LEARNING, PROBABILISTIC, AND ASYNCHRONOUS TECHNOLOGIES FOR AN ULTRA EFFICIENT DATAPATH

A Thesis
Presented to
The Academic Faculty

by

Bo Marr

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2009

LEARNING, PROBABILISTIC, AND ASYNCHRONOUS TECHNOLOGIES FOR AN ULTRA EFFICIENT DATAPATH

Approved by:

Professor Abhijit Chatterjee,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor David V. Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Paul Hasler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Maysam Ghovanloo
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Tom Conte
Department of Computer Science
Georgia Institute of Technology

Date Approved: 11 November 2009

To my wife Emily Marr,

who put up with all my shenanigans and helped me through this long but rewarding process.

ACKNOWLEDGEMENTS

I want to thank my advisors, my committee, and all my colleagues at Georgia Tech who made my academic and professional growth possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A novel microarchitecture and circuit design techniques are presented for an asynchronous datapath that not only exhibits an extremely high rate of performance, but is also energy efficient. A 0.5 um chip was fabricated and tested that contains test circuits for the asynchronous datapath. Results show an adder and multiplier design that due to the 2-dimensional bit pipelining techniques, speculative completion, dynamic asynchronous circuits, and bit-level reservation stations and reorder buffers can commit 16-bit additions and multiplications at 1 giga operation per second (GOPS). The synchronicity simulator is also shown that simulates the same architecture except at more modern transistor nodes showing adder and multiplier performances at up to 11.1 GOPS in a commerically available 65 nm process. When compared to other designs and results, these prove to be some of the fastest if not the fastest adders and multipliers to date. The chip technology also was tested down to supply voltages below threshold making it extremely energy efficient. The asynchronous architecture also allows more exotic technologies, which are presented. Learning digital circuits are presented whereby the current supplied to a digital gate can be dynamically updated with floating gate technology. Probabilistic digital signal processing is also presented where the probabilistic operation is due to the statistical delay through the asynchronous circuits. Results show successful image processing with probabilistic operation in the least significant bits of the datapath resulting in large performance and energy gains.

CHAPTER I

INTRODUCTION

As we enter a unique age in the development of computing where the straight-forward path of simple transistor scaling to create performance and efficiency gains is coming to an end, this dissertation presents research that shows novel microarchitectural and circuit design techniques that continue to allow leaps in *sequential* computing performance, which is the bottle neck to single application performance speedups shown by Ahmdal's law [106]. This work not only makes a case for the technologies presented, which are independent of technology size and logic family, but also for the continuing importance of good circuit designers in our engineering community.

This dissertation will focus on a novel chip that was fabricated and tested and is built with completely *deterministic* asynchronous technology. One of the technologies introduced is a 2-dimensional bit-level pipelining scheme only possible in the asynchronous domain that allows unprecedented performance gains. This architecture is also fully functional and tested at subthreshold voltage levels, making it extremely energy efficient when needed. The subthreshold to nominal supply range of operation with no additional overhead circuitry is also only possible with asynchronous technology. Results will show a chip built in 0.6 μm AMI technology whereby the multiplies and adds in the datapath run as fast as 1 GHz [77]. As a comparison, no other silicon results in this technology show speeds faster than 370 MHz for full N-bit additions and multiplications [83, 114]. The chip is laid out and simulated in 180 nm and 65 nm technology which show even faster results presenting an argument for the most efficient datapath architecture available.

Probabilistic CMOS concepts are introduced and used to *motivate* a new paradigm

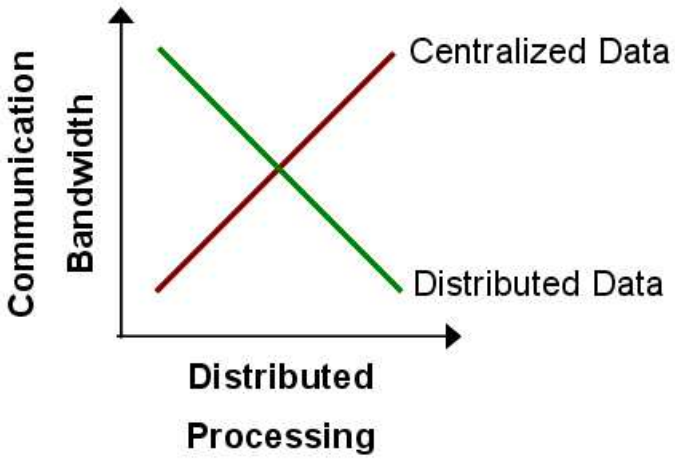


Figure 1: Distributed and parallel processing are a popular trend to increase performance in computing. However, for single threads with centralized data, we see the communication bandwidth increases with the amount of parallel processing and becomes a bottleneck to improving single threaded performance.

implemented on chip called *Speculative Completion* which uses the principles developed during PCMOS research to improve the performance of asynchronous, delay-based technology. The author also gives evidence of why thermally based and synchronously based PCMOS techniques are not preferred methods to achieve computing gains.

Finally, a second 0.35 μm chip is presented that introduces floating gate technology to further supplement the asynchronous architecture and ushers in a new paradigm of *Learning* Digital Circuits [76].

1.1 Chip-Level Parallelism Can Only Take Us So Far

Many interesting trends start to emerge in computing as we enter this era where we see the "Death" of Moore's Law in regards to frequency scaling with transistor size. One such trend is obviously parallel computing and distributed processing, however this trend has its own problems as shown in Figure 1.

The fact that communication bandwidth becomes a bottleneck to single-threaded performance in distributed computing paradigms becomes a huge problem as we know

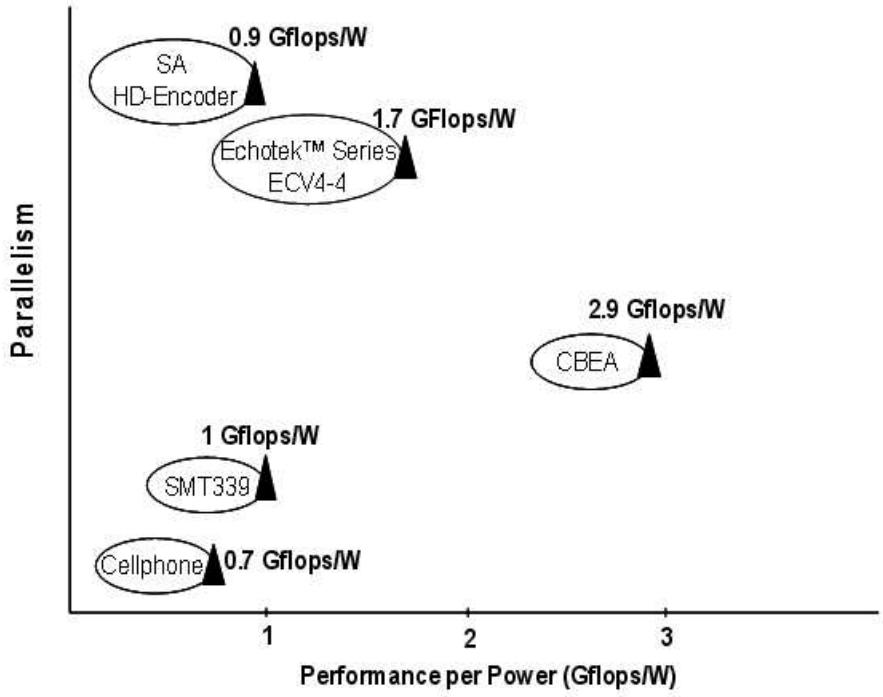


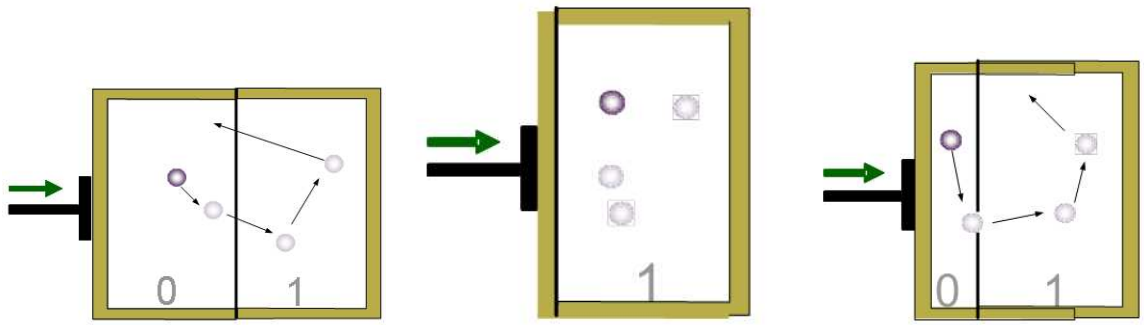
Figure 2: An interesting trend emerges as we plot Chip-Level Parallelism vs Performance per Watt. There is a point at which applications can no longer be parallelized and efficiency drops as parallelism increases. The Cell Broadband Engine Architecture (CBEA), which was built in part by me, is the most efficient chip on the market at the time of the survey.

that Ahmdal's law says that the gains of an application are limited by how fast the sequential portions (e.g. single thread) can perform [29].

The author surveyed several state-of-the-art chip architectures as of 2006 and interestingly we see a trend emerge when performance per watt is plotted vs parallelism for the chips shown in Figure 2. Note that in these plots we refer to chip-level parallelism in the macro-architectural or course-grained sense of the word.

Because applications only have a certain amount of parallelism that can be taken advantage of at the chip level, again according to Ahmdal's law, we see a saturation point in Figure 2 where increased parallelism decreases the efficiency of the application. The cell processor is the most efficient chip on the market that was surveyed [15].

This leads us to the obvious conclusion that in order to continue to increase



(a) (b) (c)

Figure 3: Fundamental limit of the switching energy of a digital bit. The calculation is derived from first physical principles shown as a thought experiment. (a) Initial state is we have two equally possible states at which the particle can arrive, 0 or 1, giving us no information about which state it will end up in. Energy=$kTln2$, assuming a heat bath at temperature T. (b) Energy is used to assure the particle is in state 1. The energy used is $\Delta E = kTln2 - kTln1 = kTln2$. (c) We can use less energy to assure the particle only has a probabiliy p of ending up in state 1. $\Delta E = kTln2p$. Energy Savings of $kTln\frac{1}{p}$ is seen for the probabilistic case.

performance and efficiency, single cores and single-threaded processing not only *must* continue to improve, but this is the *key* to continued improvement.

1.2 First Attempts at Improving Single-Thread Efficiency

Due to the apparent death of instruction level parallelism (ILP) [105], alternative methods were looked at to improve efficiency. An extremely interesting discovery was made that is dubbed "Probabilistic Computing".

It turns out that if the physics of a transistor and the associated energy equations are taken into account, there is a relationship that shows an exponential decrease in energy consumption of a transistor as the probability of correct operation of this transistor decreases linearly [34] assuming a fixed noise level. Hence, one can trade a small loss in probability of correct operation, for a large gain in energy savings. Indeed, this energy-probability relationship is confirmed at the quantum level by Richard Feynman in [30] and illustrated in Figure 3.

Figure 3 shows that the fundamental energy per switching step used for digital

bits can be lowered using probabilistic computing techniques whereby we only know with probability p that the bit is in the intended state.

Thus, we have discovered and implemented a probabilistic computing principle used to gain energy savings at the cost of some incorrect operation.

1.2.1 Failure of Thermal Noise

The initial simulation results shown in Chapter 2 and published in [34] on this topic by the author assume a level of thermal noise of $200 - 300$ mV as the source of probabilistic error.

This affect becomes dominant when supply voltage is lowered into the *noise margin* of the digital gate, which is a region defined by Equation 2 where $U_T = \frac{kT}{q}$.

$$V_{dd} > 4U_T \text{ or} \tag{1}$$

$$U_T < 0.25V_{dd} \tag{2}$$

Positive results are achieved by using a BIased VOltage Scaling scheme we call BIVOS which biases the more significant bits with higher voltage and the least significant bits with a lower voltage to allow more energy gains for less error. This works because the result of an error in the least significant bits is much less costly to the total error magnitude and thus we extract our energy savings here.

Upon closer inspection, U_T never really gets above $25mV$ and even with scaling of technologies, nominal supply voltages have stopped scaling at about 800 mV, far above the $V_{DSAT} = 4U_T$ requirement for thermal noise to factor in. Even with subthreshold voltage scaling in the most current technologies, we scale the supply down to a minimum of around 150 mV [65]. Perhaps in the future, PCMOS due to thermal noise will become a viable technology, and thus results are outlined in Chapter 2, but delay and performance penalties are the much more dominant effect.

Table 1: Energy savings for H.264 image benchmark data sets where source of probability is voltage scaling causing delay faults. Energy savings are with respect to the energy used at the nominal supply voltage.

Element	Benchmark	p_δ	$E(p_\delta)$	$\Delta E(p_\delta)$	Energy Savings
Adder	Uniform	0.9999	$0.88pJ$	$2.59pJ$	75%
$E_{nom} = 3.47pJ$	Low Quality H.264	0.9993	$0.62pJ$	$2.85pJ$	82%
δ threshold = 127	High Quality H.264	0.9998	$0.62pJ$	$2.85pJ$	82%
Multiplier	Uniform	0.9998	$8.30pJ$	$11.73pJ$	59%
$E_{nom} = 20.03pJ$	Low Quality H.264	0.9549	$2.11pJ$	$17.92pJ$	89%
δ threshold = 127	High Quality H.264	0.9862	$2.11pJ$	$17.92pJ$	89%
FIR	Uniform	0.9999	$102.89pJ$	$34.67pJ$	25%
$E_{nom} = 137.56pJ$	Low Quality H.264	0.9998	$37.37pJ$	$100.19pJ$	73%
δ threshold = 255	High Quality H.264	0.9999	$57.46pJ$	$80.1pJ$	58%

However, these probabilistic results can be extended to delay to increase both energy efficiency and performance.

1.3 Probabilistic Results Extended to Delay

Although thermal noise is not a dominant effect in current technology, delay errors, where somehow the timing and latching of operations does not happen correctly due to voltage scaling and device mismatch, *are* the dominant effect. Indeed initial experiments were setup to test this effect.

A Fixed Point Adder, Multiplier and FIR Filter were built in cadence and simulated with 0.25 μm TSMC technology. A metric p_δ was used whereby the calculation is only considered incorrect if it is more than δ from the correct answer. The δ is listed and results compared to nominal operation in Table 1.

The results from Table 1 are very encouraging but are in simulation only and do not take into account the real circuits problem of latching and metastability which makes the concept impossible to implement in synchronous technology.

Thus the introduction of the idea for a *Probabilistic Asynchronous Datapath*!

1.4 Target Applications

The target applications of this research are applications that are datapath and mathematically intensive, which spend a large percentage of their computing power doing arithmetic. As a counterexample, in general purpose computing, a large percentage of the instructions are control based, involve memory lookups and memory address manipulation, and the key innovation to achieve low power in those designs are low power memories. Further, a memory address lookup must be exact so probabilistic computing principles would not be applicable to this area. Alternatively, signal processing applications involve as much as thousands of addition and multiplication operations per every memory lookup, and a large percentage of the power is used in the datapath. Signal processing applications also regularly operate with error, namely quantization errors, so this domain is particulary well-suited to probabilistic computing principles. The embedded computing domain is dominated by these types of applications and thus the focus will be in this area, not general purpose computing.

1.5 Introducing the rest of the Dissertation

Chapter 2 reviews the advances made by the author up to date on probabilistic computing principles. These are derived from formula and simulation only, and serve as a guide to the asynchronous research pursued in the later chapters. Chapters 3 and 4 introduce the necessary background for the asynchronous and floating gate circuits .

Chapter 5 details macro-architectural superscalar ideas that are used and transferred in a novel way to the micro-architecture used on chip. Chapter 6 presents this asynchronous micro-architecture.

Chapter 7 discusses the details of the 0.6 μm AMI asynchronous chip that was built and Chapter 8 presents the silicon results. Chapter 9 presents simulation results for the asynchronous chip architecture with more modern process technologies.

Chapter 10 reviews how PCMOS results can be applied to the asynchronous system presented. Chapter 11 presents an exciting *learning* digital floating gate technology that is still in the early stage in a technology's development lifetime, but is used to nicely supplement the asynchronous technology presented. The thesis is summarized and contributions are given in Chapter 12.

CHAPTER II

FUNDAMENTALS OF PROBABILISTIC DESIGN AND COMPUTING

Probabilistic design takes advantage of the fact that in many embedded and DSP calculations such as image processing, the result of a calculation need not be exactly correct to get an acceptable overall result, *e.g.* image quality. This phenomenon is already apparent with the effect of quantization noise in DSP systems. There are two interesting properties that arise from this fact. One is that it takes much less computing power to get an *estimate* of a calculation as opposed to the exact result. The second interesting property is that the completion detection logic that is necessary in asynchronous systems can be significantly reduced with a probabilistic implemention. These two properties combined allow for a new design space to be explored in arithmetic-intensive applications.

Probabilistic design of digital systems is viewed as an inevitable eventuality as CMOS device feature sizes continue to shrink into the nanometer regime [11] in accordance with Moore's Law. Due to a variety of phenomena ranging from *noise* [58], parameter variations [131], and manufacturing defects [113], computing circuits are becoming non-deterministic or probabilistic. As a response to the need for a design strategy to cope with these probabilistic faults, the concept of *probabilistic* CMOS (PCMOS) based architectures in DATE-2006 [16] have been developed.

The inevitability of these error-causing phenomena has led to the creation of the concept of probabilistic design: designing deterministic circuits in the presence of probabilistic errors. As this proposed research focuses on arithmetic, special attention to probabilistic arithmetic will be paid here. Probabilistic arithmetic is defined in

previous work by the author [34]:

> Informally, a *probabilistic arithmetic operation* is an operation where each i^{th} bit of the computational primitive—addition studied here—has an associated probability of correctness, p_i. A k bit probabilistic arithmetic operation is a function $\mathcal{O}_P$ where $\mathcal{O} : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^l$ is a function and $P =< p_0, p_1, \cdots, p_{l-1} >: 0 \leq p_i \leq 1$ is the *probability parameter* where p_i corresponds to the probability that the i^{th} bit of the output is correct. The case where $P \equiv< 1 >$ corresponds to a conventional (deterministic) function.

Previous research on probabilistic computing has focused on achieving low energy applications [34, 61, 44, 16, 46] in the presence of thermal noise. To summarize, one can lower the supply voltage of computing circuits so much so that the supply voltage becomes comparable with the voltage level of thermal noise causing errors. By employing probabilistic computing techniques, these errors due to thermal noise can be tolerated and thus allow for *overscaling* the supply voltage to achieve extremely low energy consumption. Figure 2 shows the probability distribution function (PDF) of a noisy gate where the output is correct on a probabilistic basis due to thermal noise. This figure illustrates the effect of thermal noise changing the desired output voltage to the incorrect value.

An experiment that was done in [78] shows the effects of thermal noise on an ultra-voltage scaled H.264 decoder circuit simulation. The resulting images can be seen in Figure 5 where the energy-performance product EPP was improved by a factor of $4.62X$ by using probabilistic design, and the degradation in the image due to the probabilistic errors are practically unobservable.

Another type of probabilistic design methodology is to lower the supply voltage such that the circuit no longer operates at a speed fast enough to do all computations

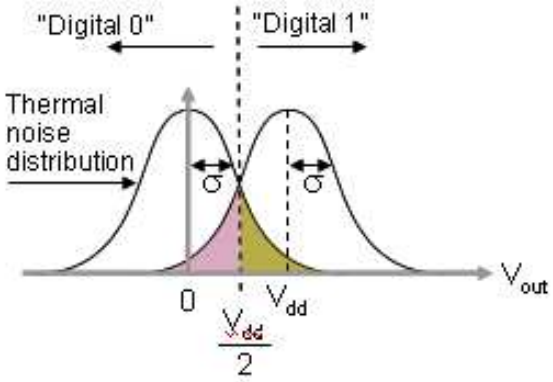


Figure 4: The probability distribution function (PDF) of the output voltage of a digital switch affected by thermal noise.

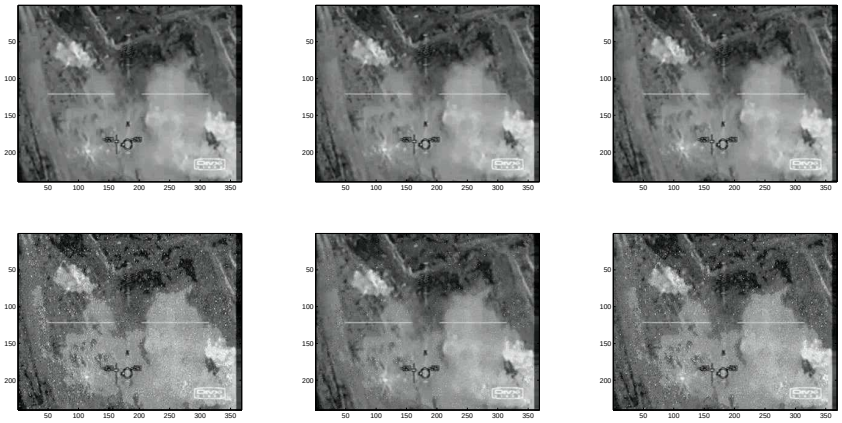


Figure 5: Application level impact of probabilistic design on 3 frames of a military, H.264 video comparing voltage scaled PCMOS [bottom](with an EPP ratio of $4.62X$) to the original H.264 frames [top]

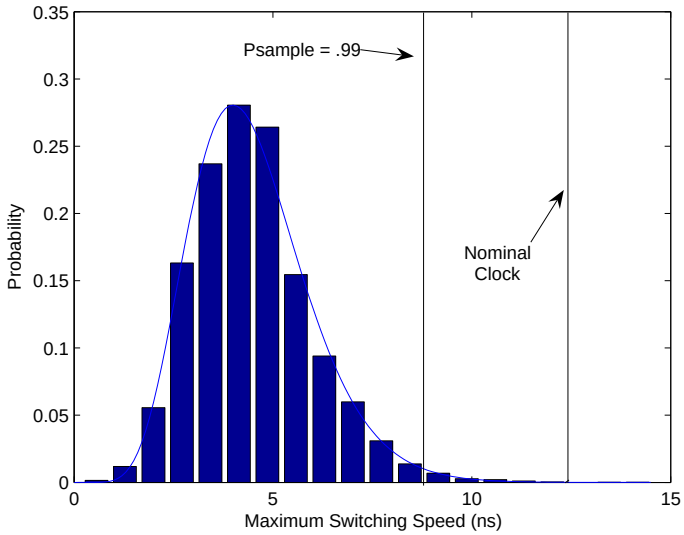


Figure 6: Histogram of maximum propagation delay over a uniform input set for an 18x18 bit array multiplier. A 33% increase in clock speed would only cause delay faults (when the switching speed of the circuit is longer than the clock period) in 1% of the inputs.

before the clock pulse resulting in delay faults [7, 46]. Since the speed at which datapath circuits compute is highly dependent on the input data, these delay faults occur probabilistically based on the input set. An illustration of this phenomenon can be seen in Figure 6. This figure shows that if the nominal clock speed is increased 33%, a delay fault would occur on only 1% of the input set.

Instead of increasing the performance for an 18x18 bit multiplier as illustrated in Figure 6, the energy consumption of the multiplier could be lowered such that the circuit performed 33% slower to achieve the same accuracy of 99% correctness.

Probabilistic operation of circuits due to thermal noise has been well studied, but it is the goal of this research to shed light on probabilistic operation of circuits due to temporal noise (delay faults) which has not been well studied, particularly in asynchronous design.

CHAPTER III

FUNDAMENTALS OF ASYNCHRONOUS DESIGN

The definition of an asynchronous circuit is a circuit that is self-timed and operates without a clock or other synchronizing device. The origins of asynchronous design started with Huffman who proposed the *fundamental mode assumption* where self-timed circuits operated under the assumption that the environment will apply only a single input change at a time and not another until the circuit has stabilized otherwise known as a *bounded delay model* [49].

Muller introduced a new model to greatly widen the class of available asynchronous circuits. He proposed a model with unbounded delays – that is the circuits had no assumption on which path would finish first making the circuit *delay insensitive* – and that the completion of these circuits would be detected via a Muller C-element [92]. This is the most important classical innovation although the full history can be seen in [5, 9, 93, 97, 127, 31, 32].

The next series of important innovations in asynchronous design were developed by Martin's group [83]. He fundamentally changed asynchronous computing by defining two important principles:

Definition: A circuit is *quasi delay-insensitive (QDI)* if no assumptions are made about the delays within the circuit except that the delay on different paths of an isochronic fork are equal.

Definition: An *isochronic fork* is when the output of a gate is connected to the input of multiple other gates. In other words, there is an isochronic fork when a gate

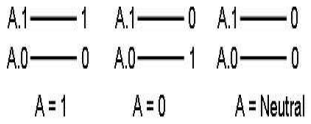


Figure 7: Dual rail encoding for bit 'A' over two data wires.

has a fanout greater than 1.

By making the QDI assumption, Martin proved that all Turing-computable functions, and thus all computing functions of interest, could be built asynchronously [83, 1, 79, 84, 82]. Martin also developed the concept of delay-insensitive hazard free coding of the bitlines resulting in the code of choice: a dual rail code [81], which can be seen in Figure 7.

In order for correct completion detection, before each calculation all bits are set to a neutral value. After a calculation, all bits are set to a valid value. Since all transitions are hazard-free and make only a single transition for any change in input, any time all ouptut bits in a dataword for a functional unit are neutral, the processor knows the functional unit is ready for the next input. Any time all output bits in the dataword are valid, the processor knows the calculation is complete. This allows for a simple handshaking system where all values in the datapath are constantly switching from neutral to valid back to neutral.

An example of a typical completion tree that signals when all bits in a dataword are either 'valid' or 'neutral' is seen in Figure 8.

A pipelined handshake removes an intermediate variable and also removes data dependency so that handshaking operations can be done in parallel. The most commonly used circuit in Martin's work, the pre-charged half buffer (PCHB) handshake circuit, is used to send a signal from channel L to channel R as in Equation 3. A schematic of the PCHB can also be seen in Figure 9. The symbols $v(L)$ and $v(R)$ represent completion trees on all the bits of channel L and R that validate whether

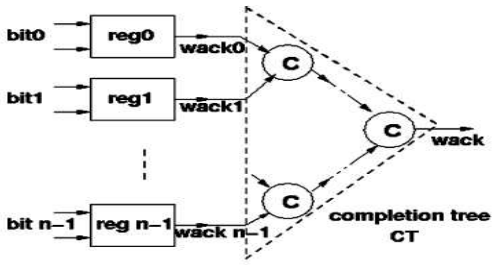


Figure 8: A completion tree for a typical N-bit dataword that sends a write acknowledge (wack) signal when write is complete.

