ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2858241

Analysis and Characterization of a Locally-Clocked Module

Article · August 2002

Source: CiteSeer

CITATION
1

READS
33

3 authors, including:

Kip Killpack
Intel
15 PUBLICATIONS **305** CITATIONS
SEE PROFILE

Erik Brunvand
University of Utah
117 PUBLICATIONS **1,342** CITATIONS
SEE PROFILE

All content following this page was uploaded by Erik Brunvand on 07 October 2014.

The user has requested enhancement of the downloaded file.

ANALYSIS AND CHARACTERIZATION OF A LOCALLY-CLOCKED MODULE

by

Kip C. Killpack

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Department of Electrical and Computer Engineering

The University of Utah

May 2002

Copyright © Kip C. Killpack 2002

All Rights Reserved

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Kip C. Killpack

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Chris J. Myers
	Erik Brunvand
	Reid R. Harrison

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Kip C. Killpack in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Chris J. Myers
Chair, Supervisory Committee

Approved for the Major Department

V. John Mathews
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

This thesis describes an evaluation of a locally-clocked module. Locally-clocked modules can be used as synchronous datapath elements in synchronous systems or as asynchronous elements in an asynchronous system. One key element of a locally-clocked module is a stoppable ring oscillator (or stoppable clock). If locally-clocked modules are to be used, their practicality must be quantified. Namely, it must be shown that a reliable and useful stoppable clock can be built. This thesis presents the design and evaluation of a fabricated locally-clocked sequential multiplier. The multiplier is used as a driving example to evaluate local clocks. The design for the stoppable clock is a hybrid of stoppable clocks from previous work. The same gates that make up the critical path of the multiplier are used to make the delay element of the stoppable clock. Although the stoppable clock is meant to track the datapath under a wide range of voltages and temperatures, it is shown that the clock requires tuning to match the critical path sometimes. This is due to the fact that it is difficult to match the critical path exactly. In addition, some temperature and voltage data points cause the cutoff path for the clock to be too slow. This problem is fixed by slowing down the clock. Future designs can focus on speeding up the cutoff path; thus, matching the critical path delay is the only limiting factor on clock frequency. A 20-bit multiplier was fabricated through MOSIS using AMI's 0.5μm process. The multiplier consumes 0.468 mm^2 and contains 8190 transistors. With a 5 volt power supply, the multiplier runs at 13.3 MHz and consumes 196.6 mW of power, while the stoppable clock runs at 174 MHz. This thesis presents latency and power measurements for the multiplier and stoppable clock in addition to a detailed analysis of stoppable clocks. Process variation is analyzed in that five chips are tested and shown to have little variation in measured values.

To my family for their love and support

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

Numerous people helped in the writing of this thesis and in developing the research that went into it. I would like to thank those who played a significant role in helping me complete this work. First, I am exceedingly grateful to my adviser Chris Myers. He always made time to discuss the research I was doing. I'm thankful for the productive meetings we had together. He guided me from my junior year through graduation and has been a wonderful mentor and friend.

I am indebted to my other committee members, Erik Brunvand and Reid Harrison. Meetings with them helped define the course of my research and gave me many ideas for how to design and test my work. In addition, I am thankful to the professors and teachers I have had throughout my education who taught me with such skill.

I would also like to thank the previous and current people who work in my office (Eric Peskin, Eric Mercer, Scott Little, Curt Nelson, Jie Dai, Hans Jacobson, Jung-lin Yang, Yanyi Zhao, Nick Seegmiller, Robert Thacker, Wendy Belloumini, Sung-Tae Jung, and Hao Zheng). I have become good friends with these people and appreciate the fun times we had. There were many discussions with people in the lab concerning my work. I appreciate the ideas and discussion that we had together. In particular, Eric Mercer was a mentor during my first years in our research lab. He became a great friend and biking partner.

Interaction with people in industry was also beneficial to this work. Keith Davis and Gerald Wilson at SONIC Innovations are to be thanked for patiently explaining their hearing aid design and giving motivation for researching low-power multipliers. I'm thankful to Ken Stevens at Intel and the rest of the Strategic CAD Lab people. Discussions with them helped refine the tests, results, and conclusions of my work. And I could quit looking for a job and focus on finishing my school

work because the SCL hired me — thanks be to Intel and SCL.

My family was especially helpful in completing my education at the University of Utah. They were always supportive and helped me through the tough times with their love and patience. I'm also grateful to Snowbird for providing a reduced-price student season pass, which only served to distract from my studies.

I am grateful to MOSIS for providing free fabrication of my chips. And also to Gabe Tau'a and the rest of the ECE lab supervisors for making the long testing hours bearable.

This work was made possible by grants from the Wayne Brown Fellowship, National Science Foundation, and Intel Corporation.

CHAPTER 1

INTRODUCTION

Portable digital assistants (PDAs) such as palm-pilots and cell phones are becoming more prevalent in society today. In conjunction with PDAs, embedded controllers and application specific integrated circuits (ASICs) are used in many types of electronic devices. The core component of these types of products is a small area and low power integrated circuit (IC). Small area budgets are required mainly because these ICs are placed in extremely small casings and packages. In addition to application size limitations, small ICs have a larger yield during manufacturing. High yields on IC wafers allow companies to market their products at lower cost and/or with higher profit margins. Much time is taken to increase yield because it has a direct effect on profit.

In addition to effecting yield, area has an effect on power consumption. Large area implies a large amount of capacitance due to routing wires. Power on a digital IC is directly proportional to capacitance. Applications that use embedded controllers and ASICs are required to run a long time on a limited amount of battery power. Therefore, designers must take time to create low power solutions in their circuits. The less power a design uses, the longer the design can run on a battery. Even large-scale microprocessors are running into the problem that their ICs consume too much power. Though in this case, the problem lies in dealing with the heat generated due to power consumption. To help circuit designers meet their power and area budgets, it is necessary to develop methodologies and circuit design practices that facilitate low power and small area design.

One approach to achieving low power and small area circuits is to use *Globally Synchronous, Locally Synchronous* (GSLS) systems. GSLS systems use a global

clock to initiate computation in locally-clocked modules. The global clock is generated by a crystal oscillator off-chip. The local clocks are generated by on-chip stoppable ring oscillators (or stoppable clocks). Most synchronous designs use one high frequency clock and subdivide it to lower frequencies as needed. Yet, driving a high frequency clock across a chip can lead to a large power drain. Also the clock must be set to the fastest frequency needed on chip. In GSLS systems, the global clock may be set to a lower frequency while local clocks handle the high frequency computations.

It is hypothesized that a chip composed of locally-clocked modules can lead to a power savings. The power savings can come from three areas. First, the global clock can be set to a lower frequency because the local clocks handle the high frequency computation. Second, when a locally-clocked module has finished its computation, its local clock is stopped. Energy is not wasted when the module is not doing "real work." Third, locally-clocked modules can be made as iterative modules, thus decreasing their area. As stated before, area has a direct impact on power consumption. It is true that iterative modules can be made without a local-clock; however, they require a high frequency clock. If that high frequency clock is not generated locally (as in GSLS systems), it must be generated on the global level, which causes a significant power drain.

In a GSLS system, iterative modules can be used without requiring the power drain of a high frequency global clock. If these iterative modules can be made smaller than noniterative modules, they also have benefits for routing and scheduling of resources. A small module resource may be duplicated numerous times on the IC next to other modules that use it. This helps mitigate wiring delay which is costly in deep submicron processes. Duplicating small module resources also helps limit power consumption because there is less routing compared to sharing a single resource. Scheduling of resources is also simplified if resources are duplicated rather than shared.

Along with all the aforementioned benefits, some issues with GSLS systems must be addressed. One part of GSLS systems that raises questions is the stoppable clock.

Generating a clock with gates on chip makes it questionable as to its reliability and robustness. For example, stoppable clocks may or may not have good noise rejection and clean edges. Stoppable clocks also have a lack of jitter-control. Future processes may also affect stoppable clocks in adverse ways. Data must be collected on how fast local clocks can run, how much tuning they need, how small locally-clocked modules are, and how much power locally-clocked modules consume. This thesis presents work done in evaluating some of the issues raised by stoppable clocks. A detailed analysis of the electrical issues pertaining to stoppable clocks is presented.

Another aspect in the design of local clocks is that they are made with delay elements meant to match the critical path delay. If the clock delay element does not track the critical path under all conditions, tuning must be added to the clock. How the clock is designed, determines how much tuning is required. This thesis discusses how to design the clock to minimize the amount of necessary tuning. An analysis is performed to see how closely the clock matches the critical path and how much extra tuning is required. Matching the critical path delay under varying conditions is useful for applications required to run at two different voltages under various modes of operation. Only one clock is necessary, rather than two. Without the local clock, two different frequencies would have to be available under the two different operating conditions.

1.1 Our Work

This thesis presents the design and evaluation of a stoppable clock. The stoppable clock is a hybrid of stoppable clocks from previous work. In order to evaluate stoppable clocks fully, a module for the clocks to synchronize is needed. This work presents a locally-clocked sequential multiplier used as a driving example to analyze stoppable clocks. The sequential multiplier presented can be used as a module in a GSLS system. As preliminary work, the sequential multiplier is compared to a combinational or parallel array multiplier. It is shown that the sequential multiplier consumes less energy and takes up less area than the parallel array multiplier. Thus, a sequential multiplier could benefit designs which have strict area and energy

budgets. It is a further benefit that the sequential design is made with a stoppable clock, thus allowing the global clock to remain at a low frequency. To continue the evaluation, a 20-bit multiplier and stoppable clock were fabricated through MOSIS using AMI's 0.5μm process. The multiplier and stoppable clock consume 0.468 mm^2 and contain 8190 transistors.

The stoppable clock consists of two main units — a delay element, and a control circuit. The control circuit is generated by `ATACS`, a tool for the synthesis and verification of timed circuits [1]. Stoppable clocks require the use of timing assumptions in order to function correctly. By using `ATACS` to generate the control, the circuit works correctly as long as the timing assumptions are adhered to. The delay element is designed using gates that match the gates on the critical path. Thus, the delay element is meant to scale the same as the critical path under various VDD, temperature, and process corners. SPICE simulations are presented to show correct functionality of the stoppable clock.

Numerous experiments are performed to help quantify the robustness of the stoppable clock and stoppable clock module. Simulated and measured latency and power numbers for the multiplier and stoppable clock are presented. With a 5 volt power supply, the multiplier runs at 13.3 MHz and consumes 196.6 mW of power, while the stoppable clock runs at 174 MHz. With a 3.0 volt power supply, the multiplier runs at 9.42 MHz and consumes 46.8 mW of power, while the stoppable clock runs at 117 MHz. In addition to these data points, the clock is tested to see if the delay element tracks the critical path gates under a wide range of voltages and temperatures.

Several clocks are built on the chip to help analyze stoppable clocks. One identical clock to the one that synchronizes the multiplier is placed on the opposite side of the chip. This is used to evaluate process variation effects on the stoppable clock. A third clock uses inverters rather than critical path gates. It is used to show how inverters scale differently than critical path gates under different conditions. Process variation is also evaluated in that five chips are tested and shown to have little variation in measured values. Through this design a greater understanding of

locally-clocked modules is achieved.

1.2 Contributions

One contribution of this work is the evaluation of a particular locally clocked example. In the stoppable clock design, the frequency of the clock is limited by two factors — the cutoff path of the clock, and the ability to match the critical path closely. Many factors make it difficult to match the critical path. For example, the critical path has coupling capacitance and is affected by noise on parallel lines. The current clock design does not make an attempt to match this. Because it is not matched perfectly, some conservative delay must be added to the clock delay. This ensures correct functionality even in the face of coupling capacitance; however, if it is possible to match the critical path exactly, the conservative delay can be removed. This speeds up the local clock, and hence the multiplier itself is faster.

Rather than making the clock completely devoid of conservative delay, the clock has tuning built in which can be set to add or remove conservative delay. Although the evaluated stoppable clock design is meant to track the datapath under a wide range of voltages and temperatures, it is shown that the clock requires tuning to match the critical path sometimes. This is because it is difficult to match the critical path exactly. It is also shown that certain voltage and temperature data points cause the cutoff path to be too slow. In these situations, clock tuning is needed to slow the clock down such that the cutoff path becomes fast enough relative to the local clock feedback. Future designs can focus on speeding up the cutoff path; thus, matching the critical path delay is the only limiting factor on clock frequency.

In ASIC and embedded controller design, there is a strict budget for area and power consumption. In these applications, iterative modules could be extremely helpful due to their small area; however, iterative modules require high frequency clocks. The work done for this thesis shows that the high frequency clocks (230 MHz) can be generated locally through the use of stoppable clocks. In this situation, the global clock can remain at a low frequency; therefore, the

design can use iterative modules without incurring a large power drain. In addition, smaller modules are inherently lower power because they have less capacitance, and power consumption on a digital IC is directly proportional to capacitance. Finally, locally-clocked modules can turn the internal local clock off to save power. Thus, stoppable clocks enable the use of GSLS systems which meet low power and small area budgets.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 begins with an analysis of previous multiplier research. It presents the architecture of our sequential multiplier and an overview of the multiplier interface. Preliminary analysis and comparison of the sequential multiplier to a parallel array multiplier comes next. Finally the test circuitry and test board is described.

The local clock design is discussed in Chapter 3. It starts with a literature survey of previous stoppable clock work. Then the architecture for our clock is described. The design of the `ATACS` synthesized control follows. The delay element design is discussed in this chapter. SPICE simulation results for the functionality of the stoppable clock are also presented.

Chapter 4 discusses experimental results from a fabricated IC of the locally-clocked multiplier. All proposed experiments are detailed first. SPICE simulation results of the multiplier come next followed by actual measured results of the multiplier. Then a detailed analysis of stoppable clocks is presented. This chapter finishes with a brief discussion of the results.

Lastly, Chapter 5 gives concluding remarks. It gives a summary of the experimental results, details the major contributions of this work, and discusses future work in the area of stoppable clocks.

CHAPTER 2

MULTIPLIER AND TEST CIRCUITRY

In order to evaluate a stoppable clock fully, a module for the clock to synchronize is needed. This chapter presents the design and analysis of a locally-clocked sequential multiplier. This multiplier is used as a driving example to analyze stoppable clocks. A discussion of previous self-timed datapath elements is included. A locally-clocked multiplier is considered a self-timed module in that it generates its own timing reference. The test circuitry for this multiplier is also discussed at the end of this chapter.

2.1 Self-timed Datapath Elements

There has been a lot of work in the area of self-timed multipliers and asynchronous datapath elements. Self-timed multipliers can be broadly grouped into two categories: parallel array and serial-parallel (i.e., iterative or sequential). A parallel array multiplier uses on the order of N^2 full adders in an $N \times N$ configuration, where N is the size of the operands. Array multipliers are often pipelined for increased throughput [2, 3, 4, 5, 6, 7, 8]. The area impact of parallel array designs can be reduced using Booth recoding [9] of radix-4 or greater; however, Booth recoding does not affect the $O(N^2)$ area bound seen in full parallel array multipliers. Self-timed and synchronous parallel array designs are compared in [10, 11] to show the power savings found in self-timed design methodologies. Moreover, [11] shows power consumption to be polynomial in both design styles. Although self-timed parallel array designs are fast, they require a considerable amount of area. Thus, they are not appropriate for strict area budget applications.

Iterative or serial-parallel multipliers use on the order of N full adders N times to complete a multiplication. Iterative designs have significantly smaller area at the

expense of increased latency. In [12], a *delay insensitive* design style is utilized to remove internal glitches on all wires, thus saving energy. Nevertheless, it requires two wires to represent each bit in the multiplication, causing an increase in area. Several self-timed designs are compared in [13], but all have a considerable area penalty to achieve self-timing.

Work in [14] employs a *bundled-data* design style to implement an iterative multiplier. This style removes the self-timed area overhead. Bundled-data designs replace the traditional fixed frequency clock in synchronous design with individual delay elements matched to the latency of each stage in the design. Each stage signals the controller when it is done by using a delay element as a timing reference. Furthermore, [14] matches critical path delays by building delay elements out of the same gates that make up the critical path. They also match the same gate loads that exist in the critical path. This allows the delay elements to scale evenly with the critical path delay under voltage, temperature, and process variations. Work in [14] further optimizes the iterative design by skipping iterations in the algorithm according to the operands (i.e., a 0-bit in the multiplicand). Although [14] achieves better area than [12], the control for the multiplier falls directly on the critical path. This is because a control communication is required at each stage of the multiplier.

An alternative approach is to use a self-timed module such as the 54-bit divider in [15]. This divider is used in the SPARC64 described in [16]. The divider uses an iterative algorithm with dual-rail domino-logic to perform division. It is not necessary to acknowledge back to the synchronous environment when the divide has completed. Instead, the divider is timed such that it completes within a given number of global clock cycles. A different solution along the same vein is to use a locally-clocked module. An example of such is the 64-bit iterative multiplier presented in [17]. The multiplier in [17] uses a stoppable clock to synchronize a partial parallel array. With either iterative module, there is no need for a high frequency global clock to synchronize iterations. This thesis is mainly concerned with an analysis of stoppable clocks, and therefore a closer look at [17] is warranted.

In a full parallel array, area is bounded by $O(N^2)$ because for each bit of the

multiplicand there is a row of N full adders. The multiplier in [17] implements several but not all rows of the parallel array and then iterates on those rows to complete the multiply. This creates a small multiplier that still has acceptable latency for a 64-bit word size. Looking at the extreme case in [17], the design would use a single row of full-adders $\frac{N}{2}$ times. It is $\frac{N}{2}$ rather than N because of the radix-4 Booth recoding that retires two bits at a time. The frequency of the stoppable clock in [17] is matched to the worst-case critical path delay in the design. When operands are ready to be multiplied, the clock is started, runs a fixed number of iterations, and is stopped. The stoppable clock removes control overhead from the critical path because communications are not required between each stage of the design. When a clock pulse arrives at the latches, it is assumed that all stages are ready for the next input.

The multiplier presented in this chapter uses a similar bundled data design style to that of [14], however it removes control from the critical path through the use of a stoppable clock as in [17]. Rather than generating a partial parallel array, this work focuses on area by evaluating the extreme case of the design in [17]. This extreme case uses a single row of full-adders on each iteration. In addition, the stoppable clock in [17] is improved based on related work done on stoppable clocks and delay elements. The multiplier presented in this chapter is also discussed in [18].

2.2 Architecture

The micro architecture for the multiplier is found in Fig. 2.1. This multiplier uses Booth recoding on the operands so that it can handle *two's complement* numbers without the need for pre or post calculations [9]. Radix-4 Booth recoding is used because it cuts the number of iterations in half compared to radix-2 Booth recoding. The multiples of B needed in radix-4 recoding are still easily computed during iterations.

The *Selector B* block provides multiples of B to the *Carry Save Adder Shift Register* block according to input from *Decode A*. The architecture of the *Selector B*

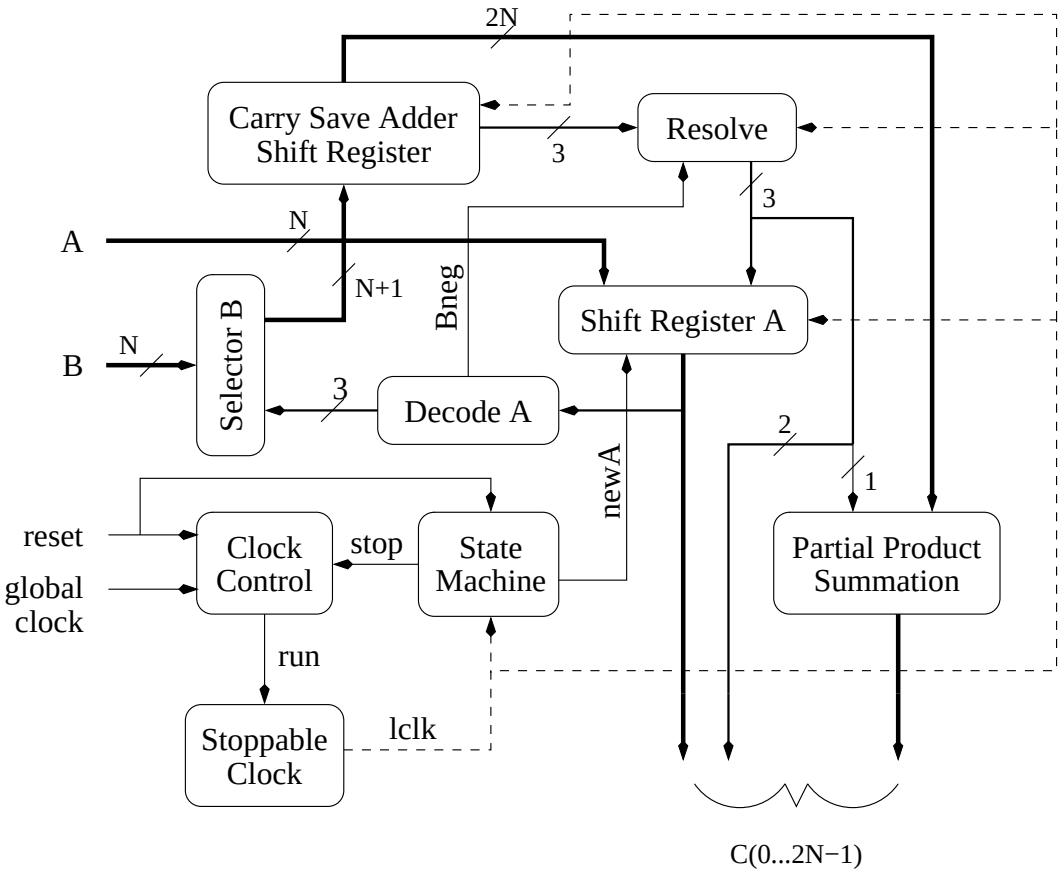


Fig. 2.1. Architecture for the multiplier.

module is shown in Fig. 2.2. In this figure, the B operand comes in on the left, is recoded according to *Decode A* signals, and goes out on the right to the *Carry Save Adder Shift Register*. For radix-4 Booth recoding, the *Selector B* circuit must produce the following multiples of the B operand: 0, B, $-B$, $2B$, and $-2B$. The multiplier uses a *two's complement* representation, so generating the multiples of B is a matter of inverting and shifting bits as appropriate. In order to complete the *two's complement* conversion for $-B$ and $-2B$, a 1 must be added into the low-order bit of the B operand. The block that handles this carry insertion is the *Resolve* circuit.

