therefore may potentially require more area and dissipate more power. The central question to ask is to what extent are the theoretical advantages of TL, i.e. the exponential to polynomial reduction in gate count, offset by the practical implementation issues. The answer to this question depends on the complexity of the function to be implemented —typically the greater the number of inputs to the gate the greater the advantage of employing TL.

From a practical perspective, it is insufficient to contend with polynomial bounds on weights and gate number, as the absolute numbers may still be prohibitively large for

leading to an efficient or even feasible implementation from a reliability perspective. Hardware implementations use a finite number of bits for weight storage (whether programmable or hard-wired) and this translates to a limited precision for the weights . The precision of the implementation is directly related to its cost, usually in circuit area and therefore increased capacitance leading to increased power dissipation. Increasing the precision requires that the TL gate be able to reliably distinguish a smaller quantity, either voltage or current, typically in a noisy circuit environment.

If the weighted sum is in the form of a voltage on a capacitor, a higher precision leads to a larger area of that capacitor (as capacitor mismatch is inversely related to area) and a greater amount of energy required to charge the capacitor. More importantly, the ability of a sense-amplifier , which implements the thresholding operation, to resolve the small voltage difference is dominated by the input-referred offset voltage. This in turn is related to transistor mismatch in the sensing circuit, which is inversely related to transistor area, leading to larger transistor requirements.

It is instructive to determine whether there is an inherent advantage of TL gates that can reliably handle a large number of inputs. We assume that the size of an n-input TL gate is approximately proportional to n. Under this assumption, it can easily be shown that a TL gate implementing an n-input AND gate can be replaced by a tree of 2-input AND gates of which the size is also approximately proportional to n, since the size of the tree is proportional to the number of 2-input AND gates, which is approximately n.

However, this scenario changes when we consider a general n-input threshold function and construct a network of two-input TL gates to implement the same function. It turns out that the size of this tree is *at least* proportional to n^2 (Mead 1989), an order of magnitude larger than the single gate. This can be understood by comparing the information content carried by the signal nodes within the 2-input network and the analog nature of the summation node of the single TL gate. Within the 2-input network, the intermediate discrete gate outputs carry less information than the intermediate analog nodes and hence a larger network is required to compensate for this lost information by providing a larger number of discrete signal nodes. In addition to allowing a smaller circuit size for implementing a given threshold function, high fan-in TL gates lead to networks with reduced depth and hence lower delay.

2.3 Threshold Logic Addition and Multiplication

In the last decade, significant progress has been made in the theory of arithmetic circuit design based on threshold logic. In this section, recent constructive results (i.e. where the reported method prescribes how to select the weight values and how to construct the network) for the addition and multiplication of two binary integers are summarized. The main issues to consider relate to the network depth (delay), maximum required weight values and gate fan-in. Despite the resurgence of activity in this area during the last decade, there have been very few implementations. The earliest examples are the *Mark I Perceptron* built by Rosenblatt (1958), Widrow's *Memistor* (1960) and the DONUT computer (Coates and Lewis 1964). Recent threshold logic based circuit implementations may be found in the floating point co-processor of MIPS R2010 (Johnson 1988) , the SUN Sparc V9 (Lev *et al.* 1995) and the Itanium 2 microprocessor in 2002 (Naffziger *et al.* 2002).

2.3.1 Addition Networks

The addition of two n-bit binary numbers, $X = x_{n-1}, x_{n-2} \ldots x_0$ and $Y = y_{n-1}, y_{n-2} \ldots y_0$, is defined as the unsigned sum, $S = s_n, s_{n-1}, s_{n-2} \ldots s_0$, of X added to Y. There are many well known techniques for performing addition (Ling 1981, Brent and Kung 1982, Knowles 1999, Beaumont-Smith and Lim 2001). The sum bits s_i may be computed using $s_i = x_i \oplus y_i \oplus c_{i-1}$ for $i = 0, 1, \ldots, n-1$, and $s_n = c_n$. The carry bits are $c_i = (x_i \cdot y_i) + (x_i \cdot c_{i-1}) + (y_i \cdot c_{i-1})$, $c_{-1} = 0$. Alternatively, the sum bits, s_i may be written as $s_i = (x_i \cdot y_i \cdot c_{i-1}) + [\bar{c}_i \cdot (x_i + y_i + c_{i-1})]$ and this expression may be implemented using a single TL gate. The addition operation essentially reduces to computing the carries, c_i, from which the sum bits are computed in the final stage of the process.

To date there have been two approaches in the study of TL based adders, theoretical and practical. The theoretical approach is concerned with the computational circuit complexity, and this may be divided into two categories. The first is the set of results that establish existence of solutions and upper and lower bounds on the various parameters such as network depth and size. The second set includes results that are constructive and provide a method of determining the weights and interconnection of TL gates, and is of particular interest to finding efficient circuit implementations.

Theoretical research has also focused on the weights, or more precisely on the capabilities of TL networks with restricted weights. Particular classes of functions are

amenable to physically realizable implementations using gates with low fan-in, small weights and in shallow depth and polynomially sized networks. This class includes addition and multiplication.

The existence of a depth-2 adder of polynomial size is well known (Siu and Bruck 1990). It was followed by the first constructive depth-2 majority gate based adder (i.e. using TL gates with weights of -1, 0 or +1) of size $O(n^4)$, first published in 1991 and commonly available in (Alon and Bruck 1994). Table 2.1 provides a more detailed summary of these results relating to network depth, TL gate count, maximum weight size and maximum gate fan-in, adapted from (Beiu 2003).

The chronilogically ordered list below summarizes the notable result on the depth, weight dynamic range and network size for the addition operation.

- 1991 – depth-3 of size $n^2/2 + 7n/2 - 1$ (Siu *et al.* 1991)
- 1991 – depth-7 of size $O(n\log n)$ (Siu *et al.* 1991)
- 1993 to 1994 – depth-5 of size $O(n\log n)$ (Beiu *et al.* 1993, Beiu *et al.* 1994b, Beiu *et al.* 1994a)
- 1994 – depth-4 to $3 + \log n$ of size $7n$ to $2n\log n + 5n$ (Beiu 1994, Beiu *et al.* 1994a)
- 1994 – depth-3 of size $5n$ with exponential weights (Beiu 1994, Beiu *et al.* 1994a)
- 1995 to 1996 – depth-3 of size $6n + 2\lceil n/\lceil n\rceil\rceil$ with exponential weights (Cannas 1995, Vassiliadis *et al.* 1996)
- 1996 – depth 3 of size $O(n^2/k\log n)$ for any $1 \leq k \leq n/\log n$ with polynomial weights (Vassiliadis *et al.* 1996)
- 1999 – depth-2 of size $n^2 + 2n$ with exponential weights (Beiu 1999)
- 1999 – depth-3 of size $n\sqrt{n} + 4n$ and weights $2^{\sqrt[1.4]{n}}$ (Beiu 1999)
- 1999 – depth-2 of size $2n$ with exponential weights 2^n (Ramos and Bohórquez 1999)
- 1999 – depth-3 of size $4n$ with weights $2^{\sqrt{n}}$ (Ramos and Bohórquez 1999).

The above results show the tradeoffs in terms of adder depth vs. size and the related weight size. The progressive improvements in small depth adders have reduced the

size of depth-2 networks from $O(n^4)$ with constant weights to $O(n)$ with exponential weights. In depth-3 and and depth-4, the size has been reduced to $O(n)$ with exponential weights. These results will be used in Chapter 4 to develop an efficient TL 64-bit CMOS adder implementation.

Although network size and weight magnitude may not grow exponentially with the adder width, the best schemes for shallow depth addition still require polynomially increasing weights. This may lead to *absolute* values for network size (TL gate count), weights and fan-in to be prohibitively large for a feasible CMOS implementation, frequently by an order of magnitude. Network depth alone is also insufficient to evaluate the likely delay or to compare delay of implementations, largely because in a circuit implementation the true delay depends on technological constraints such as wire loading and fan-out.

Table 2.1. Summary of theoretical constructive results on threshold logic addition including network depth, gate count, maximum weight and maximum fan-in. The symbol n denotes the operand width, c denotes a constant, $w_{\max}$ denotes the value of the maximum weight in the network, δ denotes an arbitrary parameter constraint and $\lceil\ \rceil$ denotes the ceiling operator (lowest integer greater than). Adapted from Beiu et al. [2003].

Year	Depth	# Gates	W_{max}	$Fan\text{-}in_{max}$
Siu & Bruck 1990	2	n^c	—	—
Alon & Bruck 1991	2	$O(n^4)$	{-1,0,+1}	n^4
Siu *et al.* 1991	3	$(n^2+7n-2)/2$	{-1,0,+1}	$2n$
	7	$O(n\log n)$	{-1,0,+1}	
Beiu *et al.* 1994	5	$O(n\log n)$	2^n	$2n$
Beiu 1994	$3+\lceil\frac{\log n}{\log\delta-1}\rceil$	$5n+2n\lceil\frac{\log n}{\log\delta-1}\rceil$	$2^{\delta/2}$	δ
	4	$7n$	2^n	$2n$
	d	$2dn-n$	$2^{\sqrt[d-3]{n}}$	$2^{\sqrt[d-3]{n}}$
	$3+\frac{\log n}{\log(\log w_{\max})}$	$5n+\frac{2n\log n}{\log(\log w_{\max})}$	$w_{\max}$	$2\log w_{\max}$
Beiu 1994	$2+\lceil\frac{\log n}{\log\delta-1}\rceil$	$3n+2n\lceil\frac{\log n}{\log\delta-1}\rceil$	$2^{\delta/2}$	δ
	3	$5n$	2^n	$2n$
	4	$7n$	$2^{\sqrt{n}}$	$2\sqrt{n}$
	d	$2dn-n$	$2^{\sqrt[d-2]{n}}$	$2^{\sqrt[d-2]{n}}$
	$2+\frac{\log n}{\log(\log w_{\max})}$	$3n+\frac{2n\log n}{\log(\log w_{\max})}$	$w_{\max}$	$2\log w_{\max}$
Vassiliadis *et al.* 1996	3	$6n+2\lceil n/\lceil\sqrt{n}\rceil\rceil$	$2^{\lceil\sqrt{n}\rceil}$	$2\lceil\frac{n}{\lceil\sqrt{n}\rceil}\rceil+3$
Beiu 1999	$1+\lceil\frac{\log n}{\log\delta-1}\rceil$	$\frac{n\delta}{2}+2n\lceil\frac{\log n}{\log\delta-1}\rceil$	$2^{0.7\delta}$	δ
	d	$n\sqrt[d-1]{n}+2n(d-1)$	$2^{1.4\sqrt[d-1]{n}}$	$2^{\sqrt[d-1]{n}}$
	$1+\frac{\log n}{\log(\log w_{\max}^{0.7})}$	$\frac{2n\log n}{\log(\log w_{\max}^{0.7})}+O(n)$	$w_{\max}$	$1.4\log w_{\max}$
Ramos & Bohórquez 1999	d	$2(d-1)n-2(d-3)\sqrt[d-1]{n^{d-2}}$	$2^{\sqrt[d-1]{n^{d-2}}}$	$2^{\sqrt[d-1]{n^{d-2}}}+1$
	$1+\frac{\log n}{\log(\log w_{\max})}$	$\frac{2n\log n}{\log(\log w_{\max})}+O(n)$	$w_{\max}$	$2\log w_{\max}+1$

2.3.2 Multiplication Networks

Multiplication is closely related to addition. The product of two n-bit numbers requires the computation of the sum of n $2n$-bit numbers that form the rows of the partial products matrix. Siu and Bruck (1990) showed, based on considerations of multi operand addition, that the product of two n-bit numbers can be computed in depth-4 using a polynomially sized (in n) network with polynomially bounded weights.

The multiplication operation consists of three steps. In the first stage of the multiplier, the partial product bits are generated, typically using AND gates. In the second stage, the matrix of partial products is reduced to two numbers which are simply added in the third stage.

The second stage is by far the most expensive in terms of the number of gates and fan-in per gate. Although it is theoretically possible to implement the second stage using a depth-2 network, the excessive hardware requirements render conventional approaches prohibitive. Lauwereins and Bruck (1991) observed that a depth-2 implementation using Block Save Addition (BSA) (Siu and Bruck 1990) of the second stage of a 32×32-bit multiplier requires gates with fan-in ≥1000, and over 20 million connections. In BSA, the partial product columns of n numbers are divided into column-blocks of $\log n$ bits. The sum of one column-block is therefore at most $2\log n$ bits wide and hence overlaps only the column-block to which it belongs and at most one adjacent column-block. As a result, the even and odd column-block sums do not overlap and the sums of the even and odd column-blocks are concatenated to two partial sum-words. Siu and Bruck (1990) proved that the sum of a column-block of n numbers $\log n$ bits wide can be determined in a depth-2 polynomially bounded network. Table 2.2 (Lauwereins and Bruck 1991) lists the number of gates and fan-in requirements for common multiplier widths using the BSA scheme.

In order to reduce the hardware requirements, Lauwereins and Bruck (1991) described two minimizations, while maintaining the depth-2 strategy for the second stage. While this resulted in significant wiring and gate savings, practical implementation based on this method for word lengths beyond 16 bits is still not feasible. To reach a practical hardware solution, Lawereins explored trading off cost for an increase in delay, based on a Hierarchical Block Save Addition (HBSA) approach. Instead of performing the entire reduction of the partial product bits in a single step using a depth-2 network, the n partial product rows are broken down into blocks of numbers, and each block is reduced to two numbers in one reduction step, and the process is repeated until

Table 2.2. Block Save Addition (BSA) hardware requirements for the partial product reduction to two partial sum-words in depth-2. The table shows that the number of TL gates and the maximum gate fan-in are very high for a 32×32 multiplier.

Multiplier Width	# TL Gates	Max Fan-In
4×4	272	16
8×8	1,950	64
16×16	16,448	256
32×32	133,250	1024

Table 2.3. Partial Product Reduction using Hierarchical Block Save Addition (HBSA) and Telescopic Sums with Gate Sharing (TSGS). The table shows a significant reduction in the number of gates required compared to the BSA approach.

Multiplier Width	# TL Gates - HBSA	# TL Gates - TSGS	Max Fan-In
4×4	88	40	16
8×8	310	144	64
16×16	1264	464	256
32×32	4966	1820	1024

two numbers remain. Based on a block size of 8, the 32×32-bit multiplication second stage now requires approximately 5300 gates with maximum fan-in of 24 (Lauwereins and Bruck 1991). Further optimizations, based on the observation that certain bits in the partial product reduction steps are constantly zero, reduce the gate number by approximately a factor of two. It was also observed that compared to carry save addition (CSA) using various reduction schemes, the delay of the HBSA is significantly lower for the same fan-in requirement.

Vassiliadis *et al.* (1995) proposed further optimizations to the BSA scheme by introducing a scheme which incorporates Minnick's telescopic sums (Minnick 1961) and a minimization based on gate sharing. While the scheme has the same asymptotic bounds for the number of gates as that proposed by Lauwereins, for practical multiplier sizes it reduces the gate count and number of wires by approximately a factor of two, and the gate count comparison for a depth-2 reduction is shown in Table 2.3 (Vassiliadis *et al.* 1995). The maximum fan-in is the same for both schemes.

2.4 Chapter Summary

This Chapter has introduced the concept of threshold logic and its potential advantages when compared to conventional logic design. An outline of the practical issues faced in the implementation of TL circuits was given and the significant developments in the design of threshold logic based networks for the arithmetic operations of addition and multiplication were summarized. The following Chapter takes the opposite perspective to the network theoretical results presented thus far and looks at the details of implementing threshold logic circuits using CMOS technology.

Chapter 3

Threshold Logic Circuit Implementations

THE focus of this Chapter is the review of significant, recent developments in Threshold Logic circuit design techniques using CMOS. A number of gate implementations are described in detail and their relative performance is evaluated and compared. A survey of notable practical applications in computer arithmetic is given including a comparison of the relative improvement in one or more metrics for the same function when compared to conventional static CMOS logic.

3.1 Threshold Gate Implementations

Despite the development of threshold logic over four decades ago, and the theoretically motivated promise of superior performance circuits compared to AON implementations, lack of efficient physical realizations has meant that TL has, over the years, had little impact on VLSI. Efficient TL gate realizations have recently become available (Shibata and Ohmi 1991, Kotani *et al.* 1995, Avedillo *et al.* 1995, Özdemir *et al.* 1996, Bobba and Hajj 2000, Celinski *et al.* 2001, Padure *et al.* 2002b), and a number of applications based on TL gates have demonstrated its ability to achieve high operating speed and significantly reduced area (Özdemir *et al.* 1996, Celinski *et al.* 2001, Padure *et al.* 2002b, Celinski *et al.* 2003a, Celinski *et al.* 2004).

A number of recently proposed CMOS TL gate designs are now reviewed. Rather than presenting an exhaustive review of all previously reported gate designs of note, the main focus is on those designs that have led to the design of high-speed TL based circuits. The reason for this is to establish a framework for developing digital systems, based on CMOS TL gates, which have performance competitive with that of conventional static or dynamic-CMOS. A short summary of other notable gate implementations is also included, a comprehensive historical survey may be found in (Beiu *et al.* 2003).

The requirements for high speed and high fan-in lead to the consideration of predominantly dynamic, differential gates. The gates considered here may be classified as either voltage mode or current/conductance mode. The voltage mode designs discussed here include neuron-MOS (ν**MOS**) (Shibata and Ohmi 1992) (the only static gate considered), Capacitive Threshold Logic (**CTL**) (Özdemir *et al.* 1996), Latched neuron-MOS (**L-**ν**MOS**) (Kotani *et al.* 1998) and Charge Recycling Threshold Logic (**CRTL**) (Celinski *et al.* 2001). The current mode designs include Latched Comparator Threshold Logic (**LCTL**) (Avedillo *et al.* 1995), Equalized Current-Mode Threshold Logic (**ECMTL**) (Bobba and Hajj 2000) and Differential Current-Switch Threshold Logic (**DCSTL**) (Padure *et al.* 2002b).

The important gate features are speed of operation, maximum sum-of-weights (or, equivalently, the fan-in where all weights are equal), area and power dissipation. The maximum sum-of-weights sets the minimum signal level (voltage or current) which is required to be resolved by the comparator in the TL gate. A high value of fan-in generally leads to shallow depth logic networks and therefore reduced circuit delay . Another desirable feature is the ability to easily implement negative weights, since some

arithmetic operations are conveniently expressed in the form of Equation 2.1 where the values of w_i are negative. Ease of dynamic re-programmability of the weights and threshold may also be important in reconfigurable logic operations.

3.1.1 Voltage/Charge Mode TL Gates

The voltage mode TL gate implementations described here are based on the principle of the capacitive synapse proposed in (Cilingiroglu 1991). The underlying concept is the use of an array of capacitors to implement the weighted sum of inputs, connected to the gate of a MOS transistor (the so-called νMOS transistor (Shibata and Ohmi 1991), also known as the Multiple Input Floating Gate transistor). Typically, in CMOS technology these capacitors are implemented between the polysilicon 1 and polysilicon 2 layers. Figures 3.1 and 3.2 depict the four voltage mode gates considered here, including νMOS, Capacitive Threshold Logic, Latched νMOS and Charge Recycling Threshold Logic.

νMOS

The νMOS transistor based static TL gate proposed in the early 1990s uses an array of capacitors to implement the input weights, followed by one or more inverters to implement the thresholding operation, as shown in Fig. 3.1(a).

The input of the first inverter in the chain in Fig. 3.1(a) is effectively floating, and its voltage is given by

$$\text{Œ} = \frac{\sum_{i=1}^{n} C_i x_i}{C_{\text{tot}}}, \tag{3.1}$$

where C_{tot} is the sum of all capacitances at the floating node, including parasitic capacitances. The switching point of the first inverter in the chain, the *primary inverter*, determines the gate threshold, and the subsequent inverters serve to generate a full swing rail-to-rail output voltage. This expression assumes that no charge is initially present on the floating node. The presence of this charge is, however, unavoidable as a result of fabrication, and for this reason techniques such as UV erasure must be used (Shibata and Ohmi 1991). The νMOS gate suffers from limited fan-in (typically <12) due to variability of the switching threshold of the primary inverter as a result of process variations. It is also relatively slow and has a high data dependent power dissipation as a result of the static current flowing from V_{dd} to Gnd in the primary inverter when the floating gate voltage is not at one of the supply rails. The gate is highly

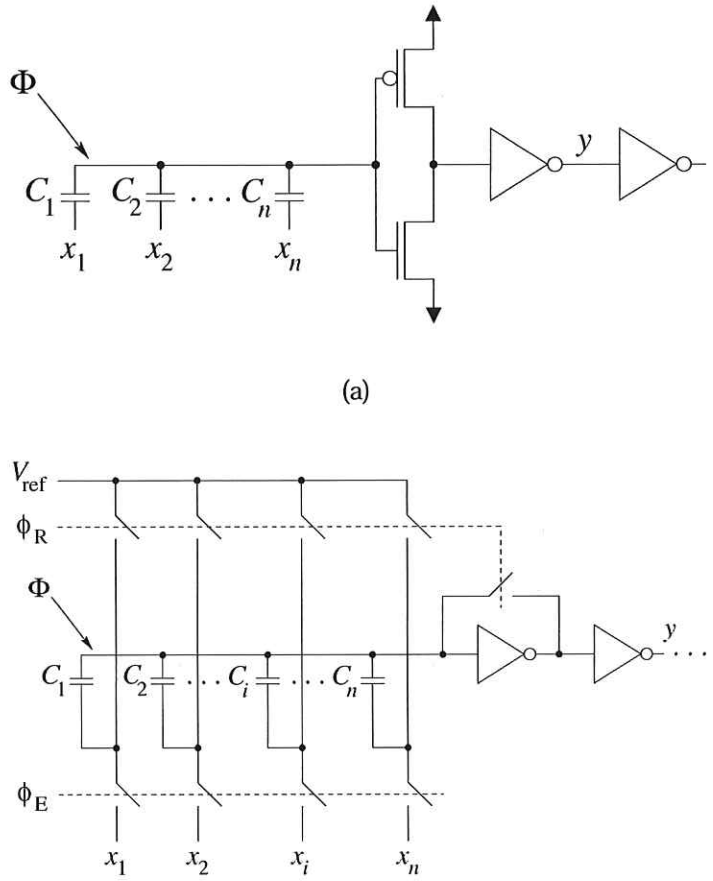


Figure 3.1. Single-ended voltage mode threshold gates. These include (a) νMOS and (b) Capacitive Threshold Logic. The inputs x_i are applied to weights implemented using capacitors C_i. The weighted sum of inputs corresponds to the floating gate voltage ϕ in both cases. The inverter chain is used to perform the thresholding operation to generate output y.

compact and relatively simple to design, and the gate threshold may be programmed by adding control capacitors (Shibata and Ohmi 1992). Negative weights can not be implemented in the νMOS gate.

Capacitive Threshold Logic

To overcome the limited fan-in of νMOS, Capacitive Threshold Logic was proposed (Özdemir *et al.* 1996). The CTL gate is similar to the clocked-νMOS (Kotani *et al.* 1995) gate, proposed at approximately the same time. The circuit schematic of the CTL gate

is shown in Fig. 3.1(b). An n-input CTL gate comprises n weight-implementing capacitors (C_i) followed by one or more inverters which function as voltage comparators to generate the binary output. The main difference between CTL and neuron-MOS lies in the way the value of the gate threshold is set. In CTL, the threshold value is a function of an external reference voltage V_{ref}.

The CTL gate operates in a two-phase non-overlapping clock scheme consisting of a reset phase ϕ_R and an evaluate phase ϕ_E. During the reset phase the row voltage ϕ is reset to the threshold voltage V_{th} of the primary inverter, while the capacitor bottom plates are set to the reference voltage V_{ref}. During the evaluation phase, the row voltage is perturbed from V_{th} by the inputs x_i which now become capacitively coupled onto the effectively floating input to the primary inverter. The magnitude of this perturbation is a function of V_{ref}, which effectively controls the threshold of the gate. The floating gate voltage, during the evaluation phase, is given by

$$\text{Œ} = V_{\text{th}} + \frac{\sum_{i=1}^{n} C_i(x_i - V_{\text{ref}})}{C_{\text{tot}}}. \tag{3.2}$$

Due to the reset mechanism, CTL does not require UV erasure of residual floating gate charge. The gate also has a significantly increased fan-in of up to 255 inputs (Özdemir *et al.* 1996) compared to νMOS since the switching point variability of the primary inverter no longer influences the effective gate threshold. This is similar to the offset cancellation mechanism in chopper type comparators.

The drawbacks of the CTL gate are that it requires a complex clocking scheme and it also suffers from high static power dissipation and low speed for similar reasons as νMOS. The gate also requires an analog reference voltage to set the threshold value which leads to difficulties in implementing CTL circuits with a large number of gates with different threshold values. The analog-reference voltage problem was overcome by the introduction of the Capacitor Programmable Capacitive Threshold Logic gate which requires only binary logic levels and $V_{\text{dd}}/2$ for programming (Stokman *et al.* 1998).

Latched neuron-MOS

The Latched νMOS gate (Kotani *et al.* 1998, Luck *et al.* 2000)—also referred to as Sense-Amplifier νMOS—was introduced to overcome the high power dissipation of the νMOS and CTL gates. The gate uses a current-controlled latch-sense amplifier circuit (cross

coupled transistors M1-M4) instead of an inverter to perform thresholding. The gate uses the previously described νMOS structure to compute the weighted sum of inputs as illustrated in Fig. 3.2(a). Device parameter fluctuations are compensated by the differential amplifier configuration and the gate was shown to have significantly reduced power dissipation when compared to CTL (or clocked-νMOS) and static CMOS (Kotani *et al.* 1998). The fan-in is also expected to be higher than νMOS and negative weights may be implemented, due to the differential circuit structure. The threshold value may also be programmed conveniently by adding additional control capacitors. UV erasure or other technological measures must be used to remove residual floating gate charge (Luck *et al.* 2000).

Charge Recycling Threshold Logic

Another CMOS threshold gate realization has been proposed that was intended to solve the problem of high power dissipation and low speed of CTL and νMOS. The CRTL gate has low power dissipation while providing very high operating speed (Celinski *et al.* 2001). Fig. 3.2(b) shows the circuit of the Charge Recycling Threshold Logic (CRTL) gate. The sense amplifier (cross coupled transistors M1-M4) generates output y and its complement $\bar{y}$. Precharge and evaluate are specified by the enable clock signal E and its complement $\bar{E}$. The inputs x_i are capacitively coupled onto the floating gate ϕ of M5, and the threshold is set by the gate voltage T of M6. The potential ϕ is given by $\phi = \sum_{i=1}^{n} C_i x_i / C_{\text{tot}}$, where C_{tot} is the sum of all capacitances, including parasitics, at the floating node. Weight values are thus realized by setting capacitors C_i to appropriate values. Negative weights may be easily implemented by using a second capacitive array on the gate of M6 to generate T. The gate was shown to reliably operate at high speed (with fan >20) (Celinski *et al.* 2001). The gate does not have static power dissipation. The drawback of the gate is the requirement for UV erasure of residual floating gate charge. A related self-timed gate implementation has also been proposed (Celinski *et al.* 2002a).

3.1.2 Current/Conductance Mode TL Gates

The second class of TL gate implementations described here is based on the principle of comparing current (or conductance). Early conductance mode gates based on pseudo-nMOS or output-wired inverters (Schultz *et al.* 1990) were fast but suffered from high power dissipation and limited fan-in. Recently, a number of high-speed differential

solutions have been proposed, including Latched Comparator Threshold Logic, Equalized Current-Mode Threshold Logic and Differential Current-Switch Threshold Logic.

The underlying principle of these schemes is the use of an array of MOS transistors to implement the weighted sum of inputs in the form of a net conductance, which is compared to the net conductance of a similar array of MOS transistors used to implement the gate threshold. To obtain high speed, the comparator is implemented in the form of cross-coupled inverters. Figures 3.3 and 3.4 depict the three current mode gates considered here.

Latched Comparator Threshold Logic

The Latched Comparator Threshold Logic (Avedillo *et al.* 1995) gate shown in Fig. 3.3(a) consists of a current-controlled latch formed by transistors M1-M4. The input transistor array is a set of identical NMOS parallel transistors with gates connected to inputs $x_1,\ldots,x_n$. Similarly, parallel transistors with gates connected to $t_1,\ldots,t_n$ implement the threshold. Input weights are implemented by connecting each input signals to one ore more input transistor gates. Two additional transistors, Mx and My, ensure correct operation in the event that the weighted sum is equal to the threshold value.

The clock signal (`clk`) controls the precharge and activation of the sense circuit. The gate has two phases of operation, the precharge phase and the evaluate phase. When `clk` is low the nodes y_1 and $\bar{y}_1$ are precharged to V_{dd}. When `clk` is high, the input array and threshold setting array draw unequal currents from the precharged nodes y_1 and $\bar{y}_1$. The current-controlled latch amplifies the difference in potential now present between y_1 and $\bar{y}_1$, accelerating the transition until either y_1 or $\bar{y}_1$ reaches V_{dd}. In this way the circuit structure determines whether the net conductance of the input array is greater than or less than the net conductance of the threshold array and a TL gate is realized.

The gate does not have static power dissipation. Reliable operation with fan-in up to 20 has been reported, the gate can implement negative weights and the threshold value may be dynamically programmed. The drawback of the gate is the large capacitance at nodes x and y which slows down the sensing. To overcome this, a modified implementation in the form of Cross-coupled Inverters with Asymmetric Loads Threshold Logic (Ramos *et al.* 1998a) was proposed.

Equalized Current Mode Threshold Logic

The Equalized Current Mode Threshold Logic (Bobba and Hajj 2000) gate is shown in Fig. 3.3(b). The gate also consists of a current-controlled latch formed by transistors M1-M4 and two banks of identical PMOS input transistors and threshold setting transistors. The important distinction is that the voltage swing is limited on the input nodes $x_1,\ldots,x_n$ and internal nodes to provide very low power dissipation on interconnects while maintaining high switching speed. The interconnect voltage is generated by a low voltage-swing interconnect driver (not shown) which converts the rail-to-rail outputs y and $\bar{y}$ to a low voltage which is used to drive subsequent ECMTL gate inputs.

The clock signal (`clk`) controls the equalization and activation of the sense circuit. The gate has two phases of operation, the equalize phase and the evaluate phase. When `clk` is low the nodes y and $\bar{y}$ are equalized. When `clk` is high, the input array and threshold setting array draw unequal currents from V_{dd} and charge nodes y and $\bar{y}$ at different rates. The current-controlled latch regenerates the difference in potential between y and $\bar{y}$, accelerating the transition to full swing outputs.

Reliable operation of ECMTL with up to 8 inputs has been reported, the gate can implement negative weights and the threshold value may be dynamically programmed. The main drawback of the gate is the requirement of a low-voltage swing driver with a separate low voltage supply to drive the gate interconnects. The gate has no static power dissipation.

Differential Current-Switch Threshold Logic

Finally, the Differential Current-Switch Threshold Logic (Padure *et al.* 2002b) gate is shown in Fig. 3.3(c). The gate is based on the Differential Cascode Voltage Switch Transistor Logic (DCSTL). It consists of a cross-coupled inverter latch formed by transistors M1-M4 and two banks of identical n-channel MOS input transistors and threshold setting transistors. The clock signal (`clk`) controls the precharge and activation of the latch. The gate has two phases of operation, the precharge phase and the evaluate phase. When `clk` is low the nodes y_1 and $\bar{y}_1$ are precharged to V_{dd}. When `clk` is high, the input array and threshold setting array draw unequal currents and discharge nodes y_1 and $\bar{y}_1$ at different rates. The latch regenerates the difference in potential between y and $\bar{y}$, accelerating the transition to full swing outputs. Transistors M6 and M7 reduce the voltage swing on nodes `a` and `b` and cut off the static current during evaluation, reducing the power dissipation.

Reliable operation has been reported for DCSTL gates with up to 64 inputs (Padure *et al.* 2003), the gate can implement negative weights and the threshold value may be dynamically programmed. The gate has no static power dissipation.

3.1.3 Other Gate Implementations

A large number of VLSI circuit implementations of the threshold gate have been proposed (Beiu *et al.* 2003). In addition to the previously mentioned "high performance" gates, other notable implementations include:

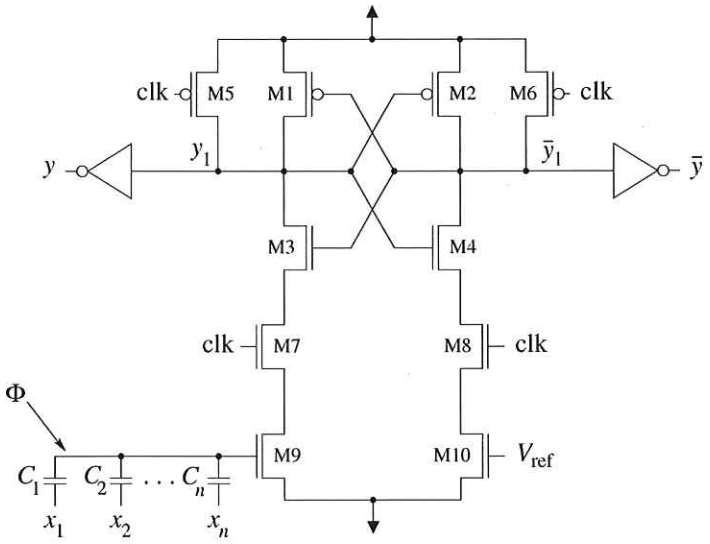
- Static CMOS majority gate based circuits.
- Null Convention Logic (NCL) (Fant and Brandt 1994).
- Complementary Pass Transistor Logic (CPL) (Quintana *et al.* 2001).
- Balanced-Capacitive Threshold Logic (B-CTL) (Garcia *et al.* 2000).
- CMOS Capacitor Coupling Logic (C^3L) (Huang and Wang 2000).
- Output Wired Inverter (also known as Ganged-CMOS) (Schultz *et al.* 1990) and its variations.
- Pseudo-nMOS and its variations.

3.1.4 Gate Implementation Comparison

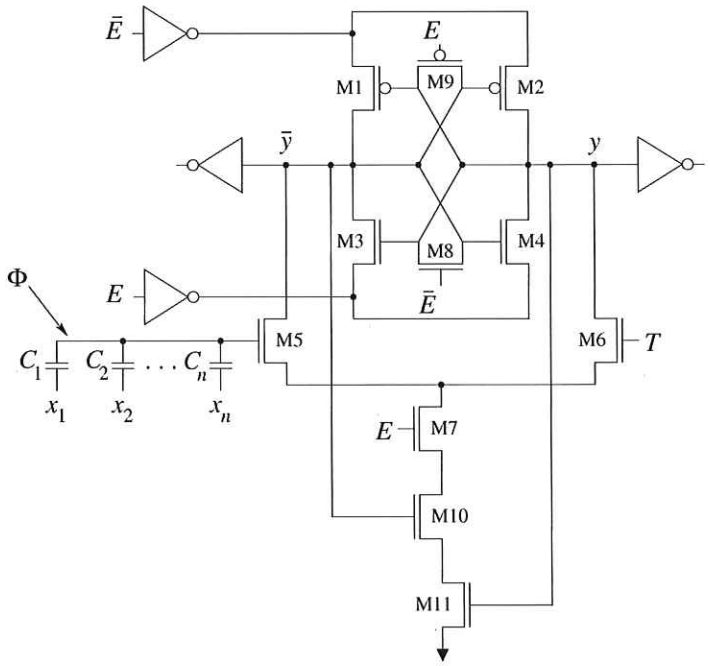
To compare the performance of the gates, the results published in the literature must be normalized. The main difficulties in providing a fair comparison are that results for the different gates are reported for various process technologies and in different circuit applications. More often than not, results presented are based on simulations alone and not measurements from fabricated circuits. This may lead to optimistic delay, power dissipation or estimated area results where layout techniques to minimize noise and reduce the influence of device parameter variation to ensure robust operation are not considered. In some instances, the results presented may also be skewed to favour a particular figure of merit (delay, area, power, power-delay product, maximum sum-of-weights etc.) without providing a complete evaluation. For these reasons

the evaluation provided here is in the form of a collection of reported results in the literature, for delay and power dissipation, and a qualitative assessment of the relative advantages of each gate. From these results, conclusions may be drawn about the suitability of a particular gate for a given circuit design problem.

The focus of this section is on representative gates and small-scale circuits. Large scale arithmetic circuit applications are the subject of the next section. Table 3.1 summarizes the delay results reported in the literature for single gate circuits for each TL gate discussed in this work. The table gives the circuit function implemented using the given gate, the process technology used and the gate delay. In the case of the L-νMOS gate, delay is not explicitly reported, so a delay number is inferred from the reported data rate. The CTL delay numbers exclude the reset-phase time, which is of the order of 1000's of evaluation cycles (Padure *et al.* 1999).



(a)



(b)

Figure 3.2. Differential voltage mode threshold gates including (a) Latched νMOS and (b) Charge Recycling Threshold Logic. In both gates, the inputs x_i are applied to weights implemented using capacitors C_i and the weighted sum of inputs corresponds to the floating gate voltage ϕ in both cases. The gate thresholds are set using analog voltages on the opposite side of the sense amplifier. The cross-coupled inverter based sense-amplifier is used to perform the thresholding operation to generate output y and its complement.

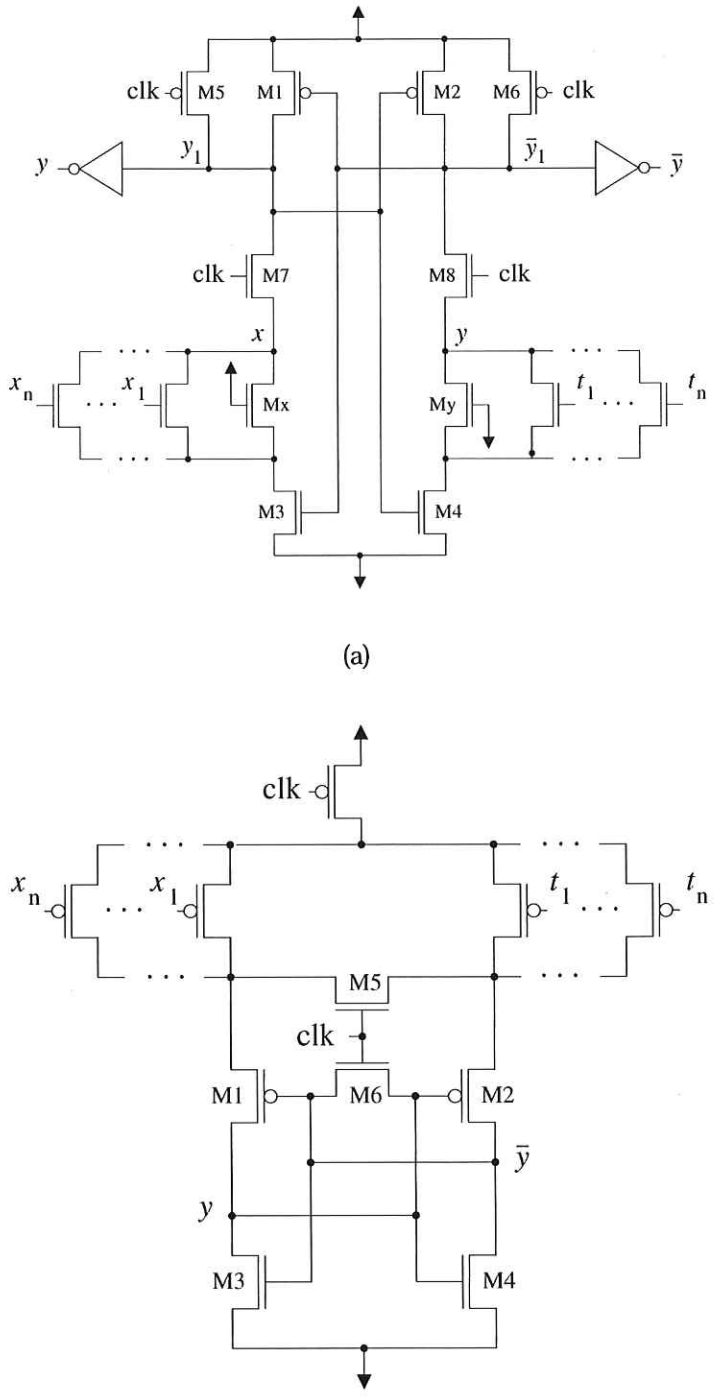


Figure 3.3. Current mode threshold gates including (a) Latched Comparator Threshold Logic (LCTL) and (b) Equalized Current Mode Threshold Logic (ECMTL). The inputs x_i are applied to weights implemented using transistors. The weighted sum of inputs corresponds to the current one one side of the circut. Cross-coupled inverter based current sensing amplifiers are used to perform the thresholding operation to generate output y and its complement.

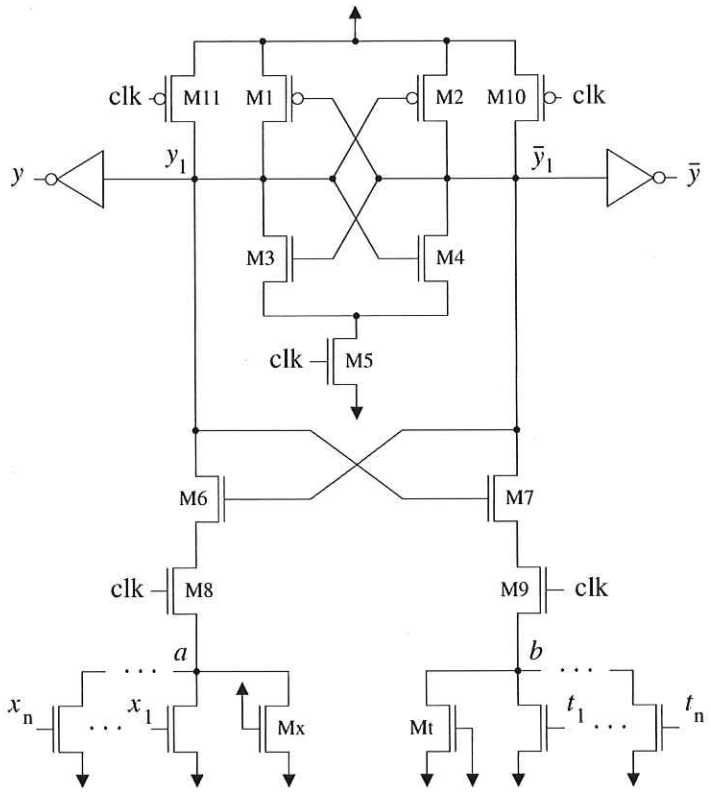


Figure 3.4. Differential Current-Switch Threshold Logic. The inputs x_i are applied to weights implemented using transistors. The weighted sum of inputs corresponds to the current one one side of the circut. The gate threshold is set using a bank of transistors on the opposite side of the sense amplifier. Cross-coupled inverter based current sensing amplifiers are used to perform the thresholding operation to generate output y and its complement.

