STATE UNIVERSITY OF NEW YORK AT BUFFALO

This is to certify that I have examined this copy of a M.S. thesis by

ALWIN TAREEN

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

NIHAR R. MAHAPATRA

---

Name of Faculty Advisor

---

Signature of Faculty Advisor

---

Date

GRADUATE SCHOOL

# ANALYZING DELAY ELEMENTS AND GATE TRIGGERING FOR GLITCH POWER ELIMINATION IN ARITHMETIC UNITS

by

ALWIN TAREEN
February 1, 2001

A thesis submitted to the
Faculty of the Graduate School of
State University of New York at Buffalo
in partial fulfillment of the requirements for the
degree of

Master of Science

Department of Electrical Engineering

## ACKNOWLEDGEMENTS

I sincerely thank my advisor Dr. Nihar Mahapatra for his constant support and encouragement throughout this research work. I thank him for giving me the opportunity to work in this interesting area. It has benefited me immensely in terms of both intellectual and personal growth.

Acknowledgements are due to my thesis committee members, Dr. Nihar Mahapatra, Dr. Ramalingam Sridhar, and Dr. Shambhu Upadhyaya for their valuable time and feedback.

Last and most important, I thank my family for their constant support and encouragement. It is their upbringing and education that has allowed me to reach this point in my career.

*To my parents*

## ABSTRACT

This thesis presents a new framework called gate triggering for systematically minimizing glitch power dissipation in static CMOS ICs. It is based on the idea that glitches can be effectively minimized by triggering logic evaluation at a gate only when all of its inputs have stabilized. For this purpose, to every potentially glitchy gate (or a suitable subset of such gates) is added a small amount of control logic, which, when enabled, triggers logic evaluation to the gate. I present six specific techniques based on gate triggering that differ in the type of control logic, and the way each is used to control a gate. A clocking mechanism is employed to generate enable signals at the proper times for all gates to be triggered. This mechanism uses delay elements, and I comprehensively review four categories of them: transmission gate based, cascaded inverter based, voltage controlled based and thyristor based. I compare the delay elements, both analytically and through simulations, in terms of four important parameters: delay, signal integrity, power consumption and area. Out of the previously mentioned six gate triggering techniques, the best one is selected for application on a ripple carry adder and carry save multiplier. The power reducing effectiveness is measured and compared with other existing glitch minimization techniques.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction: An Overview of the Glitching Problem

## 1.1 Glitching

A static CMOS gate is simply a device that is used to produce an output of logical-0 or logical-1, depending upon the value of its inputs. When a series of different input values are applied to the gate, its output will change accordingly, either staying at the same value or making a transition. Such is the case if we assume that all of the inputs arrive at the gate at the same time. In practice, such an assumption is not always valid. Many of today's integrated circuits consist of intricate networks of gates, where the output of one gate contributes an input to one or more other gates. Often, a gate has an input that depends upon the evaluation of an entire logic module. Any scenario where a static CMOS gate has one or more of its inputs arriving at different times, means that the gate is susceptible to glitching.

The output of a static CMOS gate can make a transition when a new set of inputs is applied to

it. In theory, the output should make no more than one transition after the gate has evaluated its new inputs. Under certain conditions, the gate output may make several transitions before settling down to a final output value. These intermediate transitions are useless, and they are referred to as glitches, critical races, or hazards. Since static CMOS gates consume power during switching, glitches can be a major source of wasted power in a circuit. Also, if the glitchy output of one gate feeds the input of another, incorrect logical information is being passed. The subsequent gate will take in the glitch as a valid input, and use it to evaluate the outputs. In this manner, glitches can propagate throughout the circuit, and in some cases they can multiply substantially. Eventually, the gates will reach will reach their correct logical output values, but the glitchy information that passes through the circuit wastes a great deal of power.

There are two conditions that must be present for a glitch to occur. The first condition requires that a specific input pattern which causes glitches must be applied. These glitch-causing patterns are unique to each type of gate, or static CMOS network. The second condition proposes that the interval time between the arrival of different inputs at a gate must be larger than the propagation delay of that gate. The gate will calculate a glitch using the earliest arriving input, and the value left behind from a previous calculation on the other input. The terminology given to the types of glitches that can occur is as follows. A static hazard is where a gate's output undergoes two or more transitions when it is expected to remain unchanged. A dynamic hazard occurs when the output is expected to make a single useful transition, either from 0 to 1 or 1 to 0. In these cases, the output makes several spurious transitions before settling down to its final output value [15].

Glitching occurs in real circuits primarily because of the signal mismatching in the network. Each gate in a CMOS circuit has a certain amount of propagation delay associated with it. These delays have different values for different gates. Also, the gates are interconnected according to the

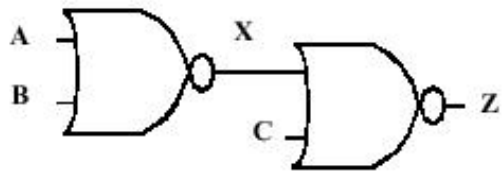



Figure 1.1: Two NOR gates

type of logic function that needs to be implemented. These factors mean that signals can arrive at intermediate nodes in the network at widely varying times. Consider the example of two 2-input NOR gates in series [16], where the output of one of the gates feeds an input of the other(Figure 1.1). Each gate has a similar amount of propagation delay. The second NOR gate will initially evaluate input $C$ and the value on intermediate node $X$ from the previous input set.

This evaluation will show up on the output. When the first NOR gate produces an output from $A$ and $B$ after its delay, the second NOR gate will evaluate from this new value. It will produce an output based upon this. The second NOR gate evaluates twice, with the first evaluation being logically unnecessary. If it caused a power consuming transition, then that amount of power would be wasted. To further illustrate this effect, consider the transition from $ABC = 101$ to $ABC = 000$ in this network. The propagation delay of the first NOR gate creates a static-0 hazard on the output.

| | | **BC** | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| **A** | 0 | 1 | 0 | 1 | 0 |
| | 1 | 0 | 0 | 0 | 0 |

Table 1.1: Karnaugh Map of $F = A'B'C' + A'BC$

Consider the Karnaugh map of Table 1.1. If we apply a set of inputs such that $ABC = 000$, the output is $F = 1$. Suppose we want to make a transition from $ABC = 000$ to $ABC = 011$. In an ideal situation, the output should remain at $F = 1$. However, if input $B$ is delayed so that the inputs are

briefly at $ABC = 001$, the output will change to $F = 0$ until the high signal at input $B$ arrives.

## 1.2 Power Concerns

There is a growing need for effective, low-power design techniques for static CMOS circuits. Portable, battery operated electronic devices are currently experiencing a tremendous surge of growth. In particular, the communications market is rapidly expanding, with each provider seeking an advantage in enhanced system performance. Unfortunately, the rapid development in VLSI technology has not been reflected in developments in battery technology. Ultimately, the impetus is on the chip designer to adopt emerging technologies that provide complex high-throughput low-power systems.

A further motivation for low-power design is simply that there are limits to how much power a component can dissipate without the need for special cooling. High performance CMOS designs are already exceeding these limits and, for these, the costs of the ultimate system are dominated by the costs of the cooling apparatus. Significant market advantage can be gained by using low- power IC's which avoid the need for this special hardware.

Reducing the power consumption of a circuit by scaling down various device parameters such as power supply voltage ($V_{DD}$) has been an effective low- power technique. However, we are rapidly reaching the limitations of how far these parameters can be reduced. Therefore, other low-power design techniques are being called upon more often to solve the power consumption problem. The quest for faster, better performing circuits only exasperates this issue. Device sizes have been linearly decreasing over the last four decades, since smaller devices improve the speed and power properties of a circuit.

Smaller devices also mean that more of them can fit on a single die. This increase in transistor

density means that the overall power consumption of the die also increases. Also, this effect increases the power dissipation of the die, and at high temperatures special cooling techniques may be required. Increasing the frequency speed of a circuit means that the gate will switch more frequently in a given time frame. More switching means that the circuit will consume more power. Portable computer systems rely on low-power techniques of maximize use of limited battery resources.

The power consumption of a CMOS gate can be divided into 2 parts: standby power consumption and active power consumption. Standby power consumption is caused by static conductive paths between the supply rails or subthreshold leakage currents. It is a relatively small value. Active power consumption occurs only when the gate is switching. It is due to the charging of capacitors and the temporary current paths between the supply rails. It is directly proportional to the switching frequency. More power is consumed by gates with very high switching frequencies.

# Chapter 2

# An Analysis of the Techniques Currently Used to Overcome Glitching

## 2.1 Circuit Level

### 2.1.1 Retiming (signal path balancing)

The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of a network.

Consider the gate configurations of Figure 2.1. Assume that all operators $F$ have the same unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate $F3$, one input settles at time 0, while the second one only arrives at time 2. Redesigning the network so that all arrival times

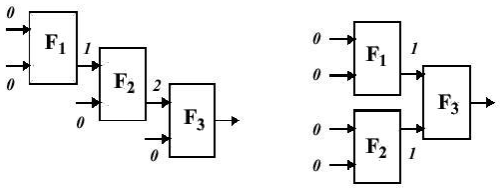



Figure 2.1: Gates with unequal path lengths

are identical can dramatically reduce the number of transitions (network b). In a few instances, it is possible to rearrange the gates in a network while retaining the logical integrity of the system. In other cases, such rearrangements would lead to incorrect logical outputs being produced. If the penalty of rearranging a network is too high, an alternate solution would be to introduce buffers along the signal paths. Inserting buffers along short signal paths allows them to synchronize with longer signal paths. Unfortunately, this solution ensures that the worst-case signal path time is applied everywhere, a situation that is undesirable in some circumstances. Also, buffers introduce an additional power and area overhead into the circuit, and they are notorious for degrading a signal's voltage integrity.

### 2.1.2 Gate resizing

This technique eliminates glitches by equalizing all path delays in a combinational logic network. Path equalization is obtained by downsizing gates that do not lie on critical paths, thereby slowing down fast propagation paths, without changing the worst case delay of the circuit [12] [13]. Effective path equalization by resizing requires large technology libraries with many differently sized functionally equivalent logic gates. In addition, perfect equalization can be difficult to achieve if path delays are distributed very unevenly. In fact, if the delay of a short path is still smaller than that of the critical path when all gates on such a short path have been scaled down to minimum size,

spurious activity remains possible(multipliers are typical examples of circuits with widely different path delays). Finally, the resizing approach is applied before physical design. Therefore, the cost metric employed for driving the resizing procedure has limited accuracy. If we want to have approximately the same rise and fall times for an inverter, we must make

$$W_p \approx (2 \rightarrow 3)W_n \tag{2.1}$$

where $W_p$ is the channel width of the p device and $W_n$ is the channel width of the n device. However, a first-order analysis shows us that increasing the transistor sizes also increases the diffusion capacitance, load capacitance and gate capacitance. The last component increases the fan-out factor of the driving gate, and hence increases the propagation delay. Therefore, we can use a clever gate resizing algorithm to alter the timing delay of a signal path, to synchronize with other gates and hence reduce glitching. Gate resizing doesn't affect functional transitions, but it affects glitch transitions considerably. It's been described as an ill-behaved problem, because larger transistor sizes consume more power, which adversely affects the power savings that glitch reduction brings.

### 2.1.3 Guarded evaluation

This method exploits automatic latch insertion to reduce power consumption. In this approach, the internal signals that are available in the combinational network are used as latch-enable controls. The rationale in guarded evaluation is that of preventing the switching of nonobservable sections of a combinational logic network by inserting latches controlled by signals that imply nonobservability. The method has produced promising results, but it suffers from the same drawback of the retiming technique, i.e., it is applied before physical synthesis. Latch insertion significantly perturbs both cell count and netlist connectivity; thus, postlayout results are not very predictable and optimization may be quite inaccurate.

### 2.1.4 Register transfer level (RTL) circuit glitch reduction

A number of register transfer level transformations for glitch power optimization have been introduced in [14]. Here, the focus is on glitches generated by misaligned control signals that have long propagation paths within the data path and cause sizable power consumption. The main objection to this approach is that at the register transfer level only approximate timing information is available, and the glitch power reduction predicted at this level may be an inaccurate estimate of the result achieved after logic synthesis, technology mapping, placement and routing. On the other hand, a key advantage of the method is that it performs local low-impact transformations on control signals that fan out to large portions of the data path. Register transfer level transformations reduce glitching by a vast amount that is unlikely to be completely cancelled in the later phases of the synthesis flow.

## 2.2 Design Level

### 2.2.1 Redundant logic

The strategy for detecting and eliminating hazards in two-level networks is relatively straightforward. For static-1 hazards, we start with the Karnaugh map of a function, and examine it to make sure that all adjacent elements of the on- set are covered by a prime implicant. If they are not, we add redundant prime implicants until all elements of the on-set are covered.

We follow a similar procedure to eliminate static-0 hazards. Given the sum of products form for the function that eliminates static-1 hazards, we write it in a product of sums form using Boolean algebra. Then we verify that the adjacent elements of the off-set are covered by a common prime implicant in the product of sums form. If necessary, we add more prime implicants to cover any

uncovered adjacencies.

We can use a generalized version of this procedure to eliminate glitching in multilevel networks.

### 2.2.2 SOP/SOP complex gate realization

CMOS complex gate networks can be implemented in many different ways. The standard technique to implement the functions $f$ and $f'$ is to obtain a sum of products for one transistor network(p or n), and calculate the dual of this to obtain the complementary product of sums network. By dual networks, we mean that a parallel connection of transistors in the pull-up network corresponds to a series connection of corresponding devices in the pull-down network and vice versa. The duality condition is a satisfying but not a necessary requirement. Other PUN/PDN combinations are valid, such as the SOP/SOP form of complex gates. Here, both $f$ and $f'$ are implemented as sum of products networks [10].

We will first examine the hazard behavior of the SOP/SOP form of realization for CMOS complex gates. In order to do this, we will examine both single-input change(SIC) and multiple-input change(MIC) static and dynamic transitions on a case-by-case basis.

#### 2.2.2.1 Case 1: static transitions

For static transitions a SOP/SOP complex gate circuit is hazard-free at the output. Both SIC and MIC static hazards occur when a given static transition causes a change from one cube of the cover to another, causing a brief period when the transistor network is not conducting. Consider $f$ and $f'$ to be on-set and off-set covers respectively implemented as p and n transistor networks in a complex gate. It has been shown that a sum of products implementation of the on-set $f$ does not have any $0-0$ hazards. Similarly, $f'$ does not have any $1-1$ hazards. This means that in a sum of products form, a static transition over function $f$ is always outside the cover of $f'$ and vice versa. As a result,

for a SOP/SOP form of complex gates, even if the transistor network of $f$ has a static hazard(that is, a brief moment when no p stack is conducting), the transistor network of $f'$ remains off(that is, no n stack will conduct during the transition). Therefore, the output capacitance of a SOP/SOP complex gate holds its current charge for the duration of a static hazard(we have no conducting path to either power or ground), and the hazard is not seen at the output.

As an example, consider the Karnaugh map and complex gate implementation in [10]. Transition t, from $abc : 011 - 010$ is a static $1 - 1$ transition. The on-set is implemented in the p-transistor pass network, using cubes $A = bc'$ and $B = a'c$. This cover is hazardous for a simple gate AND-OR network. During transition t, the AND-gate for $B$ goes low and the AND-gate for $A$ goes high. If the AND-gate for $A$ is slower than the AND-gate for $B$, the OR-gate output will glitch. In contrast, the single complex-gate network is hazard-free. Although the p-transistor stacks for $A$ and $B$ can briefly be off at the same time(when $c$ goes low), no n-transistor pass network will conduct during the transition. As a result, the output will hold its current charge. Therefore, there is no need to avoid static hazards during synthesis of $f$ and $f'$.

#### 2.2.2.2 Case 2: dynamic transitions

For the case of SIC transitions, it has been shown that a dynamic SIC hazard cannot occur(assuming no product contains both a variable and its complement). Since $f$ and $f'$ are in two-level AND-OR form, no hazards will occur in the complex-gate output in this case. For the case of MIC transitions, though, we will have to make the p and n pass networks hazard-free for dynamic transitions. Otherwise, even in the SOP/SOP form, both the p network and n network may have dynamic hazards, creating a hazard at the output of the complex gate.

# Chapter 3

# A New Glitch Minimization Framework: The Control Logic Architectures

## 3.1 Proposed Methodology

The overall methodology upon which our glitch minimization techniques are based are presented here. The key idea we employ to minimize glitches is to trigger logic evaluation at a gate only after all of its inputs have stabilized. For this purpose, to every potentially glitchy gate(i. e., a gate with unequal propagation delays for its inputs), we add some small control logic. When it is enabled, the gate may proceed with the logical evaluation of its inputs. Essentially, this control logic regulates the connection between the p-network and $V_{DD}$, or the n-network and GND($V_{SS}$). In order to enable the control logic for different gates in a combinational logic block at the proper times, a timing simulation must first be performed. A timing simulation is an essential step in the design flow of a VLSI chip. It determines the critical-path delay in a combinational block, which is used to calculate the clock period. Therefore, it does not represent an extra step in the application of our method.

From this timing simulation, we obtain the delays of different gates and the latest times by which the various inputs of a gate will have stabilized. For instance, in Figure 3.1, gate g01 has a delay of five time units, and its top, middle and bottom inputs have their latest steady- state values at five, four and two time units, respectively.

In order to prevent glitches at the output of gate g01, its control logic can be enabled at time five and should remain enabled for at least five time units. This is to ensure that the gate logic may evaluate completely, since five is the delay of the gate. Therefore, we use an enable signal for the combinational block with a high period equal to the maximum delay for any gate in the block. As shown in Figure 3.2, the enable signal is generated by taking the AND of the complemented clock signal with the clock signal delayed by the maximum delay. This initial enable signal is then delayed by various amounts using a delay chain consisting of a series of delay elements. The output of a delay element in this chain provides an appropriately delayed version of the initial enable signal that can be used to trigger a potentially glitchy gate.

Using this approach, all potentially glitchy gates are triggered by the enable signal when the last input to them stabilizes, ideally preventing all glitches in the combinational block. In practice, however, minor or partial glitches may occur due to the nonideal behavior of the transistors. It should also be noted that short-circuit power dissipation in all triggered gates can be minimized by triggering them after the last input has stabilized. However, in most cases it will not be cost-effective to control all gates in this manner because of the overhead it will entail. It is best to control a few select gates where the potential for both glitch and short-circuit power savings is a maximum.