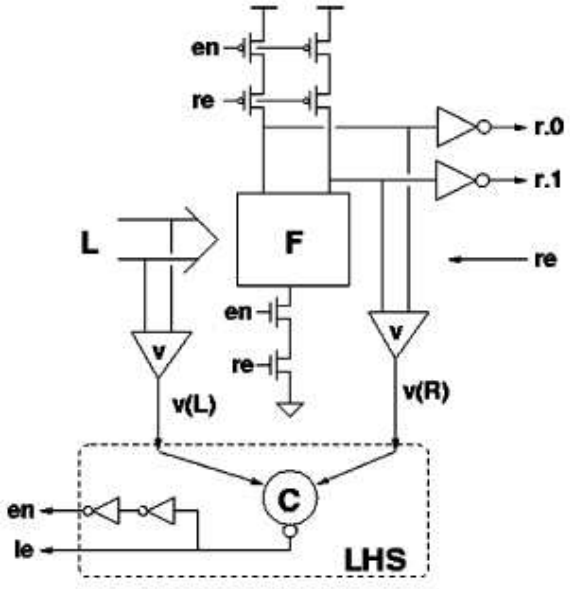


Figure 9: Schematic of an asynchronous pre-charged half buffer circuit that uses pipelined handshaking.

the data is completely valid or completely neutral. The symbol F represents a block that implements a function of the variables from channel L. The symbol en is a sleep transistor and unimportant for this example.

$$\begin{aligned} \text{PCHB} \equiv\ & *[[R^e]; [L^0 \rightarrow R^0\uparrow \| L^1 \rightarrow R^1\uparrow]; L^e\downarrow]; \\ & [\neg R^e]; R^0\downarrow; R^1\downarrow; [\neg L^0 \& \neg L^1]; L^e\uparrow] \end{aligned} \tag{3}$$

There have been many n-bit adders proposed that are deemed both power and speed effecient in the asynchronous domain. Evaluation of adders is an extremely

important topic in computing, particularly in digital signal processing and other data intensive computing, due to the fact that 72% of datapath instructions are additions [24]. The most efficient QDI, asynchronous implementation for a full addder was found in [81], which will be used in this research and is shown in Figure 10. In this figure, the two inputs to the full adder are a and b with dual rails a_t, a_f, b_t, b_f and carry in c with dual rails c_t and c_f. The outputs are the sum bit s with dual rails s_t and s_f and carry out d with dual rails d_t and d_f. The sum and carry out outputs are inverted.

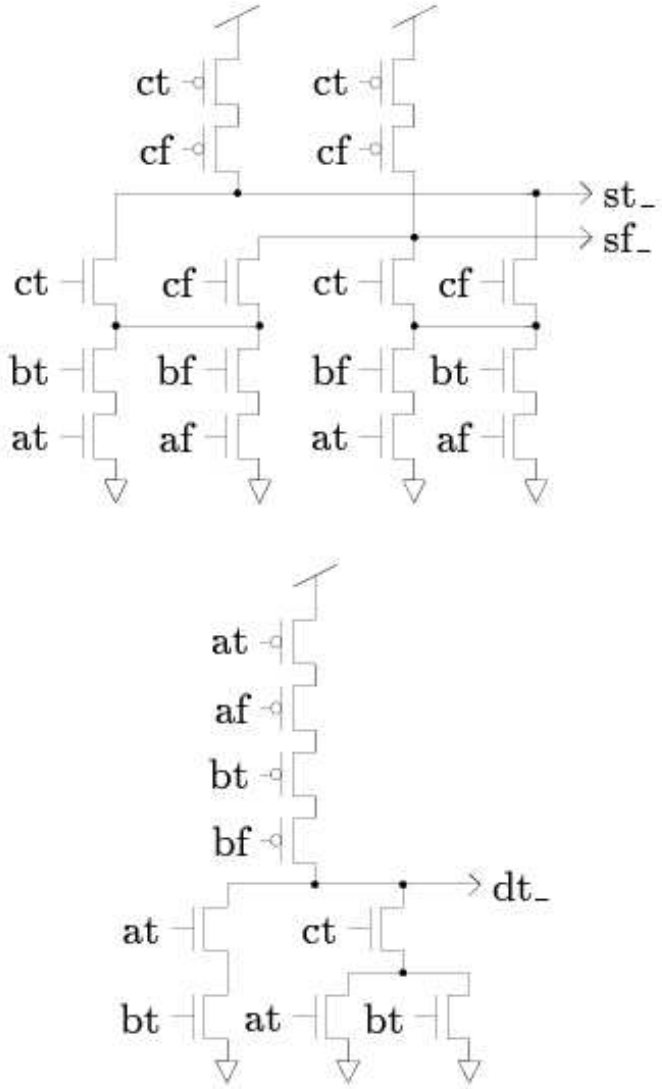


Figure 10: A QDI, dual rail full adder implemtation with inverted outputs.

CHAPTER IV

FUNDAMENTALS OF FLOATING-GATES

A floating-gate is a transistor that has the ability to hold charge on its gate because of a capacitance and a lack of any paths to discharge. In this way electrons 'float' on the gate until such time as they are discharged by the transistor switching. The first floating-gate was discovered in 1967 in Bell Labs as a mechanism for non-volatile memory storage [53]. Because of its ability to hold charge for extremely long periods of time without further energy requirements, floating-gates have become popular in non-volatile memories and memories for mobile devices such as in flash memory and EEPROM's [85, 66].

The most wide use of floating-gates is for digital memory systems, however recently they have been used as circuit elements [41]. An initial example of this was in the ETAN chip in 1989 where analog floating-gate cells were used to create a neural network where the gates were used as direct inputs to multipliers to do intense arithmetic and neuromorphic calculations [47].

Althoug the majority of the floating-gate circuits are used in neuromorphic applications, they have also shown themselves to be quite useful in capacitive-based circuits such as digital arithmetic [41]. In [40], Hasler *et al.* showed that pFET floating-gate circuit elements could be 'programmed' or charged by electron tunneling and charge was removed through hot-electron injection. With this programming ability, floating-gates can be used as dynamic and adaptive circuit elements to act as potentiometers, voltage sources, or a varying capacitive circuit.

Floating-gates allow for the placement of programmable capacitive circuit elements in the paths of arithmetic units that allow for additional charge or additional

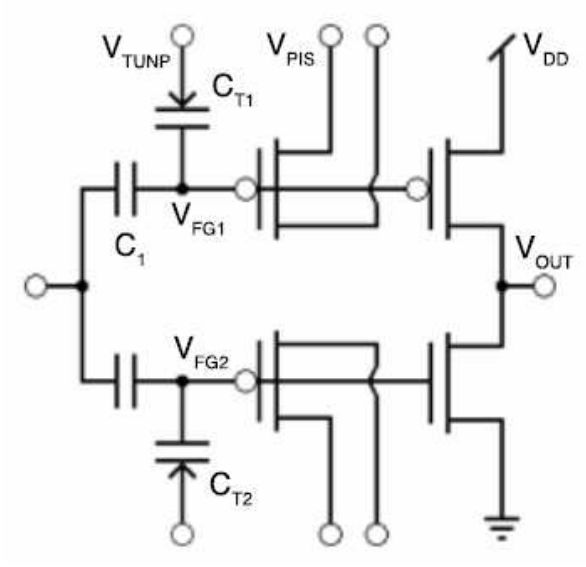


Figure 11: Inverter with a programmable floating-gate connected to each transistor to control the input voltage at which the transistor switches.

capacitance to be added so that these paths are slowed down (and consume less energy) or sped up [42]. To summarize, the voltage threshold (V_{th}) of a gate can be dynamically altered [120]. In [27], it is shown that the switching speed of an inverter can be controlled by programmable floating-gates that are connected to each transistor (FET) in the design. A transistor-level circuit diagram of this circuit can be seen in Figure 11.

It is possible that only one floating-gate on each pull-down, pull-up network in a CMOS instantiation is possible, and on-going work is being done to test the best implementation on other gates [27]. It is in this regard that floating-gates will be used in the proposed research.

CHAPTER V

FINE-GRAINED PARALLEL ASYNCHRONOUS PIPELINED ARCHITECTURE: A PREFACE

5.1 Motivation

Presented here is an asynchronous architecture that allows for parallel computation and pipelining at the finest grained level available in digital systems: the bit level. This parallel asynchronous pipelined architecture (PAPA) takes advantage of the statistical nature of data dependencies to allow functional units at the bit level to compute and commit data instead of being hindered by having to wait for all the bits in the datapath to finish before committing data.

An illustrative example of this phenomenon is shown by the standard 16-bit, ripple-carry adder. The example of an adder is also appropriate here because statistics show that in general purpose computing it is estimated that up to 70% of instructions are adds, and this number is suspected to be even higher for digital signal processing (DSP) systems. A ripple-carry adder is shown in Figure 12.

In synchronous architectures, the adder is generally a single pipeline stage, and the adder is also considered to compute as a single unit with all bits completing their calculation according to the same clock.

This PAPA asynchronous design is motivated by similar techniques used in super-scalar computing, such as deep-pipelining and instruction level parallelism.

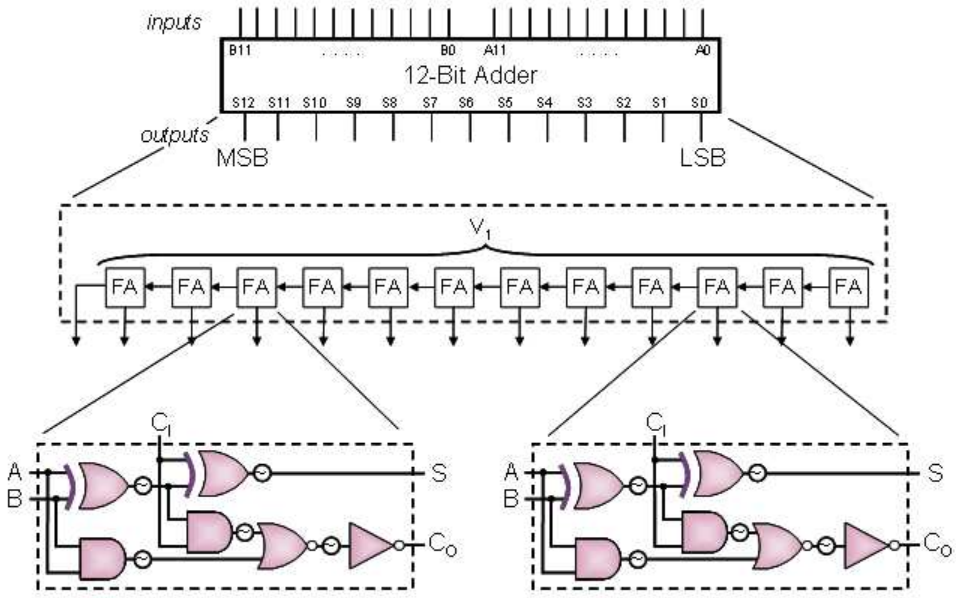


Figure 12: A 12-bit ripple-carry adder. Adders are generally not pipelined in synchronous systems due to their simplicity and the limitations of synchronous systems. Also because of the global clock paradigm where all bit-level computations are considered complete according to the same clock pulse.

5.2 Super Scalar Architecture

5.2.1 A Brief History

Super scalar architectures, which spawned a revolution in computer design, were arguably first presented in their modern form with the 1995 release of the MIPS R10000 [128]. Perhaps the defining characteristic of a superscalar architecture is that it issues multiple instructions per clock cycle and implements a group of techniques called *instruction level paralleslim* (ILP). Data dependencies are detected and eliminated between instructions with register renaming and other substitutations to allow multiple instructions to be issued at a time from a single instruction stream.

Specialized functional units allow multiple instructions to be executed at a time by employing several *execution units* on a single CPU such as adders, multipliers, and ALUS of which a single superscalar CPU may have 3 or more of each.

Pipelining is also a critical part of a superscalar architecture where only fractions of instructions are executed per clock cycle which allows copies of the same instruction

to be issued faster and not have to wait for the previous copy of the instruction to complete before a new copy is dispatched.

Speculative branch prediciton, introduced in the 1990s, was employed to keep the pipelines and funcitonal units at capacity with instructions and operations to execute. The computer must "guess" what future instructions will be to keep the pipelines full [105].

However, many of these concepts date back much further than 1995. Availability of multiple arithmetic units and exploiting them for multiple instruction issue was developed by Tomasulo in a seminal paper resulting in the famous Tomasulo's algorithm in 1967 [118]. For the next 3 decades this allowed shorter and shorter pipeline stages, deeper and deeper pipelines and ever increasing clock frequencies and higher throughputs.

However, the MIPS R10000 is arguably given credit for the rise of the modern superscalar architecture with the invention of *out of order instruction scheduling*, and reservation stations in addition to implementations of branch prediction, deep pipelines, and specialized funcitonal units. Reservation stations allow the ability for multiple instructions to queue for a specific functional unit, such as an adder, ALU, or multiplier, and for them to get issued to the functional unit when the operands are available, possibly out of order. A schematic of a superscalar architecture with reservation stations is show in Figure 13.

With the use of reservation stations and out-of-order execution, we can continue to keep the pipeline full so as to take advantage of the novel form of asynchronous pipelining presented here.

There are many that would argue that instruction level parallelism and improvement in sequential processing is dead, however according to Amdhal's law, improving the processing of a sequential element of a program is the only way we can increase the performance of individual applications [29]. Bit-level parallel asynchronous pipelined

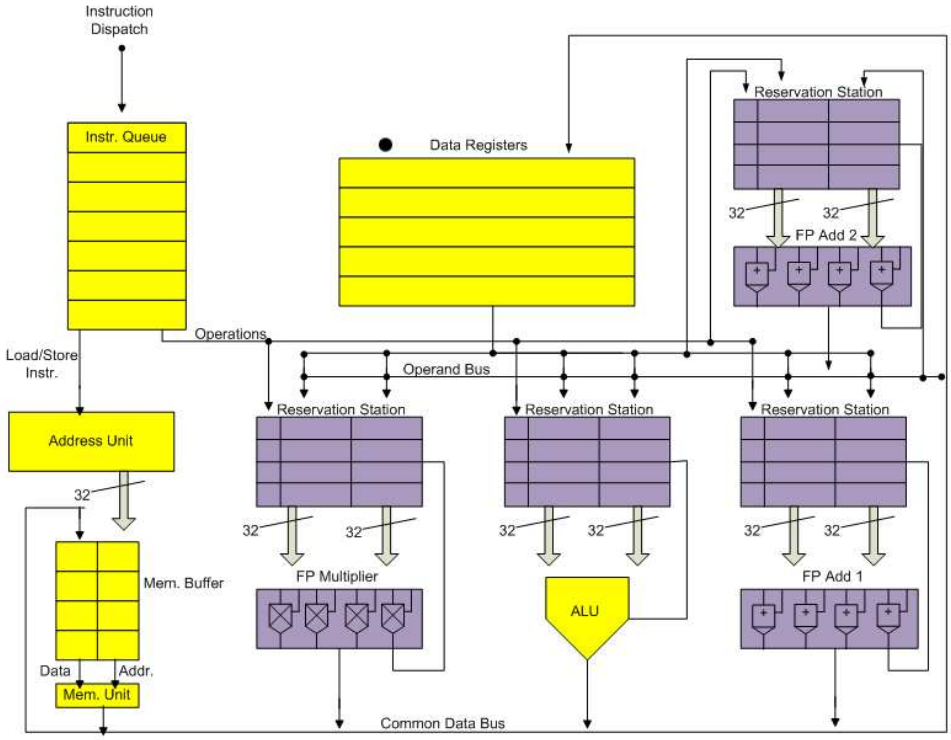


Figure 13: A high level schematic of a modern superscalar architecture in its simplest form. There is a single instruction dispatch queue reading and writing from a block of memory. Operations are dispatched to reservation station which buffer operations and scan the common data bus for the needed operands. When these operands are received, the reservation station commits the operation to the functional unit for (possibly out-of-order) execution.

architecture is presented here as a solution to increasing sequential execution performance.

CHAPTER VI

A NOVEL BIT-LEVEL ASYNCHRONOUS PIPELINE

6.1 Death of Small Depth Synchronous Pipelines

Barriers associated with synchronous architectures have arisen to halt pipelining in synchronous architectures resulting in the *death* of frequency scaling and the move to multi-core and heavily parallel architectures to continue to scale up performance and throughput.

Clock skew, difficulty in routing such an extensive clock network, and most importantly circuit variations have presented the fundamental barriers to scaling of synchronous pipeline stages [13]. An example of a multi-stage pipeline is shown in Figure 14.

As the pipeline stages from Figure 14 get smaller down to the gate level, circuit variations start to become the dominant timing effect where more timing margin is allowed for circuit varitions and latch setup and hold times than the actual gate delays themselves. Single gate pipeline stages in synchronous systems become infeasable due to this effect. Figure 15 shows the extreme delay variation in single gates in an Intel

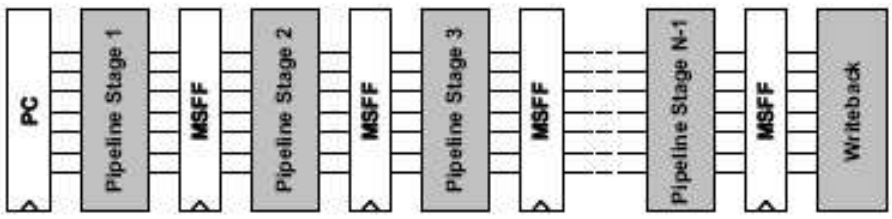


Figure 14: A typical N-stage pipeline originating at the program counter (PC) going through several multi-stage flip-flops (MSFF) and finally being written back to memory *courtesy Keith Bowman.*

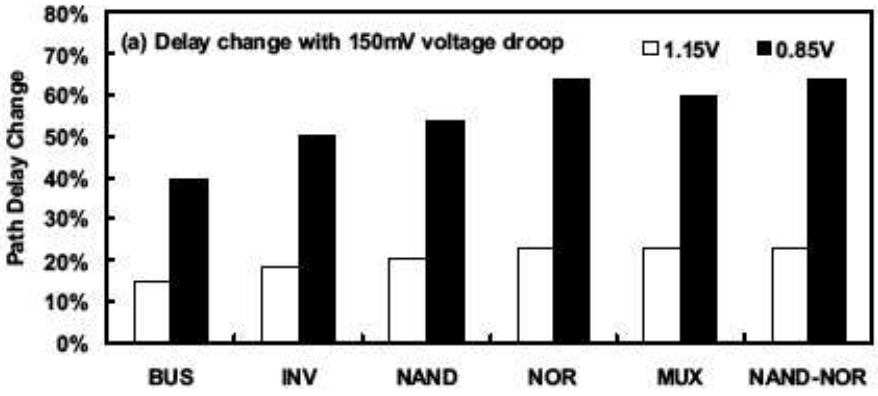


Figure 15: Plot showing delay variation in gates in an Intel 45 nm process due to supply voltage droop of 150 mV. The plot shows that single-stage and 2-stage gates have as much as 80% variation in their delay *courtesy Keith Bowman.*

45 nm process with just a 150 mV droop in supply voltage, which is quite common.

From Figure 15 we see that delays vary as much as 80% in pipeline stages with single gate and 2 gate delays due to parametric, voltage, and temperature variations. Accounting for clock skew, a synchronous designer would be forced to leave a margin of 2X the actual delay, making timing gains non-existent in such short pipeline stages.

Another study done by Professor Saibal Mukhopadhyay [2] illustrated in Figure 16 shows that as the logic depth of a pipeline stage increases from 16 to 49, the effect of current variation on delay variation decreases. However, below a logic depth of 16 there is almost a one-to-one correspondance between current variation through a given device and the variation in the delay of the pipeline stage. Once again, a fundamental limitation to synchronous pipeline scaling is presented due to PVT variations.

6.2 An Asynchronous Pipeline Solution

Asynchronous architectures allow for the most fine-grained pipelines possible down to single gate pipeline stages unlike synchronous architectures that have become effectively limited to 3-16 gates per pipeline due to the reasons listed in Section 6.1.

Asynchronous dual rail completion logic explained in Section 3 is the technology at the heart of the single gate asynchronous pipeline, shown in Figure 17. This is an

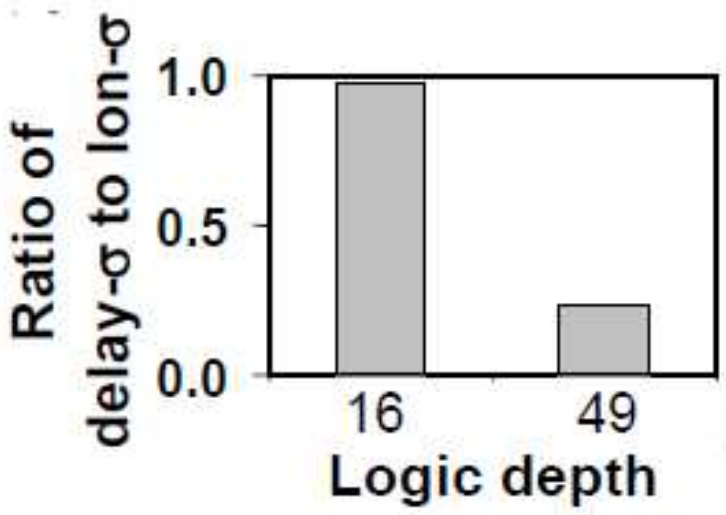


Figure 16: Graph showing that as the logic depth of a pipeline stage increases, the effect of current variation on delay variation decreases. This implies longer pipeline stages are more delay variation-tolerant. *courtesy Saibal Mukhopadhyay.*

interleaved computation and control style [114, 83].

The functional blocks in Figure 17 can be as complex or as simple as the designer wishes, although for high-throughput datapath operations, these functional blocks will often be single-gate or single-bit functions.

Figure 18 shows the full cycle for block F_1, and each block goes through this same cycle. Since block F_1 feeds data to block F_2, block F_2 must complete its operations

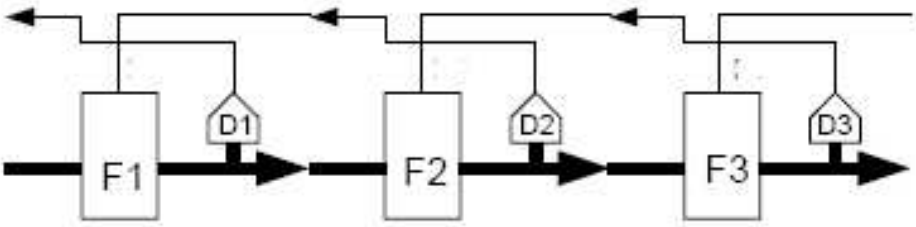


Figure 17: An asynchronous pipeline with interleaved computation and control. A function is evaluated in the F_N blocks and the result sent to the next pipeline stage, F_{N+1}. A conrol signal is then generated D_N and sent to the previous block F_{N-1}. Alternate waves of data and reset signals are sent through the pipeline, the reset signal is used to reset the asynchronous blocks to prepare them for a new calculation in accordance with dual-rail logic.

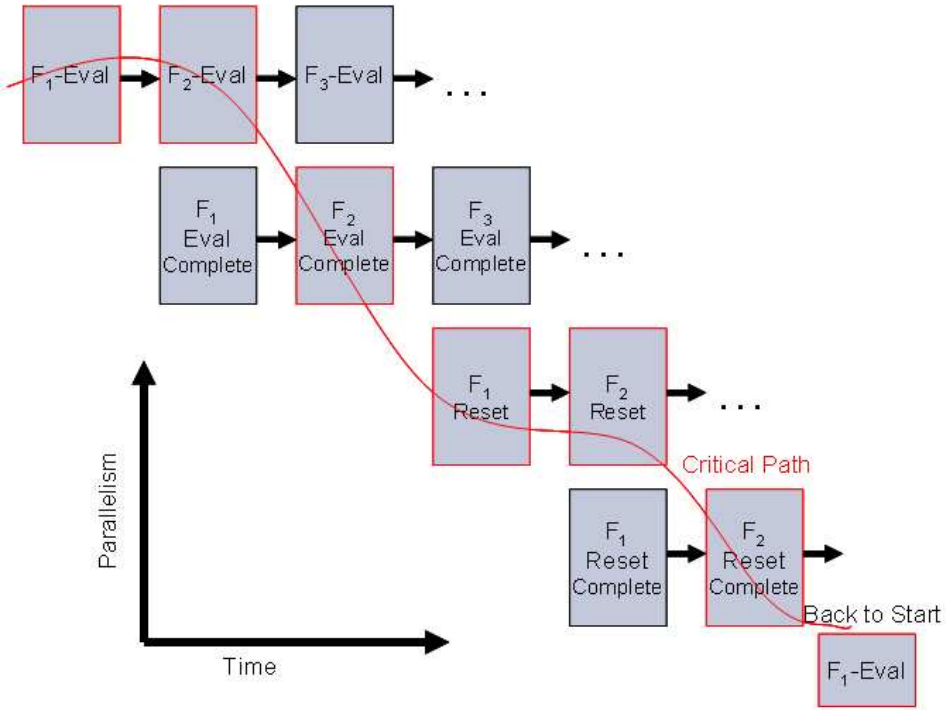


Figure 18: A graph showing the pipelining of the evaluation, completion, and reset operations of the structure shown in Figure 17. A functional block F_N computes and sends the data to block F_{N+1}. In parallel, the completion block D_{N+1} signals that computation F_{N+1} is complete, which signals to the previous block F_N to reset, then F_{N+1} is reset and so on. The same completion block D_{N+1} signals that block F_{N+1} has completed resetting, which signals that block F_N is ready for a new cycle.

before F_1 starts a new calculation. The cycle time for this architecture is given in Equation 4 where t_{Eval} is the time for block F_N to *evaluate* or compute the data, t_{CD} is time to do *completion detection*, and t_{Reset} is the time to *reset* block F_N.

$$\mathcal{T}_{interleave} = 2 \cdot t_{Eval} + 2 \cdot t_{CD} + 2 \cdot t_{Reset} \tag{4}$$

In single-bit pipelines where each functional block F_N represents a single bit, t_{CD} is almost insignificant as each bit takes the time of a single transistor switching on the completion detection block. However, as the bit width scales by N, t_{CD} scales by $NlogN$ due to the nature of the completion tree in asynchronous handshaking. Thus scaling down to a single bit is highly advantageous on reducing the timing of completion signals. And of course, asynchronous logic is not affected by delay variation since the delays do not have to be synced with a clocking system, making asynchronous logic very amenable to short stages.

6.3 A Decoupled Asynchronous Pipeline Solution

A more efficient way to do bit level pipelinig is to decouple the control and evaluate logic, which is the style proposed in this dissertation, at the risk of device mismatch.

A *decoupled* asynchronous pipeline is shown in Figure 19.

As Figure 19 shows, the completion blocks C_N and the functional blocks F_N are decoupled. The completion mechanism shown here actually uses a *lookahead pipeline* (LP) method first proposed in [115], although the design presented here is adapted for dual-rail static logic rather than the power inefficient dynamic logic in [115].