Table 3.1. Summary of Reported TL Gate Delay Results. The table summarizes the reported delay for a range of TL based circuits implemented in a range of process technologies.

Gate	Function	Process	Delay
νMOS	5-I/P majority (Lashevsky *et al.* 1999)	0.8 μm, 5 V	0.7 ns
CTL	20-I/P OR (Padure *et al.* 1999)	0.5 μm, 5 V	5 ns
	30-I/P majority (Özdemir *et al.* 1996)	1.2 μm	6 ns
L-νMOS	7-I/P no. detector (Kotani *et al.* 1998)	0.6 μm	1 ns
CRTL	4-bit carry (Vassiliadis *et al.* 1996, Celinski *et al.* 2003a)	0.35 μm, 3.3 V	250 ps
LCTL	20-I/P majority (Avedillo *et al.* 1995)	0.7 μm, 3.3 V	3.5 ns
	31-I/P AND (Padure *et al.* 2002b)	1.6 μm, 3.3 V	1.7 ns
ECMTL	8-I/P majority (Bobba and Hajj 2000), AND and OR	0.18 μm, 1.5 V	300 ps
DCSTL	31-I/P AND (Padure *et al.* 2002b)	1.6 μm, 3.3 V	0.7 ns
	4-bit carry (Vassiliadis *et al.* 1996, Padure *et al.* 2002c)	0.25 μm, 2.5 V	290 ps

Table 3.2. Summary of Reported TL Gate Power Dissipation Results. The table summarizes the reported power dissipation for a range of functions implemented using various TL gates implemented in a range of process technologies.

Gate	Function	Process	Power Dissipation
νMOS	5-I/P majority (Lashevsky *et al.* 1999)	0.8 μm, 5 V	280 μW (static)
CTL	20-I/P majority (Celinski *et al.* 2001)	0.25 μm, 2 V	410 μW at 200 MHz
L-νMOS	7-I/P no. detector (Kotani *et al.* 1998)	0.6 μm	400 μW at 100 MHz
CRTL	20-I/P majority (Celinski *et al.* 2001)	0.25 μm, 2 V	200 μW at 200 MHz
LCTL	20-I/P majority (Celinski *et al.* 2001) 20-I/P majority (Avedillo *et al.* 1995)	0.25 μm, 2 V 0.7 μm, 3.3 V	350 μW at 200 MHz 410 μW at 100 MHz
ECMTL	8-I/P majority (Bobba and Hajj 2000)	0.18 μm, 1.5 V	45 μW at 200 MHz
DCSTL	31-I/P AND (Padure *et al.* 2002b)	1.6 μm, 3.3 V	390 μW (unknown freq.)

Table 3.2 summarizes the power dissipation results reported in the literature for single gate circuits for each TL gate discussed in this work. The table shows the circuit function implemented using the given gate, the process technology used and the power dissipation result under the given conditions.

3.1.5 Design Considerations

A number of TL specific circuit design issues related to the mixed signal (multi-valued) nature of the circuits must be considered, where each distinct weighted sum is represented by an analogue current or voltage. The first issue relates to the reliable operation of networks based on the threshold gates. The input offset voltage of the comparator is

Table 3.3. Summary of Reported TL Gate Applications. The table summarizes the reported chip area utilization for a range of TL based circuits implemented in a range of process technologies. Where available, a static CMOS comparison is included, illustrating a significant advantage of TL in either or both area and speed.

Gate	Function	Process	Results	CMOS Comparison
νMOS	5-input majority (Lashevsky *et al.* 1999)	0.8 μm	390 μm^2	60% less area, 17% slower
	Full adder (Hirose and Yasuura 1996)	2 μm	831 μm^2 area (meas.)	45% less area, 30% slower
	(15,4) counter using νMOS sorter (Rodriguez-Villegas *et al.* 2000)	1.2 μm	8 ns delay	28% faster
	8-input Muller-C element (Rodriguez-Villegas *et al.* 2001)	0.8 μm	4930 μm^2, 1.8 ns delay, 30 μW at 100 MHz (meas.)	50% less area, 44% faster, 94% lower power
	(4:2) compressor (Quintana *et al.* 2002)	0.8 μm	1.2 ns delay	61% faster, 20% lower power delay product
	(6:2) compressor (Quintana *et al.* 2002)	0.8 μm	1.6 ns delay	64% faster, 22% lower power delay product
CTL	(31,5) counter (Leblebici *et al.* 1996)	1.2 μm	4.2 ns delay (meas.), 80000 μm^2	approx. 50% less area
	(8×8)-bit multiplier (Leblebici *et al.* 1996)	1.2 μm	70000 μm^2, up to 30 MHz data rate (meas.)	(no comparison provided)

much more significant than the capacitor mismatch (Luck *et al.* 2000) in gates that use a capacitive array is to compute the weighted sum of inputs.

Conversely, the current mode gates are susceptible to the significant input and threshold setting transistor mismatch. In addition, to maintain a symmetrical load on in the comparator circuit, dummy transistors (Padure *et al.* 2002b) may have to be used, which do not perform a computational function and increase the gate area and contribute to increased power dissipation and delay.

Each of the gates must be considered as an analog circuit requiring the use of known layout techniques to match devices and minimize the impact of noise. These include substrate voltage control, shielding, isolation, same-orientation layout of transistors and the use of small parallel transistors to realize larger devices with reduced statistical parameter variations. It is also worth mentioning that various techniques can can be used for designing reliable circuits based on unreliable gates, including coding (Hedge and Shanbhag 1998), hardware redundancy (Muroga 1971) and synthesis techniques that introduce *don't care* conditions.

3.1.6 TL Based System Implementations

To provide a perspective on the attainable performance of digital systems designed using threshold logic, Tables 3.3 and 3.4 provide a collection of results on the design of digital circuits reported in the literature. The tables include reported measured and simulated results and show the comparison to equivalent circuits designed using conventional CMOS logic. Where available, the reported comparison is of the given circuit relative to conventional CMOS logic, either dynamic-CMOS or static-CMOS. For example, the νMOS full adder was shown to occupy 45% less area and was 30% slower than the static-CMOS implementation (Hirose and Yasuura 1996). To the best of the author's knowledge, no ECMTL applications, other than single gates, have been reported. Where the reported results are based on fabricated chip measurements, this is denoted in the table by "(meas.)," all other results are based on circuit or extracted layout simulations. The table shows that TL potentially offers great advantages compared to conventional logic in terms of area, power or delay across a wide range of digital circuit designs.

3.2 Chapter Summary

This Chapter has reviewed a number of of significant, recent developments in Threshold Logic circuit design techniques using CMOS. A number of gate implementations were described in detail and their relative performance evaluated and compared. A survey of notable practical applications in computer arithmetic was given including a comparison of the relative improvement in one or more metrics for the same function when compared to conventional static CMOS logic. In the next Chapter, two new capacitive threshold gate implementations are desribed and a delay model is developed for the systematic evaluation and design for minimal delay of circuits based on these gate implementations.

Table 3.4. Summary of Reported TL Gate Applications (continued).

Gate	Function	Process	Results	CMOS Comparison
CRTL	(7,3) shared-capacitor counter (Celinski *et al.* 2003b)	0.35 μm	460 ps delay, 70 trans., 1850 μm^2	6% area reduction, 45% faster
	(15,4) shared-capacitor counter (Celinski *et al.* 2003b)	0.35 μm	480 ps delay, 140 trans., 3960 μm^2	(no comparison provided)
	4-bit adder (Celinski *et al.* 2002b)	0.25 μm	2280 μm^2, 104 trans., 500 μW at 100 MHz	41% less area, 75% lower power
	64-bit hybrid CRTL/CMOS adder (Celinski *et al.* 2003a)	0.35 μm	670 ps critical path delay, 4325 trans.	30% faster, 30% fewer trans.
LCTL	16-bit adder (Ramos *et al.* 1998b)	1 μm	126000 μm^2, 11 ns delay, 37 μW at 3.3V, 100 MHz, 1328 trans.	(no comparison provided)
DCSTL	(7,3) hybrid DCSTL/CMOS counter (Padure *et al.* 2002a)	0.25 μm	345 ps delay, 237 trans.	53% faster, 67% more trans.
	Mod. DCSTL 16-input embedded TL flip-flop (Padure *et al.* 2003)	0.25 μm	1470 μm^2, 1.0 to 2.1 ns worst case delay (meas.)	(no comparison provided)

Chapter 4

Capacitive Threshold Logic Circuit Techniques

"The efficient mapping of a system onto its implementation medium, be it neuron or silicon, is the essence of the design problem."
— Carver Mead

THIS Chapter proposes two new realizations for CMOS threshold gates, Charge Recycling Threshold Logic (CRTL) and Self-Timed Threshold Logic (STTL). CRTL and STTL have a very low evaluation delay, are suitable for implementing functions with a large number of variables and have a low overall power dissipation. CRTL operates on a single phase clock, while STTL is suitable for self-timed operation. A Logical Effort based delay model is developed for CRTL, and the model is used to evaluate numerous design examples. Fabricated test chip measurement results are used to verify the correct functionality of both CRTL and STTL for a range of gate configurations.

4.1 Charge Recycling Threshold Logic

Both static and dynamic TL gate implementations have been widely reported (Beiu *et al.* 2003). Purely static gates such as neuron-MOS (Shibata and Ohmi 1991) suffer from limited fan-in (Kotani *et al.* 1995), typically less than 12 inputs. Also, some of the existing dynamic gates have relatively high static power dissipation, and some require multiple clock phases (Kotani *et al.* 1995, Özdemir *et al.* 1996, Stokman *et al.* 1998, Huang and Wang 2000), introducing the drawbacks associated with clock signal routing cost, clock skew management and clock power dissipation. Although the dynamic approach proposed in Avedillo *et al.* (1995) dissipates no static power, its dynamic power dissipation is comparable to the total power dissipation of other existing approaches.

Fig. 4.1 shows the proposed circuit structure for implementing a threshold gate with positive or negative weights and threshold. It is based on the charge recycling Asynchronous Sense Differential Logic (ASDL) developed by Kong *et al.* (1999). The ASDL comparator architecture from which the CRTL gate is derived implements high performance, energy efficient operation by recycling charge. The main element is the sense amplifier (cross coupled transistors M1-M4) which generates output Q and its complement Q_b. The inputs x_i are capacitively coupled onto the floating gate ϕ of M5, and the threshold is set by the gate voltage t of M6. The potential ϕ is given by

$$\phi = \sum_{i=1}^{n} C_i x_i / C_{\text{tot}}, \tag{4.1}$$

where C_{tot} is the sum of all capacitances, including parasitics, at the floating node. Weight values are thus realised by setting capacitors C_i to appropriate values. Typically, these capacitors are implemented between the polysilicon 1 and polysilicon 2 layers, commonly available in mixed signal processes, although alternatives, such as trench capacitors available in some processes, may also be used.

The enable signal E controls the precharge and activation of the sense circuit. The gate has two phases of operation, the equalize phase and the evaluate phase. When $\bar{E}$ is high the output voltages are equalized to $V_{\text{dd}}/2$ provided that the capacitive loads at nodes Q and Q_b are equal. When E is high, the outputs are disconnected and the differential circuit (M5-M7, M10, M11) draws unequal currents from the formerly equalized nodes Q and Q_b. The sense amplifier is activated after the delay of the enable inverters and amplifies the difference in potential now present between Q and Q_b, accelerating the transition. In this way the circuit evaluates whether the weighted sum of the

inputs, ϕ, is greater or less than the threshold, T, and a TL gate is realized. The advantage of equalizing the nodes Q and Q_b to $V_{dd}/2$, as opposed to precharging to V_{dd} as in clocked-νMOS, is that evaluation speed is increased since the cross coupled inverters begin evaluation in their transition state. Transistors M10 and M11 turn off the differential circuit after evaluation is completed to reduce the power dissipation. The gate is *charge recycling* because charge which was drawn from the supply rails during evaluation is re-used during equalization, thus reducing the power dissipation during one half of the clock cycle.

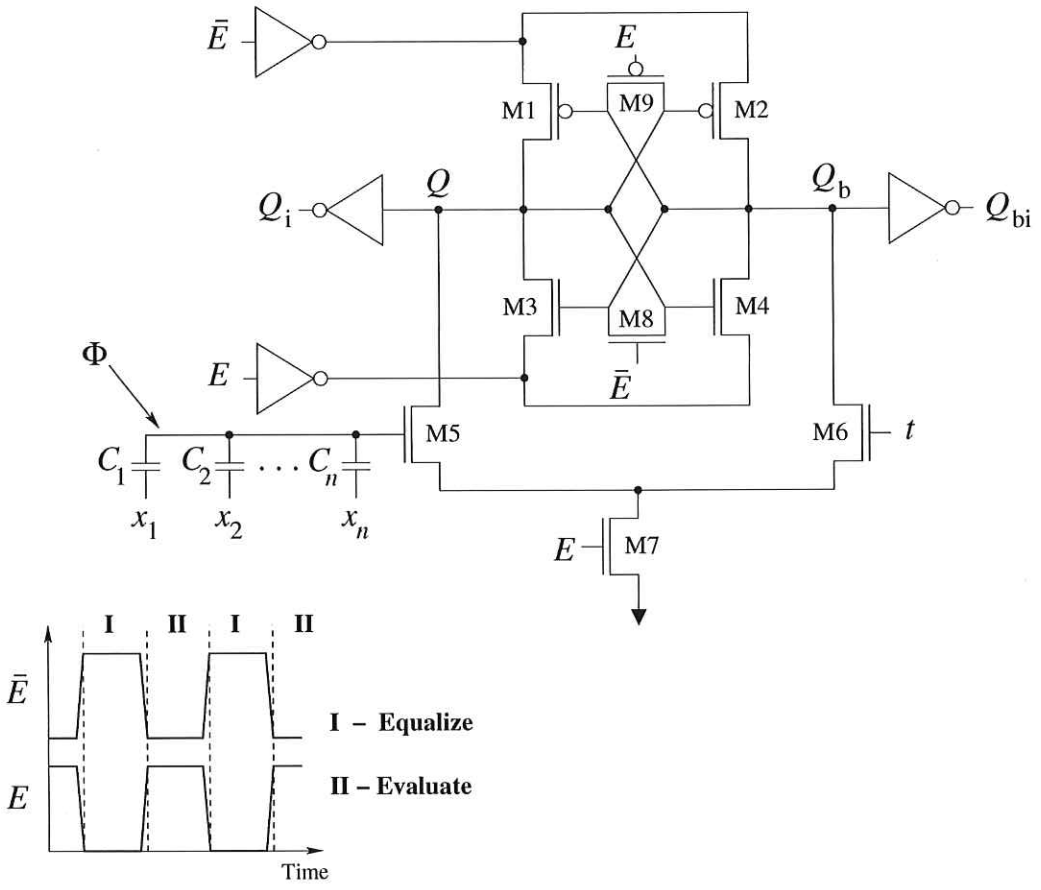


Figure 4.1. Charge Recycling Threshold Logic (CRTL) Gate circuit. The inputs x_i are applied to weights implemented using capacitors C_i and the weighted sum of inputs corresponds to the floating gate voltage ϕ. The gate thresholds are set using an analog voltage t on the opposite side of the sense amplifier. The cross-coupled inverter based sense-amplifier is used to perform the thresholding operation to generate output Q and its complement Q_b.

To ensure reliable operation, the gate layout must be symmetrical to minimize the transistor mismatches and interconnects should be of similar length and width to eliminate

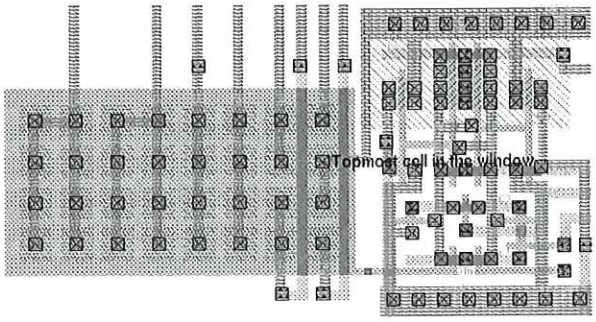


Figure 4.2. Layout of the CRTL circuit of Fig 4.1 in a 2P/4M 0.35 μm process. The bank of input weight setting capacitors is on the left hand side with the inputs applied from above, and the right hand side of the diagram shows the sense amplifier circuit.

interconnect related mismatch. The delay of the enable inverter must be sufficiently large so that the output nodes have sufficient voltage difference at the start of sensing to overcome any offset voltage present in the cross coupled sensing amplifier caused by device mismatch.

To evaluate and compare the performance of the proposed CRTL gate against other CMOS TL gate implementations, a 20-input majority gate ($T = 10$, achieved by setting voltage t=$V_{dd}/2$), was designed in a 0.25 μm process. The 20-input majority function was also implemented using clocked neuron-MOS (Kotani *et al.* 1995), CMOS Capacitor Coupling Logic (C^3L) (Huang and Wang 2000) and the TL gate implementation reported in Avedillo *et al.* (1995) (LPTL). The unit capacitance value used in each implementation was 5 fF. To compare the power dissipation, each of the gates was designed to have similar delay, output rise and fall times, and was loaded by equally sized inverters. All transistors were of minimum length for each implementation and transistor widths were selected to achieve the above timing requirements. All inputs to each gate were switched such that during each evaluation cycle the minimum majority or minority was achieved (11 out of 20 inputs were high or low, respectively), corresponding to the worst-case delay and power dissipation scenario. Also, the power dissipated in the inverters driving the clock and data inputs was included in the total power dissipation measured for each gate. Fig. 4.3 shows the HSPICE power dissipation simulation results for each of the gates versus operating frequency for a 2 V supply. As shown in Figure 4.3, for a typical operating frequency of 200 MHz, CRTL improves power dissipation by between 15% and 30% over the other CMOS threshold gate implementations.

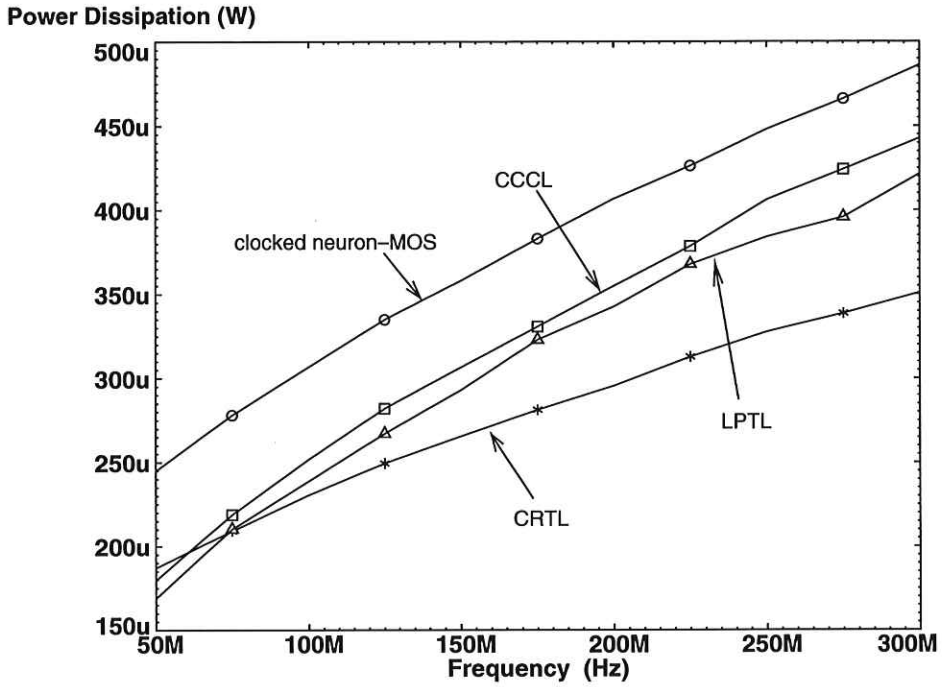


Figure 4.3. Power Dissipation vs. Frequency comparison of the CRTL gate in a 0.35 μm process. Above an operating frequency of 70 MHz, the CRTL gate has noticeably lower power dissipation compared to clocked neuron-MOS, CCCL and LPTL threshold gate implementations. The improvement increases with frequency.

To ensure correct behavior under process and operating point variations, the proposed gate was tested at 45 corners (V_{dd} at 2 V, 2.5 V and 3 V, process Slow-Slow, Slow-Fast, Fast-Slow, Fast-Fast and Typical-Typical, and temperature at -25^{o}C, 75^{o}C and 125^{o}C). Fig. 4.4 shows the transient waveform results from the HSPICE simulation for the 2 V typical-75^{o}C corner at 300 MHz.

4.2 Self Timed Threshold Logic

Fig. 4.5 shows the proposed circuit structure for implementing a self-timed threshold gate variant of the CRTL implementation, which does not use charge recycling. The main element is the cross coupled NMOS transistor pair (M3, M4) that generates the output Q and its complement Q_b after buffering by the two inverters. The gate operates in two phases. Precharge and evaluate are specified by the dual enable signals E and its complement E_b. The inputs x_i are capacitively coupled onto the floating gate ϕ of

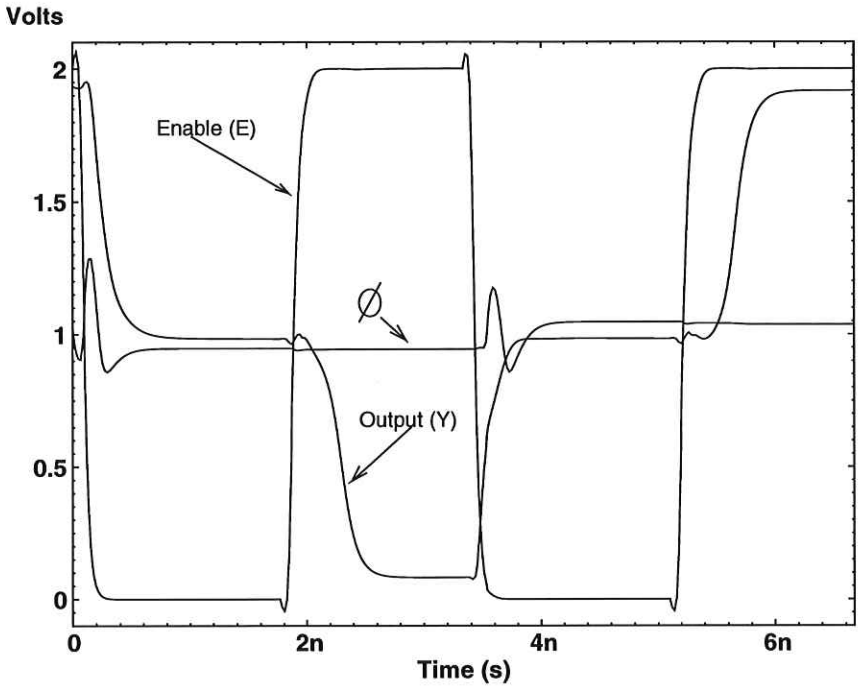


Figure 4.4. Input, Enable and Output analog waveforms.. Simulation results of the CRTL gate in a 0.35 μm process.

M10, and the threshold is set by the gate voltage T of M11. The potential ϕ is given by $\phi = \sum_{i=1}^{n} C_i x_i / C_{\text{tot}}$, where C_{tot} is the sum of all capacitances, including parasitics, at the floating node. Weight values are thus realised by setting capacitors C_i to appropriate values.

The enable signals, E and E_b, control the precharge and activation of the sense circuit. When E is high the voltages at nodes A and B are discharged to ground. When E is low and E_b is high, the outputs are disconnected from ground and the differential circuit, formed by M10 and M11, draws different currents from the supply via M8 and M9. The currents in M8 and M9 are mirrored by M1 and M2 respectively, and the gates of M3 and M4 (nodes A and B) begin to charge at different rates. As the charging rates are different and the capacitances at those two nodes are the same (ensured by identical sizing of the two buffer inverters), a voltage difference begins to develop between nodes A and B. When this difference is sufficiently large, either M3 or M4 turns on, but not both. The outputs Q and Q_b are evaluated and passed to the next stage. In this way, the circuit structure effectively determines if the weighted sum of the inputs, ϕ, is greater or less than the threshold, T, thus realizing a thresholding

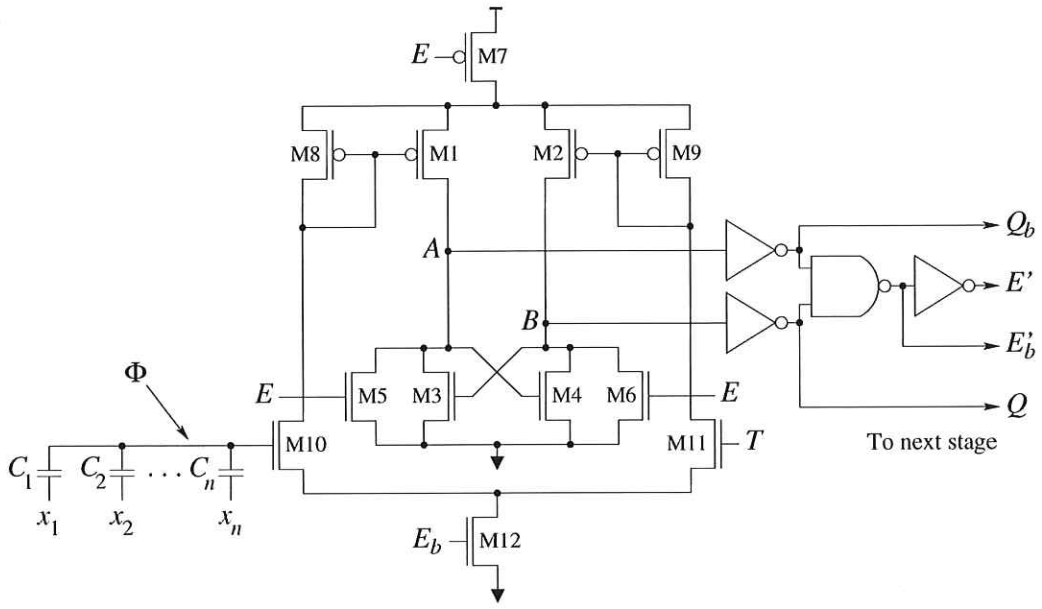


Figure 4.5. The Self-Timed Threshold Logic gate structure circuit. The inputs x_i are applied to weights implemented using capacitors C_i and the weighted sum of inputs corresponds to the floating gate voltage ϕ. The gate threshold is set using analog voltage T on the opposite side of the sense amplifier. The cross-coupled inverter based sense-amplifier is used to perform the thresholding operation to generate output Q and its complement Q_b. The circuit includes logic for the self-timed equalization and evaluation signals for the next stage.

operation. The two buffer inverters serve to provide a balanced capacitive load for nodes A and B and also to drive the inputs of the next stage.

The next STTL gate is held in precharge until the previous gate evaluates its output. The enable signals for the next stage are generated by the NAND gate output E'_b and its inverse E'. During the precharge phase of the first stage, the enable signals for the next stage are E'_b=0 and E'=1, hence the second stage and all subsequent stages are also in the precharge phase, and only begins to evaluate after the outputs of the first stage (Q and Q_b) are established. Correct timing is ensured by setting the combined delay of the two buffer inverters and the NAND gate to be larger than the evaluation delay of the first gate. Thus, outputs of each gate propagate through the chain in a self-timed fashion. An additional benefit of the above scheme is that spurious transitions and hence power dissipation associated with glitching are removed.

4.3 Delay Modeling

In this section, a delay model for CRTL is developed based on the recently proposed theory of Logical Effort . The delay model enables the prediction of the delay of CRTL based circuits and a systematic comparison of CRTL designs with conventional logic. Another motivator for developing the model is the desire to avoid the common and largely unsatisfactory presentation of circuit performance results commonly found in the literature in the form of delay numbers with insufficient information to allow comparison across different process technologies and loading conditions. The model is applied to the design of a number of common circuits, including 4-bit carry generate blocks, AND trees, and a (7,3) counter critical path.

4.3.1 Logical Effort

Logical effort (LE) is a design methodology for estimating the delay of CMOS logic circuits, implementing a given logic function (Sutherland and Sproull 1991, Sutherland *et al.* 1999). It provides a means to determine the best number of logic stages, including buffers, required to implement a given logic function, and to size the transistors to minimize the delay.

Logical effort is based on a reformulation of the conventional RC model of CMOS gate delay that separates the effects on delay of gate size, topology, parasitics and load. The relative simplicity of the method compared to other delay modeling techniques and sufficient accuracy allow it to be used early in the design process to evaluate alternative circuits.

The total delay of a gate, d, is comprised of two parts, an intrinsic parasitic delay p, and an effort delay, f, driving the capacitive load. The parasitic delay is largely independent of the transistor sizes in the gate, since wider transistors which provide increased current have correspondingly larger diffusion capacitances. The effort delay in turn depends on two factors, the ratio of the sizes of the transistors in the gate to the load capacitance and the complexity of the gate. The former term is called *electrical effort*, h, and the latter is called *logical effort*, g.

Electrical effort is defined as

$$h = \frac{C_{\text{out}}}{C_{\text{in}}}, \tag{4.2}$$

where C_{out} and C_{in} are the gate load capacitance and input capacitance, respectively. The logical effort, g, characterizes the gate complexity, and is defined as the ratio of the

input capacitance of the gate to the input capacitance of an inverter that can produce equal output current. Alternatively, the logical effort describes how much larger than an inverter the transistors in the gate must be to be able to drive loads equally well as the inverter. By definition an inverter has a logical effort of 1.

The delay of a single logic gate can be expressed as

$$d = gh + p. \tag{4.3}$$

This delay is in units of τ, which is, the delay of an inverter driving an identical copy of itself, without parasitics. This normalization enables the comparison of delay across different technologies. The product gh is called the gate or stage effort.

The considerations so far apply to single gates, but may be extended to the treatment of delay through a path . Using uppercase to denote path parameters, the path electrical effort, H, is similarly defined as the ratio of the path load capacitance to the path input capacitance. The path logical effort, G, is given by

$$G = \prod g_i, \tag{4.4}$$

where the subscript i indexes the logic gates along the path. The effect of fan-out, which causes some of the available drive current to be directed off the path being analyzed, is accounted for by considering the branching effort, b, which is defined as

$$b = \frac{C_{\text{on-path}} + C_{\text{off-path}}}{C_{\text{on-path}}}. \tag{4.5}$$

and the path branching effort is given by

$$B = \prod b_i. \tag{4.6}$$

Finally, the path effort, F, is given by the product of the path logical effort, the path branching effort and the path electrical effort

$$F = GBH. \tag{4.7}$$

The path delay, D, is the sum of the delays of each of the gate stages in the path, d_i, and consists of the *path effort delay*, D_F, and the *path parasitic delay*, P,

$$\begin{aligned} D &= \sum d_i \\ &= D_F + P \\ &= \sum g_i h_i + \sum p_i. \end{aligned} \tag{4.8}$$

It can be shown that the path delay is minimized when each stage in the path bears the same stage effort and the minimum delay is achieved when the stage effort is

$$f_{\min} = g_i h_i = F^{1/N}. \tag{4.9}$$

This leads to the main result of logical effort, which is the expression for minimum path delay

$$D_{\min} = NF^{1/N} + P. \tag{4.10}$$

To equalize the effort borne by each stage in the path, the transistor sizes in each logic gate must be chosen according to the electrical effort given by Equation (4.9)

$$h_{i,\min} = \frac{F^{1/N}}{g_i}. \tag{4.11}$$

This allows us to calculate the input capacitance and hence transistor size (width, assuming minimum length transistors) by applying the transformation

$$C_{\text{in},i} = \frac{g_i C_{\text{out},i}}{f_{\min}}. \tag{4.12}$$

This input capacitance is distributed among the transistors within the gate connected to the input.

The preceding steps dictate how to size the gates along a path for minimum delay, taking into account the differing complexity of the gates as given by the logical effort. Equally important is the selection of the correct number of stages. It has been shown (Sutherland *et al.* 1999) that for static CMOS logic, the near optimal stage effort is approximately 4, and stage efforts from 2.4 to 6 give delays within 15% of the minumum. Hence the best number of stages is approximately

$$N \approx \log_4 \mathrm{F}. \tag{4.13}$$

For domino logic the optimal stage effort is 2 to 2.75 (Sutherland *et al.* 1999).

To minimize delay, the design should use the correct number of stages of logic and gates with low logical effort and parasitic delay. Path design may involve iteration, because the path's logical effort depends on the topology of individual gates, but the best number of stages is not known without knowing the path effort.

The simulated values of logical effort for a range of fanin NAND and NOR gates in a 0.18 μm, 1.8 V CMOS technology were shown to be significantly different from the theoretical value (Yu *et al.* 2001). In the same work it was also shown that the delay

value predicted by Equation (4.3) differed from simulation results on average by over 20% for the same range of gates, mainly as a result of the impact on delay time of the input transition times. However, the accuracy of the delay predicted by Equation (4.3) can be improved by calibrating the model by simulating the delay as a function of load (electrical effort) and fitting a straight line to extract τ, the inverter parasitic delay, p_{inv}, and the logical effort, g. This technique will be used to develop a calibrated logical effort based model for the delay of the CRTL gates.

4.3.2 CRTL Delay Model

This section begins by providing a set of assumptions which will simplify the analysis, a proposed expression for the worst case delay of the CRTL gate and a derivation of the model's parameters. The model is then applied to two practical circuit examples. The method described below may similarly be applied to other sense amplifier based linear threshold gates.

Notation and Assumptions

The TL gate is assumed to have n logic inputs (fan-in), the total number of gate inputs connected to logic one is denoted by N, and T is the threshold of the gate. The potential of the gate of transistor M6, t, in Fig. 4.1 is given by

$$t = \frac{T}{n} \times V_{\text{dd}}. \tag{4.14}$$

In the worst case, the voltage ϕ in Equation (4.1) takes the values

$$\phi = t \pm \frac{\delta}{2} \tag{4.15}$$

where δ is given by

$$\delta = \frac{V_{\text{dd}}}{n}. \tag{4.16}$$

Equation (4.15) expresses the worst case (greatest delay) condition where the difference between ϕ and t is minimal, ie. the step voltage generated by the sum of inputs with respect to the threshold voltage is smallest. The value of $\phi = t - \delta/2$ corresponds to the rising and falling edges of the nodes Q and Q_b, respectively, in Fig. 4.1, and conversely for $\phi = t + \delta/2$.

The gate inputs are assumed to have unit weights, ie. $w_i = 1$, since the delay depends only on the value of N and T. Also, without loss of generality, it will be assumed that

the weights and threshold are positive, since negative weights may easily be accommodated in the differential structure of the gate by using a network of input capacitors connected to the gate of M6.

Since the gate is clocked, delay will be measured from the clock E to Q_i-Q_{bi}. Specifically, delay will be measured as the average of the 50% point on two falling transitions of E to the 50% points on the corresponding falling and rising edges of Q_i and Q_{bi}. Generally, the delay will depend on the threshold voltage, t, the step size, δ, and the capacitive output load on Q_i and Q_{bi}. To simplify the analysis, the value of t will be fixed at 1.5 V. This value is close to the required gate threshold voltage in typical circuit applications. Therefore the worst case delay depends only on the fan-in and gate loading, and allows us to propose a model based on expressions similar to those for conventional logic based on the theory of logical effort.

Formulation of the Model and Parameter Extraction

The delay of the CRTL gate may be expressed as Equation (4.17). This delay is the total delay of the sense amplifier and the buffer inverters connected to Q and Q_b, and depends on the load, h, and the fanin, n, as follows

$$d_{E\rightarrow Q_i} = \{g(n)h + p(n)\}\tau. \tag{4.17}$$

The load, h, is defined as the ratio of load capacitance on Q_i (assuming the loads on Q_i and Q_{bi} are equal) and the CRTL gate unit weight capacitance. Both logical effort and parasitic delay in Equation (4.17) are a function of the fanin.

To determine the values of the parameters in Equation (4.17), the value of the parasitic delay of an inverter, p_{inv}, is first determined. From Equation (4.3), the inverter delay is $d = \tau(gh + p_{\text{inv}})$, where by definition $g = 1$ for an inverter. To obtain the values of τ and p_{inv}, it is possible to measure from HSPICE simulations the inverter delay for various values of electrical effort h, and plot the delay versus h. The slope of this straight line gives the value of τ and the $h = 0$ axis intercept gives τp_{inv}.

The delay parameters for the industrial 0.35 μm process used to obtain the simulation results presented here and the simulated FO4 (fan-out of four) inverter delay are given in Table 4.1. The value of τ is found to be 40 ps.

The values of g and p in Equation 4.17 were extracted by linear regression from simulation results for a range of fanin from $n = 2$ to 60 while the electrical effort was swept

Table 4.1. Delay parameters of the 0.35 μm, 3.3 V, 4M/2P process at 75°C. This includes the intrinsic delay parameter τ, inverter parasitic delay p_{inv} and the FO4 (fan-out of four) inverter delay.

τ	p_{inv}	FO4 delay
40 ps	1.18	204 ps

Table 4.2. Extracted CRTL gate logical effort, g, parasitic delay, p, parameters. For n = 2 to 60 for the 0.35 μm, 3.3 V, 4M/2P process at 75°C and the gate delay normalized to FO4 for electrical effort h = 1, 5 and 10.

n	g	p	$d_{E\rightarrow Qi}$, h=1	$d_{E\rightarrow Qi}$, h=5	$d_{E\rightarrow Qi}$, h=10
2	0.346	2.5	0.55	0.82	1.15
5	0.357	3.3	0.71	0.98	1.33
10	0.365	4.0	0.84	1.13	1.48
15	0.376	4.3	0.90	1.19	1.56
20	0.375	4.7	0.98	1.27	1.63
30	0.400	5.0	1.04	1.35	1.74
40	0.424	5.1	1.07	1.40	1.80
50	0.439	5.2	1.09	1.43	1.85
60	0.460	5.2	1.09	1.45	1.90

from h = 0 to 20 as shown in Table 4.2. The Table also gives the absolute gate delay for three values of electrical effort, h = 1, 5 and 10, where h is the ratio of the load capacitance to the unit input capacitance of 3.37 fF.

By fitting a curve to the parameters g and p, CRTL gate delay may be approximated in closed form by

$$d_{E\rightarrow Qi} = \{(0.002n + 0.34)h + \ln(n) + 1.6\}\tau. \tag{4.18}$$

It should be noted that the FO4 delay predicted by the LE model, $d = \tau(gh + P_{\text{inv}}) = 40(4 + 1.18) = 207$ ps, agrees well with the simulated result of 204 ps. Additionally, the FO4 delay value is approximately 20% higher than that reported in the literature for a typical 0.35 μm process (see for example (Horowitz *et al.* 1998)), most likely due to the lower temperature used to obtain those results. For comparison, the FO4 delay across various process technologies may be closely approximated by 500 ps/micron (gate length) (Horowitz *et al.* 1998).

In order to use the parameters in Table 4.2 and Equation (4.18), it is necessary to compensate for the parasitic capacitance at the floating gate of M5. From Equation(4.1), the parasitic capacitance, C_p, contributes to a reduced voltage step, δ, on the gate of M5 in Fig. 4.1 with respect to the threshold voltage, t, as given by Equation (4.19),

$$\delta_{\text{eff}} = \left\{ \frac{\sum_{i=1}^{n} C_i}{\sum_{i=1}^{n} C_i + C_p} \right\} \delta_0, \tag{4.19}$$

where δ_0 is the nominal step given by Equation (4.16). This reduction in δ is equivalent to an increased value for the fanin. This effective fanin, n_{eff}, is given by

$$n_{\text{eff}} = \left\{ \frac{\sum_{i=1}^{n} C_i + C_p}{\sum_{i=1}^{n} C_i} \right\} n_0, \tag{4.20}$$

where n_0 is the number of inputs to the gate and n_{eff} is the value used to calculate the delay. Typically, for a large fanin CRTL gate, by far the major contribution to the parasitic capacitance will be from the bottom plate of the floating capacitors used to implement the weights. In the process used in this work, this corresponds to the poly1 plate capacitance to the underlying n-well used to reduce substrate noise coupling to the floating node.

For example, for a 32 input CRTL gate with 3.37 fF poly1-poly2 unit capacitors (4 μm^2), the parasitic capacitance of poly1 to substrate is 29 fF, and the $\sum_{i=1}^{n} C_i = 32 \times 3.37 =$ 108 fF. From Equation (4.20) the effective fanin to be used in the delay calculation is ((108+29)/108) $\times$ 32 $\approx$ 41.

4.3.3 Applying the Model—Design Comparison Examples

In order to illustrate the application of the model presented in the previous Section, the delay of wide AND gates used in ALUs, the 4-bit carry generate function used in adders, and the carry-out of a (7,3) parallel counter, designed using both domino and CRTL are evaluated and compared. In all examples the transistors of the domino circuits (first stage only for multi stage circuits) are sized to present the same input capacitance as the minimum sized inverter (1.8 μm of gate width, minimum length), which is approximately equal to the CRTL unit weight input capacitance, to ensure all designs affect the delay of the driver equally.

4-bit Carry Generate

The carry generate signal, α, of a 4-bit block may be calculated using a single TL gate as follows (Vassiliadis *et al.* 1996) (a_i and b_i are the input bits at the i^{th} bit significance)

$$\alpha = \mathrm{sgn}\left\{\sum_{i=0}^{3} 2^i(a_i + b_i) - 2^4\right\}. \tag{4.21}$$

A realistic load corresponding to $h = 10$ (ie. C_L = 33.7 fF) will be assumed. The sum of weights $N = 30$, so the worst case delay of this gate will correspond to the delay of a gate of effective fanin, n_{eff}, of approximately 40. From Table 4.2, the expected delay is 1.8 FO4, or 372 ps. Using Equation (4.18), the calculated delay is 379 ps.