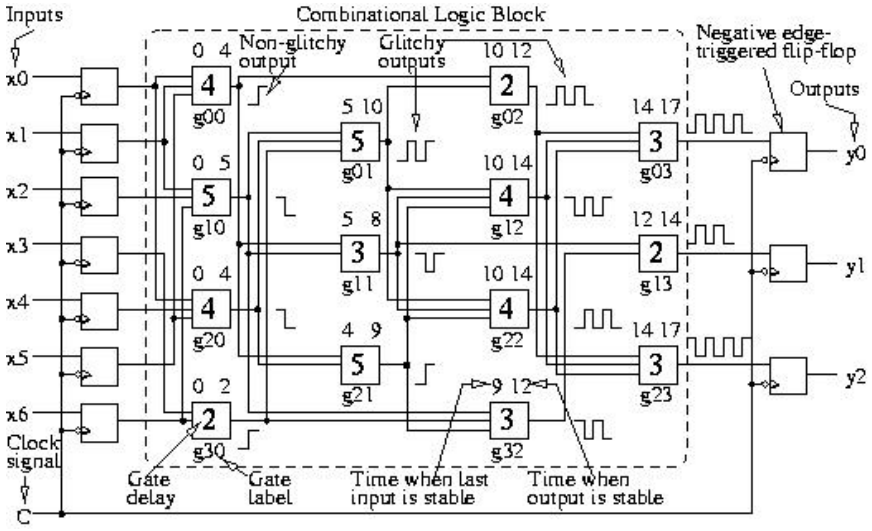



Figure 3.1: Glitchy circuit

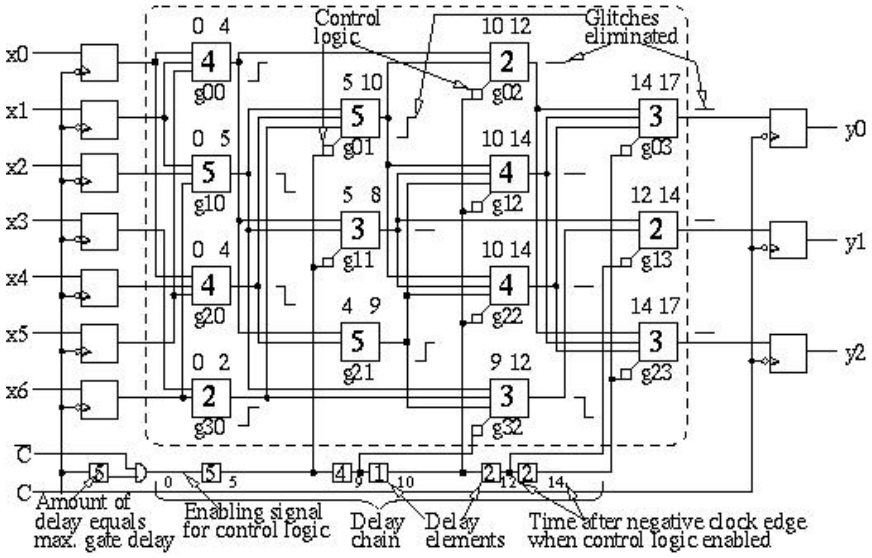



Figure 3.2: Glitch minimization applied

## 3.2 Delay Element Optimization

The overhead of the delay chain depends upon the type of delay element that is used. Different delay element architectures have varying values of area, delay and power consumption. Our goal is to use a delay element that occupies a minimal amount of area, and consumes very little power. In order for our glitch minimization method to be effective, the power overhead introduced by the delay chain must be less than the power saved by reducing the glitches in the combinational logic. The number of delay elements needed depends upon the number of gates that we wish to control. Some gates need to be triggered at the same time as others, and for these we can take the same enable signal from the delay chain. For example, in Figure 3.2, the set of gates g03, g11 and g21, are triggered synchronously by the enable signal delayed by five time units. Note that to synchronize, control logic is added to some gates which are triggered later than they normally would be(such as g21). Also, the gates selected for late triggering are not on any critical path(shown in bold lines in Figure 3.2) so as not to increase the overall delay of the combinational block. The application of this optimization thus results in a much reduced number of delay elements in the delay chain, with little additional control logic overhead. A smaller number of delay elements means a smaller amount of wiring, since a fewer number of distinct enable signals are being routed.

## 3.3 Control Logic Optimization

There are two ways in which the control logic may be optimized. The first way consists of using the same control logic to govern all gates that are to be triggered simultaneously. For instance, in Figure 3.2, the set of gates g01, g11 and g21 can be controlled with the same control logic. However, sharing the control logic in this manner would mean that the sizes of the control logic's transistors

would have to be increased to comply with the greater fan-out. Another way to reduce the amount of control logic is to schedule the triggering of earlier gates so that the inputs to the later gates are synchronized. For example, in Figure 3.2, gates g11 and g02 can be triggered later than normal so that all inputs to gates g12, g22, g03 and g23 are synchronized. This eliminates the need for using control logic on these gates. The gates selected for late triggering are not on the critical path so that the overall delay of the combinational block remains the same. Synchronizing the inputs to later gates in this manner reduces the delay element and wiring overhead, since some enable signals no longer need to be generated and routed.

## 3.4 Glitch Minimization Techniques

### 3.4.1 Simulation methodology

Based on the gate triggering framework discussed previously, we have developed six specific techniques for glitch minimization. These techniques differ in the type of control logic used and the way it is connected to a gate to prevent glitches. In order to more easily analyze and compare the glitch minimization capabilities of these techniques, we first apply them to a highly glitchy test gate. By studying the influence on one gate, the effectiveness of the techniques on a combinational logic block consisting of many gates can be inferred. The test gate used is shown in Figure 3.3 and has eight inputs, which change one after another producing eight transitions at the output. The asynchronous arrival of inputs to the test gate model unequal propagation delays from the input of a combinational block to the inputs of an interior gate. We analyzed the test gate output $F$ in Figure 3.3 for all four possible initial-final output value combinations. The original test gate(without glitch minimization) is referred to as ckt0. The test gates with one of the six glitch minimization techniques applied are referred to as ckt1 through ckt6.

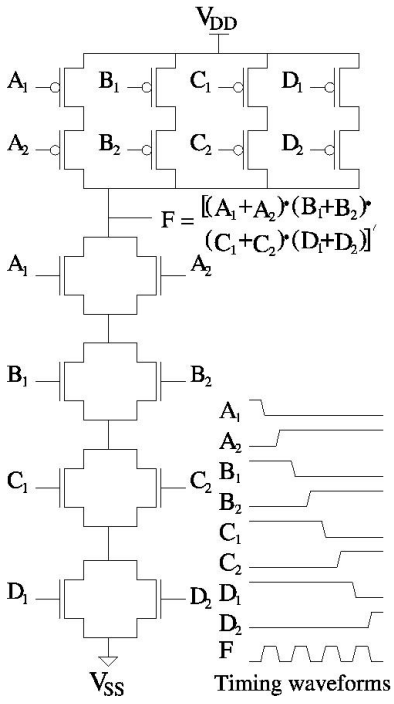



Figure 3.3: Test gate (ckt0)

### 3.4.2 Technique 1: Transmission gate at the output

In this technique, the control logic is a transmission gate, which is placed at the output of a potentially glitchy gate. The transmission gate prevents glitch propagation to the fanout gates by providing a reasonably clean output at $F$ as seen in the plots for ckt1 in Figures 3.11 to 3.14. As seen in Figure 3.4, glitches occurring in the earlier gates of a combinational logic block cause many more glitches at later gates. Thus, this technique can lead to significant glitch power reduction in combinational circuits. However, the technique has several drawbacks. First, glitchy transitions still occur and power is dissipated at the node just before the transmission gate. Second, a transmission gate is needed for every potentially glitchy gate not connected to an output flip-flop. Third, the introduction of a transmission gate increases the overall delay of the combinational block if the gate is on the critical path. The delay introduced by a transmission gate can be reduced somewhat by increasing the width of its transistors, for this reduces the resistance(the delay of a transmission gate

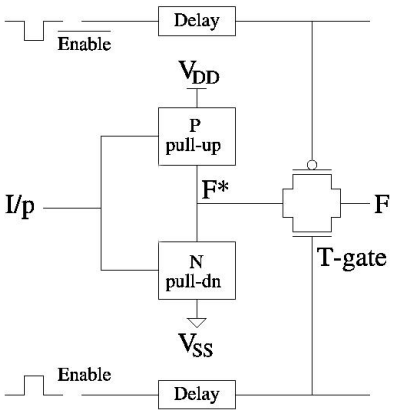



Figure 3.4: Technique 1 (ckt1)

is proportional to the product of its load capacitance and transistor resistance). Finally, in order to enable the transmission gate, two complimentary enable signals are required. This necessitates two delay chains that generate enable signals: one for the n transistor, and the other for the p transistor. There is an option of using a single delay chain that uses inverters to tap the complimentary enable signal from the main line. In such a configuration, attention must be paid to the delay of the inverter, which may put the main and complimentary enable signals out of sync. Due to this and the fact that an inverter occupies more area than a transmission gate, the dual delay chain configuration is preferred.

### 3.4.3 Technique 2: Control device between a transistor network and a supply line

In contrast to the technique described above, the remaining techniques not only prevent glitch propagation from a gate, but also minimize glitch power dissipation at the gate itself. They do so by controlling the connection of the gate output to $V_{DD}$ and/or $V_{SS}$ by means of an n and/or p control transistor. For instance, in the second technique shown in Figure 3.5, an n(or p) control transistor is connected between the n pull-down network of a potentially glitchy gate and $V_{SS}(V_{DD})$. As a result, before the last input stabilizes, the output $F$ of the gate can only rise to $V_{DD}$, but it can not discharge

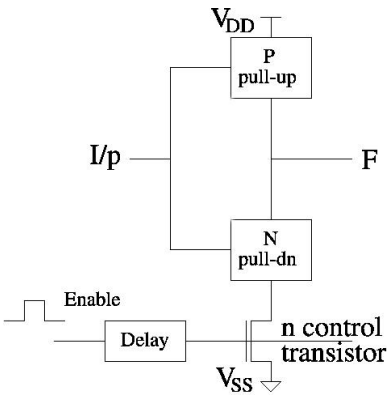



Figure 3.5: Technique 2 (ckt2)

to $V_{SS}$, since the n control transistor is off. Therefore, a glitch occurs with this technique in only one out of four cases when both the initial and final values of the output are zero. However, it should be noted that the output is partially glitchy in all cases since, between the first and the last input changes, it falls to some appreciable extent and then rises. The reason for this is as follows. In the previous clock cycle when the n control transistor was enabled, some of the n transistors in the n pull-down network were connected to $V_{SS}$, so that their drain capacitances were discharged to $V_{SS}$. In the current clock cycle, before the n control transistor is enabled and after the output rises to $V_{DD}$, these discharged internal capacitances may get connected to the gate output node because of some new input. This causes charge sharing between the output load capacitance and the internal capacitances of the n pull-down, thus bringing down the output voltage. These partial glitches consume some power and also get propagated, but to a lesser extent.

### 3.4.4 Technique 3: Control device between a transistor network and the output

To prevent charge sharing and thus provide a cleaner output, this technique(Figure 3.6) proposes that an n(or p) control transistor is connected between the n pull- down network(p pull-up network) and the output node. In this case, similar to the previous one, the output can only rise to $V_{DD}$ before

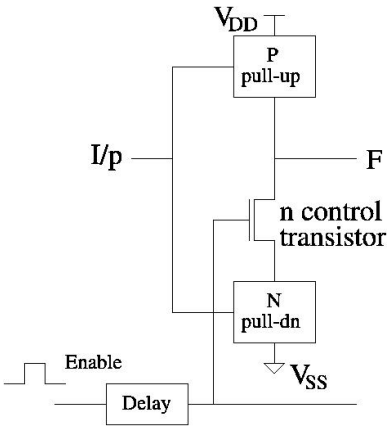



Figure 3.6: Technique 3 (ckt3)

the last input has stabilized, and there is a glitch only when the output has an initial and final value of zero. A peculiarity of this technique is that, when the final output is high, the output temporarily dips a little before rising. This happens because when the n control transistor is enabled, some of the discharged internal capacitances of the n pull-down network may get connected to the high output, bringing it down temporarily.

### 3.4.5 Technique 4: p transistor between $V_{DD}$ and p pull-up, n transistor between $V_{SS}$ and n pull-down

This technique(Figure 3.7) is designed to prevent the gate output from charging to $V_{DD}$ or discharging to $V_{SS}$ before the last input has stabilized, and thus prevent the single glitch that occurs in both of the previous techniques. Although the output does not charge up or discharge completely, it does so partially when it initially has a high value. This is because the initially charged up internal capacitances in the p pull-up network and the output capacitance share charge with the initially discharged internal capacitances in the n pull-down network. Thus, when the initial-final output value is $1-0$, the output discharges in steps. This is known as adiabatic discharge, and it consumes less power than a straight discharge. When the intial-final output value is $1-1$, there is a partial glitch, which

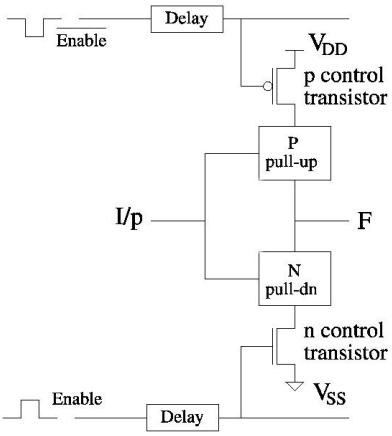



Figure 3.7: Technique 4 (ckt4)

gets propagated to the fanout gates.

### 3.4.6 Technique 5: p transistor between p pull-up and output, n transistor between n pull-down and output

This technique(Figure 3.8) prevents charge sharing between the output capacitance and the internal capacitances of the n pull-down and p pull-up, and provides a cleaner, glitch-free output. A minor dip occurs when the initial-final output value is $1-1$. This occurs because of the connection between the discharged internal capacitances in the n pull-down network and the charged capacitances in the p pull-up network.

### 3.4.7 Technique 6: p transistors isolating p pull-up, n transistors isolating n pull-down

The previous technique is plagued by the discharging of its internal capacitances, which results in power consumption. This is prevented in the final technique, Figure 3.9, in which two n and two p transistors are used to prevent unnecessary charge sharing and charging/discharging of the output capacitance and the internal capacitances in the n pull-down and p pull-up networks. This provides

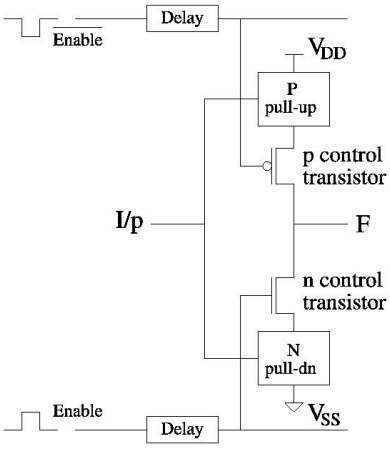



Figure 3.8: Technique 5 (ckt5)

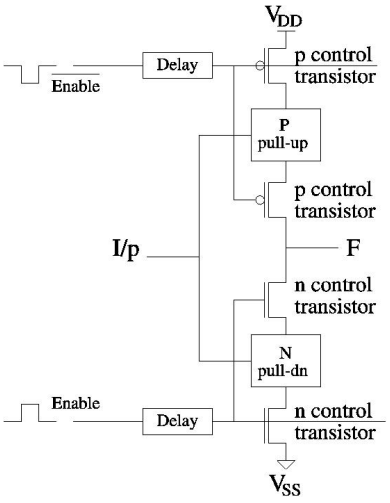



Figure 3.9: Technique 6 (ckt6)

a clean output signal, and minimizes the glitch power dissipation.

## 3.5 Comparison of the Glitch Minimization Techniques

There are several parameters which we can use to compare the performance of the glitch minimization techniques. They are glitch minimization capability, area overhead, timing delay overhead, and ease of application to a general circuit.

### 3.5.1 Glitch minimization capability

Technique 1 only provides glitch filtering, while the other techniques completely eliminate any glitches that may occur. Glitch minimization increasingly improves from technique 1 to technique 6, with one possible exception. In some cases, technique 4 may provide better glitch minimization than technique 5. During a $1-0$ output transition, technique 4's load capacitor discharges adiabatically(i.e. in stages). This leads to lower power consumption, because the only glitch that occurs is a partial glitch.

### 3.5.2 Area overhead

In terms of relative area overhead, techniques 2 and 3 are the lowest, each requiring just one control transistor. Also, they only use a single delay chain, while the other techniques require two delay chains. Techniques 1, 4 and 5 come next, since they each use two control transistors. The most costly one in terms of area is technique 6, for it requires four control transistors. One of the benefits of our glitch minimization architecture is that control logic can be amortized over several gates that need to be triggered at the same time. For instance, the control transistor overheads for techniques 2, 4 and 6 can be reduced by sharing the transistors connected to $V_{DD}$ or $V_{SS}$ with other potentially glitchy gates. However, such a scenario may require resizing the width of the shared control logic transistors to avoid increasing the delay of the critical path. Further overhead reduction may be achieved by selectively triggering only those gates that are expected to contribute the most to the glitch power dissipation.

### 3.5.3 Delay overhead

Technique 1 increases the overall delay of the gate to which it is applied, but this effect can be reduced by decreasing the widths of the control logic's transistors. Generally, the control logic used in the remaining techniques increase the rise and fall times of their gate's output signal. This effect can be minimized by decreasing the transistor widths of the control logic.

### 3.5.4 Ease of application

Given a certain gate, techniques 1, 2 and 4 can be applied with no modification of the layout. The other techniques require slight modifications because the n and p control transistors need to be introduced between the N pull-down network and the P pull-up network.

From the above analysis, technique 4 is the best overall because it has close to optimal glitch minimization capability and a low overhead. However, in any particular application, one or more techniques may be used for different gates. For example, technique 2 may be used for simpler gates(since it only requires one control transistor). Technique 4 may be used for gates of medium complexity, and technique 6 may be used on the most complicated gates.

### 3.5.5 Experimental conclusions

The point of testing different glitch minimization techniques on a highly glitchy test gate was to determine the best one for practical implementation. For arithmetic units, the most suitable technique was ckt4. It was the one that was selected for further experimentation in Chapter 5. The following is Table 3.1, comparing the average power dissipation using various glitch-minimization techniques relative to the test gate(ckt0) for different fanouts and input inter-arrival times. The fanouts used in this experimentation were quantities of 1, 2, and 4 inverters respectively. The input inter-arrival

| Fan out | Circuit | Input inter-arrival time | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0.5 Stage Delay | | 1.0 Stage Delay | | 1.5 Stage Delay | | 2.0 Stage Delay | |
| | | Avg. Power ($\mu W$) | % Reduction | Avg. Power ($\mu W$) | % Reduction | Avg. Power ($\mu W$) | % Reduction | Avg. Power ($\mu W$) | % Reduction |
| 1 | ckt0 | 44.98 | — | 59.22 | — | 59.75 | — | 60.82 | — |
| | ckt1 | 42.99 | 4.42 | 53.15 | 10.25 | 53.09 | 11.15 | 52.61 | 13.50 |
| | ckt2 | 24.11 | 46.40 | 24.83 | 58.07 | 25.07 | 58.11 | 25.39 | 58.25 |
| | ckt3 | 10.27 | 77.17 | 10.28 | 82.64 | 10.45 | 82.51 | 10.50 | 82.74 |
| | ckt4 | 4.45 | 90.11 | 5.06 | 91.46 | 5.22 | 91.26 | 5.26 | 91.35 |
| | ckt5 | 5.06 | 88.75 | 4.92 | 91.69 | 4.98 | 91.67 | 5.08 | 91.65 |
| | ckt6 | 3.34 | 92.57 | 3.14 | 94.70 | 3.37 | 94.36 | 3.27 | 94.62 |
| 2 | ckt0 | 46.24 | — | 71.26 | — | 73.77 | — | 73.48 | — |
| | ckt1 | 43.82 | 5.23 | 54.13 | 24.04 | 54.58 | 26.01 | 55.11 | 25.00 |
| | ckt2 | 27.78 | 39.92 | 28.84 | 59.53 | 29.14 | 60.50 | 29.26 | 60.18 |
| | ckt3 | 14.20 | 69.29 | 14.05 | 80.28 | 14.41 | 80.47 | 14.39 | 80.42 |
| | ckt4 | 4.58 | 90.10 | 5.07 | 92.89 | 5.22 | 92.92 | 5.21 | 92.91 |
| | ckt5 | 5.31 | 88.52 | 5.16 | 92.76 | 5.35 | 92.75 | 5.32 | 92.76 |
| | ckt6 | 3.48 | 92.47 | 3.29 | 95.38 | 3.50 | 95.26 | 3.42 | 95.35 |
| 4 | ckt0 | 46.84 | — | 90.78 | — | 98.66 | — | 98.56 | — |
| | ckt1 | 44.47 | 5.06 | 56.49 | 37.77 | 58.23 | 40.98 | 57.97 | 41.18 |
| | ckt2 | 34.48 | 26.39 | 35.87 | 60.49 | 35.83 | 63.68 | 36.01 | 63.46 |
| | ckt3 | 20.70 | 55.81 | 20.93 | 76.94 | 21.56 | 78.46 | 21.48 | 78.21 |
| | ckt4 | 4.74 | 89.88 | 5.32 | 94.14 | 5.39 | 94.54 | 5.42 | 94.50 |
| | ckt5 | 5.51 | 88.24 | 5.41 | 94.04 | 5.55 | 94.37 | 5.65 | 94.27 |
| | ckt6 | 3.73 | 92.04 | 3.56 | 96.08 | 3.62 | 96.33 | 3.62 | 96.33 |

Table 3.1: Comparing the reduction in average power dissipation for each glitch minimization technique

times were derived from the amount of time impeded by a single delay element. They were quantified as 0.5, 1.0, 1.5, and 2.0 stage delays. Following Table 3.1 is Figure 3.10, depicting the types of signal transitions produced by the glitchy test gate(ckt0). Then, the performance of the various glitch minimization techniques with each of these tranistion types are outlined in Figures 3.11 to 3.14.