Radix-4 Booth recoding retires two bits on each iteration; therefore, there are situations when unresolved carries are shifted out of the *Carry Save Adder Shift*

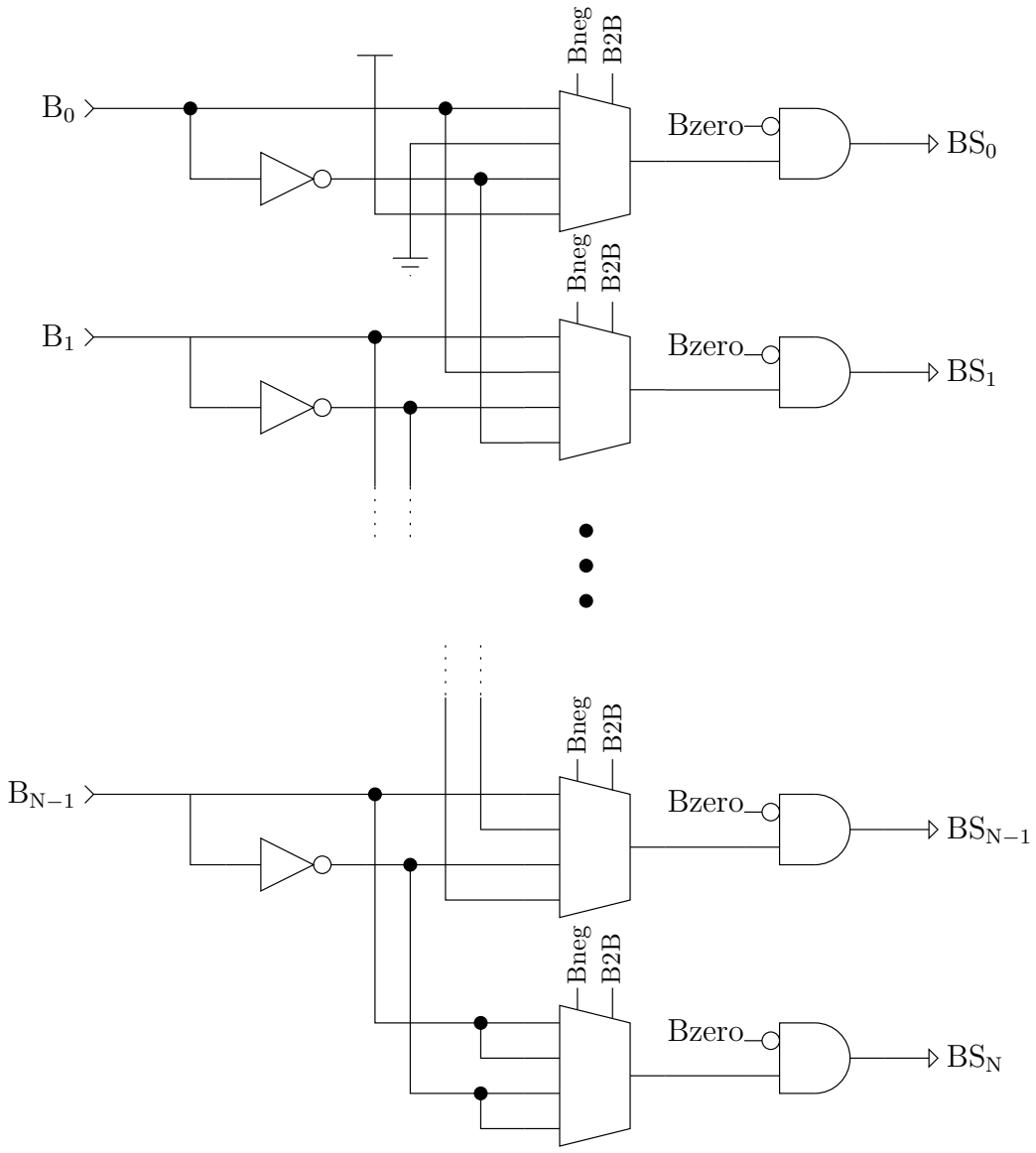


Fig. 2.2. Selector B module.

Register. These carries must be resolved before the two bits can be retired into the final answer. The *Resolve* block takes a shifted out carry bit with two shifted out sum bits and adds them to resolve the shifted out carry. The *Resolve* circuit is shown in Fig. 2.3. The carry-out bit of this circuit is fed back around to the same circuit to be added in on the next cycle. When the iterations are complete, the carry-out bit of this circuit still needs to be resolved. Thus, it is passed to the *Partial Products Summation* unit as a carry-in to that circuit. The *Resolve* circuit generates two final answer bits on each cycle and passes them to *Shift Register A* for temporary storage. As mentioned above, the *Resolve* circuit also injects a *two's*

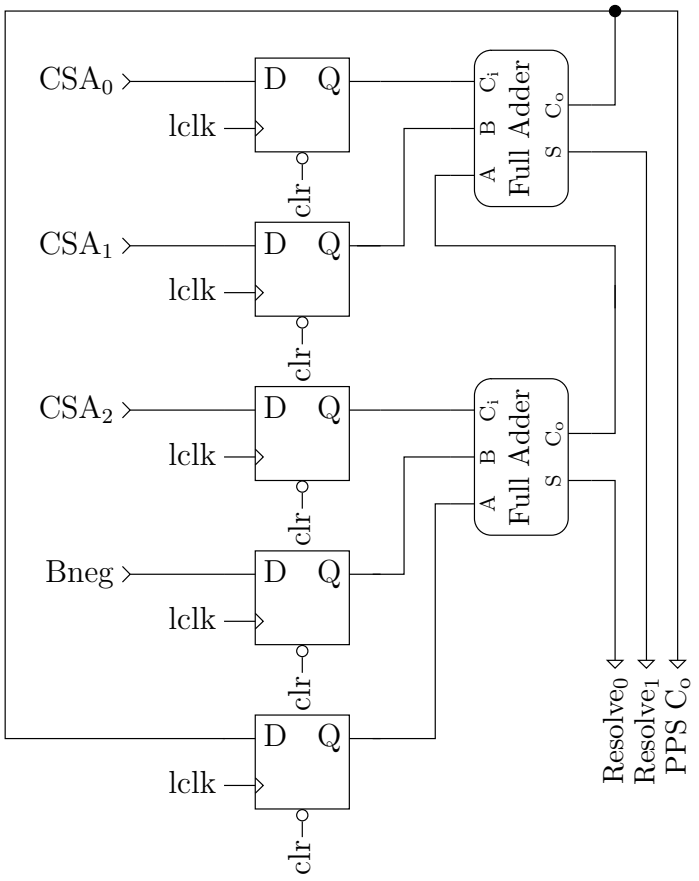


Fig. 2.3. Resolve module.

complement carry when the B operand is negative. The *Decode A* block uses the *Bneg* signal to tell *Resolve* when to do this.

The *Shift Register A* block, shown in Fig. 2.4, loads a new value of A when beginning a multiply. On each cycle after that, the block shifts the current value of A by two bits toward the least significant bit. In the figure, the *Shift Register A* is shifting from top to bottom two bits at a time. The *Shift Register A* passes its two low-order bits and the last shifted out bit to the *Decode A* block. The *Decode A* block uses these three bits to determine the multiple of B to be added to the running sum in the *Carry Save Adder Shift Register*. *Shift Register A* is also used to temporarily store the final answer bits retired from the *Resolve* block during iterations. Two answer bits are shifted in on each cycle. When the iterations are completed, *Shift Register A* holds the lower $N - 2$ final answer bits.

The *Carry Save Adder Shift Register* consists of a single row of full adders fed by flip-flops. Fig. 2.5 shows the interconnection between the full adders and flip-flops. Rather than having the carries ripple through the row, a carry save interconnect is used [19]. This means that carries are passed to flip-flops to be resolved on the next cycle. Therefore the latency of each iteration is independent of the operand width N. Shifting is performed each cycle by passing outputs of the full adders to flip-flops two bits toward the least significant bit (LSB). In the figure, the most significant bit (MSB) is on the top of the page, and the LSB is on the bottom. The shift register must be sign extended when it shifts, so the inputs to the MSB flip-flops come from feedback.

The first cycle after the local clock starts up is used to clear the *Carry Save Adder Shift Register* while *Shift Register A* loads a new value of A in parallel. For the iterations, the multiplier is broken into a three stage pipeline with locally-clocked flip-flops. The first stage consists of *Decode A* and *Selector B*. The second stage is the *Carry Save Adder Shift Register*. And the third stage is the *Resolve* block. *Shift Register A* is not included in a stage because it is essentially a pipeline latch between stages. The first stage has the longest latency and is therefore the critical path.

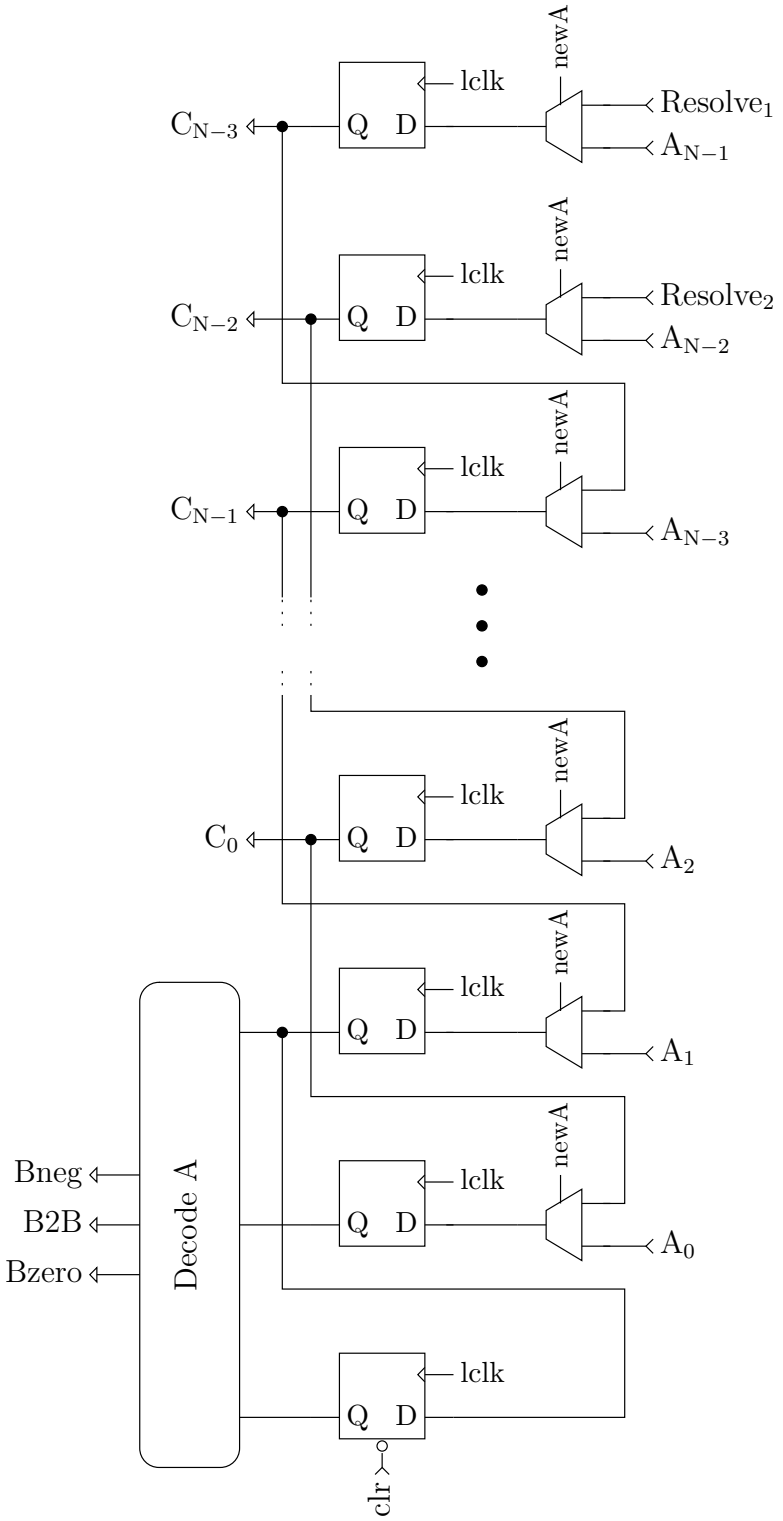


Fig. 2.4. Shift register A and Decode A module.

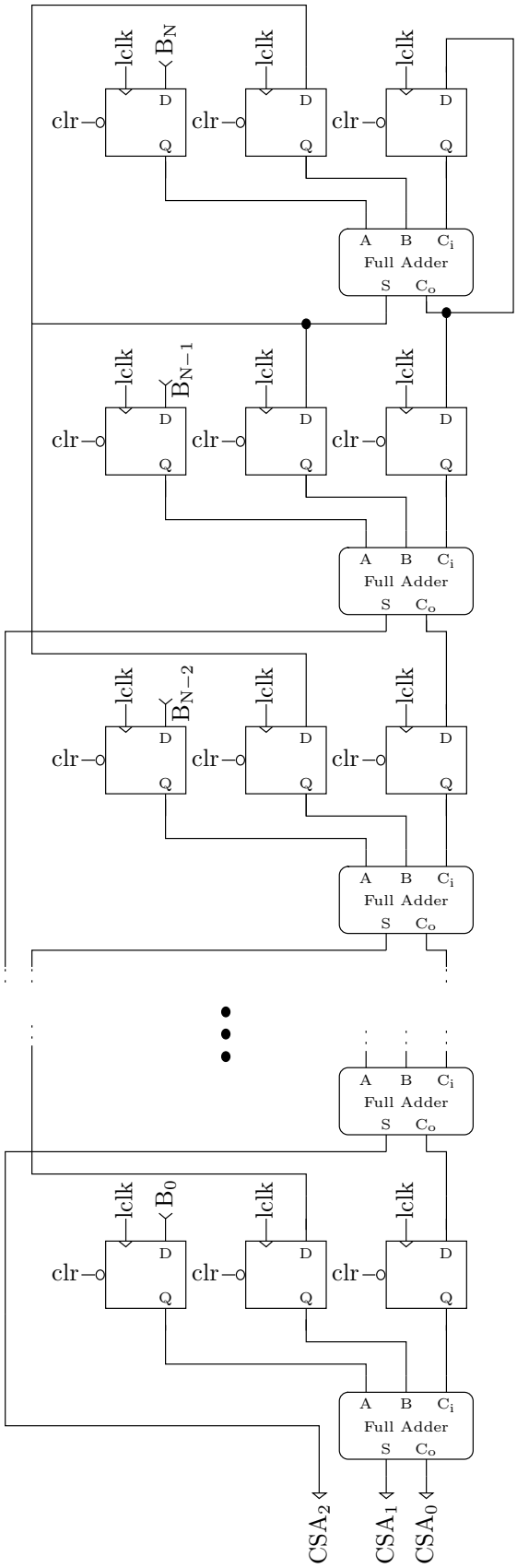


Fig. 2.5. Carry save adder shift register module.

Control for the sequential multiplier is shown in Fig. 2.6. The *Reset* signal is a global reset signal that sets the local clock control in the initial start state. The multiplier is then idle until *gclk* has a rising edge. A rising *gclk* causes *run* to rise, which in turn starts the local clock. The local clock synchronizes a local state machine that keeps track of iterations. When the correct number of iterations is complete, the state machine raises the *stop* signal which causes *run* to fall, which then stops the local clock. Then the cycle repeats as the multiplier is idle and waits for a *gclk* rising edge. Thus the *gclk* signal acts as a start signal for the multiplier. If it is necessary to keep the multiplier idle, the *gclk* signal can be gated off so it produces no rising edges. The state machine in Fig. 2.6 is implemented in a traditional synchronous manner. The *Stoppable Clock* and *Clock Control* blocks are discussed further in Chapter 3.

When iterations have completed, the lower $N-2$ bits of the answer have already been resolved and are held in *Shift Register A*. The *Resolve* block holds the next two resolved answer bits, plus one unresolved carry bit. The *Carry Save Adder Shift Register* holds the upper N final answer bits in unresolved form. The multiplier is broken into a two stage pipeline clocked by the global clock — the iterative portion, and the *Partial Products Summation* block. The final answer bits, are passed to the *Partial Products Summation* block to be resolved on the next global clock cycle. The *Partial Products Summation* block consists of a basic ripple-carry adder. It is

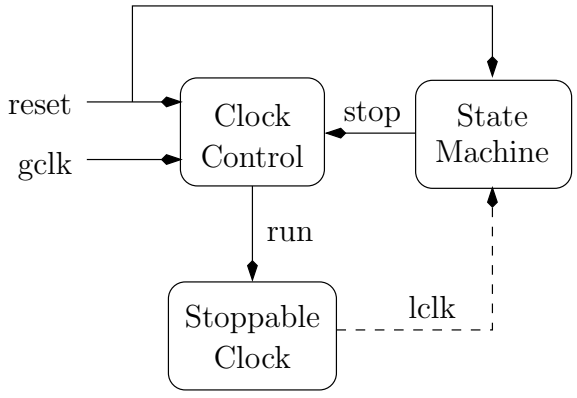


Fig. 2.6. Stoppable clock interface architecture.

much faster than the iterative portion of the multiplier; therefore, it is not necessary to optimize it.

2.3 Preliminary Analysis

A high-level analysis of this locally-clocked sequential multiplier is performed to understand how it scales with word size. Estimates for latency, area, and energy are calculated for the multiplier implemented at various word sizes. These estimates are compared with similar estimates of a combinational full parallel array multiplier. This is simply a baseline comparison to understand how the sequential multiplier performs. It is assumed that the reader can relate other multiplier designs to the combinational array for further comparisons.

The only difference in operation of the parallel array and the sequential multiplier is that the parallel array does multiplications on unsigned positive numbers rather than signed *two's complement* numbers. The array can be used to multiply *two's complement* numbers if the operands are converted to unsigned numbers first and the final answer is converted back to a *two's complement* number. These sign-magnitude conversions are not included in the comparisons.

Timing numbers for gates used in these comparisons are from Atmel corporation's $0.8\mu m$ process. SONIC Innovations used this process for the design of one of their low voltage hearing aids. They provided timing numbers for a 0.8 shrink of their $0.8\mu m$ process gates with a VDD of 1.25 V [20]. These numbers are incorporated in the simulations used to estimate energy consumption. They are also used when estimating latency of the multipliers.

The energy per operation of the sequential multiplier is compared to that of the combinational design. Energy is chosen over power for this comparison because power depends on the average time it takes to perform an operation [21]. If the parallel array is run at maximum speed, its power would be far above that of the sequential design due to its higher frequency. Therefore, such a comparison can be misleading. The energy per operation is independent of the amount of time each design takes to complete and is akin to running each multiplier at the same global

clock frequency without any voltage scaling. An estimate of the energy consumed per operation is obtained using simulation over a large set of random data.

Energy is consumed in a node when it switches either from high-to-low or low-to-high. The amount of energy consumed in either case is

$$E = \frac{1}{2}CV^2,$$

where C is the output capacitance connected to that node and V is the voltage swing of the transition. Energy consumed by an entire module is calculated as

$$E_{\text{total}} = \frac{1}{2}V^2 \sum_i n_i C_i,$$

where i ranges over all nodes in the module. V is the voltage swing that varies depending on what VDD is set to. C_i is the capacitive load connected to the output of node i. Since transistor sizing is not considered here, input capacitances on gates are assumed to be equal for all gates. This reduces the capacitive load calculation at node i to $C_i = FO_i C_g$, where FO_i is the fanout of node i and C_g is a constant gate input capacitance. The n_i term is called the activity factor of node i. The activity factor is defined to be the average number of times node i switches (either high-to-low or low-to-high) during each multiply. Simulation of a large set of random multiplies is used to find the n_i term for each node of a module.

A plot of normalized energy estimates on the typical corner for various sizes of N is found in Fig. 2.7. The normalization constant is the energy of the combinational design on the typical corner for $N = 12$. Energy grows polynomially with word size for both designs, but the coefficient for the sequential design is much smaller than the combinational design. Table 2.1 shows normalized energy estimates for various sizes of N and different process corners. The polynomial growth of energy for the sequential design is most easily seen in the typical corner column of Table 2.1. The self-timed design has polynomial growth for energy because it has on the order of $O(N)$ full adders used on the order of $O(N)$ times.

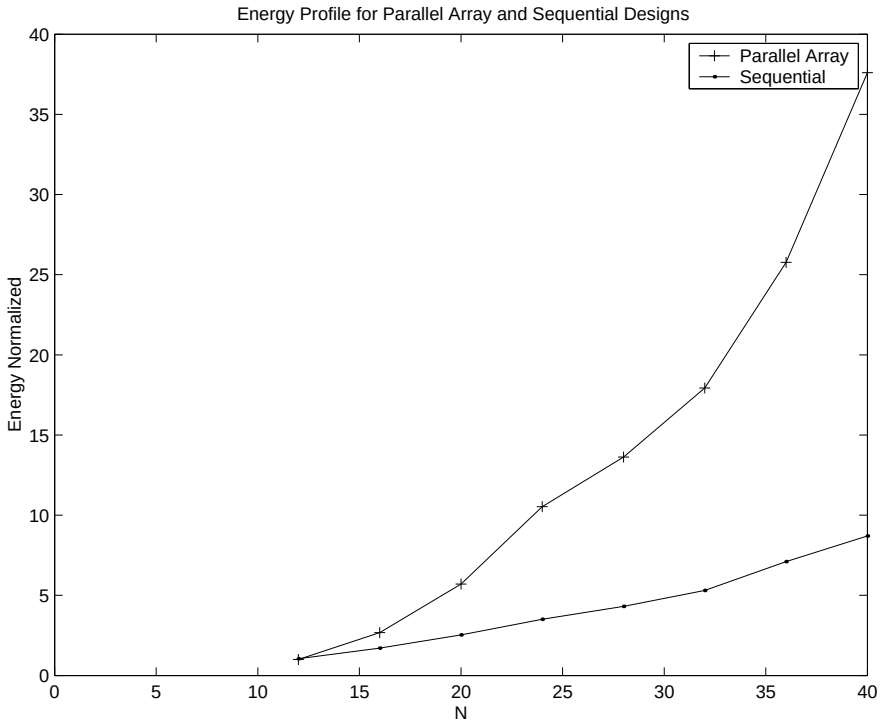


Fig. 2.7. Plot of energy estimates as word size increases.

TABLE 2.1
Energy estimates normalized by N=12 combinational slow

	Sequential			Combinational		
N	Fast	Typ	Slow	Fast	Typ	Slow
12	1.54	1.24	1.03	1.45	1.19	1.00
16	2.53	2.03	1.67	3.92	3.19	2.65
20	3.74	3.01	2.50	8.47	6.80	5.59
24	5.19	4.18	3.47	15.82	12.55	10.24
28	6.41	5.14	4.25	20.92	16.24	13.04
32	7.85	6.32	5.21	27.97	21.37	16.96

The combinational design has on the order of $O(N^2)$ full adders that switch an average of $O(N)$ times per multiply. It is important to note that the energy estimates assume no correlation between the incoming operands, so the probability of any input bit transitioning for subsequent multiplies is 50%. In many applications, there is less switching frequency due to inherent data correlation. Thus, these energy estimates are conservatively large.

Fig. 2.8 shows the area plot of the sequential and combinational multiplier designs. Area is measured in terms of transistor count. The value is calculated by first counting each type of gate and flip-flop in each design. Each individual gate count is multiplied by the number of transistors required to implement the gate or flip-flop. The area calculation includes some electrical and fanout considerations by conservatively adding buffer trees on high fanout nodes. This area calculation does not account for the actual size of the transistors. In addition the area due to routing is neglected and is assumed to be of equal cost in both designs. This benefits the parallel array, which clearly has a larger routing penalty. As shown in Fig. 2.8, the two designs have approximately equal area at $N = 7$. The area growth of the sequential design is linear in N, while the combinational design follows N^2. Table 2.2 shows the transistor count for various sizes of N. The scale factor column is the normalized area using the area of the sequential design at $N = 12$ as the normalization constant.