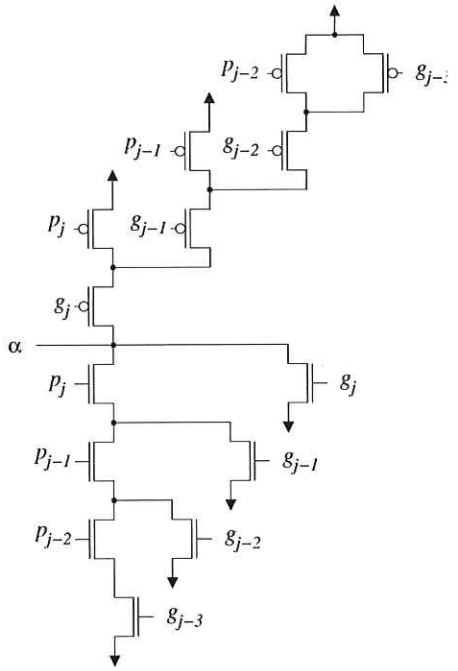


Figure 4.6. Static CMOS 4-bit carry generate circuit. The bit-wise carry propagate signals are denoted using p_i and the carry generate signals are denoted using g_i. The gate output is denoted by α.

The static CMOS gate used to compute the same function is shown in Fig. 4.6 (Beaumont-Smith and Lim 2001). To obtain a fair comparison, the transistors of the static CMOS gate were sized so that the CMOS gate input capacitance is equal to the input weight unit capacitance of the CRTL gate, corresponding to $w_0 = 1$, or C_{in} = 3.37 fF and the same load was used. The simulated slowest, g_{j-3}, input delay delay is 1.07 ns (5.3 FO4) and the fastest, g_j, input delay is 634 ps (3.1 FO4). The gate is then sized so that the

input capacitance is equal the the largest input capacitance of the CRTL gate, corresponding to $w_3 = 8$, which is $8 \times 3.37 = 27$ fF. In this case the simulated slowest and fastest input delays are 514 ps (2.5 FO4) and 197 ps (0.96 FO4) respectively.

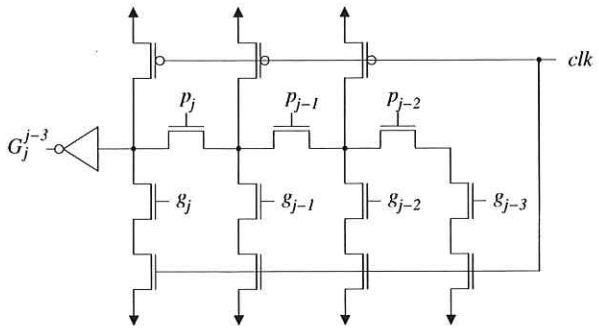


Figure 4.7. Dynamic-CMOS 4-bit carry generate, domino circuit. The bit-wise carry propagate signals are denoted using p_i and the carry generate signals are denoted using g_i. The gate output is denoted using G_j^{j-3}.

The domino gate used to compute the same function for comparison is the well known Manchester-carry circuit, shown in Fig. 4.7. The electrical effort and parasitic delay for the slowest input, g_{j-3}, were extracted from simulation (Sutherland *et al.* 1999) and used to calculate the worst case delay for $h = 1$ and $h = 10$. The results are summarized in Table 4.5. It should be noted that the domino gate delay numbers exclude the delay of generating the bitwise p_i and g_i signals.

Under the conditions of equal input capacitance and load, the CRTL gate is 1.3 to 1.6 times faster. This is a significant delay improvement even in this case of a function with a small number of logic inputs.

Wide AND

As a second example, the design of wide AND gates, used for example in ALUs for zero detection, is considered. The minimum delay domino trees for 8 to 64 inputs are listed in Table 4.4, e.g. 4, 4, 2 denotes a 3-layer tree design of the 32-input AND consisting of four 4-input AND gates in the input layer, four 4-input gates in the second layer and a 2-input gate in the third layer. These minimum delay trees were obtained by extracting the logical effort and parasitic delay of 2-, 4- and 8-input domino AND gates from simulations (see Chap. 5 of Sutherland *et al.* (1999) for details), and finding the tree which minimizes the sum of effort and parasitic delay. The calibrated electrical effort and parasitic delay were used in the domino delay calculation.

Table 4.3. Minimum-delay domino-CMOS AND tree designs. Fanin $n = 8$, 16, 32 and FO4 delay comparison with CRTL for path electrical effort $H = 1$ and 10. The CRTL implementation has significantly lower delay for both values of path electrical effort.

n	Domino	$H = 1$		$H = 10$	
	tree	Domino	CRTL	Domino	CRTL
8	4, 2	1.91	0.84	2.34	1.48
16	4, 4	2.33	0.98	2.8	1.63
32	4, 4, 2	3.22	1.07	3.47	1.80
64	4, 4, 4	3.5	1.1	3.93	1.95
Average		2.74	1.0	3.14	1.72

Table 4.4. Static CMOS AND tree designs. FO4 delays for minimum delay for $n = 8$, 16, 32 and 64 for path electrical effort $H = 1$ and $H = 5$.

n	Tree	FO4 delay H = 1	FO4 delay H = 5
8	4, 2	1.94	2.84
16	4, 4	2.58	3.38
32	4, 2, 2, 2	3.32	3.98
64	4, 2, 4, 2	3.86	4.54

Table 4.4 shows the FO4 delay for domino and CRTL AND gates with fanin from 8 to 64, for path electrical effort $H = 1$ and $H = 10$, corresponding to the values of load $h = 1$ and $h = 10$ in Table 4.2. Note that increased vales of n_{eff} are used to obtain the correct CRTL delay values from Table 4.2.

Comparing Tables 4.2 and 4.4, the CRTL gate design is on average approximately 1.8 to 2.7 times faster than domino-CMOS for a path electrical effort of 10 and 1, respectively.

Table 4.4 shows the FO4 delay for static CMOS AND trees with fanin from 8 to 64, for $H = 1$ and $H = 5$ (Sutherland *et al.* 1999), corresponding to the values of load $h = 1$ and $h = 5$ in Table 4.2.

Comparing Tables 4.2 and 4.4, the CRTL gate design is on average 3 times faster and 2.8 times faster for $H = 1$ and $H = 5$, respectively.

Table 4.5. 4-bit carry generate, G_j^{j-3}, and (7,3) counter c_{out} FO4 delay comparison with CRTL for path electrical effort $H = 1$ and 10. The CRTL implementation has significantly lower delay for both values of path electrical effort.

	$H = 1$		$H = 10$	
Function	Domino	CRTL	Domino	CRTL
G_j^{j-3}	1.7	1.07	2.42	1.8
(7,3) c_{out}	1.5	0.84	1.9	1.48

(7,3) Counter Critical Path

As the final design example, consider the critical path of a (7,3) parallel counter, commonly used in multipliers. The domino critical path for c_{out} consists of two full adders. The CRTL implementation computes the majority function using a single gate, where the output is logic 1 if 4 or more inputs are 1. The delay results are shown in Table 4.5. Under the given loading conditions, CRTL implementation is between 1.3 and 1.8 times faster.

4.4 Test Chip Results

A test chip was implemnted using an industrial double-poly, 4-metal process. The chip circuit schematic is shown in Fig. 4.8. The main focus was firstly on proof-of-concept of the functionality of both CRTL and STTL gates. Secondly, the fan-in cabability of was investigated, and finally delay measurements to verify the validity of the model presented in Section 4.3.2 were obtained. Both types of TL gate designs were agressively sized for near optimal delay, as would be required in a performance driven application, like a high-speed adder design. As will be shown below, the measurements verify correct functionality.

The micrograph of the fabricated test chip is shown in Fig. 4.9. Due to the limited area and number of pads available on the test chip[1] there are seven gates on the chip, four CRTL gates and three STTL gates, with fan-in (sum-of-weight) ranging from 8 to 64. The chip consists of a serial-to-parallel shift register (six chained flip-flops) that provide the data inputs to each of the TL gates and the select signals to the demultiplexer

[1]Test chip was kindly fabricated by the Microelectronic Circuits and Analogue Devices Research Group, Department of Engineering Science, University of Oxford.

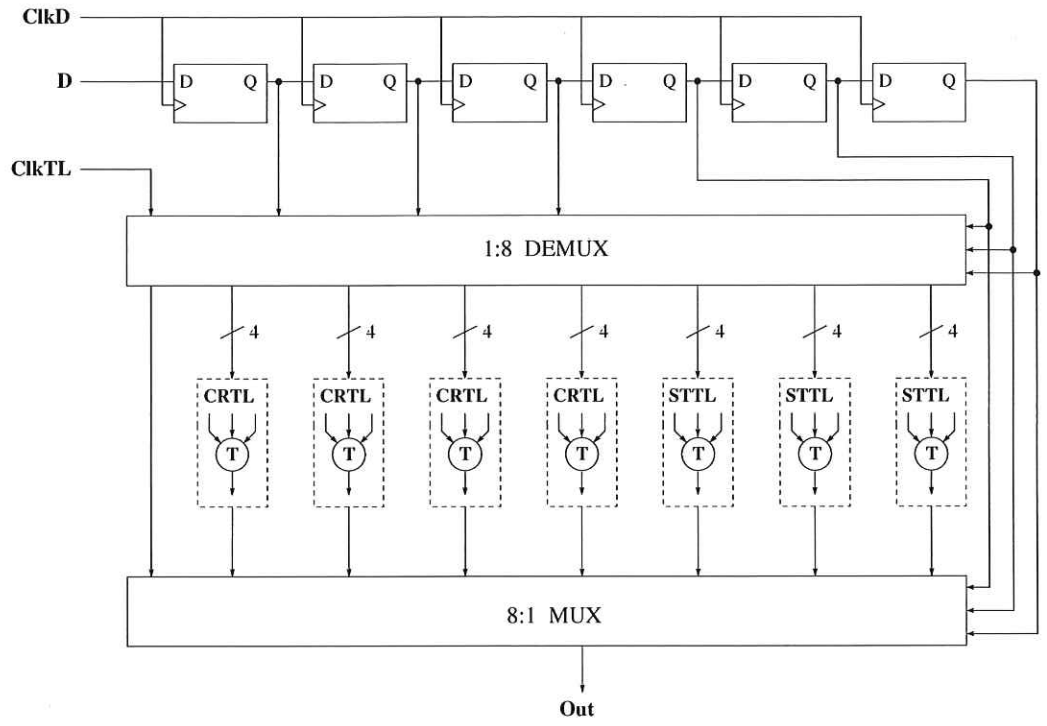


Figure 4.8. Schematic of test chip for experimental functionality verification and delay measurements. From top to bottom, the diagram shows the input shift register, 1:8 demultiplexer, bank of CRTL and STTL gates and 8:1 multiplexer.

and multiplexer. Through the 1:8 demultiplexer and 8:1 multiplexer, the shift register outputs also control the signal path of the input data and TL gate clock to one of the seven gates at a time, and the output of that gate, buffered, to the output pad of the chip. In this way, the functionality of each gate may be observed externally and in isolation. Also, by providing one separate path for the TL clock from its input pad to the output which is identical to its usual path (where it is used to clock the CRTL and STTL gates), except that it does not pass through a TL gate, the gate delay ($d_{E \rightarrow Qi}$) may be measured.

Table 4.6 describes the implemented CRTL and STTL gates. Each gate was designed with three inputs, with the weights for each input given in the table. The total sum of weights for the gates is 16 (CRTL16), 32 (CRTL32), 48 (CRTL48), 8 (STTL8), 32 (STTL32) and 64 (STTL64) and this reflects the fan-in corresponding to inputs with unit weights. The threshold voltage is externally adjustable through an analogue pad connection, Adjusting the analogue threshold voltage from Gnd to V_{dd} allows the gate threshold T to be set anywhere in the range from 0 up to n, where n is the sum of weights. In the case of the CRTL(32,32) gate, a network of capacitors is used to adjust the threshold, as

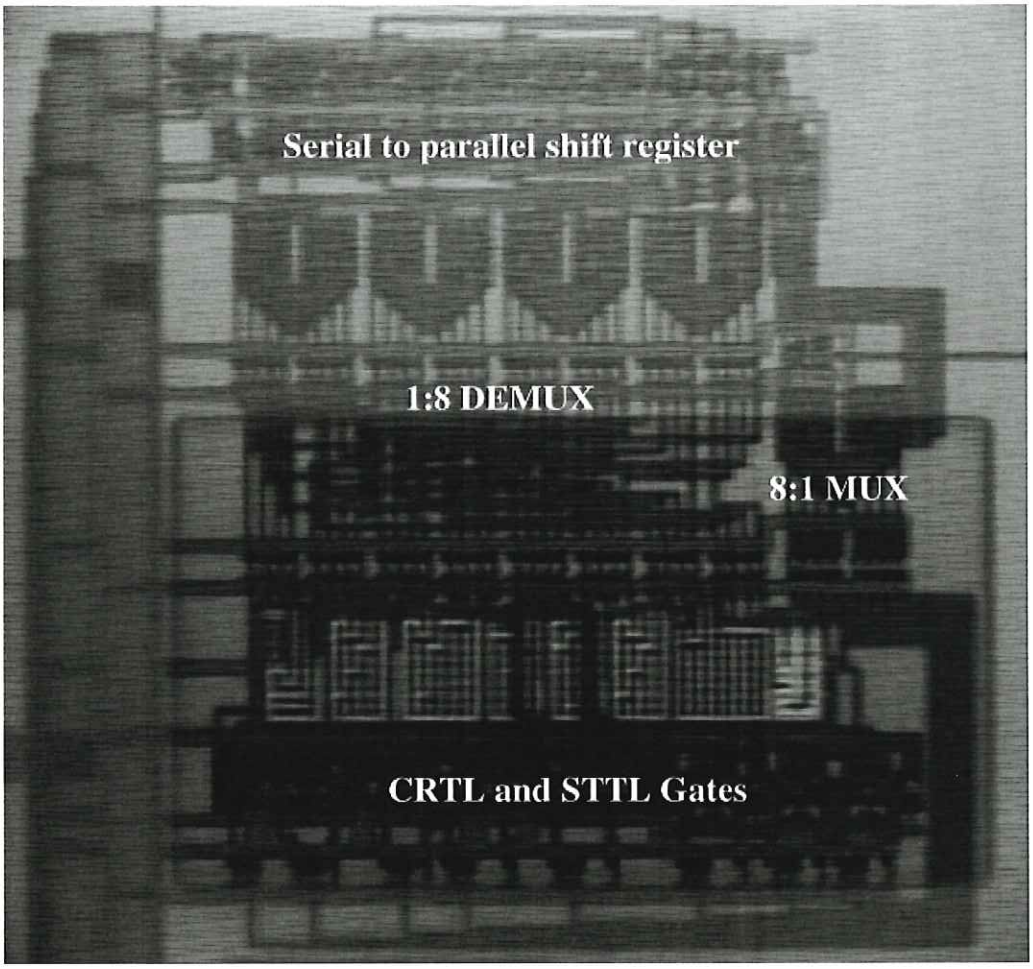


Figure 4.9. Micrograph of fabricated test chip for experimental functionality and delay measurements in a 3.3 V, 0.35 μm AMS CMOS process. The diagram shows the input shift register, 1:8 demultiplexer, bank of CRTL and STTL gates and 8:1 multiplexer corresponding to the functional blocks in Figure 4.8.

described in Section 4.1. In this gate, the three gate inputs are connected to the three threshold setting capacitors as well as the three input weight capacitors.

This scheme allows testing of a wide range of sum-of-input-weights and threshold combinations. For example, from Table 4.6, the weight values of [1,5,10] for the CRTL16 gate allow for the input sum-of-weights to take values of 0, 1, 5, 6, 10, 11, 15 and 16. For the CRTL(32,32) gate, the possible sum-of-inputs and threshold value pairs are (0,0), (2,3), (12,14), (14,17), (18,15), (20,18), (30, 29) and (32,32).

It is important to note that the CRTL and STTL gates are relatively compact when compared to other differential implementations. For example, compared to the DCSTL gate areas reported in (Padure *et al.* 2003), a 32-input CRTL gate, including the capacitor

network to implement the weights, occupies less than 30% of the area of the 32-input DCSTL gate—implemented in a smaller feature size technology of 0.25 μm, although not optimized for delay. As discussed previously, this is largely because DCSTL and other non-capacitive TL gates require dummy transistors, which add a significant area overhead. Table 4.6 summarizes the area results of the implemented CRTL and STTL gates. The layouts and corresponding chip micrographs of a CRTL 16-input gate and an STTL 8-input gate are shown in Fig. 4.10.

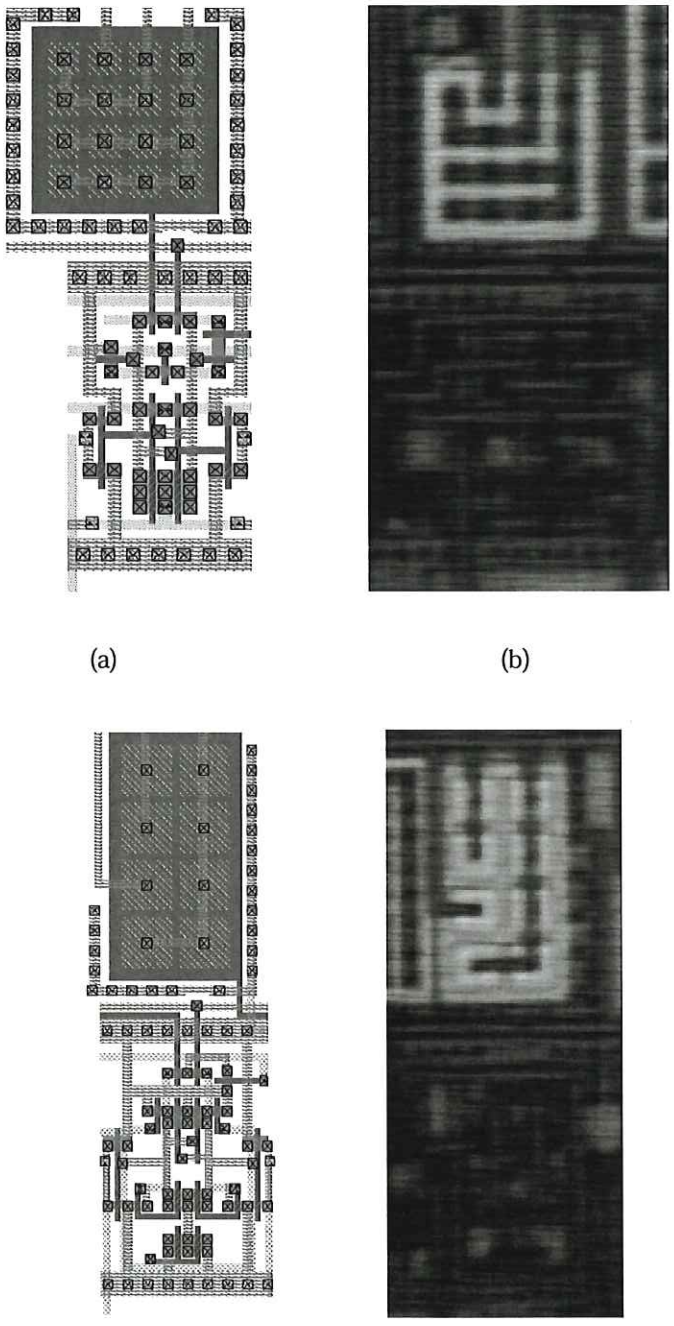


Figure 4.10. TL gates layouts and micrographs. (a) 16-input CRTL layout, (b) 16-input CRTL micrograph, (c) 8-input STTL layout and (d) 8-input STTL micrograph.

Table 4.6. CRTL and STTL gates implemented on the test chip, the weight values, threshold range and occupied chip area.

Gate	Weights	Threshold	Area (μm^2)
CRTL16	{1,5,10}	{0 → 16}	576
CRTL32	{1,11,20}	{0 → 32}	608
CRTL48	{1,15,32}	{0 → 48}	734
CRTL(32,32)	{2,12,18}	{3,14,15}	918
STTL8	{1,2,5}	{0 → 8}	600
STTL32	{1,11,20}	{0 → 32}	688
STTL64	{1,23,40}	{0 → 64}	800

Table 4.7. Residual floating gate voltage (ϕ_{res}), measured threshold (all data inputs x_i set to 0), calculated n_{eff} and measured n_{eff}.

Gate	ϕ_{res} V	t_{meas} V	$\frac{\sum_{i=1}^{n} C_i + C_{\text{p}}}{\sum_{i=1}^{n} C_i}$	Calculated n_{eff}	Measured n_{eff}
CRTL16	1.86	2.01	1.3	21	22
CRTL32	1.75	1.83	1.28	41	41
CRTL48	1.58	1.52	1.26	60	55

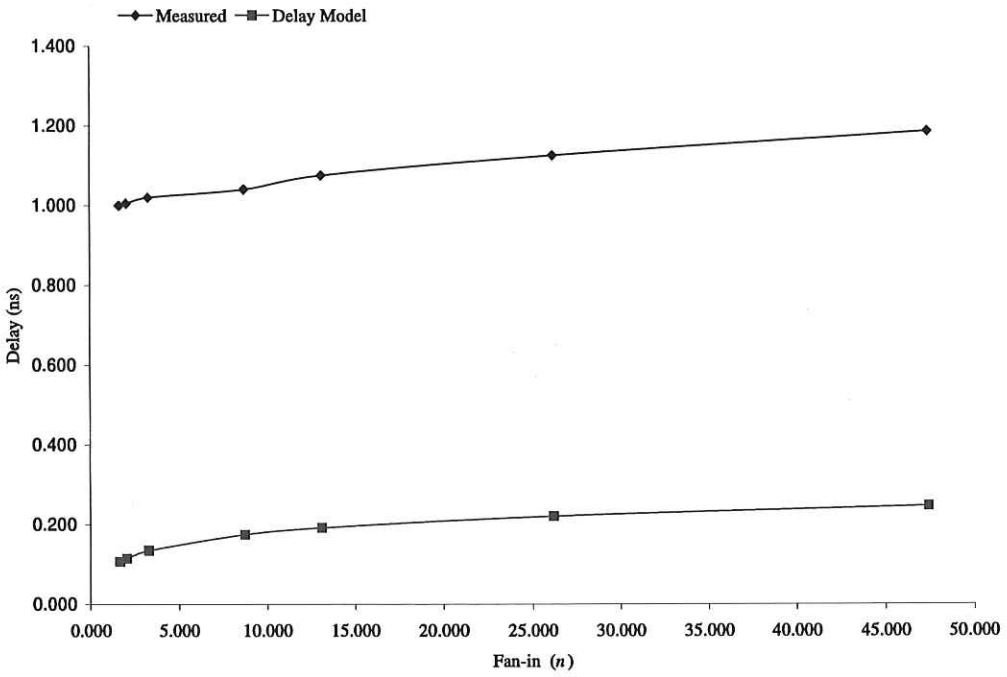


Figure 4.11. Plot of measured and predicted CRTL gate delay, $d_{E \to Qi}$, vs. fan-in, n.

Table 4.7 summarizes the measured residual floating gate voltage for each gate. The table also lists the measured and calculated values of the effective fan-in, n_{eff} as predicted by Equation (4.20), based on values of capacitance extracted from the layout. As can be seen from the table, the calculated and measured values of n_{eff} are in close agreement across a wide range of fan-in.

Figure 4.11 shows the measured and predicted delay results. The predicted (calculated) delay values are based on Equation (4.18). The two plotted lines are for two different processes (which have different values of τ) and hence can not be compared directly. However, what can be clearly seen from Figure 4.11 is that the form of the derived model of Equation (4.18) is in close agreement with measurement. It can clearly be seen that, as predicted, the delay has a very weak dependence on fan-in, it increases very slowly with n.

4.5 Chapter Summary

This Chapter proposes two new realizations for CMOS threshold gates, Charge Recycling Threshold Logic (CRTL) and Self-Timed Threshold Logic (STTL). A delay model is developed for CRTL and test chip measurement results are used to verify the correct functionality of both CRTL and STTL. The next Chapter builds on these results to develop new adder architectures and uses the delay modelling techniques to evaluate the performance of these adders.

Chapter 5

Threshold Logic Addition

THIS chapter proposes two new realizations for 64-bit CMOS, threshold logic adders, and the realizations are shown to be the fastest 64-bit CMOS adders, not requiring a large number of non-overlapping clock phases, proposed to date.

The chapter begins by formulating the well known carry-lookahead prefix scheme in a form suitable for implementation in threshold logic. The first proposed 64-bit adder is based on a hybrid carry-lookahead/carry-select scheme using threshold logic and conventional CMOS logic. The adder is designed, laid out and simulated in a 0.35 μm process. The worst case critical path delay from the simulated extracted layout is shown to be faster than previously proposed high speed Boolean dynamic logic adders.

The second proposed adder is designed based on systematic transistor level delay estimation using Logical Effort (LE) presented in the previous Chapter. The adder is a variant hybrid design consisting of domino and CRTL. The delay evaluation is based on LE modeling of the delay of the domino and CRTL gates. From the initial estimations, the 8-bit sparse carry lookahead/carry-select scheme is selected. Simulations indicate a delay of less than 5 FO4, which is 1.1 FO4 or 17% faster than the nearest domino design, based on a comprehensive survey of published adder results to date.

Table 5.1. Summary of 64-bit TL adder results including network depth, gate count, maximum weight and maximum fan-in. The adders presented in this work use a hybrid of CMOS and CRTL. The depth of 4 TL gates is in addition to a dynamic-CMOS 2-input OR gate and a static CMOS 2:1 MUX gate.

Year	Depth	# Gates	W_{max}	Fan-in$_{max}$
Siu *et al.* 1991	3	2271	1	128
Beiu 1994	4	448	2^{64}	128
	5	576	256	16
Beiu *et al.* 1994	3	320	2^{64}	128
	4	448	256	16
	5	576	16	8
Vassiliadis *et al.* 1996	3	400	64	19
Beiu 1999	2	4224	2^{89}	128
	3	768	2352	16
	4	392	49	8
	5	693	16	5.6
Ramos & Bohórquez 1999	2	128	2^{64}	129
	3	240	256	17
	4	352	16	9
	5	693	16	5.6
→ This Work (Prefix-8 Adder)	4	62	1	8

5.1 Threshold Logic Addition Schemes

Addition is one of the most critical operations performed by VLSI processors. Adders are used in ALUs, floating point arithmetic units, memory addressing and program counter updates. The critical requirement of the adder is speed, but low power dissipation and area efficiency have become increasingly important in recent years.

Table 5.1 provides a summary of absolute value results for known threshold logic addition schemes, discussed in Chapter 2, for 64-bit wide adders. The table focuses on low depth (up to 5) adders, and lists the depth, number of gates, maximum weight requirement and maximum fan-in. The addition schemes proposed in this chapter significantly improve on these results.

The total gate count is the CRTL gate count shown in Table 5.1 and the conventional CMOS logic used in the design.

5.2 Carry Lookahead Addition

Carry lookahead is a well-known technique for decreasing the latency of addition by reducing the logic depth to $O(\log_2 w)$, where w is the word length of the addends. It is one of the fastest addition algorithms, and allows significant design trade-offs to be made in terms of latency, area and power. The key factor in the proposed addition schemes is the introduction of high-valency threshold logic carry generate and propagate cells, which leads to reduced logic depth addition networks, and hence reduced area and power dissipation.

The addition problem can be expressed in the well known prefix notation in terms of generate, G_j^i, propagate, P_j^i, and carry c_j signals at each bit position j for a width w adder as follows:

$$G_j^i = \begin{cases} a_j.b_j, & \text{for} \quad i = j \\ G_j^k + P_j^k.G_{k-1}^i, & \text{for} \quad j \geq k > i \end{cases} \tag{5.1}$$

$$P_j^i = \begin{cases} a_j + b_j, & \text{for} \quad i = j \\ P_j^k.P_k^i, & \text{for} \quad j \geq k > i, \end{cases} \tag{5.2}$$

where $j = 0, \ldots, w-1$, c_j denotes the carry generated at position i and c_{-1} denotes the carry into the LSB position. Assuming that $c_{-1} = 0$, then the carry signal at position j, c_j, is given by:

$$c_j = G_j^0. \tag{5.3}$$

The direct approach to implementing this scheme in, for example, static CMOS is not practical for any useful wordlength $w \geq 16$, since the amount of circuitry required to assimilate the MSB carry becomes prohibitive. For this reason, and also because of the associative nature of the expressions for G_i^j and P_i^j, carry lookahead adders are usually built using a parallel prefix-tree structure. The threshold logic approach can, however, be used to design circuits which implement carry lookahead addition in a more direct and efficient way than the static CMOS prefix-tree approach.

A modified set of the Boolean Equations (5.1)-(5.3) will now be derived in a form suitable for implementation in threshold logic. The proposed scheme takes advantage of the high fan-in capability of TL to design high valency threshold logic prefix-cells, that

is, prefix-cells which compute group propagate, group generate and carry signals from a large number of input bits.

The input operands $(a_i,\ b_i)$, $i = 0, \ldots, w-1$, are grouped into n-bit blocks. The first stage starts with the computation of the group-generate and group-propagate signals (G_j^{j-n+1} and P_j^{j-n+1}) for each block, directly from the input operands. A carry is generated in a group if the sum of the n bits in the group exceeds (is strictly greater than) the maximum number representable by n sum bits. Therefore a group-propagate signal G_j^{j-n+1} is 1 if the sum of the n bits in the group exceeds the maximum number representable by n sum bits. Similarly, the group propagates a carry originating in the neighbouring group of lower significance and the group propagate signal P_j^{j-n+1} is 1 if the sum of the n bits in the group is equal to or greater than the maximum number representable by n sum bits. This may be written in general equation form as:

$$G_j^{j-n+1} = \mathrm{sgn}\Big(\sum_{k=j-n+1}^{j} 2^{k-(j-n+1)}(a_k + b_k) - 2^n\Big) \tag{5.4}$$

$$P_j^{j-n+1} = \mathrm{sgn}\Big(\sum_{k=j-n+1}^{j} 2^{k-(j-n+1)}(a_k + b_k) - 2^n - 1\Big). \tag{5.5}$$

Equations (5.4) and (5.5) are exactly in the same form as Equation 2.1, which describes the operation of a threshold gate. The input weights for calculating G_j^{j-n+1} and P_j^{j-n+1} are the same, and the gate thresholds differ by 1.

An example will serve to illustrate the ideas. Consider a 3-bit grouping of the input bits $(a_5, b_5, a_4, b_4, a_3, b_3)$. The group generate signal G_5^3 is 1 if the sum of the inputs is greater than the largest number representable in the three sum bits (s_5, s_4, s_3), which is 7. The group propagate P_5^3 signal is 1 if the sum of the inputs is greater than or equal to 7. This can be expressed as:

$$G_5^3 = \mathrm{sgn}(4a_5 + 4b_5 + 2a_4 + 2b_4 + a_3 + b_3 - 8) \tag{5.6}$$

$$P_5^3 = \mathrm{sgn}(4a_5 + 4b_5 + 2a_4 + 2b_4 + a_3 + b_3 - 7). \tag{5.7}$$

An expression for calculating G_j^0 may be written by combining the intermediate group generate and group propagate signals in the following way:

$$G_j^0 = G_j^k + P_j^k G_{k-1}^l + P_j^k P_{k-1}^l G_{l-1}^m + P_j^k P_{k-1}^l P_{l-1}^m \ldots P_{x-1}^z G_{z-1}^0. \tag{5.8}$$

Equation (5.8) can be interpreted as expressing the partitioning of the w inputs into contiguous blocks in which it is determined where a carry signal is generated and propagated. Such an expression may also be easily converted into TL form. This is illustrated by the following example for G_{15}^0, where we partition the 16 bits into 4 groups, and use 4 bit group generate and group propagate signals as follows:

$$\begin{aligned} G_{15}^0 &= G_{15}^{12} + P_{15}^{12}G_{11}^8 + P_{15}^{12}P_{11}^8 G_7^4 + P_{15}^{12}P_{11}^8 P_7^4 G_3^0 \\ &= \mathrm{sgn}\Big(8G_{15}^{12} + 4P_{15}^{12} + 4G_{11}^8 + 2P_{11}^8 + 2G_7^4 + \\ &\qquad P_7^4 + G_3^0 - 8\Big). \end{aligned} \tag{5.9}$$

Finally, the sum bits are computed from the truth table for addition as follows:

$$\begin{aligned} s_j &= \mathrm{sgn}(a_j + b_j + c_{j-1} - 2c_j - 1) \\ &= \mathrm{sgn}(a_j + b_j + c_{j-1} + 2\bar{c}_j - 3), \end{aligned} \tag{5.10}$$

where we have used $-c_j = \bar{c}_j - 1$ so that all weights are positive.

By exploiting the parallelism inherent in the computation of carry signals as expressed in Equation (5.8), we can construct carry lookahead adders of significantly reduced logic depth compared to previous prefix-tree approaches.

One possible 16-bit carry lookahead tree structure is shown in Fig. 5.2. The black cells consist of two CRTL gates and compute GP_j^i signal pairs, the grey cells compute the carry signals G_j^0 and the white cells compute the sum according to Equation (5.10). Only one capacitive input network is required for computing the pair of GP_j^i signals because the input weights for computing group propagate and group generate are the same, and it is shared by the two CRTL gates which have different thresholds. The adder has a depth of only 4 gates. This is a significant improvement compared to, for example, a conventional 16-bit Brent-Kung adder which has a critical path consisting of 9 gates, i.e. 7 gates for the prefix-tree, one gate for generating p_i and g_i and finally one XOR gate for computing the sum bits.

To achieve bit-level pipelined operation, the input operands (a_j, b_j) must propagate through the CLA because the sum bits s_j are computed from the input operands as well as the two carries (c_{j-1}, c_j). Therefore each cell in the CLA would also need to include two D-latches. This would result in a compact and potentially low-power pipelined adder suitable for DSP applications. In addition, the proposed CLA has a number of very desirable properties. The adder consists primarily of only one type of CRTL

Table 5.2. MODCVS and CRTL 4-bit adder comparison. This shows the significant advantage of CRTL in the number of transistors required, occupied area and average power dissipation.

	MODCVS	CRTL	Reduction
Transistors	176	104	41%
Area	52×72 μm^2	43×53 μm^2	39%
Avg. power (@ 100 MHz)	1.9 mW	0.5 mW	74%

gate, which means only one gate requires careful design and optimization in addition to the relatively simple capacitive networks. The regularity of each cell also means that networks of the type shown in Fig. 5.2 are highly suitable for automated layout generation.

A 4-bit adder was chosen for initial evaluation of the ideas by simulation. To measure the power dissipation, the adder was loaded with minimum sized inverters and the input vectors were set to (a_3 a_2 a_1 a_0) = (0 0 0 0) and (b_3 b_2 b_1 b_0) = (1 1 1 1) and simulated using HSPICE. The c_{-1} was switched from 0 to 2 V at a frequency of 100 MHz and each gate was clocked at 200 MHz. To perform a comparison of the proposed CRTL design, a 4-bit multi-output differential cascode voltage switch (MODCVS) logic adder (Ruiz 1996) was also designed in the same 0.25 μm process and simulated under the same switching conditions. The MODVCS design presented in (Ruiz 1996) calculates all 4 carries in parallel and is claimed to have low power dissipation, low transistor count and low area compared to previous DCVS logic designs. For this reason it was chosen for comparison with CRTL. Both adders were laid out in the same 0.25 μm, 4-metal, 2-poly process. The poly1-poly2 capacitance in this process is 0.8 fF/μm^2, and as was mentioned earlier, the unit capacitance was chosen to be 5 fF. The comparison results are shown in Table 6.7 and demonstrate that CRTL offers significant improvement over the MODCVS design. It is interesting to note that the area of the capacitive networks in the CRTL adder comprise approximately half of the total area of the adder. The reason for this is the relatively low poly1-poly2 capacitance value in the chosen process, and further significant area reduction would be possible by using processes with dedicated capacitive layers such as MIM.

5.3 The A-DELTA Adder

In this section, the design of the Adelaide-Delft Threshold Logic Adder (the A-DELTA Adder) is presented. The underlying approach for the design of the 64-bit adder is to utilize at the circuit level, a hybrid scheme of combined conventional CMOS logic and TL.

5.3.1 Adder Architecture

At the architecture level, the design is structured to optimize speed based on the advantages and limitations of the logic gates used. A combination of anticipated-computation, carry-lookahead, and carry-select is used. At the implementation level within the blocks comprising the 64-bit adder, CRTL is used to achieve the minimum critical path delay, and where possible, static CMOS is used to reduce the area.

The block diagram of the proposed 64-bit adder is depicted in Fig. 5.7. In this scheme, the 64-bit input addends are divided into four 16-bit adder blocks. Each 16-bit block generates a carry signal. These carry signals form the inputs of the global carry-lookahead unit which generates the select signals, c_{31}, c_{47} and c_{64}. These select signals are used to select the appropriate 16-bit final sums using three 32 to 16 multiplexers. The critical path of the 64-bit adder, shown using the thick dashed line in Fig. 5.7, consists of the generation of the carry-outputs of one 16-bit block, the global carry-lookahead computation that computes c_{47} which then selects the most significant 16-bit anticipated sum, and one 32 to 16 MUX delay.

The three anticipated sums are computed in the 16-bit blocks using pairs of 16-bit adders with $c_{\text{in}} = 0$ and $c_{\text{in}} = 1$. The 16-bit anticipated sums must be ready before the global carries arrive to perform the selection. It is this balance of quickly computing the carries out of the 16-bit adders and the global carries considering the maximum fan-in and delay of CRTL, which dictates a partition of the 64-bit inputs into four 16-bit wide blocks.

To achieve the lowest critical path delay, the computation of the carry-output of each 16-bit adder and the global carries must be as fast as possible. In theory, it is possible to compute the carry-output signal for such a 16-bit block in one TL gate using the scheme first presented in (Vassiliadis *et al.* 1996). However, this leads to a requirement for the sum of weights in the TL gate of $2^{17} - 2$, which is prohibitively large for the

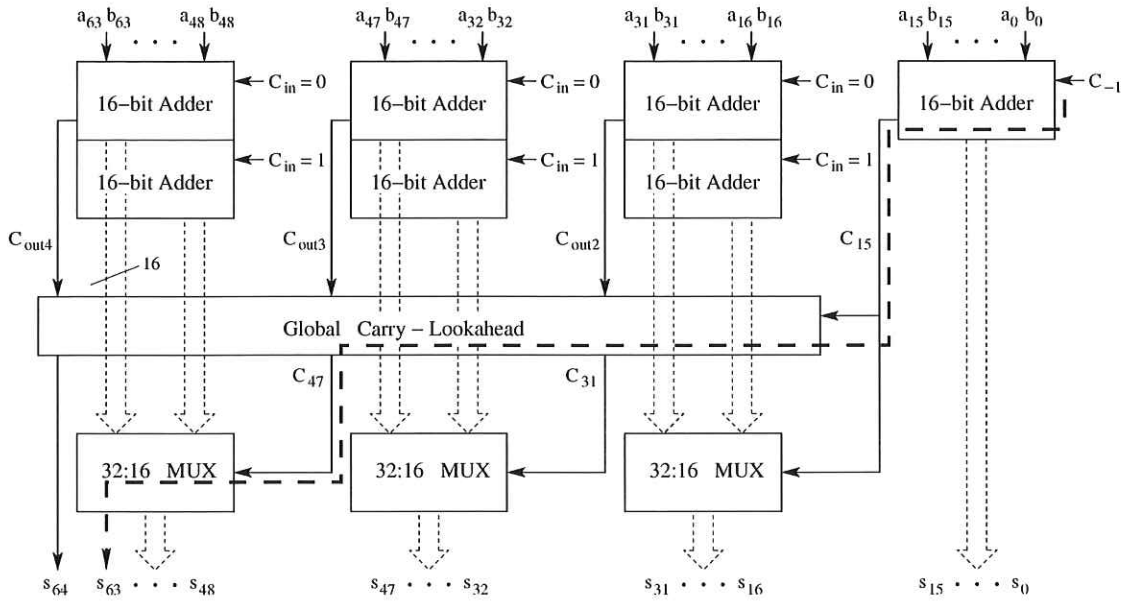


Figure 5.1. Proposed 64-bit adder block diagram. The critical path is shown using the thick dashed line. From top to bottom, the adder consists of 16-bit sub-adders using a carry-select scheme, a global carry-lookahead and multiplexers to select the correct anticipated sub-adder result.

state-of-the-art CRTL (or any other TL gate implementation for that matter). For this reason, the 16-bit carry computation is performed in two levels, as discussed in the following section.

5.3.2 Design of the 16-bit Adders

The design of the 16-bit adders uses carry lookahead. This is a well-known technique for designing logarithmic depth addition networks. To design the high speed 16-bit adder, the high fan-in capability of CRTL is utilized to implement carry-lookahead prefix-cells which compute group carry-generate and group carry-propagate signals from a large number of input bits.

The design of the CRTL prefix-cells is influenced by the maximum sum of weights of CRTL. This is limited by the available precision with which the capacitive weights are implemented and, more significantly, the resolution of the sense amplifier under the influence of process mismatch and noise (Luck *et al.* 2000). For this reason we limit the sum of weights to approximately 30, and use this as the basis on which to optimize the critical path delay of the 16-bit adder. This, as will be shown, leads to the CRTL prefix-cells having 4-bit word inputs.

The group generate , G_3^0, and group propagate , P_3^0, signals are computed from the 4-bit grouping of the input addend bits as follows:

$$G_3^0 = \text{sgn}\{8a_3 + 8b_3 + 4a_2 + 4b_2 + 2a_1 + 2b_1 + a_0 + b_0 - 16\} \quad (5.11)$$

$$P_3^0 = \text{sgn}\{8a_3 + 8b_3 + 4a_2 + 4b_2 + 2a_1 + 2b_1 + a_0 + b_0 - 15\}. \quad (5.12)$$

In the second level, 2-, 3- and 4-input static CMOS prefix cells (Beaumont-Smith and Lim 2001) assimilate the 4-bit group-generate and group-propagate signals to compute the carries at each bit position.