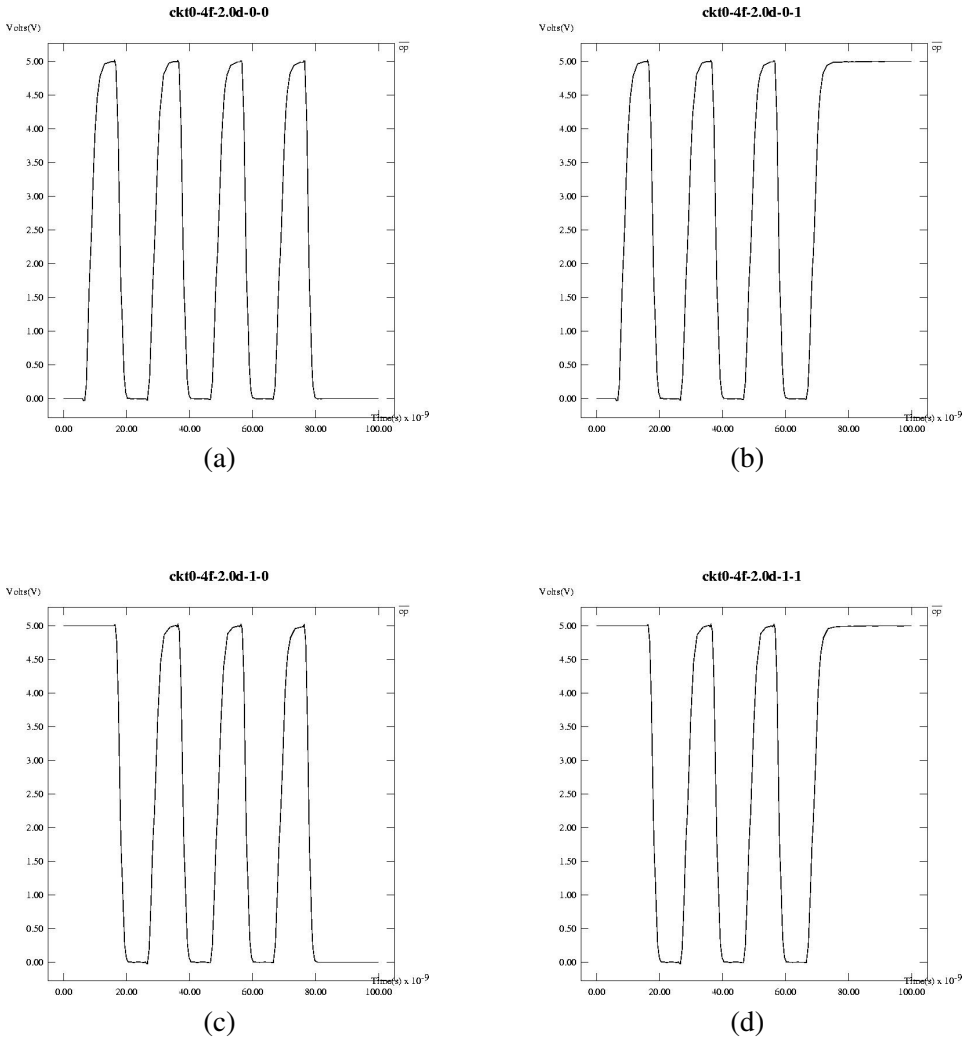



Figure 3.10: Test gate (ckt0) transitions: (a) $0-0$ (b) $0-1$ (c) $1-0$ (d) $1-1$

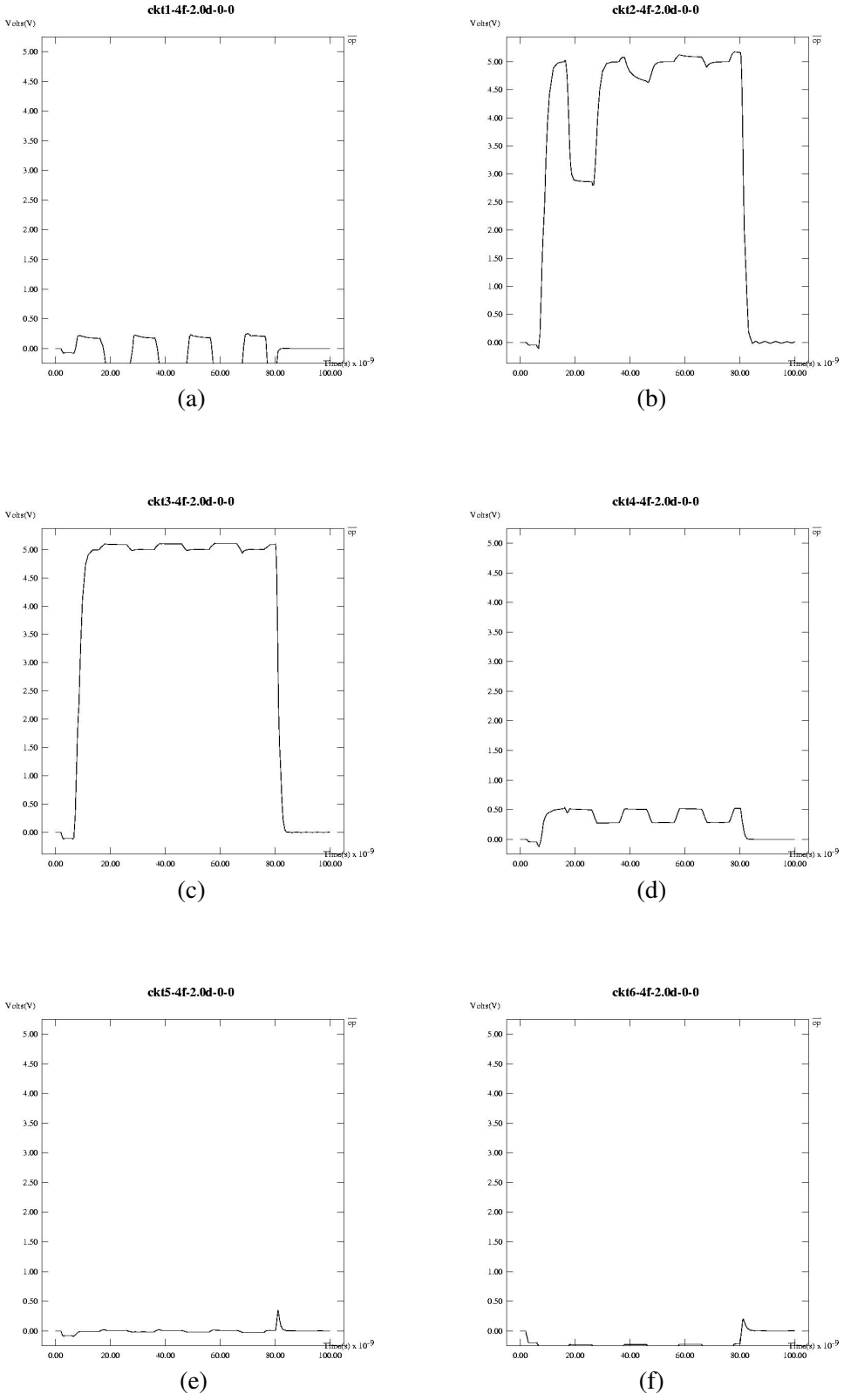



Figure 3.11: Transition $0-0$: (a) ckt1 (b) ckt2 (c) ckt3 (d) ckt4 (e) ckt5 (f) ckt6

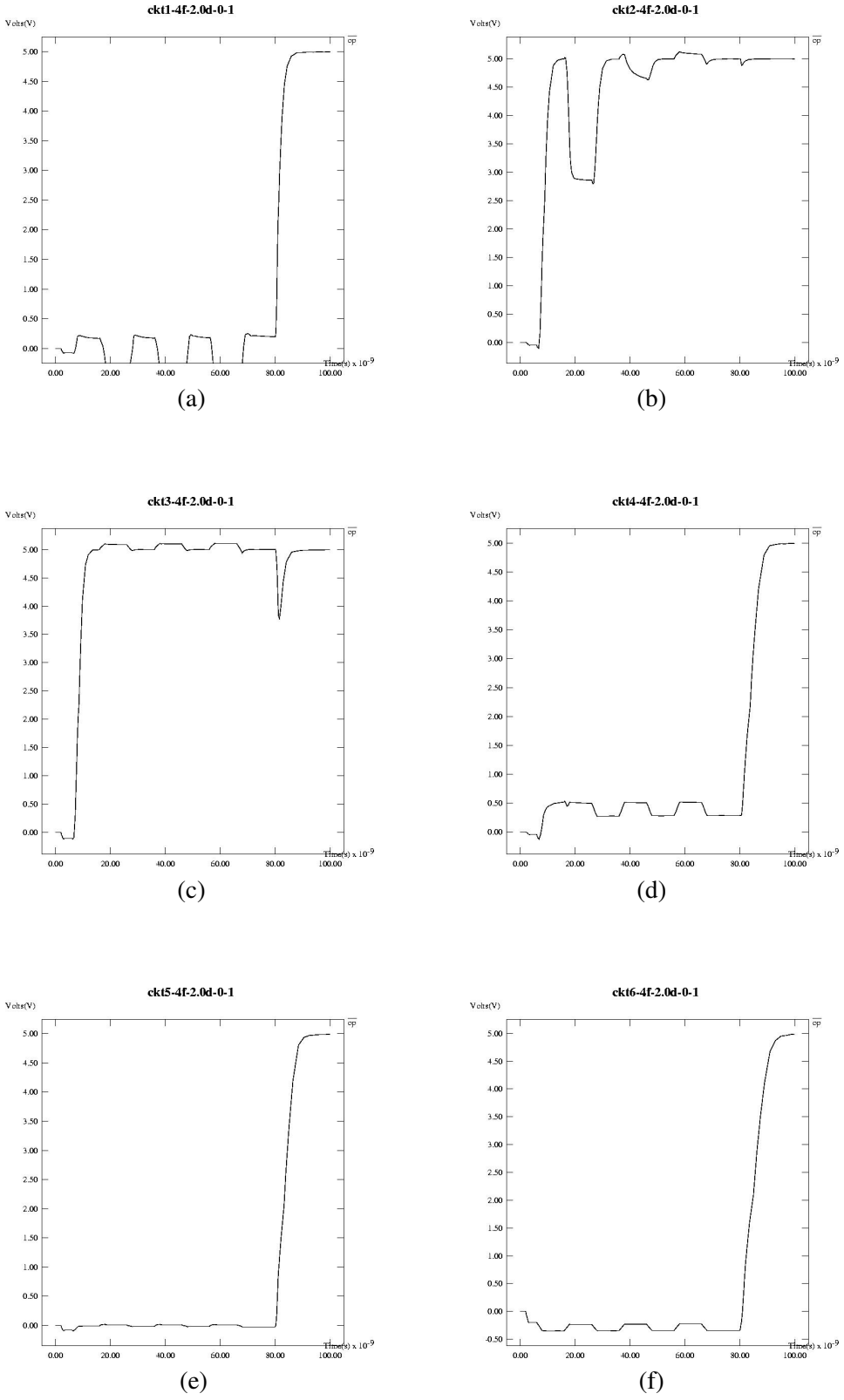



Figure 3.12: Transition 0 – 1: (a) ckt1 (b) ckt2 (c) ckt3 (d) ckt4 (e) ckt5 (f) ckt6

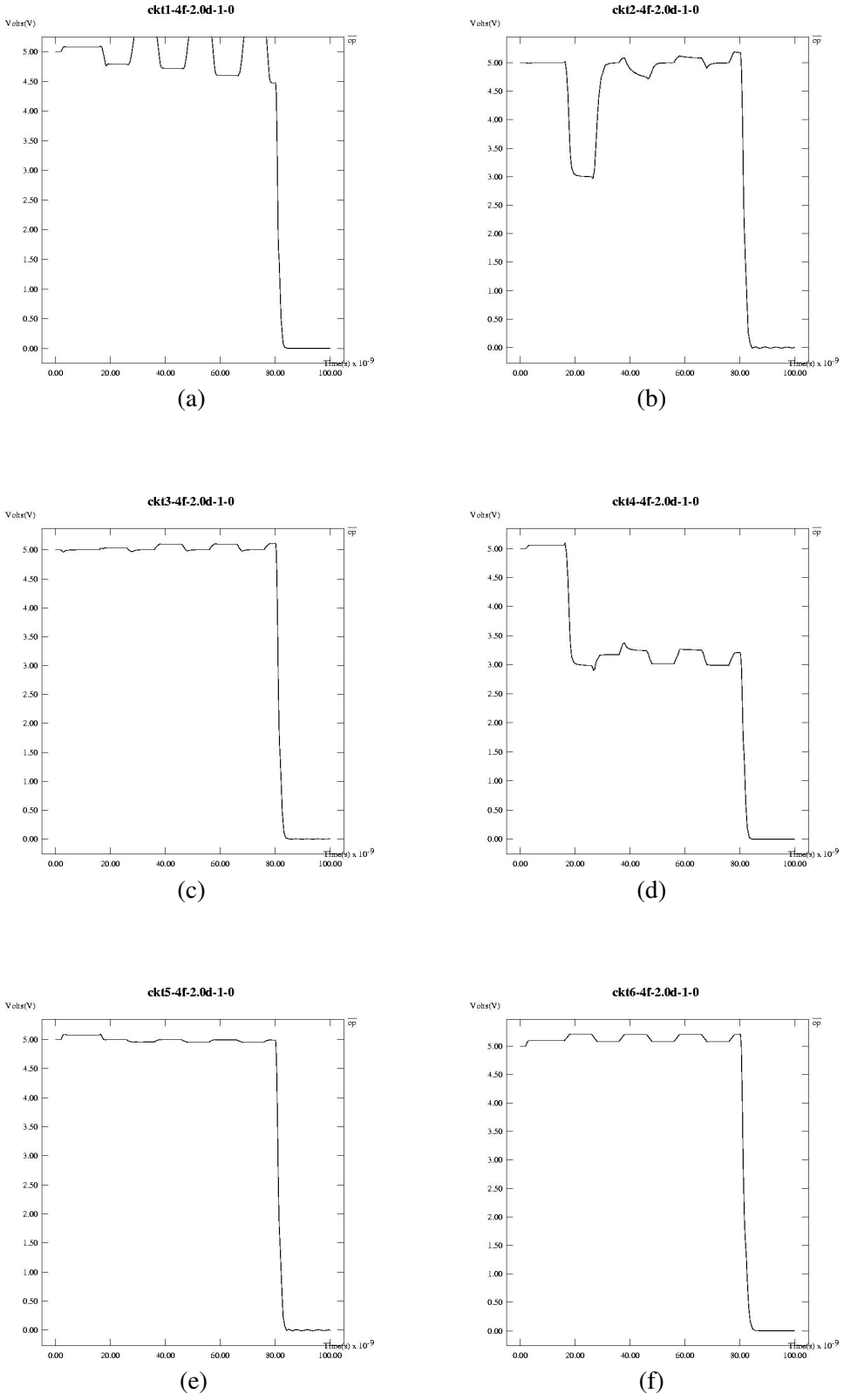



Figure 3.13: Transition $1-0$: (a) ckt1 (b) ckt2 (c) ckt3 (d) ckt4 (e) ckt5 (f) ckt6

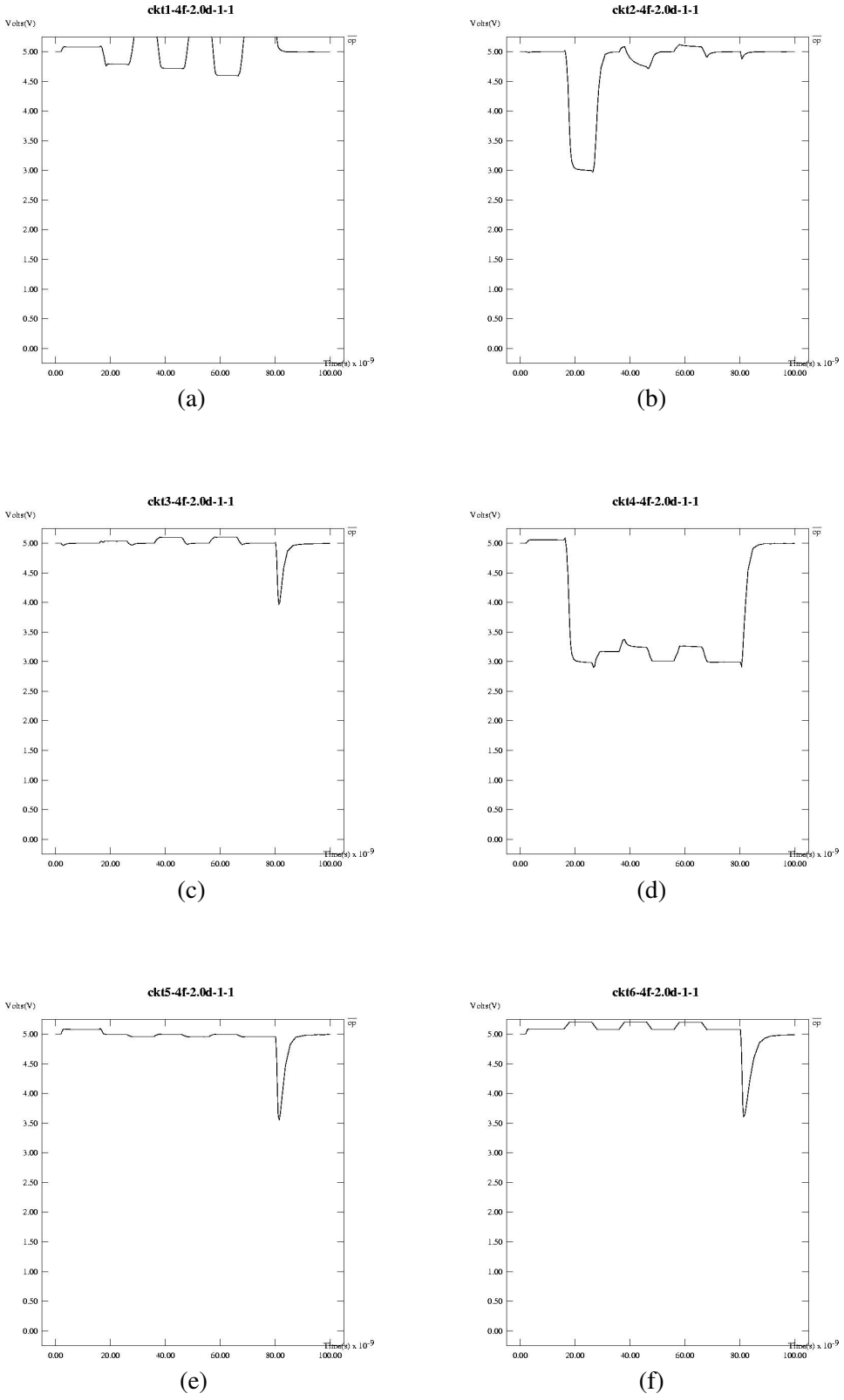



Figure 3.14: Transition 1 – 1: (a) ckt1 (b) ckt2 (c) ckt3 (d) ckt4 (e) ckt5 (f) ckt6

# Chapter 4

# Design and Analysis of Delay Elements: Their Architecture and Usage

## 4.1 Introduction

A delay element is a circuit that produces an output waveform similar to its input waveform, only delayed by a certain amount of time. Delay elements find wide use in digital systems [47]. Asynchronous or self-timed designs, in which the global clock is eliminated, make extensive use of delay elements [48]. Most asynchronous cells need to generate a completion signal to indicate that their outputs have been evaluated. A delay element can provide this as long as its delay amount is larger than the worst-case delay of the cell [16]. For such structures as self-timed multipliers, delay elements are needed on the micropipeline [48]. Even circuits that perform complex mathematical calculations like the discrete cosine transform require delay elements in their architecture [49]. Yet another use of delay elements is in glitch minimization in combinational circuits, wherein delay elements enable triggering of potentially glitchy gates at suitable times [27]. Finally, delay elements

are used for phase modulation in delay locked loops and phase locked loops [43, 44, 45, 46].

A variety of delay elements have been proposed in previous work. Inverter chain and RC delay methods have been most commonly employed [43, 44, 45, 46]. Differential stage chains have also been used, but require extra circuitry because they have complementary inputs and outputs [16]. A delay element in the form of a CMOS thyristor has been proposed in [47]. Finally, a voltage-controlled delay element in which the amount of delay can be altered by changing a control voltage is described in [48]. In this chapter, we compare a number of previously used delay elements. These are the transmission gate, cascaded inverters, voltage-controlled delay element [48] and CMOS thyristor [47]. Based on these architectures, we propose four new delay elements, two of which are based on the voltage controlled principle. The other two delay elements utilize a Schmitt trigger in the output stage. These new architectures are compared with the other previously proposed elements. Each delay element has its own advantages and disadvantages. Comparison results presented in this chapter should prove useful to designers in selecting the best delay element for their application.