The sequential design has an obvious advantage over the combinational design in terms of area and energy consumption. Yet, this advantage comes at the cost of a latency penalty. Latency numbers are found using paper and pencil methods. The critical path in each design is calculated by hand. Then timing numbers are applied to each gate on the critical path. The latency calculation takes into account capacitive loading on large fanout gates. Fig. 2.9 shows the latency of each design in nanoseconds for various sizes of N. The latency for both designs scales linearly with N, but the sequential design has a larger constant. Table 2.3 shows the normalized latency of the sequential design compared to the combinational design. The normalization constant is 45.71ns. This constant is the latency of the

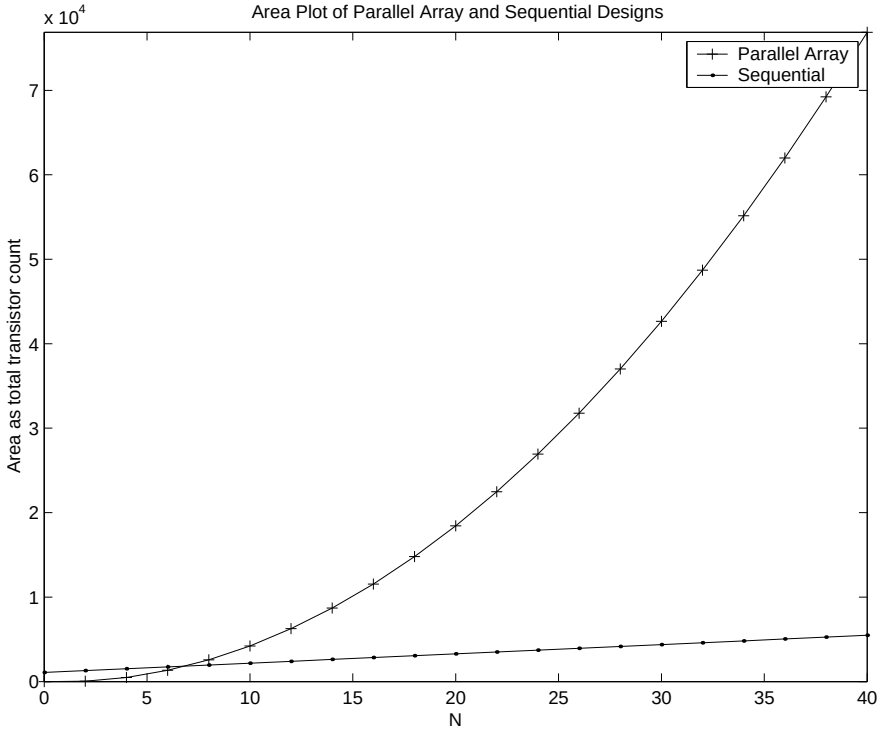


Fig. 2.8. Plot of area estimates as word size increases.

TABLE 2.2
Area estimates as number of transistors

	Sequential		Combinational	
N	Area	Scale	Area	Scale
12	2400	1.00	6264	2.61
16	2840	1.18	11552	4.81
20	3280	1.37	18440	7.68
24	3720	1.55	26928	11.22
28	4160	1.73	37016	15.42
32	4600	1.92	48704	20.29

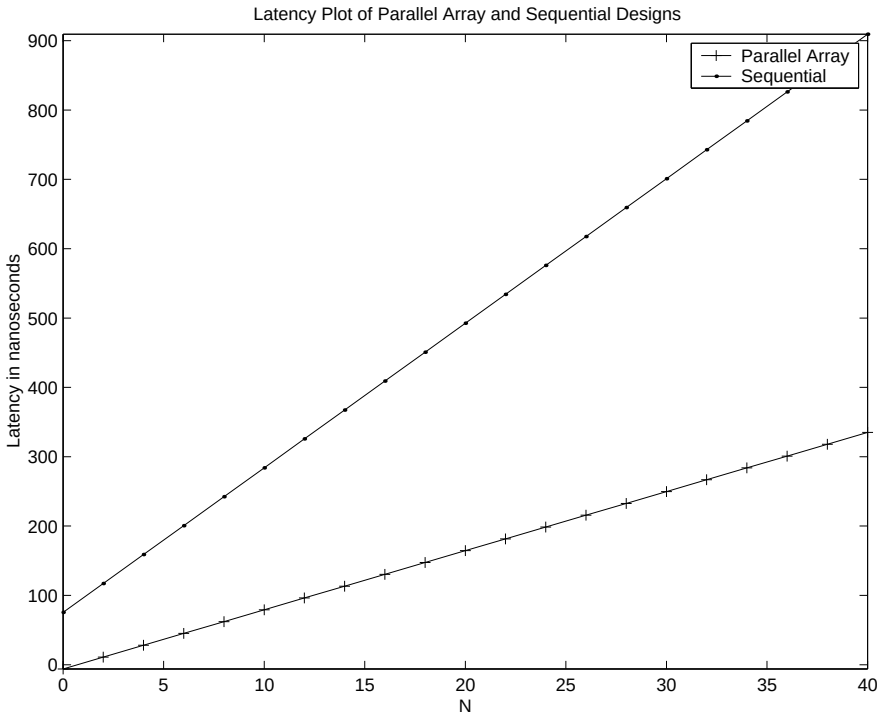


Fig. 2.9. Plot of latency estimates as word size increases.

TABLE 2.3
Latency estimates normalized by N=12 combinational fast

	Sequential			Combinational		
N	Fast	Typ	Slow	Fast	Typ	Slow
12	3.36	7.12	20.10	1.00	2.10	5.97
16	4.22	8.95	25.18	1.35	2.85	8.10
20	5.08	10.77	30.26	1.71	3.60	10.20
24	5.95	12.60	35.35	2.06	4.35	12.32
28	6.81	14.42	40.43	2.42	5.09	14.44
32	7.67	16.25	45.52	2.77	5.84	16.56

combinational design on the fast corner for $N = 12$.

A few latency optimizations for the sequential multiplier exist. For example, pipeline stages can be rearranged. Currently, the decode of A and selection of B is extremely long compared to the other stages. It is possible to move the locally-clocked pipeline latches or to implement time-sharing. Both options would allow the local clock frequency to be increased. Another approach is to design with aggressive circuit families. In the current implementation, the multiplier is restricted to use only basic standard-cells. Moving to more aggressive circuit families would increase design time and complexity. Since one of our goals is simplicity and ease of design, these alternative circuit families are not considered here.

Preliminary latency, energy, and area estimates look favorable for a sequential multiplier. Thus a locally-clocked module could be an advantage for power and area critical systems. Even so, the high-level estimates are achieved through simulation and paper and pencil methods. Actual fabrication is needed to confirm the estimates made. In addition to verifying the estimates, fabrication helps determine the pros and cons of using stoppable clocks.

2.4 Test Circuitry and Test Board

A 20-bit implementation of the multiplier presented in this chapter was fabricated through MOSIS using AMI's 0.5μm process. This section describes the on-chip testing circuitry and test board. The operands of the multiplier can be applied from three different sources: the off-chip operand bus, an operand latch which latches from the operand bus, or a built in self-test (BIST) structure [22]. The BIST structure allows at-speed testing and generates a single pass/fail signal on chip. With a functional BIST, the external test circuitry is less complicated. The BIST is composed of two main parts: a test pattern generator and a result evaluator.

An exhaustive test pattern generator can be made from an N-bit counter where N is the desired width of the input. Nevertheless, a linear feedback shift register (LFSR) is much smaller and easier to design. With appropriate feedback taps,

an LFSR has a period of $2^N - 1$. This type of LFSR is called a maximal-length sequence generator or a pseudo-random number generator. Appropriate feedback taps for LFSRs of size $N = 3$ to $N = 168$ are found in [23]. This reference also contains a literature survey of the mathematics behind maximal-length sequences. An example of a 3-bit LFSR of maximal-length is shown in Fig. 2.10. The output sequence from this circuit is located in Table 2.4. The circuit cycles through all eight possible 3-bit patterns except 0. The sequence is pseudo-random because it is generated by a mathematically predictable circuit. Yet, for practical purposes it is a random sequence.

To exhaustively test a 20-bit multiplier, a 40-bit LFSR is necessary. Unfortunately, a 40-bit exhaustive test for this multiplier takes too long to be practical. For example, with a 5 volt power supply the multiplier runs at 13.3 MHz and takes 23 hours to test 2^{40} multiplies. At 1.6 V the multiplier runs at 3.13 MHz and takes 4 days to test! Rather, a 20-bit LFSR is designed to feed the B operand while the A operand is held at one value. In this situation, it takes a tenth of a second to test at 5 V VDD, and three tenths of a second to test at 1.6 V VDD. This still gives excellent test coverage because of the iterative nature of the multiplier. The A operand is decoded three bits at a time, and can be set to a value which is decoded into all eight possibilities. Therefore, the B operand logic is exhaustively tested with the LFSR, and the A decode logic is exhaustively tested by setting A for maximal 3-bit decode sequences.

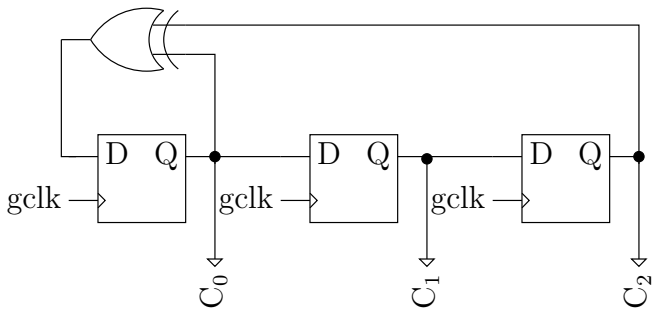


Fig. 2.10. 3-bit linear feedback shift register.

TABLE 2.4
3-bit linear feedback shift register output

C_0	C_1	C_2
0	0	1
1	0	0
1	1	0
1	1	1
0	1	1
1	0	1
0	1	0
0	0	1
	⋮	

Even with 20-bit rather than 40-bit exhaustive tests, the test runs through $2^{20} - 1 = 1,048,575$ (or 1M) multiplies. It is impractical to store all multiply answers on the chip. Rather than recording the answers off-chip at-speed, the answers can be compressed into a single word through the use of a signature file. A signature file is a multiple input shift register (MISR), which is a mathematical extension of an LFSR. A mathematical evaluation of signature files is found in [24], which also contains a survey of literature on signature files. A 3-bit MISR is shown in Fig. 2.11. The result word is xor'ed into the LFSR along with the regular LFSR feedback taps. A good signature must be generated either by a circuit known to be correct, or through simulation. A C-code simulation of this multiplier and 20-bit signature file is written to generate good signatures.

Compressing a sequence of 1M answers down to one causes a loss of information which means that some faulty answers may not be detected. This is called aliasing or fault masking. According to [24], if all possible errors are equally probable, the probability of failing to detect an error (the aliasing probability) is

$$P_{ap} = \frac{\#\text{ undetectable errors}}{\#\text{ possible errors}} = \frac{2^{L-N} - 1}{2^L - 1},$$

where L is the length of the sequence and N is the size of the signature file. When

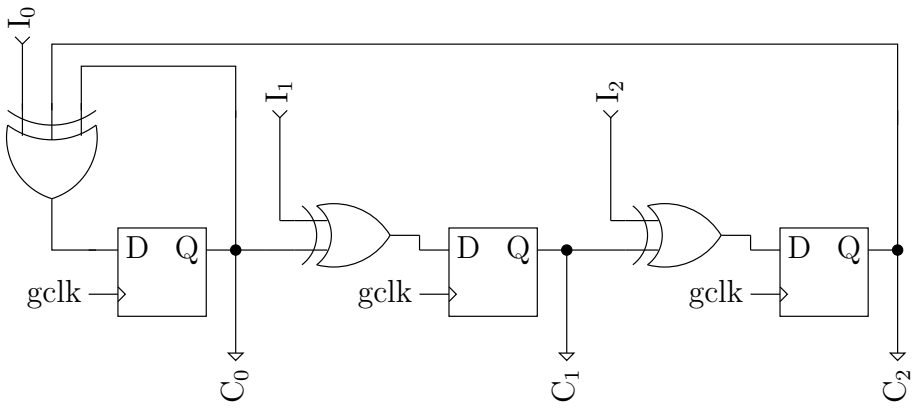


Fig. 2.11. 3-bit multiple input shift register.

$L >> N$, the probability reduces to

$$P_{ap} \approx \frac{1}{2^N}.$$

For a 40 bit signature file and a sequence of 1M multiplies, $P_{ap} \approx 9.1 \times 10^{-13}$. Unfortunately, all possible errors are not equally probable, so P_{ap} is higher but still negligible.

The multiplier functionality is tested through the structures described above. In addition to the functionality tests, latency and power numbers are desired. To test power, the chip has three separate VDD lines: one for the multiplier, one for the local clock, and one for the test circuitry. The measured average current draw on a VDD line can be used to calculate average power for the circuit it feeds. Latency is tested by routing clock control signals off-chip. One signal indicates when the multiplier is done. This signal can be viewed on an oscilloscope to determine how long the multiply takes.

For detailed testing of local clocks, three clocks are placed on the chip with their outputs routed off-chip. A critical path delay signal is routed off-chip for comparison to the local clock delays. Because of the BIST on-chip, the test board is relatively simple to design. The main test board component is a 40-bit data path bus. This bus can be driven by the signature file on-chip, the multiplier on-chip,

or by hex switches on the test board. The test board provides push-buttons for single-stepping the multiplier. Many internal chip signals are routed to LEDs on the test board for use in single-step mode. The multiplier latches can be synchronized by the on-chip local clock or from off-chip during single-step mode.

The chip and test board have a feedback mode where the multiplier continuously multiplies and the signature is checked every 1M multiplies. When used in conjunction with the B operand pseudo-random number generator, this mode gives a continuous pass/fail check. It can be used to test the multiplier while changing variables such as VDD and temperature. With the test circuitry and test board described here, a detailed analysis of the multiplier and stoppable clocks is performed.

CHAPTER 3

STOPPABLE CLOCK DESIGN

High-level evaluation of the multiplier in Chapter 2 neglected the details of the stoppable clock. This chapter focuses on the design of our stoppable clock. It presents a literature survey of stoppable clock and delay element work. Then the architecture for our clock is described, including the control and delay element design. SPICE simulation results showing the functionality of the stoppable clock are presented last.

3.1 Stoppable Clock Related Work

Stoppable clocks were used at Evans and Sutherland in the 1960's [25, 26]. According to [25], "it is the clock pulses that are *not* generated that are the key to the value of the stoppable clock." Stoppable clocks have been further developed for use in *Globally Asynchronous, Locally Synchronous* (GALS) systems [27, 28, 29, 30]. These systems use locally-clocked modules with two- or four-phase handshake protocols between the modules. GSLS modules are similar to GALS modules in their use of locally-clocked modules. Nevertheless, GSLS modules do not have the overhead of asynchronous communication between modules. It is assumed that GSLS modules have completed computation before the next global clock cycle arrives. In addition, the interface to GSLS modules is different from that of GALS modules and therefore requires further evaluation.

One major use for stoppable clocks is to avoid synchronization failure. When inputs are asynchronous with respect to the clock, they can cause the input latches to become metastable. Metastable latches can in turn cause a synchronization failure if they do not stabilize before the inputs are needed locally. In order to avoid synchronization failure, a stoppable clock can be used [31, 32, 27, 28, 33, 34].

When latches become metastable, the clock is stopped until the latches stabilize. This is a way to synchronize independently clocked systems. GSLS modules exist in a synchronous environment, so their inputs and outputs are synchronized to the global clock. Therefore, synchronization failure is not an issue for GSLS modules if set-up and hold times are not violated by the environment or the module itself.

Design of stoppable clocks is difficult due to timing assumptions [31, 35]. If the signal used to stop the clock is asynchronous with respect to the local clock signal, a mutual exclusion (ME) element is needed. The ME makes sure that clock pulses and stop-clock signals do not cause the ring oscillator to generate runt pulses or thin pulses. Runt pulses can be perceived in later circuitry as meaningful pulses causing erroneous latching. One example of a stoppable clock with an ME is in Fig. 3.1. The *stop_ack* signal is necessary because there is no time constraint on when the clock is cut off. It is only known to be cut off when *stop_ack* rises.

If the ME in Fig. 3.1 were replaced with an *And* gate as in Fig. 3.2, it introduces new timing constraints. If these timing constraints are violated, it is possible to cause runt pulses and thin pulses on the *lclk* line. For instance, if *stop* goes high at the same time, or just after *A* rises, a runt pulse occurs on *lclk*. If *stop* goes high

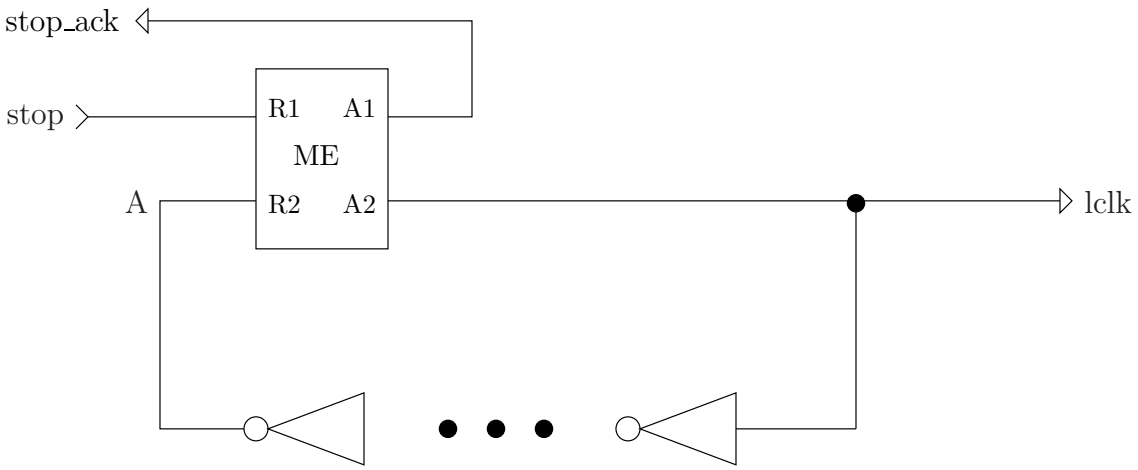


Fig. 3.1. Stoppable clock with an ME.

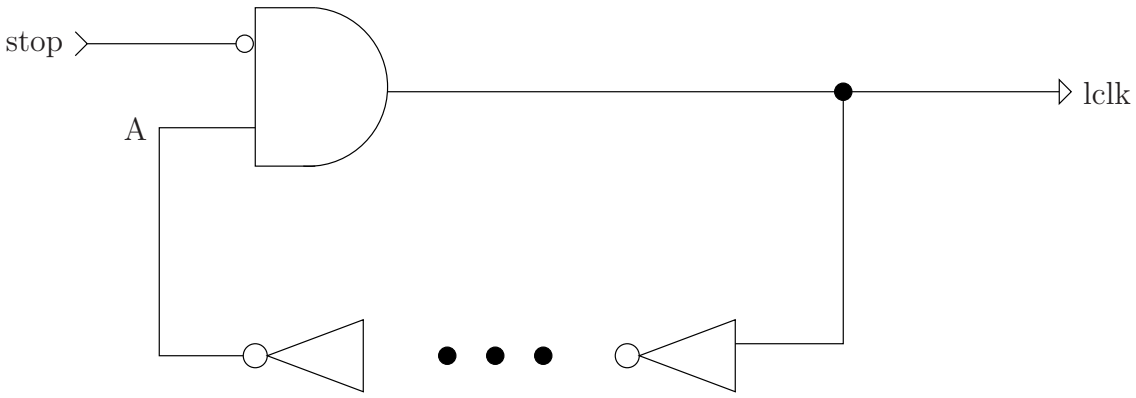


Fig. 3.2. Stoppable clock with an *And* gate.

any time while A is high, it forces *lclk* to go low; making a thin pulse. On the other hand, if the *stop* signal going high is timed so that it comes some time after A goes low, and some time before A goes high again, the ME can be replaced with an *And* gate safely. In this situation, the *stop* signal is synchronized with the clock pulses. This synchronized situation allows the design to remain in the standard-cell realm, thus facilitating ease of design.

A third cutoff gate is shown in Fig. 3.3. This clock is presented in [35]. The cutoff gate of this clock is a generalized C-element (gC). It eliminates one of the timing constraints of the clock in Fig. 3.2. If *stop* goes high while A is high, the gC waits until A goes low before cutting off the clock. Thus, the clock of Fig. 3.3 only constrains *stop* from going high too close to a rising edge on A.

An alternative system is proposed in [35]. It is a *Globally Synchronous, Locally Asynchronous* (GSLA) system. In this system, a stoppable clock is distributed to the entire synchronous design. Individual pipeline stages can be synchronous or asynchronous. If local asynchronous modules are not done computing within a given clock cycle, they stall the clock and therefore the entire design. This system, and the others described above, bring up many issues pertaining to stoppable clocks. Namely:

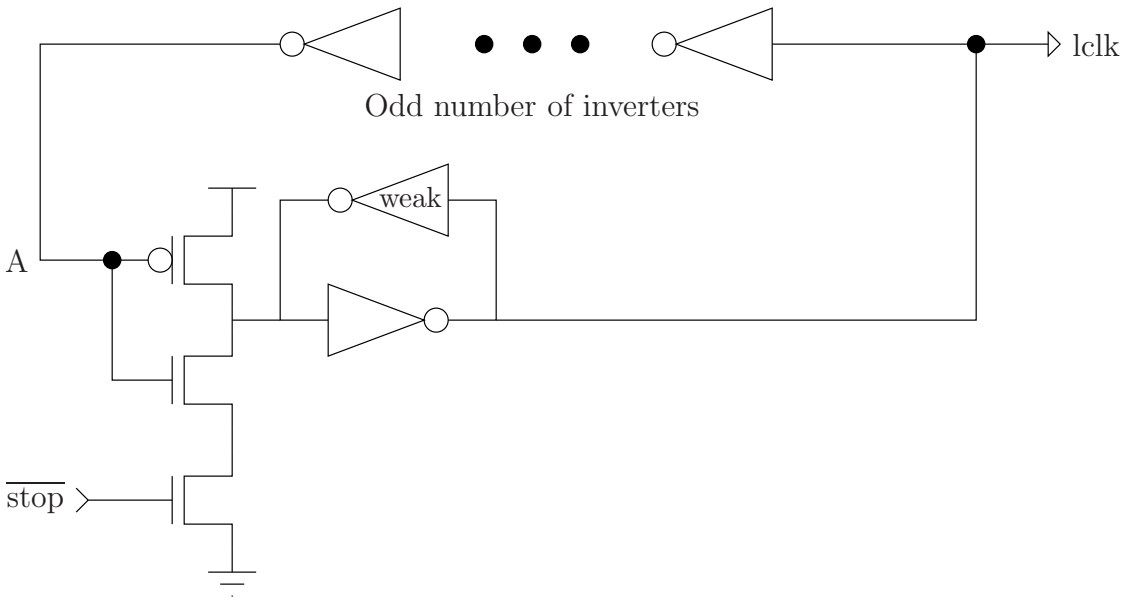


Fig. 3.3. Stoppable clock with a gC.

- lack of jitter control,
- latency overhead for starting and stopping the clock,
- how to make the delay element,
- and how much tuning is needed.

This thesis addresses some of these very issues.