Before block F_N is reset, block F_{N+1} must be done evaluating and block F_{N+2} must be done resetting, so that 2 out of the 3 blocks are always in evaluate phase. This technique effectively takes the 'reset' computation out of the critical path of the cycle since stage N evaluates when stage $N+1$ has *started* to reset.

However, with single-stage, gate-level pipelining where block F_N has a single gate

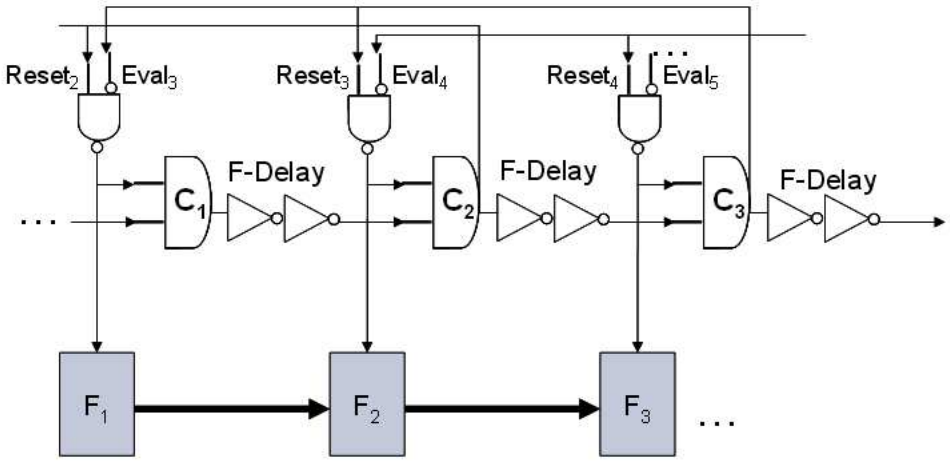


Figure 19: A decoupled asynchronous pipeline architecture. Instead of the completion block, C_{N+1}, detecting the completion of block F_{N+1}, it simply detects when block F_{N+1} has *started* its evaluation/reset calculation and uses the delay through the *F-Delay* block to make sure F_N is not reset before F_{N+1} is done with the data. The *F-Delay* block is designed to be a *matched delay* to the F_N blocks.

delay which is equal to or greater the gate delay of the C-element gate, no matched delay is needed taking out yet another delay as shown in Figure 20.

This new architecture brings the cycle time of an asynchronous handshake system down by several gate delays. The cycle flow graph is given in Figure 21.

Because the control logic and evaluation logic are decoupled, the completion gate can operate in parallel with the function blocks, a change from the original architecture presented. First F_1 evaluates with block C_1 asserting in parallel then block C_2 asserts while F_2 is evaluating. F_1 resets while F_3 is evaluating all while C_1 signals that F_1 is done resetting and starting the reset signal on C_2. Thus while F_2 is resetting, the reset signal from C_2 and evaluate signal from C_3 are already being sent back to let block F_1 know it can evaluate again. It reduces the evaluation time as shown in Equation 5.

$$\mathcal{T}_{decoupled} = t_{Eval} + t_{CD} + t_{Reset} + t_{NAND} \tag{5}$$

This modern pipeling architecture has reduced the cycle time of the asynchronous

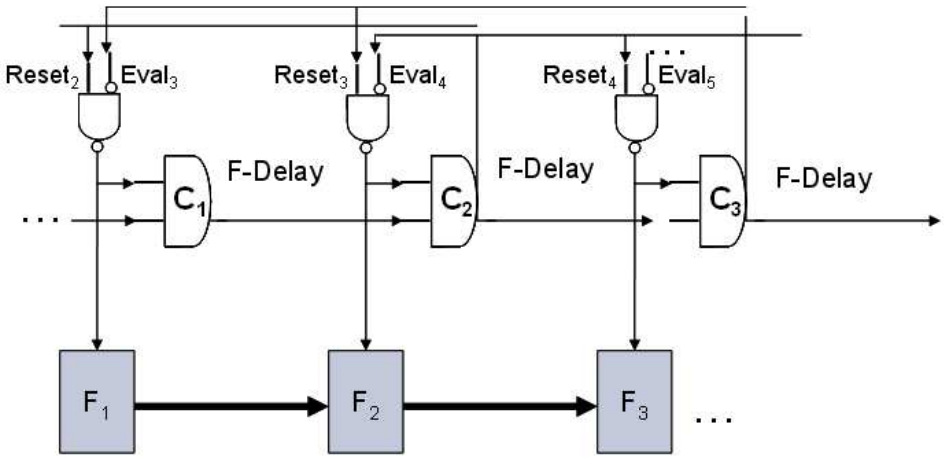


Figure 20: A decoupled asynchronous, gate-level pipeline architecture. Since the F_N blocks are singel gates in this implementation, no *matched delay* is needed, removing an additional delay from the timing cycle.

handshake to less than the setup time and margin time put into asynchronous systems giving this asynchronous architecture a design win even without considering the average case timing and gate-level pipelining paradigms that also give asynchronous technology a decided timing advantage.

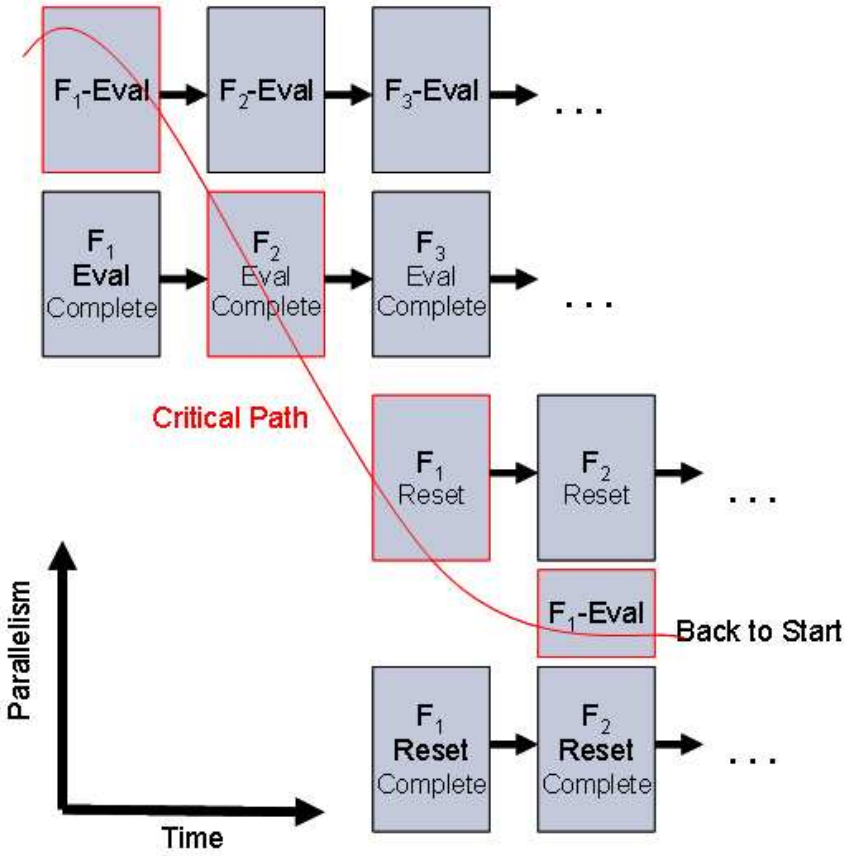


Figure 21: In the new architecture seen in Figure 19, the completion gates switch concurrently with the function blocks and use the matched delay buffers to make sure timing errors do not occur. Another improvement is stage F_N can evaluate as again as soon as either stage F_{N+1} *starts* its reset phase since the completion gate switches at the start of this reset phase, or as soon as stage F_{N+2} finishes its evaluate phase.

CHAPTER VII

ASYNCHRONOUS CHIP ARCHITECTURE, ANALYSIS, AND SIMULATED RESULTS

7.1 Summary of Design and Motivation

7.1.1 Executive Summary of Design

Motivated by the unwillingness to accept the worst-case timing constraint that synchronous logic imposes, and additionally motivated by finding a supply voltage scaling scheme for datapath circuits that is unconstrained by timing errors in memory elements, this chapter presents an asynchronous datapath that is embedded seamlessly into a synchronous register file system. This chapter will show that not only does asynchronous arithmetic logic exhibit many characteristics that allow it to be inherently lower power, but it is significantly faster than any synchronous counterpart and is a perfect candidate technology for datapath acceleration. Further, novel circuits are presented that allow asynchronous datapath units to be embedded in a synchronous environment with little overhead while the dual-rail asynchronous encoding scheme is successfully converted with equally low overhead. A fully functioning chip is presented, and simulated results are given. The circuits on this chip will be discussed showing this design to be both energy and performance efficient when compared to other known datapath designs.

"The danger is generally in the delay" - M. de Cervantes. Don Quixote

7.1.2 Design Motivation

Embedded devices loom as the most challenging digital systems problem to date because of the dual reality of needing performance capable of real-time multi-media

and graphics processing, but also long battery life, and battery life does *not* scale with Moore's Law. It is clear by now the importance of solving these problems as an array of interesting statics emerge such as by 2011, it is predicted more users will gain internet access through mobile devices than PCs [119]. Applications once programmed to be used on super computers are being used on cell phones.

This chapter seeks to address some of these challenges through a novel concept, asynchronously embedded datapath design. The idea is to take advantage of asynchronous circuits and place them where they are best suited to yield performance gains and energy savings, and use synchronous logic for control and other types of operations where asynchronous circuits would need extraneous overhead. The great advantage asynchronous logic has over its synchronous counterpart is twofold: it takes advantage of *average-case timing* instead of worse-case timing and is immune to timing errors that result from voltage scaling or process variations [83].

Hence datapath circuits are a prime candidate for such a study. Asynchronous logic is much more performance efficient in arithmetic computations because the *average-case* delay is orders of magnitude faster than the worse case for these types of calculations; the delay complexity is $\mathcal{O}(loglogN)$ instead of $\mathcal{O}(N)$ [72]. Asynchronous circuits also do not suffer from *static and dynamic switching hazards* that cause spurious dynamic switching currents to drain power, and these *hazards* are much more prevalent in datapath calculations then anywhere else in digital circuits [18].

This chapter will illustrate several novel concepts inherent to an asynchronously embedded datapath and will present the architecture, simulations, and results of a 0.5 μm chip that puts these theories into practice, the layout of which is seen in Figure 26. The contributions this chapter makes are the following:

- Methods, circuits, and architectures for seamlessly embedding an asynchronous datapath within a synchronous design.

- A novel asynchronous full-adder circuit design to reduce the critical path in

aritmetic units to a single transition per bit.

- A completion algorithm for *probabilistic or bit-pipelined* completion detection to increase performance and decrease energy consumption of the asynchronous handshake.

- Complexity and power analysis of the aforementioned circuits as well as several other general arithmetic circuits.

- Simulation results from the chip being presented.

A brief discussion, background, and analysis on asynchronous logic as well as the proposed asynchronous full adder circuits will be presented in Section 7.2. The chip and the accompanying architecture is illustrated in Section 7.3, and results and conclusions are given in Section 10.4.

7.2 Trading Datapath Timing Information for Energy

7.2.1 Asynchronous Design

Asynchronous design is the key to being able to trade timing observability for both lower power and more performance-efficient design. Note that in this work we seek to separate the concepts of timing or clocking a circuit from performance efficiency. The easiest way to relax this concept of time observability is through asynchronous design on which a brief introduction and background will be given here.

In synchronous designs, a clock pulses at regular intervals into memory elements where these memory elements latch and store a calculation on the clock pulse. Thus a calculation must begin and complete within the clock interval to be latched correctly, and consequently every calculation is observed to take time $\tau = \frac{1}{f}$ where f is the frequency of the clock.

Conversely, asynchronous design is built with *hazard-free* logic where each bit in the datapath is guaranteed to switch only once per calculation and the completion of

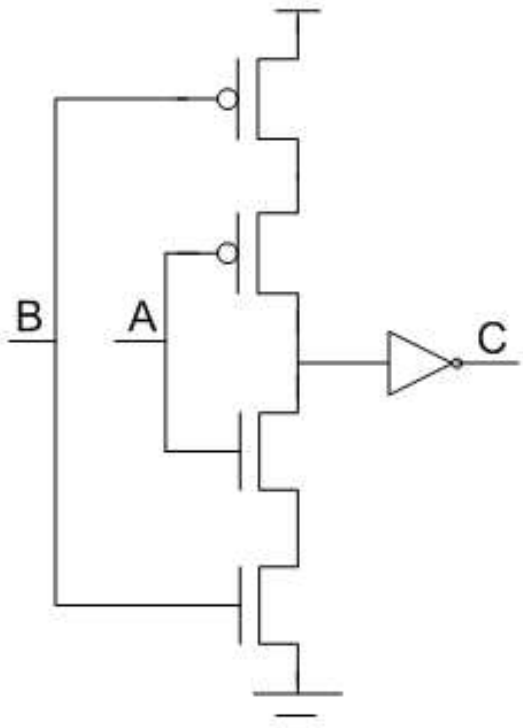


Figure 22: A 2-input Muller C-element for asynchronous completion detection. Detects whether 2 signals are both '0' or both '1' and holds its current state otherwise.

this transition is flagged immediately with a completion element known as a Muller C-element shown in Figure 22.

To be *hazard-free*, asynchronous logic must use an encoding scheme, the most space-efficient of which is dual-rail encoding where each bit is encoded with a 'True' line and a 'False' line so that a signal S is encoded as S_t and S_f. The circuit to calculate the 'True' line of the carry-out bit of a full adder, $\overline{D_t}$, is shown in Figure 23. The output is inverted that is designated by the underscore, $\overline{D_t}$. Dual-rail encoded datapaths were first proposed in [81].

The circuit in Figure 24 is a single-bit adder used to calculate every odd carry-out bit, D_t, because it can utilize the inverted carry-out bit from the previous full-adder circuit shown in Figure 23. By using these alternating inverted/noninverted-output carry functions, no extra inverters are needed in the critical path of an adder or other

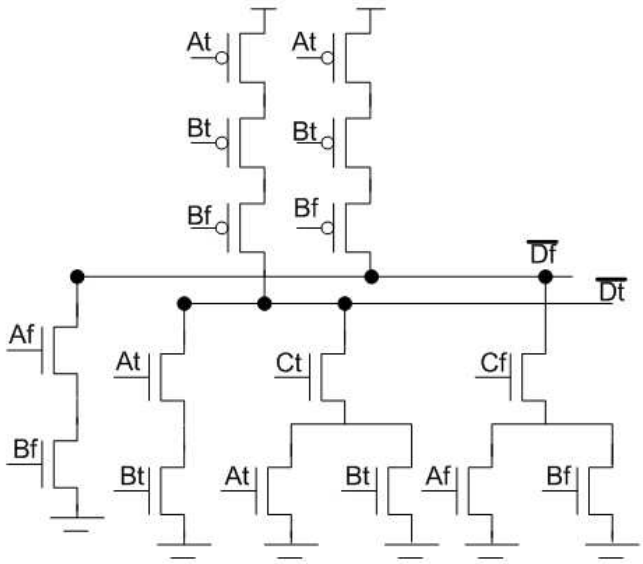


Figure 23: The carry-out signals of a dual-rail asynchronous full adder with inputs A and B, carry-in C, and carry-out D. Because PMOS transistors are more efficient at pulling voltage at its source node up (passing a '1'), but PMOS transistors are used to produce the *off-set* (the '0' logic) for the carry-out function, the outputs are inverted.

arithmetic units built with this technology. This is one of the novel contributions of this work: there is only a single gate (transition) per bit in the critical path. The circuit to calculate the sum bit is similarly built, and the design alternates every other bit. Note that on average, each full adder only uses 44 transistors, which is only slightly more than the 40 transistors used in the 'conventional' full-adder so popular in the literature [18].

Dual-rail encoded logic has a simple encoding scheme outlined in Equation 6.

$$\begin{aligned} &\text{if } S_t \uparrow \wedge S_f \downarrow \quad \text{then} \quad && S = 1 \\ &\text{if } S_t \downarrow \wedge S_f \uparrow \quad \text{then} \quad && S = 0 \\ &\text{if } S_t \downarrow \wedge S_f \downarrow \quad \text{then} \quad && S = \text{Neutral} \end{aligned} \tag{6}$$

A neutral state, where the dual-rail encoded bit does not represent a valid state, is necessary so that completion detection and handshaking algorithms can be used. If the state of a circuit is neutral, the circuit is ready for new inputs, and if the circuit is in a valid state (e.g. $S = 1 \parallel S = 0$), the circuit has completed its calculation.

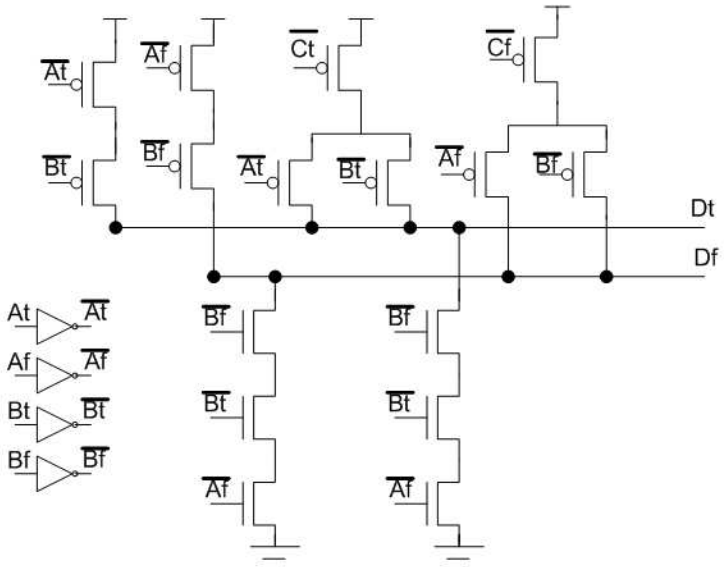


Figure 24: 'True' and 'False' lines of non-inverted carryout bit for an asynchronous full-adder. The $\overline{D_t}$ bit of the inverted-output full adder from above feeds into the C_t bit of this noninverted-output full adder so that no inverters are needed in the critical path.

Thus a tree of completion elements are fed by the output bits of the datapath, and the entire N-bit calculation is complete when the completion element at the root of the tree goes high or low depending on the convention used.

Version 2 of the chip presents another method for resetting the functional units (full adders) on the chip to allow bit-level pipelining. The pullup network is replaced with single reset switches. This is a form of dynamic asynchronous logic as shown in Figure 25.

7.2.2 Power Comparison

Asynchronous circuits have several attractive properties in regard to power consumption. The power consumption of digital CMOS circuits is derived from the now well-known equation [18]:

$$P_{total} = \alpha f C V_{dd}^2 + I_{short} V_{dd} + I_{leak} V_{dd} \tag{7}$$

For current technology nodes, dynamic switching consumes the majority of the

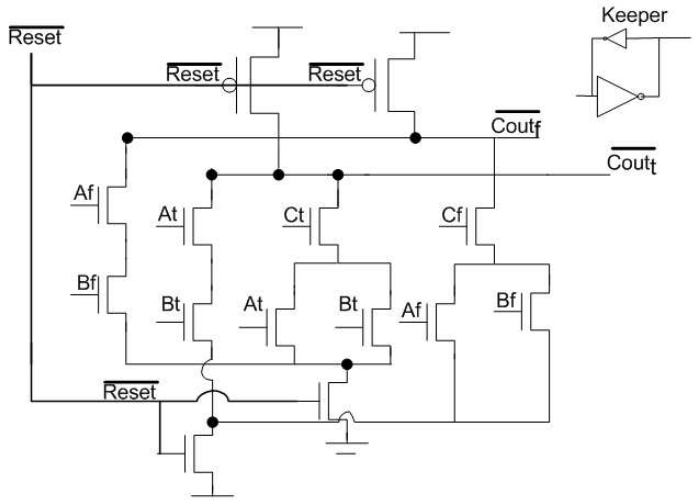


Figure 25: The pullup network of the full adder designs are replaced with reset transistors and a form of dynamic asynchronous full adders are created on version 2 of the chip.

power, and this is especially so in datapath circuits [18, 110]. Dynamic switching power is encompassed by the term $\alpha f C V_{dd}^2$ where α is the activity factor or the probability that a circuit will make a power consuming transition during the current computational step, f is the switching frequency or number of computational steps, C is the load capacitance, and V_{dd} is the supply voltage.

We will analyze the critical path of adder circuits here. For both asynchronous circuits and conventional static synchronous full adder circuits the activity factor α is $\frac{1}{4}$. This is because the probability that power will be consumed, or when charge is drawn from the power supply on a $0 \rightarrow 1$ transition, is the probability the carry-out is a 0 multiplied by the probability that the next value is a 1 or $p(0)p(1) = (\frac{1}{2})(\frac{1}{2})$.

However, it has been shown that static arithmetic circuits suffer from *switching hazards* due to spurious timing causing bit transitions to occur multiple times for a single input. This adds up to 30% onto the activity factor for static arithmetic circuits [18]. On the other hand, since asynchronous circuits are *hazard-free* and are designed to switch only a single time for every input, they do not suffer from extra dynamic energy consumption.

Further, if the load capacitance is calculated on the carry output of conventional synchronous static full-adders, the result is a load of $5 - inverter$ or the load of 5 minimum-sized inverters. The asynchronous circuits presented here on the other hand have only a $4.5 - inverter$ load. The total dynamic power consumption is summarized in Equation 8. We assume the number of computational steps and the supply voltage is the same for both circuits.

$$\begin{aligned} P_{async} &= \alpha_{async} f C_{async} V_{dd}^2 \\ P_{synchronous} &= \tfrac{13}{10}\alpha_{async} f \tfrac{10}{9} C_{async} V_{dd}^2 \\ &= \tfrac{13}{9}\alpha_{async} f C_{async} V_{dd}^2 \\ &= \tfrac{13}{9} P_{async} \\ &\approx 1.44 P_{async} \end{aligned} \tag{8}$$

Thus a first order power approximation shows conventional static arithmetic circuits consume up to 44% more dynamic power than their asynchronous counterparts. Because asynchronous logic spends a disproportionate about of time in a single state, the neutral state, leakage current can more easily be reduced as well. Many studies have shown that voltage scaling, which has been shown to allow the biggest energy savings across design paradigms [18], can be done in asynchronous logic without limitation because no timing faults or metastability results. Asynchronous logic has many power saving advantages.

7.3 Chip Features

7.3.1 Chip Architecture

To carry out the high-performance, low-power, bit-pipelining experiments with asynchronous datapath circuits, a test chip was built shown in Figure 26

The chip was fabricated in 0.5μm technology available through MOSIS due to economic factors, but the concepts that will be demonstrated are technology independent. The datapath is "asynchronously embedded" because several architectural

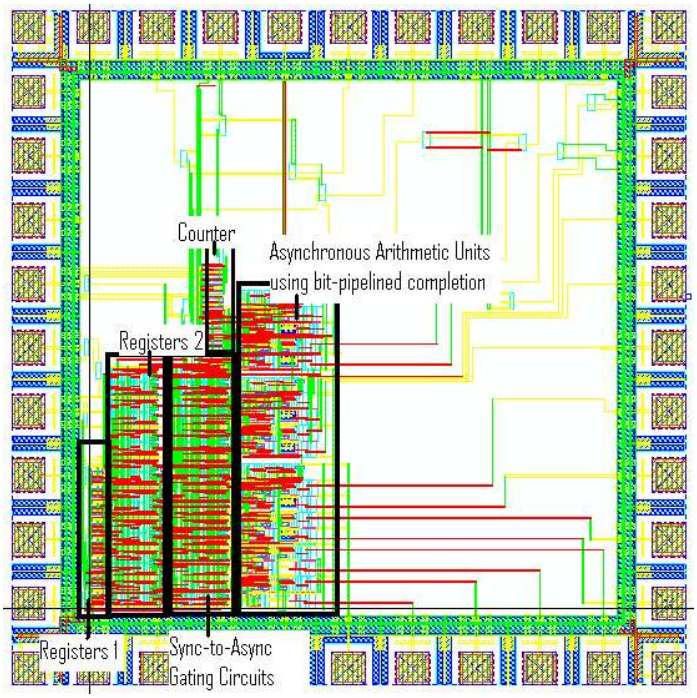


Figure 26: An asynchronously embedded 16-bit fixed point datapath testchip. Built in 0.5μm technology available through MOSIS. Multiple registerfiles are loaded from off chip via an on chip shift-register. The registerfiles feed reservation stations via multiplexers that are controlled with off-chip control signals. These units together make up the "Sync-to-Async Gating Circuits". The datapath includes an asynchronous carry-skip adder with probabilistic, bit-pipelined completion and allows multiplication with the same asynchronous full adder technology. A counter has also been added to track the performance of the functional units.

constructs were used to create a seamless conversion between a synchronous register file and asynchronous fixed-point computational units so that asynchronous arithmetic units could be "embedded" in a synchronous system.

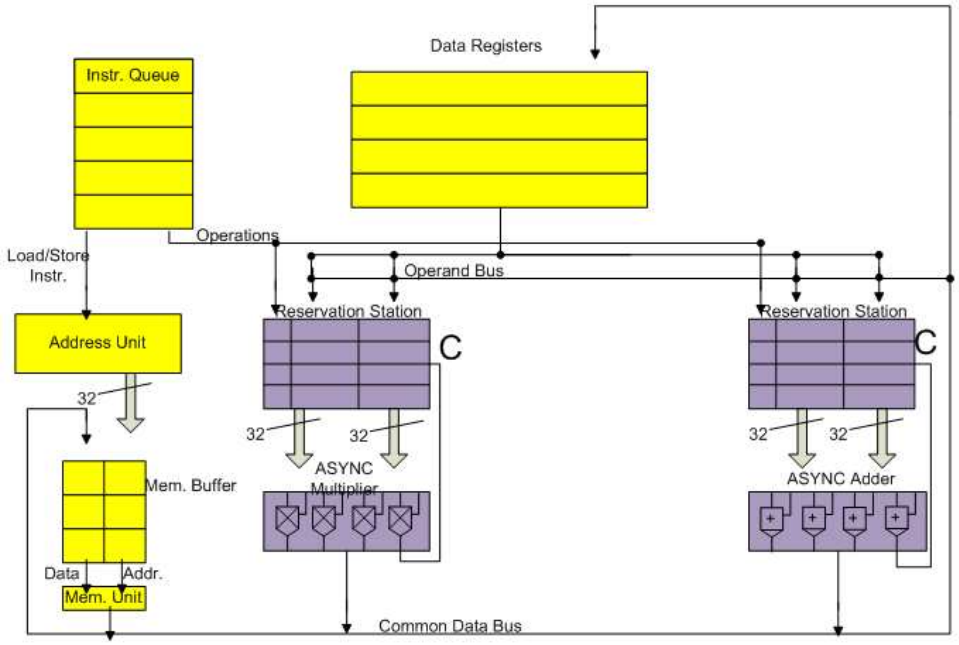
The macro-architecture behind the asynchronous embedded concept can be seen in Figure 27(a). The micro-architecture is shown in Figure 27(b). The embedded asynchronous circuits with completion-flagged reservation stations are unique to this work but the general macro-architecture is discussed in [106].