The sum bits are computed using conventional CMOS XOR gates,

$$h_j = a_j \oplus b_j \quad (5.13)$$

$$s_j = h_j \oplus c_{j-1}. \quad (5.14)$$

The computation of the half-sum, h_j, is performed in parallel with the calculation of the carries in each 16-bit block. The sum bits, s_j, are calculated in parallel with the 64-bit adder global carry-lookahead.

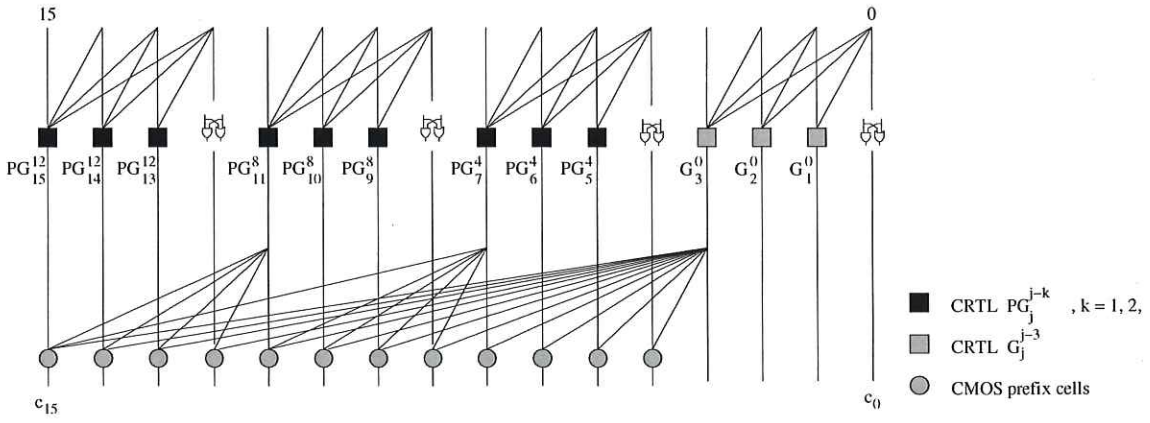


Figure 5.2. 16-bit adder carry prefix-tree schematic. The grey squares represent group carry generate circuits, black squares represent combined group generate and propagate circuits and the grey circles represent the final stage carry-generate circuits.

The schematic diagram of the proposed 16-bit adder carry prefix-tree is shown in Fig. 5.2. The critical path uses CRTL prefix-cells to compute the carry-propagate and

carry-generate signals, PG_j^i, for groups of 4-bits using Equations (5.4) and (5.5) as indicated by the black and grey squares in Fig 5.2. Each black square cell consists of two CRTL gates with thresholds differing by 1 and sharing the same capacitive network for computing the weighted sum of inputs. Grey squares represent group carry generate cells G_j^i. The remaining cells of the prefix-tree consist of conventional static-CMOS carry generate and propagate cells to reduce overall area. These are indicated by grey circles in Fig. 5.2 and the circuit for the case of G_j^{j-3} is given in Fig. 5.3. The MUX circuit was implemented using static-CMOS, and Wang's XOR (Zimmermann and Fichtner 1997) gate design was used in the half/complete sum computation.

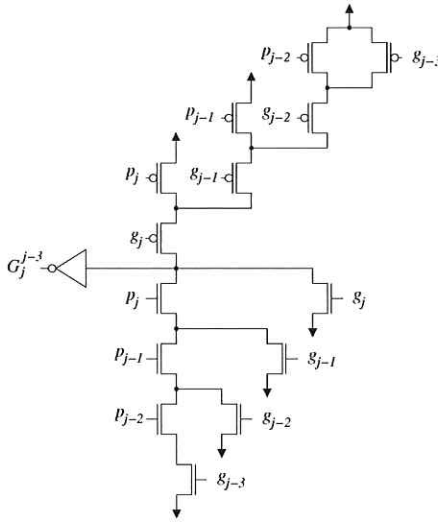
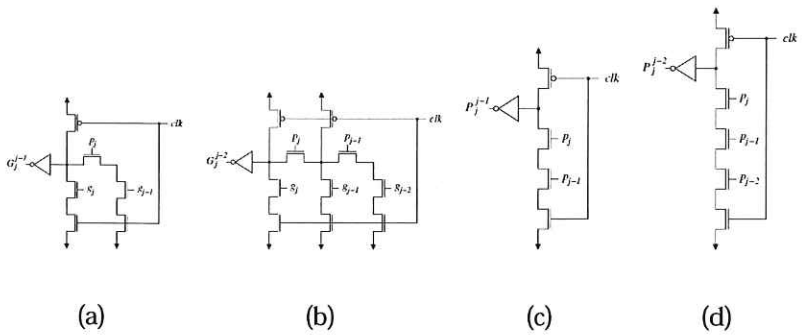


Figure 5.3. Static-CMOS prefix cell. This prefix cell is for computing G_j^{j-3}.

5.3.3 Layout and Simulation Results of A-DELTA

The layout of a 16-bit adder block using a double-polysilicon, 4-metal, 0.35 μm process is shown in Fig. 5.5, and the layers of cells corresponding to Fig 5.2 are indicated. The complete 64-bit adder layout is shown in Fig. 5.6. The global carry-lookahead block cells are merged into the 16-bit adder blocks. The layout of the CRTL gates in the adder must be considered as an analog circuit requiring the use of known layout techniques to match devices and minimize the impact of noise. These include substrate voltage control, shielding, isolation and same-orientation layout of transistors.

The critical path of the 64-bit adder consists of one CRTL gate to compute 4-bit group generate/propagate signals, followed by one static-CMOS prefix cell to compute c_{15}, one static-CMOS cell to compute c_{47} in the global carry-lookahead block and the MUX



(a) (b) (c) (d)

Figure 5.4. Dynamic-CMOS circuits. (a) G_j^{j-1}, (b) G_j^{j-2}, (c) P_j^{j-1} and (d) P_j^{j-2}. The bit-wise carry propagate signals are denoted using p_i and the carry generate signals are denoted using g_i. The gate output is denoted using G_j^{j-i}.

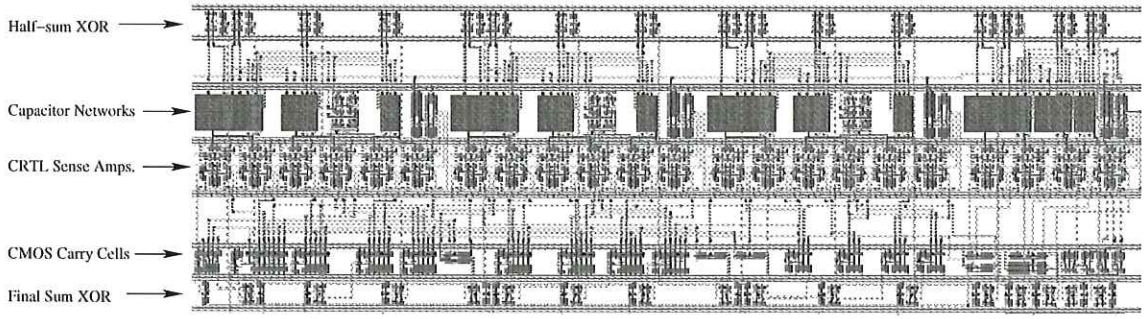


Figure 5.5. 16-bit Adder Layout. From top to bottom, the layout consists of the half-sum XOR gates, weight bank capacitors, CRTL gate sense amplifiers, static CMOS carry cells and final sum XOR gates.

used to select the sum bits $s_{63}, \ldots, s_{48}$. The critical path latency of the extracted 64-bit adder layout was obtained from HSPICE simulation and was found to be 1.02 ns. To compare with the delay of other recently reported adders, the delay was normalized with respect to the "fanout-of-four inverter delay", FO4. The FO4 delay is the delay of a minimum sized inverter driving four minimum sized inverters and the FO4 normalized delay of combinational logic is relatively constant over a wide range of processes (Ho *et al.* 2001).

The FO4 normalized delay and area comparison was made with three other dynamic high speed adders (Sun *et al.* 2001b), (Naffziger 1996) and (Woo *et al.* 2000). The area comparison is made using the reported adder dimensions, normalized with respect to the value of the corresponding process λ. It should be noted that the delay results presented in Sun *et al.* (2001b) are based on simulation and the results in both Naffziger

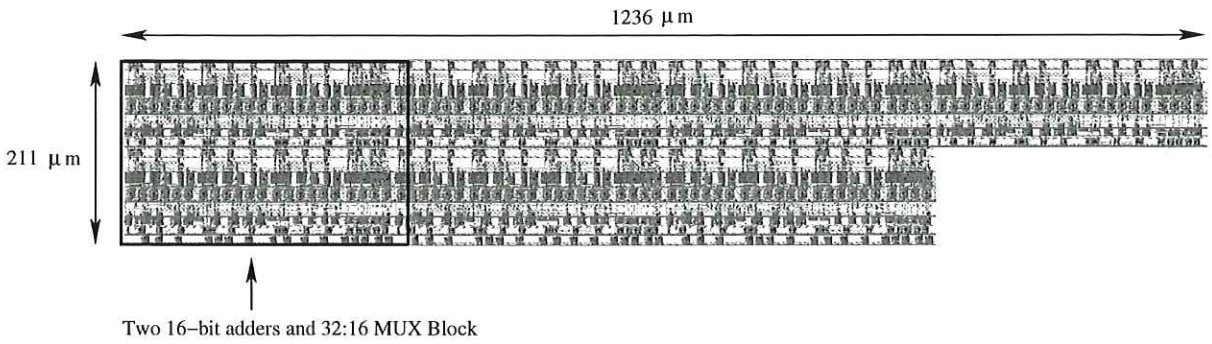


Figure 5.6. 64-bit Adder Layout.. Two 16-bit adders and with a 32:16 mux block are repeated.

(1996) and Woo *et al.* (2000) are from chip measurements. An average speedup of almost 20% is achieved, with an increase in area compared to (Naffziger 1996), and a reduction in area compared to both (Woo *et al.* 2000) and (Sun *et al.* 2001b).

5.4 Prefix-8 Adder

This section presents the second high-speed 64-bit hybrid TL adder design. The delay estimation based on logical effort has been carried out for a number of high speed adders (Dao and Oklobdzija 2001), including dynamic Kogge-Stone (D-KSA), dynamic carry look-ahead (D-CLA), dynamic Ling/conditional-sum (D-LCNSA) and Intel's Quarternary (D-QTA) (Mathew *et al.* 2002). This work is extended to include CRTL based adders. For completeness, a comparison with the HP Ling adder (Naffziger 1996), Harris' adder (Horowitz 1999) and the Output Prediction Logic adder developed by Sechen's group (Sun *et al.* 2001a) is also included.

5.4.1 Adder Architecture

The selection of the adder architecture is heavily influenced by the availability of fast high fan-in CRTL gates. This leads us to use CLA (carry look-ahead) and CSA (carry-select) blocks. The adders described in (Naffziger 1996) and (Horowitz 1999) are based on 4-bit CLA blocks, which is usually the optimal trade off between the depth of the CLA tree and the number of series transistors in a CMOS gate. The carries in these adders are generated at 16-bit boundaries, requiring 16-bit sub-adders for carry-select blocks.

As an initial test of the relative performance of CRTL, we designed the 4-bit CLA block using both CRTL and domino logic. The carry generate signal, α, of a 4-bit block may be calculated using a single TL gate as follows (Vassiliadis *et al.* 1996):

$$\alpha = \text{sgn}\left\{\sum_{i=0}^{3} 2^i(a_i + b_i) - 2^4\right\}. \tag{5.15}$$

The sum of weights $N = 30$, so the worst case delay of this gate will correspond to the delay of a gate of effective fanin, n_{eff}, of approximately 40.

The domino gate used to compute the same function for comparison is the well known Manchester-carry circuit. The electrical effort and parasitic delay for the slowest input, g_{j-3}, were extracted from simulation (Sutherland *et al.* 1999) and used to calculate the worst case delay for $h = 1$ and $h = 10$. Under the conditions of equal input capacitance and load, the CRTL gate is 1.3 to 1.6 times faster. This is a significant delay improvement even in this case of a function with a small number of logic inputs.

Increasing the number of bits handled by a CLA block to 8-bits results in fewer logic levels and a more regular design and layout (Sun *et al.* 2001a). This is impractical in conventional CMOS logic, since it requires 8 series transistors. We can, however, take advantage of the wide AND gates in CRTL. We obtain the regular structure shown in Fig. 5.7. In this scheme, the 64-bit input addends are divided into eight 8-bit blocks, and it has $\log_8 64$, or two levels of carry look-ahead. The Kogge-Stone scheme generates carries for each bit position, so no carry select is needed. The 4- and 8-bit block versions have depth $\log_4 64 = 3$ and $\log_8 64 = 2$, respectively. However, they consist of many more CLA blocks with significantly increased wiring and fanouts.

The structure of the proposed adder is a sparse carry prefix tree. In the first layer, the bitwise propagate and generate signals, p_i, g_i, are formed, followed by the computation of eight pairs of 8-bit group generate and propagate signals P_j^{j-7}, G_j^{j-7} in the second layer. These are then assimilated in the global carry look-ahead block to generate the sum selection carries, c_7, c_{47}, ..., c_{55}, which select pre-computed 8-bit sums. These 8-bit adders are also based on carry look-ahead.

The CLA equations may be written as given as Equations (5.16)-(5.19). Each CLA level consists of an AND and an OR gate, which requires significantly lower sum of weights in the CRTL implementation that a single gate AND-OR implementation. The six stage critical path of the 64-bit adder consists the domino-OR2 to generate p_7 (despite the lower logical effort and parasitic delay, this gate has a higher fanout than g_0), AND8

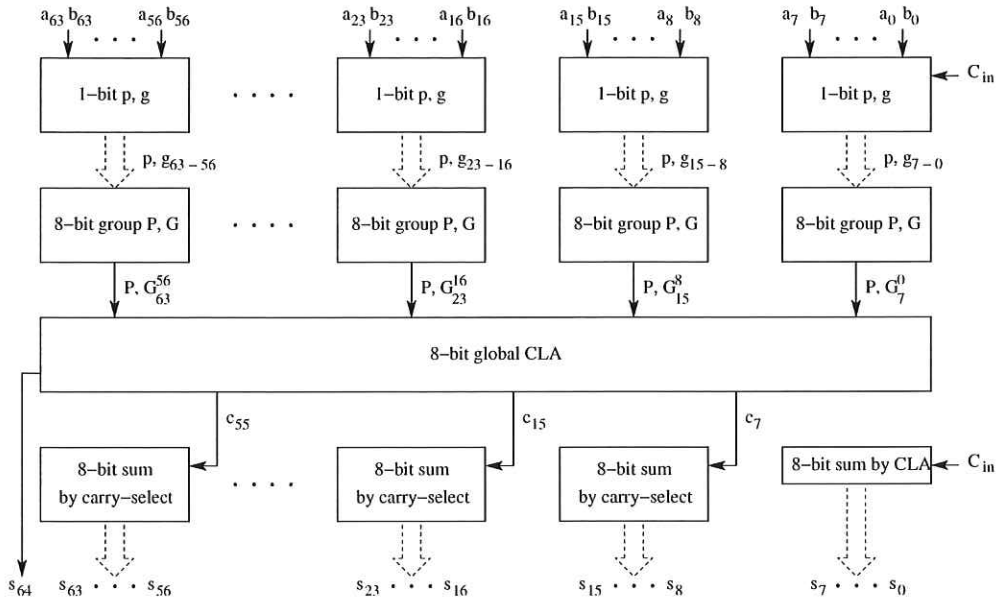


Figure 5.7. 64-bit adder block diagram. This shows the inputs (a_i, b_i), bit-wise carry propagate and generate stage, 8-bit group-wise carry generate and propagate stage, global carry-lookahead and final sum by carry-select stage.

and OR8 to generate G_7^0, AND8 and OR8 to generate c_{55} in the global CLA block and a 2:1 MUX to select the sum.

The bitwise propagate and generate signals are computed as follows

$$p_i = a_i + b_i \tag{5.16}$$

$$g_i = a_i \cdot b_i\,, \tag{5.17}$$

and from these the 8-bit block group propagate and generate signals are given by

$$P_7^0 = p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \tag{5.18}$$

$$G_7^0 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + \ldots + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0\,. \tag{5.19}$$

Finally the 8-bit block carry outputs are given by

$$c_7 = G_7^0\,. \tag{5.20}$$

A similar expression to Equation (5.19) may be written for generating the global look-ahead carries.

Table 5.3. Normalized Logical Effort (LE) parameters of various gates. This includes an inverter, Hi-skew inverter, dynamic 2-input NAND, dynamic 2-input NOR and a 2:1 static MUX—in 0.35 μm technology, in units of $\tau = 40$ ps.

Gate Type	LE, (g)	Parasitic delay, p
Inverter	1	1.18
Hi-skew Inverter	0.7	1
dyn-NAND2	0.4	1.8
dyn-NOR2	0.3	1.4
2:1 static MUX	1.13	2.6

5.4.2 Delay Estimation and Comparison

In addition to the delay model for CRTL discussed earlier, in order to evaluate the adder delay it is necessary to characterize the domino gates using HSPICE simulation of the gate delay for various output loads, according to the LE rules. Characterization of domino gates considers only the one transition of interest, which is the falling transition for the dynamic pull down and rising transition for the hi-skew static inverter. This is repeated for each of the gates, and the results are shown in Table 5.3. Note that the dynamic gates listed consist of the pull down path only, excluding the static inverter.

The delay of the critical path, s_{63}, dyn-OR2 $\rightarrow$ CRTL-AND8 $\rightarrow$ CRTL-OR8 $\rightarrow$ CRTL-AND8 $\rightarrow$ CRTL-OR8 $\rightarrow$ MUX2 is calculated using Equations (4.8) and (4.18) and Table 5.3. For the 8-input CRTL gates we ues an n_{eff} value of 10. In addition, we must consider the fan-out of 7 of the dyn-OR2 gate (which drives 7 unit weight CRTL inputs). The other gates have a unity electrical effort. From Equation 4.10 the optimized delay of the two stage dyn-OR2 gate is therefore given by

$$\begin{aligned} d_{\text{OR2,min}} &= NF^{1/N} + P \\ &= 2\{g_{\text{NOR2}} \times g_{\text{HS-Inv}} \times h_{\text{HS-Inv}}\}^{0.5} \\ &\quad + p_{\text{NOR2}} + p_{\text{HS-Inv}} \\ &= 2\{0.3 \times 0.7 \times 7\}^{0.5} + 1.4 + 1 \\ &= 4.8\tau\,. \end{aligned} \tag{5.21}$$

Table 5.4. Comparison of high speed 64-bit adders showing that the pproposed Prefix-8 adder has significantly lower delay compared to previously published results.

64-bit Adder	# Stages	Tech. μm	LE FO4	Sim. FO4
D-CLA (Dao and Oklobdzija 2001)	14	0.18	11.1	13.6
D-LCNSA (Dao and Oklobdzija 2001)	9	0.18	9.0	9.5
Intel D-QTA (Mathew *et al.* 2002)	10	0.10	8.3	-
D-HCA (Oklobdzija *et al.* 2003)	10	0.10	8.26	-
D-KSA (Dao and Oklobdzija 2001)	6	0.18	6.2	7.4
HP mod. Ling (Naffziger 1996)	4	0.5	-	7
Harris (Horowitz 1999)	-	0.6	-	6.4
OPL (Sun *et al.* 2001a)	8	0.25	-	2.9
→ This Work	6	0.35	4.9	5.3

From this the critical path delay is calculated as follows

$$\begin{aligned} d_{s_{63}} &= d_{\mathrm{OR2,min}} + 4 \times d_{\mathrm{CRTL10}} + (gh+p)_{\mathrm{MUX2}} \\ &= 4.8 + 4 \times 4.26 + 1.13 + 2.6 \\ &= 25.6\tau \\ &= 4.9\ \mathrm{FO4}\,. \end{aligned} \tag{5.22}$$

The proposed adder consists of 3653 transistors and 342 unit capacitors. The critical path was also simulated, including wiring capacitance estimations based on traversed CRTL and domino cell pitch, and the extracted gate layouts and the critical path delay thus obtained was 5.3 FO4. Note that the 207 ps FO4 delay is a very slow process corner for a drawn channel length of 0.4 μm, and is the fastest we had available, (Sun *et al.* 2001a) similarly reports 162 ps for the 0.25 μm process used in that work. It is therefore not surprising that the 930 ps delay for the 0.5 μm process reported in (Naffziger 1996) has a FO4 delay less than ours, especially if a faster process corner was used.

The FO4 delay comparison with eight other dynamic high speed adders is shown in Table 6.7, with the logical effort estimate and simulated or measured delay values listed where available. The comparison suggests a significant delay speed improvement of almost 1.1 FO4 or 17% compared to Harris' agressive domino design. The Output Prediction Logic (OPL) adder is included for completeness to acknowledge other novel circuit techniques in full. It has the significant drawback of requiring 8 clock phases

which has prohibitive power dissipation issues, in addition to the reduced noise margin of OPL gates. Table 6.7 also shows that delay is related to but not proportional to the number of gate levels on the critical path, so comparing delay estimates based on this simple metric is inconclusive.

5.5 Chapter Summary

Two high speed 64-bit adders based on a hybrid carry look-ahead/carry-select scheme using Charge Recycling Threshold Logic and conventional domino logic have been proposed. The worst case critical path delays were shown to be significantly improved compared to previously proposed domino high-speed adders. The results show that by combining TL and conventional CMOS logic with the appropriate architectural strategy, relatively fast arithmetic circuits may be achieved.

The work presented here leaves a number of unresolved questions. The important issue of power dissipation has not been addressed. Power dissipation may be traded for delay and the energy-delay curves for adders may cross (Oklobdzija *et al.* 2003), which implies that single point delay comparisons such as in Table 6.7 are not always meaningful. The results presented here suggest that the substantial delay improvement over domino logic justifies the added design complexity of CRTL.

The following Chapter continues the pursuit of higher performance arithmetic by examining the multiplication operation. Novel parallel counters used in partial product reduction trees are designed, also using CRTL gates and performance is evaluated using the logical effort based delay model developed in the previous Chapter.

Chapter 6

Threshold Logic Multiplication

"At least one good reason for studying multiplication and division is that there is an infinite number of ways of performing these operations and hence there is an infinte number of PhDs (or expense-paid trips to conferences in the USA) to be won from inventing new forms of multiplier."

— Alan Clements, The Principles of Computer Hardware

THIS chapter proposes new realizations parallel counters used in parallel multipliers based on partial-product reduction trees. Significant latency and device count improvements are achieved compared to previously published results.

The chapter begins by developing a number of novel counter circuits based on CRTL and hybrid domino-CMOS/CRTL circuits. The delay results for these counters are extrapolated in the context of partial-product reduction trees (PPRTs) and the performance of multipliers based on these PPRTs are evaluated. Due to the ability of CRTL based circuits to cope with a high number of inputs in each gate while maintaining a delay almost independent of fan-in, a significantly better latency compared to static and domino CMOS logic is achieved.

6.1 Parallel Multipliers

High-performance multiplication is a critical operation in digital signal processing, computer graphics, and necessary for many other applications. Pipelined parallel multipliers provide the highest possible performance, and partial product reduction tree (PPRT) multiplier topologies, typically implemented using parallel counters, provide the best known means of implementation. This chapter presents a number of novel parallel counters using a hybrid of domino and threshold logic. These counters are then used to implement PPRT circuits which outperform existing known implementations. For example, the proposed 64×64 PPRT was found to reduce latency by almost 40% and device count by almost 40% compared to the domino logic equivalent.

The goal of the work described here is to implement high performance parallel counters and partial product reduction tree (PPRT) circuits using a hybrid threshold logic/ domino approach. The main aim is to derive PPRT schemes with minimal latency to produce multipliers with a reduced number of pipeline stages, saving area and power. Area is measured in terms of device count, and it is assumed that power dissipation is a monotonic function of device count.

Initially an introduction to parallel counters is given in Section 6.1.1 and partial product reduction trees in Section 6.1.2. Logical effort analysis is used to estimate delay.

Section 6.2 presents and compares several 3:2 and 2:2 counter designs. Section 6.3 examines larger counters, and introduces novel TL and hybrid domino/TL circuits. Finally, Section 6.5 examines algorithms for PPRT generation using these counters, and compares the performance and efficiency of these circuits.

6.1.1 Parallel Counters

An $m{:}n$ binary counter is a combinatorial network which generates a binary coded output vector of length n, corresponding to the number of logic ones in the m-bit input vector. Typically $n = \log_2(m+1)$—such counters are referred to as *saturated*. The well-known "full adder" is a saturated 3:2 counter; the "half adder" is an unsaturated 2:2 counter.

In multipliers, parallel counters are used to reduce (add) the matrix of partial products (bits)—a 3:2 counter reduces the overall count by one bit; larger counters reduce by more than one.

The concept of parallel counters was originally proposed by Dadda [1965], who also developed a scheme for interconnecting small counters to design counters with a larger number of inputs (Dadda 1980). Oklobdzija et al. [1996] then extended this scheme to the entire PPRT, noting that the addition of partial products of a given binary significance (column) is essentially a counter design problem. In conventional logic, higher order counters, such as (7,3), (15,4) or (31,5), have traditionally been implemented by using trees of (3,2) counters because of the disadvantages of a direct implementation (Song and Micheli 1991). However, counters consisting of such full adder trees have a relatively high delay and grow rapidly with input vector size in terms of the required number of full adders. Swartzlander (1973) reported the number of full adders for an m-input population counter as $m - \log_2 m$. It would be ideal if it were possible to design area efficient higher order counters which could operate at much higher speeds than the same counters built using trees of full adders (Oklobdzija *et al.* 1996). Threshold logic offers this possibility.

Conventional CMOS logic is only effective at low fan-in. Thus PPRT design typically uses small counters, of which the 3:2 is generally regarded as the most efficient (Oklobdzija *et al.* 1996). However, each 3:2 counter removes two bits from the column, and so occasionally a 2:2 counter is required to change from an odd to an even bit count, or vice versa.

6.1.2 Partial Product Reduction Tree (PPRT) Multipliers

Parallel multipliers provide the highest throughput, lowest latency and most area-efficient implementation of the multiplication operation. Although array multipliers provide a higher degree of layout regularity, the less regular partial product reduction tree (PPRT) multipliers are preferable due to significantly lower latency—$O(\log\ n)$ vs $O(n)$ stages for an $n \times n$ multiplier. Also, when pipelined, the latch count for the PPRT multiplier is much lower—due to a combination of fewer latches per pipeline stage and fewer stages overall—providing area improvement over the array multiplier. The benefits are most pronounced at high wordlengths.

PPRT-based multipliers were pioneered in the 1960s by Wallace (1964) and Dadda (1965). There are three stages to a PPRT-based multiplier (1) partial product generation, (2) the PPRT itself, and (3) the vector-merging adder. The former is responsible for generating the matrix of partial products, and can be a simple sea of AND gates

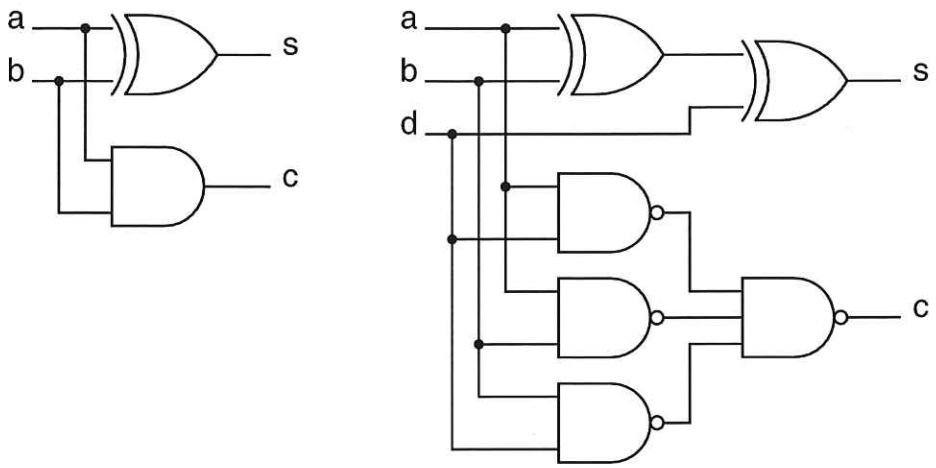


Figure 6.1. Standard 2:2 and 3:2 CMOS counter circuits. The 2:2 counter is shown on the left, it consists of an XOR gate and an AND gate, with two inputs a and b and a sum output s and carry output c. The 3:2 counter on the right consists of two XOR gates, three two-input AND gates and one three-input AND gate and has the additional input d. After Townsend et al [2004].

or a more sophisticated re-coding (such as Booth encoding). The PPRT reduces (adds) these bits to at most two per column, which are combined in the final vector-merging adder. The PPRT is by far the most delay and area intensive portion of the computation Stelling (1998) and is the focus of this work.

6.2 3:2 and 2:2 Counters

6.2.1 Standard CMOS Counters

Although numerous CMOS counter implementations exist, a standard 3:2 counter ("full adder") circuit (Oklobdzija *et al.* 1996) and associated model are the most widely used, and are shown in Figure 6.1. Although the final delay of the circuit in question is slightly worse than some alternative implementations, the overall performance—when the faster carry output path is also considered—is believed to be highly competitive.

The model for the delays in this 2:2 counter circuit sets sum delay $s = b + x_{2h}$ and carry-out delay $c = b + y_{2h}$, where x_{2h} is the delay of the XOR gate, y_{2h} is the delay of the AND function, and b the slower input (i.e. $a \leq b$). For the 3:2 counter circuit, $s = \max(b + x_2, d + x_3)$ and $c = d + y_3$, where x_2 is the delay through both XOR

Table 6.1. Static CMOS 3:2 counter—logical effort analysis. The transistor relative widths for each input, logical effort g, electrical effort h, parasitic delay p, output delay $d(\tau)$ and relative output delay compared to an XOR gate $d(\mathrm{XOR})$.

Gate	width	g	h	p	$d(\tau)$	$d(XOR)$
XOR (a,b)	1	4	1	4.72	8.72	0.77
XOR (d)	1	4	1.67	4.72	11.39	1
NAND-2 (all)	1	1.33	2.5	2.36	5.69	0.5
NAND-3	2	1.67	2	3.54	6.87	0.6

gates, x_3 the delay through only the second XOR, y_3 the delay through the carry output function, and with the inputs arriving such that $a \leq b \leq d$.

The standard model for 3:2 counters (from Stelling (1998)) uses the XOR delay as the unit of time, and sets $x_2 = 2$, $x_3 = 1$ and $y_3 = 1$. For 2:2 counters, $x_{2h} = 1$ and $y_{2h} = 0.5$. Logical effort analysis was performed using the results in (Sutherland *et al.* 1999) and is shown in Table 6.1. We see that the delay of the majority function is about 10% worse than that of the XOR, so the unit delay is 12.56τ.

From an area perspective, each 3:2 counter requires 34 transistors and each 2:2 counter requires 12 (although this may vary if a different XOR gate topology is used).

6.2.2 Domino Logic

The standard CMOS counters are easily adapted to domino logic by simply replacing static CMOS gates with their domino equivalents. All domino logic in this work is dual rail, sized to provide an input loading equivalent to a minimum-sized inverter, with such an inverter buffering each gate's output. Thus, for any domino gate driving any other domino gate, $h = 1$.

Logical effort analysis is again used to estimate performance, using parameters derived from HSPICE simulations. Table 6.2 presents this analysis. Each domino gate is followed by an inverter to provide adequate buffering, and all gates are unit drive strength (width).

Here the delay profile is rather different—the majority function, instead of being slightly slower than one XOR, is slightly slower than *two*. Thus we model it as two XOR delays, that is, $y_3 = 2$ and $c = d + 2$ while s is unchanged.

Table 6.2. Domino 3:2 counter—logical effort analysis. The logical effort g, electrical effort h, parasitic delay p, output delay $d(\tau)$ and relative output delay compared to an XOR gate $d(\mathrm{XOR})$.

Gate	g	h	p	$d(\tau)$	$d(XOR)$
XOR (a,b)	0.44	1	1.8	2.24	0.69
+ inverter	1	1	1.18	2.18	
XOR (d)	0.44	1	1.8	2.24	1
+ inverter	1	3	1.18	4.18	
NAND-2 (all)	0.42	1	1.4	1.82	0.62
+ inverter	1	1	1.18	2.18	
NAND-3	0.52	1	2.3	2.82	1.09
+ inverter	1	3	1.18	4.18	

We may use the same process to obtain a domino 2:2 counter—obtaining $c = b + 1$ and $s = b + 1$. The transistor count for the 3:2 is 66; for the 2:2 it is 22—however, this includes output inverters and a pair of clocking transistors on each gate, some of which may be able to be shared amongst multiple gates.

6.2.3 Threshold Logic—Kautz 3:2

The three most common threshold logic networks for implementing counters are based on networks for computing symmetric functions developed by Kautz (1961), Minnick (1961) and Muroga (1971), which we will refer to as the Kautz, Minnick and Muroga counters respectively. They differ in the number of threshold gates, network depth, maximum fan-in and maximum sum of weights per gate. The counter based on Muroga's network has a delay of 2 gates but is more expensive in terms of gate number. Furthermore, even though the primary inputs are only required in the input layer, the larger number of input layer gates results in an increased interconnect overhead compared to the Kautz and Minnick designs. At 3:2, these counters are identical, but they diverge at larger sizes (7:3 and above). Initially they were designed for computing parity; Cotofana observed that the intermediate outputs available in the Kautz network for computing parity could be used as generalized counter outputs (Cotofana and Vassiliadis 1998).

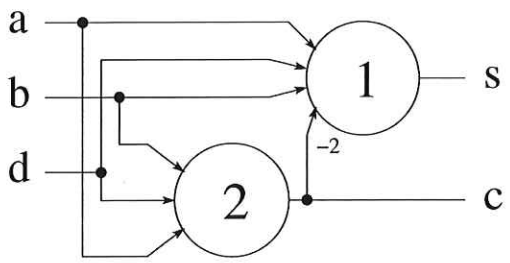


Figure 6.2. Kautz 3:2 counter circuit consisting of two threshold gates. Inputs a, b and d are applied to both gates, with the input weights being equal to 1 in each case and the gate thresholds shown inside the circles. The output of the gate with threshold 2 is fed into the second gate with weight -2. A sum output s and carry output c are generated. After Townsend et al [2004].

The Kautz 3:2 counter is seen in Figure 6.2, and works by initially computing the carry output with a single threshold gate. This is then weighted by two, as it is double the binary weight of the inputs and subtracted from the input vector in a second threshold gate, which thresholds at one to produce the sum output.

The input loading of this CRTL counter is two unit capacitors, so a homogeneous circuit built using them would place an output loading of 4κ on the first gate and 2κ on the second. Logical effort analysis estimates the delays of these gates to be 4.33τ and 4.15τ respectively. Thus, with a base delay unit (quantum) of 4.33τ—faster than both static and domino—we obtain $c = d+1$ and $s = d+2$. This counter uses two threshold gates and eight capacitor units, resulting in an area requirement of 36 devices.

TL 2:2 counters are not considered as the domino half adder performs equivalently and requires less area.

6.2.4 A Hybrid TL/Domino 3:2 counter

CRTL is significantly faster at computing the carry output than domino, and so a single CRTL gate is used—in identical fashion to the Kautz 3:2 counter's carry circuit. The domino sum implementation is somewhat more effective than the TL version due to the presence of a fast input—although the time quantum (defined as the XOR delay for domino and 3-input CRTL delay for TL) is larger. The time quantum ceases to be a factor if the counter is used in parallel with larger TL gates or any domino-only circuits, and so under such circumstances the hybrid is the best performer.

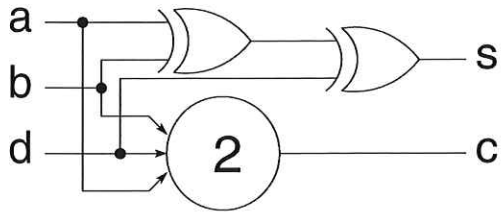


Figure 6.3. Hybrid 3:2 counter circuit consisting of two XOR gates and one three-input threshold gate with unity weights and a threshold of 2. Inputs a, b and d are applied as shown. A sum output s and carry output c are generated. After Townsend et al [2004].

The area requirement of this 3:2 counter—seen in Figure 6.3—is 41 devices. The delay equations are, in fact, identical to those of the standard static CMOS 3:2 counter, although the XOR delay unit is reduced to 5.42τ—as the input loading of this counter is 2κ.

The hybrid 2:2 counter is not considered here, as the threshold gate requires more area than an equivalently-performing domino AND function.

6.2.5 Counter Comparison

From a performance perspective, two possibilities are considered. Initially, we will consider homogeneous circuits—each 3:2 counter will be considered separately and will run at an optimal rate. That is, we will select an optimal time quantum for each counter, and round up each gate's delay to the next multiple of that figure.

Table 6.3 shows area, loading and performance (latency and throughput) information for each of the 3:2 counters. The time quantum is a measure of flexibility in clock frequency choice—the lower the better—and therefore for throughput. Latency is measured in terms of the three delay parameters x_2, x_3 and y_3. Area is measured in terms of device (transistor or capacitor) count.

It can be seen that the CRTL-only circuit is able to provide the smallest time quantum (and thus the greatest flexibility when it comes to pipelining); however, due to the presence of a fast input, the hybrid circuit will outperform the CRTL-only circuit under some circumstances. The hybrid and CRTL circuits both out perform the domino counter in performance and area; static CMOS is more area-efficient but significantly slower.

Table 6.3. A comparison of 3:2 counters (for homogeneous circuits). This shows the device count (transistors and unit capacitors), input load κ, time quantum in number of delay units τ and delay parameters x_i in number of delay units τ.

Circuit	Device Count	Input Load (κ)	Quantum (τ)	x_2 (τ)	x_3 (τ)	y_3 (τ)
Static CMOS	34	6.67	12.57	25.1	12.6	12.6
Domino	66	3	6.42	12.8	6.4	12.8
CRTL	36	2	4.33	8.6	8.6	4.3
Hybrid	41	2	5.42	10.8	5.4	5.4

For heterogeneous circuits, synchronisation will not only need to occur within each cell, but between cells of differing types—domino, TL and hybrid static CMOS, being single rail, requires some interfacing and suffers a severe performance penalty. The common delay quantum will be set $\phi = 6.42\tau$—this is the delay of a two-input domino XOR gate with an output loading of three equivalent gates. That is, the delay of each stage of each counter circuit is rounded up to the nearest integer multiple of ϕ, allowing easy synchronisation and comparison.

This interval (ϕ) is chosen on the basis of the domino 3:2 counter—it is the delay of the slower XOR gate, and exceeds half the delay of the majority function used to generate the carry output. It should be noted that this comparison will slightly favour the domino circuit as a result of this choice.

The results of this comparison are seen in Table 6.4. The hybrid counter is dominant—it is equal to or better than each of the others on every delay path. Thus there is no possible performance gain from producing heterogeneous circuits using only 3:2 (and smaller) counters. The hybrid 3:2 counter will prove useful when combined with larger structures.

6.3 Larger Counters

6.3.1 Implementation

Charge Recycling Threshold Logic (CRTL) can effectively support higher fan-in than typical CMOS circuits. For static CMOS and CRTL, the logical effort g is $O(n)$ in terms

Table 6.4. A comparison of 3:2 counters (for heterogeneous circuits). This includes domino, CRTL based Kautz and a hybrid implementation, showing the delay parameters x_i in number of delay units τ.

Circuit	x_2 (ϕ)	x_3 (ϕ)	y_3 (ϕ)
Domino	2	1	2
Kautz CRTL	2	2	2
Hybrid	2	1	1

of the number of inputs. However, as was shown in previous chapters, CRTL's constant of proportionality is much lower—0.002, against 0.33 for static NAND and 0.67 for static NOR.

It is theoretically possible to implement arbitrarily large counters in threshold logic with near-constant delay. However, in reality, the design of the CRTL counters is influenced by the maximum sum of weights on CRTL gates. The maximum sum of weights sets the minimum voltage which is required to be resolved by the sense amplifier in the CRTL gate. For this reason, we limit the sum of weights to approximately 30, and use this as the basis on which to select counter sizes. In most counter topologies, this will allow us to implement 7:3 and 15:4 counters (and their unsaturated counterparts).

6.3.2 Kautz TL Counters

The Kautz (1961) architecture implements a $2^n - 1:n$ counter using n TL gates, and generates one output after each TL gate delay. That is, we generate the most significant output after one gate delay, the second after two, the third after three and so on.

For example, the Kautz 15:4 counter is seen in Figure 6.4. This circuit accepts a 15-bit input vector V and produces output O_8—most significant—after one gate delay, then subsequently O_4 (two gate delays), O_2 (three) and finally O_1 (four). This is the most area-efficient TL counter architecture, and also places a relatively low load ($n\kappa$) on the input vector. However, it is somewhat slow, with logarithmic worst case delay.

We will differentiate between the various TL counter architectures using their output delay profiles—that is, counter (n_1, n_2, n_3) produces the most significant n_1 output bits after one TL gate delay, the next n_2 after an additional TL gate delay, and the final n_3

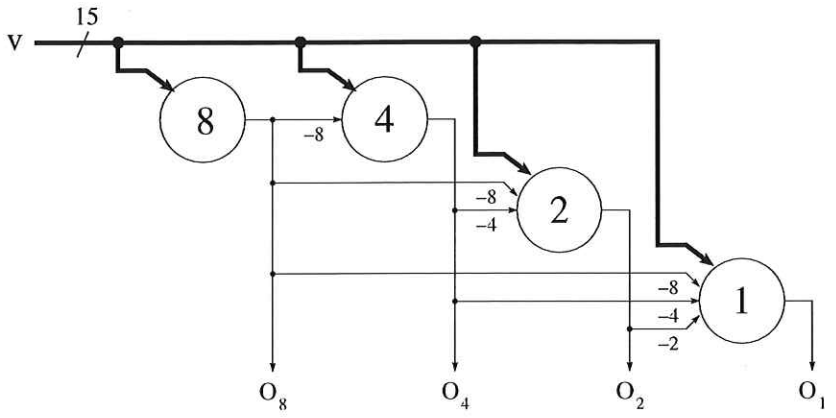


Figure 6.4. Kautz 15:4 counter circuit. The 15-input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each gate input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unit. The Kautz 15:4 network has a depth (critical path delay) of four threshold gates.

after a total of three TL gate delays. Thus the Kautz $2^n - 1 : n$ counter is the saturated[2] $(1, 1, \ldots, 1)$ TL counter (with n delays in total). The Figure 6.4 example is thus the $(1, 1, 1, 1)$ TL counter.