3.2 Delay Element Related Work

The core of an on-chip stoppable clock is a delay element. Tunable delay elements and clocks are discussed in [28, 36, 37, 17]. An analog voltage is used to set the delay in [28]; however, the circuitry for the delay is not standard-cell nor is it easy to design and implement. The notion of taps on the delay line is introduced in [36, 37, 17]. A set of signals are used to gate portions of the delay line to make it faster or slower. Tuning is helpful when trying to match the delay to the critical path of a fabricated chip. A stoppable clock that is self tuning is implemented in [36]. The stoppable clock references an off chip crystal and calibrates itself. This

makes the internal clock have a fixed frequency. This method can be used to match the critical path in the sequential multiplier of Chapter 2. Nevertheless, it adds an unnecessary overhead of circuitry and design time when a tunable clock without a fixed frequency also works.

The choice in gates for the delay element has a direct impact on the amount of tuning the local clock needs. A chain of inverters can be used as a delay element; however, inverters do not scale the same as the components on the critical path. Process variation, temperature, and voltage swings have different effects on gates of different types. To illustrate this point, timing numbers for Atmel corporation's $0.8\mu m$ process at 1.25 V are used again. With gates running at the slow corner, it takes 18 inverters to match the worst-case critical path of the sequential multiplier. In that same process, if all components are running at the fast corner, 44 inverters are needed to match the worst-case critical path delay. One solution to the mismatch in gate scaling is to make the ring oscillator delay out of gates that match the critical path [14]. Fanout capacitance for each gate is also duplicated to help ensure that the ring oscillator delay matches that of the critical path throughout all conditions. If matching is done carefully, the tuning required is lessened and possibly eliminated.

When matching with inverters, it is easiest to match half the critical path. Then it requires two passes through an inverting delay element to generate one clock cycle as in Fig. 3.4(a). This creates an approximate 50% duty cycle square wave which makes using standard edge-triggered flip-flops possible. On the other hand, when trying to match the critical path using similar gates, it is difficult to design a two pass, 50% duty cycle delay. Two solutions exist when matching the entire critical path delay using similar gates. In this situation, one pass through the delay must equal one clock cycle and must trigger the flip-flops. Therefore, a pulse can be sent through a noninverting delay element and to the flip-flops as in Fig. 3.4(b). Unfortunately, rise and fall times differ enough that the pulse can be lost or distorted within the delay element as it cycles. Another solution is to send single transitions through the delay element and use double edge triggered

D-flip-flops (DETDFF) as in Fig. 3.4(c) [38]. These DETDFFs are two times as big as standard D-flip-flops (DFF). The larger area also implies larger capacitance which in turn implies more energy consumption for using DETDFFs. Rather, a solution which is more area and energy efficient is sought for in our design.

3.3 Our Stoppable Clock

This thesis leverages stoppable clock and delay element related work to design and evaluate a stoppable clock. The desirable characteristics from multiple stoppable clocks are combined into a full-featured clock design. Preliminary design of the clock circuit is shown in Fig. 3.5. This stoppable clock uses a delay element made from the same gates as those of the multiplier critical path. Thus one pass through the delay element constitutes one clock cycle and must cause latching in the flip-flops. To do this, a *One-shot* is placed on the output of the traditional

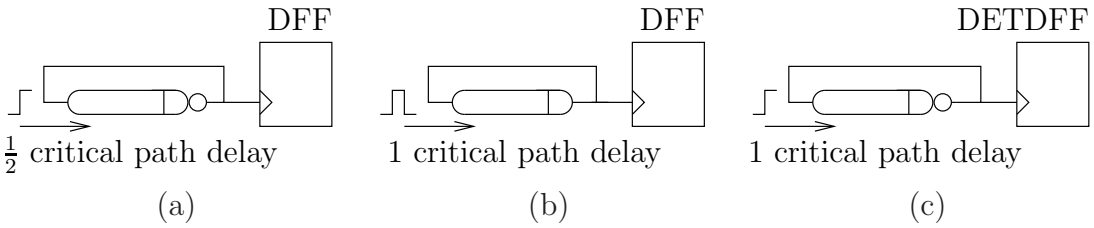


Fig. 3.4. Types of ring oscillators. (a) Delay made of inverters, single transitions flowing through the delay, and standard DFFs used. (b) Delay made of critical path gates, pulses flowing through the delay, and standard DFFs used. (c) Delay made of critical path gates, single transitions flowing through the delay, and DETDFFs used.

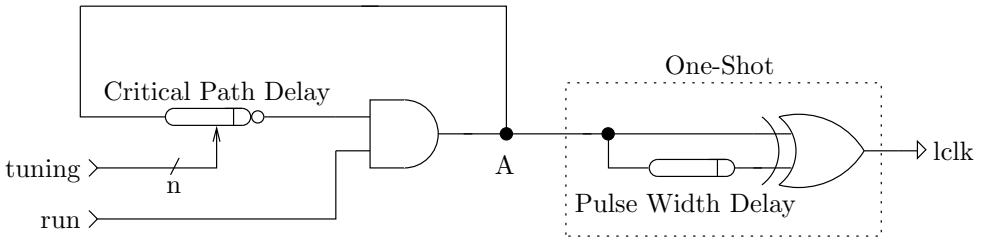


Fig. 3.5. Initial stoppable clock design.

ring oscillator — node A. Every transition that occurs on A causes the *One-shot* to send a pulse to the flip-flops. In this way, standard DFFs are used and the entire critical path is matched in the delay element. The *One-shot* delay adds to the start up overhead of the local-clock, but in steady-state oscillating the *One-shot* has no effect on performance. Additional buffers with taps are added to the delay element to make the clock tunable from off chip after fabrication.

This clock is designed to synchronize a locally-clocked module in a GSLS system. When the clock is idle, *run* is low. A global clock rising edge causes *run* to rise which makes the local clock generate clock pulses. A local state machine causes *run* to fall when the multiply is done. This local state machine is synchronized by the local clock. The clock design in Fig. 3.5 does not use an ME to stop the clock. It is able to do this because the cutoff signal (i.e., *run* falling) is synchronized with the local clock. It is assumed that the local clock is stopped before the next global clock rising edge.

One drawback to using an *And* gate as the cutoff gate is that the number of pulses generated is always even. Fig. 3.6 illustrates this point. Every transition on A in Fig. 3.6(a) causes a pulse on the *lclk* signal. Fig. 3.6(b) is a waveform showing correct behavior because the *run* signal is lowered after an even number of transitions on A. If an attempt is made to cut off the clock after five pulses, as in Fig. 3.6(c), the *And* gate still causes a sixth transition on A. Not only is there an extra pulse, but the extra pulse comes earlier than the sixth pulse should.

To overcome this drawback, a transparent latch is used as the cutoff gate. Fig. 3.7(a) contains this more flexible local clock circuit. The waveform in Fig. 3.7(b) shows how the circuit can accommodate an even or an odd number of pulses. After three transitions on A, *run* is lowered, and the cutoff latch holds a high value. The clock is started again by raising *run*. This time *run* is lowered after two transitions, and the cutoff latch holds a high value again. This local clock works independent of the parity of transitions on A or the initial value of A.

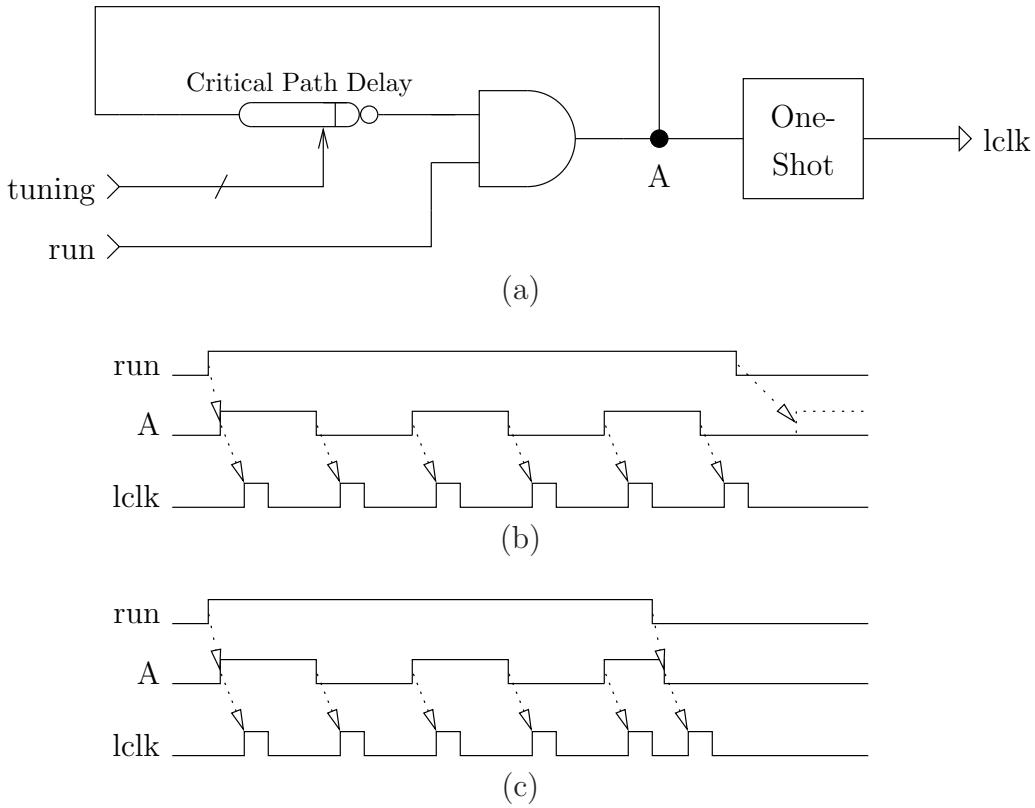


Fig. 3.6. Local clock supporting an even number of pulses. (a) The local clock circuit. (b) Waveform showing an even number of pulses. (c) Waveform showing an error.

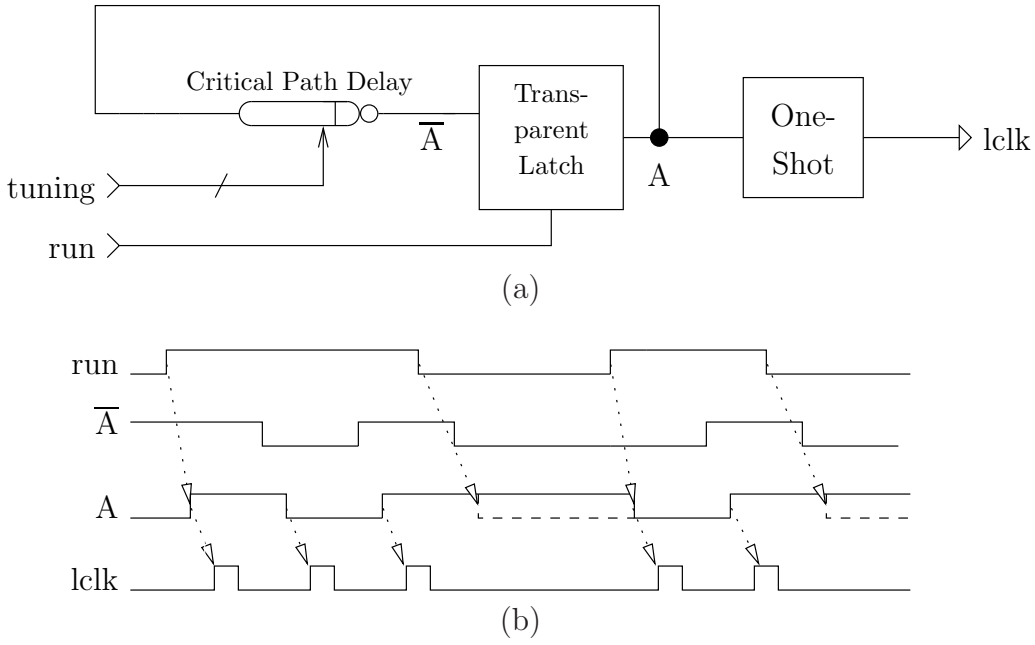


Fig. 3.7. Local clock supporting an even or odd number of pulses. (a) The local clock circuit. (b) Waveform showing an even and odd number of pulses.

3.4 Delay Element Design

Our stoppable clock uses a delay element made from the same gates as those of the multiplier critical path. The critical path of the multiplier is found in Fig. 3.8. In this circuit, *dec1* is routed across many bit-slices of the multiplier; therefore, *dec1* is a long wire in layout. The final DFF in the figure is not part of the critical path delay. Nevertheless, the setup time on this gate must be added to the overall delay of the clock. It is noted that the critical path delay starts with a DFF. Making the clock delay start with a DFF implies that the delay input must be pulsed. To accomplish this, the feedback for the clock is taken from after the *One-shot* as shown in Fig. 3.9. In the circuit of Fig. 3.9, the transparent latch and *One-shot* add to the overall delay of the clock; therefore, the clock frequency is conservatively slower. The circuit in Fig. 3.9 is the final clock design in use on the fabricated chip.

Clock tuning circuitry comes between the delay element and the transparent latch; however, the tuning circuit is noninverting so it only serves to change the delay latency. The clock tuner is shown in Fig. 3.10. The *td01* and *td02* gates

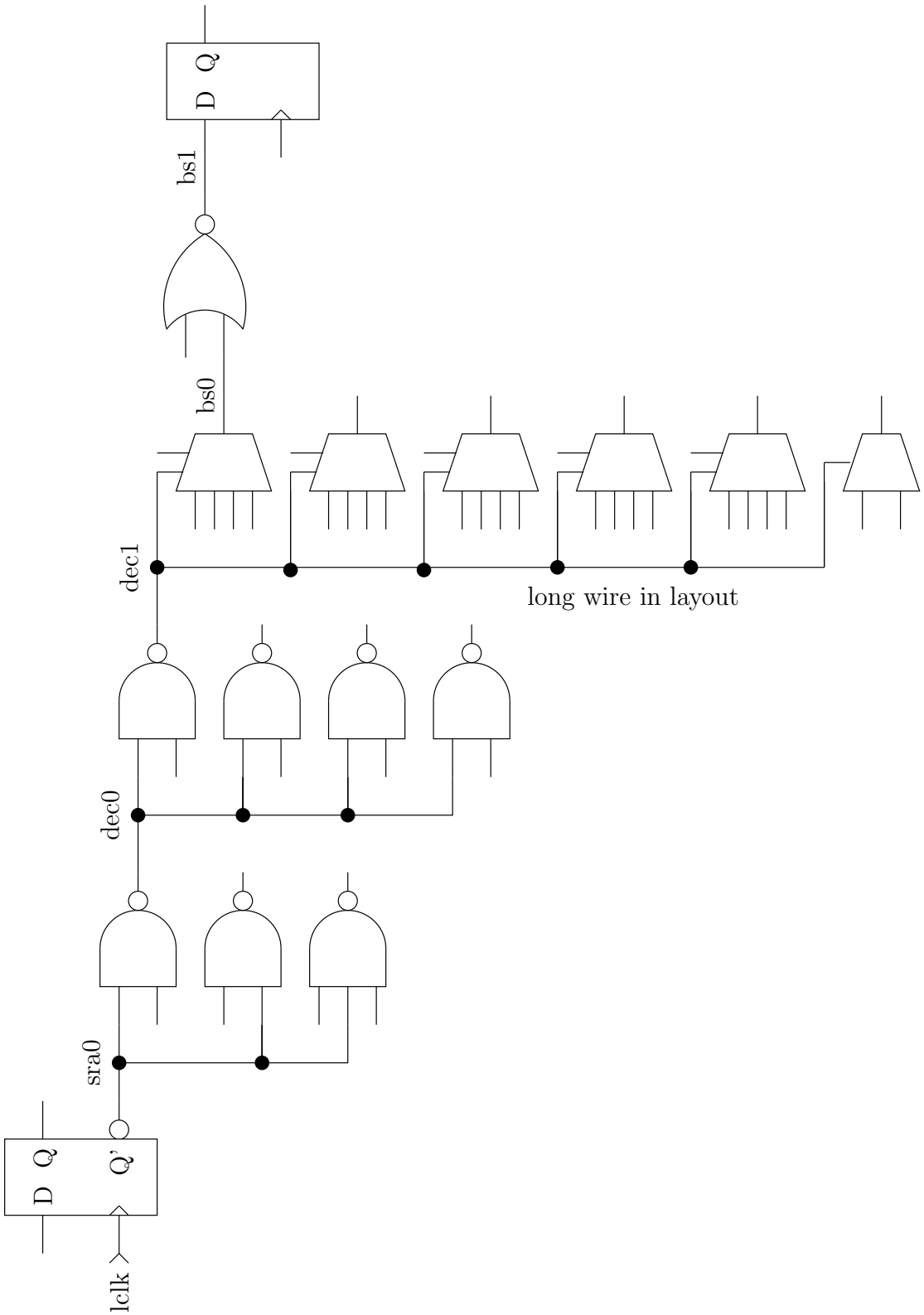


Fig. 3.8. The multiplier critical path.

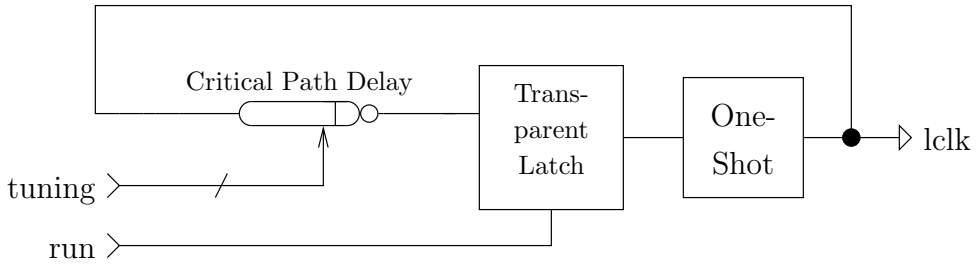


Fig. 3.9. Final stoppable clock design.

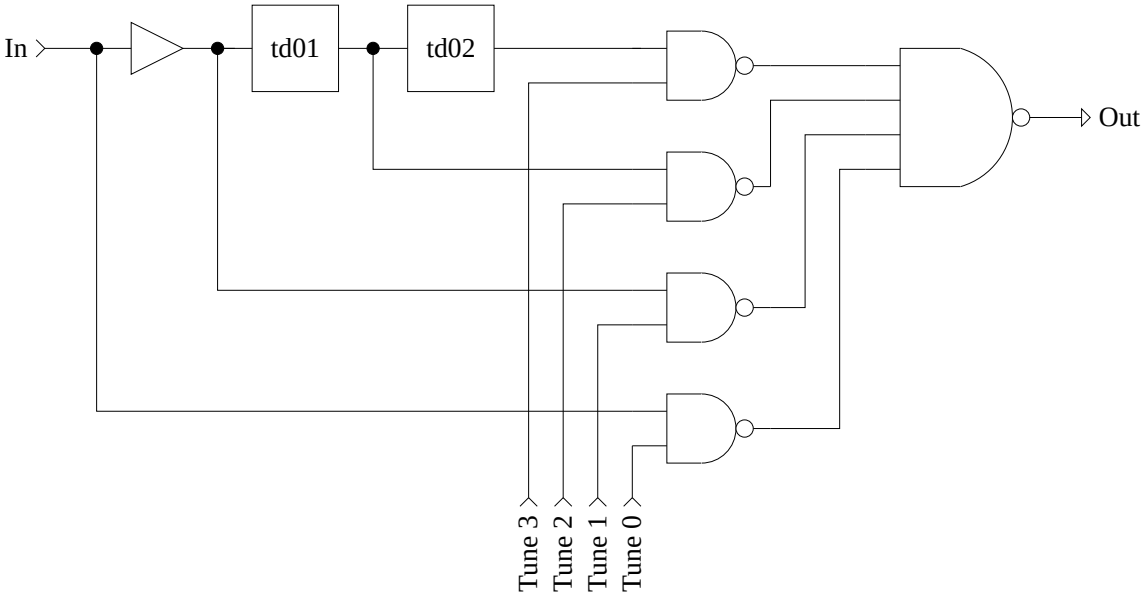


Fig. 3.10. Tuning delay circuit.

are delay elements of about 1 ns and 2 ns respectively. The four tuning inputs are one-hot encoded. The Tune 0 setting adds the minimal amount of conservative tuning — a *Nand2* and a *Nand4*. So the overall minimal conservative delay consists of a *Nand2*, a *Nand4*, a transparent latch, and an *Xor2* from the *One-shot*.

The circuit used to match the critical path delay of Fig. 3.8 is shown in Fig. 3.11. Each gate acts like an inverter; including the *mux* which switches between the 0 input and the 2 input each cycle. With five inverters, the delay is an inverting delay; thus, every clock cycle *bs1* makes a new transition. The output of this delay

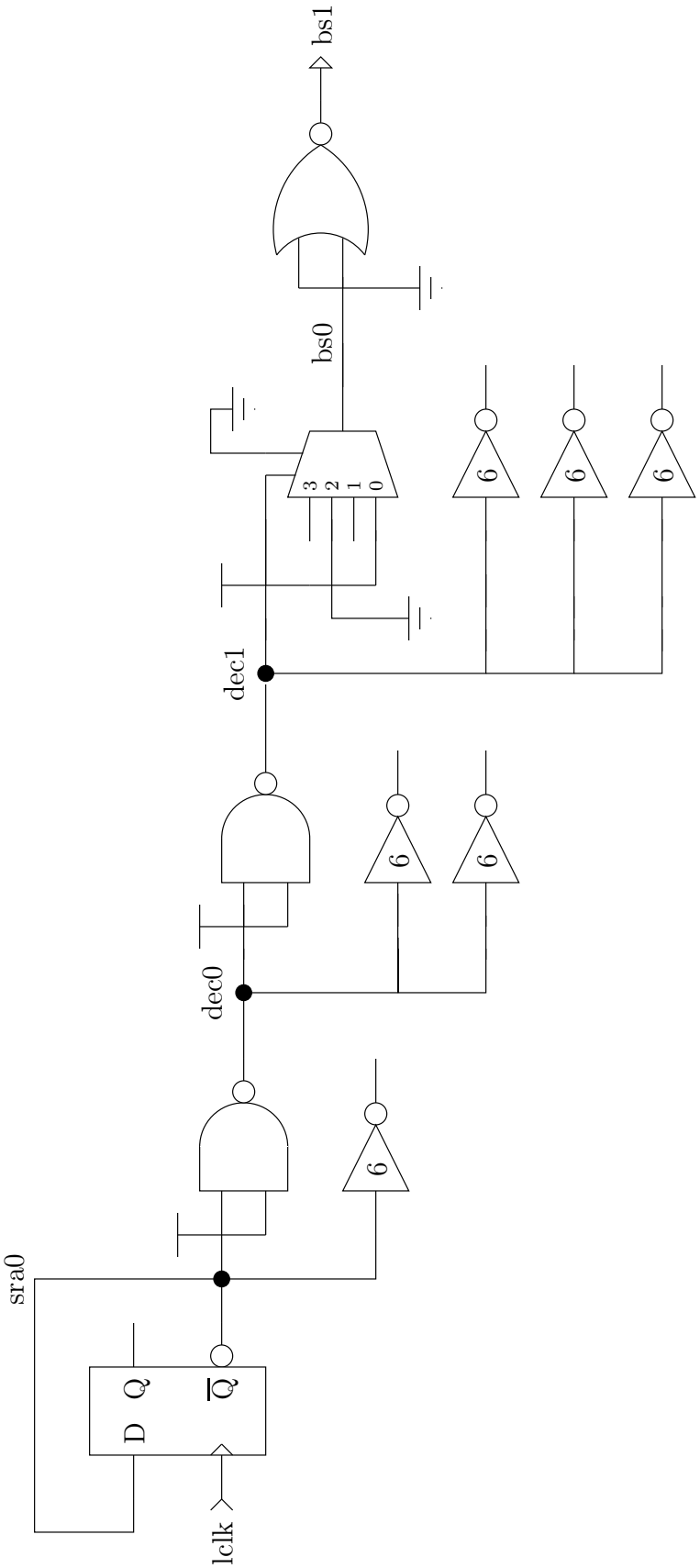


Fig. 3.11. The clock delay element used to match the multiplier critical path.

element is routed through the tuner and transparent latch to the *One-shot* so every transition on *bs1* is converted to a clock pulse.