7.3.2 Bit Pipelining On Chip

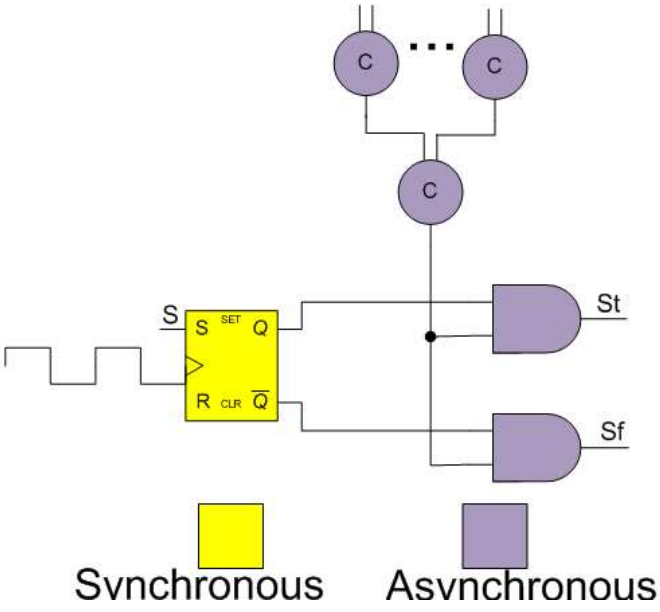
The adder shown in Figure 26 is a single 16-bit fixed point asynchronous carry-skip adder unit where the results are multiplexed such that they can be fed directly back into the reservation station, data registers or off chip to "memory". Completion detection logic is used on all bits but are multiplexed such that only the bits most likely to be in the critical path, or the most significant bits (MSB), can be used to writeback results and set the **C** flag. We coin this concept *Asynchronous Speculative Completion*. However, each bit is also multiplexed such that each bit can individually commit and flag the reservation station that it is ready for a new result. This concept is demonstrated in Figure 28.

Using the asynchronous bit pipelining technology presented in Figure 28, each bit is free to compute as fast as possible. And since each completion tree is connected to the clock input of the latches in the asynchronous reservation station, new inputs can be clocked through on a bit-by-bit basis. This cuts down on cycle times from having to wait for the worst case of all N bits having to compute as in synchronous systems to waiting for only 1 bit to compute for an N-bit datapath.

In order to commit the operations, a bit-level buffer similar to a reorder buffer is used. This phenomenon is shown in Figure 29. As soon as a column of registers has valid data and the completion tree asserts, the data is ready to be committed and is



(a)



(b)

Figure 27: Architecture behind embedded asychronous datapath concept, synchronous units are shown in yellow, asynchronous-compatible units in purple. (a) The macro-architecture of the chip is shown, which is a subset of the super scalar design referenced in Figure 13. A standard synchronous instruction queue, memory lookup, and register file is used, although the intruction queue and memory are operated off chip. The register file is operated on chip, fed by an off chip instruction and data memory system. The fixed-point and computationally intensive units are be accelerated with asynchronous design. Reservation stations are used to buffer the inputs to these units and to transfer from asynchronous to synchronous domains. When the completion flag **C** is issued new operations are written back and dispatched. (b) The transition to dual-rail logic is done by a standard single-rail register being split into two lines and logically ANDED with the completion flag **C** to create neutral values.

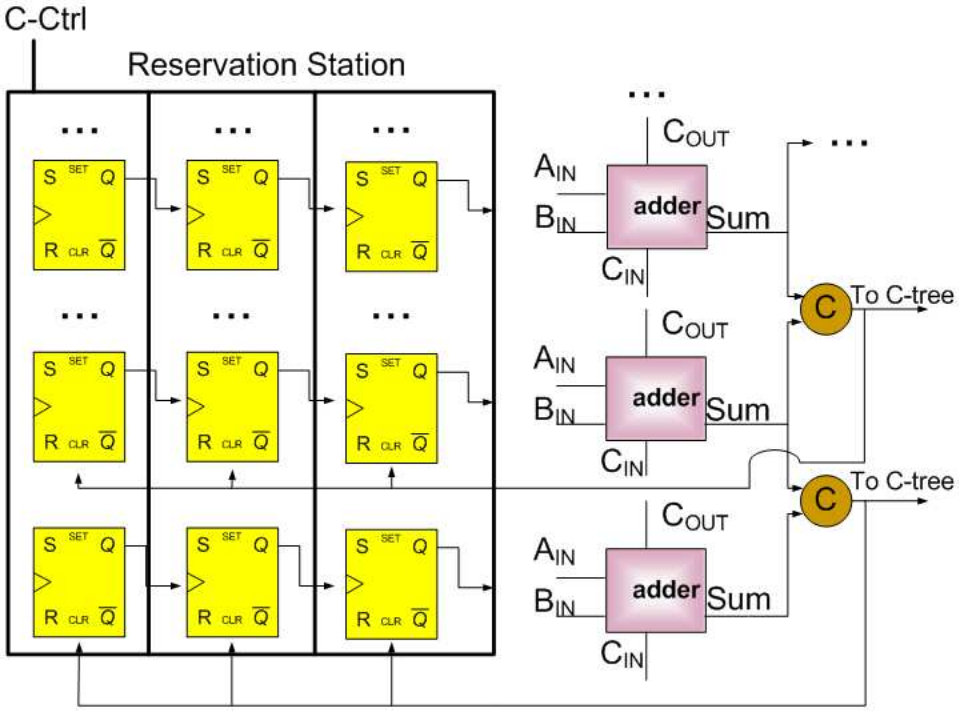


Figure 28: Schematic of fixed-point, asynchronous adder interacting with the reservation station in a bit-pipelined fashion. Bit N and bit $N+1$ feed the completion gate for bit N, which is fed back to the reservation station. Upon a successful completion, each bit is shifted up one position for bit N, allowing bit-level pipelineing. A C-CTRL line is used to flag the reservation station that the current operations have been committed and the reservation station is ready for new ops. Conversely it stalls the pipeline if the data is not ready, and invalid data sits at the input of the adders until the reservation station, which is just a FIFO register, shifts in valid data.

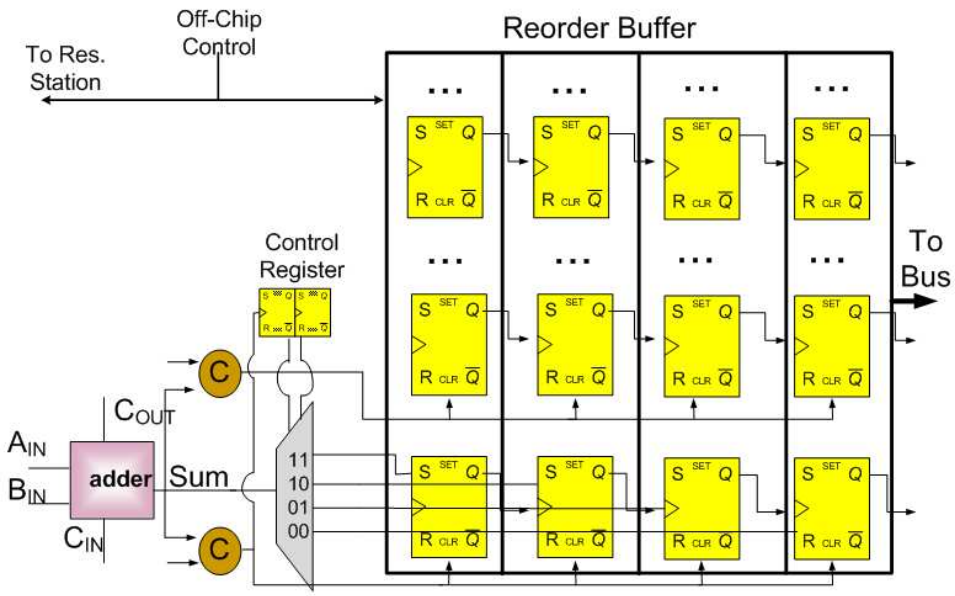


Figure 29: The reorder buffer used to commit and writeback the data at the bit-parallel level. The individual bit completion signal for bit i clocks the reorder buffer registers for bit i. As adds are commited, the result is demultiplexed to one of four columns of registers. The registers all have restorative clock buffers so that the completion signals are only loaded by one register. The completion tree and completion flag are raised in the register column signaling the column of data is ready to commit. Once the data is grabbed, a control signal resets the registers and notifies the reservation station that data can continue to be sent through.

fed to the bus. An off chip control C-CTRL signal is used to both commit the data in the commit buffer and to allow the continuation of data flow in the reservation station. If C-CTRL is not asserted, the pipeline stalls to allow all the data already in the pipeline to be processed before new data is allowed in the pipeline.

All of the registers were built with restoring clock signal buffers so that each completion gate only has to load a single register. Another energy saving advantage is seen here: all of the registers are automatically "clock gated" by this system. No register switches unless it is clocking in data.

7.3.3 Speculative Completion

Speculative completion is another experimental concept tested on this chip. Completion speculation allows faster completion (2 less transistions necessary for a 16-bit

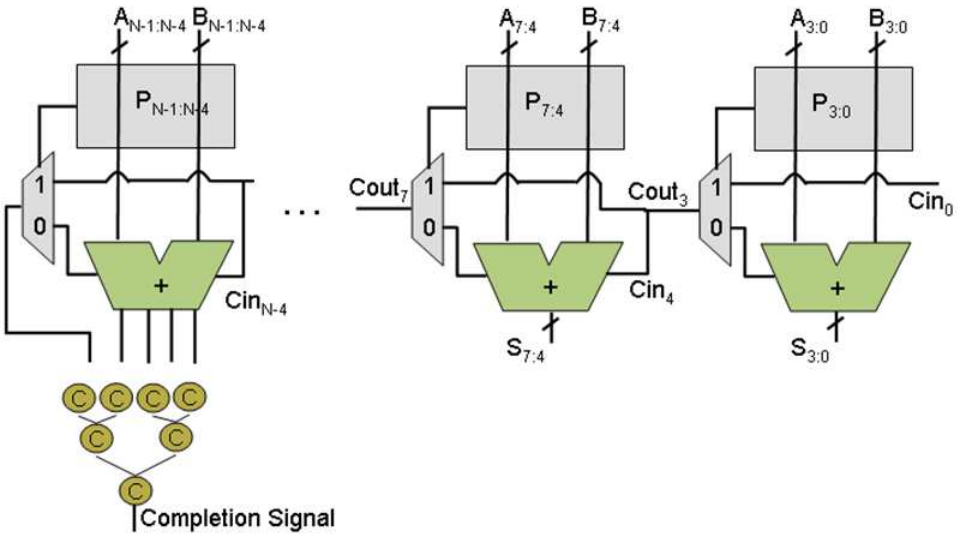


Figure 30: An asynchronous carry-skip adder with *speculative completion*. This concept allows results to commit with log$N$ less gate delays where N is the bit width of the datapath. It also reduces power usage from $\mathcal{O}(N)$ to $\mathcal{O}(logN)$.

datapath) and uses much less power ($\mathcal{O}(logN)$ instead of $\mathcal{O}(N)$) than a full completion tree paradigm. Speculative completion is defined here as using completion data for only the most significant bits, and is shown in Figure 30 for a carry-skip adder. This design takes advantage of the fact that the path with the longest completion delays will most likely occur in the critical path of the design, and thus only uses these completion signals instead of wasting valuable resources reading the completion signals for all paths.

This concept draws from probabilistic computing principles which say that by far the most likely bit position to finish last will be in the MSBs and that on the small probability that an error does occur, this will not degrade application output in digital signal processing systems [34]. This concept also draws from superscalar processing which often uses speculative instruction execution. Again, a schematic of the adder is shown in Figure 30.

This completion algorithm is significant because comparatively insignificant overhead is needed for completion detection. Further, setting the computation circuits back to neutral is also pipelined and interleaved with computations so that very little delay overhead is required and more than made up for with the accelerated bit-pipelining and average-case completion time of asynchronous logic [83].

7.4 Promising Results and Complexity Analysis

7.4.1 Complexity Analysis

Before simulation results are presented, the complexity analysis of the power and performance of the asynchronous arithmetic proposed will be discussed. The comparison to synchronous logic is inevitable, and complexity analysis of performance is one of the clearest ways to show the advantages of this flavor of asynchronous logic design over its synchronous counterpart.

In regard to performance, synchronous logic must always abide by worst-case timing rules because in synchronous logic, the timing of each calculation is defined as the amount of time to complete the worste-case critical path computation. Among $N - bit$ synchronous adders based on the ripple-carry adder such as the propagate-generate adder, the carry-skip adder, and even the carry-look-ahead (CLA) adder, the best-case performance is $\mathcal{O}(N/2) = \mathcal{O}(N)$; the time to ripple through all N bits [39]. The CLA is the fastest of the aforementioned group, but still has order $\mathcal{O}(N)$ delay, but uses the most power on the order of $3N$ for a power-delay complexity of $\mathcal{O}(N^2)$. Complexity analysis is important because it predicts how the power and performance will scale for wider bit-widths or larger data sets.

Tree adders or *parallel prefix* adders are popular because they use a tree structure to achieve $\mathcal{O}(logN)$ delay complexity such as the Brent-Kung and Kogge-Stone adders. However, due to the density of logic needed to achieve such performance efficiency, these adders are even more power hungry, burning order $\mathcal{O}(NlogN)$ power

for a power-delay complexity of $\mathcal{O}(Nlog^2N)$.

Alternatively, asynchronous adders are not bound by any timing observability constraints. For example, a ripple-carry adder on average only has $logN$ ripples, and thus when the timing is defined asynchronously, the performance complexity is $\mathcal{O}(logN)$. Manohar *et al.* has created a parallel-prefix asynchronous adder exhibiting $\mathcal{O}(loglogN)$ performance complexity *for its computation* [72]. However, the completion detection of such an N-bit elements has $\mathcal{O}(logN)$ and has $\mathcal{O}(NlogN)$ power complexity as discussed previously.

The adder presented in this work, because of its bit-pipelined completion which only detects completion on the most significant $logN$ bits and the fact that completion detection on M bits takes $logM$ time, has completion performance complexity of $loglogN$. However, because it is implemented with a carry-skip architecture its power complexity is only $\mathcal{N}$ resulting in the lowest delay-energy complexity of any known adder.

7.4.2 Simulation Results

The chip was simulated in Hspice with the help of a wrapper written in C++. Ten-thousand random input combinations were fed into the arithmetic unit and the supply voltage was scaled to study the resulting delay and energy effects. Completion gating was used to interleave each calculation with a neutral input to reset the asynchronous unit.

Hspice schematics and the C-wrapper were used to simulate the delay for Martin's asynchronous arithmetic using the same architecture but a full completion tree [81], and the results were compared to the energy and delay for carry-select and ripple-carry adders built in the same AMI 0.5 μm technology. Results can be seen in Table 2.

Results were also compared to a fast synchronous non parallel-prefix adder, a

Table 2: 16-bit Asynchronous Arithmetic Spice Simulation Results. Delay results are given for only $V_{DD} = 5$ because performance data at lower voltages was not available for the systems being compared.

	Avg. Total Delay per Input (ns)			**Avg. Energy per Calculation (pJ)**		
V_{dd}	Our Async. Carry-Skip	Martin's adder	(Fast) Sync. Carry-Select	Our Async. adder	Carry Select	Ripple Carry
5 V	1.02	2.87	4.785	5.21	20.96	7.33
4.5 V	–	–	–	3.52	16.64	4.89
4 V	–	–	–	2.18	13.08	3.004
3.5 V	–	–	–	1.41	9.904	1.643

carry-select adder, which happens to be the most energy efficient in terms of complexity. A parallel prefix adder was discounted due to the comparitively much higher energy consumption.

Delay numbers were only given for the adder at full V_{DD}, which for 0.6 μm AMI technology is 5 Volts, because these results were not available for the systems being compared. The real data matched perfectly with simulation results at full V_{DD}. Real data from silicon will be presented in the next chapter for these lower voltages.

In conclusion, the asynchronous arithmetic units and the accompanying architecture shown in this work outperform both asynchronous and synchronous counterparts in terms of both energy and performance. Theoretical performance results are presented here in terms of complexity analysis and energy calculations are given in terms of Spice simulations. Real performance data will be given in the next chapter.

A novel asynchronous adder architecture along with a novel bit-pipelined completion scheme was proposed. The power and performance analysis of not only this circuit, but of asynchronous logic compared to synchronous logic in general was given. However, because of the many advantages synchronous logic has over asynchronous in terms of control logic and memory retrieval, the authors have proposed that just an asynchronous datapath be embedded in a synchronous system. Completion gating

and buffered reservation stations with a completion flag detector were proposed to integrate synchronous and asynchronous components.

CHAPTER VIII

CHIP DATA

Version 2 of the bit pipelined asynchronous chip presented in the last chapter was fabricated and returned for testing in July 2009. Several metal mask changes were required for correct latch operation to go from Version 1 to Version 2. Several improvements to the chip architecture were also made, namely the bit pipelined architecture presented in the last chapter.

8.1 Design of Experiment

8.1.1 Experimental setup

The chip was packaged in a 40 pin DIP and placed into a bread board. The breadboard had 8 opamps for analog voltage inputs and 8 inverters for digital inputs. A Bus line was connected from a desktop workstation to the 16 analog and digital voltage lines on the breadboard and controlled via matlab commands. A voltage source was hooked up to the bread board to serve as the supply. An oscilloscope was connected to the board and was recorded via a GPIB connection.

Experimental trials run:

- To account for randomness, random inputs were shifted into the on-chip shift registers via matlab and the resulting wave forms were averaged.
- The outputs from the shift register were put through a buffer and sent off chip to be measured as were the outputs from the arithmetic units. The delay was measured using the $50\% - 50\% V_{DD}$ point from the output of the register to the arithmetic completion signal.
- This measurement was taken for incremental voltages from 550 mV up to 5 V.

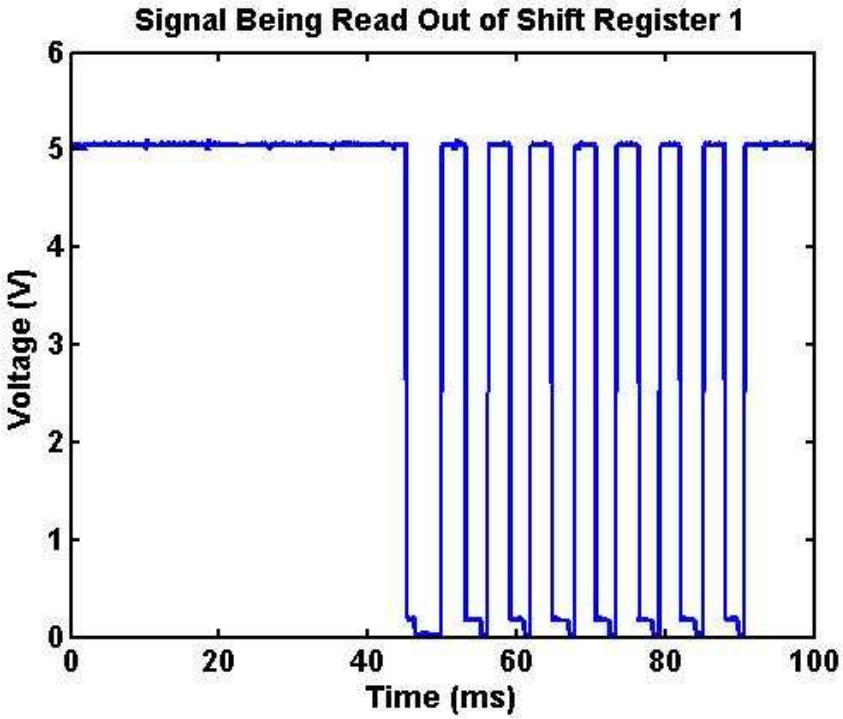


Figure 31: Data being shifted in at 200 Hz and read out. The shift register was buffered to a pin so it could be checked and tested. Note the ground bounce. The measurements were taken after the data was shifted into the register so the speed at this point is irrelevant. The data was read out of the commit buffer and the registers there at a similar speed due to limitations of equipment.

8.2 Data Plots and Analysis

The first plot shows data being clocked in at a 200 Hz rate using an off-chip clock signal generated by matlab via an op-amp on the bread board, shown in Figure 31.

Figure 31 shows the signal being clocked into the shift register at 200 Hz, an alternating pulse train was used to test that alternating 1's and 0's did not cause any problems and that every bit in the shift register was switching.

After the data was shifted into the shift registers, a scope was used to capture rising and falling edges off of the output of the registerfile and the completion signal off of the adder as shown in Figure 32.

The feature to note in Figure 32 is that the difference between the signals is unnoticable at the scale of μs.

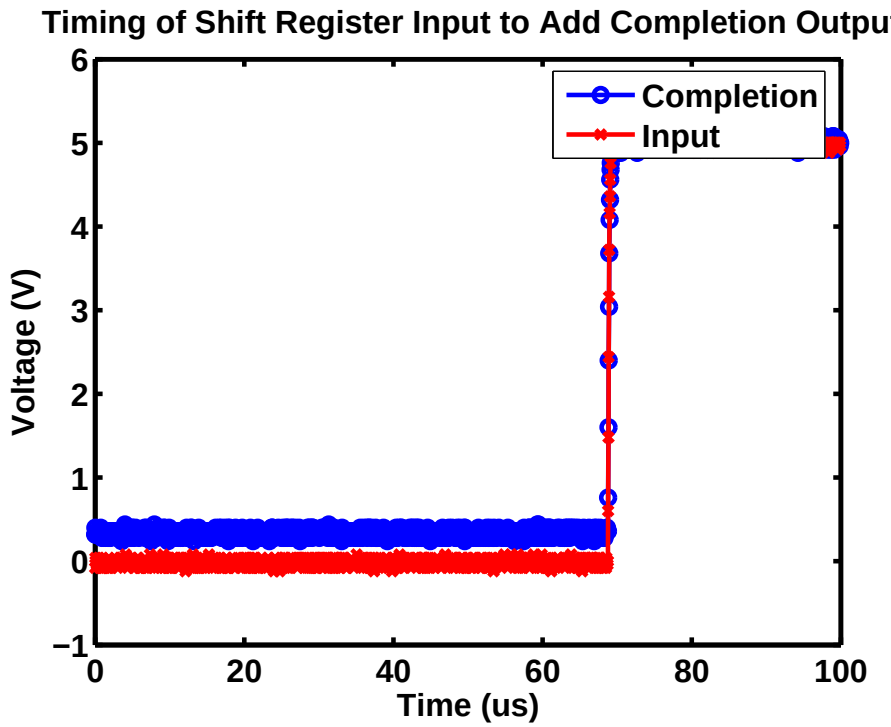


Figure 32: Screen shot of the rising edge being captured off of the completion flag off a full adder and the shift register that feeds the adder pipeline.

The next plot is a zoomed in version by about 1000X of the last image to show the $50\% - 50\%$ delay of a single bit in the asynchronous bitpipelined adder; this is illustrated in Figure 33.

The data in Figure 33 show 1 GHz clock rates for adds to be committed at $V_{DD} = 5V$ in 0.6 μm AMI technology. These results are quite encouraging given these are the fastest results reported on silicon in this technology to date, all other results of similar quality reported thus far are simulation results. Given that, the only results reported thus far that are competitive and in the same technology node are 1 GHz times for single bit additions using the much less power efficient dynamic logic family reported in [115]. The 1 GHz time reported is for single bit additions, not entire add commits. The results shown in Figure 33 is the average times for an entire 16-bit addition to be committed assuming the pipeline presented in Chapter 7 is full.

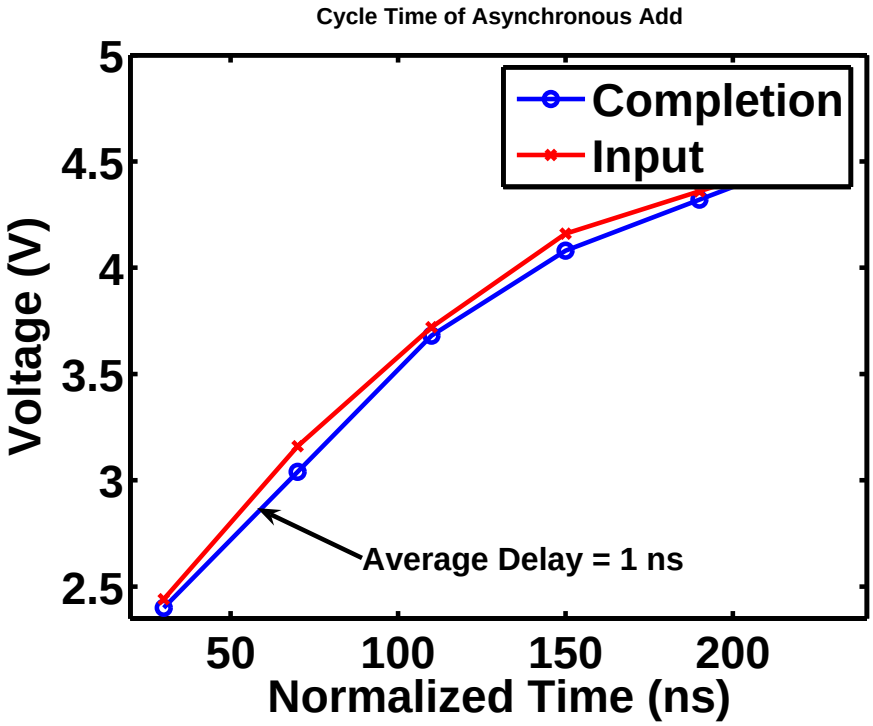


Figure 33: Zoomed in screen shot of the oscilloscope measurements from Figure 32. Note that at the $50\% - 50\%$ delay point of 2.5 V, the delay is actually less than 1 ns. On average the time from the data leaving the register to the time the completion flag was asserted was 1 ns at $V_{DD} = 5V$. Due to the bit-pipelined nature of the adder, adds were able to complete at a rate of up to 1 GHz in the 0.6 μm AMI technology. These are the fastest results reported to date for data from silicon. The precision of the measurements was limited by the oscilloscope used. The delay measured here is conservative since the signals were loaded with large capacitances so they could be read off chip.