6.3.3 Minnick TL Counters

The faster Minnick (1961) counter, exemplified by Figure 6.5 (15:4), has a worst case delay equal to that of two threshold gates—independent of the order of the counter. The most significant output always has a delay of one threshold gate, with all others computed after a second threshold gate delay. However, it is more expensive in terms of gate count and area; in addition, it places a very large load on the input and on some internal gates, so buffering may be required.

The Minnick scheme operates by thresholding the input at several places in parallel. The first two outputs are generated identically to the Kautz scheme, but all others are generated in parallel with the second. By initially thresholding the input vector at 2, 4, 6, etc we can produce O_1 by providing negative feedback for any pairs of bits—thus, effectively, the gate generating O_1 sees one if an odd number of inputs are high, and zero otherwise. Higher-order outputs need only take a smaller number of negative-feedback inputs (multiples of 4 for O_2, 8 for O_3, etc).

[2]We will assume all counters are saturated unless otherwise stated.

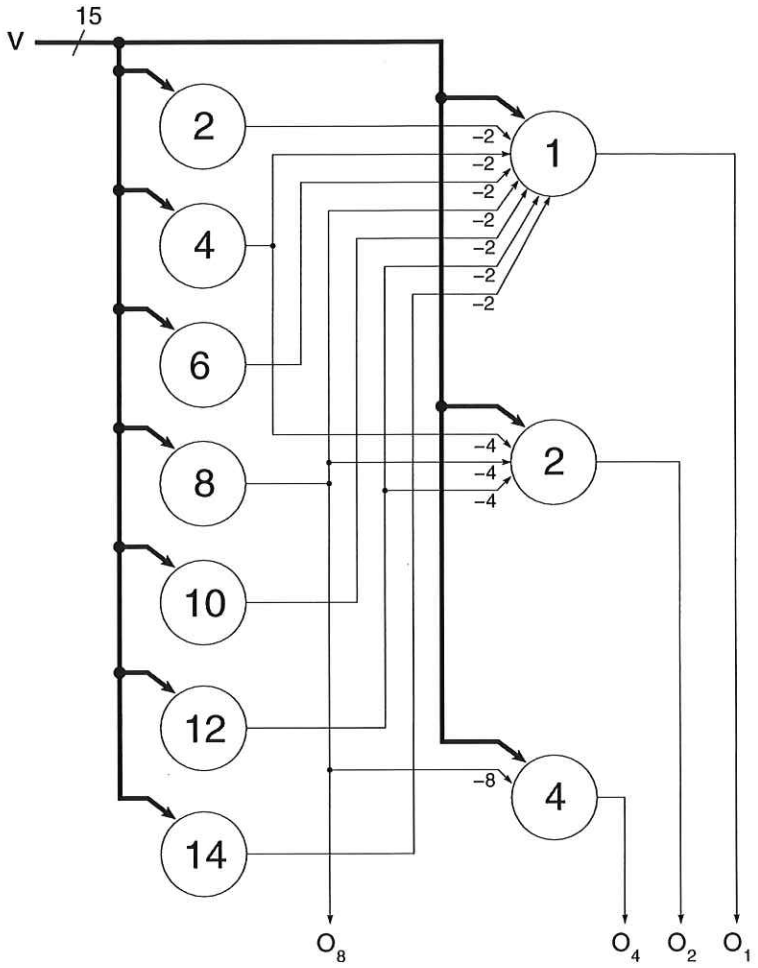


Figure 6.5. Minnick 15:4 counter circuit. The 15-input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unity. The Minnick 15:4 network has a depth (critical path delay) of two threshold gates.

Consistent with our earlier notation, the Minnick $2^n - 1:n$ counter is dubbed the $(1, n-1)$TL counter. In terms of gate delays, this is the fastest known TL architecture, but loading, area and power issues must be carefully considered when weighing this against other options.

As an illustrative example, Fig. 6.6 shows the truth table and TL network for the (7,3) Minnick counter. The input v consists of the seven input bit lines, each having a weight of 1, and is denoted by a thick black line to differentiate it from the single bit lines.

The truth table for the (7,3) counter, and the (7,3) Minnick counter design are shown in Fig. 6.6. The input v consists of the seven input bit lines, each having a weight of 1,

and is denoted by a thick black line to differentiate it from the single bit lines. In effect v represents the arithmetic sum of 1's in the 7 inputs. From the truth table, the MSB of the output, y_2, is 1 when $v \geq 4$, hence y_2 is the output of the first layer gate which has a threshold of 4. The y_1 output is 1 when $2 \leq v < 4$ and $v \geq 6$. Therefore the second layer gate which has threshold 2 computes y_1. This gate has an input weight of -4 from the first layer gate which has threshold 4. Similar reasoning can be extended to the output y_0. In the general case, the MSB will be computed by a first layer gate, and the lesser significance outputs are computed in the second layer. The second layer gates have as inputs, in addition to v, the negatively weighted outputs from the first layer to isolate the desired ranges of v where those outputs are 1.

v	O_4	O_2	O_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

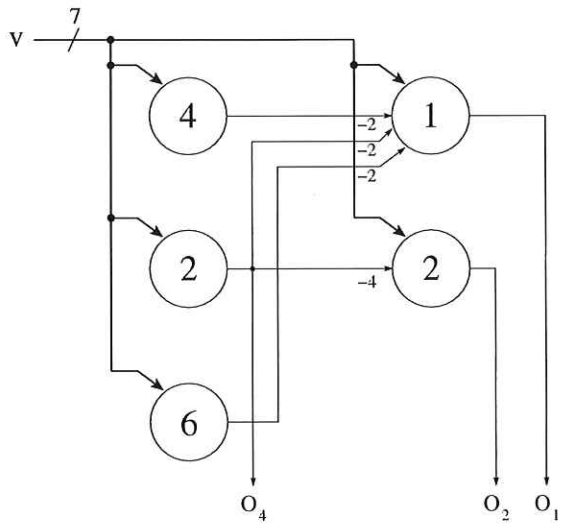


Figure 6.6. The (7,3) counter truth table and the Minnick TL network. The 7-input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unity. The Minnick 7:3 network has a depth (critical path delay) of two threshold gates.

The operation of the (7,3) Minnick counter can be described by the following expressions:

$$\begin{aligned} y_2 &= \mathrm{sgn}(v-4) = 4 \\ y_1 &= \mathrm{sgn}(v-2-4\cdot 4) \\ y_0 &= \mathrm{sgn}(v-1-2\cdot 2-2\cdot 4-2\cdot 6). \end{aligned} \tag{6.1}$$

A different approach (Rodriguez-Villegas *et al.* 2000) to counter design is claimed to have been developed. This approach is based on a sorter circuit followed by one layer

of threshold gates to obtain the counter outputs. This circuit is in fact equivalent in structure to the Muroga counter reported earlier. While it was shown to have improved speed over a conventional logic full adder based design for the case of a (15,4) counter, its logic depth is the same, its interconnect requirement is significantly higher and its gate count is almost double that of the Minnick counter.

6.3.4 A Spectrum of TL Counters

The aim of this section is to develop a range of counters which may be considered to lie in-between the ends of the counter topology spectrum ranging from the Minnick design to Kautz. We focus on small counters which are feasible to implement in the currently available CMOS processes. Typically, the available precision with which the weights are implemented in VLSI is of the order of 4 to 6 bits (Lauwereins and Bruck 1991) for an analog implementation, and for this reason we limit the maximum sum of weights inputs per TL gate to approximately 30.

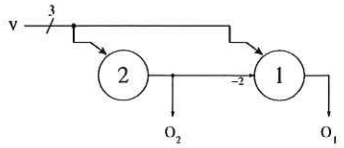
The objective will be to develop binary counters which can be used in conjunction with the known Minnick and Kautz networks to design near area optimal multiplier partial product matrix reduction schemes, given a particular CMOS TL gate implementation. Counters which trade off network depth against gate count and fan-in can be used off the critical path of the partial product reduction tree, reducing the overall area of the multiplier without a penalty in latency. The following sections show the known existing topologies for counters which have a maximum of 29 inputs per TL gate, including the Minnick and Kautz networks, and the proposed in-between (15,4) counters.

Depth 2 (3,2), (7,3) and (15,4) Counters

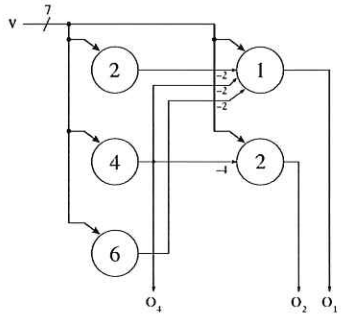
Figures 6.7(a)-(c) show the saturated depth 2 Minnick counters up to 15 inputs. The delay of the most significant output bit is always one TL gate, and all other outputs have a delay of 2 gates. For the (7,3) Minnick counter, the delay for each output is denoted by [1,2,2].

Logarithmic Depth (7,3) and (15,4) Counters

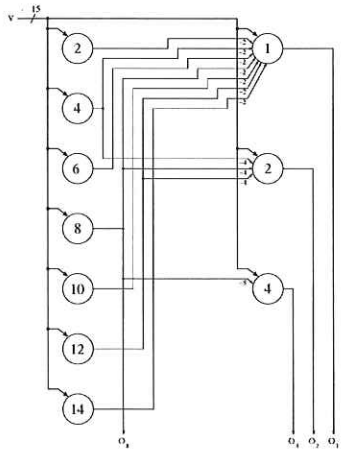
Figures 6.8(a)-(b) show the saturated logarithmic depth Kautz counters up to 15 inputs. The (3,2) Kautz counter is identical to the (3,2) Minnick counter. The (7,3) Kautz



(a)



(b)



(c)

Figure 6.7. Minnick counter networks. (a) Depth 2 (3,2) counter with [1,2] delay profile, (b) depth 2 (7,3) counter with [1,2,2] delay profile and (c) depth 2 (15,4) counter with [1,2,2,2] delay profile. The input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unity and the network depth (critical path delay) is 2 threshold gates in each case.

counter has the [1,2,3] delay profile, and the (15,4) Kautz counter has the [1,2,3,4] delay profile.

Proposed Depth 3 (15,4) Counters

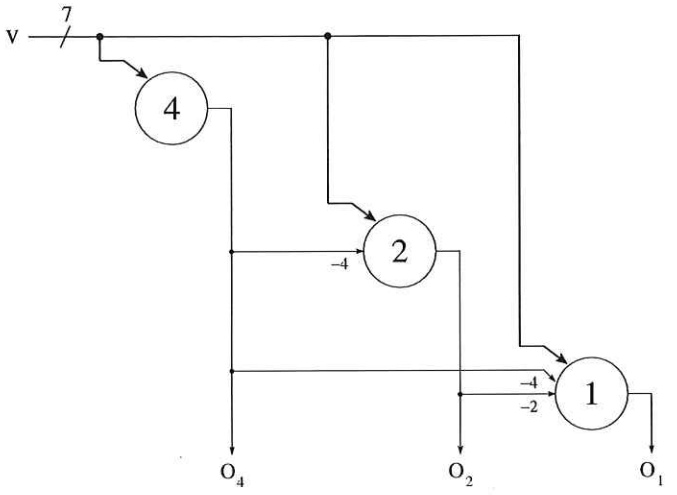
In addition to the depth 2 and logarithmic depth counters, it is possible to devise networks by using a hybrid approach. Figure 6.9(a) shows the proposed (15,4) counter with a [1,2,2,3] delay profile. This means that the output O_8 is generated after 1 gate delay, outputs O_4 and O_2 are generated after 2 delays and the O_1 output after 3 delays. The O_2 and O_4 outputs are generated as in the Minnick counter in Figure 6.7(c) and the least significant output bit O_1 is generated in Kautz fashion as in Figure 6.8(b) using O_2, O_4 and O_8. Figure 6.9(b) shows the (15,4) counter with [1,2,3,3] delay profile. This counter computes the outputs O_1, O_2 and O_4 in Minnick fashion based on the value of O_8. Both counters have a maximum sum of weights equal to 29, but the [1,2,2,3] counter has a total of 34 non-I/O weights, compared to 50 for the [1,2,3,3] design. Other counters such as [1,3,3,3] and [1,3,2,3] are possible but would not reduce the gate count, thus increasing delay with no reduction in cost and are therefore ignored.

Because the hybrid designs have a reduced cost compared to the (15,4) Minnick networks, they are useful in reducing the gate count and area wherever 3 gate delay counters do not increase the critical path delay of the overall network, and where the 4 gate delays of the Kautz design would do so. The cost of each of the counters designs will be evaluated and compared in the next section.

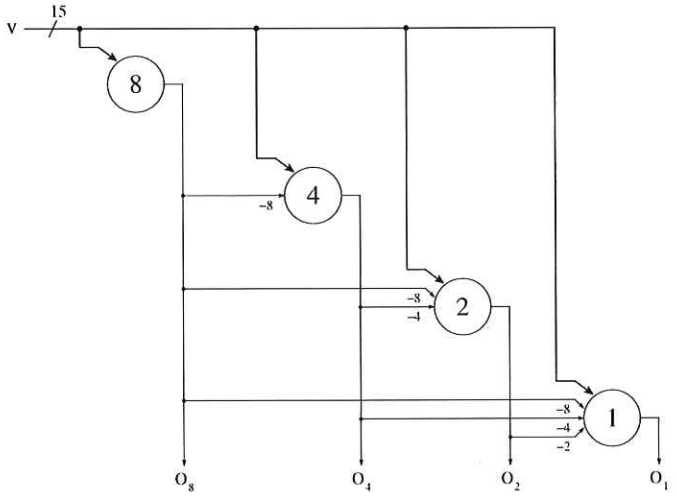
For higher order counters (such as 31:5), more options are available. The first stage can only produce one output bit, but subsequent stages may produce anything from one bit up to all remaining bits. We have two choices for each bit after the initial two—produce it serially (as in Kautz) or in parallel (as in Minnick) to the previous one—so there are a total of 2^{n-2} such counter topologies for the $2^n - 1:n$ counter (including pure Kautz and Minnick counters).

Some counters will be clearly superior to others though—not all have a niche. For example, the $(1, 2, 1)$ 15:4 counter is superior to the $(1, 1, 2)$ in area (lower weights) and latency (second output is faster), and is otherwise largely equivalent.

It is interesting to note that the maximum sum of weights is constant across all equal-sized counters in the spectrum—it always occurs on the gate generating O_1, and is $2^{n+1} - 3$ for a $2^n - 1:n$ counter. Thus with an upper limit of approximately 30 for



(a)



(b)

Figure 6.8. Kautz counter networks. (a) Depth 3 (7,3) counter with [1,2,3] delay profile and (b) depth 4 (15,4) counter with [1,2,3,4] delay profile. The input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unity. Each network has a depth (critical path delay) which is a logarithmic function of the input vector width.

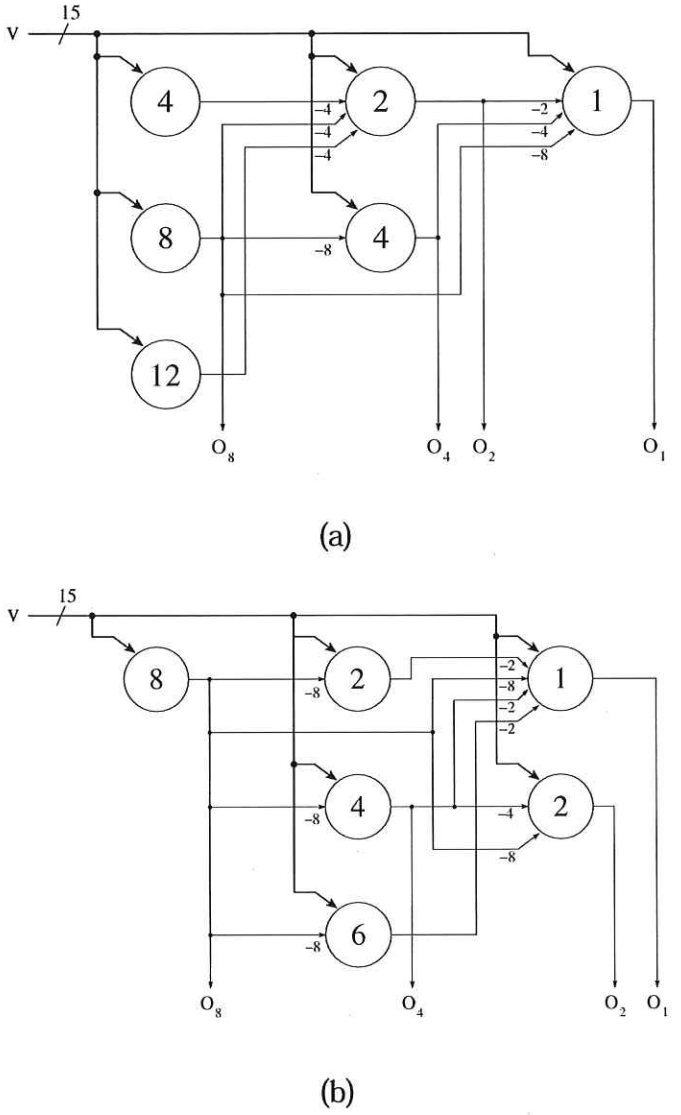


Figure 6.9. Hybrid counter networks. (a) Depth 3 (15,4) counter with [1,2,2,3] delay profile and (b) depth 3 (15,4) counter with [1,2,3,3] delay profile. The input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. The input weights are unity and the network depth (critical path delay) is 3 threshold gates.

CRTL, we may implement all 15:4 counters in the spectrum as their maximum sum of weights is 29.

6.3.5 Hybrid TL/Domino Counters

We may extend this spectrum of counters even further by providing a domino logic option for the least significant output—they are unable to provide sufficient performance for higher outputs, as CRTL leads for the 3:2 carry function, and gains further at higher fan-in. This increases the size of the spectrum by half[3]—as well as the existing two choices (parallel and serial) for the final bit, we have the additional option of using an XOR function. We will represent such a choice as $(\ldots, X)$—e.g. the hybrid 3:2 counter is the $(1, X)$ counter. Note that the position of the X is always at the end, and does not signify timing—as the XOR gates compute the least significant output in parallel.

The least significant output of a counter is equal to the XOR of all the inputs. Since the XOR operation is both associative and commutative, we may combine the input bits in any order and configuration in order to compute the final result. For a $2^n - 1:n$ counter, we require $2^n - n - 1$ two-input XOR gates. If all inputs arrive at the same time, this should be a depth-n binary tree with one fast input, as the tree is not quite balanced. Algorithms similar to those used in PPRT generation may be used to generate optimal structures *inside* the counter, on a per-placement basis.

The hybrids provide several advantages over TL-only counters. Firstly, for parallel structures (Minnick counters being the extreme examples), the number of TL gates is greatly reduced—converting the 15:4 Minnick counter to the equivalent $(1, 2, X)$ hybrid, seen in Figure 6.10, reduces the TL gate count from 10 to 5. Better still, the overall input loading is reduced from 10κ to 6κ, and the output load on the slowest gate (that driving O_8) is reduced by at least 2κ—more if the input is not carefully buffered. We found this to be equivalent to an area saving—once buffering is factored in—of around 30%. Finally, the maximum sum of weights is decreased by two, as the most heavily weighted gate is always removed.

As a result of the reduced loading, the remaining TL gates operate faster—in our 15:4 example, a 27% latency reduction was achieved in the first stage (including input

[3]The number of counters in the spectrum at each size is increased by exactly half, except for 3:2—where it increases from one to two.

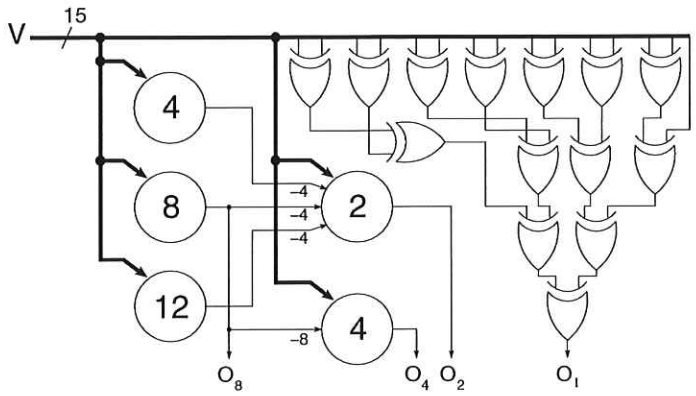


Figure 6.10. The $(1, 2, X)$ hybrid $15:4$ counter circuit consisting of XOR gates and a threshold logic network. The input vector is denoted by v and the outputs are denoted by O_i. The weights and their values for each input are denoted by the number beside each gate input and the gate thresholds are given by the number inside the circles. After Townsend et al [2004].

buffering but ignoring synchronisation), although there was only a 1% boost on the second stage. The XOR tree, assuming equal-delay inputs, will complete calculation in a similar time to the Minnick counter—in terms of ϕ, both are complete by time 4.

Compared with the $(1, 2, 1)$ TL counter, the relative gain is less clear: the delay favours the $(1, 2, X)$ by 23% on the first stage and 1% on the second, with the XOR tree significantly faster (effectively by ϕ after synchronisation) in computing the final input. Area, however, favours the $(1, 2, 1)$ by 32%—although it is worth noting that slower and more area efficient options $(1, 1, 1, X)$ and $(1, 1, 1, 1)$ exist.

6.3.6 Counter Comparison

Large counters are only likely to be used in heterogeneous circuits —3:2 counters are required at the final stage to reduce the bit count down to the maximum of two per column for the final vector-merging adder. Thus we will expect delays at the end of each gate to be synchronised to multiples of ϕ, in order to simplify circuit generation.

In order to ensure each XOR gate is able to keep pace with this, the input loading on some of the counters must be reduced to 3κ. This will only be necessary for TL gates, as the XORs present an input loading of 1κ and the hybrid 3:2 counter is a better choice from both area and performance perspectives than the domino-only 3:2 counter.

Table 6.5. Comparison of saturated counters. Comparison is for area efficiency including domnino and static-CMOS implementations. After Townsend et al [2004].

Circuit	O_8 (ϕ)	O_4 (ϕ)	O_2 (ϕ)	O_1 (ϕ)	Area eff. vs domino	Area eff. vs static
Domino 3:2			2	2	100%	52%
$(1,1)$			1	2	183%	94%
$(1,X)$			1	2	161%	83%
$(1,1,1)$		2	3	4	362%	186%
$(1,1,X)$		1	2	3	224%	115%
$(1,2)$		2	3	3	185%	95%
$(1,1,1,1)$	3	5	7	8	346%	178%
$(1,1,1,X)$	2	4	5	4	217%	112%
$(1,2,1)$	3	5	5	6	271%	140%
$(1,2,X)$	2	3	3	4	185%	95%
$(1,3)$	3	4	4	4	129%	66%

Since all gates being considered (domino and CRTL) are dual-rail, we may use single inverters as buffers, and simply swap the inverted and noninverted rails, thereby halving the number of inverters needed.

Table 6.5 provides latency and area data for a number of different counters, including the best performers. For simplicity, we are ignoring the possibility of a "fast input"—each of the domino/TL hybrids are capable of computing O_1 in one less delay quantum (ϕ) than is shown if the last input is slower than all others (by at least ϕ). Area numbers are normalised against the static and domino 3:2 counters—that is, for a counter reducing n bits (e.g. the 15:4 counters reduce 11 bits), the area is divided by the area of n reference 3:2 counters. Thus numbers exceeding 100% indicate that the circuit is more area efficient than the reference, and vice versa.

It should also be noted that it is possible to save some area if unsaturated versions of these counters are used—for example, the 11:4 $(1,2,X)$ counter will never trigger the threshold-12 CRTL gate in the first column of the saturated equivalent, so that gate can be removed from the circuit. However, we found that all such unsaturated hybrid counters switched at the same speed (after quantisation) as their saturated equivalents, and that none of the unsaturated TL-only counters are able to match the performance

Table 6.6. Comparison of the various designs for saturated (3,2), (7,3) and (15,4) counters.

		Gates	Sum of Non I/P Weights	Sum of I/P Weights	Maximum Sum of Weights per Gate	Cost
(3,2)	Muroga	5	3	12	3	$5g + 15w$
	Kautz	2	2	6	5	$2g + 8w$
	Minnick	2	2	6	5	$2g + 8w$
(7,3)	Muroga	12	10	70	7	$12g + 80w$
	Kautz	3	10	21	13	$3g + 31w$
	Minnick	5	10	35	13	$5g + 45w$
(15,4)	Muroga	25	25	330	15	$25g + 355w$
	Kautz	4	34	60	29	$4g + 94w$
	Minnick	10	34	150	29	$10g + 184w$
	[1,2,2,3]	6	34	90	29	$6g + 124w$
	[1,2,3,3]	6	50	90	29	$6g + 140w$

of the hybrids. Thus no further performance gain is possible, and power and area savings are the only benefits.

From the table, it is clear that the best performing counters are, in each case, the hybrids based on the Minnick topology—that is, $(1, X)$, $(1, 1, X)$ and $(1, 2, X)$. The Kautz counters are, unsurprisingly, the most area-efficient—although static CMOS is slightly better at 3:2 level. Note, though, that the area efficiency of both domino and CRTL may be improved through careful sharing of components—transistors in the former case and capacitors in the latter (Celinski *et al.* 2002a)—so static CMOS is perhaps less area efficient than it may seem from the table. These improvements are difficult to quantify, and so have not been considered here.

Table 6.7 compares the various counter designs for up to 15 inputs in terms of gate count, sum of non-input weights and the maximum sum of weights per gate.

In order to determine the area cost of each of the counter networks, it is necessary to define a cost function which accounts for interconnect, weight area and the area of the sense amplifier component of each TL gate area. The area of the sense amplifier component will be referred to as the *gate area*, and does not include the area required to implement the weights. Unlike conventional AND-OR-NOT (AON) logic, total TL circuit size is not easily related to the number of gates as, in addition to interconnect, the area of TL circuits depends on the number of inputs as well as the weight size. The

contribution of each component to the total cost depends on the CMOS technology and the gate design. Given these considerations, a reasonable definition of a suitable cost function, C, is a follows:

$$C = \sum \text{gates} \times g + \sum w_{\text{I/P}} \times w_1 + \sum w_{\text{nonI/P}} \times w_2, \tag{6.2}$$

where g, w_1 and w_2 are technology and circuit dependent parameters that account for the relative contribution of the gates, input weights, hidden layer weights and interconnect, respectively. The factor $w_{\text{I/P}}$ refers to the number of input layer unit weights and $w_{\text{nonI/P}}$ refers to number of hidden layer unit weights. Typically the input and hidden layer weights will contribute equally to area and thus $w_1 = w_2 (= w)$, which means that a unit input weight occupies the same area as a hidden layer unit weight. However, certain gate designs, especially capacitive differential type gates (Garcia *et al.* 2000, Celinski *et al.* 2002a), are particularly well suited to the area efficient implementation of counter networks and allow the input weights to be shared, thus reducing the total cost.

Given the definition in Equation (6.2), the cost for each counter is evaluated in terms of the g and w parameters in the final column of Table 6.7. As discussed previously, the parameters g and w and therefore the cost depend on the particular circuit design and technology used in the implementation of the TL gate. For example, in a 0.35 μm, 4M-2P STTL (Celinski *et al.* 2002a) implementation without input weight sharing, the area occupied by 10 weight forming capacitors is equal to the area of one gate, so that $g = 10w$. In this case the (15,4) Minnick counter occupies 2.1 times more area than (15,4) Kautz but has 2 gates less worst case delay. Similar calculations can be performed to compare cost for other technologies and implementations. Generally, for a given counter, the Kautz network is the cheapest, followed by Minick and Muroga.

6.3.7 Counter Choice

In theory, it is possible to optimise counter choice for performance across an entire PPRT. However, in reality this is subject to massive computational requirements, and is infeasible for all but the simplest circuits.

Comparison of equal-sized counters is achievable by comparing the vectors of bits in each output column. However, this method breaks down when the number of bits produced in each column varies between counters, as typically occurs when attempting to compare counters of different sizes.

To obtain a formative comparison, the aim will be to compare rather than prove dominance, and we will primarily target the most critical case, which occurs when adjacent columns are of roughly equal size in terms of partial product count. The larger counters are less flexible, so they will only be considered when there is a number of partial products at the same delay—3:2 counters are likely to be more efficient when the delays vary.

To compare, symmetric circuits are used—for each carry out, there is a matching carry in at the same delay, i.e. all adjacent column circuits are exactly equal. Effectively, this means feeding the carry bits back in to this "virtual column" and continuing to reduce down to the desired number of bits. As an example, consider the case where there are six bits at delay zero. A symmetric 3:2 circuit reduces this to three bits in the fashion seen in Figure 6.11—each rectangle represents a 3:2 counter, with inputs at the top and outputs at the bottom. This circuit produces outputs at time 2, 3 and 3 (in terms of ϕ)—compared with the 6:3 unsaturated hybrid counter, which produces outputs at time 1, 2 and 3. Thus larger counters are a better choice in this situation.

The 7:3 hybrid and its unsaturated equivalents compare favourably to the 3:2 hybrid when there are six or more bits available; for five bits, the result is unclear, and the 3:2 circuit will be used as it provides greater adaptability. Similar reasoning shows the 15:4 hybrid counter to be superior when there are eleven or more bits arriving at the same time.

6.4 TL Counter Circuits

The Minnick counter offers the best tradeoff in terms of area and delay. The worst case delay for all outputs is equal to two threshold gates and is independent of the order of the counter. The most significant output always has a delay of one threshold gate.

For these reasons the Minnick counter is chosen as the basis for our modified implementation, and will be referred to as the Modified Minnick Counter (MMC).

As was described in the previous section, each of the gates in the first and second layers includes among its inputs all of the inputs to the counter. This means that the network

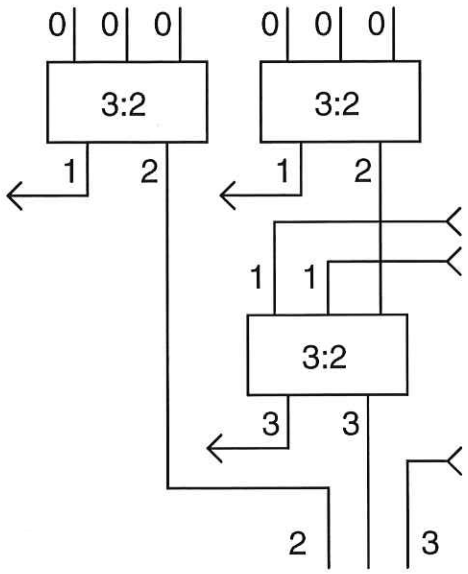


Figure 6.11. Symmetric 6 to 3 reduction circuit using standard 3:2 counters. The numbers next to the 3:2 outputs denote the number of delay units for that output. After Townsend et al [2004].

that performs the weighted summation of the counter inputs (with all weights being 1) in the VLSI layout of the counter is replicated at each threshold gate. In the recently proposed capacitive threshold gate designs, this contributes to a very significant portion of the total counter area. For example, in the (7,3) counter discussed previously, the total number of capacitors performing the summation of the input bits at each of the 5 gates is 35 (5 gates $\times$ 7 input bits). Additionally, there is significant area associated with routing the 7 interconnect lines to each of the 5 gates. These drawbacks have an even greater impact on total area for higher order counters.

The new design proposed here is based on separating, at the circuit level, the two functions performed by the threshold gate, namely weighted addition and thresholding. In other words, the capacitive network, which calculates the analog value of the sum of the counter input bits, needs only to be implemented once, and this value becomes one input of the sense amplifier in any number of CRTL or STTL gates. In the second layer gates, the other sense amplifier input is connected to the capacitive networks which implement the negative weights of the layer 1 to layer 2 interconnections. Additional capacitors can also be connected to the other input to set the gate threshold. Such an arrangement is possible only because of the differential nature of the CRTL or STTL gate and is not possible with other recent TL gate designs including neuron-MOS (Shibata and Ohmi 1991), LPTL (Avedillo *et al.* 1995), CTL (Özdemir *et al.* 1996) or the approach

described in (Garcia *et al.* 2000). It reduces the number of capacitors required from 39 to 22 in the Modified Minnick (7,3) counter implementation.

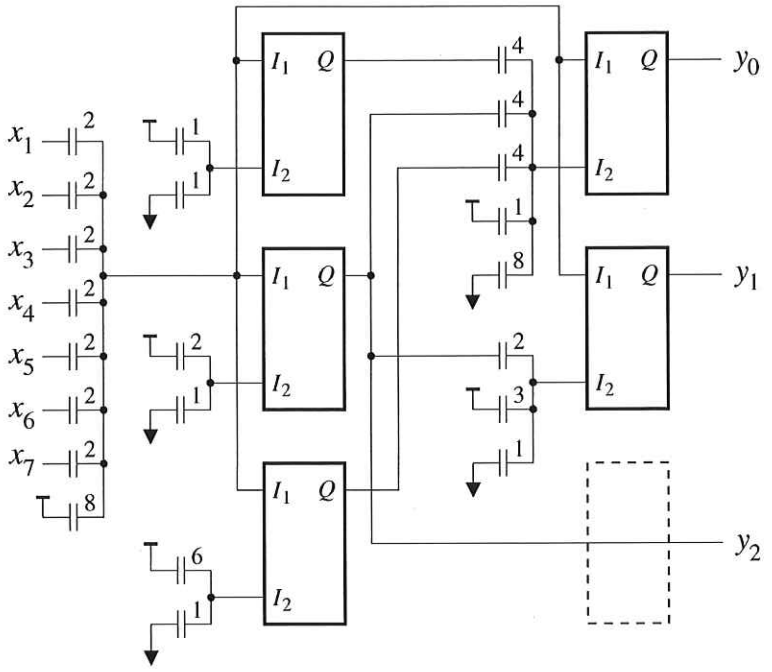


Figure 6.12. Circuit diagram of the proposed STTL Modified Minnick (7,3) counter. The inputs are denoted by x_i and the outputs are denoted by y_i. The capacitor values are given as a multiple of the minimum unit capacitance for the process. The dashed-line box indicates a possible STTL gate to balance all output delays.

The circuit diagram showing this design is shown in Fig. 6.12. The numbers next to the capacitors indicate the multiple of the unit capacitor. The enable signals, E and E_b are not shown to improve clarity. The two gates in the second layer are enabled after the outputs from the first layer are evaluated, as discussed in Section 4.2. The two enable signals of one of the first layer gates drive the enable inputs of both second layer gates. The capacitors shown connected to 0 V and V_{dd} adjust the effective threshold of each STTL gate. The outputs of the first layer gates are connected to the capacitors which implement the negative weights. The inputs denoted by I_1 and I_2 in Fig. 6.12 correspond to the ϕ and T inputs, respectively, shown in Fig. 4.5

Counter Simulation Results

The counter circuit shown in Fig. 6.12 was simulated with HSPICE using 0.25 μm process parameters at a supply voltage of 2 V. The value of the unit capacitor was chosen to be 5 fF. The simulation waveforms are shown in Fig. 6.14 and include the first layer

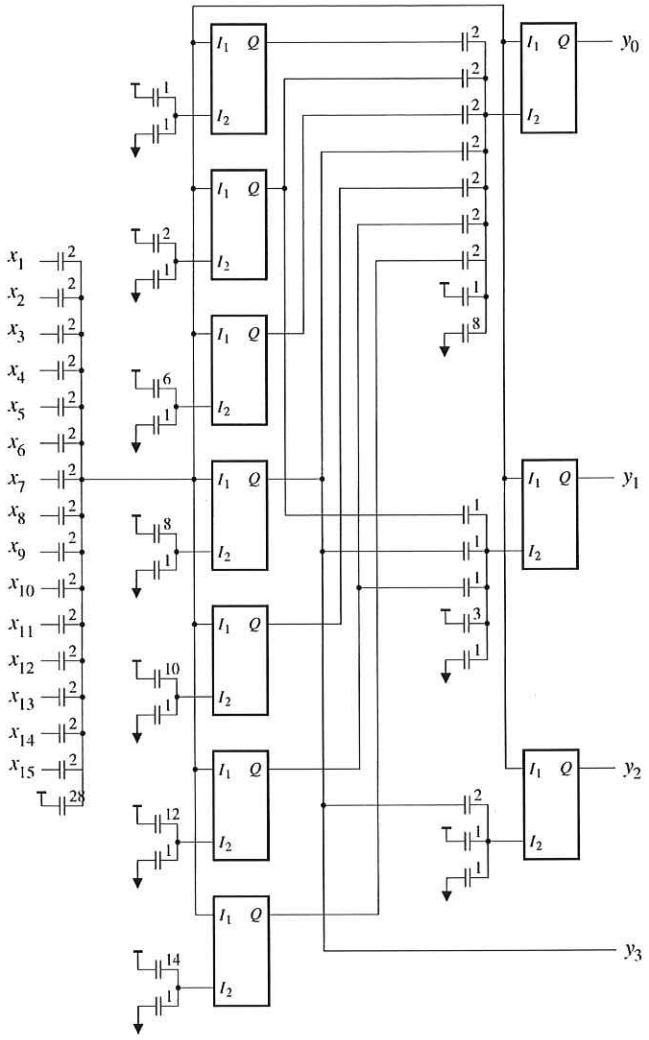


Figure 6.13. Circuit diagram of the proposed CRTL Minnick scheme based (15,4) counter. The inputs are denoted by x_i and the outputs are denoted by y_i. The capacitor values are given as a multiple of the minimum unit capacitance for the process.

enable signal, E, the weighted input vector signal, I_1, and the three output bits. The waveform that has the form of a staircase is the I_1 input to each STTL gate and increases as the number of 1's in the input vector, $(x_1, \ldots, x_7)$, is increased from 0 to 7. It can be seen that when E goes low, the outputs y_2, y_1 and y_0 evaluate correctly for all values of the input vector.

It should be noted that the output y_2 is available after one gate delay and the remaining two outputs are available after two gate delays. All outputs can be made to evaluate simultaneously by adding one additional STTL gate which would act as a delay element for y_2. This gate is shown with dashed lines in Fig. 6.12. The enable signal frequency for the first layer gates was 300 MHz and the power dissipation was measured to be

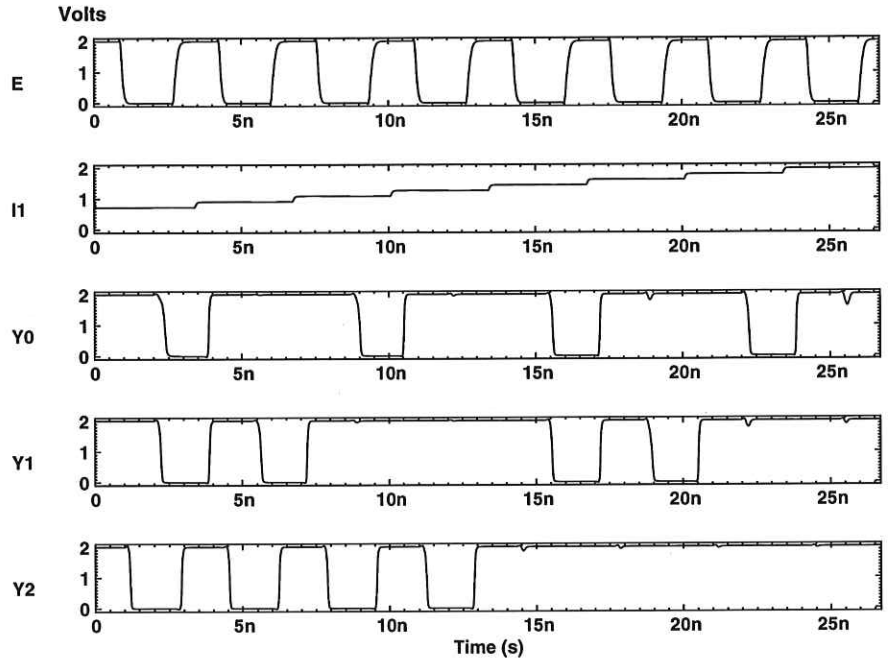


Figure 6.14. Simulation results of the STTL Modified Minnick (7,3) counter.

870 μW. The counter delay is less than 1.4 ns, measured from the falling edge of the enable signal to y_0 or y_1. Compared to other Minnick (7,3) counter implementations, the number of capacitors required is reduced from 39 to 22.

Compared to the design of the (7,3) counter based on Minnick's scheme which does not use capacitor sharing, the required number of (unit) capacitors has been reduced from 94 to 61, and for the (15,4) counter the required number of (unit) capacitors has been reduced from 384 to 152.

To evaluate the area savings, the (7,3) and (15,4) counters were laid out using the proposed capacitor sharing technique (SCRTL) and without sharing (CRTL) using a 3.3 V, double-poly, 4-metal 0.35 μm process. For comparison, a (7,3) counter using the traditional Boolean full-adder (BL) based scheme (Koren 2002) was also laid out in the same process. The extracted layouts were simulated using HSPICE. Table 6.7 shows the area and delay results. For the CRTL based counters, delay refers to the latency, since the gates are clocked.

As is shown in the table, the CRTL based designs are approximately 45% faster than the full-adder implementation for the (7,3) counter. The CRTL (7,3) counter without sharing occupies 50% more area than the full-adder design, and the capacitor-sharing

Table 6.7. Counter comparison (0.35 μm 2P/4M process) for Boolean logic (BL), Charge-Recycling Threshold Logic (CRTL and Shared-capacitor CRTL (SCRTL) implementations. The table compares the delay, transistor count, number of unit capacitors required, the occupied area and the normalized area compared to the Boolean logic implementation.