In all our simulations to compare delay elements, we fixed the fan-in and fan-out of a delay element to be a single minimum-sized inverter. These inverters are provided with their own power supply, separate from the one connected to the delay element. The $W_p/W_n$ ratio for the delay elements is set at 3, so that the rise and fall times are similar. To study both rising and falling transitions, each delay element is provided two square pulse inputs, each with width 10ns. All calculations were made from the second waveform, since the first waveform just initializes the circuit. The power is determined using a similar analysis. The power due to two square 10ns input waveforms is calculated, then the power due to a single square 10ns waveform is recorded. The difference between the two results is the evaluation power of the delay element, since this calculation removes the power

due to the initialization waveform. Four main parameters are used to compare the various delay elements. These are amount of delay, signal integrity, power consumption, and active area (i. e., the size of the circumscribing rectangle). Delay is measured as the average of the time difference between input and output reaching 50% of $V_{DD}$ for rise and fall transitions. Signal integrity is measured as the average of the rise and fall times, which are defined as the time for the output to transition between 10% and 90% of $V_{DD}$. Layouts were done in MAGIC in 0.25$\mu$m technology ($\lambda = 0.12\mu$m), $V_{DD} = 2.5$V, and simulations and power measurements [16] were performed in SPICE 3.

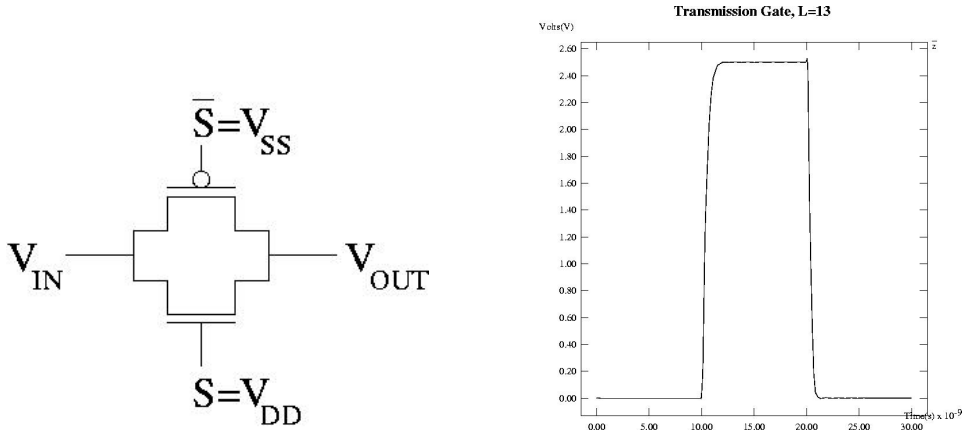
The remainder of the chapter is organized as follows. In Sections 4.2 through 4.5, we discuss and analyze in turn the delay elements studied. In each of these sections, we provide a description of the delay element followed by an analysis of its delay, signal integrity, power consumption, and area, and then discuss any other salient features it may have. For all delay elements, delay may be increased by increasing the fan-out or by decreasing the supply voltage $V_{DD}$. Therefore, in our analysis of propagation delays of delay elements, we only point out other parameters that influence the delay. Next in Section 4.6, we compare the various delay elements and present simulation results for them. Finally, we conclude in Section 4.7.

## 4.2 Transmission Gate Based

### 4.2.1 Transmission Gate

#### 4.2.1.1 Description

A transmission gate is a bidirectional switch consisting of a parallel connection of an NMOS and a PMOS transistor that are controlled by complementary control signals as shown in Figure 4.1(a). The NMOS and PMOS transistors pass a logic 0 and 1, respectively, without degradation. By




(a) (b)

Figure 4.1: Transmission gate (a) schematic (b) output signal

keeping the two transistors always on ($S = 2.5$V and $\bar{S} = 0$V in Figure 4.1(a)), the transmission gate acts as a delay element.

#### 4.2.1.2 Delay

The delay of a transmission gate is effectively determined by the time to charge or discharge a load capacitance $C_L$ at its output through the equivalent resistance $R_{eq}$ of the two transistors connected in parallel. That is,

$$V_{out}(t) = (1 - e^{-t/R_{eq}C_L})V_{DD} \tag{4.1}$$

so that propagation delay is given by [16]:

$$\begin{aligned} t_p = & \quad \ln(2)R_{eq}C_L \\ = & \quad \ln(2)\frac{2V_{DD}}{k_n(V_{DD}-V_{Tn})^2+k_p(V_{DD}-|V_{Tp}|)^2}C_L. \end{aligned} \tag{4.2}$$

Here $V_{Tn}$ and $V_{Tp}$ denote NMOS and PMOS transistor threshold voltages, respectively, and $k_n$ and $k_p$ denote gain factors ($\propto$ ratio of width and length of the channel between source and drain ($W/L$)) of the two transistors. For a given fan-out, delay may be increased compared to that of a minimum-

size transmission gate by increasing $L$ of the transistors, which linearly increases $R_{eq}$. Delay may be decreased by increasing $W$ of the transistors, which decreases $R_{eq}$; however, this effect is limited by the diffusion (or junction) capacitance also increasing, which contributes to more load capacitance $C_L$. A chain of $n$ transmission gates has a delay of [16]:

$$t_p(chain) = \ln(2) R_{eq} C_L \frac{n(n+1)}{2}, \tag{4.3}$$

where $C_L$ is the load capacitance at the output of each transmission gate. Therefore, delay increases quadratically with the number of transmission gates in the chain and hence with area.

#### 4.2.1.3 Power Consumption

There is very little power consumption in a transmission gate since it is not driven by any of the supply rails. Most of the power consumed is that for charging and discharging the output load capacitance from the input.

#### 4.2.1.4 Signal Integrity

Since the transmission gate is not driven by any of the supply rails, the output load capacitance is charged or discharged by the input. This causes its signal integrity to be poor. Since

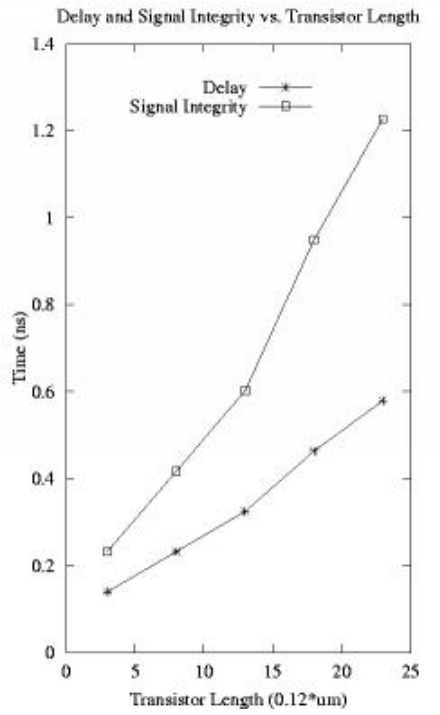
$$V_{out}(t) = (1 - e^{-t/R_{eq}C_L}) V_{DD} \tag{4.4}$$

signal integrity, which is the time for the output to transition between 10% and 90% of $V_{DD}$, is given by:

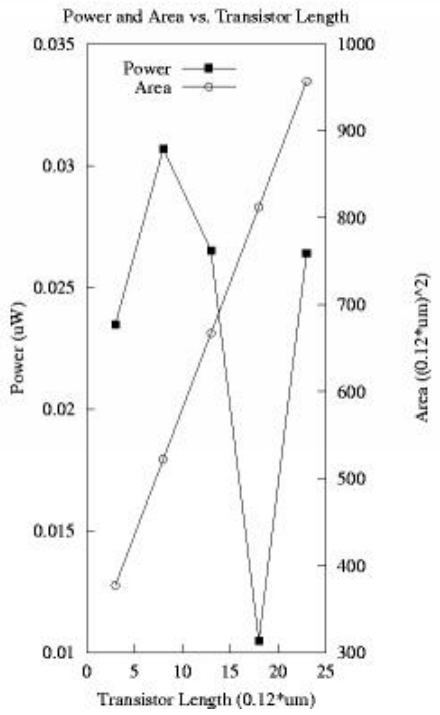
$$t_{r/f} = \ln(9) R_{eq} C_L. \tag{4.5}$$

Similar to propagation delay, for a chain of $n$ transmission gates, the signal integrity increases to:

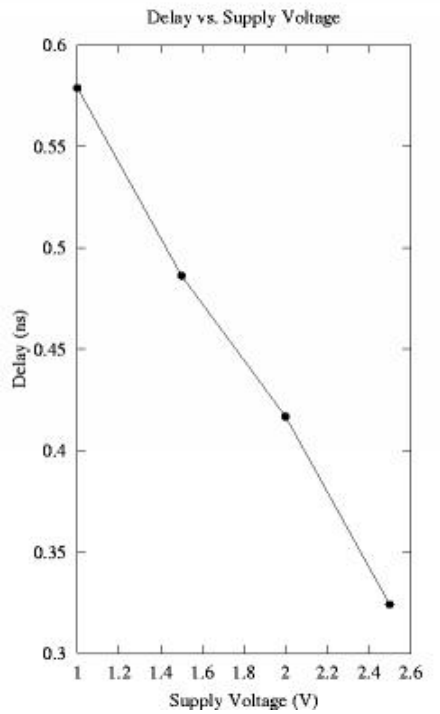
$$t_{r/f}(chain) = \ln(9) R_{eq} C_L \frac{n(n+1)}{2}. \tag{4.6}$$



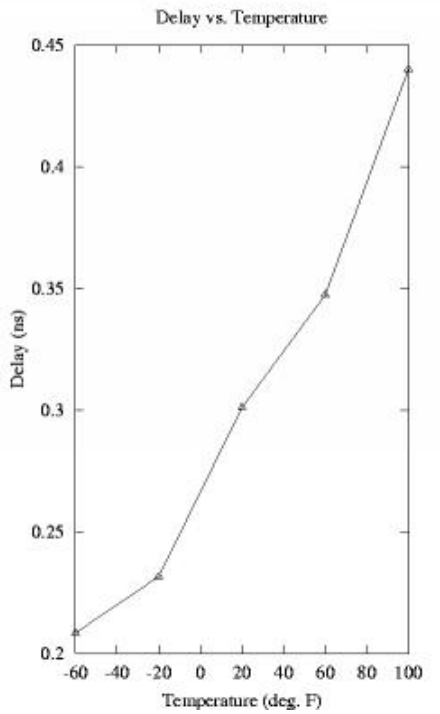

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.2: Transmission gate: graphs of the analysis parameters

Therefore, signal integrity deteriorates quadratically with number of transmission gates in a chain.

#### 4.2.1.5 Area

The transmission gate requires relatively less area with just two transistors, although it does require two complementary control signals.

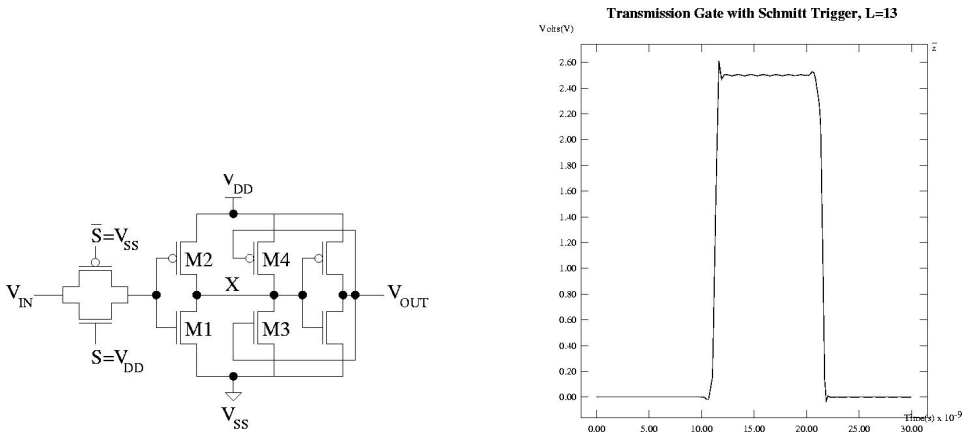
### 4.2.2 Transmission Gate Cascaded with Schmitt Trigger

#### 4.2.2.1 Description

One of the disadvantages of using a transmission gate as a delay element is that the signal integrity of its output waveform is poor. We can overcome this deficiency by placing a Schmitt trigger at the output of the transmission gate. A Schmitt trigger (Figure 4.3) is a circuit that generates a fast, clean output signal from a noisy or slowly varying input signal. This is not only useful for noise suppression, but also the steep output minimizes power consumption due to direct-path currents. If the output of the Schmitt trigger is initially low, then the output will go high only when the rising input signal reaches $V_{M+}$. Similarly, if the output is high, then it will go low only when the falling input signal reaches $V_{M-}$.

#### 4.2.2.2 Delay

The delay of this element can be changed in two ways. The first is by altering the $W/L$ ratio of the transistors of the transmission gate. Decreasing the $W/L$ ratio will increase the gate's delay. The second is by changing the switching thresholds of the Schmitt trigger. If we raise the switching threshold that is active during a rising input transition, $V_{M+}$, the rise delay will increase. Lowering the switching threshold $V_{M-}$ that is active during a falling input transition will have the same effect




(a) (b)

Figure 4.3: Transmission gate with Schmitt trigger (a) schematic (b) output signal

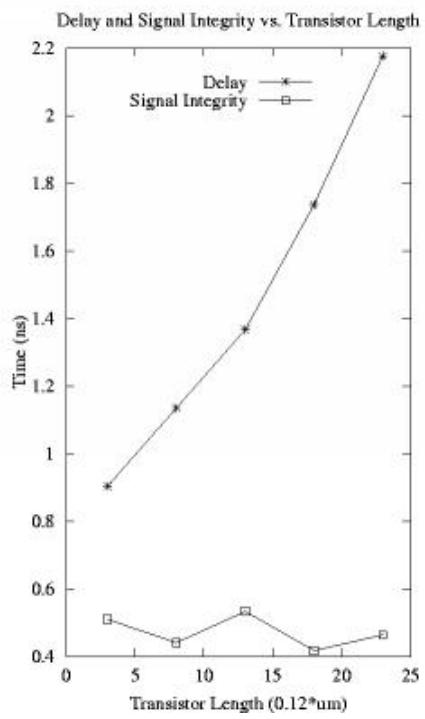
on the fall delay. The switching threshold is determined by the ratio of the gain factors between the NMOS and PMOS transistors, $k_n/k_p$. Increasing this ratio will reduce the switching threshold, and decreasing it will raise the switching threshold. If we examine the Schmitt trigger during a rising transition, we see that the two PMOS transistors $M2$ and $M4$ are activated, as well as the single NMOS transistor $M1$. Therefore, the switching threshold for this transition, $V_{M+}$, is directly proportional to the $k_n/k_p$ ratio of this network:

$$V_{M+} \propto \frac{k_{M1}}{k_{M2}+k_{M4}}, \tag{4.7}$$

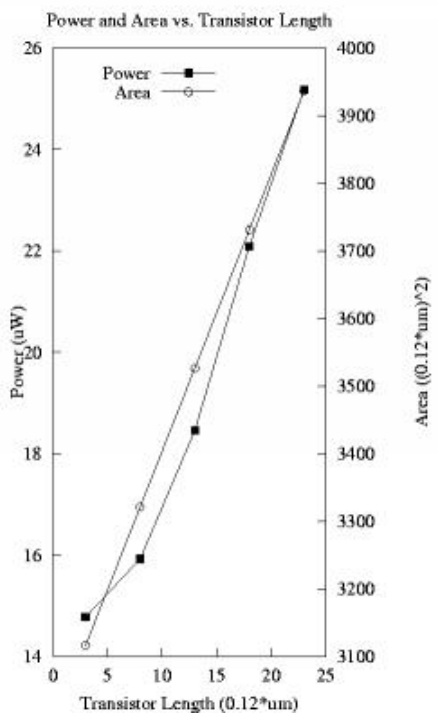
where the $k$'s represent the gain factors of the various transistors and are proportional to their $W/L$ ratios. A similar analysis can be applied to the falling transition, and we can obtain the following expression for $V_{M-}$:

$$V_{M-} \propto \frac{k_{M1}+k_{M3}}{k_{M2}}. \tag{4.8}$$

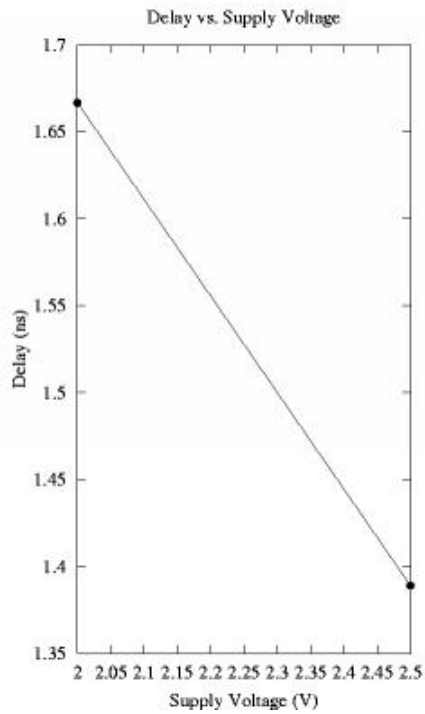
In this case, the NMOS transistors $M1$ and $M3$ are activated, as well as the PMOS device $M2$.



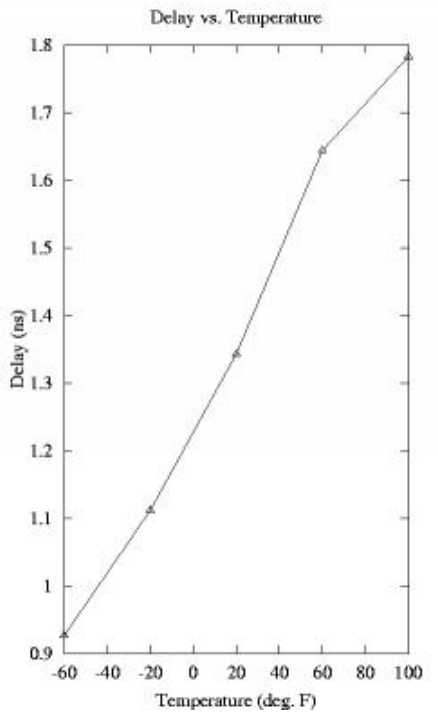

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.4: Transmission gate with Schmitt trigger: graphs of the analysis parameters

#### 4.2.2.3 Signal Integrity

Due to the use of positive feedback, the signal integrity of this delay element is very good. A particularly desirable characteristic is that the signal integrity remains virtually unchanged as the delay value increases.

#### 4.2.2.4 Power Consumption

The power consumed by a transmission gate was discussed in Section 4.2.1.3. Since the structure of a Schmitt trigger is very similar to that of a cascaded inverter, the power consumption analysis is similar.

#### 4.2.2.5 Area

This delay element requires six transistors.

## 4.3 Cascaded Inverter Based

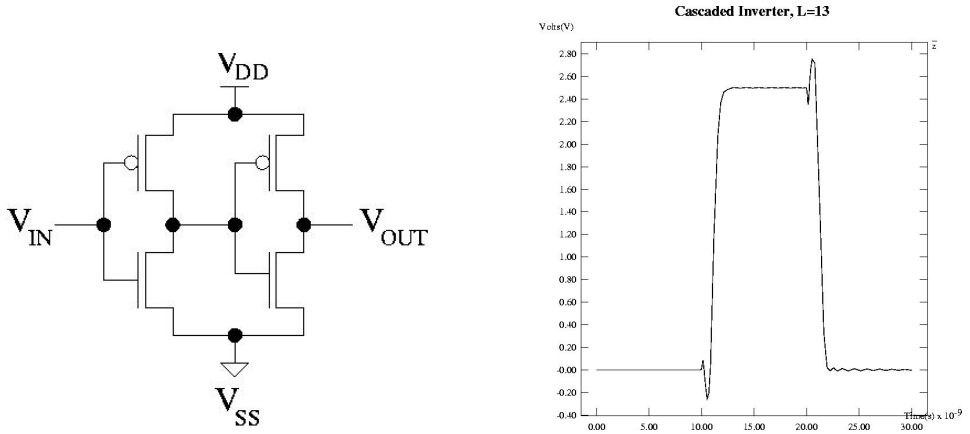
### 4.3.1 Cascaded Inverters

#### 4.3.1.1 Description

A pair of cascaded inverters can also function as a simple delay element that delays the input signal by an amount equal to the combined propagation delays of the two inverters(see Figure 4.5).