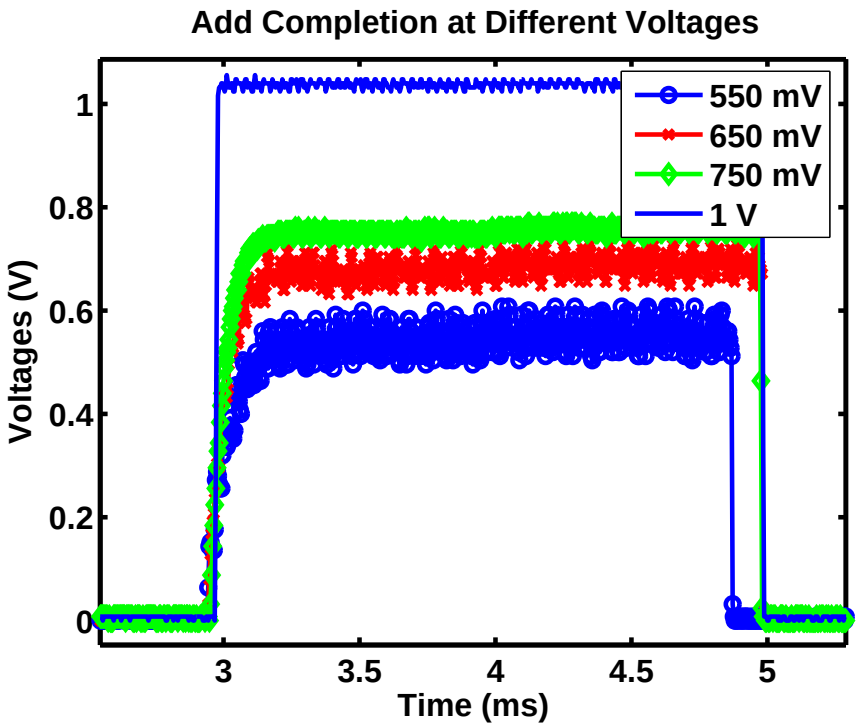


Figure 34: Plot of the completion signal response from the add operation, such as in the previous plots, at lower voltages. Note that $V_{th} = 0.57$ V for this process, thus the $550mV$ measurements were taken below threshold. This is not only significant because of the significant power savings and the fact that the chip is operational at the optimal energy-delay product point, but that the chip is operational at over an order of magnitude of range in supply voltage. Also note that the rise time gradually increases as the voltage increases as expected.

Measurements were also taken for delay at several voltages ranging all the way down to the subthreshold regime at 550 mV. These results can be seen in Figure 34.

Many studies have shown that the optimal energy-delay product point of a device occurs right at threshold. The results in Figure 34 is very encouraging then because the chip is operational at this point ($V_{th} = 0.57$ V in 0.6 μm AMI) and at subthreshold points. The rise times become very sharp at $V_{DD} = 2V_{th}$ and beyond as is shown by the sharpt rise time at 1 V in Figure 34.

Figure 35 shows the timing of the completion signal of the adder at $V_{DD} = 550$ mV.

As stated in the caption of Figure 35 the completion signal was seen fluctuating

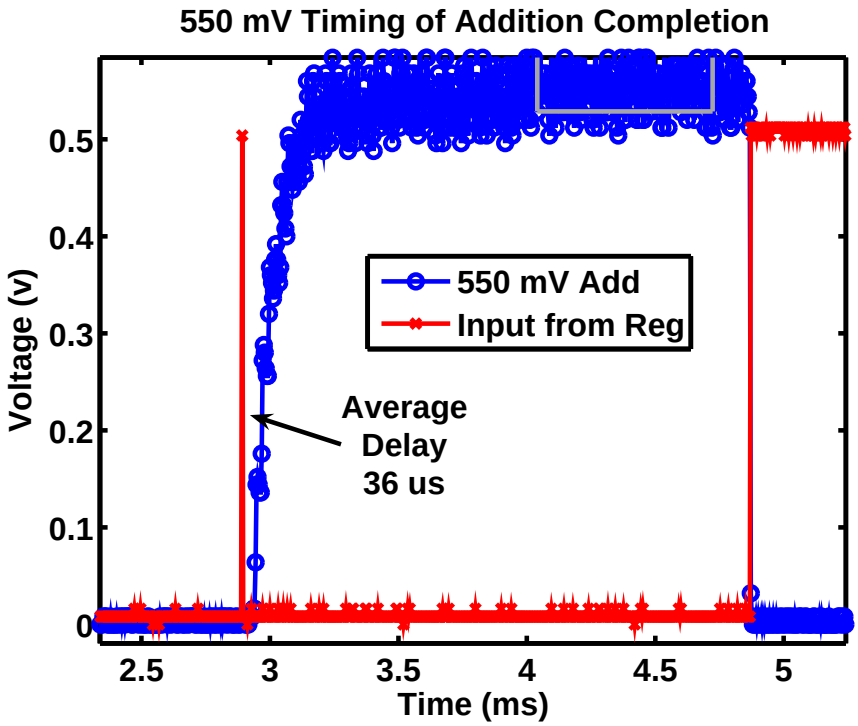


Figure 35: Plot of the delay measurement from register to completion assertation for the adder at $V_{DD} = 550\ mV$. Note that the signal is reaching its stability point at this voltage, as seen by the enormous ripples even after the signal is supposed to be stable with a high assertation. Voltage fluctuations were seen at lower voltages, but it is believed they were unmeasurable in part because of the high capacitive load the signal sees to get off chip. Interestingly, the delay was measured at 36 μs.

at lower voltages, but it is believed that it was immeasurable due to the high load capacitance.

These results have confirmed that not only can the chip operate at almost any voltage conditions, making it extremely adaptable to both energy and performance needs, but that it can operate at an unprecedented speed of 1 GHz, and commit full 16-bit adds at this speed.

To verify results at smaller technology nodes, the datapath of the chip was laid out in TSMC 0.18 μm technology and simulated for speed. Adds were able to be committed at between $3-4$ GHz. As a comparison, the Pentium 4 was fabricated in 0.18 μm technology and was able to commit adds at only 1.5 GHz.

CHAPTER IX

PRESENTING THE SYNCHRONICITY SIMULATOR AND RESULTS

A simluation environment was created and has been worked on since 2006 to allow experimentation with a variety of exotic architectures while still capturing extremely accurate circuit level voltage, timing, and energy results. The simulation structure is also used here to extrapolate the silicon results for the 0.5 μm chip to more modern technologies. Complete layout is done for 180 nm TSMC technology giving highly accurate results including parasitic and wire capacitances. Results are also given for the most modern commercially available process for which our lab could gain access to a design kit, namely a 65 nm process. Simulation models are available for even smaller transistor sizes through the Berkeley Predictive Technology Models (PTM), but these have been deemed to be highly inaccurate when compared to actual silicon (since in many cases there is no silicon comparison) and are of little value to the engineer of high moral character wishing to truthfully report accurate results.

9.1 Simulation Structure and Methodology

The simulator allows for an arbitrary combinatorial structure to be built. In this case adders, multipliers, an FIR filter, and an H.264 movie decoder datapath were built. Simulations were performed in a custom simulator designed to emulate probabilistic, asynchronous, and floating gate behavior.

The architecture is based on a 24-transistor, full adder as specified in [39]. Outputs from the full adder are left in their natural, inverted states to allow for datapath optimizations. External inverters then act as level converters between individually

biased full adders, maintaining positive logic along the datapath.

There is a design heirarchy such that a ripple-carry adder object is created for example, which is divided into smaller subcircuits comprised of either a single full adder or inverter. The subcircuit objects, in turn, are simulated behaviorally based on model data derived from HSpice simulation for the specific subcircuit. Within the simulator, inputs to each subcircuit are first evaluated to determine correct outputs, then errors are injected at a rate determined through HSpice simulation for given configuration parameters (noise RMS, supply voltage, and drive voltage). Individual subcircuits are "wired" to create the ripple-carry adder architecture and any errors that occur on the output of one subcircuit propagate to the input of adjacent subcircuit. Once a given input combination is simulated across all subcircuits the resulting output bits are compared against correct operation to determine bit error rates. Energy consumption is then determined as the sum of the energy consumption for all subcircuits comprising the ripple-carry adder for specified configuration parameters (again as determined by HSpice characterization). The simulation structure is illustarted in Figure 36

To accomplish HSpice characterization of the subcircuits necessary to implement a ripple-carry adder, layout was performed for each of the subcircuits in TSMC $0.18\mu m$ technology (layout for a complete ripple-carry adder is shown in Figure 51). HSpice simulation files were then extracted from the resulting circuits. Each subcircuit was simulated in HSpice over 1000 uniformly distributed random input combinations with supply and drive voltages varied between $0.5V$ and $2.1V$ in $0.1V$ increments. For thermally based probabilitic experiments, in the case of full adders, noise was injected at each input bit with a noise RMS of $0.2V$ (shown in Figure 38). Subcircuits were then evaluated for energy consumption, probability of correctness, and delay to realize characterization data necessary for the simulator.

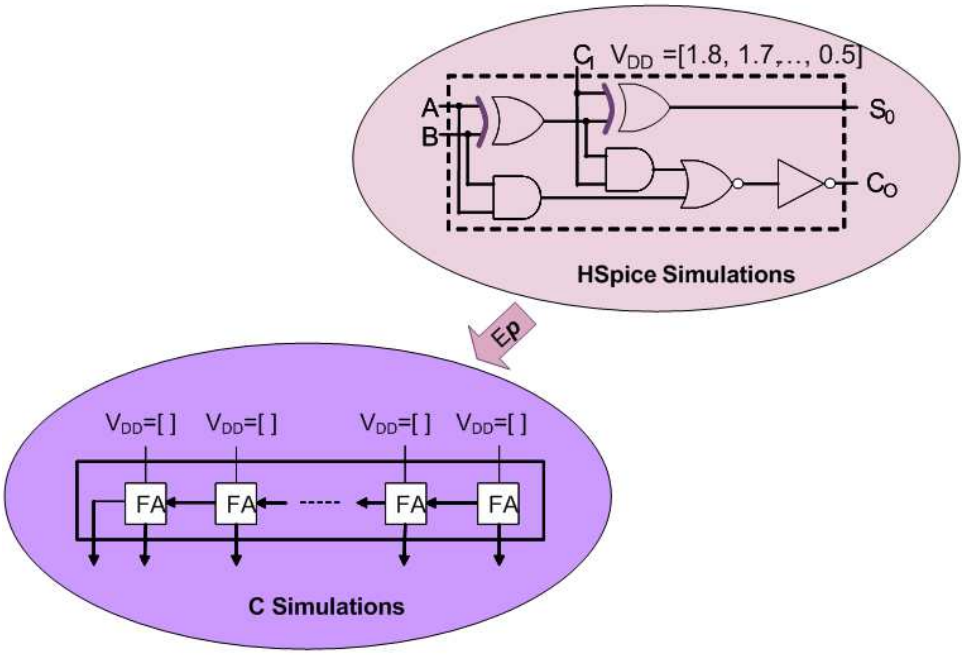


Figure 36: Pictoral example of the simulation structure used. Circuit level data for energy and delay for different voltages and noise levels were collected at the transistor and subcircuit (full adder in this case) level. Several levels of heirarchy are built around this data and a behavioral or transaction level simulator is built. Abstractions allow for an arbitrary datapath architecture to be simulated with exotic technologies such as probabilistic, asynchronous, and floating gate technologies.

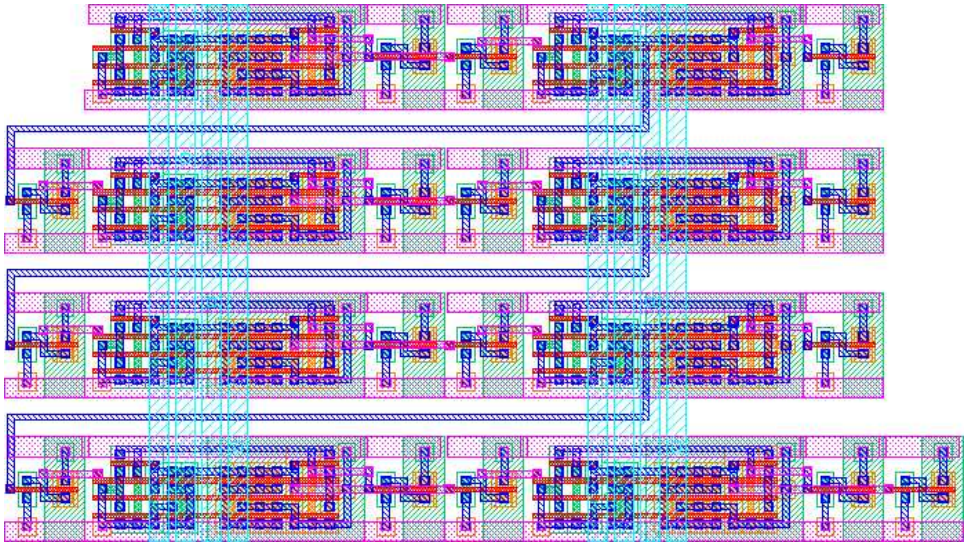


Figure 37: Circuit layout for an 8-bit, ripple-carry adder implemented in TSMC $0.18\mu m$ technology. The resulting adder is comprised of individual full-adder and inverter subcircuits and layout is performed to allow for voltage biasing of individual bit positions.

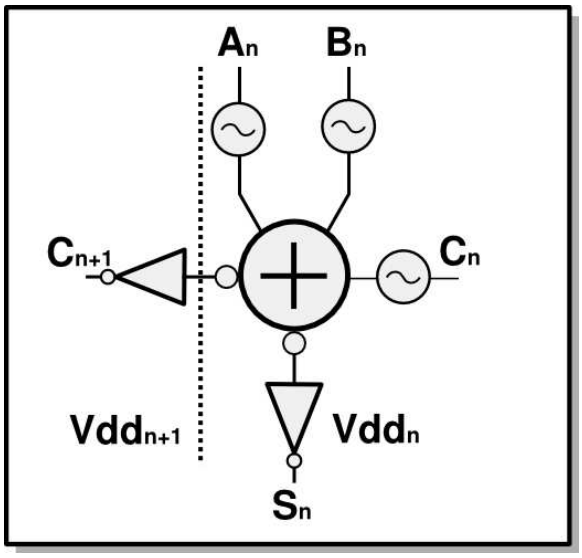


Figure 38: PCMOS full adder with noise injected at subcircuit inputs (A_n, B_n, and C_n). External inverters along the carry chain serve as level converters, minimizing static current flow to a pair of transistors at the interface across biased bit positions.

9.2 Simulation Results

The asynchronous datapath architecture that was built, fabricated, tested, and presented here that is described in Chapter 7 was built in the simulation structure. Circuit layout was done in a 180 nm process and simulated results inclusive of parasitic and wire length capacitive effects were taken into account.

Results of the simulation showing performance comparisons between the bit pipelined parallel asynchronous 16-bit adder and a synchronous 16-bit adder are shown in Figure 39.

The 0.18 μm simulations of the bit-asynchronous adder produced results that were an order of magnitude faster at all voltage levels than a custom built carry-skip synchronous adder. The bit-asycnhronous adder performed at a top speed of 3 GHz at this technology node, which is over twice as fast as the commercially built Intel Pentium 4 add commit unit built in dynamic logic. Note that this version of the bit-asynchronous adder is built in low power CMOS technology and other sections show that this adder built with dynamic logic attains much higher speeds.

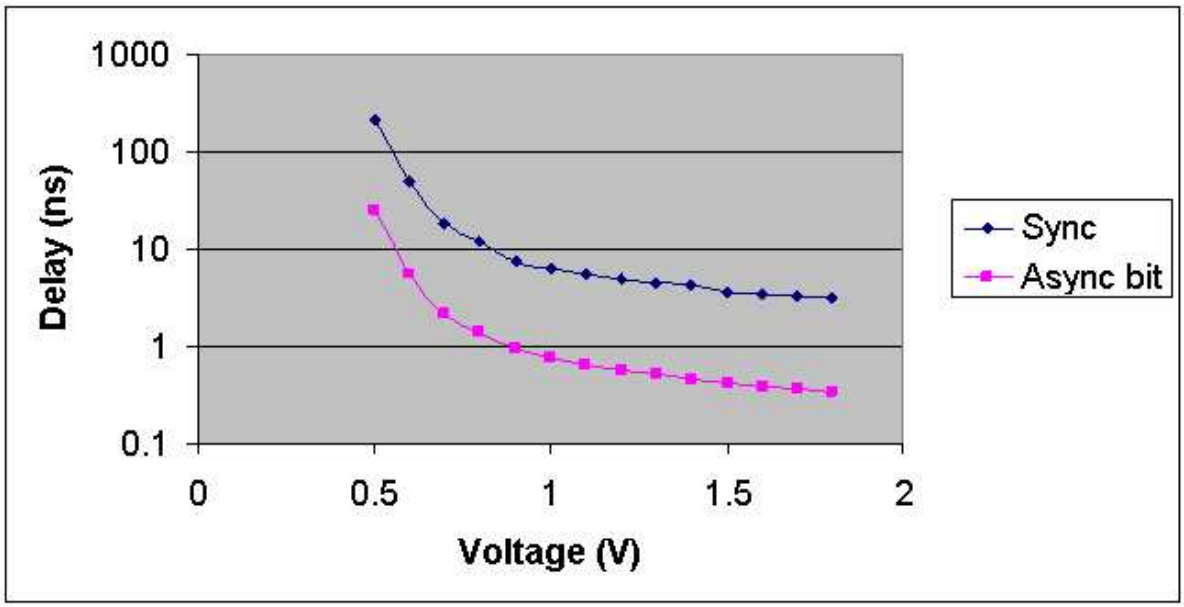


Figure 39: Delay vs Voltage on a semilog axis of a 16-bit adder built with both a synchronous carry-skip sytle and the bit pipelined parallel asynchronous architecture presented in this dissertation. Both versions were designed and laid out in 180 nm technology. Note that the bit-asynchronous architecture is an order of magnitude faster than the synchronous counter part, with the bit-asynchronous version operating at a top speed of 3 GHz. This speed also compares quite well to commercially available chips built in the same technology with the Pentium 4 committing adds at about half the bit-asynchronous speed.

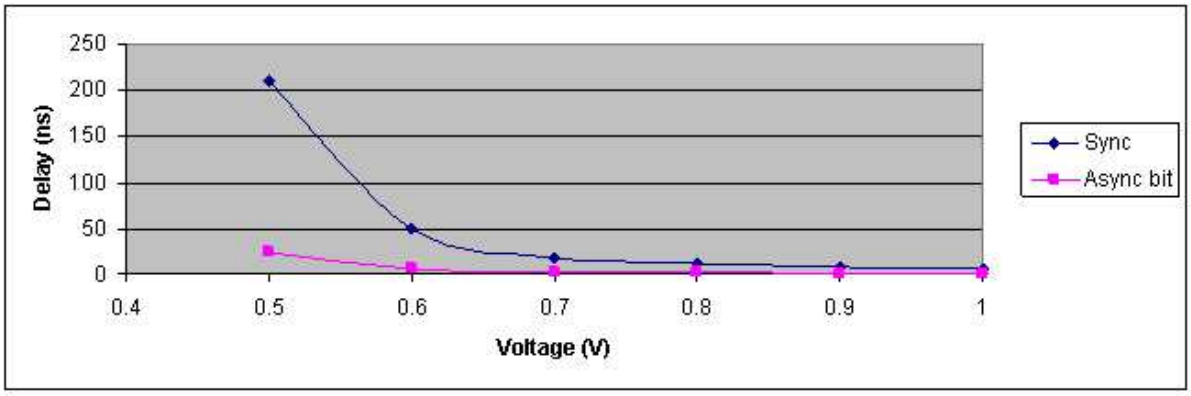


Figure 40: Delay vs Voltage on a linear plot of a 16-bit adder comparing the bit-asynchronous vs a fast synchronous architecture. This plot shows the delay for a voltage range of $0.5-1.0$ Volts to show that the bit-asynchronous design also performs exponentially better at low voltages. The nominal supply voltage for this process is 1.8 Volts.

A linear plot in Figure 40 shows the large advantage in speed the bit-asynchronous architecture has over the synchronous architecture at low voltages.

Figure 40 shows the exponential gains is performance at lower voltages of the novel bit-asynchronous architecture over a fast carry-skip synchronous architecture.

The simulation environment also allows precise energy numbers to be measured for both architectures. The energy delay product (EDP) for computations is a direct measure of efficiency of a design. The EDP calculation for the a fast synchronous design and my bit-asynchronous design is shown in Figure 41.

The EDP plot shows the bit-asynchronous design is more efficient at every voltage than the synchronous design. Notably, it becomes exponentially more efficient at lower voltages. A zoomed in version of this plot is shown in Figure 42.

At the nominal supply voltage of 1.8 Volts we see from Figure 42 there is an order of magnitude improvement in EDP when going from a bit-asynchronous architecture to a synchronous architecture. Further, the bit-asynchronous EDP trend stays flat. This tells us that the bit-asynchronous architecture is equally efficient at a wide range

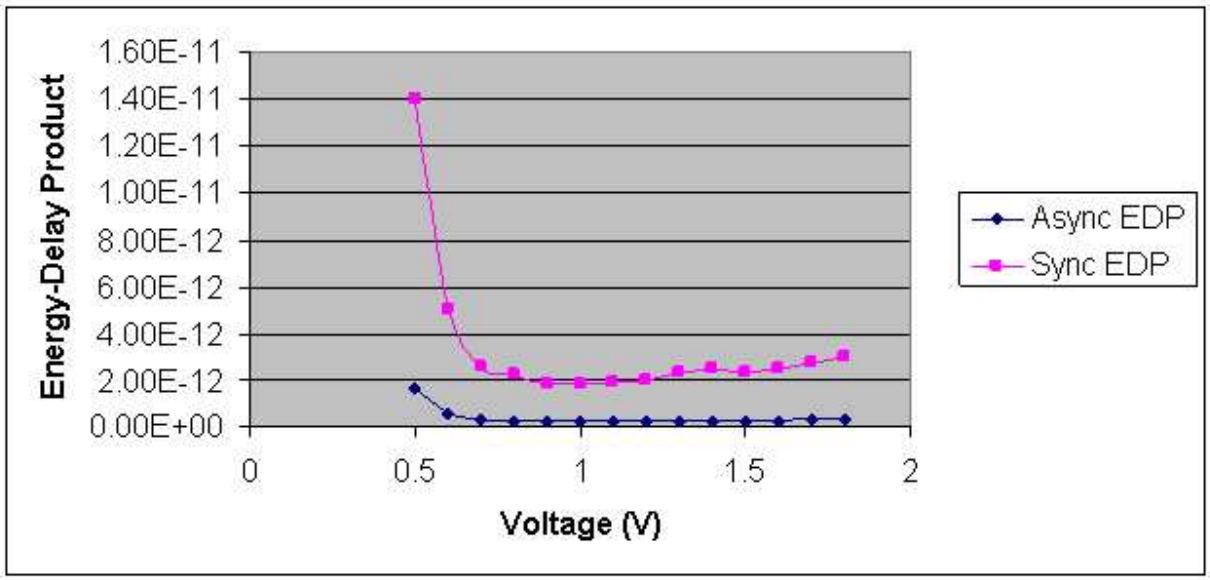


Figure 41: Energy-Delay-Product (EDP) vs Voltage of a 16-bit adder implemented in the bit-asynchronous architecture presented here and a fast carry-skip synchronous architecture for a synchronous comparison. In this plot, of note is the exponential behavior the synchronous design has: as voltage is lowered, the EDP spikes for the synchronous design where the bit-asynchronous design stays relatively flat. Thus it becomes exponentially more efficient than the synchronous design as voltages are lowered. Also of note is the bit-asynchronous design is more efficient at every voltage level.

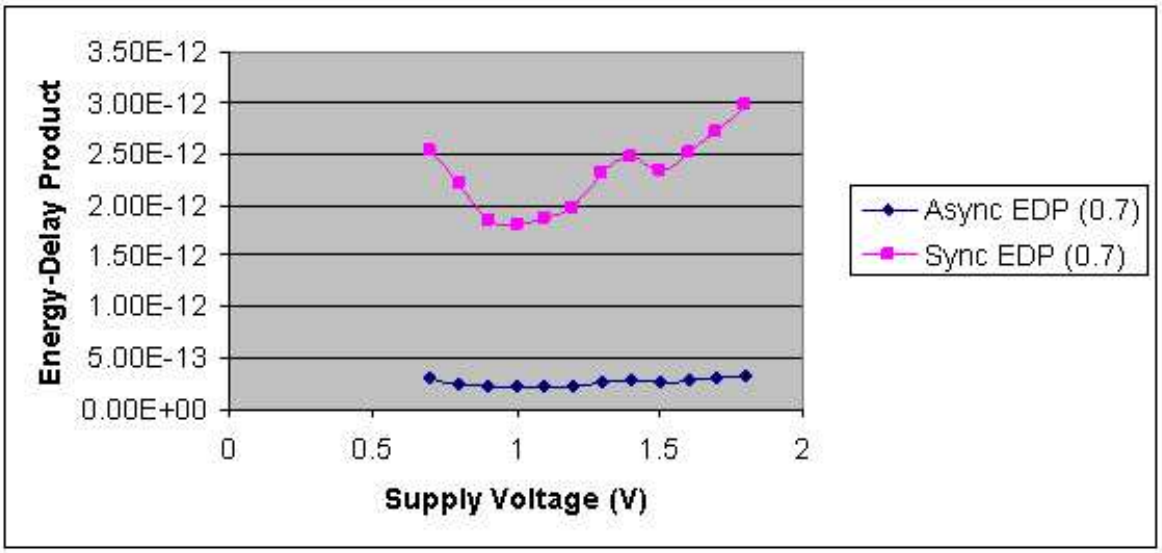


Figure 42: Zoomed in Energy-Delay-Product for the 16-bit adder implemented in bit-asynchronous and a fast synchronous architecture. Results from 0.7 Volts and below are removed to allow a zoomed in view. There is about a 10X improvement in EDP of the bit-asynchronous design over its synchronous counterpart. The energy consumption is similar for the asynchronous and synchronous architectures thus the order-of-magnitude gap in delay dominates the EDP measure. Note that the synchronous design gets more efficient as the supply voltage is scaled towards the threshold voltage V_{TH} to a certain point and then degrades with further voltage scaling while the asynchronous design continues to slightly improve at all voltages. Between this fact and the fact that the bit-asynchronous architecture can easily move into sub-threshold regimes argues for bit-asynchronous design when efficiency and low power is concerned.

Table 3: 16-bit Adder Results with Architectures of Varying Synchronicity

	Architecture Type			
	Bit Piped Async.	**Fast Sync.**	**Async. State--of-the-Art**	**Pentium 4**
Time to Commit (ns) $V_{DD} = 1.8$	0.33	3.084	1.063	0.66
Time to Commit (ns) $V_{DD} = 0.5$	24	210	76	–
Top Speed (GHz)	3	0.325	0.940	1.5
Energy-Perf. Prod. $V_{DD} = 1.8$ (pico-J-ns)	0.326	2.97	1.02	–

of voltage levels allowing for a voltage scaling implementation where optimal effiency is guaranteed at all voltages.

The results are summarized in Table 3

Table 3 shows several remarkable comparisons. Note that several blank entries are listed for the Pentium 4.[1] First, the bit-asynchronous architecture presented here can commit additions must faster than the fast synchronous adder built by the author for comparison, by a factor of almost $10X$, it is $3.11X$ faster than state-of-the-art asynchronous designs (before this work), and is even about twice as fast as the fastest commercially available Pentium 4 Processor. The Pentium 4 was chosen because it is one of the most highly regarded processors in terms of speed ever built on a 180 nm procees, the documentation behind these numbers can be found here [55].