Counter Type	Delay (ps)	# Tran.	# Unit Caps.	Area (μm^2)	Norm. Area
BL (7,3) (Koren 2002)	840	136	-	1970	1
CRTL (7,3)	460	70	94	2952	1.5
SCRTL (7,3)	460	70	61	1848	0.94
CRTL (15,4)	480	140	384	10124	1
SCRTL (15,4)	480	140	152	3956	0.4

technique provides a small area saving when compared to the full-adder based counter. The sharing of capacitors reduces the area by 37% for the (7,3) counter and by 60% for the (15,4) counter compared to the non-shared designs. It should be noted that the use of capacitor sharing does not result in an increase in delay.

A number of other circuit designs have been proposed in the literature based on different process technologies. These include the (15,4) counter using a neuron-MOS based sorter circuit (Rodriguez-Villegas *et al.* 2000) which was reported to have a delay of 8 ns in a 1.2 μm process and for which no area data was given. The design for a (31,5) counter using Capacitive Threshold Logic (CTL) was presented in (Leblebici *et al.* 1996), the worst case delay was 4.2 ns in a 1.2 μm process and the area was approximately 80000 μm^2. Recently a hybrid Threshold-Boolean logic (HTBL) design was proposed (Padure *et al.* 2002a) which uses a single-phase clocked differential cascode voltage switch circuit implementation of the TL gate in combination with static CMOS logic. The delay of the (7,3) HTBL counter in a 0.25 μm process was 345 ps and the design required 237 transistors.

6.5 Partial Product Reduction Trees

6.5.1 Three-Dimensional Method

The best-performing known schemes for PPRT circuit generation are based upon the three-dimensional method (TDM), originally developed by Oklobdzija [1996] and subsequently refined by Stelling [1998]. This method works with one column at a time, iterating from least to most significant, such that all the partial product delays (including carry input bits) for the column are known in advance. In this manner, a highly optimised circuit may be custom generated for the specific delay vector of each column.

Various algorithms are possible for allocating the counters (typically 2:2 and 3:2 CMOS) within each column. The initial attempt detailed in Oklobdzija [1996] is now known as the "Three-Greedy Method", in which one 3:2 counter is placed at a time, using the earliest three input bits. No distinction is made between the sum outputs of previously placed counters, the carry input bits from the previous column and the initially generated partial product bits. More recent works (Stelling 1998, Townsend 2001) investigated the search for optimal PPRT circuits within the framework of the TDM.

Some restrictions are placed upon the counters used for this; however, each of the 3:2 counters presented are within them. The only change required is to the relevant delay constants; the algorithms remain the same. The numerical results published in Oklobdzija [1996] still hold for the hybrid 3:2 counter, but require re-calculation for the domino 3:2 and Kautz 3:2 counters.

6.5.2 Input-Symmetric Counters

The three-dimensional method may be further optimized for input-symmetric counters—that is, counters in which the delay paths from each of the inputs are identical (there are no "fast" or "slow" inputs). For a 3:2 counter, this simplifies the delay equations to $s = d + x_3$ and $c = d + y_3$. That is, all output delays depend only on the arrival of the slowest input to each counter. It is worth noting that the XOR gate (sum only, as used in hybrid counters) and each of the TL-only counters fit this definition.

Because both sum and carry delays are fixed offsets from d, we need only examine the delays of the d inputs to each gate when comparing circuits, and so only have one vector to optimise.

It is relatively straightforward to prove, by contradiction, that the greedy algorithm is optimal for circuits of input-symmetric counters; however, space restrictions prevent the inclusion of the formal proof here.

6.5.3 Heterogeneous Circuits

In order to use the faster 7:3 and 15:4 counter circuits, we need to build heterogeneous circuits—3:2 counters will still be needed to adapt to more difficult circumstances, and to reduce the final few bits in preparation for the vector-merging adder.

Several algorithms were tested; one outperformed the others and is detailed here. This algorithm uses the optimal TDM circuit generator of Stelling [1998] to do the bulk of the work, albeit placing hybrid 3:2 counters instead of static CMOS (as was originally envisaged).

In order to find optimal 3:2 counter circuits, we require the complete input delay profile. Thus any larger counters must be placed in a segment at the beginning of the circuit, the output of which is passed to the TDM segment. In a similar fashion to the TDM, we will proceed one column at a time, starting with the least significant.

The allocation of the larger counters will proceed using a simple greedy algorithm—each n-input counter will take the earliest remaining n bits in the column segment as inputs; the sum outputs are re-integrated. The choice of action depends on the number of bits i available at the same delay as (and including) the earliest available bit:

- $i \geq 15$: place a saturated 15:4 $(1, 2, X)$ counter
- $11 \leq i < 15$: place an unsaturated i:4 $(1, 2, X)$ counter
- $7 \leq i < 11$: place a saturated 7:3 $(1, 1, X)$ counter
- $i = 6$: place an unsaturated 6:3 $(1, 1, X)$ counter
- $i \leq 5$: pass these i bits to the TDM segment.

This process repeats until there are no further bits remaining in the column segment. Once all of the large-counter segments have been processed in this fashion, the TDM optimal circuit finder [Stelling 1998]—which runs significantly faster than for the full PPRT, due to the reduced problem size—generates the remainder of the PPRT using hybrid 3:2 and domino 2:2 counters.

Table 6.8. The PPRT latency comparison (in terms of delay parameter ϕ). For a range of domino, Kautz, hybrid and heterogenous counters and input word lengths.

Algorithm	16×16	32×32	64×64	128×128
Domino 3-greedy	12	15	18	22
Domino optimal	12	15	—	—
Kautz 3:2 optimal	10	13	15	18
Hybrid 3:2 3-greedy	8	11	15	18
Hybrid 3:2 optimal	8	11	—	—
Heterogeneous	7	9	11	14[4]

6.5.4 Results

When working with fixed delay ϕ, throughput is solely a function of pipelining—so latency is the sole performance concern. Latency results are presented in Table 6.8 for various PPRT sizes; it should be noted that computational limits due to the algorithm's exponential complexity prevented us from finding optimal circuits for some cases. In those instances, the three-greedy algorithm (Oklobdzija *et al.* 1996) results are provided; the optimal circuit's performance is unlikely to exceed the three-greedy circuit by any significant amount.

These are results in terms of the domino 3:2 counter, as that was the basis for choosing ϕ. These results clearly show that the Kautz and Hybrid counters are capable of producing lower-latency circuits at the same clock rate, as is the heterogeneous algorithm described in this paper. Comparing between the other algorithms, however, is not quite so clear cut—the results are somewhat dependent on the choice of time quantum, and this may be influenced by other aspects of the circuit. Nevertheless, the heterogeneous algorithm appears to be the best choice for large circuits.

In area terms, the 3:2 circuits may be simply compared on the basis of their counter cells although this will be perturbed slightly by the few 2:2 counters needed. The heterogeneous circuits use a range of counters, but the efficiency does not vary too greatly—from 161% of the domino counter to 224%, which corresponds to a device count reduction of $38 - 55\%$. Thus area improvements are expected somewhere in this

[4]This result is taken from a sub-optimal circuit, which was generated using three-greedy rather than optimal TDM (where applicable). Computational requirements are too great for an optimal circuit to be found.

Table 6.9. Partial product reduction tree area estimates. Comparison with domino and static CMOS logic for a range of counter types.

Algorithm	Area efficiency vs domino	Area efficiency vs static
Domino	100%	52%
Kautz 3:2	183%	94%
Hybrid 3:2	161%	83%
Heterogeneous	$\sim$ 170%	$\sim$ 88%

range; the larger counters are more area-efficient, so the efficiency will likely increase slightly at higher wordlengths. These results are tabulated in Table 6.9.

6.6 Chapter Summary

The results shown by hybrid domino-TL PPRT circuits are highly promising. Due to their ability to cope with much higher fan-in than conventional CMOS logic gates, CRTL based circuits were expected to shine at very high wordlengths, but better performance is evident at wordlengths as low as 16 bits; the latency improvement at 64×64 is 39%. In addition, it is possible to achieve this with area requirements only narrowly exceeding those of static CMOS despite much higher performance, and significantly below those of domino logic.

Having focused in the last three Chapters the advantages of TL circuits using CMOS technology, the focus now changes to exploring threshold logic based techniques in other promising, although not nearly as widespread, technologies including Complementary Gallium Arsenide circuits and Self-Elecro-Optic Devices (SEEDs).

Chapter 7

Optical and GaAs Threshold Logic Techniques

THE focus of this Chapter is the development of Self-Electro-Optic Effect Device (SEED) and Complementary Gallium Arsenide (CGaAs) based threshold logic elements. Firstly, the application of threshold logic elements based on complementary GaAs technology to implement very low power, high speed logic is investigated. Secondly, the application of the counter designs proposed in earlier Chapters is extended to the optical domain and novel ultra high speed analog-to-digital (ADC) converters are proposed and experimental measurement results are discussed.

7.1 SEED Based TL

The optical domain offers a number of advantages over conventional electronics. Optical components typically have much higher bandwidth, low power consumption, improved reliability and insensitivity to electro-magnetic interference.

This section describes a novel Nyquist type analog-to-digital converter (ADC) using threshold logic counters and an opto-electronic device called the self electro-optic effect device (SEED) (Miller 1990). These multiple quantum well structures (MQWS) (Miller *et al.* 1985) consist of alternate thin layers of GaAs and AlGaAs in the intrinsic region sandwiched between *p* and *n* doped AlGaAs layers. The result is a *p-i*(MQW)*n* diode that can behave as a modulator (Souza *et al.* 1994) or as a bistable switch depending on whether the SEED is operating in the negative or positive feedback mode (Miller *et al.* 1985).

Under negative feedback self-linearised modulation, linear light-by-light modulation, optical subtraction and optical level shifting (Miller *et al.* 1985) is possible. Bistability is obtained under positive feedback and this is used to create a threshold logic gate using two SEEDs connected in series, also known as symmetric-SEED . Typically a reference beam is the input to one SEED while the signal beam is the input to the other, however for simplicity, a resistor SEED (R-SEED) network is used in this work.

Section 7.2 describes the system design considerations of an optical threshold logic ADC and a theoretical analysis of the R-SEED network is given. Section 2.3 discusses three designs of a 2-bit ADC, one of which is demonstrated experimentally and simulation results are compared with measurements.

7.2 System Design Considerations

7.2.1 Physical structure of p-i(MQW)-n diode

The devices used in this study consist of forty 95Å/40Å AlGaAs/GaAs quantum wells and were fabricated by the metal oxide chemical vapour deposition (MOCVD) process. These devices were intentionally very large, 500 μm $\times$ 500 μm to simplify fabrication, testing and experimental design. As a consequence, maximum clock speeds were low (less than 1 kHz) but still allowed verification of the proof of concept for the proposed ADC.

7.2.2 Theoretical Model

The self electro-optic effect device in its simplest form is a reverse biased *p-i*(MQW)*n* device as depicted in Figure 7.1(a) and the equivalent circuit model is shown in Figure 7.1(b). The current source represents the current generated by the absorbed incident radiation and the diode represents the dark current generated by the p-n junction. The series resistance arises from the contacts and was assumed to be negligible for simulation purposes.

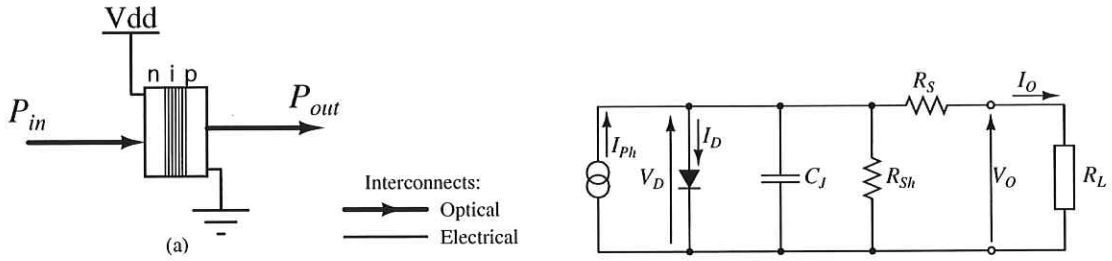


Figure 7.1. SEED device. (a) The simplest network is a SEED with a bias voltage applied across it. The output power P_{out} is a function of the input power P_{in}. (b) shows the equivalent circuit consisting of a current source generating a current I_{ph} from the input power, diode due to the dark current I_{d}, a capacitor C_j and a resistor R_{sh} in parallel and a series resistance R_{s}. After Sarros et al [2004].

Figure 7.2 illustrates a Resistor-SEED. The R-SEED is simply a resistor connected in series with a multiple quantum well *p-i-n* diode. By applying a bias voltage V_{dd}, an input power P_{in} and chosing the correct wavelength and resistor values, the output power P_{out} can be made to switch.

In determining the operation of the R-SEED, a static and dynamic analysis is performed to provide a means of simulating the behaviour and maximum operating speeds to the fabricated devices.

The static analysis of the R-SEED can be used to show that the output power is given by

$$P_{\text{out}} = P_{\text{in}} - P_{\text{abs}} - P_{\text{loss}} = P_{\text{in}} - \frac{h\nu}{\eta(V)q} I_{\text{ph}}(V) - P_{\text{loss}}, \tag{7.1}$$

where P_{loss} accounts for optical loss within the system including reflections from the front of the photo-detector.

Dynamic analysis for an R-SEED is leads to,

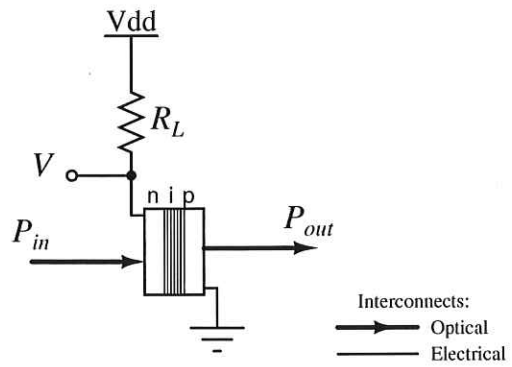


Figure 7.2. Resistor-SEED (R-SEED) network in which a resistor is connected in series with a SEED. It can be used in the bistability or modulation mode depending on the wavelength and bias voltages chosen. After Sarros et al [2004].

$$\frac{V_{dd} - V}{R_L} = P_{\text{in}} S(V) + C\frac{dV}{dt} - I_d(V). \tag{7.2}$$

Equation 7.2 shows that the dynamic behaviour of the R-SEED is a function of the self loading capacitance, the supply voltage, the dark current and the voltage dependent photo-current. Increasing the optical power has a large effect on the dynamic performance. Equation 7.2 is solved numerically in the following Subection.

7.2.3 Nyquist Analog-to-Digital Converter using Optical TL

Nyquist type ADCs have a fast conversion rate with precision typically limited by component mismatch. The optical threshold logic converters discussed here are of this type. This section describes the design and construction of a 2-bit ADC using two R-SEED networks.

The Kautz (3:2) counter described in a previous chapter is shown in Figure 7.3(a), and functions by initially computing O_2 with a single threshold gate. This is then weighted (by two, as it is double the binary weight of the inputs) and subtracted from the input vector in a second threshold gate, which thresholds at one to produce O_1.

To demonstrate this architecture in the optical domain, the weights of the system must be positive since an optical signal is unipolar. The negative weight on the second threshold is removed by rearranging and substituting the logic expression $\overline{x} = 1 - x$. A division of two is carried out to avoid any optical amplification into the next stage.

The calculations for this are shown in Equation 7.3 and the corresponding diagram is shown in Figure 7.3(b). The counter network is equivalent to a 2-bit optical ADC where the input is now a combination of an analog and digital optical signals instead of three binary inputs. The following relations describe the operation of the networks shown in Figure 7.3,

$$\begin{aligned} -2O_2 + P_{\text{in}} &\geqslant 1, \\ 2(\overline{O}_2 - 1) + P_{\text{in}} &\geqslant 1, \\ 2\overline{O}_2 + P_{\text{in}} &\geqslant 3, \\ \overline{O}_2 + \frac{P_{\text{in}}}{2} &\geqslant 1.5. \end{aligned} \tag{7.3}$$

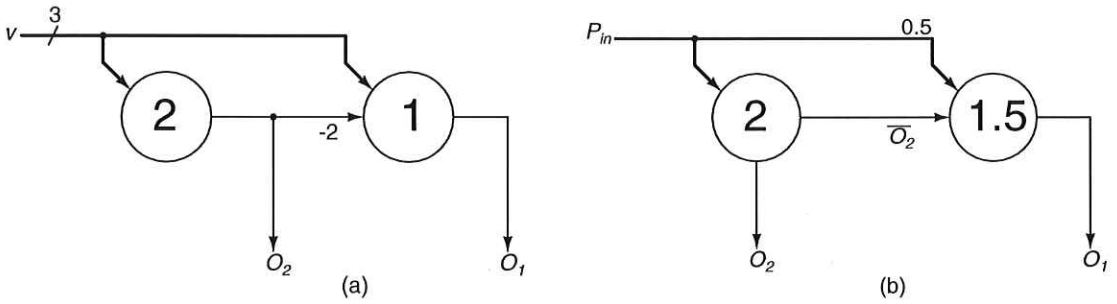


Figure 7.3. Networks for performing the calculations in Equation 7.3. (a) Depth 2 (3,2) Kautz counter with [1,2] delay profile (b) Depth 2 (3,2) Kautz counter with [1,2] delay profile using positive weights.

Figure 7.4 shows the digital output of a 2-bit ADC from an input sinusoidal waveform. This shows the sinusoidal waveform separated into four levels where each level represents a distinct 2-bit value, i.e. `11`, `10`, `01` and `00`. The corresponding digital output waveforms of O_1 and O_2 are shown, where in the case of a 2-bit R-SEED ADC, the digital outputs are inverted.

The 2-bit ADC shown in Figure 7.5 uses two R-SEED networks as described in Section 7.2.2. The R-SEED network represents the optical threshold logic gate where the resistor varies the switching point of the input power. The operation is as follows: P_{in} represents the analog input signal entering the first R-SEED network while $P_{\text{in}}/2$ is required for the second R-SEED network to satisfy Equation 7.3. The output of an R-SEED network is inherently inverted, i.e. a low input power yields a high output

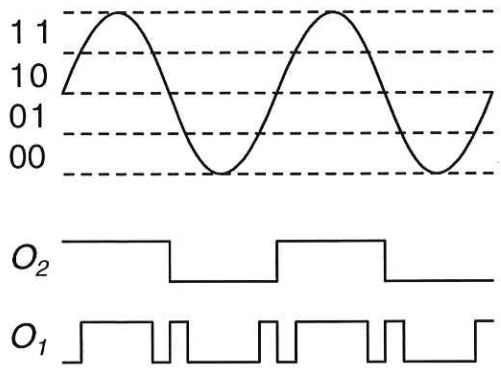


Figure 7.4. Digital output of a 2-bit A/D converter from a sinusoidal waveform. This shows the waveform separated into four levels representing a 2-bit value ie. 11, 10, 01 and 00 which are the values of O_2 and O_1, respectively. After Sarros et al [2004].

power state and vice versa. The output power of the first R-SEED network is then input to the second R-SEED in addition to half the analog input power such that (7.3) is satisfied. This is depicted diagrammatically in Figure 7.3(b).

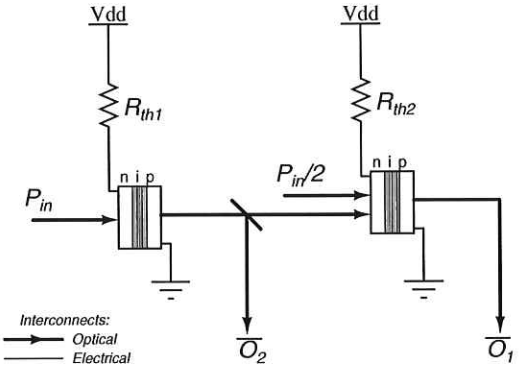


Figure 7.5. 2-bit optical threshold logic ADC. This uses two R-SEED networks where the resistors R_{th1} and R_{th2} determine the thresholds of the devices. After Sarros et al [2004].

7.2.4 Simulations and Experimental Results

This Subsection presents the simulation and experimental measurement results for the 2-bit ADC shown in Figure 7.5. A dynamic analysis is used for the simulations in which circuit equation 7.2 is solved numerically. Experimentally measured data for the device responsivity and reflectivity is used in the simulation. These measurements were obtained by applying incident power of 100 μW and measuring the photo-current

and reflectivity as a function of voltage and wavelength. The dark currents and the parasitic capacitances for each device were also measured with the incident power set to zero. The experimental data for S_1, S_2, I_{d1}, I_{d2}, C_1 and C_2 were substituted into equation 7.2 and solved numerically.

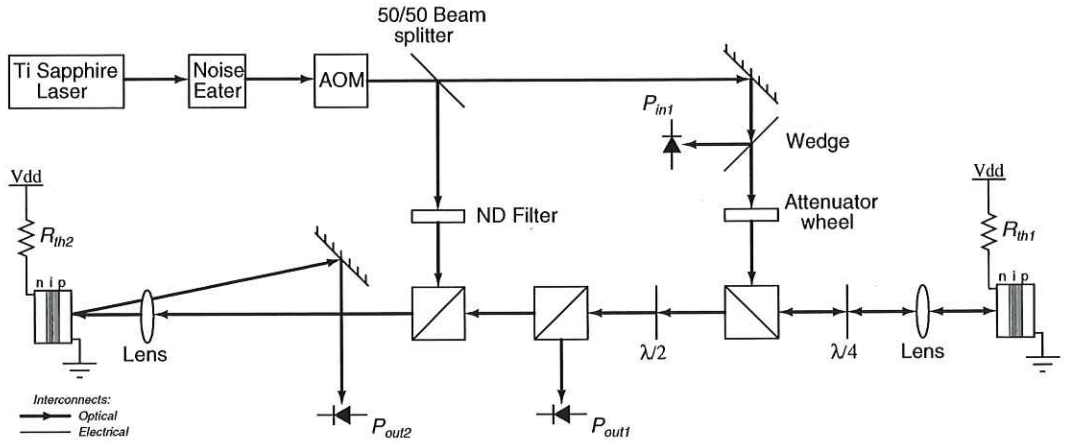


Figure 7.6. Experimental setup for a 2-bit optical threshold logic ADC. This uses two R-SEED networks shown in Figure 7.5. After Sarros et al [2004].

Figure 7.7 compares experimental and simulated results obtained from implementing the architecture shown in Figure 7.5 where the optical sine wave input frequency was 100 Hz. The operating wavelength was chosen to be 846 nm as this gave the highest contrast ratio for both devices. The bias voltage for both R-SEED networks was set at -10 V and the analog power levels on both networks were 100 μW with a 10 μW amplitude, therefore the power ranged from 90 μW to 110 μW.

In Figure 7.7 the digital output of the most significant bit complements the simulated diagrams. For the digital output of the least significant bit the simulation switches sooner from the 01 to 00 transition compared to the experimental data. Comparing to the ideal case, there is a noticeable difference due to the large hysteresis of the R-SEED network.

From simulation and experimental data for the 2-bit optical threshold logic ADC shown in Figure 7.5, it can be observed that the least significant bit has a significant hysteresis problem. One way to minimise hysteresis is with the use of an S-SEED structure such that a reference power is applied or a differential system is employed. A method for eliminating hysteresis using S-SEEDs includes a state-preset pulse operating at a wavelength several nanometers longer than the operating wavelength (Loh

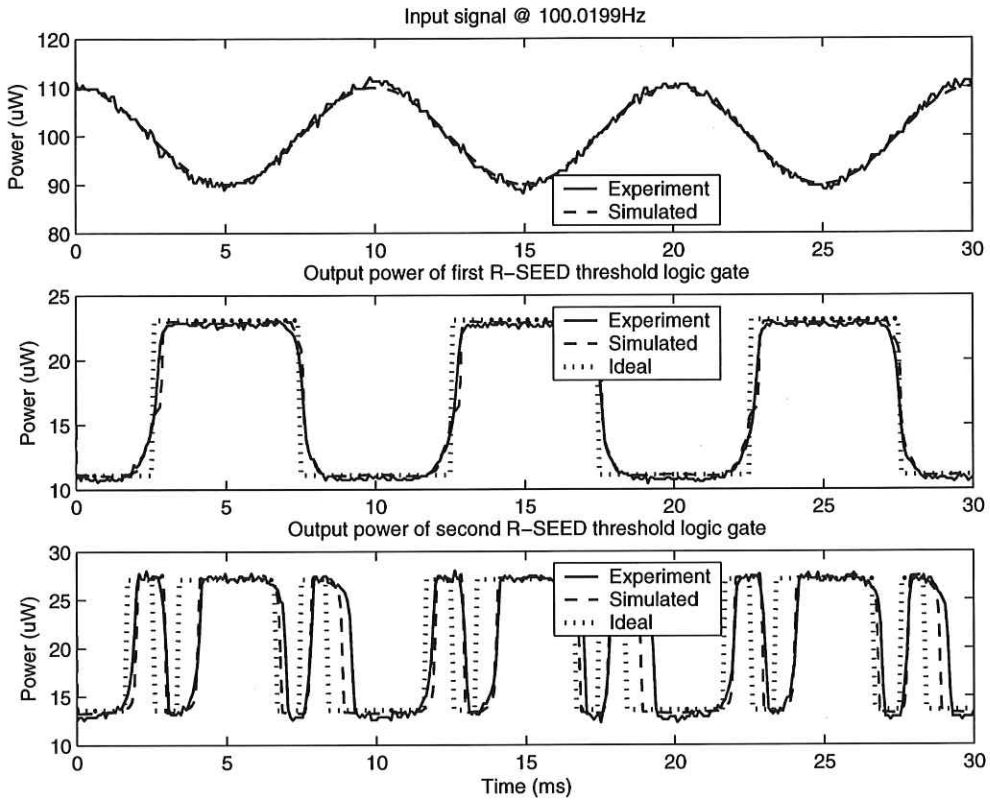


Figure 7.7. Threshold R-SEED results. From top to bottom, the analog input, most significant and least significant bits of the 2-bit optical threshold logic ADC is shown at 100 Hz. The simulated graphs indicates a good representation for predicting the behaviour of the system. The ideal graph for the most significant bit shows the R-SEED networks hysteresis problem. Measurements carried out by Tony Sarros.

and LoCicero 1996). This technique results in a single point of intersection on the I-V curve hence unambiguously determining the state of the S-SEED.

7.3 CGaAs Threshold Logic

As a further exploration of various physical implementations of threshold logic, this Section focuses on Complementary Gallium Arsenide based TL. The techniques presented in this section are closely related to the capacitive MOS TL gate implementations.

The neu-MOS transistor uses capacitively coupled inputs onto a floating gate. Neu-MOS enables the design of conventional analog and digital integrated circuits with a

significant reduction in transistor count (Wong *et al.* 1997, Gonzalez *et al.* 1998). Furthermore, neu-MOS circuit characteristics are relatively insensitive to transistor parameter variations inherent in MOS fabrication processes. Neu-MOS circuit characteristics depend primarily on the floating gate coupling capacitor ratios. It is also thought that this enhancement in the functionality of the transistor, ie. at the most elemental level in circuits, introduces a degree of flexibility which may lead to the realisation of intelligent functions at a system level (Ohmi and Shibata 1995). This Chapter extends the neu-MOS paradigm to complementary gallium arsenide based on HIGFET transistors. The design and HSPICE simulation results of a neu-GaAs ripple carry adder are presented, demonstrating the potential for very significant transistor count and area reduction through the use of neu-GaAs in VLSI design. Preliminary simulations indicate a factor of 4 reduction in transistor count for the same power dissipation as conventional complementary GaAs. The small gate leakage current is shown to be useful in eliminating unwanted charge buildup on the floating gate.

Complementary GaAs has a number of highly desirable properties for low-power, high-speed digital and mixed RF/digital applications. These include low voltage operation (0.9 V to 1.5 V), very low static power dissipation using CMOS-like designs and significantly higher operating speeds than CMOS.

The neuron-MOS transistor (neu-MOS or νMOS for short) was originally developed at Tohoku University in 1991 (Shibata and Ohmi 1991). The structure of a neu-MOS transistor is identical to an ordinary MOS transistor, but with a number of additional inputs capacitively coupled onto a floating gate, as shown in Figure 7.8. MOSFET style transistor symbols have been used to emphasize the semi-insulating nature of the HIGFET transistor gate. The floating gate potential is a weighted sum of the inputs, the weightings being determined by coupling capacitor ratios.

The use of neu-MOS transistors provides additional functionality that allows, for example, the design of a full adder cell with only 8 transistors as compared to 28 in CMOS and an area of 55% of the CMOS design (Hirose and Yasuura 1996).

The goal of this chapter is to extend the neu-MOS paradigm to the complementary GaAs technology. In particular, we demonstrate the suitability of 0.5 μm HIGFET transistors for application to neu-GaAs (Abbott *et al.* 1998), present a basic neu-GaAs circuit structure and the simulation results of a neu-GaAs 4 bit ripple carry adder, for the first time.

The use of neu-GaAs techniques in HIGFET transistor designs promises to give VLSI designers more freedom in designs where area, delay and power dissipation are critical and provides a step forward towards boosting the effective integration level.

7.3.1 Neu-GaAs Basic Structure

The neu-GaAs transistor is shown in Figure 7.8. Although the gate of a HIGFET transistor is not strictly speaking a floating node as in a MOS transistor, the analysis of this structure is identical to that of the neu-MOS transistor given in (Shibata and Ohmi 1991), and the floating gate potential is given by

$$\phi_F = \frac{C_1 V_1 + C_2 V_2 + ... + C_N V_N}{C_T} \tag{7.4}$$

where C_T is the sum of the coupling capacitors C_1 to C_n and C_0. Here C_0 is the sum of all parasitic capacitances from the floating gate to the substrate, including the floating gate to source and drain capacitances.

A basic variable threshold neu-GaAs inverter structure is shown in Figure 7.9. This neu-GaAs inverter is a fundamental building block in digital neu-GaAs design and is similar to an ordinary CMOS like ratioless inverter consisting of a p-type GaAs pull-up and an n-type pull-down transistor. The gates of the two transistors are connected and two inputs which are capacitively coupled to this floating gate are added. When the floating gate potential exceeds the inverter threshold, the inverter output becomes low and vice versa. By using two inputs, one of which is to be inverted, V_{in}, and one as a threshold control, V_{ref}, the effective neu-GaAs inverter threshold (as seen from the input V_{in}) can be made variable. The simulation results for three values of V_{ref} are shown in Figure 7.10.

7.3.2 Choice of GaAs Technology

The realisation of neu-GaAs circuits requires that the floating gate voltage remain stable for periods of time depending on the clock frequency being used. This means that a low gate leakage current is required. The HIGFET uses a semi insulating AlGaAs layer to reduce gate leakage currents (to approximately 2 nA/μm^2 of gate area) and it appears, at least in the short term, as the most viable option for complementary neu-GaAs applications.

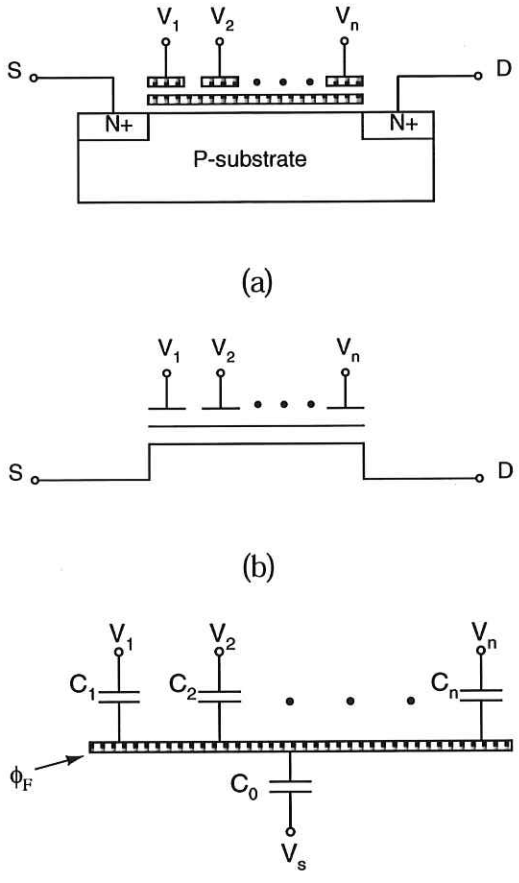


Figure 7.8. Introducing neu-MOS. (a) The structure of an n-type neu-MOS transistor, (b) its electronic symbol, and (c) the capacitance model.

Due to the proprietary nature of the complementary GaAs parameters, Table 7.1 lists the composite set of HSPICE (level 3, JFET) parameters based on a number of complementary GaAs processes, including Honeywell (Fulkerson *et al.* 1996), Sandia (Baca *et al.* 1996), Univ. Lille (Thiery *et al.* 1997) and MIT (Chen *et al.* 1996).

7.3.3 A 4-bit neu-GaAs Ripple Carry Adder

A 4-bit RCA was chosen as a simple circuit to demonstrate the feasibility of circuit design using neu-GaAs. Ripple carry adders have relatively low power dissipation and the delay for computing the final carry depends on the number of bits to be added because the carry propagates successively from the first stage to the last. The basic neu-GaAs full-adder (FA) is implemented based on the following expressions obtained

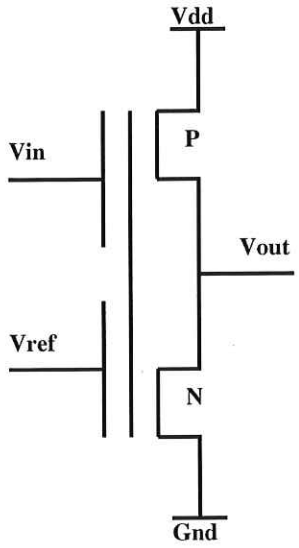


Figure 7.9. Basic neu-GaAs inverter structure .

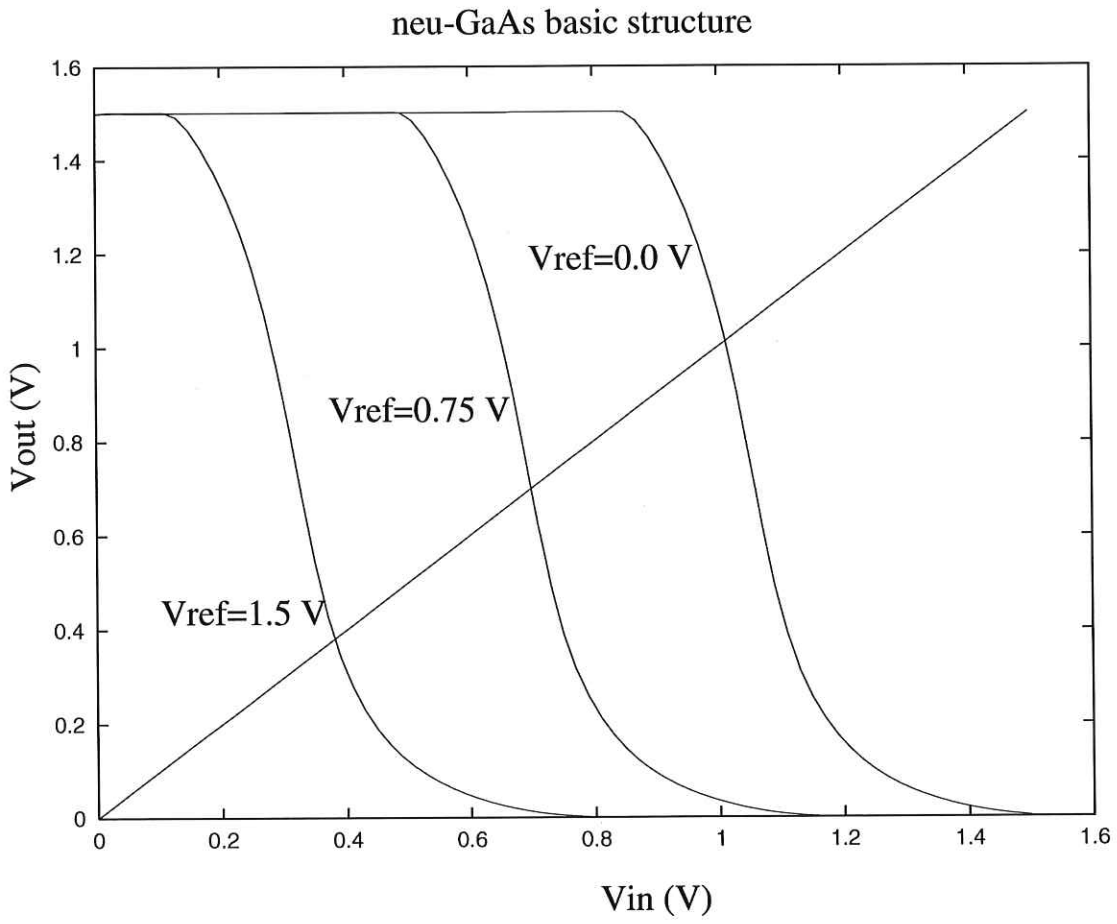


Figure 7.10. Neu-GaAs inverter structure simulation results.

Table 7.1. Composite HSPICE parameters.

Parameter Name	n-type (Range/Value)	p-type (Range/Value)	Units
is	2.4-2.6	2.3-2.6	pA
cgs	5.5-6.0	7.5-8.0	fF
cgd	1.8-2.4	3.7-4.1	fF
vto	0.65	0.52	V
beta	4.1-4.3	0.9-1.1	mA/V^2
lambda	0.13-0.15	0.25-0.27	1/V
alpha	2.2-2.3	2.3-2.4	1/V

from the truth table for addition:

$$c_i = 1 \Longleftrightarrow a_i + b_i + c_{i-1} \geq 2 \tag{7.5}$$

$$s_i = 1 \Longleftrightarrow a_i + b_i + c_{i-1} - 2c_i \geq 1 \tag{7.6}$$

where a_i and b_i are the two bits at the i^{th} position and c_i is the carry generated at the i^{th} position. The + symbol denotes algebraic addition, and a_i, b_i, c_{i-1}, and c_i take values of 0 or 1 corresponding to 0 V and V_{DD} in the actual circuit, respectively.

As there are no negative voltages in the circuit (there is only a 1.5 V supply), $-2c_{i+1}$ must be converted into $2(\overline{c_{i+1}} - 1)$ and hence:

$$c_i = 1 \Longleftrightarrow a_i + b_i + c_{i-1} \geq 2 \tag{7.7}$$

$$s_i = 1 \Longleftrightarrow a_i + b_i + c_{i-1} + 2\overline{c_i} \geq 3. \tag{7.8}$$

It should be noted that in order to compute s_i, $\overline{c_i}$ has to be pre-computed. The neu-GaAs realisation of the two inequality expressions for c_i and s_i is shown on the right hand side of Figure 7.11. The magnitudes of the coupling capacitors for each of the inputs may be found as follows. To evaluate $\overline{c_i}$ from a_i, b_i and c_{i-1}, Equation 7.7 is written in the following form:

$$\overline{c_i} = V_{DD} \Longleftrightarrow \frac{a_i V_{DD} C_1 + b_i V_{DD} C_2 + c_{i-1} V_{DD} C_3}{C_1 + C_2 + C_3 + C_0} \geq \frac{V_{DD}}{2} \tag{7.9}$$

where C_1, C_2, and C_3 are the neu-GaAs structure coupling capacitor magnitudes for the inputs a_i, b_i and c_{i-1}, respectively, and C_0 is as defined in Section 7.3.1. Furthermore, C_0 is assumed to be very small compared the sum of the coupling capacitances C_1, C_2, and C_3. The right hand side of inequality (7.9) is set to $V_{DD}/2$ which is equal to

the threshold voltage of the inverter in the neu-GaAs structure. Comparing inequality (7.9) with (7.5), it becomes clear that by setting $C_1 = C_2 = C_3 = C$, Inequality (7.9) may be rewritten as

$$\overline{c_i} = V_{DD} \Longleftrightarrow \frac{a_i C + b_i C + c_{i-1} C}{3C} \geq \frac{1}{2} \tag{7.10}$$

or equivalently

$$\overline{c_i} = V_{DD} \Longleftrightarrow a_i + b_i + c_{i-1} \geq \frac{3}{2} . \tag{7.11}$$

Since a_i, b_i, c_i and c_{i-1} may take on the values of either 1 or 0, Inequality 7.11 may be written as

$$\overline{c_i} = 1 \Longleftrightarrow a_i + b_i + c_{i-1} \geq 2 . \tag{7.12}$$

Thus the neu-GaAs circuit to evaluate c_i has equal coupling capacitances for each of the three inputs as shown in Figure 7.11. The design of the circuit to evaluate s_i follows a similar procedure.

Figure 7.11 also compares the full adder cell design in both conventional complementary GaAs and neu-GaAs and shows a significant transistor count reduction in the neu-GaAs design.

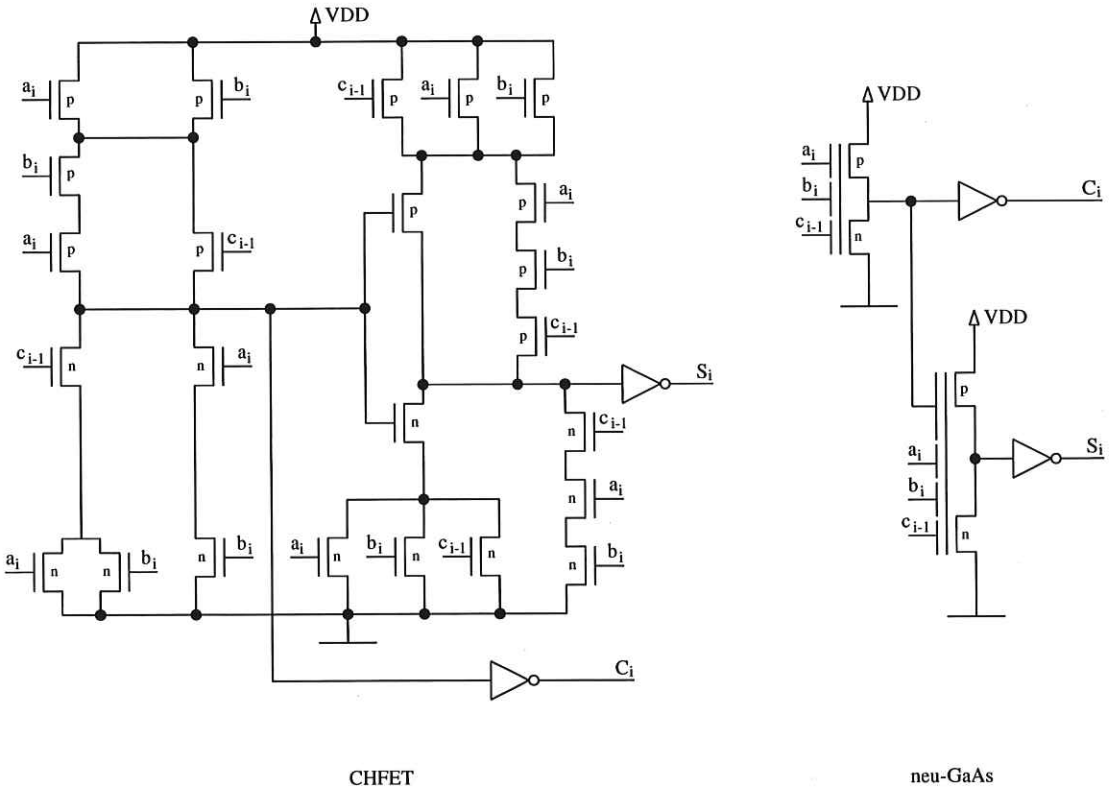


Figure 7.11. Conventional GaAs and neu-GaAs full adder designs.