#### 4.3.1.2 Delay

The propagation delay of an inverter depends upon the time taken to (dis)charge the load capacitance. An exact computation of this delay is nontrivial because of the nonlinear dependence of




(a) (b)

Figure 4.5: Cascaded inverters (a) schematic (b) output signal

the (dis)charging current on the output voltage. An approximate expression is derived by using an average value of this current equal to the saturation current of the PMOS (NMOS) transistor given by [16]:

$$I_{av} = \frac{k_p}{2}(V_{GS} - |V_{Tp}|)^2 = \frac{k_p}{2}(V_{DD} - |V_{Tp}|)^2 \approx \frac{k_p}{2}V_{DD}^2. \tag{4.9}$$

The above holds since $V_{DD} \gg |V_{Tp}|, V_{Tn}$. Based on this $I_{av}$ value, the propagation delay is as follows [16]:

$$t_p = \frac{1}{2}(t_{pLH} + t_{pHL}) = \frac{C_L}{2V_{DD}}\left(\frac{1}{k_p} + \frac{1}{k_n}\right), \tag{4.10}$$

where $t_{pLH}$ and $t_{pHL}$ denote propagation delays for low to high and high to low output transitions, respectively. The above expression is valid when the input signal makes an abrupt transition from $V_{DD}$ to $V_{SS}$ or vice versa. The effect of a nonzero input rise time $t_r > t_{pHL}$ on propagation delay $t_{pHL}$ is captured by the following equation [16]:

$$t_{pHL(actual)} = \sqrt{t_{pHL(step)}^2 + (t_r/2)^2} \tag{4.11}$$

A similar expression is valid for $t_{pLH}$. For a cascaded chain of inverters, an expression for the overall propagation delay can be obtained as the sum of the individual propagation delays, taking

into account the finite rise and fall times of the inverters. It is clear from the expressions for $t_p$ that it can be increased by decreasing $k_p$ and $k_n$, or, equivalently, the $(W/L)$ ratios of the transistors. The effect of decreasing $W$ to increase delay is offset by the fact that the gate and diffusion capacitances also decrease resulting in less load capacitances at the input and output, respectively. In contrast, the effect of increasing $L$ to increase delay is aided by the gate capacitance also increasing, which contributes to more capacitance at the input.

For small-geometry devices, when $V_{DD} \gg V_T$ and velocity saturation effects are taken into account, the expressions above for $t_p$ are not valid, since in this case $I_{av}$ tends to be proportional to $V_{DD}$ instead of $V_{DD}^2$. Under these conditions, the delay is given by [16]:

$$t_p \approx \frac{C_L}{2}\left(\frac{1}{k_p}+\frac{1}{k_n}\right). \tag{4.12}$$

Hence, in this case delay is independent of $V_{DD}$.

#### 4.3.1.3 Signal Integrity

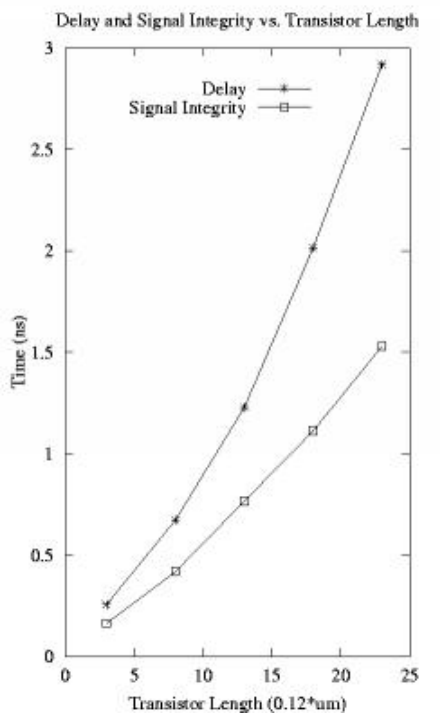
An approximate expression for signal integrity is computed using the average (dis)charging current expression in Equation 4.9 and finding out the time to charge from $0.1V_{DD}$ to $0.9V_{DD}$, or vice versa, as follows:

$$t_{r/f} = \frac{0.9V_{DD}C_L}{|I_{av}|} \approx \frac{0.9C_L}{V_{DD}}\left(\frac{1}{k_p}+\frac{1}{k_n}\right). \tag{4.13}$$

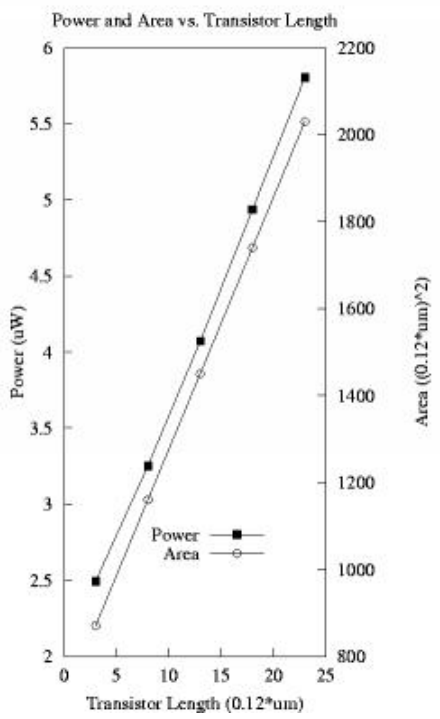
The signal integrity depends upon the same factors that the propagation delay depends upon.

#### 4.3.1.4 Power Consumption

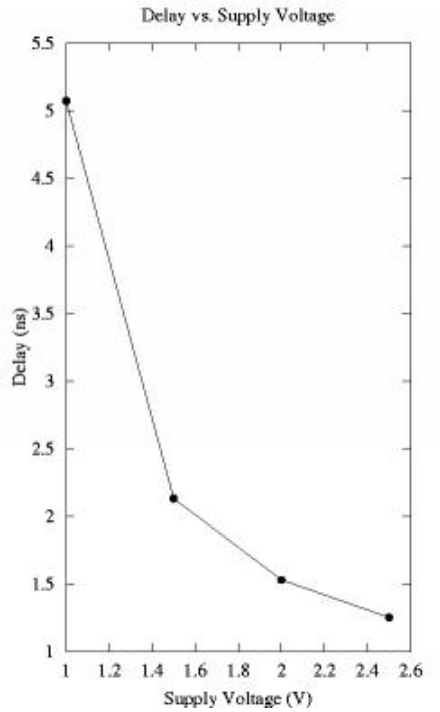
Since the PMOS and NMOS transistors are never on simultaneously in steady-state operation, the static power consumption in an inverter occurs only due to leakage currents and is generally small. Sources of leakage currents are the following [16].



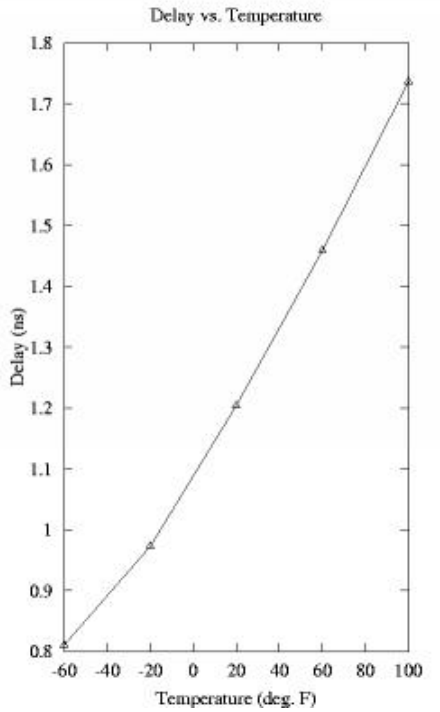

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.6: Cascaded inverters: graphs of the analysis parameters

1. Leakage current flowing through the reverse-biased diode junctions of the transistors located between the source or drain and the substrate; this contribution is, in general, very small and can be ignored. This junction leakage current is caused by thermally generated carriers and it increases exponentially with junction temperature.

2. The more important leakage current is the subthreshold current of the two transistors, which is a drain-source current that flows even when $V_{GS}$ is less than the threshold voltage. The lower the threshold voltage, the greater this current and power consumption.

The static power consumption due to both the above sources is given by:

$$P_{stat} = I_{leak} V_{DD}. \tag{4.14}$$

Most of the power is consumed during switching. This dynamic power consists of two components. The major component is due to charging and discharging of the load capacitance. During a low-to-high transition, a certain amount of power is consumed, half of which is dissipated in the PMOS transistor and the other half stored on the load capacitance. During a high-to-low transition, the stored energy in the load capacitance is discharged and dissipated in the NMOS transistor. If an inverter switches on and off $f$ times per second, this capacitive power is given by [16]:

$$P_{cap} = C_L V_D^2 f. \tag{4.15}$$

Hence this power depends quadratically upon $V_{DD}$ and linearly upon $C_L$ and $f$.

The second component of dynamic power arises due to nonzero rise and fall times of the input signal, which results in both NMOS and PMOS transistors being on (a short circuit) briefly. Let the peak current flow during this period be $I_{peak}$, then the short-circuit power consumption is [16]:

$$P_{sc} = \frac{t_r + t_f}{2} V_{DD} I_{peak} f. \tag{4.16}$$

This means that short-circuit power consumption depends linearly upon the input signal integrity and the supply voltage. It has been shown that in a chain of inverters with equal input rise and fall times, the short-circuit dissipation is less than 20% of the dynamic power dissipation [16].

The total power is therefore given by [16]:

$$\begin{aligned} P_{tot} = & \quad P_{stat} + P_{cap} + P_{sc} \\ = & \quad I_{leak}V_{DD} + C_L V_D^2 f + \tfrac{t_r+t_f}{2} V_{DD} I_{peak} f. \end{aligned} \tag{4.17}$$

Of this, the power consumed during capacitive charging ($P_{cap}$) is the dominant one, followed by $P_{sc}$, and then $P_{stat}$.

#### 4.3.1.5 Area

A cascaded inverter requires 4 transistors.

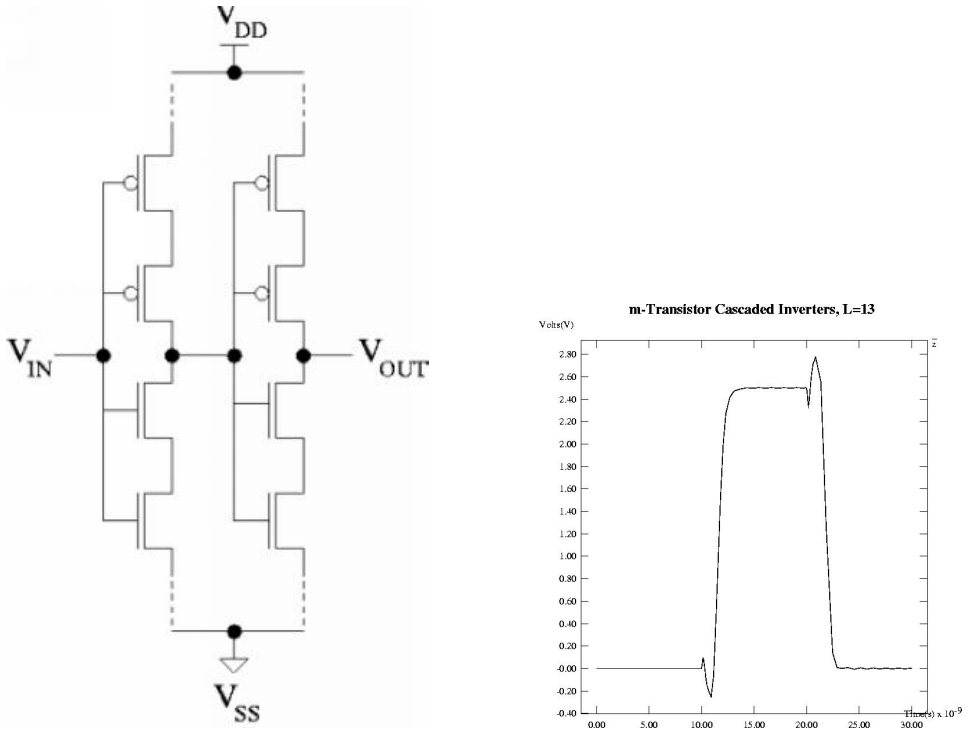
### 4.3.2 m-Transistor Cascaded Inverters

#### 4.3.2.1 Description

This delay element is a modified version of the simple cascaded inverter configuration. It has $m$ series-connected NMOS transistors and $m$ series-connected PMOS transistors in its pull-down and pull-up networks, respectively. The gates to all of these transistors are connected to the input.

#### 4.3.2.2 Delay

Increasing the fan-in ($m$) not only increases the effective (dis)charging resistance, but also increases the gate and diffusion capacitances, which contribute to more capacitance at the input and output, respectively. Therefore, the propagation delay (proportional to $R$ and $C$) depends quadratically on fan-in or $m$ [16]. Further, it increases the delay of the fan-in gate by presenting it a larger load



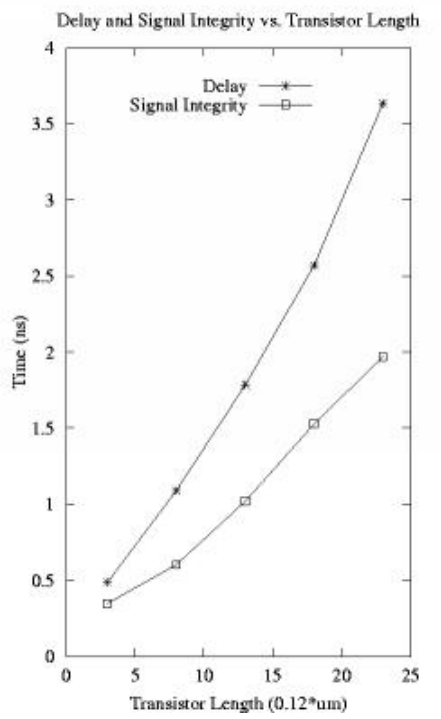

(a) (b)

Figure 4.7: m-Transistor cascaded inverters (a) schematic (b) output signal

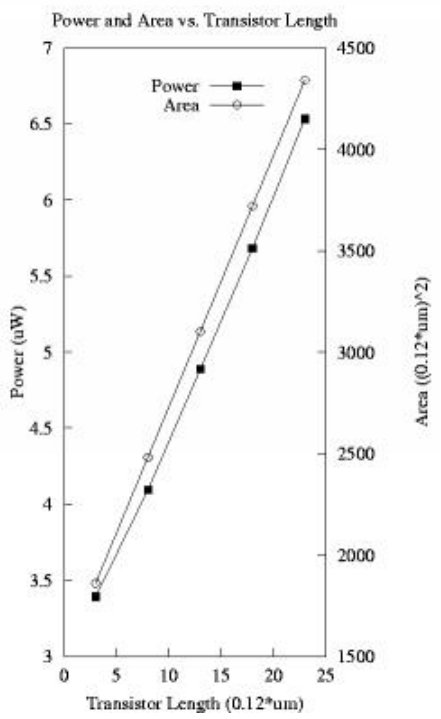
capacitance. Consequently, more delay per unit area may be obtained by using a generalized inverter with *m* series-connected transistors than by using a chain of *m* simple inverters.

#### 4.3.2.3 Signal Integrity

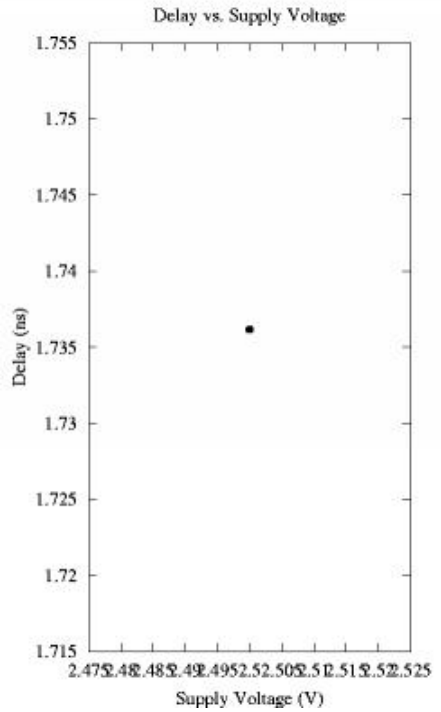
The increased fan-in (*m*) has a similar effect on the signal integrity as it does on the delay. That is, the increased effective (dis)charging resistance and increased gate and diffusion capacitances lead to a larger signal integrity value. The increases are roughly the same for both the rise (low-to-high) and fall (high-to-low) times.



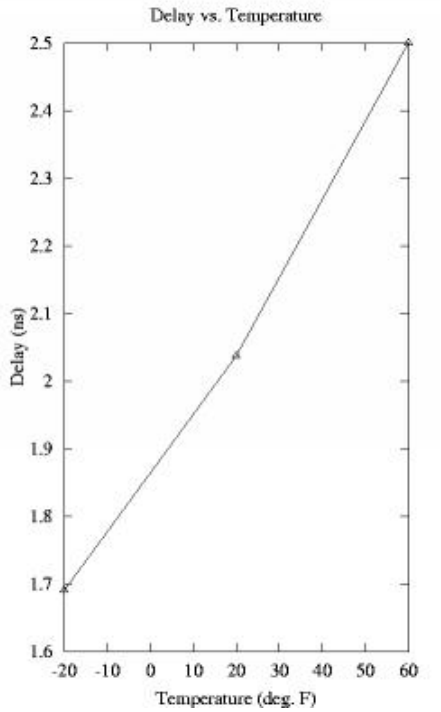

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.8: m-Transistor cascaded inverters: graphs of the analysis parameters

#### 4.3.2.4 Power Consumption

This delay element consumes slightly more power than the cascaded inverter element. Since there are more transistors in the pull-up and pull-down networks, more energy will be dissipated in these additional devices.

#### 4.3.2.5 Area

An m-transistor cascaded inverter requires $4m$ transistors.

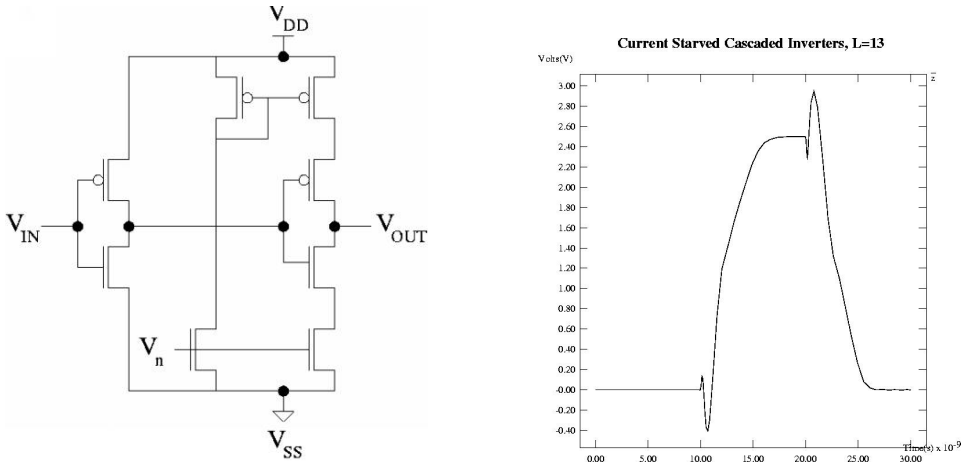
### 4.3.3 Current Starved Cascaded Inverters

#### 4.3.3.1 Description

A larger fan-in and a clever configuration of control transistors is utilized in the delay element proposed in [50]. The basic architecture is similar to the cascaded inverters. However, two additional PMOS and NMOS devices are added to extend the delay value. The gate voltage $V_n$ is applied to the additional NMOS devices, and they control the maximum current available to the inverter. The gates of the additional PMOS devices are connected to the source of the first additional NMOS device.

#### 4.3.3.2 Delay

The delay that can be achieved with this delay element is considerably greater than that achieved with the cascaded inverter or m-transistor cascaded inverter configurations. This is primarily because of its control transistors that limit the amount of current that can (dis)charge the load capacitor $C_L$. The extended (dis)charging time is what gives this element an impressive value of delay.