Another interesting comparison from Table 3 is the low power comparison at $V_{DD} = 0.5$. The bit-asynchronous architecture comes in at a healthy 24 ns while the fast synchronous architecture clocks in at 0.21 μs! The conversation switches from nanosecond times to microsecond times when one moves from asynchronous to synchronous designs with low power, 180 nm designs. Interestingly, the Pentium 4

[1]The Pentium 4 could not operate at $V_{DD} = 0.5$ Volts. Energy information is also not available for just the add unit.

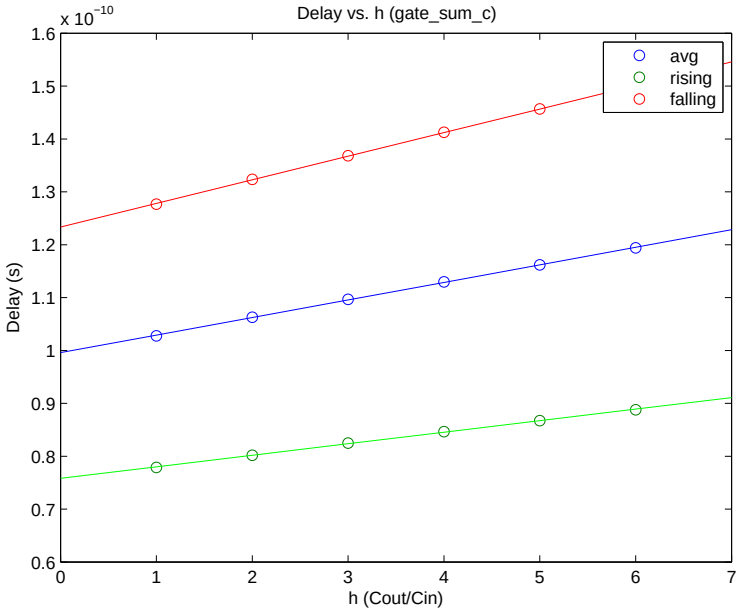


Figure 43: Delay vs Capacitive Load $h = \frac{C_{out}}{C_{in}}$ for a half cycle of one of the 2-D bit pipelined asynchronous function blocks with a 180 nm processing resulting in an average delay of about 0.21 ns or an average speed of about 4.76 GHz.

chip can not even operate at such low speeds because of the delecate deep pipelining used to achieve such speeds [55].

9.3 Advanced Process Simulation Results

Simulations are also presented in an advanced and well tested commercially available 65 nm process node to give results for modern transistor technologies.

Simulations were done in a commerically available 180 nm with dynamic logic this time instead of the static logic presented in previous simulation results. In Figure 43 we see the delay results through a half cycle of one of the 2-D bit pipelined asynchronous function blocks with a 180 nm process resulting in an average delay of about 0.21 ns or an average speed of about 4.76 GHz.

Figure 44 shows the voltage swing characteristics of the dynamic circuit for different loads (h values).

In Figure 45 we see the delay results through a half cycle of one of the 2-D bit

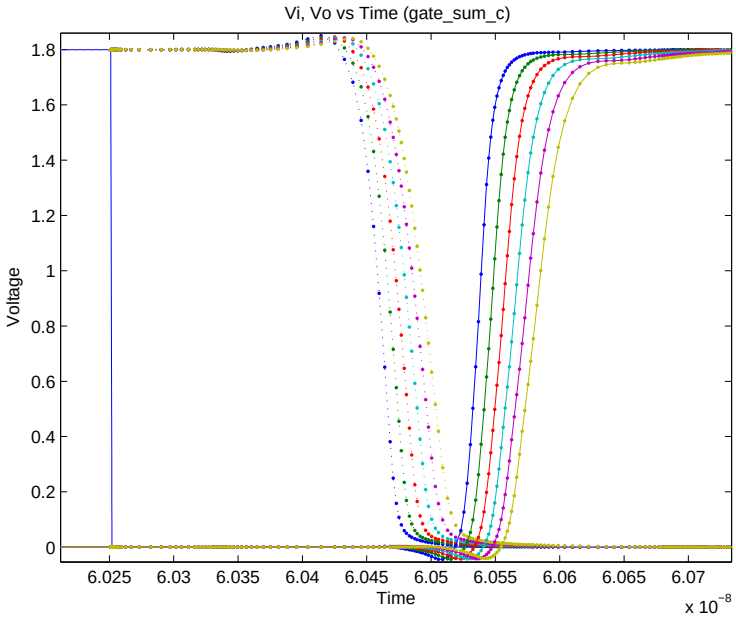


Figure 44: Voltage vs Time of the input and output voltages for a function block from the 2-D bit pipeline asynchronous system for a 180 nm process.

pipelined asynchronous function blocks with a 65 nm process resulting in an average delay of about 90 ps or an average speed of about 11.1 GHz.

Figure 46 shows the voltage swing characteristics of the dynamic circuit for different loads (h values).

As a comparison, other works show a maximum speed of additions being completed at about 4 GHz in the high end IBM and Pentium chips. Academic results that have been reported are all slower than this.

9.4 Multiplication Results

An array multiplier was built using the same technology, a schematic is shown in Figure 47.

Not only were the multiplies able to commit at rates similar to the additions, but using the probabilistic technology spoken about in the next chapter, they could sped up to $2X$ the synchronous speed while reducing power by 38%.

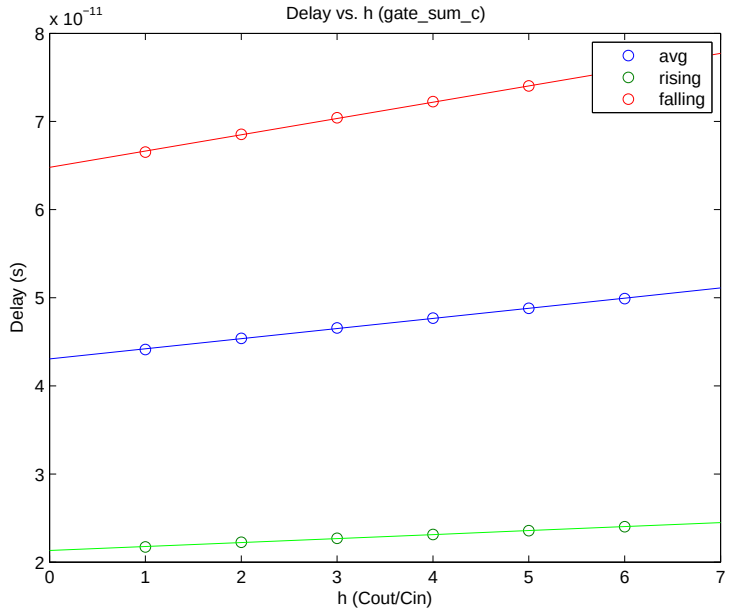


Figure 45: Delay vs Capacitive Load $h = \frac{C_{out}}{C_{in}}$ for a half cycle of one of the 2-D bit pipelined asynchronous function blocks with a 65 nm processing resulting in an average delay of about 90 ps or an average speed of about 11.1 GHz.

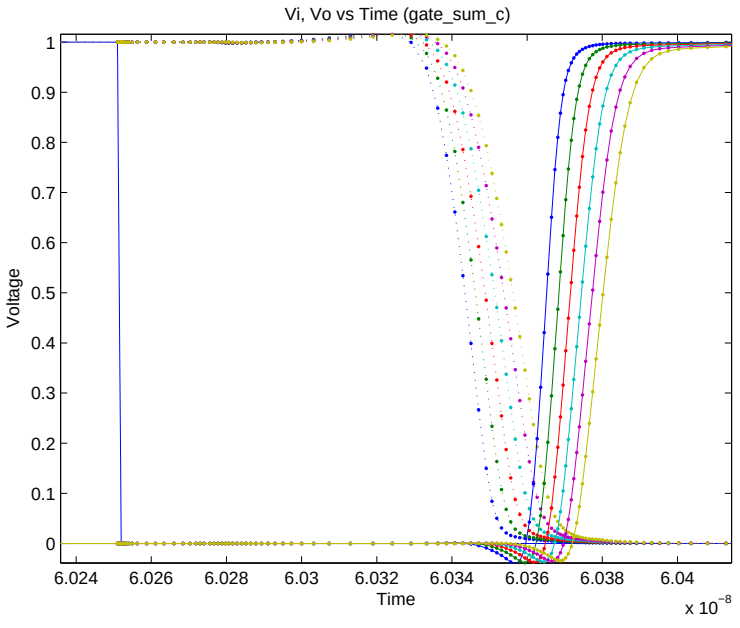


Figure 46: Voltage vs Time of the input and output voltages for a function block from the 2-D bit pipeline asynchronous system for a 65 nm process.

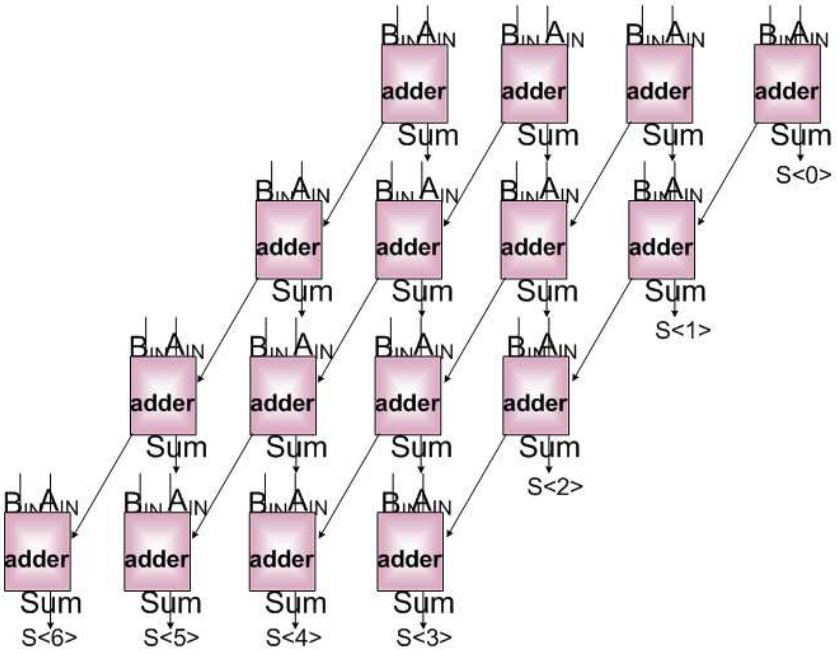


Figure 47: Schematic of array multiplier used in experiments. Array Multiplier was able to commit at approximately 600 MHz is 180 nm technology without pipelining and with static CMOS. Including the bit pipelining, the multiplies were able to commit at the same rate as the additions, assuming the pipeline was kept full.

CHAPTER X

USING PROBABILISTIC DELAY IN ASYNCHRONOUS CIRCUITS

Probabilistic CMOS technology or PCMOS , which has shown possibilities of an ultra-low-power solution particularly well suited to DSP systems, is used to build a low-power, fixed-point datapath. Novel results are presented here showing that PCMOS computes with *less noise and less power* than a reduced bit width, or *truncated*, deterministic datapath due to quantization noise injected by limited-precision, fixed-point processing. Simulation results derived from physical layout of a PCMOS fixed point datapath are given showing energy savings of up to 3X can be achieved while still meeting quantization noise constraints. Further, analysis shows that this result is more fundamental than the case study presented here.

10.1 Introduction

As energy continues to remain a chief design constraint in the signal processing community, reduced precision fixed point arithmetic has become a viable and interesting design solutions for low power. Recent works have demonstrated many signal processing applications, particuarly image processing, do not need perfect or precise computations for acceptable and even desirable results.

To this end, Probabilistic CMOS (PCMOS) has been suggested as a low-power, computing solution. As a technique, PCMOS reduces operating voltages such that the circuits are subject circuits to bit errors due to delay. The tradeoff is energy savings not otherwise realizable. Previous work applying PCMOS to arithmetic was presented in [35]. The authors use a technique dubbed BIVOS where voltage scaling is biased

to low-order bit positions. As a result, bit errors are then biased to low-order-bit positions and the impact of any bit errors is minimized.

PCMOS has the potential to significantly decrease the power consumed in signal processing systems. However, a question often raised is, 'how do the power savings compare with those realized by using fewer bits in a fixed-point representation?' We address this question for 16-bit PCMOS arithmetic on a simulated 0.18 micron process and show that for a given allowable noise level, PCMOS is more power efficient.

Presented here are several novel and interesting contributions:

- A comparison between PCMOS and reduced bit width fixed point implying a design win for PCMOS depending on the energy needs.
- A "cutoff" point whereby depending on the noise or energy constraint, above this bit width PCMOS is the better choice, below is reduced precision fixed-point.
- An optimal BIVOS voltage biasing algorithm is discovered for optimizing noise power per energy in fixed point arithmetic.
- Up to 3X energy savings is achieved with acceptable noise levels for *both* PCMOS and reduced fixed point over a full precision baseline case of 16-bits.

10.2 PCMOS Fixed Point Analysis

A PCMOS fixed point hardware unit is defined as a functional unit such that the N-bit output has an associated probability vector $\mathcal{P} < p_0, p_1, ..., p_{N-1} >$ corresponding to bits $\mathcal{B} < 0 : N-1 >$ whereby each p_i is the probability that bit i is computed correctly and follows $0.5 < p_i < 1$. The advantage of PCMOS is that each bit i whereby $p_i < 1$ has an associated energy savings. We will see that these p_i's are not independent, but can also be individually tuned which will turn out to be a powerful result.

A digital bit is correct in this context if the output voltage is at the correct level at the time when the bit is latched. This condition is defined more formally in

Equation 9:

$$\begin{aligned} &\text{at time } \tau = t_{pulse} - t_{setup} \\ &\mathcal{B} < i >= \begin{cases} 1 & : \quad V_{o,i} > \frac{V_{dd}}{2} \\ 0 & : \quad V_{o,i} < \frac{V_{dd}}{2} \end{cases} \\ &\text{where } t_{pulse} \text{ is time at which clock pulses} \\ &t_{setup} \text{ is latch setup time} \\ &\text{and } V_{o,i} \text{ is output voltage at bit i} \end{aligned} \tag{9}$$

There are two fundamental phenomena that would cause digital bits to violate Equation 9 in our fixed-point signal processing system, both of which become factors as we reduce supply voltage V_{dd} to reduce power. The more dominant effect is delay: as supply voltages are lowered to save power, delay especially in asynchronous systems is shown to increase by several orders of magnitude.

In the speculative asynchronous completion system proposed in this work in Chapter 7 excessive delay will cause errors and violate the condition Equations 9.

Image and audio processing are uniquely suited as applications where some noise is allowable. They exhibit a sliding scale of quality with power, but are extremely *resilient* to noise compared to other applications. One must still be careful to stay under the quantization noise-power level. If this barrier is broken an H.264 image such as in Figure 48 may result.

Thus a voltage biasing scheme has been developed such that lower order bits that affect the answer exponentially less are assigned lower voltages and higher order bits are assigned higher voltages. We call this Biased Voltage Scaling or BIVOS . The voltage is dropped at each bit by a small value, less than the threshold voltage V_t so as to not cause leakage currents in subsequent gates and so no level shifters need to be added causing power overhead. This leakage phenomenon is illustrated in Figure 49

Figure 48: Example of how image processing applications can slowly be degraded with noise unlike other applications which fail completely. This is the output of an H.264 image with an *extremely* noisy datapath.

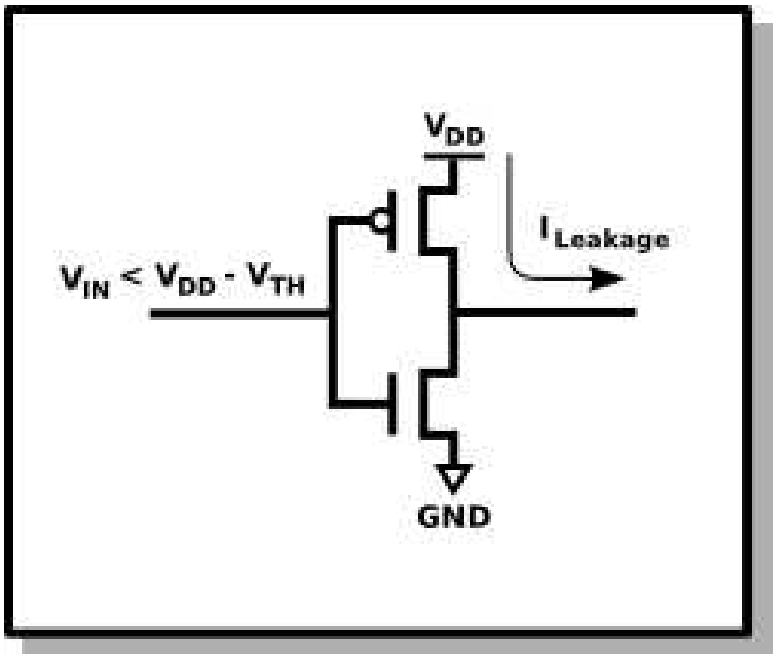


Figure 49: If a voltage drop occurs between two logic gates greater than the threshold voltage V_t the pFET transistor will turn ON when it is designed to be OFF causing exponentially greater leakage currents.

10.2.1 Delay Faults

As discussed previously, the main condition that can cause failure is a *delay fault* whereby the delay to calculate a new value through a circuit path i is longer than the clock period minus the latch setup time, $\delta_i > t_{clk} - t_{setup}$.

The delay through a logic gate is based on how fast charge can be moved on and off the capacitive load of the gate thus for a first order derivation assuming ON transistors are above threshold ($V_{dd} \gg V_t$) and saturated we have the relations for delay in Equation 10.

$$\delta_{CMOS} = \frac{V_{dd} C_L}{I} \tag{10}$$

$$\delta_{CMOS} \propto \frac{V_{dd}}{(V_{dd} - V_t)^2} \tag{11}$$

$$= \frac{1}{V_{dd}} \text{ where } V_{dd} \gg V_t \text{ In Above } V_t \tag{12}$$

$$\delta_{CMOS} \propto \frac{V_{dd}}{e^{V_{dd}}} \text{ In Sub} - \text{V}_\text{t} \tag{13}$$

Thus the delay for a single bit addition for example is quite deterministic depending only on the capacitive load C_L, the supply voltage V_{dd} and the threshold voltage V_t. However in a multi-bit adder, the length of the circuit path in question, δ_i is *data dependent* and follows the probabilistic distribution of the input set. For example the well known relationship in an N-bit adder is that the average carry chain is $\sqrt{N}$ for a uniformly random input set. A partial carry-chain addition example is shown in Figure 56.

So while scaling down the supply voltage causes a linear (or exponential in subt-V_t) increase in delay, this can be taken advantage of because of the fact that an arbitrary δ_i is rarely longer than the *critical path* of the circuit.

10.3 Architecture used and Simulation Methodology

To evaluate the effectiveness of PCMOS as compared to reduced precision arithmetic, we consider a ripple-carry adder. The architecture is based on a 24-transistor, full

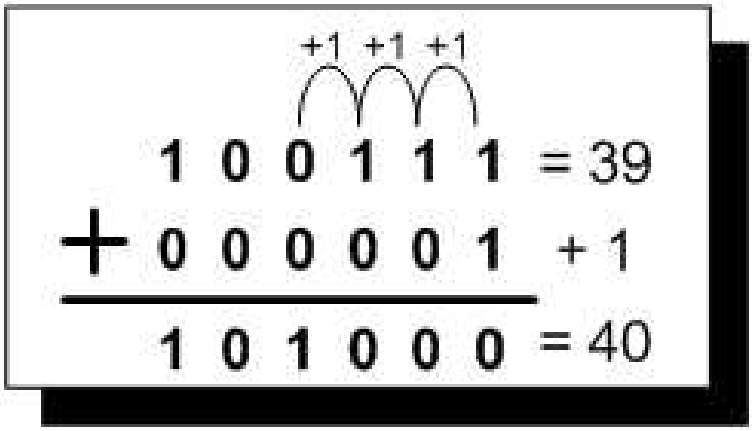


Figure 50: Addition example. Note that the carry chain does does not excite the critical path in the addition. Thus the delay seen at any given bit is no more than $4 * \delta_{FA}$ where δ_{FA} is the delay of a single Full Adder.

adder as specified in [39]. Outputs from the full adder are left in their natural, inverted states to allow for datapath optimizations. External inverters then act as level converters between individually biased full adders, maintaining positive logic along the datapath.

Simulation of the ripple-carry adder was performed in a custom PCMOS simulator designed to emulate probabilistic behavior. The adder is divided into smaller subcircuits comprised of either a single full adder or inverter. The subcircuits, in turn, are simulated behaviorally based on model data derived from HSpice simulation for the specific subcircuit. Within the PCMOS simulator, inputs to each subcircuit are first evaluated to determine correct outputs. Individual subcircuits are "wired" to create the ripple-carry-adder architecture and any errors that occur on the output of one subcircuit propagate to the input of adjacent subcircuits. Once a given input combination is simulated across all subcircuits the resulting output bits are compared against correct operation to determine bit error rates. Energy consumption is then determined as the sum of the energy consumption for all subcircuits comprising

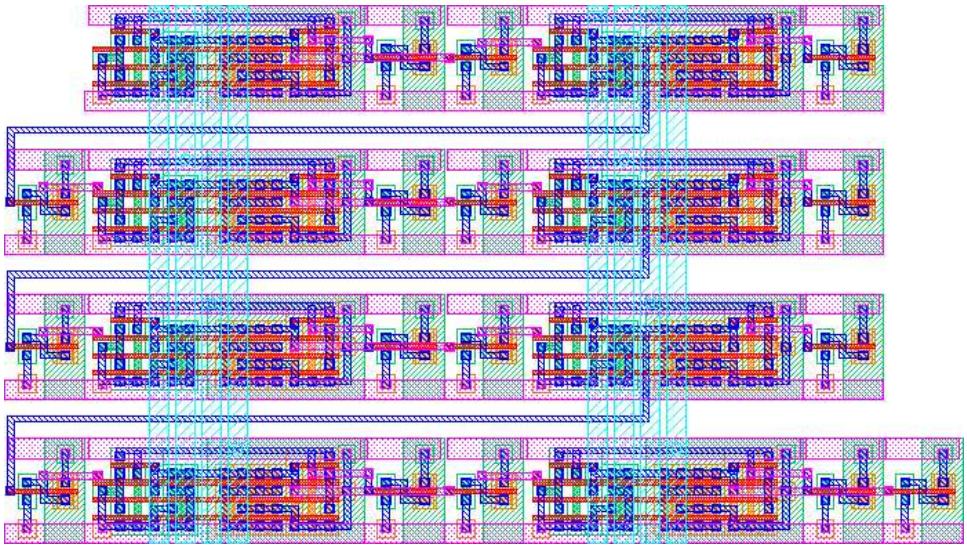


Figure 51: Circuit layout for an 8-bit, ripple-carry adder implemented in TSMC $0.18\mu m$ technology. The resulting adder is comprised of individual full-adder and inverter subcircuits and layout is performed to allow for PCMOS biasing of individual bit positions.

the ripple-carry adder for specified configuration parameters (again as determined by HSpice characterization).

To accomplish HSpice characterization of the subcircuits necessary to implement a ripple-carry adder, layout was performed for each of the subcircuits in TSMC $0.18\mu m$ technology (layout for a complete ripple-carry adder is shown in Figure 51). HSpice simulation files were then extracted from the resulting circuits. Each subcircuit was simulated in HSpice over 1000 uniformly distributed random input combinations with supply and drive voltages varied between $0.5V$ and $2.1V$ in $0.1V$ increments. Subcircuits were then evaluated for energy consumption, probability of correctness, and delay to realize characterization data necessary for the PCMOS simulator.

10.4 Results and Comparison to Truncated Fixed Point

As specified in Section 10.3, a 16-bit, fixed-point, ripple-carry adder was simulated using the previously mentioned PCMOS simulator. The resulting *noise power* per energy was then compared between the PCMOS adder and deterministic adders of

different bit widths to identify the better design solution for low power implementation: a full bit width probabilistic datapath or a reduced precsion datapath subject to quantization noise. To compare different adder designs, the standard definition for quantization noise power was used as in Equation 17.

$$\sigma_e^2 = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x^2 \frac{1}{\Delta} dx \tag{14}$$

$$= \frac{\Delta^2}{12} \text{ where } \Delta = \frac{R_{FS}}{2^B} \tag{15}$$

$$\sigma_e^2 = \frac{\left(\frac{R_{FS}}{2^B}\right)^2}{12} \tag{16}$$

$$= \frac{R_{FS}}{12 \times 2^{2B}} \tag{17}$$

Similarly we define a value for probabilistic noise power for an N-bit adder in Equation 18

$$\begin{gathered} \sum_{i=0}^{N-1} (2^{N-i-1}\Delta)^2 (1 - p_i) \\ \text{where } p_i\text{is probability of correctness} \end{gathered} \tag{18}$$

In this context the range full-scale (RFS) of 256 was used and quantization noise was compared to the base case of 16 bits. These values were chosen because of the prevalence in image processing applications and particularly in H.264. For comparision, optimal voltage biasing schemes were found for noise powers comparable to reduced-bit-width adders.

Of note in Figure 52, both reduced precision and the PCMOS fixed point technique yield over 3X energy savings. When comparing the two energy saving methods in Figure 53, the PCMOS approach has a lower noise power for a given energy for all points. PCMOS also has the advantage of a full 16-bit datapath allowing the noise power to be reduced back to 0 when the supply voltage is brougth back to full V_{dd} on the probabilistic bits.

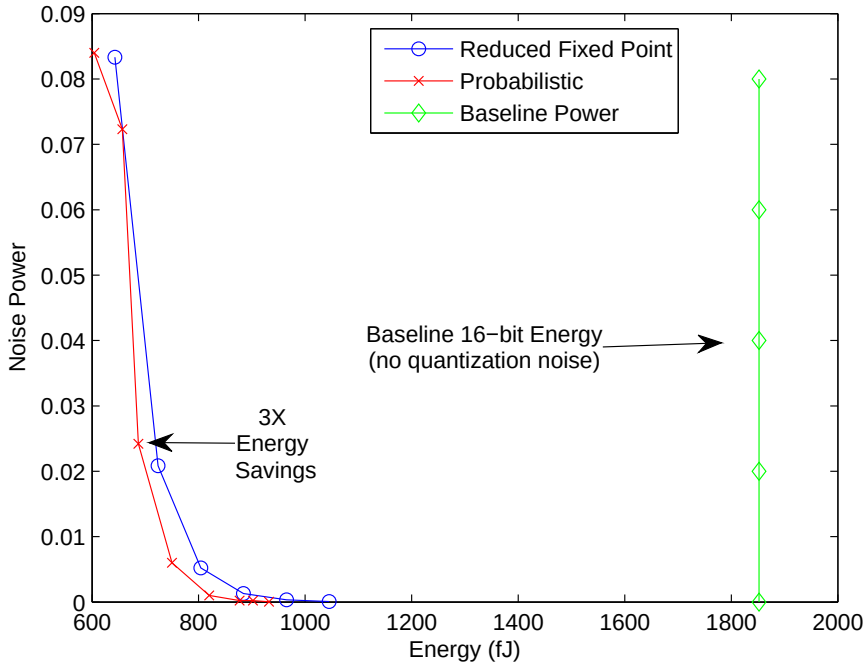


Figure 52: Noise Power vs Energy for a Fixed Point Adder. Chart comparing both the reduced bit width and 16-bit probabilistic fixed point adders against a full precision 16-bit adder. The energy savings is significant, over 3X in both cases.