The same gates as the critical path are used in the delay element except when matching fanout loads. Rather, when matching fanout loads, large inverters are used. These inverters are *inv6* gates, meaning they are six times the size of a normal inverter with six times the input capacitance. Another difference between the clock delay and the critical path exists on the *dec1* node. In the critical path, *dec1* is a long wire and therefore adds a large capacitive load. This load is calculated and matched in the delay element using inverters. Other than this long wire, routing capacitive load is neglected. Table 3.1 shows the comparison of loads between the critical path and the delay element. As can be seen in this table, the delay element is always slightly conservative when matching the load. The *bs1* load is purposely higher in the delay element to account for the setup time on the down-stream DFF. With careful matching, the required amount of tuning is minimized.

3.5 Stoppable Clock Control

The stoppable clock interface is again presented for reference in Fig. 3.12. The *Stoppable Clock* of this figure is described in the previous sections. The *Clock Control* and *State Machine* blocks are described in this section. A timing diagram for the interface is shown in Fig. 3.13. First, *gclk* causes the clock control to raise

TABLE 3.1
Critical path and delay element load comparison in fF

Node	Multiplier Critical Path	Clock Delay Element
sra0	299.73	333.72
dec0	482.04	485.13
dec1	618.00*	645.81
bs0	30.90	30.9
bs1	30.90	89.61

* 540.75 from gates and 77.25 from the long wire.

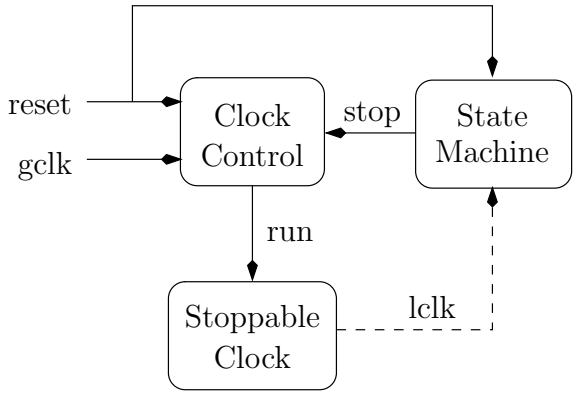


Fig. 3.12. Stoppable clock interface architecture.

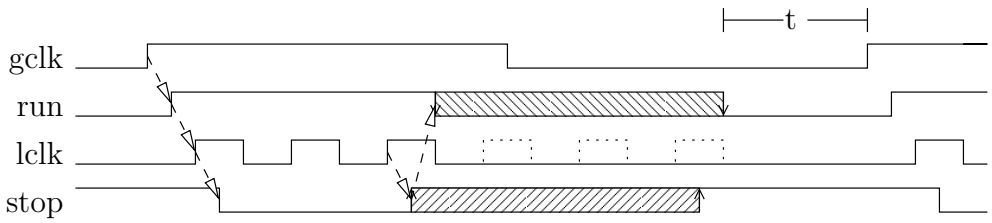


Fig. 3.13. Timing diagram for the stoppable clock interface.

run which in turn causes the local clock to begin oscillating. The first local clock pulse causes the state machine to move from its idle state; thus, *stop* is lowered. After a specified number of clock cycles, the state machine enters the idle state again; thus, *stop* is raised. A rising edge on *stop* causes the clock control to lower *run*. Depending on the global clock frequency, the number of local clock pulses, and the local clock frequency, *run* could fall before, during, or after the high-to-low transition on *gclk*. Thus, the circuit causing *run* to fall must be independent of the level of *gclk* and must ignore a falling edge on *gclk*. As stated previously, it is assumed that the clock is done oscillating some time (i.e., t) before the next rising edge of *gclk*.

The state machine is a Moore style state machine. Yet, *stop* must be hazard free and monotonic because it affects hazard sensitive circuits. To make *stop* hazard free, the state codes are Gray coded. Thus, every state change flips a single bit.

A single state-decoding *Nor* gate is used to generate *stop*. The state encoding is shown in Table 3.2. In the idle state, *stop* is high. Every other state always has a 1 in it, so the *stop Nor* gate is held low. The transition of the final iterating state to the idle state changes two state bits from 1 to 0, and the *Nor* gate changes to a 1 again.

The *run* signal is trickier to generate than the *stop* signal. A rising edge on *gclk* causes *run* to rise. The *gclk* signal is not synchronized with *lclk* and can rise or fall at any time. A rising edge on *stop* causes *run* to fall. The *stop* signal is synchronized with *lclk*. To generate the clock control circuit, `ATACS`, a tool for the synthesis and verification of timed circuits, is used [1]. `ATACS` takes a high level description and generates a circuit. The high level description for the clock control circuit is shown in Fig. 3.14.

The "process gclk_environment;" and "process stop_environment;" sections define the environment while the "process run;" section defines the behavior of *run*. It is necessary to insert an internal state variable (x) so that *run* can be generated. If it is assumed that *stop* will always rise after *gclk* is low, the state variable is

TABLE 3.2
Moore State Machine Encoding

State	Encoding	*stop*
Idle	0000	1
Initialize	0001	0
Iterating	0011	0
	0111	0
	0101	0
	0100	0
	0110	0
	1110	0
	1010	0
Iterating	1000	0
Final	1100	0
Idle	0000	1
⋮	⋮	⋮

```
module run;

process run;
      *  [  [gclk];  run+;  [~ stop];  x−;  [stop];  run−;  [~ gclk];  x+  ]
endprocess

process gclk_environment;
      *  [  gclk+;  gclk−  ]
endprocess

process stop_environment;
      *  [  [run  &  gclk];  stop−;  stop+;  [~ gclk]  ]
endprocess

endmodule
```

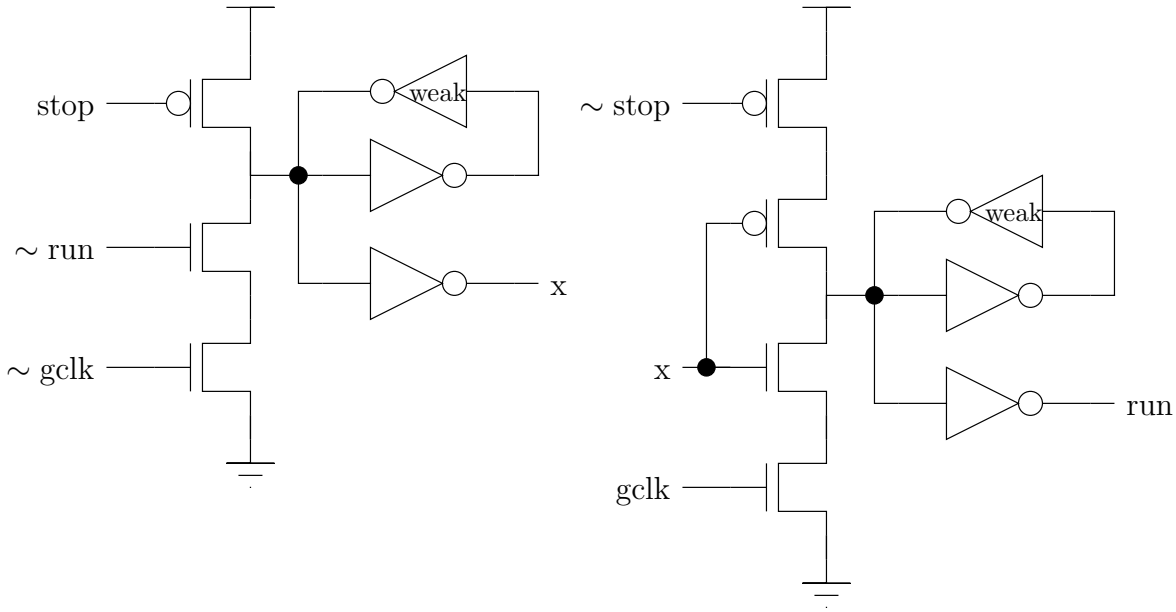
Fig. 3.14. High level description of the clock control for use in ATACS.

not needed. This involves adding timing assumptions to the design. These timing assumptions are used by ATACS to make circuit optimizations. Given the high level description, ATACS generates a production rule set (PRS). The PRS is implemented using generalized C-elements (gC). It is shown that the *run* circuit is on the critical path for cutting off the clock. Thus, a gC implementation is desirable because it is faster than a standard C-element implementation. The PRS generated by ATACS is in Fig. 3.15(a), and the gC implementation is in Fig. 3.15(b). The gC implementation is modified on the fabricated design to add *reset* and to push bubbles. The final circuit, shown in Fig. 3.16, generates $\sim$ run and $\sim$ x.

A major timing assumption of this stoppable clock is embedded in the cutoff path. It is illustrated in Fig. 3.17. A formal definition of the timing assumption is in Fig. 3.17(a). The order of timed events is shown in Fig. 3.17(b). First, a *lclk* pulse is generated. At this point it starts traveling through the delay element. Second, the state machine transitions to the idle state, and *stop* rises. Third, in the clock control, the *run* signal rises. Fourth, the transparent latch is closed. Events one through four must occur before *cpd* changes (i.e., event five). If *run* changes at the exact time *cpd* changes, a runt pulse may appear after the latch which may or may not fire the *One-shot*. If *run* changes too late, an extra *lclk* pulse is generated, sending the multiplier control into the wrong state.

+x : (∼ gclk & ∼ run)
−x : (∼ stop)
+run : (gclk & x)
−run : (stop & ∼ x)

(a)



(b)

Fig. 3.15. Clock control circuit. (a) Production rule set for the circuit. (b) gC implementation of the production rule set.

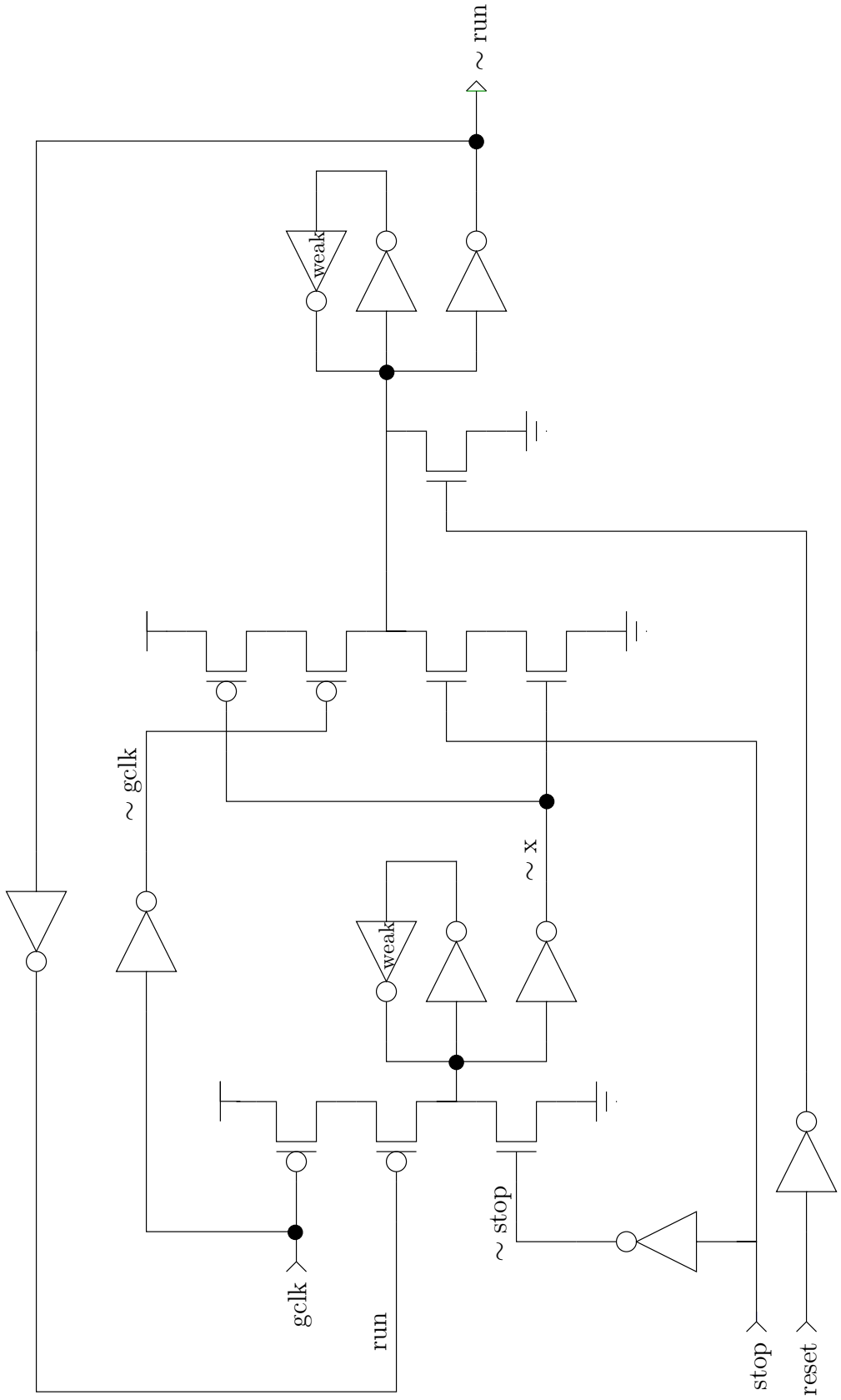
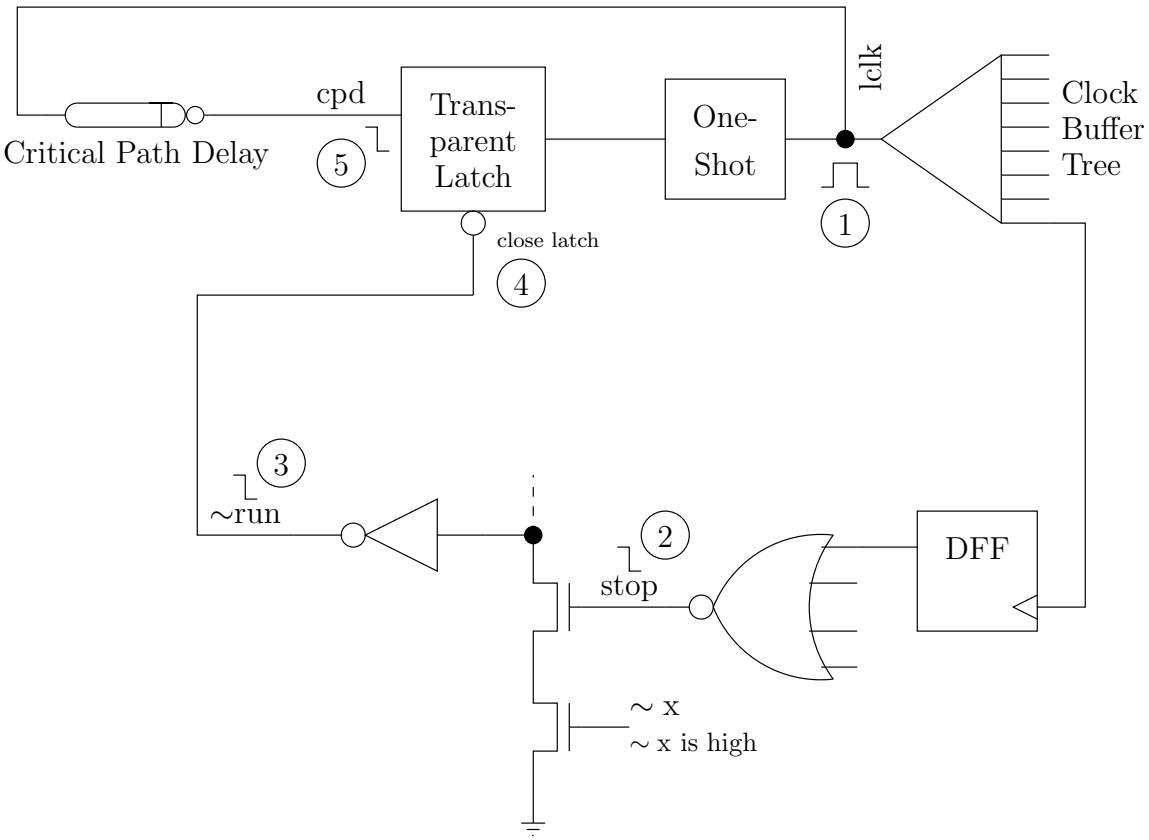


Fig. 3.16. Bubble shuffled clock control with reset added.

D_t = Clock buffer tree latency
D_s = State-machine latency to raise stop
D_r = gC latency to raise run
D_c = Cutoff latch latency
D_e = Critical path delay latency
Assumption : $D_t + D_s + D_r + D_c < D_e$

(a)



(b)

Fig. 3.17. Local clock cutoff timing assumption. (a) Formal timing assumption. (b) Events on the circuit.

It is interesting to note that the clock control exhibits 4-phase asynchronous behavior. In the high level description, *gclk* is changed to *req* and x is changed to *ack*. These changes are shown in Fig. 3.18. The *req* and *ack* 4-phase handshake is displayed in large bold font. Note that *ack* (or x) is in negative phase. This asynchronous behavior shows that x acts as a done signal; therefore, the multiplier can be used in an asynchronous environment as well as a synchronous environment. Taking a close look at Fig. 3.18 also indicates another timing constraint on subsequent multiplies: before *gclk* (or *req*) can rise to start a new multiply, x (or *ack*) must rise signaling the end of a multiply. In a synchronous environment, the x signal is ignored, and it is assumed that the multiplier is complete before the next global clock rising edge. If the timing constraints in this section are met, the clock control circuit works correctly. Simulations show that the designed circuits meet these timing constraints.

3.6 Simulation of the Stoppable Clock

To show that the stoppable clock interface circuits meet the cutoff timing constraint, SPICE simulation is used. Initially, SPICE simulation is done at the schematic level. After the design is completed to layout, SPICE simulation is performed on the parasitic extracted layout level. This level is much more accurate and helps ensure that the circuits indeed meet the timing constraint. The local clock interface circuit is simulated on three design corners (fast, typical, slow), and three voltage levels (5, 3.3, 1.5). This makes a total of nine different simulations. For the fast and typical corners, a temperature of 27° C is used. For the slow corner, a temperature of 60° C is used. In the simulations, the local clock tuning is set at minimum delay. Thus, the clock frequency is as close to the multiplier

```
process run;
      * [ [req]; run+; [~ stop]; ack-; [stop]; run−; [~ req]; ack+ ]
endprocess
```

Fig. 3.18. Asynchronous behavior of the control circuit.

critical path delay as it can be. Simulation of the parasitic extracted local clock interface at 5 V on the typical corner is shown in Fig. 3.19. As this figure shows, a rising edge on *gclk* causes $\sim run$ to fall. When $\sim run$ falls, x rises and the local clock begins oscillating. The first local clock pulse causes *stop* to fall. The clock oscillates for 11 pulses, and then *stop* rises. This causes $\sim run$ to rise which cuts off the clock and causes x to fall. Then the procedure repeats with the next *gclk* rising edge.

The SPICE simulations for the schematic level show that the clock works correctly for all nine situations with the minimal tuning setting. Simulations of the parasitic extracted layout show that the clock works correctly for all three corners at 5 and 3.3 V. The fast and typical corners work down to 1.6 V and the slow corner down to 1.7 V. With a voltage lower than that, the cutoff path is too slow, and the

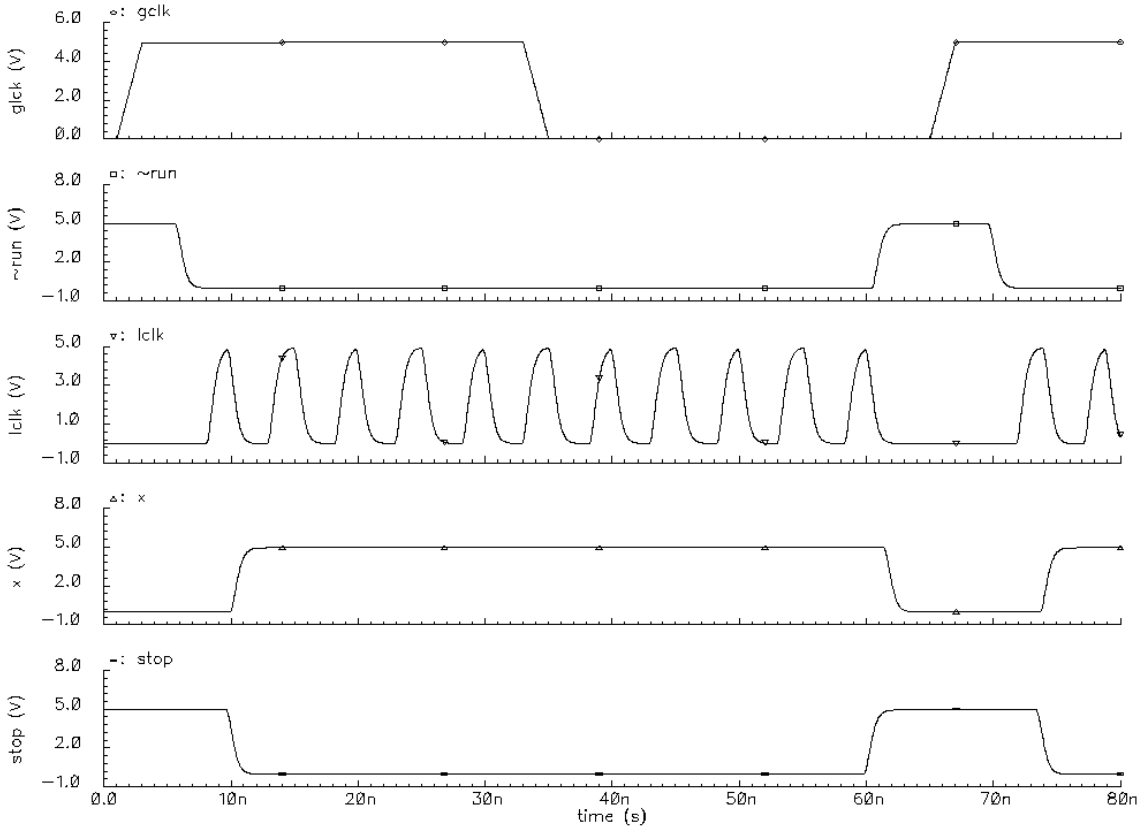


Fig. 3.19. SPICE simulation of clock interface at 5 V VDD on the typical corner.

local clock generates too many pulses. If a lower voltage is needed, the interface circuits must be redesigned and optimized such that the cutoff path is faster.