(a) (b)

Figure 4.9: Current starved cascaded inverters (a) schematic (b) output signal

#### 4.3.3.3 Signal Integrity

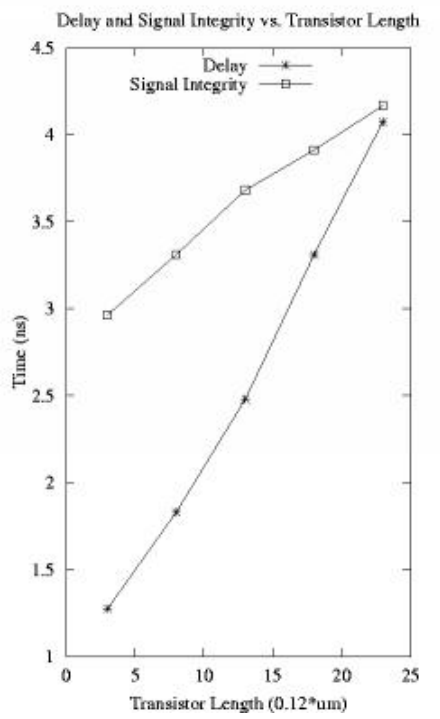
One of the drawbacks of this circuit is that its signal integrity value is very poor, due to the same property that makes its delay very good: the current-limiting capability of its control transistors. An extended (dis)charging time on the load capacitor means that the output signal slopes will be much less steep.

#### 4.3.3.4 Power Consumption

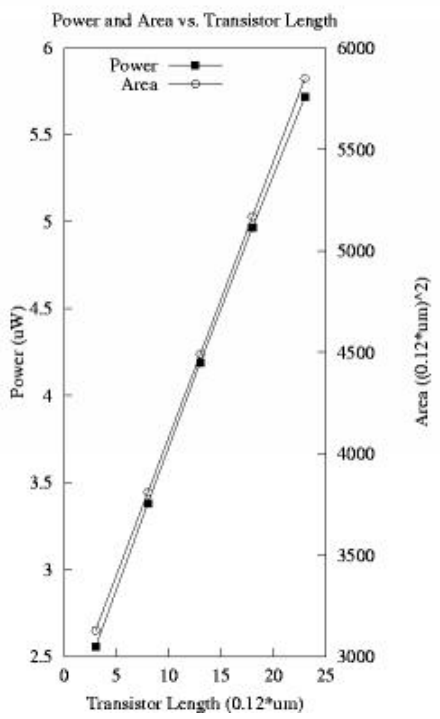
Due to their similar architectures, the current starved cascaded inverter has a similar rate of power consumption as the cascaded inverter. The extra fan-in of the current-limiting transistors will increase this parameter slightly.

#### 4.3.3.5 Area

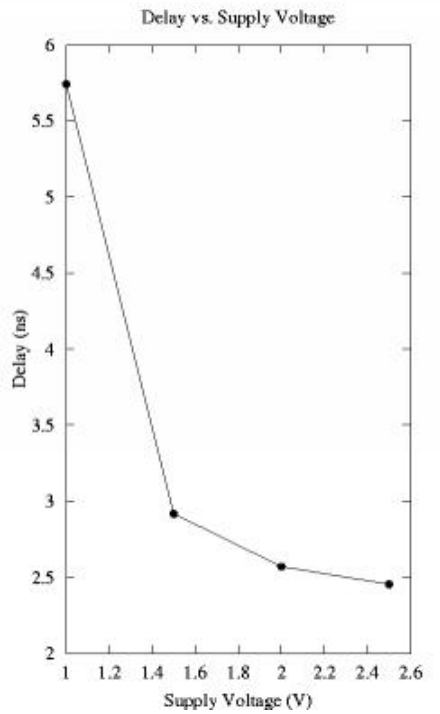
A current starved cascaded inverter requires 8 transistors.



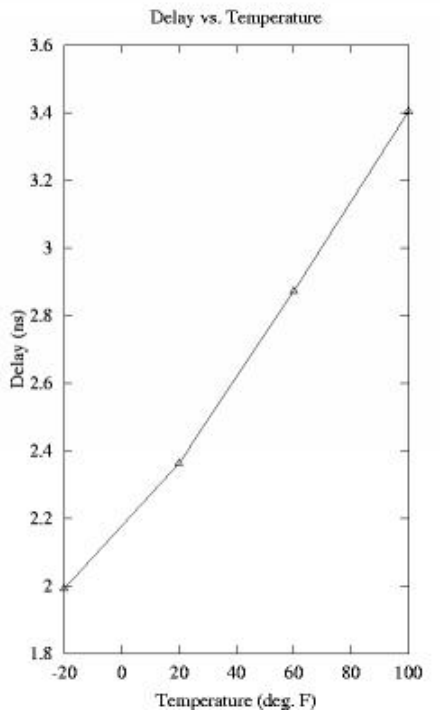

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$

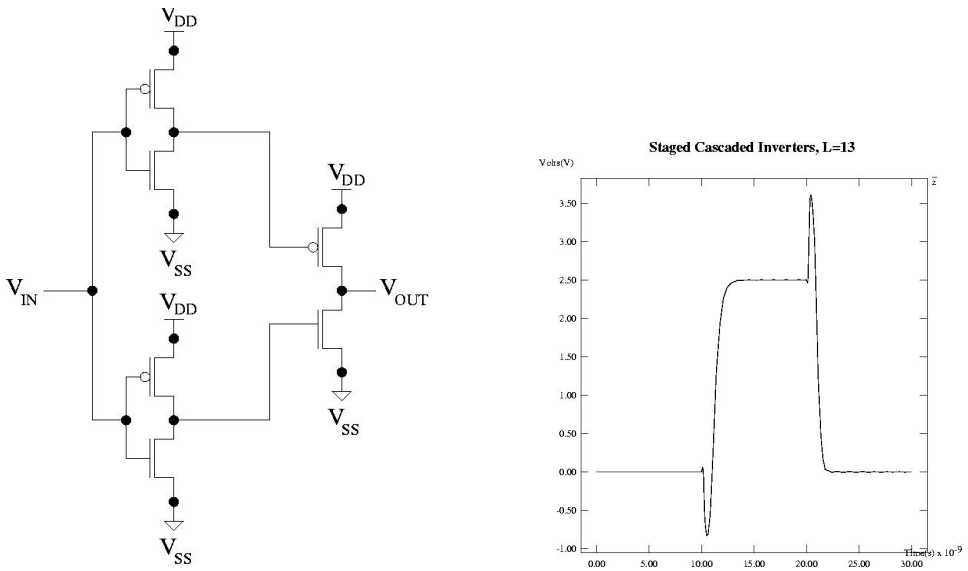



(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.10: Current starved cascaded inverters: graphs of the analysis parameters




(a) (b)

Figure 4.11: Staged cascaded inverters (a) schematic (b) output signal

### 4.3.4 Staged Cascaded Inverters

#### 4.3.4.1 Description

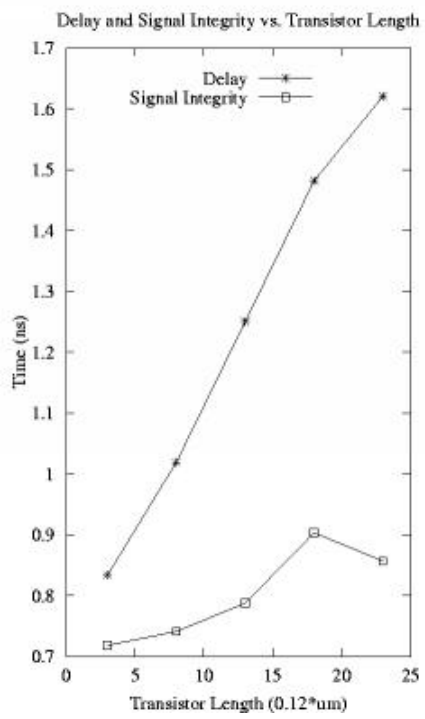
This delay element consists of an arrangement of three inverters, in two stages. The first stage consists of two inverters, where each inverter output controls a transistor on the second stage's inverter. The key intuition in this design is that the two large transistors in the output stage are never on at the same time, thus eliminating short circuit power dissipation.

#### 4.3.4.2 Delay

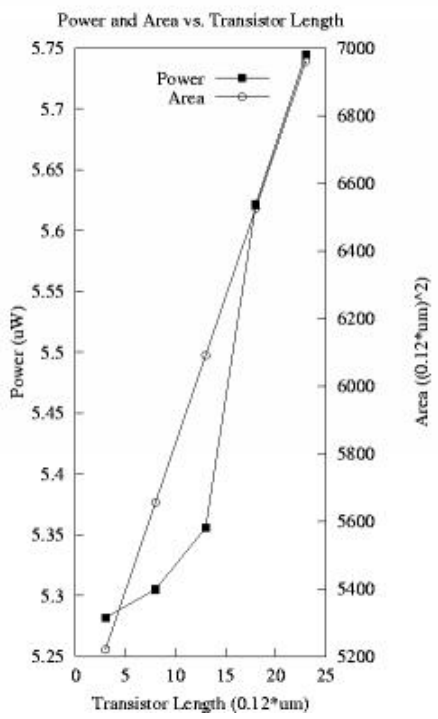
The delay is obtained by dimensioning the resistances of the two inverters in the first stage.

#### 4.3.4.3 Signal Integrity

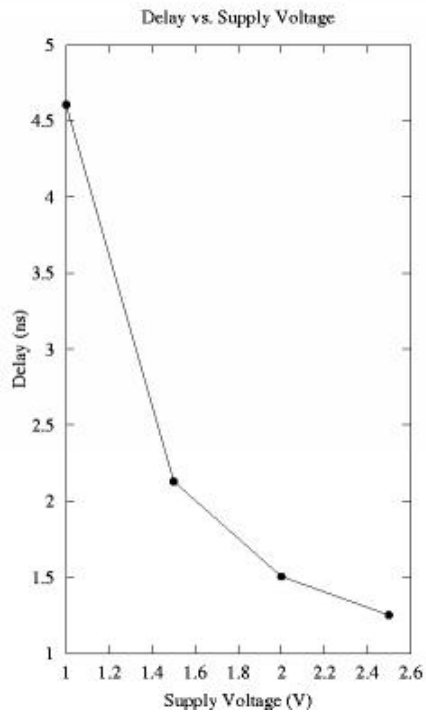
The transition that controls the output edge is always produced by the transistor in series with the resistance and it can be slowed down using large values of $R1$ and $R2$. The penalty is in less sharp



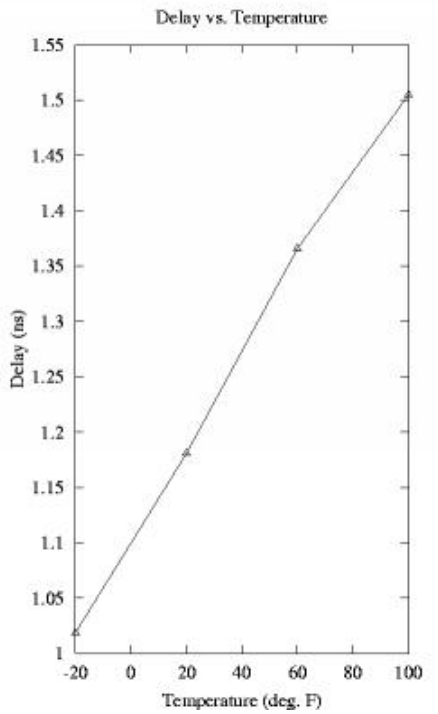

(a)Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.12: Staged cascaded inverters: graphs of the analysis parameters

output edges (although the gain of the output inverter mitigates this effect), and, when both output transistors are off, the input line is susceptible to cross-talk. Both of these effects are greatly reduced by adding another output stage (i. e., two inverters).

#### 4.3.4.4 Power Consumption

The power consumption analysis for this delay element is very similar to one for the cascaded inverters. However, this delay element architecture virtually eliminates the component due to short circuit power consumption. Therefore, it only consumes static and capacitive power consumption.

$$\begin{aligned} P_{tot} = &\quad P_{stat} + P_{cap} \\ P_{tot} = &\quad I_{leak} V_{DD} + C_L V_{DD}^2 f \end{aligned} \tag{4.18}$$

#### 4.3.4.5 Area

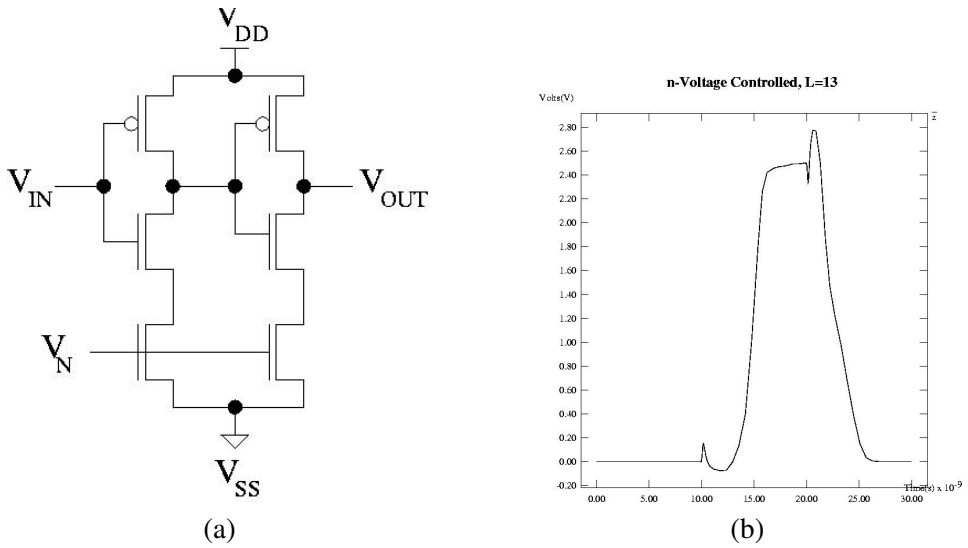
A generalized staged cascaded inverter requires $6m$ transistors.

## 4.4 Voltage Controlled Based

### 4.4.1 n-Voltage Controlled

#### 4.4.1.1 Description

An n-voltage controlled delay element, proposed in [48], is shown in Figure 4.13. It consists of a cascaded inverter pair with an dditional series-connected NMOS transistor in the pull-down of each inverter controlled by a global control voltage $V_n$, the varying of which changes the delay of this delay element. This was used in [48] for building asynchronous systems based on the micropipelining technique. The delay element is used to match with the propagation delay of a pipeline stage




(a) (b)

Figure 4.13: n-Voltage controlled (a) schematic (b) output signal

for completion signal generation. Variations in process and operating conditions make it difficult to accurately estimate the delay value of a delay element. Having an adjustable delay enables fine tuning after fabrication to optimize speed by matching the delay of the delay element with the delay of a pipeline stage [48]. Note that this delay element may be generalized in a manner similar to the m-transistor cascaded inverter case (Section 4.3.2) by having m-series connected NMOS and PMOS transistors per inverter in addition to the controlled NMOS transistor.

#### 4.4.1.2 Delay

There are three ways to change the delay of this element. The first is by changing the sizes of the transistors and the second by changing the fan-in *m*-similar to the m-transistor cascaded inverter case (Section 4.3.2). The third way is to change the control voltage $V_n$. Note that during any transition of the input $V_{in}$, one of the two inverters of the delay element will be discharging through a controlled transistor, and the other charging normally through a PMOS transistor. Therefore, the overall delay is the sum of the normal inverter delay (see Equation 4.10) and a controlled inverter

delay. The latter delay is inversely proportional to the discharging drain current $I_D$ through the control transistor. Approximating the average discharging current by the saturation current of the controlled NMOS transistor:

$$I_a v = \frac{k_n}{2}(V_{GS} - V_{Tn})^2 \approx \frac{k_n}{2} V_{GS}^2 = \frac{k_n}{2} V_n^2 \tag{4.19}$$

the propagation of this delay element becomes:

$$t_p = \frac{1}{2}(t_{pLH} + t_{pHL}) = \frac{C_L}{2}\left(\frac{1}{k_p V_{DD}} + \frac{V_{DD}}{k_n V_n^2}\right) \tag{4.20}$$

This shows that $t_p$ is inversely proportional to $V_n^2$. In [48], the value of the delay assumed is that corresponding to $V_n = 3.5$V, and then $V_n$ is varied after fabrication to fine tune the delay. A delay variation of up to 30% was obtained by changing $V_n$ in 1.2$\mu$m CMOS technology in [48].

#### 4.4.1.3 Signal Integrity

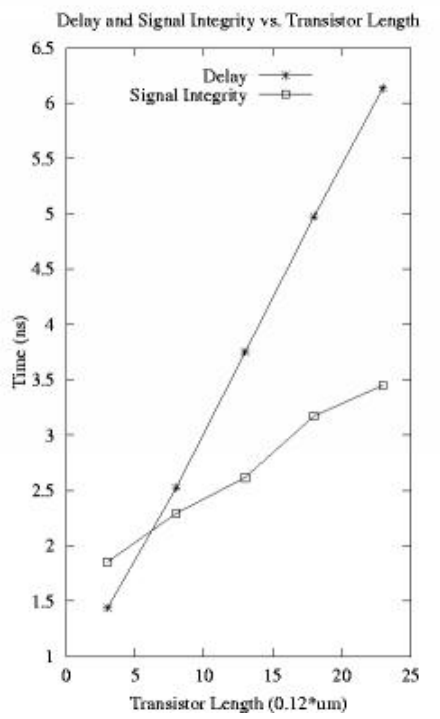
The signal integrity is computed in a manner similar to that for the cascaded inverter case. The only difference is that the rise time is similar to the cascaded inverter, but the fall time depends upon $V_n$. Therefore:

$$t_{r/f} = 0.9 C_L \left(\frac{1}{k_p V_{DD}} + \frac{V_{DD}}{k_n V_n^2}\right). \tag{4.21}$$

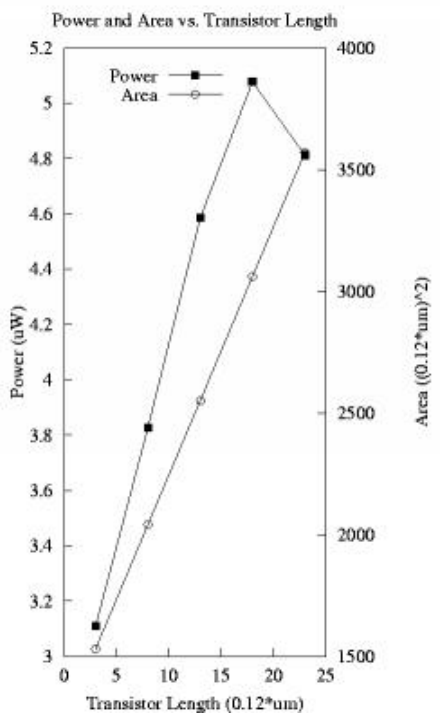
The signal integrity depends upon the same factors that the propagation delay depends upon. It will be worse than that for the cascaded inverter case because of the additional resistance and diffusion capacitance of the controlled transistors.