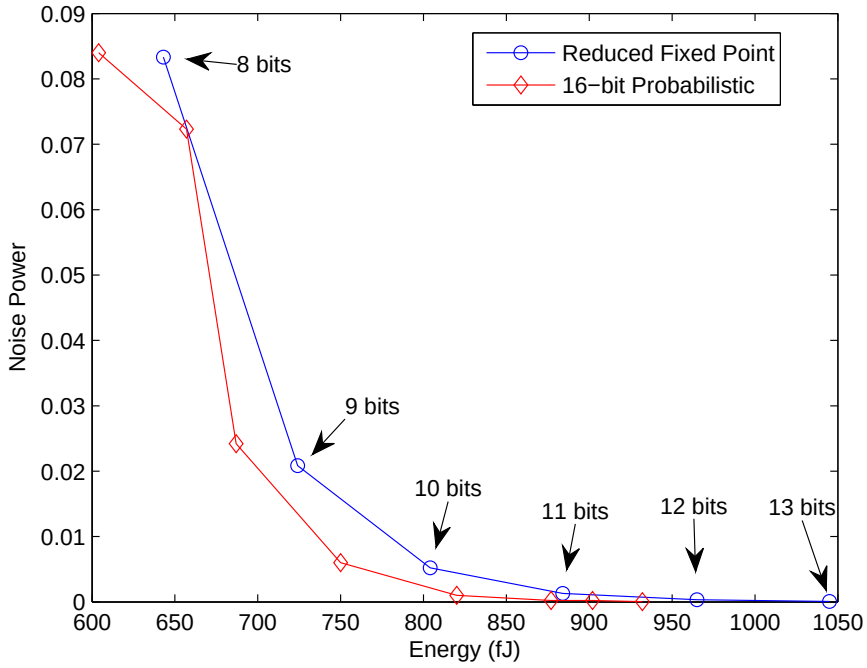


Figure 53: Zoomed in version of chart in (a). Notice that the probabilistic 16-bit adder has less noise power per given energy at all points.

10.4.1 Fundamental Result and Conclusion

These results are also more fundamental than this simple comarison. When using the BIVOS scheme, the voltage can only be dropped by an amount less than the threshold voltage V_t or about 100 mV so as to avoid leakage currents. And by the definitions of noise power in Equation 17 and Equation 18, any probability of error at bit 10 or above in the PCMOS fixed point unit will yield a greater noise power than an 8-bit, reduced fixed point unit. Thus with these two previous results: that we can only drop the voltage below full V_{dd} at bit 9 and below, and we can only reduce the voltage by about 100 mV at each subsequent bit to minimize energy consumption due to leakage currents, we have found an optimal biasing scheme for energy per noise. This energy per noise power is lower for deterministic adders of reduced bit width down to 8 bits. However, at below 8-bits the probabilistic adder can no longer yield better results when using a 16-bit baseline case, and thus we have a design cutoff point at 8-bits.

CHAPTER XI

LEARNING DIGITAL CIRCUITS

11.1 Introducing The Concept of Learning Hardware

The concept of learning digital hardware is presented here. A proof of concept of a circuit that can arbitrarily control the current, and hence the switching speed and power consumption, of a digital circuit is given. This control of current is directly tuned by the feedback from the digital circuit itself thus a learning digital system. An argument for a completely new paradigm in digital computing follows whereby an entire system of learning digital circuits is proposed.

What if processors could learn? With all of the myriad applications that our embedded systems, general purpose processors, and reconfigurable arrays of hardware are required to run, we could benefit greatly if our processors could learn exactly what it was we wanted them to do and how we wanted them to do it. Better software is not the solution for this adaptibility problem; after all, the ultimate performance of software is limited by the hardware itself. For low power processors, software only complicates the matter – the more software, the more instructions, and the more power is burned.

Proposed is a processor where the hardware itself *learns*. The hardware will learn which application it is running, adapt to create stronger circuits in the critical path of the application, will learn which paths are not critical, and tune down the power in those areas. The processor will remember what it has learned so that even when the hardware is powered down and the application reloaded at some later time, the processor will go back to the optimal state it learned for that application. Hardware designers or synthesis algorithms would no longer have to spend hours

tweaking designs when the designer does not even know for which application the circuit will ultimately be tweaked. Microcontrollers and complicated dynamic voltage scaling (DVS) algorithms would become unnecessary. Software or firmware will not be what tweaks the processor but the fabric of the circuits themselves.

Alas, many have proposed neuronal models of learning and implemented these in analog hardware [68], and some have even proposed this neuronal process in digital by implementing equations that model learning in FPGAs [14]. However, these methods all depend on spike-based neuron models using the spike time dependent plasticitiy (STDP) algorithm and cannot be of use to us for a general theory on a learning digital hardware.

11.2 A Key Circuit Element

In order for a processor or a digital circuit to *learn*, a novel circuit element must be introduced with a couple of key features. It must be able to

- Be dynamically programmable (during run-time).
- Control current flow arbitrarily in digital circuits.
- Remember or have a memory capacity.
- Be implemented with insignificant overhead to performance or power.

Since the flow of current is what ultimately determines the speed at which a digital circuit switches and its power consumption, a circuit element with the above characteristics would allow digital circuits to tune their own performance and power. Such a circuit element is given in Figure 54.

Represented in Figure 54 is a floating gate transistor used to control the speed and power of a digital circuit. The gate of the pFET is floating as it has no DC connection to a supply rail and is only capacitively coupled to other nodes, which means it can hold an arbitrary charge on the node. For a faster digital circuit, more

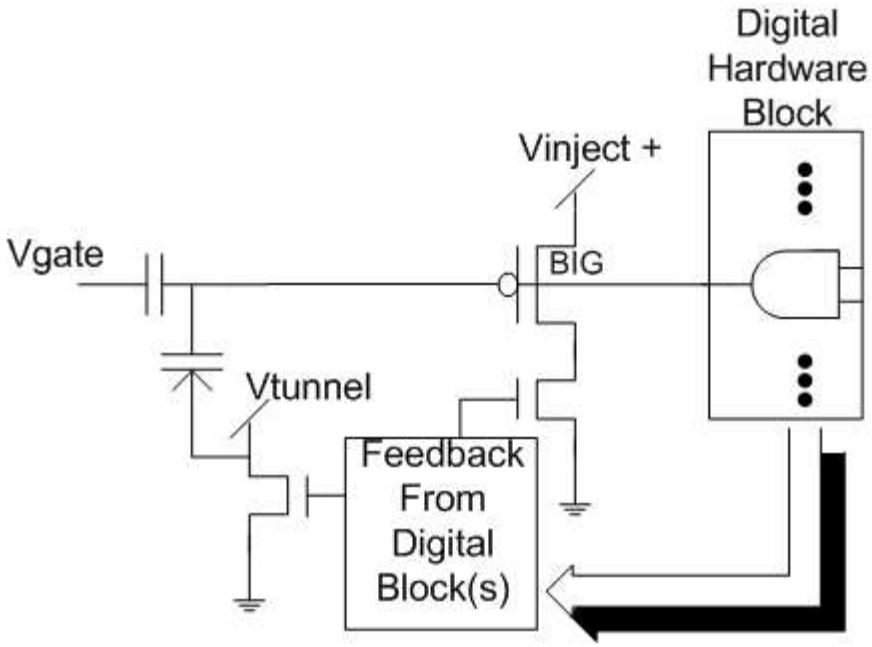


Figure 54: A floating gate transistor with digital feedback to control charge injection onto the pFET's floating node and to control electron tunneling off of the pFET's floating node. This node is also connected to a FET in the digital circuit allowing for a bias current of an arbitrary value (digital circuit of arbitrary speed).

charge is allowed onto the floating node, opening up the FET allowing more current to flow. Charge can be taken off for a more power efficient digital circuit. The voltage on the gate, V_{fg}, is reduced by charge injection (putting electrons onto the gate node) and is increased through *Fowler-Nordheim* electron tunneling [43], but otherwise does not change and *remembers* the current charge.

Recall that current through a transistor in the saturated region where $V_{DS} \geq (\kappa(V_G - V_{TH}) - V_S)$ neglecting velocity saturation is

$$I_{DS} = \frac{\mu C_{ox}}{2}\frac{W}{L}((V_{GS} - V_{TH})^2) \tag{19}$$

$$\text{Assuming } \kappa = 1 \tag{20}$$

$$I_{transistor} \propto V_{fg}^2 \tag{21}$$

Hence changing the charge on the floating node linearly would change the current, and thus the speed, in the digital circuit quadratically. In subthreshold digital, of which there has been much interest of late, the relationship is $I \propto e^{V_{fg}}$, which allows for an exponential response to the floating-gate control in ulta-low power circuits [43].

An experimental chip has been fabricated showing this concept, and a prototype is shown in Figure 55.

11.3 Datapath: A Case Study

Now that we have our key circuit element, a case study of how a processor's datapath would benefit is given. Take for example the image processing path of a digital signal processor (DSP). In many compression algorithms, video clips, and movie sequences the pixel data only changes for a very few pixels from frame to frame. Typical image data for an H.264 decoder yields repeated inputs to an FIR filter, which are made up strictly of adders and multipliers, due to the repeated pixel values being generated by the movie.

Using this H.264 movie decoder example, Figure 56 shows there are 4 carry-overs needed in the addition of a pixel value, 39, being incremented by 1, and this is the

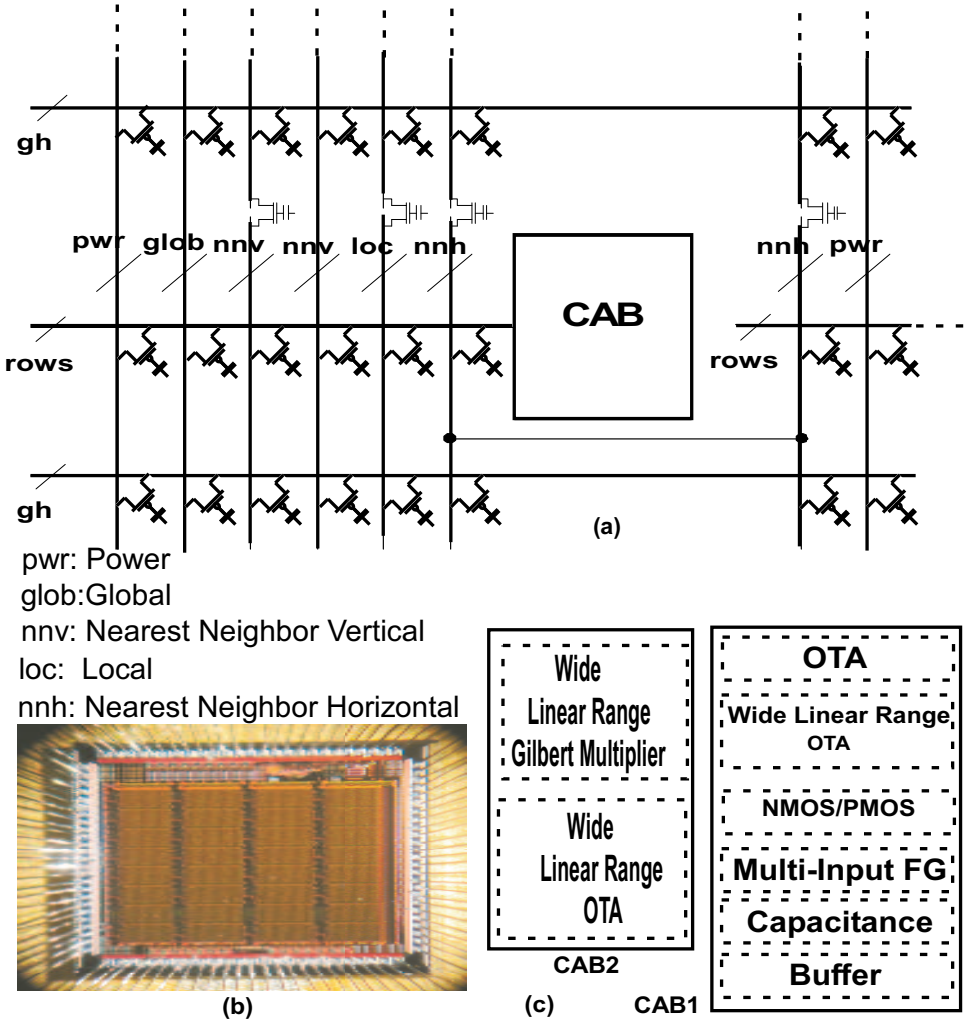
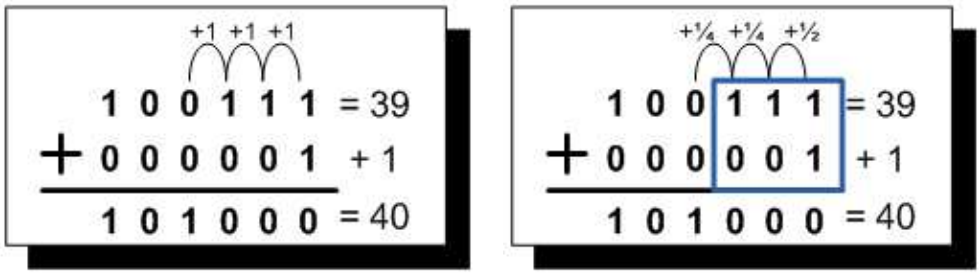


Figure 55: Die photo a Reconfirgurable Adaptive Floating-Gate Test Chip used to show proof of concept of this work. It has several dynamic floating gate structures which can be connected through the reconfigurable switching fabric to arbitrary circuits consisting of nFET and pFET transistors.



(a) (b)

Figure 56: Example showing number of carry operations in the addition of 39 and 1 where 4 carry overs are needed for the critical path. (a) Each 1-bit addition takes unit time, the critical path is equal to 4 time units here. (b) Our learning adder speeds up the first 3 1-bit additions to $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{4}$ time units respectively so the critical path only takes 2 time units total, increasing the speed of the adder *dynamically* by 2X.

critical path using a standard ripple-carry adder. If each 1-bit addition takes unit time, the critical path takes 4 time units to complete. Now, since pixel values are repeated 100s if not 1000s of times in a typical movie scene, it is likely this exact addition, or one close to it, would be repeated 1000s of times. Our datapath *learns* and *strengthens* the critical path (Figure 56b); the first 3 1-bit additions are sped up, and the critical path now takes only 2 time units total for a 2X speed increase. It has been shown that a 2X current increase, and thus this scenario, is quite plausible with floating gate technology [8].

11.4 Chip Results

The onchip dynamically programmable floating gate circuit is shown in Figure 57.

To summarize the caption of Figure 57, two versions of dynamic floating gate circuits were built and tested. One version uses a single input and the other uses a logical combination of two inputs to dynamically and in real time control the tunneling and injection of the floating gate circuit. A current starved inverter is built with floating gates to programmatically alter the period of the input signals to the floating gate system to normalize them to the time frames needed for tunneling and injection.

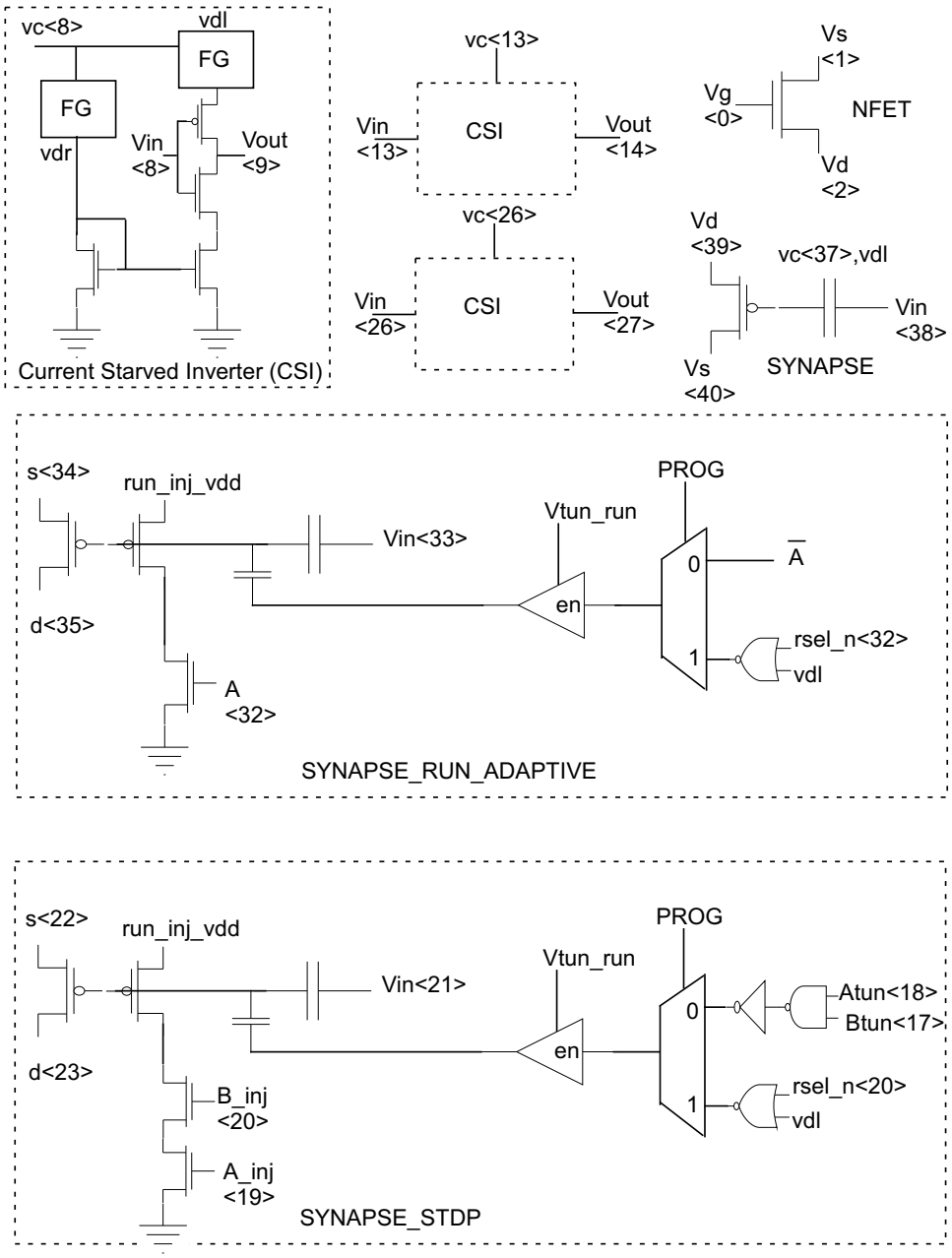


Figure 57: Test circuit schematic used to test the properties of the dynamic floating gate. The chip was built in conjunction with another chip design where the same circuit was used to build a circuit version of a synapse. The top diagram shows a current starved inverter where the current is restricted by floating gates (FG). This is used to alter the period of any incoming input signals to the dynamic floating gate so as to allow a proper time frame for tunneling and injection. The second diagram shows one version of the circuit where a single input "A" is used to control tunneling and injection. The third diagram shows another version of the dynamic floating gate where a logical combination of an "A" and "B" input are used to control tunneling and injection.

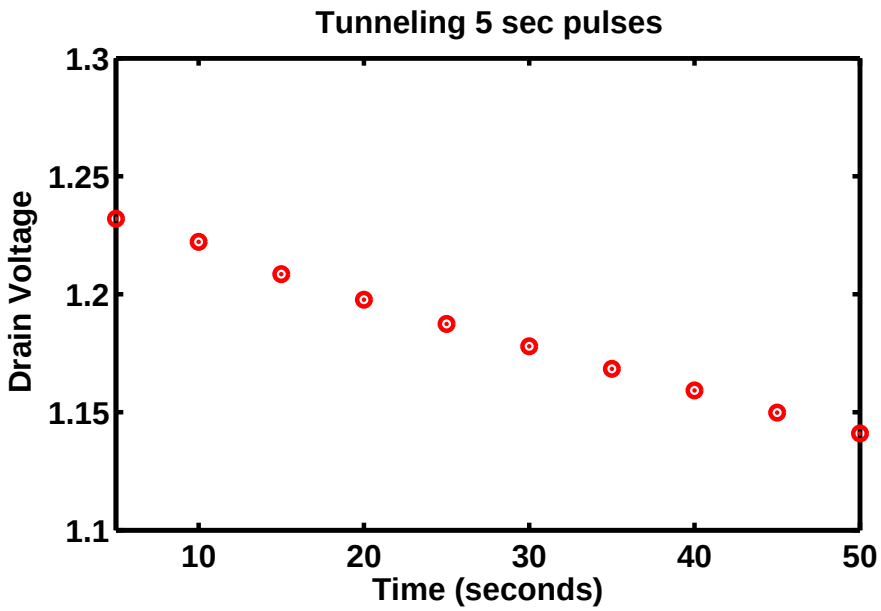


Figure 58: Drain Voltage vs Time of a floating gate dynamically programmed with 5 second long tunneling pulses, which are used to lower the floating gate voltage, V_{fg}. We see the range of voltages that can be obtained in this time frame are quite large, 100 mV.

11.4.1 Data Graphs

The goal of the test circuit shown in Figure 57 is to demonstrate the resolution, range, and speed of the dynamically adjustable floating gate circuits. Figure 58 shows the drain voltage changing on the floating gate with 5 second tunneling pulses applied.

Figure 58 shows a 100 mV range by using 5 second pulses over 50 seconds. In modern processes for which this design is targeted, this is fully 10% of the entire supply voltage range. Of course floating gates allow voltages outside the supply voltage range of operation as well in order to *overdrive* gates.

Figure 59 shows the fastest speed of 100 Hz at which the 0.35 μm floating gate could be dynamically tunneled. The average voltage change at this speed is approximately 200 μV.

Figure 60 shows the floating gate voltage being raised through injection at 200 Hz with 500 μV of change per pulse. We see that injection is much faster at $\Delta = 100\frac{mV}{sec}$ than tunneling which occurs at a rate of $\Delta = 2\frac{mV}{sec}$.

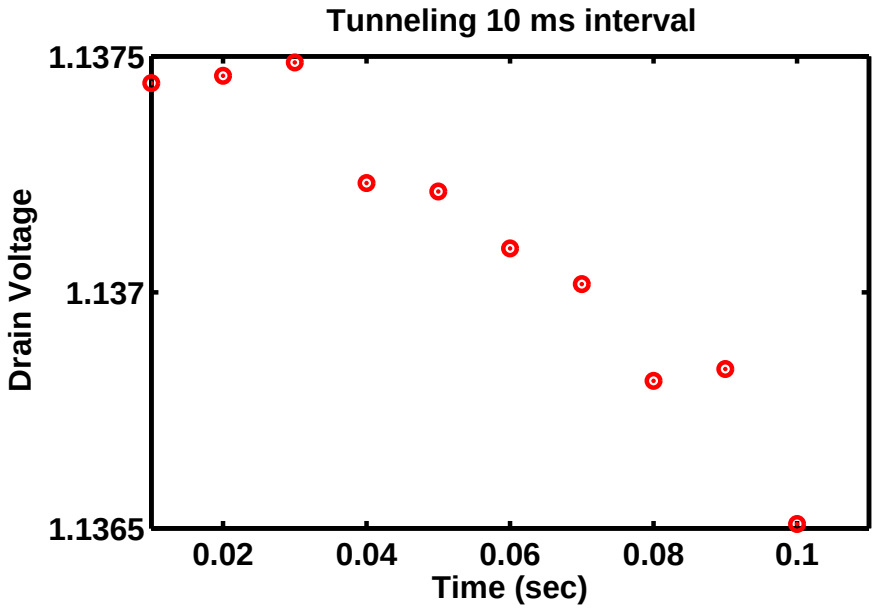


Figure 59: Drain Voltage vs Time of a floating gate with 10ms tunneling pulses. This is the smallest time frame (fastest speed at 100 Hz) at which the gate could be reliably dynamically programmed with the 0.35 μm process.

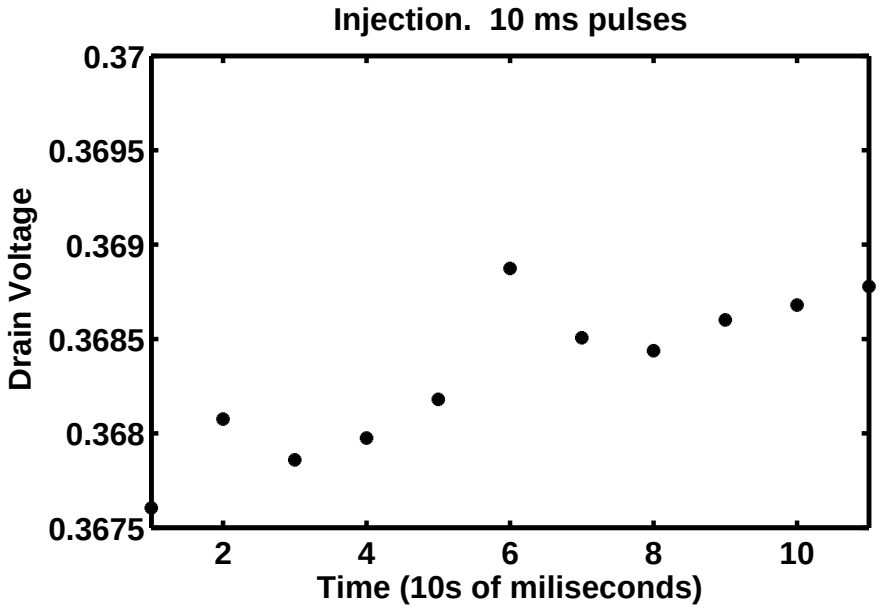


Figure 60: Drain Voltage vs Time of a floating gate with 10ms injection pulses used to raise the floating gate voltage, V_{fg}. This shows injection happening as fast as about 200 Hz with a voltage change of approximately 500 μV each pulse.