The additional clock tuning only serves to slow down the local clock; therefore, it cannot cause the clock to fail the cutoff timing constraint. Instead, it makes the clock more conservative and causes the cutoff timing assumption to be met with greater ease. Yet, tuning the clock to be slower causes the multiplier to be slower. This affects the second timing constraint mentioned in Section 3.5. It is stated that before *gclk* can rise, x must rise. After the bubble reshuffle, the phase of x is changed and the assumption becomes: before *gclk* can rise, x must fall. The longer a multiply takes, the closer the x falling edge occurs to the *gclk* rising edge. As shown in Fig. 3.20, when *gclk* rises before x falls, the multiplier ignores the

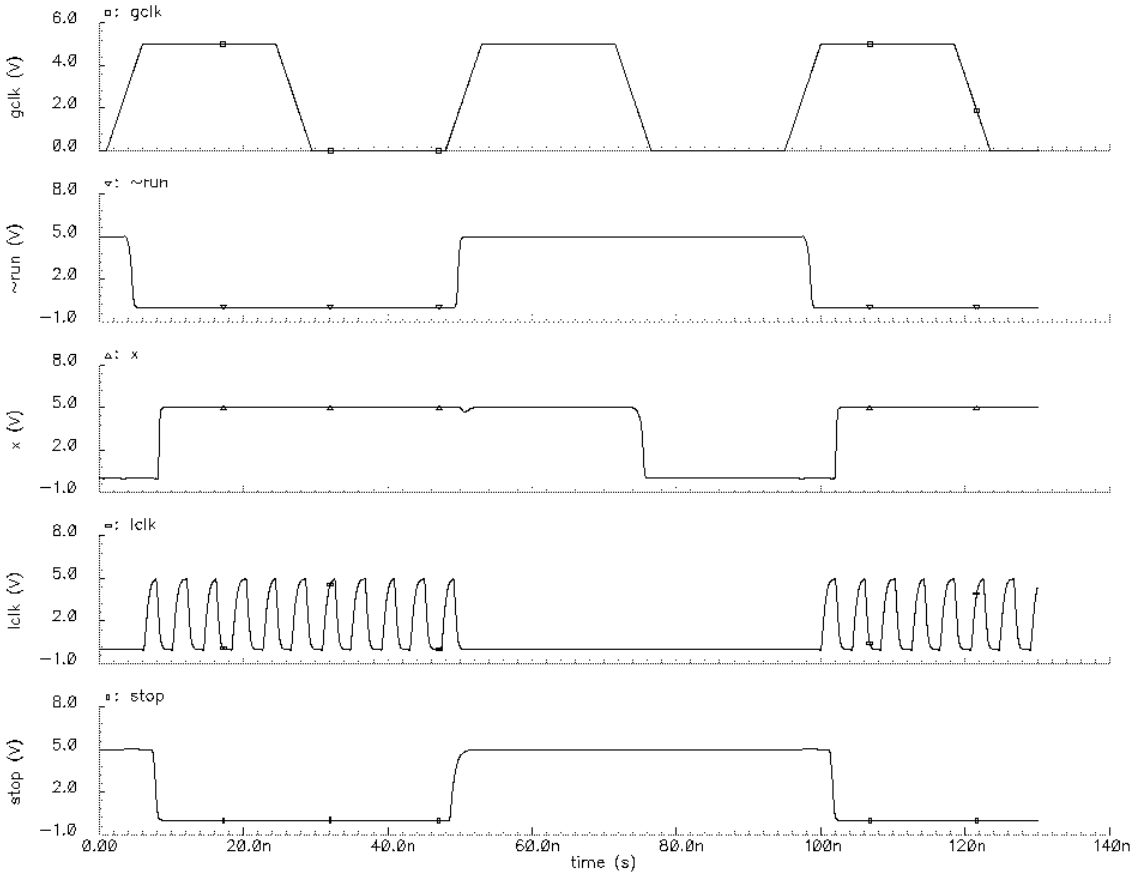


Fig. 3.20. SPICE simulation showing *gclk* rising before x falls.

gclk rising edge and remains idle for one *gclk* cycle. Note that x starts to fall at 49ns, but then *gclk* rises before x falls causing x to remain high. The multiplier works correctly on the subsequent rising edge of *gclk*; nevertheless it is still an error because one multiply is not performed.

The SPICE simulations described in this section only serve to evaluate the functionality of the local clock interface. More detailed SPICE simulations of the multiplier and local clock are provided in Section 4.2.

CHAPTER 4

EXPERIMENTAL RESULTS

In order to quantify the pros and cons of using stoppable clocks, a locally-clocked multiplier is fabricated. This chapter discusses experimental results from the fabricated IC. The proposed experiments are described followed by SPICE simulation and measured results. A detailed analysis of stoppable clocks is presented last.

4.1 Proposed Experiments

This section proposes experiments for evaluating the IC. Multiplier performance and stoppable clock performance are analyzed. The two are intertwined such that they cannot be evaluated separately.

To test multiplier functionality, single multiplies are fed in and the results are checked. A more extensive test as described in Chapter 2 uses the pseudo-random number generator and the signature file. The A operand is set to x65BC8 which decodes the sequence $0, -2B, B, -B, 0, -B, 2B, B, -2B, 2B$. The B operand comes from the pseudo-random number generator. After $2^{20}-1$ multiplies the signature is checked. If the multiplier is not working correctly, the global and local clock can be single stepped while viewing state machine bits on LEDs. Multiplier functionality is measured at different voltages and temperatures. If the multiplier generates wrong signatures, that point is noted as an error point that requires tuning.

Power consumption is calculated by measuring current draw on VDD. The A operand is set to x74CEE, and the B operand is pseudo-random. The decode sequence of A is $-2B, 0, -B, 0, B, -B, B, B, -B, 2B$. This sequence decodes a different multiple of B on every cycle except one. Thus, the *Shift Register A*, *Decode A*, and *Selector B* have close to worst-case switching frequency. Average current is measured on the multiplier VDD and clock VDD lines. Then, the power

equation $P = VI$ is applied. Power of an idle multiplier is also measured. Power consumption for different voltages and temperatures are measured.

Latency numbers are measured on an oscilloscope by viewing *gclk*, the internal variable x, and the local clock *lclk*. Remember that when x falls, the multiply is done. The delay from a *gclk* rising edge to x falling is the multiplier latency. The delay from a *gclk* rising edge to the first local clock pulse is the startup time. The delay from the last local clock edge to x falling is the stop time. Startup and stop times are control latency overhead. Latencies for different voltages and temperatures are measured. The frequency of the local clock is measured by putting the clock in free-running mode and viewing it on the scope. Local clock frequencies for different voltages are measured.

Clock behavior is evaluated by viewing the clock while multiplying and in free run mode. The clock is left in free run mode for a long time to see if it ever stops oscillating. It is necessary to determine if the clock exhibits behavior that would inhibit using the clock in a design. Clock behavior is analyzed at different voltages.

The critical path is matched in the local clock delay. If mismatch makes the local clock faster than the multiplier critical path, the multiplier generates incorrect values. In this case, the clock must be slowed down with tuning. Whenever a voltage and temperature cause the multiplier to give incorrect results, the clock is slowed down through tuning until the multiplier works. How much tuning is necessary is analyzed and discussed. Ideally, the multiplier critical path delay is measured under a variety of voltages and temperatures. This helps determine how the local clock delay scales compared to the critical path delay and how much tuning is needed. Unfortunately, the critical path signal does not appear clearly on an oscilloscope except at very low voltages. At high voltages, the local clock frequency exceeds the oscilloscope bandwidth. Thus, the only way to tell if the delay tracks the critical path is if the multiplier multiplies correctly.

A second clock is placed on the chip to measure process variation. This clock is identical to the original local clock that synchronizes the multiplier. The multiplier clock is in the upper left-hand corner of the chip, while the second clock is in the

lower right-hand corner. It is hypothesized that process variation from one side of the chip to the other may be insignificant. Thus, the clock can use the same gates that exist on the critical path, and process variation does not cause enough mismatch to cause multiply errors. Both clocks are routed to pins and the local clock frequencies are measured on an oscilloscope over a wide voltage range.

A third clock, shown in Fig. 4.1, is also designed and placed on the chip. This clock uses inverters for a delay element. The delay element is tuned to match $\frac{1}{2}$ the critical path delay. Thus, it takes two passes through the delay element to equal one clock pulse. The clock has a regular frequency every pulse. Rise and fall time differences in the inverters only serve to change the duty cycle of the third clock. This clock is used for comparison to the clock that synchronizes the multiplier. The alternate clock is placed on the chip and its output is routed to a pin for observation. It is hypothesized that this clock requires extreme amounts of tuning to match the critical path delay. Ideally, this clock delay is compared to the measured critical path delay. This shows how the multiplier critical path scales compared to an inverter delay. As stated before, the critical path signal is only measurable at low voltages. Thus, scale comparisons must be drawn from another source. These comparisons come from comparing the frequency of the third clock to the frequency of the other two clocks.

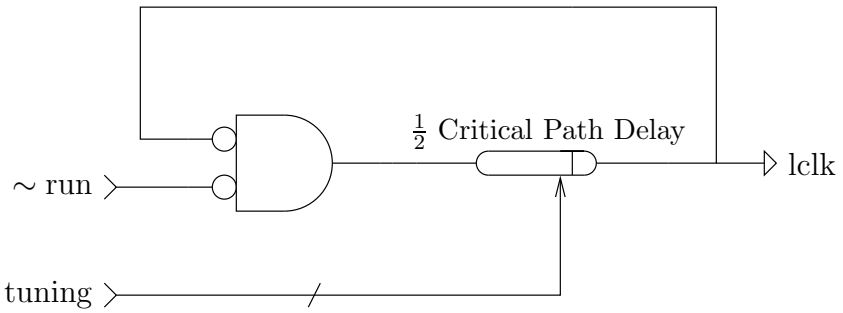


Fig. 4.1. Alternative stoppable clock design with delay of inverters.

4.2 SPICE Simulation Results

Latency and power numbers are initially calculated through SPICE simulation. After the design is completed to the layout level, a parasitic extraction view is generated. This view has parasitic capacitance, but not resistance because the extraction process available extracts only capacitance. This view is used to perform SPICE simulation on the multiplier design and local clock. The extracted view behaves closest to the actual chip than any other view.

The extracted view is simulated on the three design corners (fast, typical, slow), and at three voltage levels (5, 3.3, 1.6/1.7). As mentioned in Chapter 3, the fast and typical corners do not work below 1.6 V. And the slow corner does not work below 1.7 V. This is due to the cutoff path being too slow below these voltage levels. For the fast and typical corners, a temperature of 27° C is used. For the slow corner, a temperature of 50° C is used. In the simulations, the local clock tuning is set at minimum delay. Thus, the clock frequency is as close to the multiplier critical path delay as it can be.

For clock 1 and clock 2, the time between every other pulse is shorter. An exaggerated example of this is found in Fig. 4.2. The reason for this behavior is because of the difference in rise and fall times in the local clock delay. One pass through the delay is faster than the next. Then the two different delays are repeated. Local clock period is calculated from the average of *t1* and *t2* in Fig. 4.2. Local clock frequency is calculated from the inverse of the local clock period. The duty cycle of clocks 1 and 2 is determined by the pulse width delay in the *One-shot*. If the pulse width delay is equal to half the critical path delay, clocks 1 and 2 have a 50% duty cycle. This is not likely to happen due to the different clock periods every other cycle. SPICE latency numbers for the local clock are shown in Table 4.1. The width of local clock pulses is shown in the last column.

SPICE latency numbers for the multiplier are shown in Table 4.2. In this table, the multiplier frequency is calculated as the inverse of the multiplier latency. The *lclk* start and *lclk* stop columns show the control overhead for starting and stopping the clock. The control overhead column shows the percentage of the multiplier

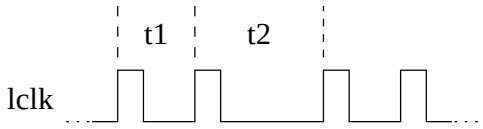


Fig. 4.2. Clock 1 output.

TABLE 4.1
Local clock latency numbers from SPICE

Corner	Temp [°C]	VDD [V]	lclk period [nS]	lclk frequency [MHz]	lclk width [nS]
fast	27	5	3.58	279.14	0.51
typ	27	5	5.01	199.45	0.83
slow	60	5	7.40	135.19	1.32
fast	27	3.3	4.75	210.68	0.63
typ	27	3.3	7.15	139.78	1.12
slow	60	3.3	11.34	88.15	1.98
fast	27	1.6	13.19	75.82	1.45
typ	27	1.6	26.77	37.36	3.37
slow	60	1.7	43.37	23.06	6.51

TABLE 4.2
Multiplier latency numbers from SPICE

Corner	Temp [°C]	VDD [V]	multiplier latency [nS]	multiplier frequency [MHz]	lclk start [nS]	lclk stop [nS]	control overhead [%]
fast	27	5	42.19	23.70	4.20	2.17	15.10
typ	27	5	59.80	16.72	6.60	3.06	16.15
slow	60	5	88.18	11.34	9.68	4.53	16.11
fast	27	3.3	56.36	17.74	6.19	2.70	15.77
typ	27	3.3	85.13	11.75	9.39	4.20	15.96
slow	60	3.3	135.07	7.40	14.81	6.81	16.00
fast	27	1.6	156.25	6.40	17.12	7.24	15.59
typ	27	1.6	321.04	3.11	36.22	17.16	16.63
slow	60	1.7	527.09	1.89	61.02	32.36	17.71

latency in starting and stopping the clock. On the typical corner, the average percentage of control overhead is 16.25%.

SPICE power numbers for the multiplier and local clock are shown in Table 4.3. Both the A and B operands are set such that they generate maximum switching frequency in the multiplier. Also, the clock is tuned to the fastest setting. Thus, worst case power is calculated. The current draw on the multiplier and local clock VDD lines is measured and averaged over the time when the multiplier is active. The power equation $P = VI$ is then applied to get the total power.

4.3 Measured Results

The 20-bit multiplier was fabricated through MOSIS using AMI's 0.5 μm process. The die size is 3 mm $\times$ 3 mm and has 84 pins. A die photo is shown in Fig. 4.3. The entire design is standard-cell except for two gC gates in the clock control. It is arguable that those gC gates are standard-cell in an asynchronous gate library. In the cells, the transistor widths are 10.2 μm for the PMOS and 7.8 μm for the NMOS. Transistor lengths are always the minimum 0.6 μm except in the weak feedback inverter for the gC gates. The weak feedback inverters have a length of 4.8 μm. The core of the chip includes the multiplier, a local clock to synchronize the

TABLE 4.3
Multiplier and local clock power numbers from SPICE

Corner	Temp [°C]	VDD [V]	mult current [mA]	lclk current [mA]	total current [mA]	total power [mW]
fast	27	5	87.05	22.57	109.60	548.10
typ	27	5	61.36	16.32	77.68	388.40
slow	60	5	42.70	11.53	54.23	271.15
fast	27	3.3	37.45	10.14	47.59	157.05
typ	27	3.3	24.91	6.895	31.81	104.96
slow	60	3.3	16.40	4.575	20.98	69.22
fast	27	1.6	5.437	1.527	6.964	11.14
typ	27	1.6	2.807	0.779	3.586	5.738
slow	60	1.7	1.969	0.544	2.513	4.272

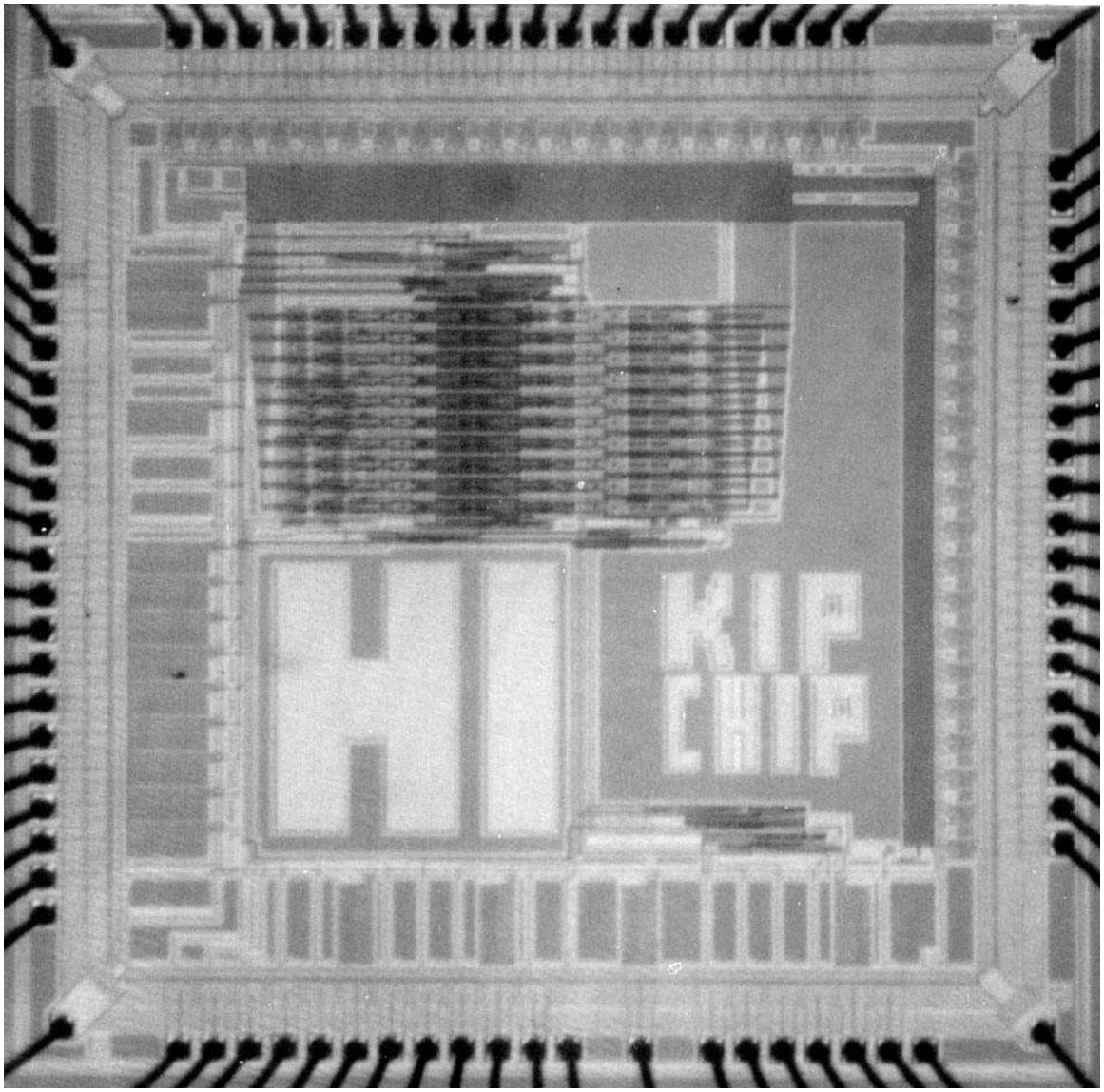


Fig. 4.3. Die photo.

multiplier, and routing for multiplier control and the local clock. The core size is 819 μm $\times$ 572 μm = 0.468 mm^2 and contains 8190 transistors. This section presents measured latency and power values for the multiplier and stoppable clocks. The local clocks are numbered 1 through 3. Local clock 1 is the clock that synchronizes the multiplier. Local clock 2 is identical to local clock 1 and is placed on the opposite side of the chip. Local clock 3 is the alternate clock with a delay made of inverters. Tuning for the local clocks is labeled 0 through 3. Tune 0 is the fastest setting and Tune 3 is the slowest. When performing high temperature testing, the test equipment consists of a Tektronix TDS 224 oscilloscope, Agilent E3615A power supply, Sigma Systems Model C4 temperature chamber, and Fluke 189 multimeter. The Tektronix oscilloscope has a bandwidth of 100 MHz with 1 GSps. For all other tests, the test equipment consists of an HP 54645D mixed-signal oscilloscope, HP E3631A power supply, and HP 34401A multimeter. The HP oscilloscope has a bandwidth of 100 MHz with 200 MSps.

Frequency numbers for the local clocks at different voltage levels are shown in Table 4.4. These values come from measurements on chip number three at room temperature. The clocks are set in free-run mode and the frequency is measured on the oscilloscope. As stated before, rise and fall times for gates in the first two local clocks differ enough that every other pulse is shorter. The frequency reported is calculated from the average of the two periods. Comparing the measured frequencies in Table 4.4 to the SPICE frequencies in Table 4.1, it is noticed that clock 1 runs slightly above the typical corner, and clock 2 runs right on the typical corner. It is interesting to note that although clocks 1 and 2 are identical, the measured frequencies are different for all voltages except 0.5 V. On average, clock 2 is 16.9% slower than clock 1. This is consistent across all 5 chips. The clocks are on different VDD lines; however, both lines have the same voltage level while performing these experiments. Explanations and hypotheses for the mismatch are discussed in Section 4.4.

To see how tuning changes the frequencies, the local clock frequencies are measured at 5V with all possible tuning settings. All five chips are measured and

TABLE 4.4
Measured local clock frequencies

VDD [V]	clock 1 [MHz]	clock 2 [MHz]	clock 3 [MHz]
5.0	230.7	195.0	290.5
4.5	219.4	177.3	276.0
4.0	195.4	164.6	256.3
3.5	176.4	154.5	235.7
3.0	152.4	135.8	204.5
2.5	128.0	102.8	163.8
2.0	86.5	74.3	116.3
1.5	43.3	38.0	60.8
1.0	5.25	4.75	7.65
0.5	171 Hz	173 Hz	245 Hz

the average frequency for each tuning setting is calculated. Table 4.5 shows the average frequency for each tuning setting.

It is found that numerous voltage levels cause the local clock to fail. Most of the failures are due to failing the cutoff timing constraint. At these voltage levels, the cutoff path is too slow, so the local clock generates 12 or more pulses on a multiply before it is cut off. But once the clock has generated 12 or more pulses, the multiplier is in the wrong state and becomes unresponsive. In SPICE simulations of Chapter 3, the clock was set to Tune 0 (i.e., the fastest setting) and all corners worked down to at least 1.7 V. This is not the case on the actual chips. For Tune 0, the local clock cutoff path is always too slow, and therefore the local

TABLE 4.5
Measured local clock frequencies with tuning at 5 V

Clock	Tune 0 [MHz]	Tune 1 [MHz]	Tune 2 [MHz]	Tune 3 [MHz]
1	227	215	174	125
2	188	181	155	116
3	295	257	161	88

clock does not function. Tune 1 and Tune 2 work for some voltage ranges, and Tune 3 works from 5 V down to 1.14 V. Table 4.6 shows the voltage ranges for each tuning setting which do not cause the local clock to fail. The ranges are an average of the failure points for all five chips. The lowest voltage where the clock still functions is 1.14 V. At this point the cutoff path is still working correctly, but the gC gates in the local clock control quit working. Explanations and hypotheses for why the clocks fail are presented in Section 4.4.