#### 4.4.1.4 Power Consumption

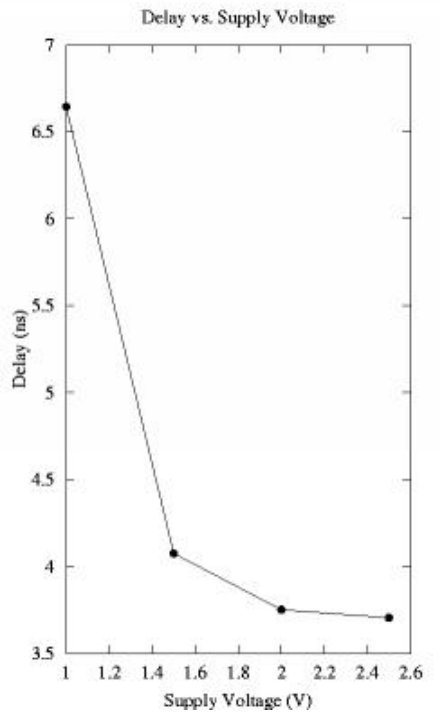
The power consumption is computed similar to the cascaded inverter case. It will be slightly more because of the additional diffusion capacitance of the controlled transistors that contributes to the



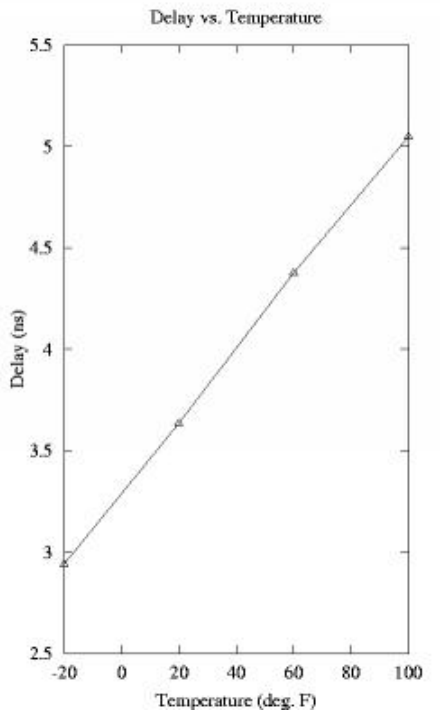

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$

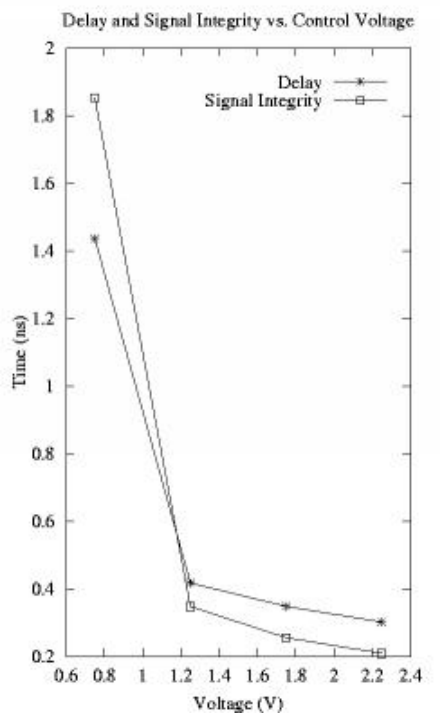



(c) Delay vs. supply voltage($V_{DD}$)

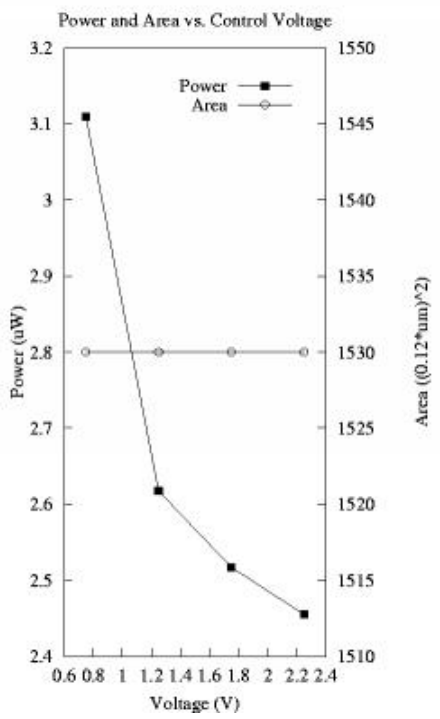



(d) Delay vs. temperature(°F)

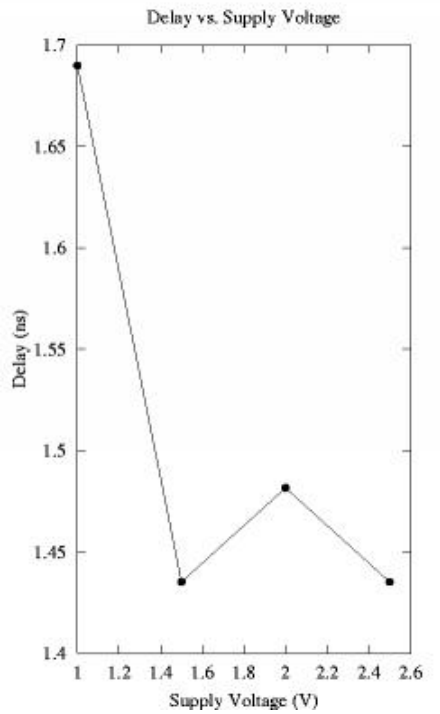
Figure 4.14: n-Voltage controlled ($V_n = 0.75$V, alter length $L$): graphs of the analysis parameters



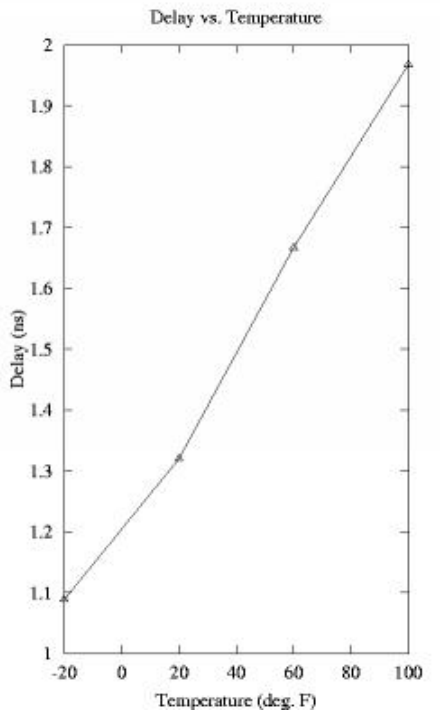

(a) Delay and signal integrity vs. $V_n$




(b) Power and area vs. $V_n$

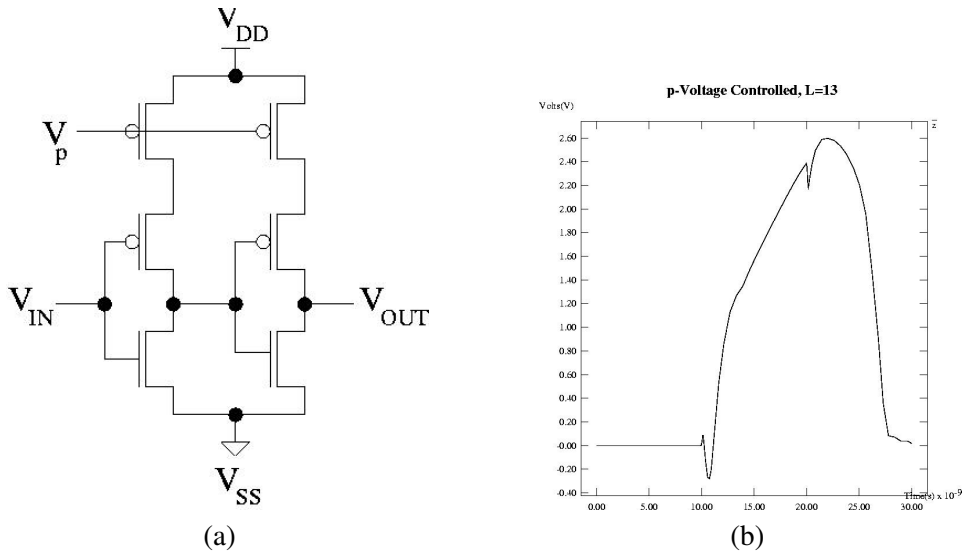



(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.15: n-Voltage controlled (alter voltage $V_n$): graphs of the analysis parameters




(a) (b)

Figure 4.16: p-Voltage controlled (a) schematic (b) output signal

total load capacitance.

#### 4.4.1.5 Area

A generalized voltage-controlled delay element requires $4m+2$ transistors, two more transistors compared to cascaded inverters.

### 4.4.2 p-Voltage Controlled

#### 4.4.2.1 Description

The voltage-controlled technique can be applied to a delay element in several different manners. This delay element uses a cascaded inverter pair with an additional series-connected PMOS transistor in the pull-up network of each inverter. The gates of these additional transistors are controlled by a control voltage $V_p$, and this value can be varied to control the amount of delay.

#### 4.4.2.2 Delay

Changing the delay of this element can be accomplished by altering the sizes of the transistors, or increasing the fan-in $m$ of the gate (m-transistor cascaded inverter case, Section 4.3.2). Another way is to change the gate voltage of the control transistor, $V_p$. In this case, the delay can be analyzed in a manner similar to the n-voltage controlled element. During an input transition of $V_{in}$, one inverter will charge its load capacitance through a controlled PMOS transistor, and the other will discharge regularly through an NMOS transistor. The overall delay is the sum of a controlled inverter delay and a normal inverter delay. The former delay is inversely proportional to the charging drain current $I_D$ through the control transistor. Approximating the average charging current by the saturation current of the controlled PMOS transistor:

$$I_a v = \frac{k_p}{2}(V_{GS} - V_{Tn})^2 \approx \frac{k_p}{2} V_{GS}^2 = \frac{k_p}{2} V_p^2 \tag{4.22}$$

The propagation delay becomes:

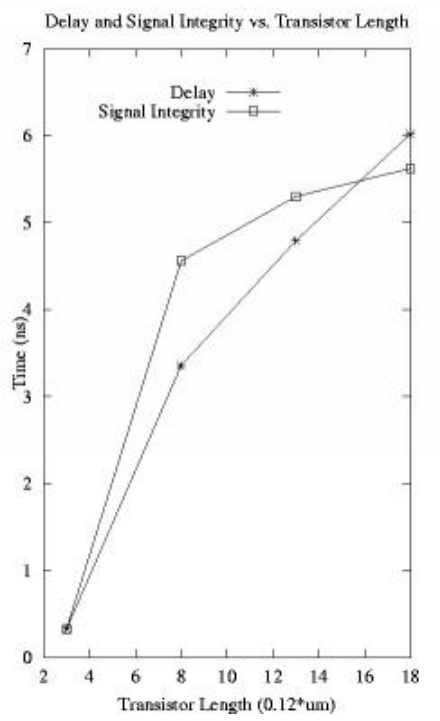
$$t_p = \frac{1}{2}(t_{pLH} + t_{pHL}) = \frac{C_L}{2}\left(\frac{V_{DD}}{k_p V_p^2} + \frac{1}{k_n V_{DD}}\right) \tag{4.23}$$

Hence, $t_p$ is proportional to $V_p^2$.

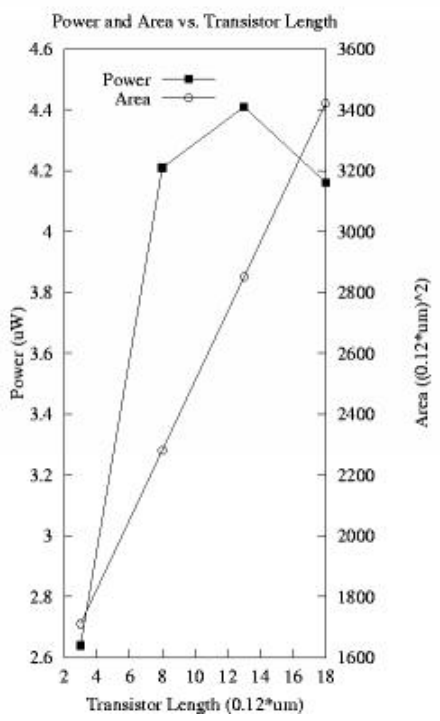
#### 4.4.2.3 Signal Integrity

The signal integrity analysis is very similar to the cascaded inverter delay element. In the p-voltage controlled element, the rise time depends on the value of $V_p$, and the fall time is similar to that of a regular inverter.

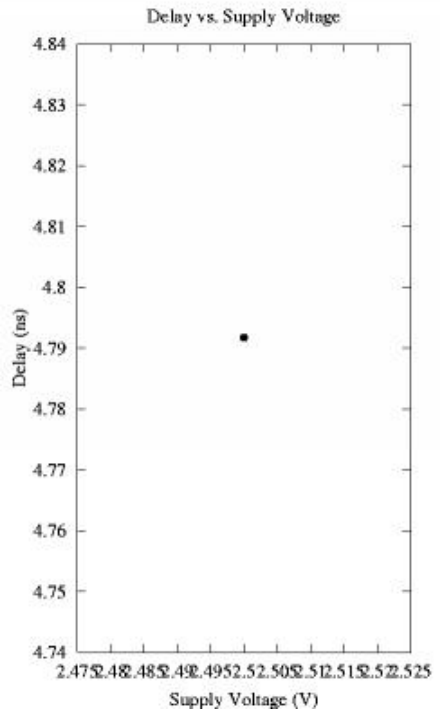
$$t_{r/f} = 0.9 C_L \left(\frac{V_{DD}}{k_p V_p^2} + \frac{1}{k_n V_{DD}}\right). \tag{4.24}$$



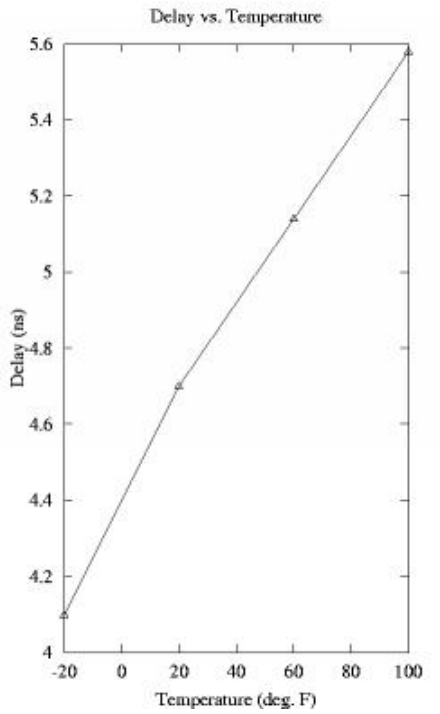

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$

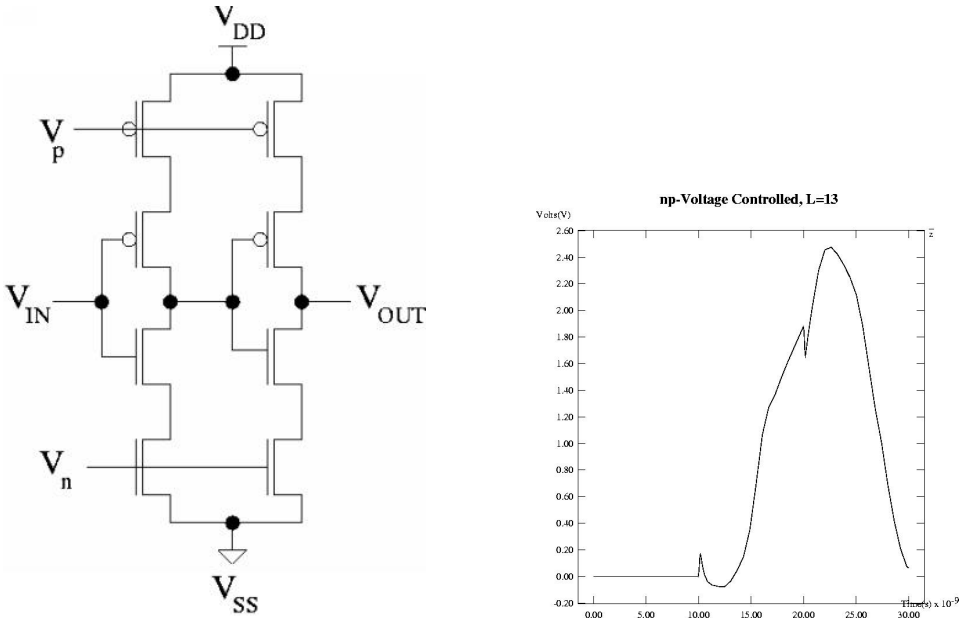



(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.17: p-Voltage controlled ($V_p = 1.75$V, alter length $L$)




(a) (b)

Figure 4.18: np-Voltage controlled (a) schematic (b) output signal

#### 4.4.2.4 Power Consumption

The power consumption is computed similar to the cascaded inverter case. It will be slightly more because of the additional diffusion capacitance of the controlled transistors that contributes to the total load capacitance.

#### 4.4.2.5 Area

Similar to the n-voltage controlled delay element, this one requires $4m+2$ transistors.

### 4.4.3 np-Voltage Controlled

#### 4.4.3.1 Description

This delay element is a combination of the n-voltage controlled and p-voltage controlled configurations. It employs control transistors in both the pull-up and pull-down networks.

#### 4.4.3.2 Delay

The delay for this element can be altered by using the methods outlined in the n-voltage controlled and p-voltage controlled sections. One advantage is that the delay can be altered by changing $V_p$ or $V_n$. The delay analysis is similar to both n-voltage controlled and p-voltage controlled elements. The only difference is that all (dis)charging takes place through a controlled transistor. The propagation delay of this element is:

$$t_p = \frac{1}{2}(t_{pLH} + t_{pHL}) = \frac{C_L V_{DD}}{2}\left(\frac{1}{k_p V_p^2} + \frac{1}{k_n V_n^2}\right) \tag{4.25}$$

Note that in this case, $t_p$ is proportional to both $V_p^2$ and $V_n^2$.

#### 4.4.3.3 Signal Integrity

The presence of control transistors in both pull-up and pull-down networks influence the rise and fall components of the signal integrity. That is, the rise time depends on $V_p$ and the fall time depends on $V_n$.

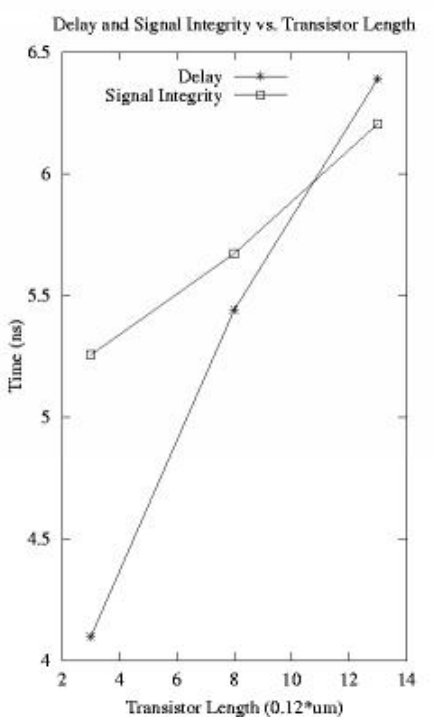
$$t_{r/f} = 0.9 C_L \left(\frac{1}{k_p V_p^2} + \frac{1}{k_n V_n^2}\right). \tag{4.26}$$

#### 4.4.3.4 Power Consumption

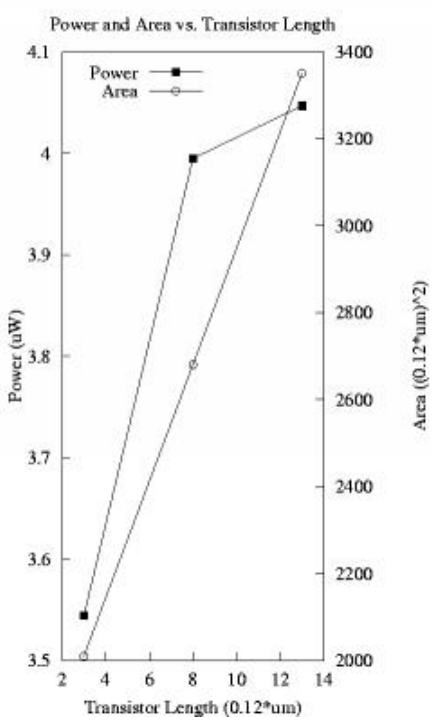
The power consumption is computed similar to the cascaded inverter case. It will be slightly more because of the additional diffusion capacitance of the controlled transistors that contributes to the total load capacitance.

#### 4.4.3.5 Area

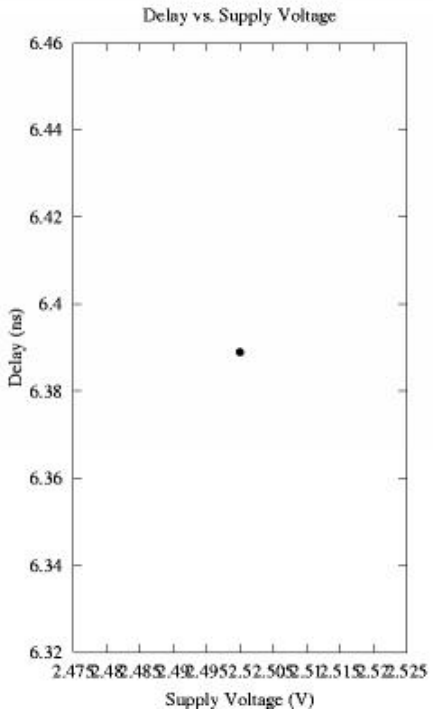
This delay element requires $4m+4$ transistors.