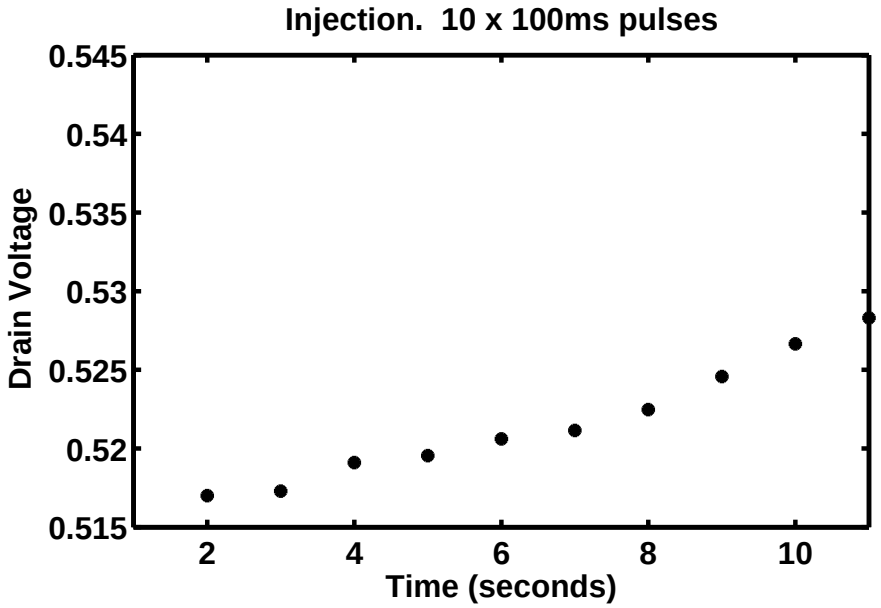


Figure 61: Drain Voltage vs Time of a floating gate with 10x100ms injection pulses used to raise the floating gate voltage, V_{fg}. This simply shows that injection is smoother over longer time periods.

Figure 61 shows that injection can be done quite smoothly with longer time frames allowing for an averaging effect of the electron movement onto the floating node.

11.5 System of the Future

In summary, the techniques presented here are a proof of concept used to create an entirely new paradigm of a learning digital computer. The next steps to be taken are to determine the best feedback mechanism (asynchronous completion cells are one method), the range of current, speedups, and power gains that can be realized, and to fabricate a datapath with this technology.

CHAPTER XII

CONCLUSION: THE CONTRIBUTION

The primary goal of any dissertation is to make a contribution to the field. What has been contributed with this thesis? The author believes the results from silicon speak for themselves. I was able to commit adds in the outdated 0.6 μm AMI technology at a very modern rate of 1 GHz. With a fully laid out datapath taking into account parasitic capacitances and interconnect delays in a more modern 0.18 μm technology (but still 4 technology generations behind today's technology), I was able to commit adds at $3-4$ GHz.

An improvement of this magnitude is due to circuit level and micro-architectural improvements inspired by superscalar and probabilistic computing techniques. A new paradigm called *Speculative Completion* is presented as well as the novel *2-Dimensional Bit-level Pipeline*. The use of PCMOS techniques with ultra voltage scaling such that thermal noise is the cause for probabilistic behavior is discounted in this work, which is valuable to the community so as valuable resources are not used on pursuing this technology. However, out of the fire of thermally based PCMOS comes the pheonix that is image processing with partial image degradation and showing that image processing with occasionally incorrect bits (due to out of order asynchronous bit completion in this case) is actually an *improvement* over the quantization noise that results from reduced bit width fixed point computation.

Overall, this work argues for continued pursuits in microarchitectural and circuit improvements to improve performance and energy efficiency, and that these improvements to speedup single threads are more important to overall application performance than more parallelism. Finally, due to the growing problem of circuit

variability and an inability to increase pipeline stages with synchronous technology, as well as the ability to pipeline and dynamically voltage scale at the bit level, asynchronous technology is presented as a solution for this continued performance and efficiency improvement.

Contributions for circuit and system level design in DSP and datapath systems have been presented arguing for the potential for asynchronous, probabilistic, and dynamic floating gate technologies.

REFERENCES

[1] J. a. Tierno, A. J. Martin, D. Borkovic, and T. K. Lee, "A 100-mips gaas asynchronous microprocessor," *IEEE Design and Test of Computers*, 1994.

[2] A. Agarwal, C. Kim, S. Mukhopadhyay, and K. Roy, "Leakage in nano-scale technologies: Mechanisms, impact, and design considerations," *Design Automation Conference (DAC)*, July 2004.

[3] M. Allam and M. Elmasry, "Low power implementation of fast addition algorithms," in *Proc. of IEEE Canadian Conference on Electrical and Computer Engineering, 1998*, vol. 2, May 1998, pp. 645 – 647.

[4] D. Anderson, P. Hasler, R. Butera, K. Palem, H. Marr, and J. George, "Probabilistic computing and biological applications," *National Science Foundation: Emerging Models and Technology (EMT) Grant Program*, 2007, anderson is Principle Investigator (PI); Hasler, Butera, and Palem are Co-PI's.

[5] "Arithmetic processor 166 instruction manual," Digial Equipment Corps, 1960, maynard, MA.

[6] "The asynchronous bibliography," [Online]. http://www.win.tue.nl/async-bib/, 2004.

[7] T. Austin, D. Blaauw, T. Mudge, and K. Flautner, "Making typical silicon matter with razor," *Computer*, pp. 57–65, Mar. 2004.

[8] A. Basu, C. M. Twigg, S. Brink, P. Hasler, C. Petre, S. Ramakrishnan, S. Koziol, and C. Schlottmann, "Rasp 2.8: A new generation of floating-gate based field programmable analog array," *Proceedings, Custom Integrated Circuits Conference CICC*, Sept. 2008.

[9] H. C. Bearley, "Illiac ii - a short description and annotated bibliography," *IEEE Transactions on Computing*, vol. C-14, no. 6, pp. 400–403, June 1965.

[10] A. C. S. Beck and L. Carro, "Transparent acceleration of data dependent instructions for general purpose processors," in *IFIP Internaiontal Conference on VLSI-SoC*, 2007.

[11] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, November/December 2005.

[12] K. Bowman, X. Tang, J. Eble, and J. Meindl, "Impact of extrinsic and intrinsic parameter varitiations on cmos system on a chip performance," *Twelfth Annual IEEE International ASIC/SOC Conference*, pp. 267–271, 1999.

[13] K. Bowman, J. Tschanz, C. Wilkerson, S.-L. Lu, T. Karnik, V. De, and S. Borkar, "Circuit techniques for dynamic variation tolerance," *Design Automation Conference (DAC)*, pp. 4–7, July 2009.

[14] A. Cassidy and A. Andreou, "Dynamical digital silicon neurons," in *IEEE Symposium on Biological Circuits and Systems (BioCAS)*, Nov. 2008.

[15] "Cell broadband engine architecture, version 1.0," http://cell.scei.co.jp, Aug. 2005.

[16] L. Chakrapani, B. E. S. Akgul, S. Cheemalavagu, P. Korkmaz, K. Palem, and B. Seshasayee, "Ultra-efficient (embedded) SOC architectures based on probabilistic cmos (PCMOS) technology," *Proc. of Design Automation and Test in Europe (DATE)*, Mar. 2006.

[17] L. Chakrapani, J. George, B. Marr, B. Akgul, and K. V. Palem, *VLSI-SOC: Research trends in VLSI and System on Chip*. Springer-Boston, 2008, ch. Probabilistic Design: A survey of probabilistic CMOS technology and future directions for Terascale IC Design, pp. 101–118.

[18] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, Apr. 1992.

[19] ——, "Low-power cmos digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 473–484, apr 1992.

[20] V. Chandramouli, E. Brunvand, and K. F. Smith, "Self-timed design in gaas: Case study of a high speed, parallel multiplier," *IEEE Transactions on VLSI Systems*, pp. 146–149, Mar. 1996.

[21] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," in *Proc. of IEEE Transactions on VLSI Systems*, vol. 5, no. 4, Dec. 1997, pp. 436 – 443.

[22] S. Cheemalavagu, P. Korkmaz, and K. V. Palem, "Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," in *Proc. of The 2004 International Conference on Solid State Devices and Materials*, Tokyo, Japan, Sept. 2004, pp. 402 – 403.

[23] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani, "A probabilistic cmos switch and its realization by exploiting noise," in *Proc. of In IFIP-VLSI SoC*, Perth, Western Australia, Oct. 2005.

[24] F.-C. Cheng, S. H. Unger, M. Theobald, and W.-C. Cho, "Delay-insensitive carry-lookahead adders," *Proceedings of th 10th International Conference on VLSI Design*, Jan. 1997.

[25] M. R. Choudhury and K. Mohanram, "Accurate and scalable reliability analysis of logic circuit," in *Design Automation and Test Europe (DATE)*, 2007.

[26] S. Das, S. Pant, D. Roberts, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "A self-tuning dvs processor using delay-error detection and correction," in *Symposium on VLSI Circuits Digest of Technical Papers*, 2005, pp. 258–261.

[27] B. P. Degnan, R. Wunderlich, and P. Hasler, "Programmable floating gate techniques for cmos inverters," *Proceedings for International Symposium on Circuits and Systems (ISCAS)*, pp. 2441–2444, 2005.

[28] M. Elgamel and M. Bayoumi, "Interconnect noise analysis and optimization in deep submicron technology," *IEEE Circuits and Systems Magazine*, vol. 3, no. 4, pp. 6 – 17, 2003.

[29] J. Emer, M. D. Hill, Y. Patt, J. Yi, D. Chiou, and R. Sendag, "Single-threaded vs multithreaded: Where should we focus?" *IEEE Micro*, pp. 14–24, Nov. 2007.

[30] R. Feynman, *Lectures on Computation.* Addison-Wesley, 1996.

[31] S. B. furber, J. D. Garside, P. Riocreux, S. Temple, P. Day, J. Liu, and N. C. Paver, "Amulet2e: An asynchronous embedded controller," *Proceedings of the IEEE*, pp. 243–256, Feb. 1999.

[32] J. D. Garside, S. B. Furber, and S.-H. Chung, "Amulet3 revealed," *Proceedings International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 51–59, Apr. 1999.

[33] P. Gemmell and M. Harchol, "Tight bounds on expected time to add correctly and mostly correctly," *Information Processing Letters*, vol. 49, pp. 77–83, 1994.

[34] J. George, B. Marr, B. E. S. Akgul, and K. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems CASES*, 2006.

[35] J. George, H. B. Marr, B. E. S. Akgul, and K. P. Palem, "Probabilistic design for ultra low-energy embedded computing," *ACM*, 2010.

[36] J. George, B. Marr, B. Akgul, and K. Palem, "Probabilistic design for low-energy computing," *To Appear in ACM Transactions on Embedded Systems*, 2008.

[37] S. Guthridge, R. E. Harper, H. Marr, and B. Abali, "Acoustic noise cancellation by phase alignment of cooling fans," in *Proceedings of the 23rd Symposium on the Thermal Measurement and Management of Semiconductors*, 2007.

[38] R. E. Harper, H. Marr, P. Manson, S. Guthridge, and B. Abali, "Mutual active cancellation of fan noise and vibration," Invention Disclosure, Endicott, NY, 2007.

[39] D. Harris and N. H. Weste, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education, 2005.

[40] P. Hasler, B. A. Minch, and C. Diorio, "Adaptive circuits using pfet floating gate devices," *Proceedings of 20th Anniversery Conference on Advanced Research in VLSI*, 1999.

[41] ——, "Floating gate devices: they are not just for digital memories anymore," *Proceedings of the International Symosium on Circuits and Systems (ISCAS)*, 1999.

[42] P. Hasler, "Floating gate devices, circuits, and systems," *Proceedings for fifth international workshop on System-on-Chip for Real Time Applications*, 2005.

[43] P. Hasler and J. Dugar, "Correlation learning rule in floating-gate pfet synapses," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Jan. 2001.

[44] R. Hedge and N. R. Shanbhag, "Energy-efficient signal processing via algorithmic noise-tolerance," in *Proc. of International Symposium on Low Power Electronics and Design*, Aug. 1999, pp. 30 – 35.

[45] ——, "Soft digital signal processing," *IEEE Transactions on VLSI*, vol. 9, no. 6, pp. 813 – 823, Dec. 2001.

[46] ——, "A voltage overscaled low-power digital filter IC," *Digital Object Identifier*, pp. 388–391, 2004.

[47] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network with 10240 floating gate synapses," *Proceedings of the International Joint Conference on Neural Networks*, 1989.

[48] "HSpice - The golden standard for Accurate Circuit Simulation. Synopsys Products and Solutions. http://www.synopsys.com/products/mixedsignal/hspice/."

[49] D. A. Huffman, "The synthesis of sequential switching circuits," *Journal of the Franklin Institute*, 1954.

[50] "http://www.itrs.net/common/2005itrs/home2005.htm," ITRS 2005 Edition.

[51] "http://www.itrs.net/links/2006update/2006updatefinal.htm," ITRS 2006 Edition.

[52] E. Jaynes, *Probability Theory: The Logic of Science*. Cambridge, UK: Cambridge University Press, 2003.

[53] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *The Bell System Technical Journal*, vol. 46, no. 4, 1967.

[54] J. Kessels and P. Marston, "Designing asynchronous standy circuits for a low-power pager," *Proceedings of the IEEE*, pp. 257–267, Feb. 1999.

[55] M. R. Khan, "Pentium 4 processor," http://www.scribd.com/doc/7343459/Pentium-4-Processor?autodown=pdf, 2008.

[56] K. Killpack, E. Mercer, and C. J. Myers, *Advanced Research in VLSI.* Los Alamitos, CA: IEEE Computer Society Press, Mar. 1996, ch. A Standard-Cell Self-Timed Multiplier for Energy and Area Critical Synchronous Systems, pp. 188–201.

[57] D. J. Kinniment, "An evaluation of asynchronous addition," *IEEE Transactions on VLSI Systems*, pp. 137–140, Mar. 1996.

[58] L. B. Kish, "End of Moore's law: Thermal (noise) death of integration in micro and nano electronics," *Physics Letters A*, vol. 305, pp. 144–149, Dec. 2002.

[59] ——, "Moore's law: Performance and power dissipation," *Dekker Encyclopedia of Nanoscience and Nanotechnology*, 2004.

[60] ——, "On the fundamental limits of digital computing," 2006, invited Talk at Georgia Tech.

[61] ——, "Thermal noise driven computing," *Applied Physics Letters*, vol. 89, Oct. 2006.

[62] P. Korkmaz, B. E. S. Akgul, K. V. Palem, and L. N. Chakrapani, "Advocating noise as an agent for ultra-low energy computing: Probabilistic CMOS devices and their characteristics," *Japanese Journal of Applied Physics, SSDM Special Issue Part 1*, pp. 3307–3316, Apr. 2006.

[63] P. Korkmaz and K. V. Palem, "The inverse error function and its asymptotic "order" of growth using O and ω," Georgia Institute of Technology, GA, Tech. Rep. CREST-TR-06-02-01, Feb. 2006.

[64] S. Krishnaswamy, G. Viamontes, I. Markov, and J. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Design Automation and Test Europe (DATE)*, 2005.

[65] J. Kwong, Y. Ramadass, N. Verma, M.Koesler, K. Huber, H. Moormann, and A. Chandrakasan, "A 65nm sub-vt microcontroller with integrated sram and switched-capacitor dc-dc converter," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2008, pp. 318–319.

[66] S. Lai, "Flash memories: where we are and where we are going," *IEEE International Device and Electron Meeting*, 1998.

[67] A. M. Lines, "Pipelined asynchronous circuits," Master's thesis, California Insitute of Technology, 1998.

[68] S.-C. Liu and R. Mockel, "Temporarily learning floating gate vlsi synapses," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2008.

[69] S. Lloyd, "Ultimate physical limits to computation," *Nature*, vol. 406, pp. 1047–1054, Aug. 2000.

[70] M. Lu, *Arithmetic and Logic in Computer Systems*. Hoboken, NJ: John Wiley & Sons, Inc., 2004.

[71] R. Manohar, "A case for asynchronous computer architecture," Ph.D. dissertation, California Institute of Technology, 1998.

[72] R. Manohar and J. Tierno, "Asynchronous parallel prefix computation," *IEEE Transactions on Computers*, vol. 47, no. 11, pp. 1244–1252, Nov. 1998.

[73] ——, "Asynchronous parallel prefix computation," *IEEE Transactions on Computers*, vol. 47, no. 11, pp. 1244–1252, Nov. 1998.

[74] A. Manzak and C. Chaktrabarti, "Variable voltage task scheduling algorithms for minimizing energy/power," in *Proc. of IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 2, Apr. 2003, pp. 270 – 276.

[75] B. Marr, J. George, D. Anderson, and P. Hasler, "Increased energy efficiency and reliability of probabilistic arithmetic," in *International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2008.

[76] B. Marr, A. Basu, S. Brink, and P. Hasler, "A learning digital computer," *Design Automation Conference (DAC)*, July 2009.

[77] B. Marr, B. Degnan, D. V. Anderson, and P. Hasler, "An asynchronously embedded datapath for performance acceleration and energy efficiency," *International Symposium on Circuits and Systems (ISCAS)*, May 2009.

[78] H. B. Marr, J. George, B. Akgul, and K. V. Palem, "Energy and performance efficiency through voltage overscaled probabilistic arithmetic," 2006, document Prepared for Conference Proceedings.

[79] A. J. Martin, "The design of an asynchronous microprocessor," in *Proceedings of the Decennial Caltech Conference on VLSI*, Mar. 1989.

[80] ——, "The limitations to delay-insensitivity in asynchronous circuits," in *Proceedings of the 6th MIT conference on Advanced Research in VLSI*, W. J. Dally, Ed., 1990, pp. 263–278.

[81] ——, "Asynchronous datapaths and the design of an asynchronous adder," *Formal Methods in System Design*, vol. 1, no. 1, pp. 119–139, 1992.

[82] A. J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth, U. Cummings, and T. K. Lee, "The design of an asynchronous mips r3000 microprocessor," in *Proceedings on the 17th Conference on Advanced Research in VLSI*, 1997.

[83] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, pp. 1089 – 1120, June 2006.

[84] A. J. Martin, M. Nystrom, and C. G. Wong, "Three generations of asynchronous microprocessors," *IEEE Design and Test of Computers: Special Issue on Clockless VLSI Design*, vol. 20, no. 6, pp. 9 – 17, Nov. 2003.

[85] F. Masuoka, R. Shirota, and K. Sakui, "Reviews and prospects of non-volatile semiconductor memories," *IEICE Transactions*, vol. E 74, no. 4, 1991.

[86] J. D. Meindl, "Low power microelectronics: Retrospect and prospect," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 619 – 635, Apr. 1995.

[87] T. H.-Y. Meng, R. Broderson, and D. G. Messerschmitt, "Automatic synthesis of asynchronous circuits from high-level specifications," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 11, pp. 1185–1205, Nov. 1989.

[88] C. Molnar, I. W. Jones, W. S. Coates, J. K. Lexau, S. M. Fairbanks, and I. E. Sutherland, "Two fifo ring performance experiments," *Proceddings of the IEEE*, pp. 297–307, Feb. 1999.

[89] S. W. Moore, G. S. Taylor, P. A. Cunningham, R. D. Mullins, and P. Robison, "Self calibrating clocks for globally asynchronous locally syncronous systems," in *International Conference on Computer Design*, 2000.

[90] T. Mudge, "Power: A first-class architectural design constraint," *IEEE Computer*, vol. 34, pp. 52–58, 2001.

[91] ——, "Keynote adress, low power robust computing," *High Performance Computing (HiPC)*, 2004.

[92] D. E. Muller and W. S. Barky, "A theory of asynchronous circuits," *Proceedings of the International Symposium on the Theory of Switching*, 1959.

[93] C. J. Myers, *Asynchronous Circuit Design.* John Wiley and Sons, 2001.

[94] K. Nepal, R. Bahar, J. Mundy, W. Patterson, and A. Zaslavsky, "Designing logic circuits for probabilistic computation in the presence of noise," in *Proc. of Design Automation Conference*, June 2005, pp. 486 – 490.

[95] ——, "Designing logic circuits for probabilistic computation in the presence of noise," in *Proc. of Design Automation Conference*, June 2005, pp. 486 – 490.

[96] S. M. Nowick, K. Y. Yun, P. A. Beere, and A. E. Dooply, "Speculative completion for the design of high-performance asynchronous dynamic adders," *Proceedings IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1997.

[97] S. M. Ornstein, M. J. Stucki, and W. A. Clark, "A functional description of macromodules," in *AFIPS Conference Proceedings: 1967 Spring Joint Computer Conference*, vol. 30. New York: Academic Press, 1967, pp. 337–355.

[98] K. V. Palem, "Proof as experiment: Probabilistic algorithms from a thermodynamic perspective," in *Proc. Intl. Symposium on Verification (Theory and Practice)*, Taormina, Sicily, June 2003.

[99] ——, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Trans. Computer*, vol. 54, no. 9, pp. 1123–1137, 2005.

[100] K. V. Palem, B. E. Akgul, J. George, and H. B. Marr, "Variable scaling for computing elements," Invention Disclosure, Atlanta, GA, Feb. 2006.

[101] K. V. Palem and B. E. S. Akgul, "The explicit use of probability in CMOS designs and the ITRS roadmap: From ultra-low energy computing to a probabilistic era of Moore's law for CMOS," SRC, Cavins Corner, Tech. Rep., Sept. 2005.

[102] K. V. Palem, L. N. Chakrapani, B. E. S. Akgul, and P. Korkmaz, "Realizing ultra low-energy application specific soc architectures through novel probabilistic CMOS (PCMOS) technology," in *Proc. of the International Conference on Solid State Devices and Materials*, Tokyo, Japan, Sept. 2005, pp. 678 – 679.

[103] S. Pant, D. Blaauw, V. Zolotov, S. Sundareswaran, and R. Panda, "A stochastic approach to power grid analysis," in *Proc. Design Automation Conference*, San Diego, California, June 2004, pp. 171 – 176.

[104] K. N. Patel, J. P. Hayes, and I. L. Markov, "Evaluating circuit reliability under probabilistic gate-level fault models," in *IWLS*, May 2003, pp. 59–64.

[105] Y. Patt, "Requirements, bottlenecks, and good fortune: Agents for a microprocessor revolution," *Proceedings of the IEEE*, pp. 1553–1559, Nov. 2001.

[106] D. Patterson and J. Hennessy, *Computer Organization and Design, 3rd Edition.* Morgan Kauffman, 2004.

[107] K. Rahimi and C. Diorio, "Design and application of adaptive delay sequential elements," *IEEE Transactions on Very Large Scale Integration (VLSI)*, vol. 14, no. 12, pp. 1354–1367, Dec. 2006.

[108] R. Ramachandran and S.-L. Lu, "Efficient arithmetic using self-timing," *IEEE Transactions on VLSI Systems*, pp. 445–454, Dec. 1996.

[109] S. rotem, K. Stevens, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, and B. Agapiev, "Rappid: An asynchronous instruction length decoder," *Proceedings International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 60–70, Apr. 1999.

[110] T. Sakurai, "Perspectives on power-aware electronics," *Proceedings of the Interational Solid-State Circuits Conference (ISSCC)*, 2003.

[111] K. L. Shepard, "Conquering noise in deep-submicron digital ics," *IEEE Design and Test of Computers*, vol. 15, no. 1, pp. 51 – 62, Jan. - Mar. 1998.

[112] B. Shim, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," in *Proc. of IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, May 2004, pp. 497 – 510.

[113] M. L. Shooman, *Reliability of Computer Systems and Networks.* John Wiley and Sons, Inc, 2002.

[114] M. Singh and S. M. Nowick, "Fine-grain pipelined asynchronous adders for high-speed dsp applications," *Proceedings IEEE Computer Society Workshop on VLSI*, pp. 111–118, Nov. 2000.

[115] ——, "The design of high-performance dynamic asynchronous pipelines: High-capacity style," *IEEE Transactions on Very Large Scale Integration (VLSI)*, pp. 1270–1283, Nov. 2007.

[116] R. F. Sproull, I. e. Sutherland, and C. E. Molnar, "The counterflow pipeline processor architecture," *IEEE Design and Test of Computers*, pp. 48–59, 1994.

[117] K.-U. Stein, "Noise-induced error rate as a limiting factor for energy per operation in digital ICs," *IEEE J. Solid-State Circuits*, vol. 12, pp. 527–530, Oct. 1977.

[118] R. M. Tomasulo, "An efficient algorithm for exploiting multiple arithmetic units," *IBM Journal of Research and Development*, vol. 11, no. 1, pp. 25–33, 1967.

[119] Tweeny, www.eetimes.com, 2007.

[120] C. M. Twigg, J. D. Gray, and P. E. Hasler, "Programmable floating gate fpaa switches are not dead weight," *Proceedings of the International Symposium on Circuits and Systems*, pp. 169–172, May 2007.

[121] K. v. Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij, "A fully-asynchronous low-power error corrector the dcc player," *IEEE Journal of Solid State Circuits*, pp. 1429–1439, Dec. 1994.

[122] H. v. Gageldonk, K. v. Berkel, D. Gloor, A. Peeters, and G. Stegmann, "An asynchronous low-power 80c51 microcontroller," *Proceedings International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998.

[123] J. von Neumann, *Automata Studies*. Princeton University Press, 1956.

[124] L. Wang and N. R. Shanbhag, "Low-power filtering via adaptive error-cancellation," *IEEE Transactions on Signal Processing*, vol. 51, pp. 575 – 583, Feb. 2003.

[125] T. E. Williams and M. A. Horowitz, "A zero-overhead self-timed 160-ns 54-b cmos divider," *IEEE Journal of Solid State Circuits*, vol. 26, no. 11, pp. 1651–1661, Nov. 1991.

[126] C. G. Wong and A. J. Martin, "High level synthesis of asynchronous systems by data-driven decomposition," in *Proceedings of the Design Automation Conference (DAC)*, 2003, pp. 508–513.

[127] J. V. Woods, P. Day, S. B. Furber, J. D. Garside, N. C. Paver, and S. Temple, "Amulet1: An asynchronous arm processor," *IEEE Transactions on Computers*, pp. 385–398, Apr. 1997.

[128] K. C. Yeager, "The mips r10000 superscalar microprocessor," *IEEE Micro*, pp. 28–41, Apr. 1996.

[129] Y. Yeh and S. Kuo, "An optimization-based low-power voltage scaling technique using multiple supply voltages," in *Proc. of IEEE Internaitonal Symposium on ISCAS 2001*, vol. 5, May 2001, pp. 535 – 538.

[130] Y. Yeh, S. Kuo, and J. Jou, "Converter-free multiple-voltage scaling techniques for low-power cmos digital design," in *Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, Jan. 2001, pp. 172 – 176.

[131] S. Zhang, V. Wason, and K. Banerjee, "A probabilistic framework to estimate full-chip subthreshhold leakage power distribution considering within-die and die-to-die p-t-v variations," *Proceedings of the International Symposium on Low Power Electronics and Design*, 2004.

[132] R. Zimmerman and W. Fichtner, "Low power logic styles: Cmos versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1079–1090, jul 1997.