7.3.4 Simulation results for the neu-GaAs RCA

Figure 7.13 shows the HSPICE simulation results for a single full adder while c_{in} is switched. The graph on the right in Figure 7.13 shows that there is no degradation of the c_{out} output even when c_{in} is maintained high for an extended period of time (25 ns). The coupling capacitors used were 30 fF for all inputs with the exception of the $\overline{c_i}$ intermediate output, which was 60 fF.

The technique used to simulate the RCA structure using HSPICE was as follows. The two input words were set to (a_3 a_2 a_1 a_0) = (0 0 0 0) and (b_3 b_2 b_1 b_0) = (1 1 1 1) as shown in Figure 7.12. The c_{in} was switched from 0 to 1.5 V at a frequency of 200 MHz. This causes the output carry c_3 to switch when the input carry c_{in} propagates through the four bit slices.

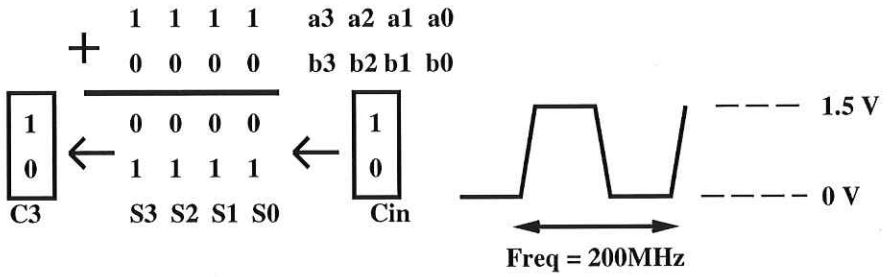


Figure 7.12. Switching of c_3 during the HSPICE simulation.

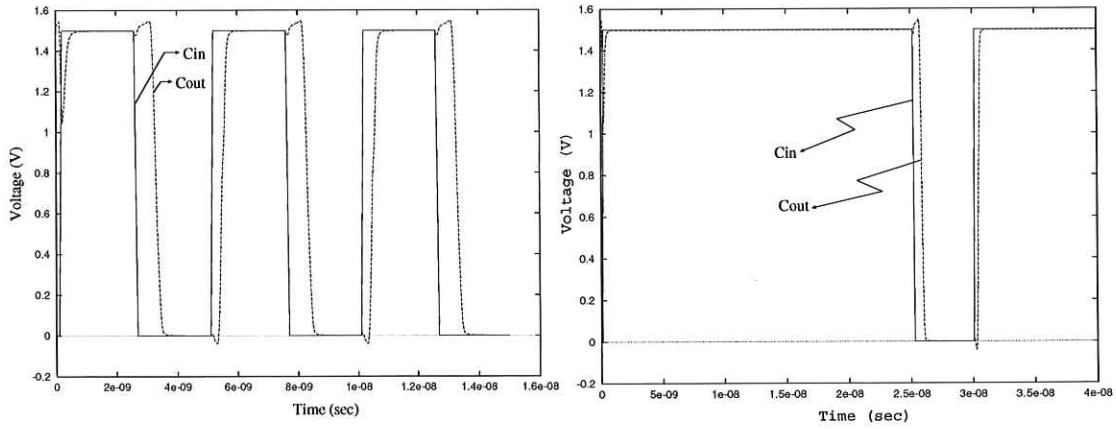


Figure 7.13. C_{out} and C_{in} Simulation for a neu-GaAs full adder.

The propagation delay and power dissipation of the carry signal through both the 4-bit neu-GaAs and the 4-bit conventional complementary GaAs ripple carry adders operating at 200 MHz were then measured as a function of the supply voltage. The

Table 7.2. Simulation Results for a 4-bit RCA.

Adder Type	Power Dissipation	Carry Delay
neu-GaAs (1.5 V)	0.66 mW	1.5 ns
complementary GaAs (1.5 V)	0.68 mW	1.4 ns
neu-MOS (3.3 V)	9.0 mW	3.7 ns
CMOS (3.3 V)	0.34 mW	1.9 ns

results are plotted in Figures 7.14 and 7.15. For a typical supply voltage of 1.5 V, the power dissipation and carry dalay are approximately equal for both the 4-bit neu-GaAs and 4-bit conventional adders based on the designs shown in Figure 7.11.

Table 7.2 shows the carry delay and power dissipation for both the 4-bit neu-GaAs and 4-bit conventional complementary GaAs adders for a supply voltage of 1.5 V. For completeness, the carry delay and power dissipation for 4-bit RCAs designed in neu-MOS and CMOS using a 3.3 V supply voltage have also been included in the table. For comparison purposes, we have used both 0.5 μm CMOS and complementary GaAs.

7.3.5 Gate Leakage

Net charge present on the floating gate after the fabrication process in neu-MOS transistors causes fluctuations in transistor inversion threshold voltage (Shibata and Ohmi 1991). The residual charge on the floating gate can be removed by irradiating the neu-MOS structure with ultra-violet (UV) light. This, however, is an additional step necessary in the construction of neu-MOS circuits. Alternatively, transistor switches connected to the floating gate may be used to set the floating gate potential to a known value and to refresh the floating gate potential periodically.

The proposed neu-GaAs transistor structure does not require a specific procedure to remove residual floating gate charge or floating gate potential initialisation. The reason for this is the presence of a small gate leakage current.

7.4 Chapter Summary

The first part of this Chapter has presented novel ways of digitising signals using optical threshold logic techniques. Using experimental data for SEED device characteristics, a novel 2-bit ADC using optical threshold logic, the Kautz 3:2 counter algorithm

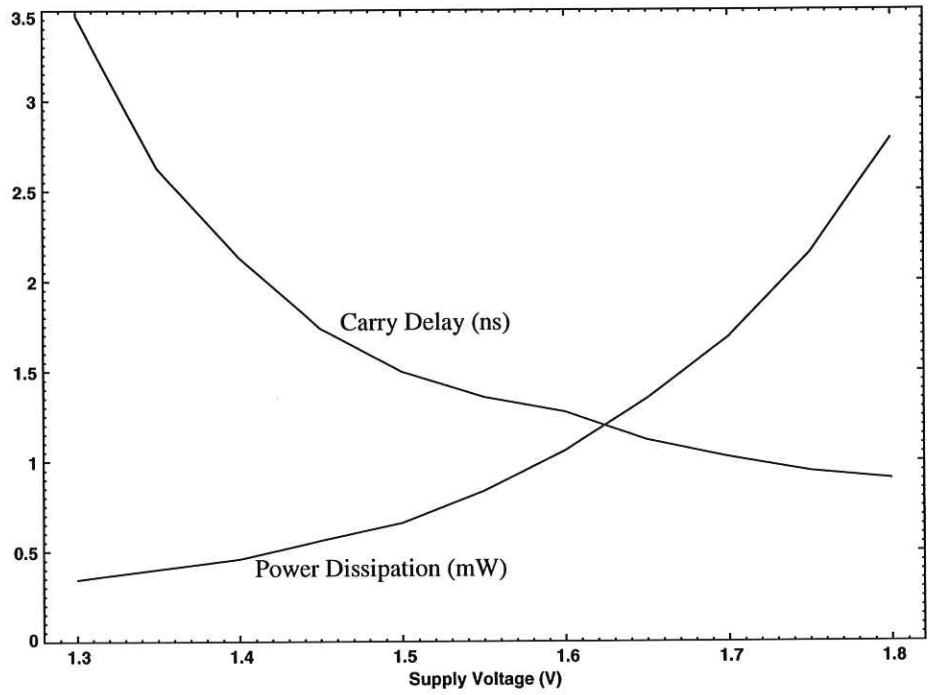


Figure 7.14. Delay and power dissipation vs. supply voltage for a 4-bit neu-GaAs RCA. Vertical axis is Power Dissipation (mW) or Carry Delay (ns).

and two R-SEED networks have been demonstrated. These results may be extrapolated to similar architectures on an optical semiconductor integrated circuit operating at much faster speeds.

The second part of this Chapter has explored complementary neu-GaAs. It has been shown that by using neu-GaAs, a very significant reduction in the number of transistors is attainable over conventional complementary GaAs adder designs despite the presence of a small gate leakage current. Moreover, it is anticipated that the gate leakage removes the need for UV erasure of residual floating gate charge as is required in CMOS. The use of neu-GaAs techniques in HIGFET transistor designs promises to give VLSI designers more freedom in designs where area, delay and power dissipation are critical and provides a step forward towards boosting the effective integration level.

Until this point, the focus has been on theoretical results for specifica arithmetic functions and gate implementations. The following and final Chapter explores the synthesis of TL based networks for arbitrarily specified Boolean functions using a mapping

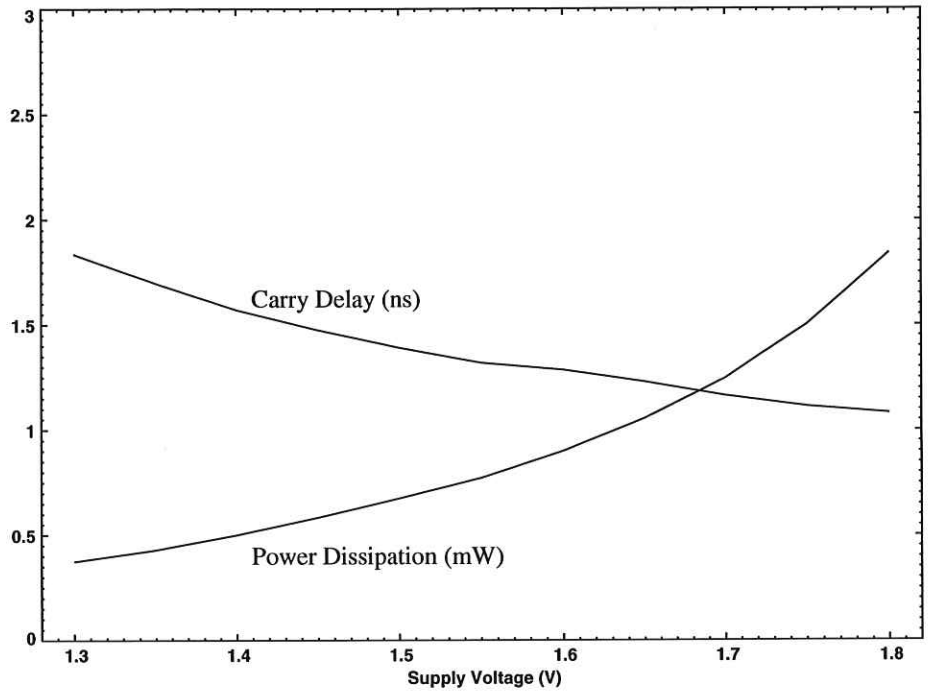


Figure 7.15. Delay and Power Dissipation vs Supply Voltage for a 4-bit conventional GaAs RCA. Vertical axis is Power Dissipation (mW) or Carry Delay (ns).

technique. This is seen as an extension of the well-known Karnaugh map technique commonly used in hand minimization of Boolean logic functions with a small (six or fewer) logic variables.

Chapter 8

Mapping TL Functions of a Small Number of Variables

THIS Chapter focuses on the development of a systematic design technique for implementing Boolean functions of up to 4 variables using threshold logic (TL) gates. The proposed method is a generalization of the standard Karnaugh map logic minimization technique to include Boolean functions which are also threshold functions. The proposed method is illustrated through two worked circuit design examples, using the capacitive neuron-MOS TL technique.

One of the more interesting problems of designing logic circuits using threshold logic is that, unlike for conventional CMOS, there exist no general systematic techniques for the implementation of arbitrary Boolean functions in TL. More precisely, given the truth-table fully specifying a Boolean logic function, there do not exist any techniques of efficiently implementing that function using a TL gate network. Efficiency can be thought of in terms of a minimum gate count realization or a minimum depth network or both, or some other such measure.

The methods which until now have been used to map conventional designs to, for example, neuron-MOS, are somewhat *ad hoc* and only applicable to relatively trivial functions. The aim will be to develop a Karnaugh Map based mapping technique for implementing logic functions in TL, and a version of the design methodology for 3 and 4 variable circuits will be discussed.

Section 8.1 provides a review of Karnaugh-map function minimization, followed by a discussion of how Boolean logic functions may be computed using TL networks in Section 8.2. Section 8.3 outlines the proposed method and two examples of its use are given in Section 8.4. The translation of a TL circuit to the neuron-MOS implementation is illustrated in Section 8.5. Section 8.6 proposes areas for further investigation and a brief conclusion follows in Section 8.7.

8.1 Preliminary Observations

This section begins with an overview of the key aspects of the geometrical representation of Boolean functions, including n-cubes and the operation of Karnaugh map (K-map) logic minimization.

An n-variable or n-input Boolean function can be represented as a cube in n-dimensional Boolean space, where there is one axis for each variable, and each variable can take the values 0 or 1. To map the function onto the n-cube, we assign the value of the function corresponding to the coordinates of a vertex (ie. the value of the input variables) to each vertex of the n-cube. Fig. 8.1 shows the n-cube for 1 (a line), 2 (a square), 3 (a cube), and 4 (a 4-cube) dimensions.

The conventional Karnaugh map logic minimization method provides a means of identifying *adjacency planes* on the n-cube. Each adjacency plane corresponds to a product (AND) term of the function (in sum-of-products form), and an m-dimensional adjacency plane within an n-dimensional cube will produce a term with $n - m$ literals. In other words, the fewer planes (Karnaugh map groupings) the fewer the terms in the function's final (reduced) expression. Also, the higher the dimension of each adjacency plane, the fewer the literals in the product term corresponding to that plane. The minimization process corresponds to finding the minimum cover of the function.

An example is shown in Fig. 8.2 for the full adder carry-out function in both truth table and 3-cube form (black dots on vertices correspond to logic 1's).

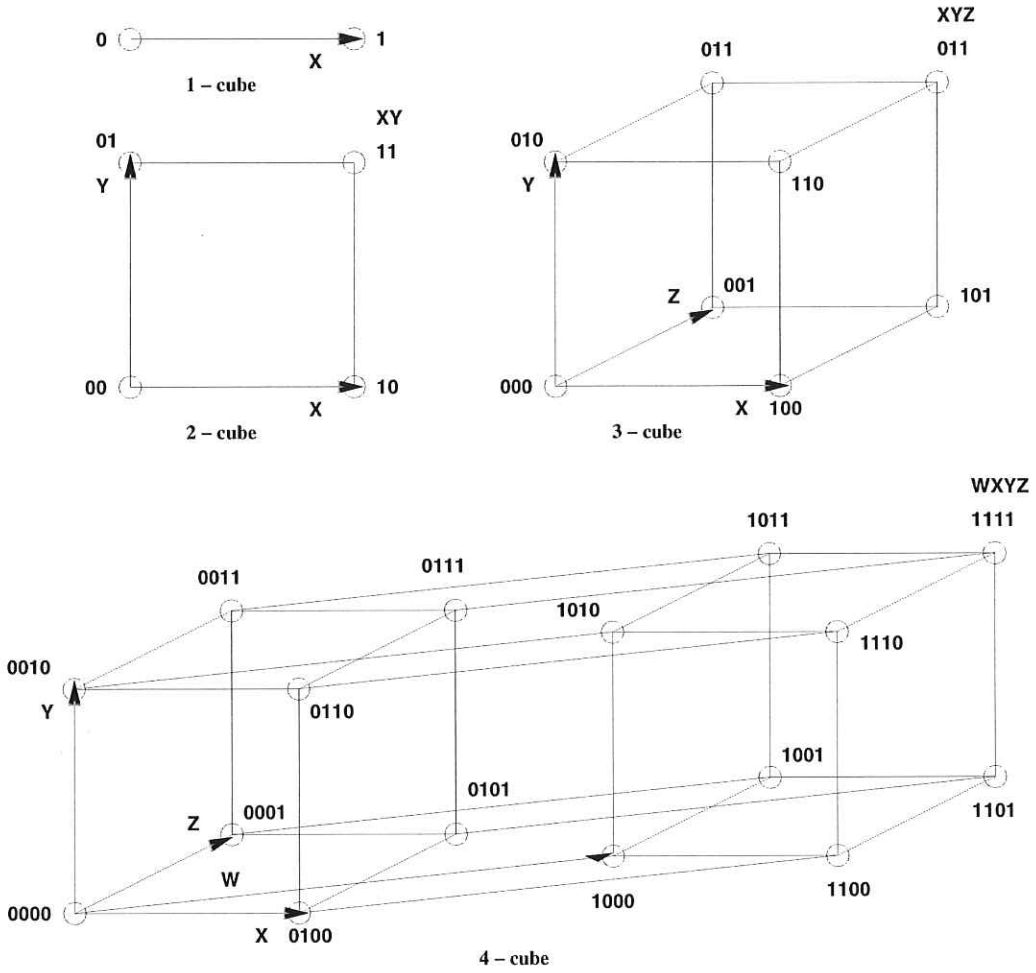


Figure 8.1. N-Dimensional cubes shown for 1, 2, 3 and 4 dimensions.

While the classical K-map minimization technique works by identifying adjacency planes and hence product terms, our technique uses the K-map to identify sets of vertices which are linearly separable from the other vertices on the n-cube and are thus "computable" by one or more threshold functions. Just as there may be more than one product term in the reduced sum-of-products expression, when using a TL function realization there may be more than one plane, and hence threshold function, required to achieve a separation of sets of vertices to compute the given function.

8.2 Computing Boolean Functions Using Threshold Gates

As was mentioned earlier, a depth-2 network of threshold gates suffices to compute any Boolean function. The first layer computes the separation of each cluster of 1's from the remainder of the n-cube, and the second layer performs an OR operation on

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

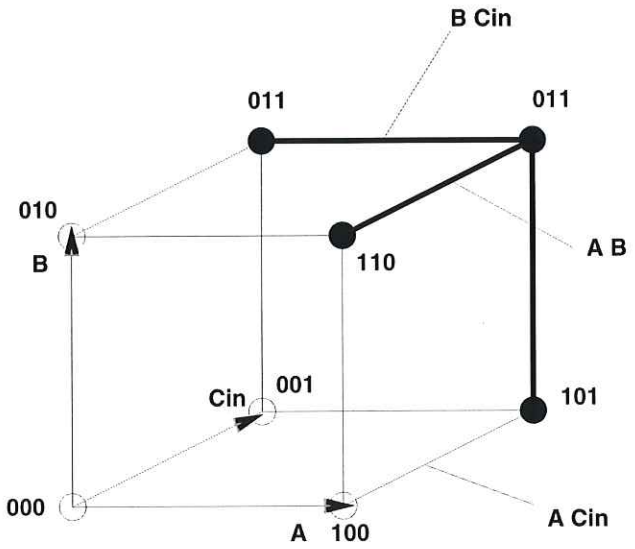


Figure 8.2. Full adder carry-out function shown on a 3-cube and the three adjacency planes (edges) corresponding to the three product terms.

all the outputs from the first layer—using either a conventional CMOS OR gate, or a TL version. A cluster of vertices of 1's which is separated using a single hyper-plane from the rest of the vertices is called a cut-complex (Emamy 1999). The number of such cut-complexes determines how many TL gates are required in the first layer of the network.

To illustrate how a threshold gate operates, we return to the example of Fig. 8.2. This function is computable by a single threshold gate since only one plane is required to achieve a separation of vertices to which are assigned 1's from those to which are assigned 0's. Such a Boolean function is *linearly separable*. An example of a function $(\bar{X}\bar{Y}\bar{Z} + YZ)$ which requires two threshold gates in the first layer and the corresponding planes defined by the TL network required to compute the function are shown in Fig. 8.3.

To create a TL network which is robust and immune to noise it is desirable to achieve cluster separations such that each threshold function defining plane is as far removed from the vertices as possible.

Other techniques which have been developed for the design of logic using TL include the "Floating Gate Potential Diagram" method (Shibata and Ohmi 1993) (which was first reported in a slightly different form by Sheng (1965)), and the linear-programming

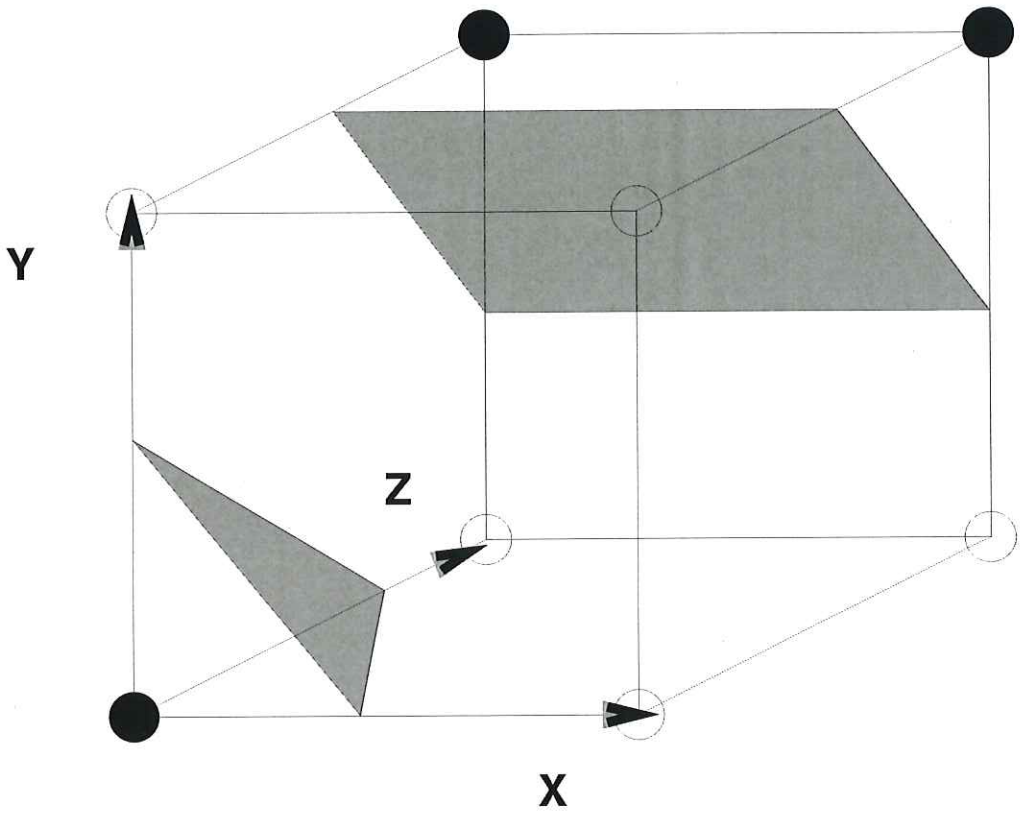


Figure 8.3. An example of a function requiring two threshold gates in the first (input) layer. The two thresold functions are represented by the two shaded planes which seperate the 1's (indicated by filled vertices on cube vertices) from the 0's (indicated by empty circles).

approach (Ike *et al.* 1998). Neither of these methods, however, is well suited to the design of TL circuits by hand, except for relatively trivial functions. One example of a mathematical treatment of the subject is the development of a technique for enumerating linear threshold functions of n-dimensional binary inputs (Ojha 2000).

8.3 A Simple Design Technique

The technique which we have developed consists of a number of steps, and the process is somewhat similar to Karnaugh map minimization. The method will first be outlined, followed by two examples which will seek to clarify the procedure.

1. The first step is to draw the Karnaugh representation of the function to be implemented in a threshold gate network. The 1's are then grouped into shapes,

using the smallest number of shapes. The set of shapes is chosen such that the intersection (or union) of the shapes on the Karnaugh map covers the function precisely. The shapes are chosen from a collection of valid cut-complex shapes, which are shown for up to 4-dimensions in Fig. 8.4. We have termed this process finding the *minimum threshold cover* of the function.

2. If necessary, each of the cut-complex defining shapes from step 1 need to be converted into minimally-weighted cut-complexes. This is done by re-assigning the coordinates such that the shape is effectively centred about the origin (0,0,...,0) in the n-cube. This is achieved by swapping and/or inverting one or more input variables. A function is defined by codewords which represent the input values of the corresponding 1's on the Karnaugh map (ie. codeword 110 implies inputs ABC=110). The codeword weight is the sum of the number of 1's in the codeword and the weight of a shape is the sum of the codeword weights which comprise the shape. A *minimally weighted cut-complex* is one in that the weight of the shape is minimal. A minimally weighted cut-complex results in the plane for that cut-complex being such that the weights which define the plane are positive. This is necessary for implementation, for example, in neuron-MOS.

3. The next step is to determine the planes (threshold function weights) for each minimally weighted cut-complex. This is done by choosing n orthogonal edges (where n is the number of input variables)—the midpoints of these edges then define the plane. The edges must be chosen correctly so that the convex-hull defined by the plane includes all—but no more than the number of—points in the cut-complex.

4. The weight values can then be calculated by solving the matrix equation

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdot\cdot \\ w_n \end{bmatrix} = T \begin{bmatrix} 1 \\ 1 \\ \cdot \\ 1 \end{bmatrix}. \tag{8.1}$$

This may be re-written as

$$A\,\vec{w} = T\,\vec{e} \tag{8.2}$$

solving for $\vec{w}$ we obtain

$$\vec{w} = A^{-1}\,T\,\vec{e} \tag{8.3}$$

where A is the matrix, which has as its rows the coordinates of the edge midpoints that define the plane, $\vec{w}$ is the weight vector and T is the gate threshold. If the sum of the weights is constrained to equal 1, then T is calculated as follows

$$T = ((\vec{e})^t A^{-1} \vec{e})^{-1}. \tag{8.4}$$

5. Finally the network may be constructed by combining the outputs of each threshold gate in the input layer into an OR gate or AND gate, depending on whether the function is defined using union or intersection of the shapes, respectively. The output of this OR or AND gate is the function output.

The short black lines in Fig. 8.4 represent the edges that are used when choosing midpoints. The edges are chosen such that they cut the perimiter of the shape and they correspond to pairs of adjacent positions on the Karnaugh map (adjacent points on the n-cube). For those shapes which are sub-cubes of the n-cube (eg. a square in a 3-cube), one or more input variables is removed. For example, some shapes on the 4-variable Karnaugh map are shown to have less than 4 chosen edges. The inputs removed are those which are not partitioned by a chosen edge.

Two shapes are said to be *isomorphic* if one can be derived from another by a combination of swapping and negation of inputs, as shown in the example of Fig. 8.5. A minimum weighted cut-complex is always isomorphic to its original shape. The corresponding chosen edges are also transformed by such operations. Isomorphic shapes on the Karnaugh map correspond to identical shapes on the n-cube, merely translated and/or rotated. Also, it should be noted that the complements (white shapes) of the shaded shapes in Fig. 8.4 are also valid shapes, with the same chosen edges as the shaded counterparts.

8.4 Two Design Examples

In this Section we illustrate how the proposed method may be used to design the threshold gate network to implement Boolean functions. The first example is of the 3-variable function $Y = \bar{A} + B\bar{C}$, and the second example is of a 2-bit non-equivalence $(A_1 A_0 \neq B_1 B_0)$ function.

Example 1

The Karnaugh map for $Y = \bar{A} + B\bar{C}$ is shown in Fig. 8.6.

Following the process outlined in Section 8.3, the 1's are grouped using the shaded shape shown in Fig. 8.6(a). This is a valid shape because its complement is a valid shape as shown in the fourth 3-dimensional Karnaugh map in Fig. 8.4. The cut complex defined by this shape in its current position on the map is not minimally weighted, and the shape needs to be shifted to that shown in Fig. 8.6(b). The shape is translated vertically on the map by one position to achieve a weight of shape reduction from 6 to 5. The resulting transformation of the inputs is also shown in Fig. 8.6(b). The three orthogonal edges are then chosen as indicated by the short thick lines in Fig. 8.6(b), and the coordinates of these edges are (0.5,1,0), (1,0.5,1) and (0,1,0.5). From these the weight vector may be calculated as follows

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 & 0 \\ 1 & 0.5 & 1 \\ 0 & 1 & 0.5 \end{bmatrix}^{-1} T \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{8.5}$$

Selecting a threshold value $T = 7/10$, the weights become $w_1 = 1/5$, $w_2 = 3/5$, and $w_3 = 1/5$. The AON and TL implementations of this gate are shown in Fig. 8.7.

Example 2

The Karnaugh map for $Y \equiv (A_1 A_0 \neq B_1 B_0)$ is shown in Fig. 8.8.

Following the process outlined in Section 8.3, the 1's are gouped using the minimum number of largest shapes, which in this case is two, one shape for $A < B$ and a second shape for $A > B$. The resulting minimum threshold cover is shown in Fig. 8.8(a) and (b). The shape used is the third to last shape shown on the 4-dimensional Karnaugh map in Fig. 8.4. The cut complexes defined by the two shapes in their current positions on the map are not minimally weighted, and both shapes need to be shifted to that shown in Fig. 8.8(d). The resulting transformation of the inputs for both $A < B$ and $A > B$ is shown in Fig. 8.8(e). The four orthogonal edges are then chosen as indicated by the short thick lines in Fig. 8.8(d), and the coordinates of these edges are (0.5,1,0,1), (0,0.5,1,0), (0,1,0.5,1) and (1,0,0,0.5). The weight vector for both $A < B$ and $A > B$ is the same (but different input variables appear inverted in the two threshold gates as

shown in the input transformation in Fig. 8.8(e)) and may be calculated as follows

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0.5 & 1 & 0 & 1 \\ 0 & 0.5 & 1 & 0 \\ 0 & 1 & 0.5 & 1 \\ 1 & 0 & 0 & 0.5 \end{bmatrix}^{-1} T \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{8.6}$$

As we will see later, it is sometimes desirable that the sum of the weights equal 1, which in this case means we must set the threshold value $T = 7/16$. The weights become $w_1 = 3/8$, $w_2 = 1/8$, $w_3 = 3/8$ and $w_4 = 1/8$. The TL implementation of this gate is shown in Fig. 8.9. The amount of conventional AON logic which this circuit replaces is clearly quite large.

8.5 Mapping the TL Network to neuron-MOS

To map a TL circuit such as that shown in Fig. 8.9 to a neuron-MOS implementation, values for the capacitors used to implement the weights must be found, and the inverter threshold voltage of the primary inverter must be set correctly.

The ratios of the capacitances are chosen to be the same as the weights in Fig. 8.9 since the weights were chosen such that their sum was equal to 1. Using inverters sized for a threshold voltage of $7V_{DD}/16$, the neuron-MOS circuit for the network of Fig. 8.9 is shown in Fig. 8.10.

8.6 Future Work

There are a number of issues related to the proposed method which require further investigation. The first issue relates to the proximity of a threshold function defining plane to vertices. In some cases, especially in higher dimensions, a plane could be oriented such that it passes too close to a vertex to achieve an acceptable level of noise margin. In such cases it would be preferred to replace a single plane with two or more planes of different orientations while still achieving the same n-cube separation. This would require using a non-minimum threshold cover for the function, and a systematic technique for determining when this is required needs to be developed.

The second issue relates to the choice of points to define the threshold function. We have used midpoints in our discussions, and it is relatively simple to prove that a plane defined by n orthogonal edge midpoints for 4 dimensions or less does not pass through a vertex (ie. the threshold function is feasible). It is unknown whether this is also the case in higher dimensions. It would also be worthwhile to determine the bounds on the "dynamic range" of weights determined using edge midpoints.

8.7 Chapter Summary

A methodology was developed for the systematic paper-and-pencil design of 3 and 4 variable threshold gate networks implementing arbitrary Boolean functions. The operation of the proposed design methodology was illustrated by two worked examples. The mapping of a TL network designed using the proposed technique to neuron-MOS based TL gates was shown. This is seen as an extension of the well-known Karnaugh Map design technique commonly used in hand minimization of Boolean logic functions with a small (six or fewer) logic variables.

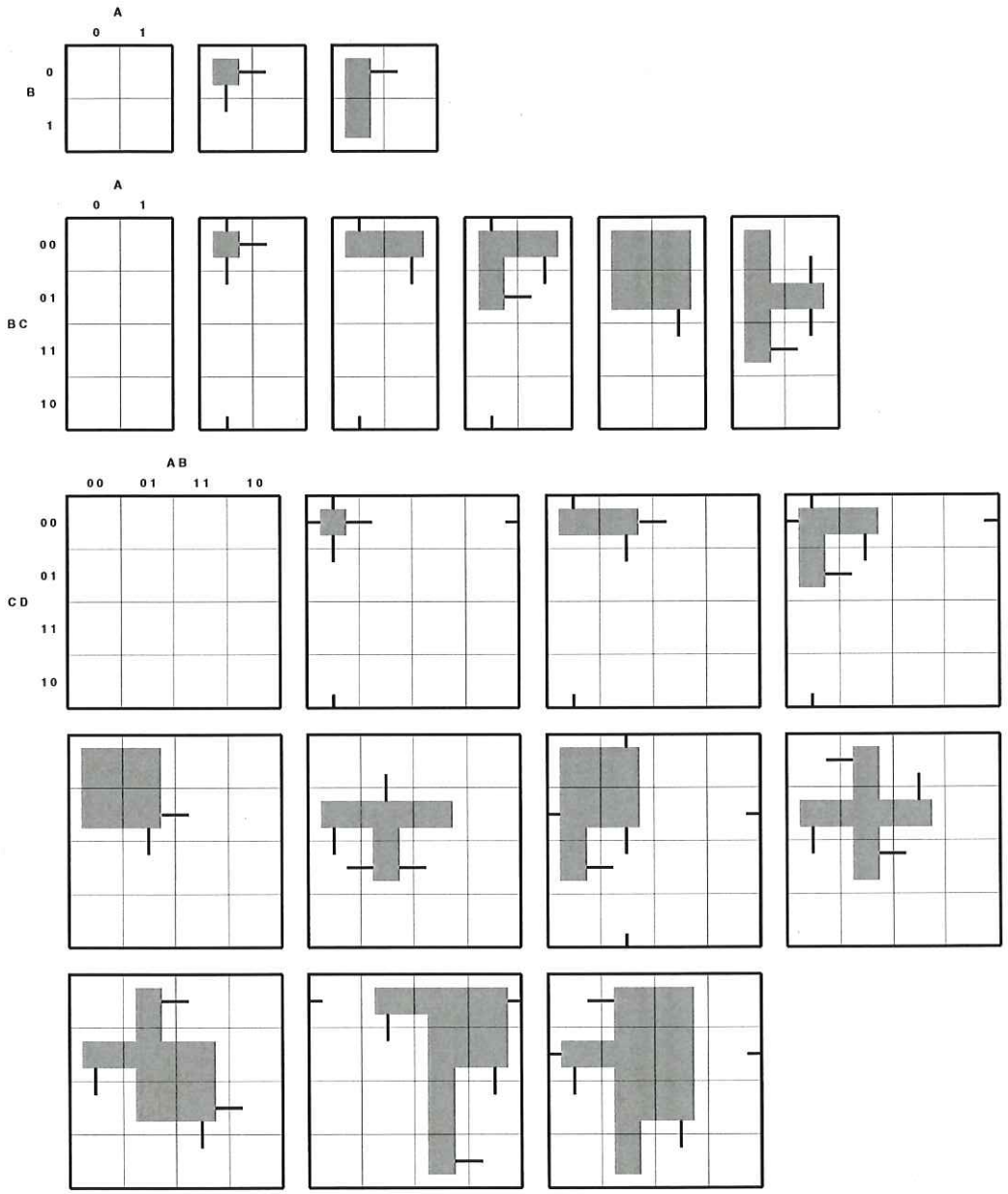


Figure 8.4. All of the Karnaugh map shapes (up to isomorphism) and selected orthogonal edges for possible cut-complexes in 2, 3 and 4-dimensions. The grey shaded regions are groupings of 1s.

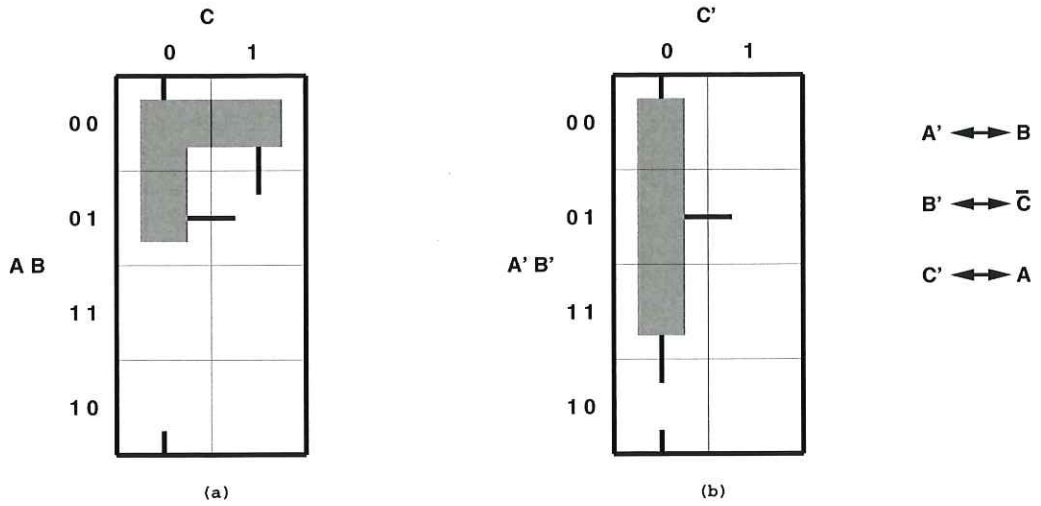


Figure 8.5. (a) Original Shape on Karnaugh map. (b) A shape isomorphic to the shape shown in (a).

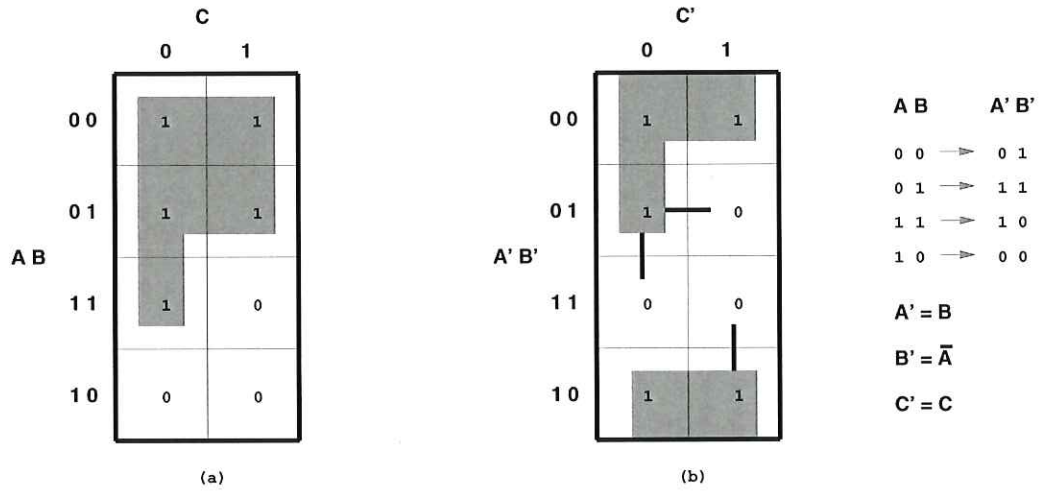


Figure 8.6. (a) Karnaugh map and minimum threshold cover for $Y = \bar{A} + B\bar{C}$ (weight of shape = 6) (b) Corresponding minimally weighted cut-complex (with re-assigned inputs) and selected orthogonal edges (weight of shape = 5).

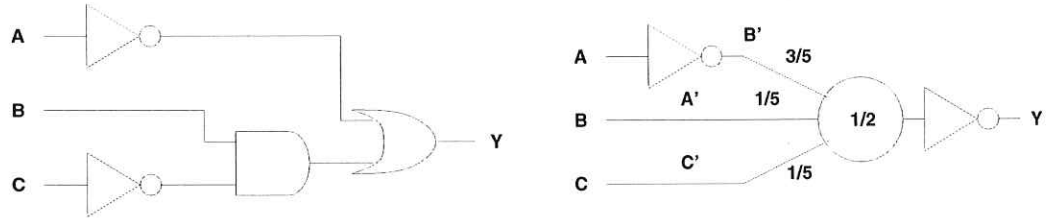


Figure 8.7. The AON and threshold logic implementations of $Y = \bar{A} + B\bar{C}$.