Measured multiplier latency numbers at room temperature are shown in Table 4.7. Only certain voltage levels and tuning settings work without local clock failures. These levels and tuning settings are the ones tested and presented in Table 4.7. The latency numbers are measured by viewing *gclk*, x, and *lclk* on an oscilloscope. It is the same procedure as the SPICE calculations described in Section 4.2. The start and stop local clock delays are independent of the tuning setting of the local clock. Rather, they are determined by the voltage level. The control overhead at 1.2 V is significantly higher than the other voltage levels. This is because at 1.2 V, the gC gates are on the verge of failing and they become very slow. This is discussed further in Section 4.4. There are slight errors in the latency calculations due to pad delay. The global clock must be routed through a pad and pin to the internal circuit, and x must be routed off chip through a pad and pin. Both of these delays add to the latency calculation. The same situation exists for the local clock start delay. The stop latency has an error if routing x off the chip has a different delay than routing the local clock off the chip. However both of these signals come from a similar portion of the chip and have pins close together.

TABLE 4.6
Measured working points for the local clock

Tune 0 [V]	Tune 1 [V]	Tune 2 [V]	Tune 3 [V]
NA	5.00 - 3.84	5.00 - 3.63 3.06 - 2.46 1.64 - 1.14	5.00 - 1.14

TABLE 4.7
Measured multiplier latency numbers

Tuning	VDD [V]	multiplier latency [nS]	multiplier frequency [MHz]	lclk start [nS]	lclk stop [nS]	control overhead [%]
1	5.0	58.8	17.01	8.8	4.4	22.4
1	4.0	68.4	14.62	9.6	4.8	21.1
2	5.0	71.2	14.04	7.8	4.8	17.7
2	4.0	80.4	12.44	8.8	3.0	14.7
2	3.0	101.6	9.84	11.6	5.6	16.9
2	1.6	302.0	3.31	32.0	19.0	16.9
2	1.2	3260.0	0.306	1590.0	910.0	76.0
3	5.0	93.6	10.68	8.0	4.0	12.8
3	4.0	107.6	9.29	8.4	4.8	12.3
3	3.0	144.0	6.94	11.2	6.4	12.2
3	1.6	443.0	2.26	30.0	23.0	12.0
3	1.2	4220.0	0.237	1450.0	920.0	56.0

The multiplier functionality is tested in many ways. Initially, single multiplies are fed in, and correct results are returned. A much more extensive test as described in Chapter 2 is performed next. The A operand is set to x65BC8 which decodes the sequence $0, -2B, B, -B, 0, -B, 2B, B, -2B, 2B$. The B operand comes from the pseudo-random number generator. After $2^{20} - 1$ multiplies the signature is checked. The local clock is set on Tune 3 (i.e., the slowest setting) so that multiplier functionality is tested independent of clock issues. Again, the correct signature is generated for all voltages tested.

Testing multiplier functionality at faster speeds is also performed. While multiplier latency and power numbers are measured, a pass/fail multiplier functionality test is performed. For the pass/fail test, the A operand is set to x74CEE while the B operand comes from the pseudo-random number generator. The decode sequence of A is $-2B, 0, -B, 0, B, -B, B, B, -B, 2B$. This sequence decodes a different multiple of B on every cycle except one. Thus, the *Shift Register A*, *Decode A*, and *Selector B* have close to worst-case switching frequency. A signature calculated

with a C program is latched into a comparison register. Every $2^{20} - 1$ multiplies, the signature file is checked against the comparison register and a pass/fail latch is updated. Then the signature file is reset. Thus, a continuous pass/fail check is performed and the pass/fail latch is routed off chip for observation. It is found that Tune 1 generates an incorrect signature for all voltages measured. This implies that the local clock frequency with Tune 1 is too fast compared to the multiplier critical path. When the local clock is set on Tune 2 and Tune 3, the multiplier generates a correct signature for all voltage levels and temperatures tested. The local clock is designed such that Tune 0 matches the multiplier critical path conservatively. Tune 1 is even more conservative than Tune 0. Thus, it is surprising that Tune 1 is too fast compared to the critical path. Explanations and hypotheses for the mismatch are presented in Section 4.4.

Measured multiplier latency numbers at higher temperatures are presented in Table 4.8. The test board is placed in a temperature chamber and the temperature is set to 50° C. This temperature is chosen based on temperature restrictions on the 10x probes. The *gclk*, *x*, and *lclk* signals are again viewed on an oscilloscope. The multiplier only generates correct results for Tune 2 and the slower Tune 3. Thus, high temperature latency numbers are generated only for Tune 2. At 50° C, the multiplier works down to 1.06 V. The high temperature latency numbers are slower than room temperature latency numbers except at extremely low voltages. At 1.12 V, the high temperature latency numbers are actually faster than at room

TABLE 4.8
Measured multiplier latency numbers at 50° C on Tune 2

VDD [V]	multiplier latency [nS]	multiplier frequency [MHz]	lclk start [nS]	lclk stop [nS]	control overhead [%]
5.00	74	13.51	7	4	14.8
3.00	106	9.43	10	6	15.1
1.60	312	3.20	28	24	16.6
1.12	1250	0.800	150	130	22.4
1.06	3540	0.282	1320	660	55.9

temperature. This is because the gCs are not on the verge of failing at 1.12 V; thus, start and stop times at 1.12 V are not as drastic.

Power consumption numbers are calculated by measuring the current draw on the multiplier and local clock VDD lines while continuously multiplying. The A operand is set to x74CEE, and the B operand is pseudo-random. Thus, the pass/fail test is performed while measuring power numbers. The A operand decodes a different multiple of B on nearly every cycle, so the *Shift Register A*, *Decode A*, and *Selector B* have close to worst-case switching frequency. Average current is measured on the multiplier VDD and clock VDD lines. Then, the power equation $P = VI$ is applied. Power consumption is affected by the frequency of *gclk*. When measuring power, the frequency of *gclk* is set conservatively such that the local clock does not fail and the multiplier generates a correct signature. The measured power numbers for both room temperature and 50° C are found in Table 4.9. Even though the multiplier is running slower at 50° C, it consumes nearly the same amount of power as at room temperature.

These power numbers are significantly smaller than the SPICE simulation power numbers of Table 4.3. This is due to three factors. First, the local clock is set to Tune 2, and therefore the multiplier runs slower and draws less current over

TABLE 4.9
Measured power numbers on Tune 2

Temp [°C]	VDD [V]	mult freq [MHz]	average current [mA]	average power [mW]
27	5.00	13.30	39.32	196.6
27	3.00	9.42	15.60	46.80
27	1.60	3.13	2.60	4.16
27	1.12	234K	0.14	0.16
50	5.00	11.49	37.80	189.0
50	3.00	7.37	13.78	41.34
50	1.60	3.06	2.22	3.55
50	1.12	374K	0.39	0.44
50	1.06	207K	0.18μ	0.19μ

time. Second, the B operand cycles through all possible values causing average-case switching in the multiplier rather than worst-case switching. Third, the SPICE current is only measured while the multiplier is active. There is some dead time between multiplies where the multiplier is idle. This dead time has very little current draw. The measured current is an average over time, which includes the dead time, and therefore the average is lower.

Power consumption for an idle multiplier is also measured. There are two ways to make the multiplier idle. One is to cut off *gclk* completely. In this case, no switching occurs in the multiplier, and power consumption is due solely to leakage current. The second way is to allow *gclk* to pulse, but to hold *reset* low. While *reset* is low, the $\sim run$ signal remains high and the local clock never oscillates. The global clock is routed to ≈ 60 latches through a small clock buffer tree. In the second situation, the only switching in the circuit is the *gclk* buffer tree, and the clock input to ≈ 60 latches. The power measured in the second situation is obviously dependent on the frequency of *gclk*. For each voltage level, the *gclk* frequency is set to the frequency of the multiplier at that voltage level. The idle circuit power measurements are found in Table 4.10.

TABLE 4.10
Measured power numbers when idle

VDD [V]	power no glck [μW]	power gclk & no lclk [mW]	gclk frequency [MHz]
5.0	29.0	10.10	14.0
4.5	27.5	7.83	13.5
4.0	26.8	5.68	12.4
3.5	3.5	3.92	11.2
3.0	0.0	2.49	9.8
2.5	0.0	1.33	7.8
2.0	0.0	0.48	4.0
1.5	0.0	0.18	3.3
1.0	0.0	0.008	0.31
0.5	0.0	0.0	0.11

The final node in the multiplier critical path is routed off chip for observation. It is meant to measure the time from a local clock pulse to a change on the final node of the critical path. Thus, the multiplier critical path delay is measured. Unfortunately, this signal is only observable on the oscilloscope at very low power supply voltages. At high voltages, the local clock frequency exceeds the oscilloscope bandwidth. On chip 4 at room temperature and 1.5 V, the critical path takes 13.6 ns to fall, and 20 ns to rise. This implies that the multiplier can be run at approximately 50 MHz at that voltage. At 1.2 V, the critical path takes 27 ns to fall and 47 ns to rise. Thus, the multiplier can be run at approximately 21 MHz at that voltage. Local clock 1 runs at 45 MHz on Tune 0 at 1.5 V. Thus, if the cutoff path worked with Tune 0 at 1.5 V, the multiplier would generate correct results.

4.4 Stoppable Clock Electrical Issues

This section discusses some of the stoppable clock electrical issues in more detail. For instance, a free-running local clock is tested to see if it ever stops oscillating. The clock is left in free run mode for an hour and observed on an oscilloscope. A plot of clock 1 after an hour of free-run mode at 5 V VDD is shown in Fig. 4.4. It does not look like a digital signal because the clock frequency is well above the oscilloscope bandwidth. After an hour the clock continues to oscillate without stopping.

Another issue in the design of stoppable clocks is the sizing of gC transistors. Sizing is found to be a critical aspect for performance of the clock control. The set of feedback inverters on a gC is called a keeper. The keeper must be strong enough to hold state when it is not being actively driven by an NMOS or PMOS stack. Nevertheless, it must be weak enough that an NMOS or PMOS stack can overdrive it. A stronger keeper is intolerant to noise and power bumps, but is harder to overdrive. A weaker keeper is easier to overdrive, but is more susceptible to noise and power bumps. Sizing of the keeper also affects how fast the gC output changes and the lowest VDD at which the gC functions. The weak feedback inverter is made by sizing the transistors longer than normal. Simulations are performed to

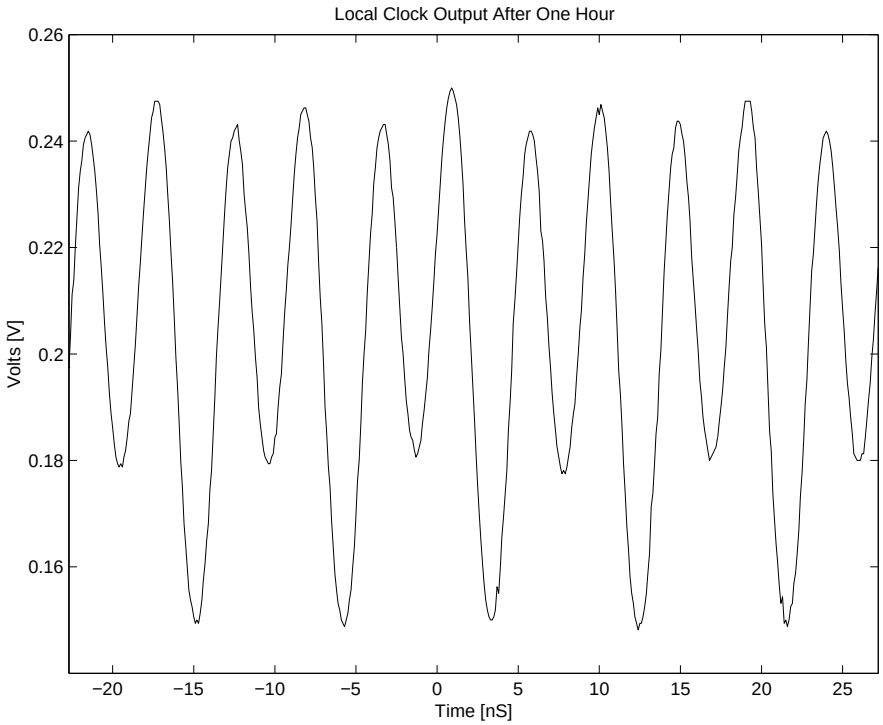


Fig. 4.4. Free running clock after one hour.

find how long to make the weak feedback inverter transistors. This length is found to be six to eight times as long as a minimum length transistor. The simulated gC is shown in Fig. 4.5. A plot showing gC keeper simulation with a VDD of 5 V is shown in Fig. 4.6. When the NMOS input rises, the output of the gC rises, and when the PMOS input falls, the output falls. With the keeper sizing used in the local clock design and an average load comparable to a fanout of two, the gC gate takes 0.31 ns to rise and 0.74 ns to fall at 5 V VDD. In the stoppable clock, when a gC is on the critical path, it is always one input to that gC that changes while the others are stable. Thus, simulations of gCs are simplified in that only one input is required to change at a time.

It is mentioned in Section 4.3 that the gC gates quit functioning at 1.14 V VDD. The reason for this is that there exists an error in the sizing of keepers on the gC gates. The keeper transistors are sized too wide and are therefore too strong at low voltages. Ideally, the NMOS and PMOS stacks have normal widths while the

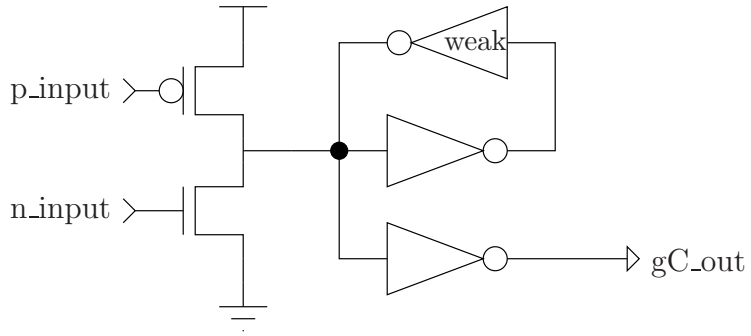


Fig. 4.5. A gC gate.

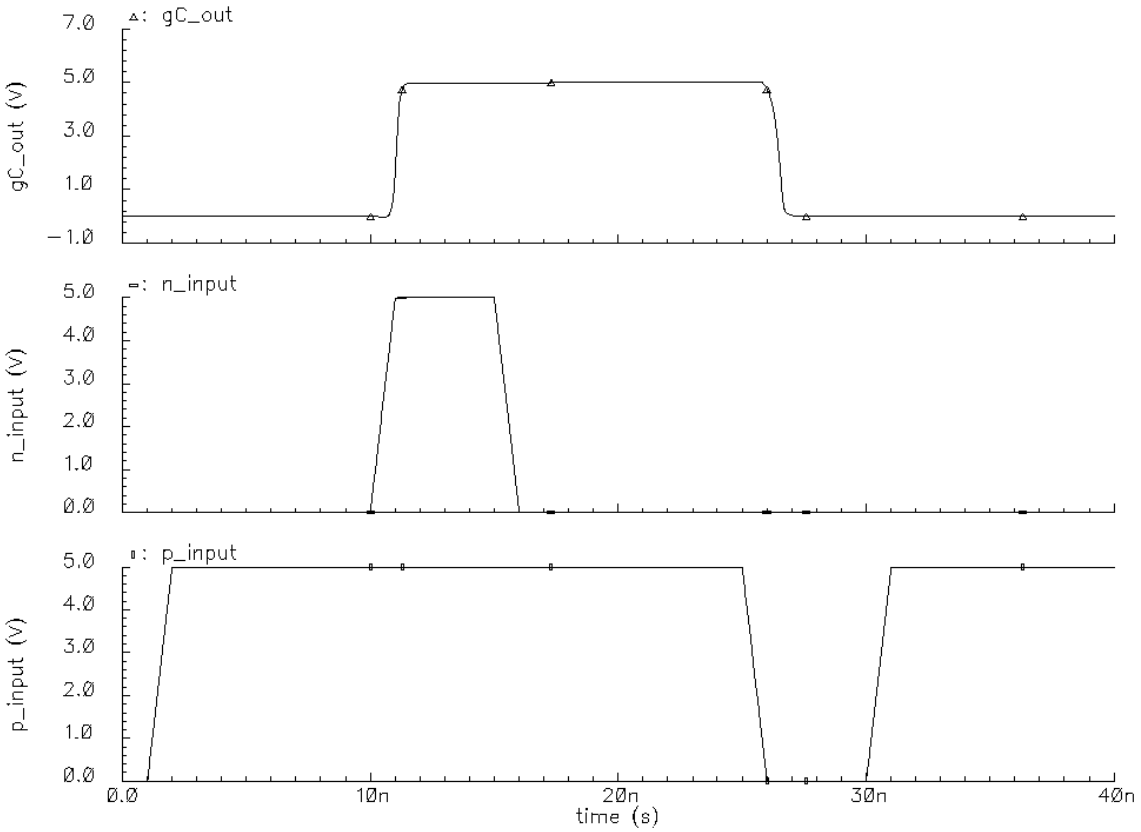


Fig. 4.6. Simulation of gC keeper at 5 V VDD.

keeper transistors are minimum size. When the keeper is designed this way, the gC functions correctly down to extremely low voltages. A slightly different gC design with correct transistor sizing is shown in Fig. 4.7. With the new keeper sizing and an average load comparable to a fanout of two, the gC gate takes 0.38 ns to rise and 0.16 ns to fall at 5 V VDD. The original keeper design uses three inverters to isolate the internal keeper node from the output load. This third inverter is a conservative measure and is not necessary if sizing of the driving NMOS and PMOS transistors is done correctly.

Simulations are performed on the circuit of Fig. 4.7 and the gC circuit that is currently used on the IC. Latency numbers for extremely low voltages are shown in Table 4.11. In these simulations, a single input is changed and the output latency is measured. The gC gates have an average load comparable to a fanout of two. In simulation, the original gC fails at 1.22 V VDD, and the new gC works below V_t (0.7 V). Rise-times for the gC gates become extremely slow as they approach the failure point. This is manifest in the control overhead of the multiplier at 1.12 V. At that voltage, control overhead is 76% on Tune 2. The rise-times of the gC gates can be shortened if the driving PMOS widths are increased while leaving the keeper widths as is. Significant improvement exists for doubling or tripling the width of the PMOS transistors. This becomes the classic tradeoff between area and latency. Yet, care must be taken that noise on the gate of the PMOS does not turn it on enough that the weak keeper is overdriven. The larger the PMOS, the more likely it has strength to overdrive the keeper with very little noise.

Another failure point in the local clock design exists in the cutoff path timing constraint. The timing constraint is again presented in Fig. 4.8 for reference. Section 4.3 mentions that for certain voltage levels and tuning settings, the cutoff path is too slow. When this occurs, the local clock generates too many pulses, and the multiplier locks up. Table 4.6 shows which tuning settings work at what voltages. It is interesting to note that the SPICE simulations in Section 4.2 claim that Tune 0 works down to at least 1.7 V. And the other tuning settings only serve to slow the clock down, making the cutoff timing constraint easier to meet. In the

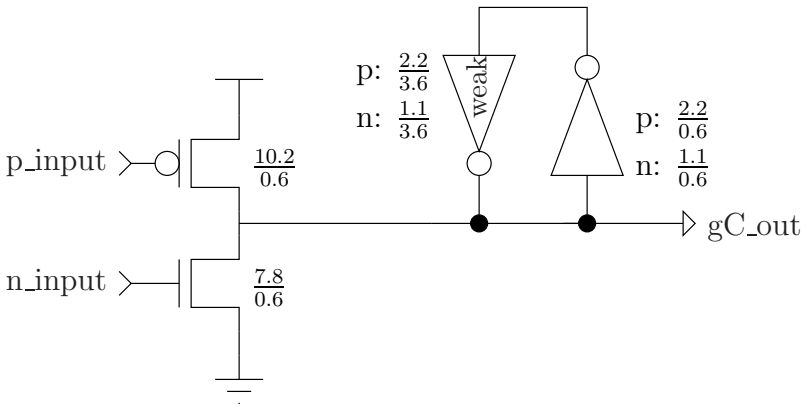


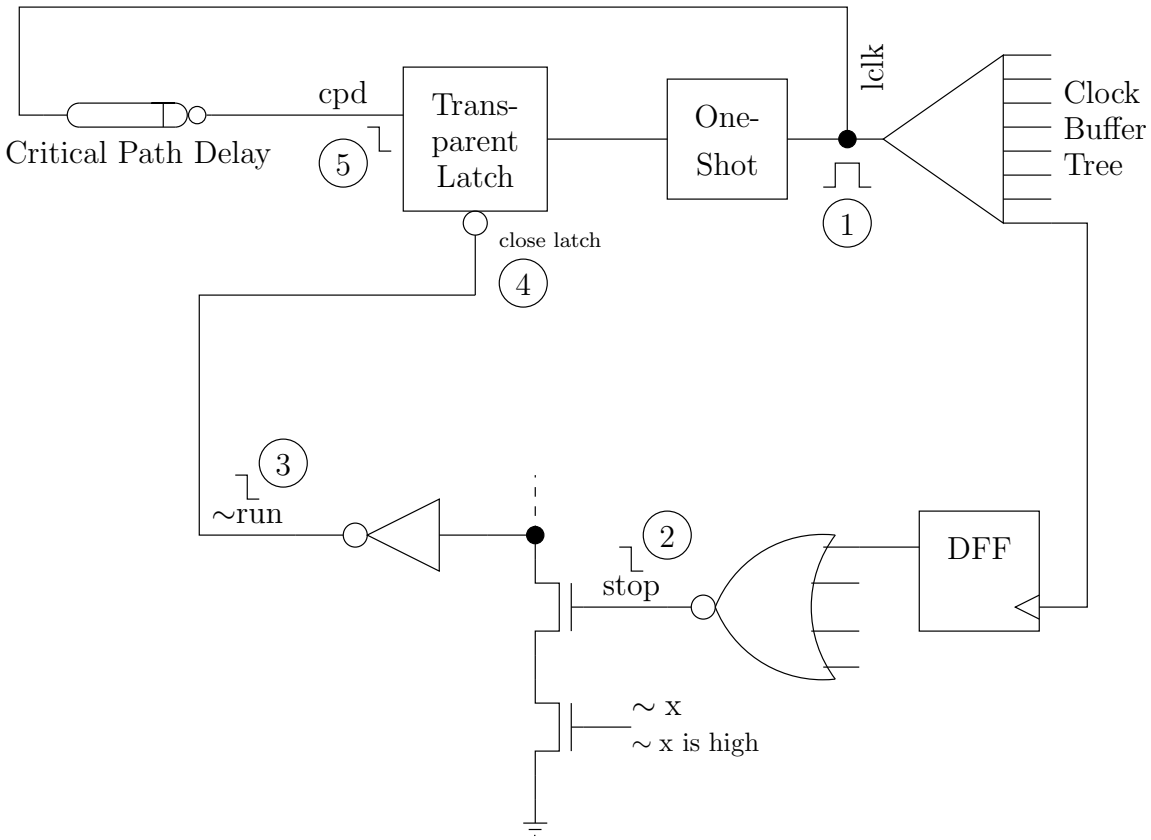
Fig. 4.7. A gC gate with better sizing.