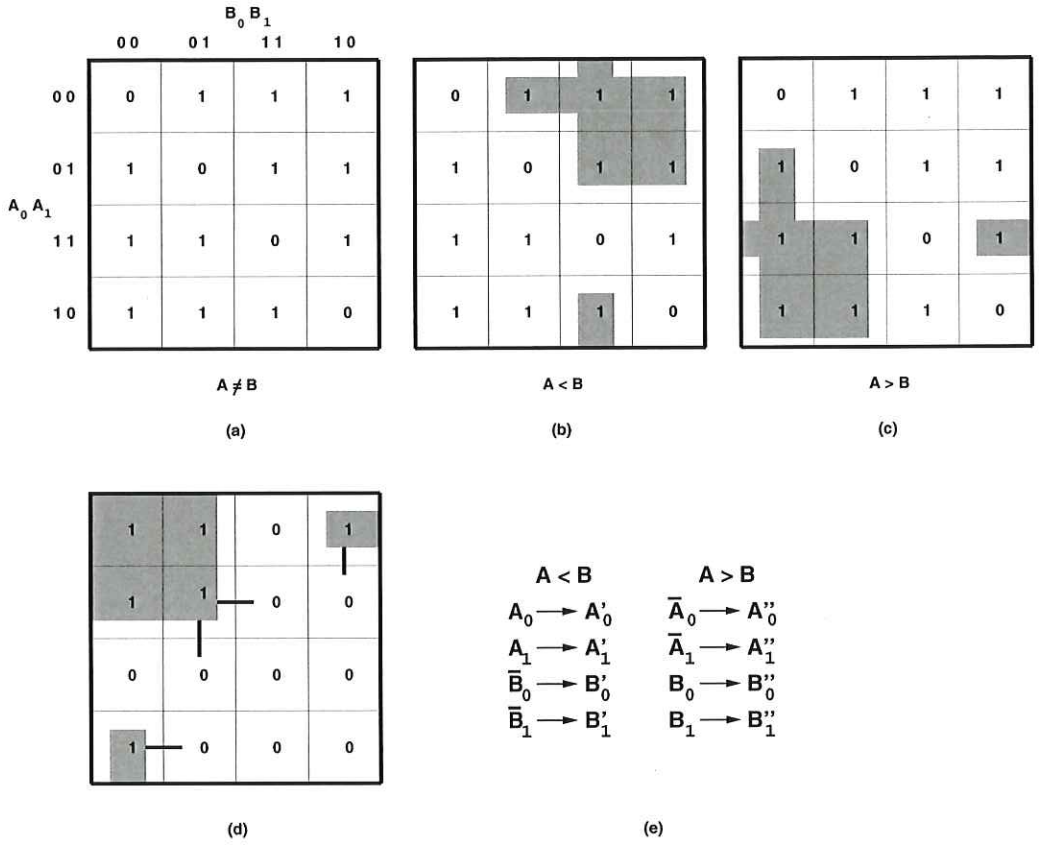


Figure 8.8. (a) Karnaugh map for $Y \equiv (A_1A_0 \neq B_1B_0)$, (b) The minimum threshold cover for $A < B$, (c) The minimum threshold cover for $A > B$, (d) The minimally weighted cut-complex and selected orthogonal edges, (e) The required input re-assignment for both (b) and (c) to obtain (d).

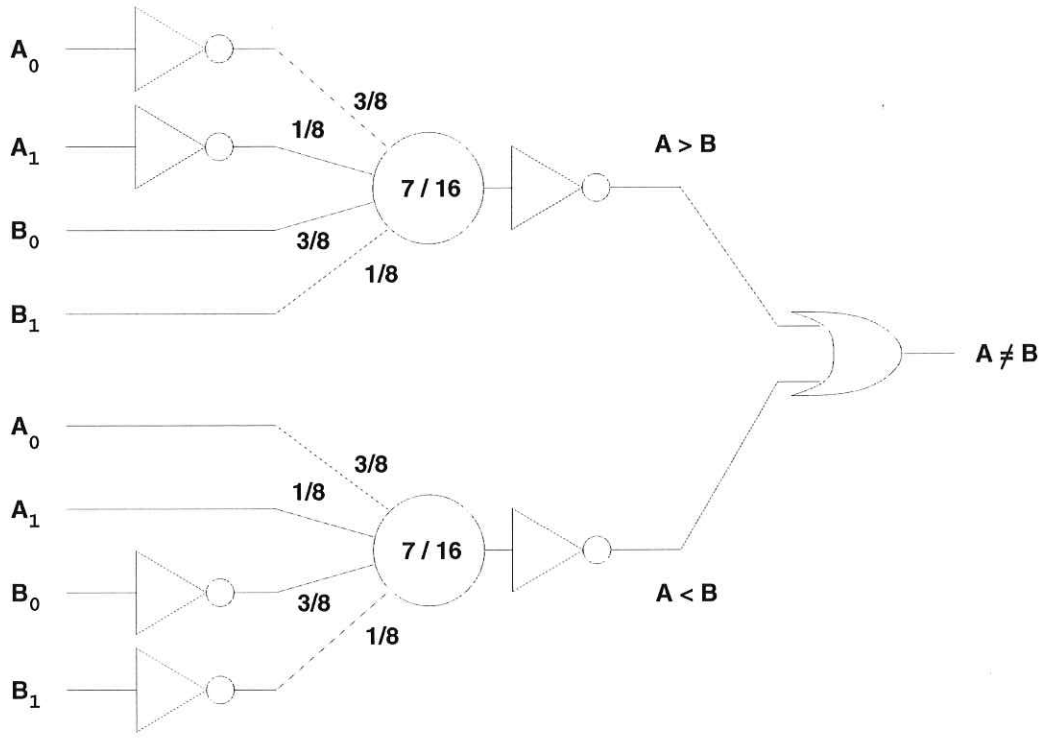


Figure 8.9. The threshold logic implementation of $Y \equiv (A_1 A_0 \neq B_1 B_0)$.

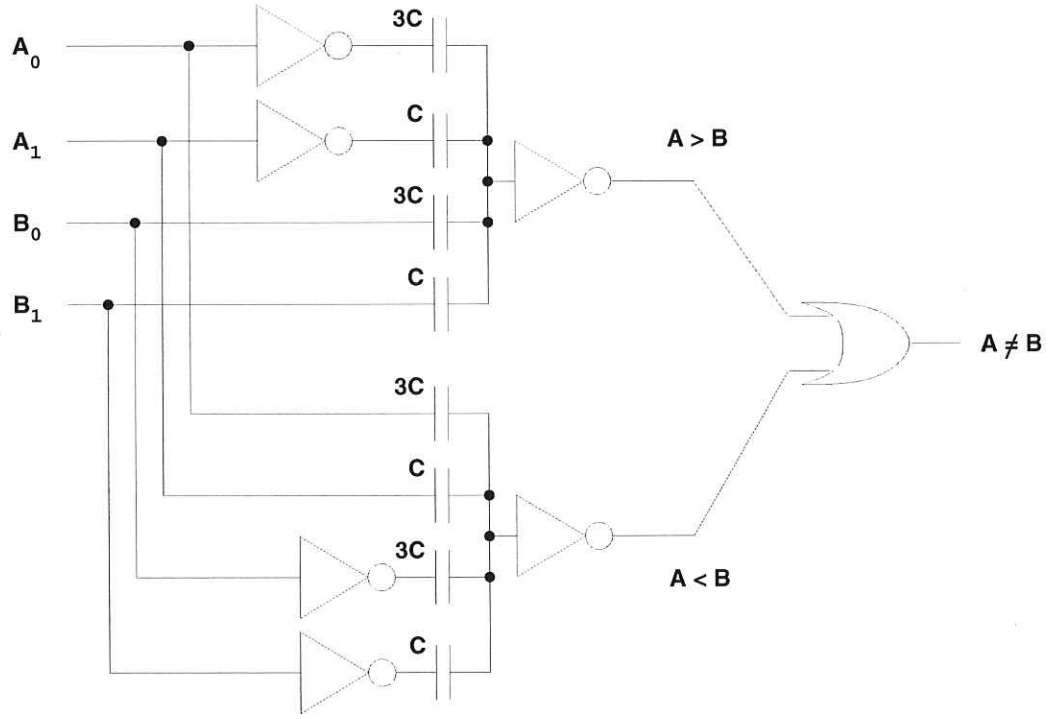


Figure 8.10. The neuron-MOS implementation of $Y \equiv (A_1 A_0 \neq B_1 B_0)$.

Chapter 9

Summary and Conclusions

THIS Chapter draws the final conclusions from the work described in this Thesis and suggests directions for further exploration and research. Until this point, a number of novel contributions to the study of threshold logic have been made, including new TL gate circuits and their delay models, adders, counters, multipliers, optical and GaAs based gates as well as a simple mapping technique. The future of threshold logic is promising, particularly in the application to high-speed datapath design, signal processing as well as compact and low-power circuits.

9.1 Thesis Conclusions

9.1.1 Review of Threshold Logic

Chapter 1 introduced threshold logic and provided motivation for the study of this field. The schematic diagram of the Boolean model of an artificial neuron, namely the linear Threshold Logic (TL) gate, is repeated below in Fig. 9.1 below.

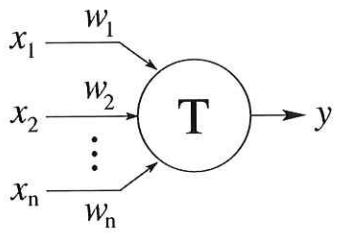


Figure 9.1. Model of the Threshold Logic Gate. The gate shows the outputs y after a thresholding operation T that sums inputs x_1 to x_n after multiplication by weights w_n.

The TL gate computes a neural-like Boolean function of binary inputs. There is strong evidence that TL circuits are more efficient than AON circuits in implementing a number of important functions including arithmetic functions, potentially allowing circuit realizations which require fewer threshold gates than standard logic gates

9.1.2 Review of Threshold Logic Circuits

With regard to silicon implementations of TL gates, of particular importance is sensitivity to transistor mismatch and noise. A device that implements the theoretical model of Fig. 9.1 must compute the linear weighted sum of the binary inputs, store the threshold value and compare the weighted sum to this threshold. The various gate implementations reviewed in Chapter 3 differ in the way they implement the weights, threshold and comparison. They rely on representing each distinct weighted sum of inputs and the threshold level by an analogue voltage or current.

9.1.3 Capacitive Threshold Logic Circuits

Chapter 4 proposes two novel CMOS threshold gate circuit topologies, Charge Recycling Threshold Logic (CRTL) and Self-Timed Threshold Logic (STTL). A logical effort

based delay model is developed, and these gates are shown to exhibit superior delay performance and lower power dissipation compared to any previously proposed threshold gate. Finally, measurements of a test chip fabricated in a 0.25 μm process verify the correct functionality of a range of gates with up to 64 inputs, including the form of the delay model for the CRTL gate.

9.1.4 Threshold Logic Addition

To demonstrate the advantages of the threshold logic design paradigm on a non-trivial circuit problem, two new hybrid CRTL/CMOS-domino adder designs are developed in Chapter 5. The prefix-8 adder design is shown to be over 1 FO4 delay faster than any other 64-bit CMOS adder design proposed to date that does not require multiple non-overlapping clock phases. This is a significant achievement, in light of the tremendous importance of adders in datapath design in conventional microprocessors and DSPs.

9.1.5 Threshold Logic Multiplication

New parallel counters networks and multiplier circuits are developed in Chapter 6, and the proposed partial product reduction schemes are shown to be significantly faster than previously published implementations. A shared input-weight circuit technique is also proposed to significantly reduce the area of the input weight arrays.

9.1.6 Optical and GaAs Threshold Logic

Chapter 7 shows how threshold logic techniques may be applied to technologies other than CMOS, including Complementary Gallium Arsenide (CGaAs) and Self-Electro-Optical (SEED) devices. A novel optical analog-to-digital converter architecture based on SEED TL networks is proposed and test circuit measurements verify the correct functionality of this approach.

9.1.7 Mapping Threshold Logic Functions

The Thesis is concluded in Chapter 8 with a simple mapping technique developed to aid in the implementation of logic functions of a small number of variables, using threshold gates.

The original contributions of this Thesis offer benefits to high performance and low power VLSI processor design and to potential applications of threshold logic in compact low power, portable, wireless devices. Portability is particularly important for aerospace applications, smart sensors and personal communication systems.

9.2 Recommendations for Future Work

A review of the significant recent developments in TL has been presented, including gate implementations and applications in computer arithmetic. The summary of applications has shown that the reviewed TL gates are suitable for the design of high performance digital circuits with reduced area, power dissipation and delay. The advantages of TL are essentially a matter of increased efficiency. Area or power reduction leads to reduced cost, or alternatively, increased functionality for the same power or cost.

Threshold logic, however, continues to remain almost exclusively the subject of research work. A large number of publications claim that a particular TL based circuit or gate reduces power/delay/area by a certain percentage relative to conventional CMOS, and it is likely that more work of this nature will not contribute to the incorporation of TL techniques in industrial applications. The author believs that in order for TL to gain industry acceptance, a number of large scale system designs based on TL must be able to demonstrate significant advantages compared to state-of-the-art conventional CMOS logic designs. This Thesis has presented considerable evidence that a hybrid conventional-logic/TL approach (Celinski *et al.* 2003a, Padure *et al.* 2002a) could help achieve this.

9.3 Summary of Original Contributions

The original contributions represented by this work are discussed in Sec. 1.4. In summary, they include contributions in this Thesis in the area of threshold logic technology lie in (i) threshold gate circuit techniques and (ii) threshold logic network design for important arithmetic operations, including addition and multiplication. The originality of these contributions is evidenced by the list of patents, journal and conference publications on Pages xix to xxii.

1. The work in this Thesis proposed two original CMOS threshold gate circuit topologies, Charge Recycling Threshold Logic (CRTL) and Self-Timed Threshold Logic (STTL). These gates exhibit superior delay performance and lower power dissipation compared to any previously proposed threshold gate (Ch. 3). The CRTL and STTL gates were experimentally verified, and a new weight-sharing circuit technique was proposed, which greatly reduces the circuit area cost of CRTL and STTL based circuits and may be applied to other TL gate designs (Ch. 3).

2. The work presented here has also led to the development of a new, patented level sensitive latch for use in the design of high-speed, compact flip-flop circuits with low internal power dissipation and clock load as well as new, patented analog-to-digital converters based on optical Self-Electro-Optic Devices (SEEDs).

3. Following the development of CRTL and STTL, a delay modelling technique is proposed, based on the principles of Logical Effort, for the systematic evaluation and design for minimal delay of CRTL based circuits. The method developed is applicable to a renge of sense-amplifier based TL gates.

4. To demonstrate the advantages of the threshold logic design paradigm on a non-trivial circuit problem, two new hybrid CRTL/CMOS-domino adder designs are developed. The prefix-8 adder design is shown to be over 1 FO4 (fan-out of four inverter delay) faster than any other 64-bit CMOS adder design proposed to date that does not require multiple non-overlapping clock phases.

5. New parallel counters networks and multiplier circuits were also developed, and the proposed partial product reduction schemes are shown to be significantly faster than previously published.

6. The adder and multiplier circuits shown in this Thesis are the first published large-scale designs based on threshold logic which demonstrate conclusively the advantage of threshold logic over conventional CMOS in arithmetic application.

7. The original contributions of this Thesis offer benefits to high performance and low power VLSI processor design and to potential applications of threshold logic in compact low power, portable, wireless devices.

9.4 Conclusion

This Chapter has given a summary of the major conclusions of this Thesis and presented a number recommendations for future work in the field. As summarized in preceding section, the work in this thesis has led to a number of significant, novel contributions and provides a foundation for further work in the field.

Threshold logic theory has developed considerably during the last five decades and is a mature and well understood field. However, it has only been since the early 1990s that practical TL gate circuit implementations have appeared. There are currently a dozen or more research groups world-wide studying this field, yet many unresolved challanges still remain. The main challenge is the relatively low awareness of TL circuit techniques in the VLSI research and commercial design communities. Other challenges include a lack of commercially available synthesis tools for TL based logic design, including a lack of support for the TL paradigm in hardware description languages. From discussions with insiders in one of the largest microprocessor manufacturers, it appears that there is a reluctance to embrace unconventional and potentially risky design techniques, relying mostly on progress in scaling of process technologies to provide generational performance improvement. This is a well proven approach which has withstood the test of time, however, a point will be reached where unconventional circuit techniques may well find their place in the quest for ever higher performance or lower power dissipation, and TL appears to be a prime candidate.

Bibliography

ABBOTT-D., AL-SARAWI-S., GONZALEZ-B., LÓPEZ-J., AUSTIN-CROWE-J., AND ESHRAGHIAN-K. (1998). Neu-MOS (MOS) circuits for smart sensors and an extension to a novel neu-GaAs (GaAs) paradigm, *IEEE ICECS98*, Vol. 3, Lisbon, pp. 379–404.

ALON-N., AND BRUCK-J. (1994). Explicit constructions of depth-2 majority circuits for comparison and addition, *SIAM Journal on Discrete Mathematics*, **7**(1), pp. 1–8.

AVEDILLO-M., QUINTANA-J., RUEDA-A., AND JIMÉNEZ-E. (1995). Low-power CMOS threshold-logic gate, *IEE Electronics Letters*, **31**(25), pp. 2157–2159.

BACA-A., ZOLPER-J., DUBBERT-D., HIETALA-V., SLOAN-L., SHUL-R., SHERWIN-M., AND HAFICH-M. (1996). Complemetary HFET technology for wireless digital and microwave applications, *in* F. Ren., S. Chu., C. Wu., and S. Pearton. (eds.), *Proc. of the Symposium on High Speed III-V Electronics for Wireless Applications and the 25th State of the Art Program on Compound Semiconductors*, Electrochem. Soc, Pennington, NJ, USA, pp. 73–83.

BEAUMONT-SMITH-A., AND LIM-C. C. (2001). Parallel prefix adder design, *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, Vail, USA, pp. 218–225.

BEIU-V. (1994). *Neural Networks Using Threshold Gates: A Complexity Analysis of Their Area-and Time-Efficient VLSI Implementations*, PhD thesis, Katholieke Univ. Leuven, Belgium.

BEIU-V. (1999). Neural addition and Fibonacci numbers, *Proc. IWANN 99, LNCS 1607*, Vol. 2, pp. 198–207.

BEIU-V. (2003). Constructive threshold logic addition, *oceedings ICANN*, Turkey, pp. 745–752.

BEIU-V., PEPERSTRAETE-J., VANDEWALLE-J., AND LAUWEREINS-R. (1993). Comparison and threshold gate decomposition, *Proc. MicroNeuro 93*, pp. 83–90.

BEIU-V., PEPERSTRAETE-J., VANDEWALLE-J., AND LAUWEREINS-R. (1994a). Addition using constrained threshold gates, *Proc. ConTI 94*, pp. 166–177.

BEIU-V., PEPERSTRAETE-J., VANDEWALLE-J., AND LAUWEREINS-R. (1994b). Area-time performances of some neural computation, *Proc. SPRANN 94*, pp. 664–668.

BEIU-V., QUINTANA-J. M., AND AVEDILLO-M. J. (2003). Vlsi implementations of threshold logic: A comprehensive survey, *IEEE Transaction on Neural Networks*, **14**(5), pp. 1217–1243.

BOBBA-S., AND HAJJ-I. (2000). Current-mode threshold logic gates, *Proceedings of the IEEE International Conference on Computer Design*, Austin, Texas, USA, pp. 235–240.

BRENT-R., AND KUNG-H. (1982). A regular layout for parallel adders, *IEEE Transactions on Computers*, **31**(3), pp. 260–264.

CANNAS-S. (1995). Arithmetic perceptrons, *Neural Computation*, **7**(1), pp. 173–181.

CELINSKI-P., AL SARAWI-S., ABBOTT-D., COTOFANA-S., AND VASSILIADIS-S. (2004). Logical effort based design exploration of 64-bit adders using a mixed dynamic-cmos/threshold-logic approach, *Proc. IEEE Computer Society Annual Symposium on VLSI - Emerging Trends in VLSI Systems Design (ISVLSI'04)*, Lafayette, pp. 127–132.

CELINSKI-P., COTOFANA-S. D., AND ABBOTT-D. (2003a). A-DELTA: A 64-bit high speed, compact, hybrid dynamic-CMOS/Threshold-Logic adder, *Proceedings of the 7th International Work Conference on Artificial and Natural Neural Networks (IWANN 2003)*, Lecture Notes in Computer Science, Spain, pp. 73–80.

CELINSKI-P., COTOFANA-S. D., AND ABBOTT-D. (2003b). Area efficient, high speed parallel counter circuits using charge recycling threshold logic, *Proc. IEEE International Symposium on Circuits and Systems*, Vol. 5, pp. 233–236.

CELINSKI-P., COTOFANA-S., LÓPEZ-J. F., AL-SARAWI-S., AND ABBOTT-D. (2003c). State-of-the-art in CMOS threshold logic VLSI gate implementations and applications, *Proceedings of the VLSI Circuits and Systems Conference*, Vol. 5117, SPIE, Spain, pp. 53–64.

CELINSKI-P., LÓPEZ-J. F., AL-SARAWI-S., AND ABBOTT-D. (2001). Low power, high speed, charge recycling CMOS threshold logic gate, *IEE Electronics Letters*, **37**(17), pp. 1067–1069.

CELINSKI-P., LÓPEZ-J. F., AL-SARAWI-S., AND ABBOTT-D. (2002a). Compact parallel (m,n) counters based on self timed threshold logic, *IEE Electronics Letters*, **38**(13), pp. 633–635.

CELINSKI-P., LÓPEZ-J. F., AL-SARAWI-S., AND ABBOTT-D. (2002b). Low depth carry lookahead addition using charge recycling threshold logic, *Proc. IEEE International Symposium on Circuits and Systems*, Phoenix, pp. 469–472.

CHEN-C., MAHONEY-L., NICHOLS-K., AND BROWN-E. (1996). Self-aligned complementary GaAs MISFETs using a low- temperature-grown GaAs gate insulator, *Electronics Letters*, **32**(4), pp. 407–409.

CILINGIROGLU-U. (1991). A purely capacitive synaptic matrix for fixed-weight neural networks, *IEEE Transactions on Circuits and Systems*, **38**(2), pp. 210–217.

COATES-C., AND LEWIS-P. (1964). DONUT: a Threshold gate computer, *IEEE Transactions on Electronic Computers*, **EC-13**(1), pp. 240–247.

COTOFANA-S. D., AND VASSILIADIS-S. (1998). Periodic symmetric functions, serial addition and multiplication with neural networks, *IEEE Transactions on Neural Networks*, **9**(6), pp. 1118–1128.

DADDA-L. (1965). Some schemes for parallel multipliers, *Alta Freq.*, **34**, pp. 349–355.

DADDA-L. (1980). Composite parallel counters, *IEEE Transactions on Computers*, **C-29**(10), pp. 942–946.

DAO-H., AND OKLOBDZIJA-V. G. (2001). Application of logical effort techniques for speed optimization and analysis of representative adders, *Proc. Thirty-Fifth Asilomar Conferenceon Signals, Systems and Computers, 2001*, Vol. 2, pp. 1666–1669.

DERTOUZOS-M. L. (1965). *Threshold Logic: A Synthesis Approach*, MIT Press, Cambridge, Mass.

EMAMY-W. R. (1999). Geometry of cut-complexes and threshold logic, *Journal of Geometry*, **65**, pp. 91–100.

FANT-K., AND BRANDT-S. (1994). NULL convention logic, US patent 5,305,463 April 19.

FULKERSON-D., BAIER-S., NOHAVA-J., AND HOCHHALTER-R. (1996). Complementary heterostructure FET technology for low power, high speed, digital applications, *Solid State Electronics*, **39**(4), pp. 461–469.

GARCIA-J. L., RAMOS-J. F., AND BOHÓRQUEZ-A. G. (2000). A balanced capacitive threshold logic gate, *Proc. DCICS'2000*, Montpellier, France, pp. 61–69.

GONZALEZ-B., ABBOTT-D., AL-SARAWI-S., HERNANDEZ-A., GARCIA-J., AND LÓPEZ-J. (1998). Efficient transistor count reduction for a low power GaAs A/D converter, *Proceedings XIII Design of Circuits and Integrated Systems conference (DCIS'98)*, Madrid, Spain, pp. 62–66.

HASLER-P., DIORIO-C., MINCH-B. A., AND MEAD-C. (1995). Single transistor learning synapses, *in* G. Tesauro., D. Touretzky., and T. Leen. (eds.), *Advances in Neural Information Processing Systems*, Vol. 7, MIT Press, pp. 817–824.

HEDGE-R., AND SHANBHAG-N. R. (1998). Energy-efficiency in presence of deep submicron noise, *Proceedings of ICCAD*, pp. 228–234.

HIROSE-K., AND YASUURA-H. (1996). A comparison of parallel multipliers with neuron MOS and CMOS technologies, *Proceedings of IEEE Asia Pacific Conference on Circuits and Systems 96*, IEEE, pp. 488–491.

HOLLER-M., TAM-S., CASTRO-H., AND BENSON-R. (1989). An electrically trainable artificial neural network (etann) with 1024 'floating gate' syanpses, *Proceedings of IACNN*, pp. 191–196.

HO-R., MAI-K. W., AND HOROWITZ-M. (2001). The future of wires, *Proceedings of the IEEE*, **89**(4), pp. 490–504.

HOROWITZ-M. (1999). EE271 class notes (adders), Stanford University, http://eeclass.stanford.edu/ee371/.

HOROWITZ-M., YANG-C.-K. K., AND SIDIROPOULOS-S. (1998). High-speed electrical signaling, *Micro, IEEE*, **18**, pp. 12–24.

HUANG-H., AND WANG-T. (2000). CMOS capacitor coupling logic (C^3L) circuits, *Proc. of IEEE Asia Pacific Conference on ASIC*, pp. 33–36.

HU-S. (1965). *Threshold Logic*, University of California Press, Berkeley.

IKE-K., HIROSE-K., AND YASUURA-H. (1998). A module generator of 2-level neuron MOS circuits, *Computers and Electrical Engineering*, **24**(1-2), pp. 33–41.

IMPAGLIAZZO-R., PATURI-R., AND SAKS-M. E. (1997). Size-depth tradeoffs for threshold circuits, *SIAM Journal on Computing*, **26**(3), pp. 693–707.

JOHNSON-M. (1988). A symmetric cmos nor gate for high-speed applications, *IEEE J. Solid-State Circuits*, **23**(10), pp. 1233–1236.

KAUTZ-W. H. (1961). The realization of symmetric switching functions with linear input logical elements, *IRE Transactions on Electronic Computers*, **EC-10**, pp. 371–378.

KNOWLES-S. (1999). A family of adders, *Proceedings 14th IEEE Symposium on Computer Arithmetic*, pp. 30–34.

KONG-B., IM-J., KIM-Y., JANG-S., AND JUN-Y. (1999). Asynchronous sense differential logic, *ISSCC Digest of Technical Papers.*, pp. 284–285.

KOREN-I. (2002). *Computer Arithmetic Algorithms*, A. K. Peters, Natick, MA.

KOTANI-K., SHIBATA-T., IMAI-M., AND OHMI-T. (1995). Clocked-neuron-MOS logic circuits employing auto-threshold-adjustment, *ISSCC Digest of Technical Papers*, pp. 320–321.

KOTANI-K., SHIBATA-T., IMAI-M., AND OHMI-T. (1998). Clock-controlled neuron-mos logic gates, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **45**(4), pp. 518–522.

LASHEVSKY-R., TAKAARA-K., AND SOUMA-M. (1999). The efficiency of neuron-MOS transistors in threshold logic, *Soft Computing*, **3**(1), pp. 20–29.

LAUWEREINS-R., AND BRUCK-J. (1991). Efficient implementation of a neural multiplier, *Proc. 2nd Intern. Conference on Microelectronics for Neural Networks*, pp. 217–230.

LEBLEBICI-Y., ÖZDEMIR-H., KEPKEP-A., AND ÇILINIROĞLU-U. (1996). A compact high-speed (31-5) parallel counter circuit based on capacitive threshold-logic gates, *IEEE JSSC*, **31**(8), pp. 1177–1183.

LEV-L. A., CHARNAS-A., TREMBLAY-M., AND ET AL.. (1995). A 64-b microprocessor with multimedia support, *IEEE Journal of Solid-State Circuits*, **30**(11), pp. 1227–1238.

LEWIS-P. M., AND COATES-C. L. (1967). *Threshold Logic*, Wiley, New York.

LING-H. (1981). High-speed binary adder, *IBM J. Res. Develop.*, **25**(3), pp. 155–166.

LOH-L. M., AND LOCICERO-J. L. (1996). Subnanosecond sampling all-optical analog-to-digital converter using symmetric self-electro-optic effect devices, *Optical Engineering, Society of Photo-Optical Instrumentation Engineers*, pp. 457–466.

LUCK-A., JUNG-S., BREDERLOW-R., THEWES-R., GOSER-K., AND WEBER-W. (2000). On the design robustness of threshold logic gates using multiple-input floating gate MOS transistors, *IEEE Transactions on Electron Devices*, **47**(6), pp. 1231–1239.

MATHEW-S., ANDERS-M., KRISHNAMURTHY-R., AND BORKAR-S. (2002). A 4 GHz 130nm address generation unit with 32-bit sparse-tree adder core, *Symposium on VLSI Digest of Technical Papers*, IEEE, pp. 126–127.

MEAD-C. (1989). *Analog Vlsi and Neural Systems*, Addison-Wesley, chapter Appendix D, pp. 354–355.

MILLER-D. A. B. (1990). Quantum-well self-electro-optic effect devices, *Optical and Quantum Electronics*, **22**, pp. S61–S98.

MILLER-D., CHEMLA-D., DAMEN-T., WOOD-T., BURRUS-C., GOSSARD-A., AND WIEGMANN-W. (1985). The quantum well self-electro-optic effect device: Optoelectronic bistability and oscillation, and self- linearized modulation, *IEEE Journal of Quantum Electronics*, **21**(9), pp. 1462–1476.

MINNICK-R. C. (1961). Linear-Input Logic, *IRE Transactions on Electronic Computers*, **EC-10**, pp. 6–16.

MOORE-G. E. (1965). Cramming more components onto integrated circuits, *Electronics*, pp. 114–117.

MUROGA-S. (1971). *Threshold Logic and Its Applications*, Wiley, New York.

NAFFZIGER-S. (1996). A sub-nanosecond 0.5 μm 64b adder design, *International Solid State Circuits Conference Digest of Technical Papers*, IEEE, pp. 362–363.

NAFFZIGER-S. D., COLON-BONET-G., FISCHER-T., RIEDLINGER-R., SULLIVAN-T. J., AND GRUTKOWSKI-T. (2002). The implementation of the itanium 2 microprocessor, *IEEE Journal of Solid-State Circuits*, **37**(11), pp. 1448–1460.

OHMI-T., AND SHIBATA-T. (1995). Intelligence implementation on silicon based on four-terminal device electronics, *Proc. of 20th Int. Conference on Microelectronics*, Vol. 1, pp. 11–18.

OJHA-P. C. (2000). Enumeration of linear threshold functions from the lattice of hyperplane intersections, *IEEE Transactions on Neural Networks*, **11**(4), pp. 839–850.

OKLOBDZIJA-V. G., VILLEGER-D., AND LIU-S. S. (1996). A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach, *IEEE Transactions on Computers*, **C-45**(3), pp. 294–305.

OKLOBDZIJA-V., ZEYDEL-B., HOANG-D., MATHEW-S., AND KRISHNAMURTHY-R. (2003). Energy-delay estimation technique for high-performance microprocessor VLSI adders, *Proceedings. 16th IEEE Symposium on Computer Arithmetic*, pp. 272–279.

ÖZDEMIR-H., KEPKEP-A., PAMIR-B., LEBLEBICI-Y., AND ÇILINIROĞLU-U. (1996). A capacitive threshold-logic gate, *IEEE JSSC*, **31**(8), pp. 1141–1149.

PADURE-M., COTOFANA-S., AND VASSILIADIS-S. (2002a). High-speed hybrid Threshold-Boolean logic counters and compressors, *Proceedings of the 45th IEEE International Midwest Symposium on Circuits and Systems*, pp. 457–460.

PADURE-M., COTOFANA-S., AND VASSILIADIS-S. (2002b). A low-power threshold logic family, *Proc. IEEE International Conference on Electronics, Circuits and Systems*, pp. 657–660.

PADURE-M., COTOFANA-S. D., AND VASSILIADIS-S. (2003). Design and experimental results of a CMOS flip-flop featuring embedded threshold logic, *Proc. IEEE International Symposium on Circuits and Systems*, pp. 253–256.

PADURE-M., COTOFANA-S. D., BODEA-M., AND VASSILIADIS-S. (1999). Capacitive threshold logic: a designer perspective, *CAS '99 Proceedings*, pp. 81 –84.

PADURE-M., COTOFANA-S. D., VASSILIADIS-S., DAN-C., AND BODEA-M. (2002c). Compact delay model for latch-based threshold logic gates, *Proceedings of the IEEE International Semiconductor Conference*, Vol. 2, pp. 317–320.

PUCKNELL-D. A., AND ESHRAGHIAN-K. (1988). *Basic VLSI Design : Systems and Circuits*, second edn, Prentice-Hall.

QUINTANA-J. M., AVEDILLO-M. J., JIMENEZ-R., AND RODRIGUEZ-VILLEGAS-E. (2001). Practical low-cost cpl implementations of threshold logic functions, *Proc. Great Lakes Symp. VLSI*, West Lafayette, pp. 139–144.

QUINTANA-J. M., AVEDILLO-M. J., RODRIGUEZ-VILLEGAS-E., AND RUEDA-A. (2002). Threshold-logic-based design of compressors, *Proc. International Conference on Electronics, Circuits and Systems*, Vol. 2, pp. 661–664.

RAMOS-J. F., AND BOHÓRQUEZ-A. G. (1999). Two operand binary adders with threshold logic, *IEEE Trans. Computers*, **48**(12), pp. 1324–1337.

RAMOS-J. F., LOPEZ-J. A. H., MARTIN-M. J., TEJERO-J. C., AND GAGO-A. (1998a). A threshold logic gate based on clocked coupled inverters, *International Journal of Electronics*, **84**(4), pp. 371–382.

RAMOS-J. F., TEJERO-J. C., HIDALGO-J. A., MARTIN-M. J., AND GAGO-A. (1998b). Two operand binary adders with threshold Logic, *Proceedings DCICS'98*, Madrid.

RODRIGUEZ-VILLEGAS-E., AVEDILLO-M., QUINTANA-J., HUERTAS-G., AND RUEDA-A. (2000). A *v*MOS based sorter for arithmetic applications, *VLSI Design*, **11**(2), pp. 129–136.

RODRIGUEZ-VILLEGAS-E., HUERTAS-G., AVEDILLO-M. J., QUINTANA-J. M., AND RUEDA-A. (2001). A practical floating-gate muller-c element using vMOS threshold gates, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **48**(1), pp. 102–106.

RUIZ-G. A. (1996). Compact four bit carry look-ahead CMOS adder in multi-output DCVS logic, *IEE Electronics Letters*, **32**(17), pp. 1556–1557.

SCHULTZ-K. J., FRANCIS-R. J., AND SMITH-K. C. (1990). Ganged CMOS: Trading standby power for speed, *IEEE JSSC*, **25**, pp. 870–873.

SHENG-C. L. (1965). Graphical interpretation of realization of symmetric boolean functions with threshold elements, *IEEE Transactions on Electronic Computers*, **v EC-14**, pp. 8–18.

SHENG-C. L. (1969). *Threshold Logic*, Academic Press, New York.

SHIBATA-T., AND OHMI-T. (1991). An intelligent MOS transistor featuring gate-level weighted sum and threshold operations, *IEDM, Technical Digest*, IEEE, New York, NY, USA, pp. 919–922.

SHIBATA-T., AND OHMI-T. (1992). A functional MOS transistor featuring gate-level weighted sum and threshold operations, *IEEE JSSC*, **39**, pp. 1444–1455.

SHIBATA-T., AND OHMI-T. (1993). Neuron MOS binary-logic integrated circuits - part 1 design fundamentals and soft-hardware logic circuit implementation, *IEEE Transactions on Electron Devices*, **40**(3), pp. 570–575.

SIA. (2001). *Summary of the International Technology Roadmap for Semiconductors*, Semiconductor Industry Association. Web Reference *www.sia-online.org*.

SIU-K., AND BRUCK-J. (1990). Neural computation of arithmetic functions, *Proceedings of the IEEE*, **78**(10), pp. 1669–1675.

SIU-K., ROYCHOWDHURY-V., AND KAILATH-T. (1991). Depth-Size tradeoffs for neural computation, *IEEE Trans. Computers*, **40**(12), pp. 1402–1412.

SONG-P. J., AND MICHELI-G. D. (1991). Circuit and architecture tradeoffs for high-speed multiplication, *IEEE Journal of Solid State Circuits*, **26**, pp. 1184–1198.

SOUZA-E., CARRARESI-L., BOYD-G., AND MILLER-D. (1994). Self-linearized analog differential self electro-optic-effect device, *Applied Optics*, **33**(8), pp. 1492–1497.

STOKMAN-A., COTOFANA-S., AND VASILLIADIS-S. (1998). A versatile threshold logic gate, *CAS '98 Proceedings*, pp. 163–166.

SUN-S., MCMURCHIE-L., AND SECHEN-C. (2001a). A high-performance 64-bit adder implemented in output prediction logic, *Proceedings Conference on Advanced Research in VLSI*, pp. 213–222.

SUN-X., MAO-Z., LAI-F., AND YE-Y. (2001b). A high speed 0.25μm 64-bit CMOS adder design, *Proceedings of ASIC*, IEEE, pp. 581–583.

SUTHERLAND-I., AND SPROULL-B. (1991). Logical effort: Designing for speed on the back of an envelope, *in* C. H. Sequin. (ed.), *Proceedings of the 1991 University of California Advanced Research in VLSI Conference*, MIT Press, pp. 1–16.

SUTHERLAND-I. E., SPROULL-R. F., AND HARRIS-D. L. (1999). *Logical Effort, Designing Fast CMOS Circuits*, Morgan Kaufmann.

SWARTZLANDER-E. E. (1973). Parallel counters, *IEEE Transactions on Computers*, **C-22**, pp. 1021–1024.

THIERY-J., FAWAZ-H., BOURZGUI-N., NUYEN-L., AND SALMER-G. (1997). Temperature dependence of pseudomorphic complementary HIGFET devices electrical characteristics, *in* Shur., and Suris. (eds.), *Proc. of the 23rd Inter. Symposium on Compound Semiconductors*, IOP Publishing, UK, pp. 523–526.

VASSILIADIS-S., COTOFANA-S., AND BERTELS-K. (1996). 2-1 addition and related arithmetic operations with threshold logic, *IEEE Trans. Computers*, **45**(9), pp. 1062–1067.

VASSILIADIS-S., COTOFANA-S., AND HOEKSTRA-J. (1995). Block save addition with threshold logic, *Proc. of 29th ASILOMAR Conference on Signals, Systems, and Computers*, pp. 575–579.

WEGENER-I. (1991). The complexity of the parity function in unbounded fan-in, unbounded depth circuits, *Theor. Comput. Sci.*, **85**(1), pp. 155–170.

WESTE-N. H. E., AND ESHRAGHIAN-K. (1995). *Principles of CMOS VLSI Design*, second edn, Addison Wesley.

WINDER-R. (1963). Bounds on threshold gate realizability, *IRE Transactions on Electronic Computers*, **EC-12**, pp. 561–564.

WONG-L., KWOK-C. Y., AND RIGBY-G. A. (1997). 0.9 V 5 MS/S CMOS D/A converter with multiple-input floating-gate MOS, *Proc. of the 1997 IEEE Custom Integrated Circuits Conference*, Santa Clara, CA, USA, pp. 305–308.

WOO-R., LEE-S.-J., AND YOO-H.-J. (2000). A 670 ps, 64bit dynamic low-power adder design, *Proceedings of the International Symposium on Circuits and Systems*, IEEE, Switzerland, pp. I28–I31.

YU-X. Y., OKLOBDZIJA-V. G., AND WALKER-W. W. (2001). Application of logical effort on design of arithmetic blocks, *Proceedings of 35th Annual Asilomar Conference on Signals, Systems and Computers*, pp. 872–874.

ZIMMERMANN-R., AND FICHTNER-W. (1997). Low-power logic styles: CMOS versus pass-transistor logic, *IEEE Journal of Solid State Circuits*, **32**(7), pp. 1079–90.

Biography

Peter Celinski graduated from The University of Adelaide with a Bachelor in Engineering (Electrical and Electronic) in 1998. He commenced a PhD under under the supervision of Professor Derek Abbott (The Centre for Biomedical Engineering) at The University of Adelaide in 1999.

He has been awarded numerous scholarships and stipends, including the D. R. Stranks Travelling Fellowship in 2002, IEEE SA Section Travel Scholarship in 2002 and 2003, SPIE Travel Scholarship in 2002, The University of Adelaide Research Abroad Scholarship in 2001, the Australian Postgraduate Award from 2000 to 2002, and the Sir Ross and Sir Keith Smith Scholarship in 1999.

He has been an invited researcher at a number of international institutions, including the Research Institute for Applied Microelectronics, Universidad de Las Palmas de G.C., Spain, The University of Oxford, UK, The University of Portland, USA, and the Delft University of Technology, The Netherlands, where he was a visiting Research Fellow in Professor S. Vassiliadis' Computer Engineering Group during 2003.

Peter has authored and co-authored more than 25 peer-reviewed publications and his work is regularly cited in journals and conference proceedings dealing with threshold logic. He has given more than ten international conference presentations including an invited presentation at the 2003 SPIE VLSI Circuits and Systems Conference in Gran Canaria, Spain, and one keynote presentation at the 2003 ProRISC Workshop on Circuits, Systems and Signal Processing in the Netherlands.

He has been a member of the IEEE since 1999 and was a founding member of The University of Adelaide student chapter of the IEEE. In 2003 he co-founded Avega Systems Pty Ltd (formerly Gecko Audio Pty Ltd), a venture-capital backed technology startup, which now employs more than 25 engineers with offices in Sydney and San Francisco. Avega is developing the next generation of networked home entertainment products.

Peter Celinski's Scientific Genealogy			
1774	MA	University of Cambridge	John Cranke
1782	MA	University of Cambridge	Thomas Jones
1811	MA	University of Cambridge	Adam Sedgwick
1830	MA	University of Cambridge	William Hopkins
1857	MA	University of Cambridge	Edward John Routh
1868	MA	University of Cambridge	John William Strutt (Lord Rayleigh)
1883	MA	University of Cambridge	Joseph John Thomson
1903	MA	University of Cambridge	John Sealy Townsend
1923	DPhil	University of Oxford	Victor Albert Bailey
1948	MSc	University of Sydney	Ronald Ernest Aitchison
1964	PhD	University of Sydney	Peter Harold Cole
1980	PhD	University of Adelaide	Kamran Eshraghian
1995	PhD	University of Adelaide	Derek Abbott
2006	PhD submitted	University of Adelaide	Peter Celinski