TABLE 4.11
Improved sizing gC latency numbers from SPICE

VDD	Improved rise-time [nS]	Improved fall-time [nS]	Old fall-time [nS]	Old rise-time [nS]
1.25	7.95	0.81	106.90	5.20
1.22	9.78	0.88	579.10	5.85
1.20	11.22	0.94	NA	NA
1.15	15.41	1.12	NA	NA
1.10	23.13	1.34	NA	NA
1.05	37.34	1.64	NA	NA
1.00	66.21	2.08	NA	NA
0.95	136.30	2.80	NA	NA
0.90	316.80	3.99	NA	NA
0.85	814.50	6.32	NA	NA
0.80	2349.00	11.17	NA	NA
0.75	7639.00	22.12	NA	NA
0.70	27250.00	49.60	NA	NA
0.65	104900.00	120.45	NA	NA

D_t = Clock buffer tree latency
D_s = State-machine latency to raise stop
D_r = gC latency to raise run
D_c = Cutoff latch latency
D_e = Critical path delay latency
Assumption : $D_t + D_s + D_r + D_c < D_e$

(a)



(b)

Fig. 4.8. Local clock cutoff timing assumption. (a) Formal timing assumption. (b) Events on the circuit.

SPICE simulations, the cutoff constraint is being met with a very narrow margin. In addition, the SPICE simulations only have capacitive parasitics and not resistive parasitics. Plus, the model files used for simulation are not the extracted model files provided by MOSIS after fabrication. So some error in the simulation exists. Thus, in the actual circuit, the margin is too narrow and the constraint is broken sometimes. There are numerous ways to speed up the cutoff path so that it is faster than the local clock delay element by a very large margin. For example, the local clock signal that feeds the state-machine can be taken from further back in the clock buffer tree. Bubble pushing can be implemented such that the gates with the slowest rise times are required to fall in the cutoff path. The *Nor* gate can be optimized to a faster *Nand* gate, or can be folded into the $\sim run$ gC gate. And finally, the $\sim run$ gC keeper can be changed to the keeper as shown in Fig. 4.7 with correct sizing. All in all, the cutoff path reduces to a smaller clock buffer tree, a DFF, and a transistor (only one gC input on $\sim run$ changes). These optimizations give an extremely large margin between the local clock feedback and the cutoff path signal.

The local clock is designed such that Tune 0 matches the critical path conservatively in simulation. Section 4.3 shows that the local clock is faster than the critical path at 5 V on Tune 1. The difference between Tune 0 and Tune 1 is very slight, but Tune 1 is slower than Tune 0; therefore, the local clock is too fast at 5 V on Tune 0 as well. This may be due to a number of effects. For example, there may be a temperature gradient causing the local clock to run faster than the critical path. Or a process gradient may cause part of the chip to run faster than another. Also, process variation causes up to $\pm 20\%$ mismatch in gates depending on spacial locality. In addition to these physical effects, some error in the simulation exists. The model files used for simulation are not the extracted model files given after fabrication. And the simulated extracted view does not have extracted resistance. It has only extracted capacitance. Thus, the critical path is faster in simulation than it is in implementation and/or the local clock is slower in simulation than it is in implementation.

Although the local clock delay is meant to match the critical path delay, the two are affected by voltage changes differently. The critical path is measured at two low voltage points in Section 4.3. It is shown that at 1.5 V, Tune 0 is conservative enough to synchronize the multiplier. Yet, at 5 V, Tune 0 and even Tune 1 are too fast. This is due to the fact that matching the critical path exactly is difficult. First, the gates used in the local clock for fanout load are large inverters rather than the exact critical path gates. Second, the local clock gates are set up to act like inverters, with one node tied to ground or VDD. The multiplier critical path is an inverting path by definition; however, the gates do not have an input tied to VDD or ground. Instead the other inputs can be changing simultaneously. Simultaneous switching on a series stack exhibits worst-case behavior. In addition, glitchy lines in the critical path may slow the gates down, whereas the local clock delay inputs are held stable with VDD or ground.

In order to match the critical path exactly, many issues must be taken into account. First, the same gates that exist on the critical path must be used. Second, the exact load gates that exist on the critical path should be used to match fanout load. At this point, matching becomes harder. The critical path has coupling capacitance with the other bit-slices around it and the local clock may have different coupling capacitance around it. The critical path may have long wires which exhibit RC effects. The critical path nodes may be switching from low-to-high while the local clock matching node is switching from high-to-low (or *vice-versa*). The critical path may run at a different temperature than the local clock due to high switching activity in the core. Critical path nodes may have simultaneous switching or glitchy inputs that take time to stabilize. And process variation causes up to $\pm 20\%$ mismatch in gates depending on spacial locality. All of these effects add up to make delay matching difficult.

When these effects are not matched exactly, conservative tuning is necessary. How much tuning is needed is determined by how well the critical path delay is matched. The current delay has a four input one-hot tuner with four settings. Rather, a 16-setting fine grained tuner can be made with some decode logic and

the same four inputs. The tuner should be made to add and remove delay from the local clock delay. Simulations for desired voltage and temperature levels help determine how much tuning is needed.

The measured frequencies of the three local clocks are plotted in Fig. 4.9. Clock 1 and clock 2 are identical clocks placed on opposite sides of the chip. Both clocks feed an identical load and run at the same VDD. Thus, it is surprising that clock 2 is 16.9% slower than clock 1 on average. It is hypothesized that the difference is due to a temperature gradient. When measuring clock frequency, all clocks are placed in a free-run mode. Clock 1 is isolated in the top left corner of the chip, while clock 2 is next to the high frequency clock 3 in the bottom right corner. It is possible that clocks 2 and 3 run hotter than clock 1, and therefore clock 2 is slower than clock 1. It is also possible that a process gradient exists across all five chips. This process gradient may make the top portion of the chips faster than the bottom portion.

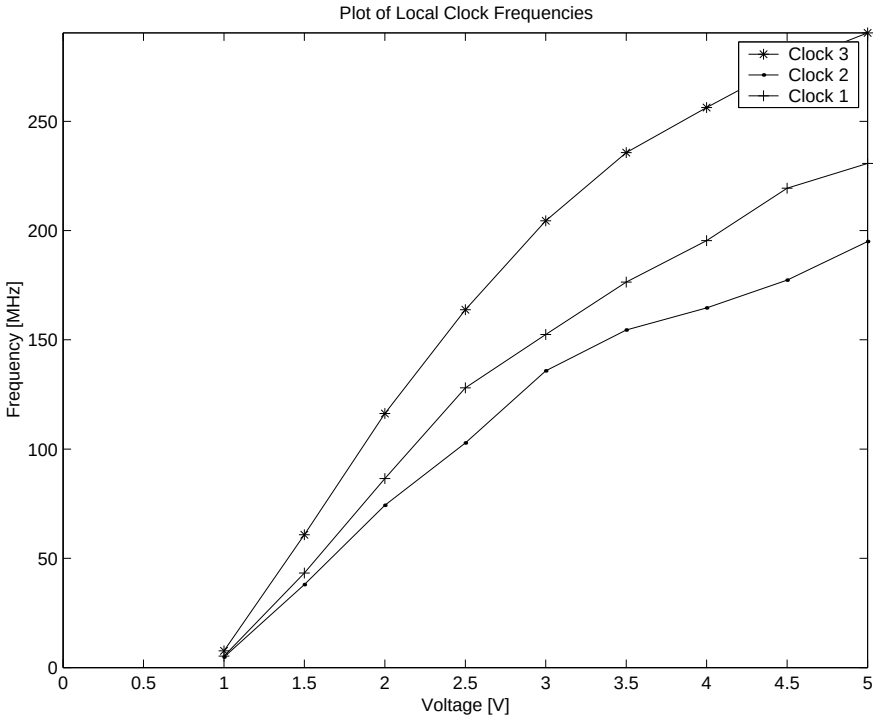


Fig. 4.9. Plot of frequencies for the three local clocks on Tune 0.

Clock 3 is a different design than the first two clocks because the delay in clock 3 is made of inverters. It is desired to know how clock 3 scales compared to the critical path. Unfortunately, the critical path cannot be measured except at very low voltages. Thus, clock 3 is compared to the first two clocks. Looking at Fig. 4.9, clock 3 scales approximately the same as the first two clocks with voltage changes. Nevertheless, as discussed above, the first two clocks do not scale the same as the critical path. Thus, it is unknown how clock 3 scales compared to the critical path. Perhaps if the first two clocks scaled the same as the critical path, they would scale differently than clock 3. It is possible to use a clock such as clock 3 if accurate simulation of the critical path is performed to find how much tuning clock 3 needs.

4.5 Discussion of Results

A thorough analysis of stoppable clocks and a locally-clocked module is performed and presented in this chapter. Performance of the module in terms of latency and power is presented. With correct tuning of the local clock, the multiplier generates correct results. With this tuning, at 5 volts VDD, the multiplier runs at 13.3 MHz and consumes 196.6 mW of power, while the stoppable clock runs at 174 MHz. At 3.0 volts VDD, the multiplier runs at 9.42 MHz and consumes 46.8 mW of power, while the stoppable clock runs at 117 MHz. As a baseline comparison, at 5 V VDD, the average latency of 6 *Nand2* gates with a fanout of 4 is 1.6 ns. And 6 *Xor2* gates have a latency of 2.6 ns. An alternate comparison is done with SPIM. SPIM is a 64×64-bit multiplier in a 1.6μm process. It has a multiplier throughput of 23.8 MHz, a local clock frequency of 85 MHz, and consumes 352.8mW of power at 5 V. Power consumption of SPIM is only 55% larger than our multiplier power. This is due to our use of standard-cells and standard-cell DFFs. It is well known that standard-cell designs consume more power than full-custom designs. Datapath elements are usually designed as full-custom modules. The multiplier should be re-implemented as a full-custom module to save power. If our multiplier is scaled to a 64×64-bit multiplier it has 22,534 transistors while SPIM has 41,000. This is an area savings of 100% not counting routing or transistor

sizing.

The analysis brings up many issues in the design of stoppable clocks. For example, the sizing of gC gates is important for speed of the cutoff path and for how low the clock control functions. Another issue is in the matching of the critical path. It is shown that matching the critical path exactly is difficult. Finally, the cutoff path is too slow for much of the testing space. Greater care must be taken to ensure that the cutoff path timing constraint is not broken.

On the other hand, the analysis showed some benefits to using a locally-clocked module. It is extremely easy to keep the multiplier idle. This is done by either cutting off all *gclk* edges, or by holding *reset* low. In either case, the local clock remains idle, and therefore the module remains idle. The only power consumption while the module is idle is from leakage current. The fact that this is such a small module makes leakage current extremely small. The small area module also has less capacitance than a large module; therefore, power consumption is smaller. And finally, it is shown that tuning is a necessary part of a local clock. Without tuning, the local clock design never meets the cutoff timing constraint. And without tuning, the local clock design is always faster than the critical path delay. Thus, tuning makes correct functionality of the multiplier possible.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This chapter describes the major contributions of this work and related future work. The analysis of local clocks in Chapter 4 brought up many issues which are not addressed in this work. These issues are left as future work in the area of stoppable clocks.

One contribution of this work is the evaluation of a particular locally clocked example. In the stoppable clock design, the frequency of the clock is limited by two factors — the cutoff path of the clock, and the ability to match the critical path closely. There are many factors which make it difficult to match the critical path. For example, simultaneous switching, fanout load, long wires, and coupling capacitance are hard to match. The current clock design only makes an attempt to match some of these. Because the critical path is not matched perfectly, some conservative delay must be added to the clock delay. This ensures correct functionality even though the critical path is not matched exactly; however, if it is possible to match the critical path exactly, the conservative delay can be removed. This speeds up the local clock and the module it synchronizes.

Rather than making the clock completely devoid of conservative delay, the clock has tuning built in which is used to change the additional conservative delay. Although the evaluated stoppable clock design is meant to track the datapath under a wide range of voltages and temperatures, it is shown that the clock requires tuning to match the critical path at some voltages and temperatures. This is because of the difficulty in matching the critical path exactly. It is also shown that certain voltage and temperature data points cause the cutoff path to be too slow. In these situations, clock tuning is needed to slow down the clock such that the cutoff path

becomes fast enough relative to local clock feedback. It is shown in this design that tuning is a necessary part of the local clock. Local clocks must be able to tune faster or slower than the simulated critical path delay. Thus, errors in delay matching are easily compensated for, and the design does not fail. Tuning can only be removed if the delay matching is excellent under all desired running situations, or if the clock is extremely conservative.

The work done in this thesis shows that local clocks can facilitate low-power design for three reasons. First, it is easy to keep a locally-clocked module idle. Either by cutting off the *gclk* signal, or by holding *reset* low. When the module is idle, power consumption is at a minimum. Second, high frequency clocks (230 MHz) can be generated locally through the use of stoppable clocks. In this situation, the global clock can remain at a low frequency while local clocks perform high frequency calculations. Third, small iterative modules which require a high frequency clock can be synchronized with local clocks without the need to increase the global clock frequency. Smaller modules are inherently lower power because they have less capacitance, and power consumption on a digital IC is directly proportional to capacitance.

The current local clock design has failure points for two reasons. First, gC sizing is incorrect. This makes the clock fail at a lower VDD limit of 1.14 V. If gC sizing is done correctly, the clock works below V_t (0.7 V). The second failure point is in the cutoff path. For many voltage and temperature settings, the cutoff path is too slow causing the clock to fail. Future designs can focus on speeding up the cutoff path; thus, matching the critical path delay is the only limiting factor on clock frequency.

The analysis of local clocks in Chapter 4 brought forth many issues that are not addressed in this work. For example, local clock frequency is limited by two major factors — the cutoff path of the clock, and the ability to match the critical path closely. In future designs the cutoff path can be optimized such that it is not a limiting factor. That leaves the critical path matching issue. This work shows that matching the critical path is difficult. Research can be done on how to match the critical path better.

Rather than attempting to match the critical path, a clock such as clock 3 can be used. This type of clock uses inverters for the local clock delay. They do not scale the same as critical path gates; however, with the right amount of tuning, this is not an issue. With the correct amount of tuning, the inverter clock is made to match the critical path under all variations in voltage and temperature. The difference is that it requires dynamic tuning if the voltage and temperature change during operation. Clock 3 is desirable because it ignores all matching issues, and it has a regular frequency, unlike clocks 1 and 2. Remember that clock 1 has two different delays between the pulses. With accurate simulation and further research in tuning methods, clock 3 is an acceptable alternative to critical path delay matching.

It is possible to make gC gates statically. Thus, the keeper is not necessary. Without the keeper, gC gates are no longer dynamic and do not suffer from dynamic gate problems; problems such as failure due to noise and power bumps. It is necessary to quantify the performance, size, power consumption, and lower VDD limit for static gC gates.

The multiplier design presented is a standard-cell design. This goes opposite to the goal of being a low-power design. It is well known that standard-cell designs consume more power than full-custom designs. Datapath elements are usually designed as full-custom modules. The multiplier should be re-implemented as a full-custom module to save power. Another power sink in the design is in the use of DFFs. It may be possible to use latches which are smaller and use less power. Some latch designs require a two-phase clock. A two-phase local clock is an area of research currently untouched.

Some issues pertaining to local clocks remain untested due to the difficulty in testing them. One such issue is the hypothesis that electromagnetic emission of many locally-clocked modules has less noise impact compared to a design with an off chip high frequency clock. Testing this hypothesis is a difficult matter because EMI in a small design is fairly insignificant. It is also desirable to perform a quantification of noise rejection for the local clock. Unfortunately it is difficult to inject white-Gaussian noise into a system in a controlled manner. It is also desirable

to know how the local clock reacts to cross-talk effects. This helps determine how shielded the local clock has to be.

Another issue is that the local clock design has a complete lack of jitter control. Each clock cycle has a different length compared to the others. Also the local clock has asymmetric pulse widths. Normally, jitter is controlled with phase-lock loops. In addition, off chip crystal oscillator frequencies are inherently more regular than local clock frequencies. With a lack of jitter control, the local clock requires conservative tuning. If local clocks are made with jitter control, some of the conservative delay can be removed.

Finally, local clocks will be built in future technologies. These technologies will affect performance and will introduce new issues to local clocks. Solutions to these new problems must be discovered as technology progresses.

REFERENCES

[1] C. J. Myers, *Computer-Aided Synthesis and Verification of Gate-Level Timed Circuits*. PhD thesis, Dept. of Elec. Eng., Stanford University, Oct. 1995.

[2] A. J. Acosta, R. Jiménez, A. Barriga, M. J. Bellido, M. Valencia, and J. L. Huertas, "Design and characterisation of a CMOS VLSI self-timed multiplier architecture based on a bit-level pipelined-array structure," *IEE Proceedings, Circuits, Devices and Systems*, vol. 145, pp. 247–253, Aug. 1998.

[3] R. G. Burford, X. Fan, and N. W. Bergmann, "An 180 MHz 16 bit multiplier using asynchronous logic design techniques," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 215–218, 1994.

[4] J.-S. Chiang and J.-Y. Liao, "A novel asynchronous control unit and the application to a pipelined multiplier," in *Proc. International Symposium on Circuits and Systems*, vol. 2, pp. 169–172, June 1998.

[5] C. H. Lau, D. Renshaw, and J. Mavor, "A self-timed wavefront array multiplier," in *Proc. International Symposium on Circuits and Systems*, pp. 138–141, 1989.

[6] J. B. Lipsher and K. Maheswaran, "A 4-bit asynchronous pipelined multiplier in the Xilinx 4000 series FPGA," tech. rep., University of California, Davis, 1994.

[7] O. Salomon and H. Klar, "Self-timed fully pipelined multipliers," in *Asynchronous Design Methodologies* (S. Furber and M. Edwards, eds.), vol. A-28 of *IFIP Transactions*, pp. 45–55, Elsevier Science Publishers, 1993.

[8] K. Y. Yun, P. A. Beerel, V. Vakilotojar, A. E. Dooply, and J. Arceo, "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 140–153, IEEE Computer Society Press, Apr. 1997.

[9] A. D. Booth, "A signed binary multiplication technique," *Quarterly Journal of Mechanical Applied Mathmatics*, vol. 4, no. 2, 1951.

[10] 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*, vol. 4, pp. 146–149, Mar. 1996.

[11] J. Haans, K. van Berkel, A. Peeters, and F. Schalij, "Asynchronous multipliers as combinational handshake circuits," in *Asynchronous Design Methodologies* (S. Furber and M. Edwards, eds.), vol. A-28 of *IFIP Transactions*, pp. 149–163, Elsevier Science Publishers, 1993.

[12] C. D. Nielsen and A. J. Martin, "Design of a delay-insensitive multiply-accumulate unit," *Integration, the VLSI journal*, vol. 15, pp. 291–311, Oct. 1993.

[13] J. Sparsø, C. D. Nielsen, L. S. Nielsen, and J. Staunstrup, "Design of self-timed multipliers: A comparison," in *Asynchronous Design Methodologies* (S. Furber and M. Edwards, eds.), vol. A-28 of *IFIP Transactions*, pp. 165–179, Elsevier Science Publishers, 1993.

[14] D. Kearney and N. W. Bergmann, "Bundled data asynchronous multipliers with data dependant computation times," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 186–197, IEEE Computer Society Press, Apr. 1997.

[15] 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, pp. 1651–1661, Nov. 1991.

[16] T. Williams, N. Patkar, and G. Shen, "Sparc64: A 64-b 64-active-instruction out-of-order-execution mcm processor," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 1215–1226, Nov. 1995.

[17] M. Santoro and M. A. Horowitz, "SPIM: A pipelined 64x64-bit iterative multiplier," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 487–493, Apr. 1989.

[18] K. Killpack, E. Mercer, and C. Myers, "A standard-cell self-timed multiplier for energy and area critical synchronous systems," in *Advanced Research in VLSI*, pp. 188–201, Mar. 2001.

[19] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequency*, vol. 34, pp. 349–356, Mar. 1965.

[20] K. Davis and G. Wilson, "*Private Communications*," Summer 2000. Keith Davis and Gerald Wilson are employees at SONIC Innovations.

[21] J. A. Tierno and A. J. Martin, "Low-energy asynchronous memory design," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 176–185, Nov. 1994.

[22] S. Brown and Z. Vranesic, "Testing of logic circuits," in *Fundamentals of Digital Logic with VHDL Design* (K. T. Kane, ed.), ch. 11, McGraw-Hill, 2000.

[23] P. Alfke, "Xapp 052," July 1996. Xilinx Application Note.

[24] M. J. Smith, "Test," in *Application-Specific Integrated Circuits*, ch. 14, Addison-Wesley VLSI Design Series, 1997.

[25] R. Sproull and I. Sutherland, "Stoppable clock," January 1985. Technical Memo 3438, Sutherland, Sproull, and Associates.

[26] C. L. Seitz, "System timing," in *Introduction to VLSI Systems* (C. A. Mead and L. A. Conway, eds.), ch. 7, Addison-Wesley, 1980.

[27] M. Afghahi and C. Svensson, "Performance of synchronous and asynchronous schemes for VLSI systems," *IEEE Trans. Computers*, vol. 41, pp. 858–872, July 1992.

[28] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Stanford University, Oct. 1984.

[29] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immediato, and K. Jenkins, "Asynchronous interlocked pipelined cmos circuits operating at 3.3-4.5ghz," in *International Solid State Circuits Conference*, Feb. 2000.

[30] K. Y. Yun and A. E. Dooply, "Pausible clocking-based heterogeneous systems," *IEEE Transactions on VLSI Systems*, vol. 7, pp. 482–488, Dec. 1999.

[31] W. Lim, "Design methodology for stoppable clock systems," *IEE Proceedings, Computers and Digital Techniques*, vol. 133, pp. 65–69, Jan. 1986.

[32] M. J. Stucki and J. J. R. Cox, "Synchronization strategies," in *Proceedings of the First Caltech Conference on Very Large Scale Integration* (C. L. Seitz, ed.), pp. 375–393, 1979.

[33] M. Pechoucek, "Anomalous response times of input synchronizers," *IEEE Trans. Computers*, vol. 25, pp. 133–139, Feb. 1976.

[34] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T.-P. Fang, "Q-modules: Internally clocked delay-insensitive modules," *IEEE Trans. Computers*, vol. C-37, pp. 1005–1018, Sept. 1988.

[35] A. E. Sjogren and C. J. Myers, "Interfacing synchronous and asynchronous modules within a high-speed pipeline," in *Advanced Research in VLSI*, pp. 47–61, Sept. 1997.

[36] G. Taylor, S. Moore, S. Wilcox, and P. Robinson, "An on-chip dynamically recalibrated delay line for embedded self-timed systems," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 45–51, Apr. 2000.

[37] J. D. Garside, W. J. Bainbridge, A. Bardsley, D. A. Edwards, S. B. Furber, J. Liu, D. W. Lloyd, S. Mohammadi, J. S. Pepper, O. Petlin, S. Temple, and J. V. Woods, "AMULET3i — an asynchronous system-on-chip," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 162–175, IEEE Computer Society Press, Apr. 2000.

[38] K. Y. Yun, P. A. Beerel, and J. Arceo, "High-performance two-phase micropipeline building blocks: double edge-triggered latches and burst-mode select and toggle circuits," *IEE Proceedings, Circuits, Devices and Systems*, vol. 143, pp. 282–288, Oct. 1996.

View publication stats