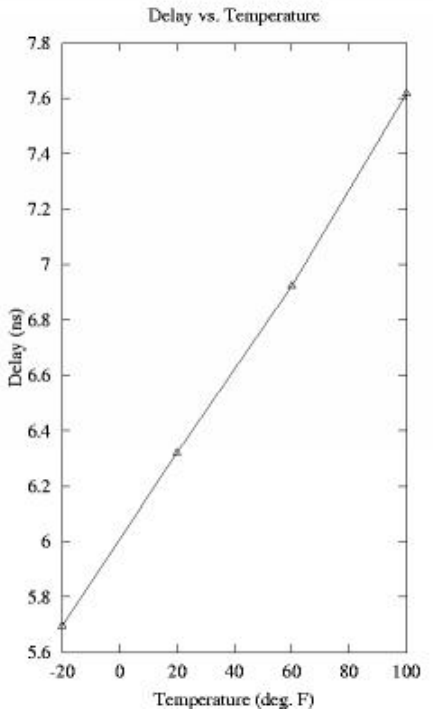

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$

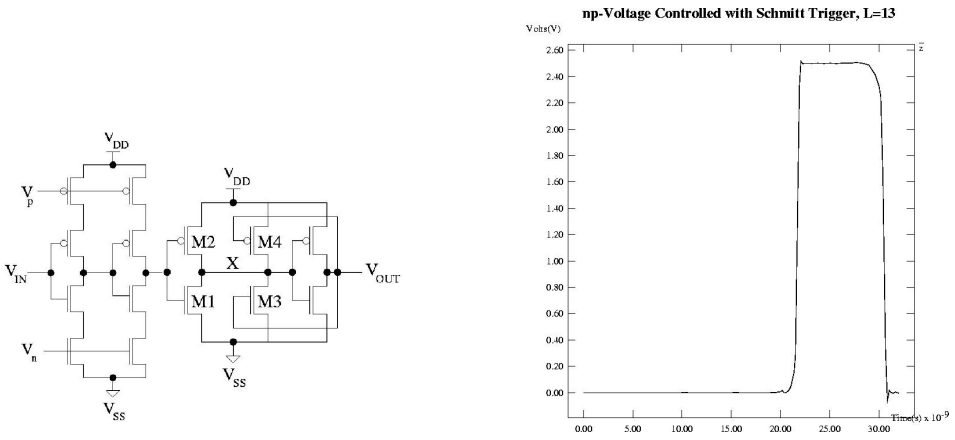



(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.19: np-Voltage controlled ($V_n = 0.75$V, $V_p = 1.75$V, alter length $L$): graphs of the analysis parameters




(a) (b)

Figure 4.20: np-Voltage controlled with Schmitt trigger (a) schematic (b) output signal

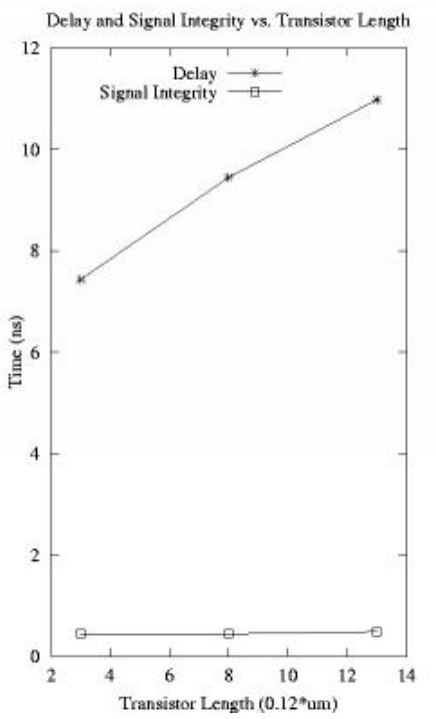
### 4.4.4 np-Voltage Controlled Cascaded with Schmitt trigger

#### 4.4.4.1 Description

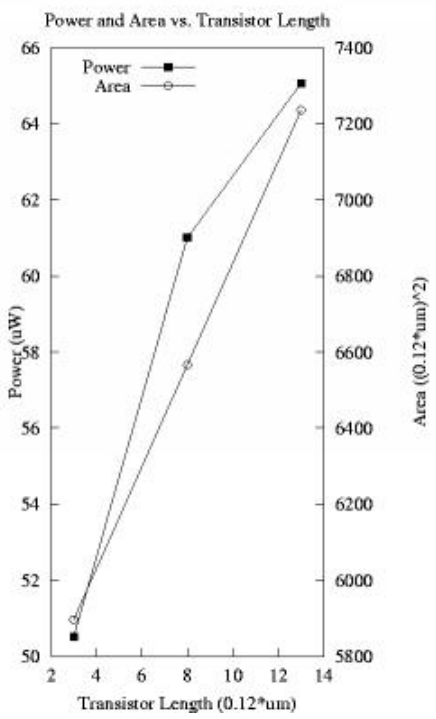
The poor signal integrity characteristic of the np-voltage controlled delay element can be improved with the addition of a Schmitt trigger to its output. As described in Section 4.2.2, a Schmitt trigger can produce a fast, clean signal from a noisy, slowly varying input.

#### 4.4.4.2 Delay

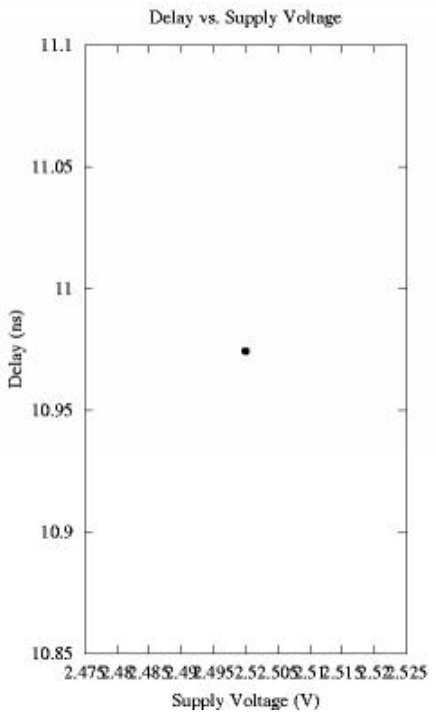
A special advantage of this delay element is that its delay can be changed in four ways. The first way involves altering the $W/L$ ratios of the transistors in the np-voltage controlled gate. Another way is to change the gate voltages applied to the control transistors in the pull-up and pull-down networks. Also, the fan-in $m$ of the gate can be increased, producing a greater delay. The final way is to change the switching thresholds of the Schmitt trigger, and this method is discussed in more detail in Section 4.2.2.



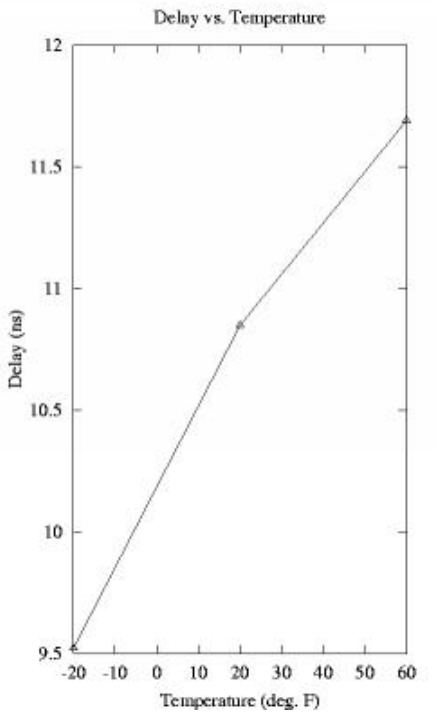

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.21: np-Voltage controlled with Schmitt trigger: graphs of the analysis parameters

#### 4.4.4.3 Signal Integrity

This delay element produces the best signal integrity characteristic out of all the delay elements proposed in this chapter. As the delay is altered by changing the $W/L$ ratios of the np-voltage controlled gate, the signal integrity remains virtually unchanged.

#### 4.4.4.4 Power Consumption

The main drawback of this circuit is that it consumes a great deal of power, much more than the other delay elements. The power analysis is similar to that of a cascaded inverter.

#### 4.4.4.5 Area

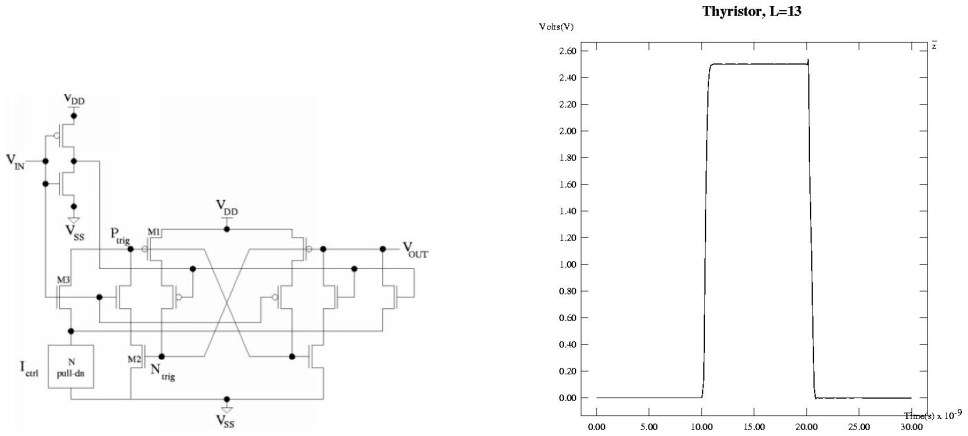
The addition of a Schmitt trigger means that this delay element requires $4m + 10$ transistors.

## 4.5 Thyristor Based

### 4.5.1 Thyristor

#### 4.5.1.1 Description

A static CMOS delay element based on the concept of a thyristor was proposed in [47] and is shown in Figure 4.22. Transistors $R1 - R5$ participate in delaying the rising edge of an input signal at $V_{in}$ to the output $V_{out}$, while transistors $F1 - F5$, connected symmetrically, do the same for the falling edge. The former are triggered by $V_{in}$, while the latter are triggered by $\bar{V}_{in}$, produced by the inverter consisting of transistors $I1 - I2$. When $V_{in}$ changes from low to high, it turns on $R1$ and $R3$, and $R1$ starts discharging $\bar{V}_{out}$ through the N pull-down network. When $\bar{V}_{out}$ drops to $V_{DD} - |V_{Tp}|$, it turns $R5$ on. $R4$ is turned on by $\bar{V}_{in}$, which charges up $V_{out}$. When $V_{out}$ rises to $V_{Tn}$, it turns on $R2$, which in




(a) (b)

Figure 4.22: Thyristor (a) schematic (b) output signal

turn pulls down $\bar{V}_{out}$ further towards $V_{SS}$ through $R3$, creating a positive feedback loop of transistors $R2-R3-R5-R4$. The positive feedback results in $V_{out}$ ($\bar{V}_{out}$) quickly rising (falling) to $V_{DD}$ ($V_{SS}$). During the rising edge, transistors $F1-F5$ are all off. The operation for the falling edge of $V_{in}$ can be understood similarly.

#### 4.5.1.2 Delay

The propagation delay of this element depends mainly upon the N pull-down, which acts as a controlled current source. In our simulations, we used an NMOS transistor with its gate connected to $V_{DD}$ as the pull-down network, but, in general, any pull-down network can be used. The more the resistance of the pull-down network, the larger the delay. This may be achieved by increasing the lengths of the transistors in the N pull-down and/or by increasing the number of transistors that are in series in the pull-down network. Increasing the lengths of the $R$ and $F$ transistors will also increase the delay. More specifically, charging of the output $V_{out}$ involves charging the gate capacitances of $R2$ and $F5$ and the diffusion capacitances of $R4$, $F1$, and $F3$ by a current that is controlled by the resistances of the N pull-down and transistors $R1-R5$. The discharging of $V_{out}$ is explained

similarly. The rise and fall delays can be separately controlled by having distinct N pull-downs for the $R$ and $F$ transistors. By changing the resistance of the N pull-down and consequently the control current, delay values from 2.6ns to 76.3ms were obtained in 0.8$\mu$m CMOS technology in [47].

#### 4.5.1.3 Signal Integrity

The signal integrity is good because of the positive feedback, which results in relatively sharp rise and fall of the output. However, it will becomes worse with the increases in resistances of the N pull-down and the $R$ and $F$ transistors.

#### 4.5.1.4 Power Consumption

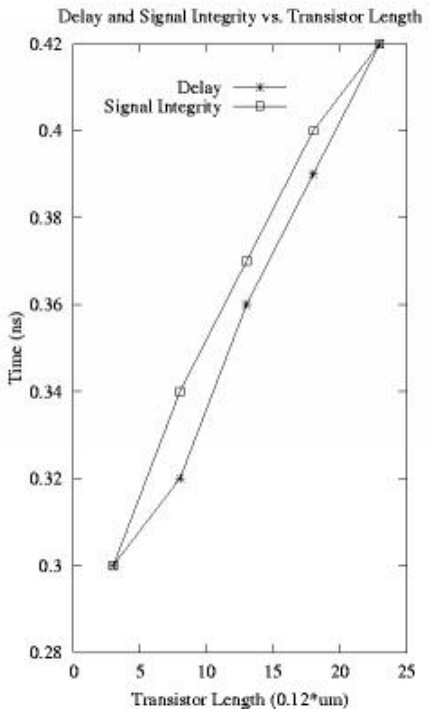
Since the signal integrity is relatively good, there is very little short-circuit power consumption. Most of the power consumption is caused by the charging of gate and diffusion capacitances of the $R$ and $F$ transistors during the rising and falling edges of the input signal.

#### 4.5.1.5 Area

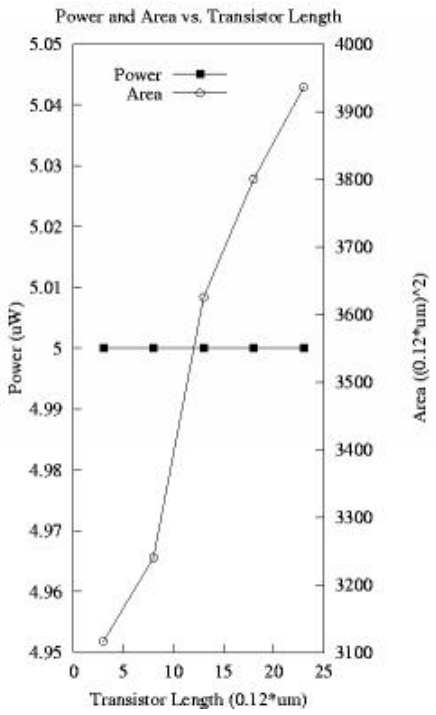
The area of the thyristor is relatively large as it uses at least 13 transistors if the N pull-down is just a single NMOS transistor, and more if the N pull-down is more complex.

#### 4.5.1.6 Other Characteristics

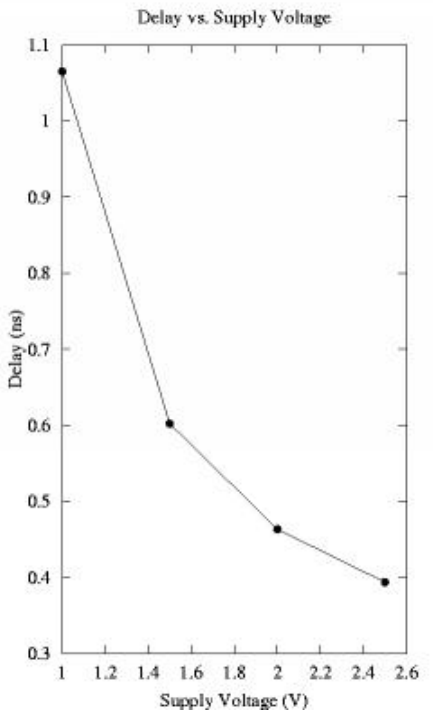
Other useful characteristics that this delay element has are insensitivity of the delay to variations and noise in the supply voltage and changes in temperature [47].



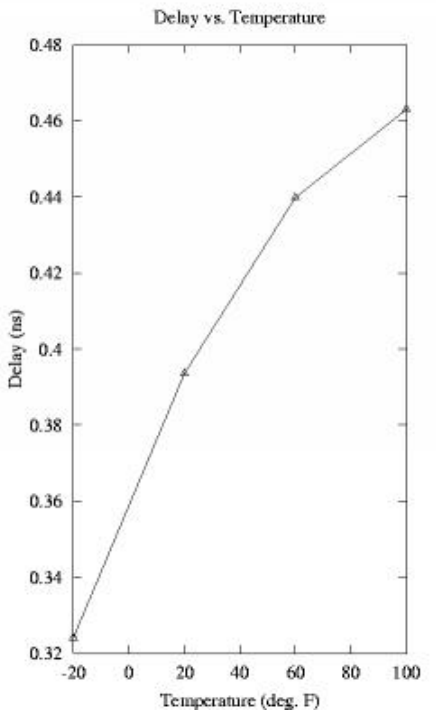

(a) Delay and signal integrity vs. $L$




(b) Power and area vs. $L$




(c) Delay vs. supply voltage($V_{DD}$)




(d) Delay vs. temperature(°F)

Figure 4.23: Thyristor: graphs of the analysis parameters

## 4.6 Comparisons

Experiments have been performed for these delay elements and the results were plotted in the preceding graphs. These delay elements were designed in 0.25$\mu$m technology. The simulations were performed by using SPICE 3. The parameters that were taken into account included the delay, the signal integrity, the power dissipation and the area. We altered the necessary transistors of the delay elements by changing the length of those transistors, while keeping their width at a constant minimum value. Then, we extracted the necessary parameters. We began with $L = 3\lambda$, and increased the transistor length in steps of $5\lambda$ to $L = 23\lambda$. A standard cell inverter was used as the fan-in and fan-out for each delay element. The items that we tested included the following:

1. Transmission gate
2. Transmission gate cascaded with Schmitt trigger
3. Cascaded inverters
4. m-transistor cascaded inverters
5. Current starved cascaded inverters
6. Staged cascaded inverters
7. n-voltage controlled ($V_n = 0.75$V, alter transistor length)
8. n-voltage controlled (alter voltage)
9. p-voltage controlled ($V_p = 1.75$V, alter transistor length)
10. np-voltage controlled ($V_n = 0.75$V, $V_p = 1.75$V, alter transistor length)

11. np-voltage controlled cascaded with Schmitt trigger ($V_n = 0.75$V, $V_p = 1.75$V, alter transistor length)

12. Thyristor (using a NMOS transistor for the current source)

For each of these twelve items, experiments were performed and four plots were drawn for each element. The plots show the relationship between the parameters of delay, signal integrity, power and area with that of the delay element's transistor length, $L$. The plots gave us the type of results we expected, according to the above analysis. From these plots, we can give several suggestions for the situations under which different delay elements can be used.

1. The advantage of the transmission gate is the small area overhead and power dissipation. Unfortunately, it has a bad signal integrity. It is suitable for delays of less than 1ns.

2. The transmission gate cascaded with Schmitt trigger has a valuable advantage in that its signal integrity changes little as its delay increases. It is suitable for delays of more than 3ns, and middle-size or large-size designs. Unfortunately, it has larger area and power overheads when compared to the other delay elements. Changing the size of transmission gate has a more profound effect on the delay when compared to changing the size of Schmitt trigger.

3. The cascaded inverter has a relatively small area and power dissipation. Its signal integrity is good for a delay value of less than 3ns. It is most suitable for a delay of 1ns to 3ns, and for a middle-size or large-size design.

4. The m-transistor cascaded inverter has similar qualities to the above mentioned cascaded inverter element. However, It has a larger range of delay that it can provide, depending upon the value of $m$ chosen. Also, it translates into a larger power requirement.

5. The current-starved cascaded inverter has a slightly larger delay range and similar power requirements to its previous two cascaded inverter counterparts. The disadvantage is that its signal integrity value is very poor, one of the worst among the delay elements examined.

6. The staged cascaded inverter provides a very small delay value for the amount of power and area it requires, and these overheads virtually negate its usefulness. One minor advantage is that its signal integrity is very good, although this is no big surprise for an element that produces a small delay.

7. The n-voltage controlled delay element, in this instance, had its control transistor $V_n$ held at 0.75V while its transistor length $L$ was altered. It is most suitable for delay values between 1ns and 6ns. It has a slightly greater area and power consumption when compared to the cascaded inverters, because of the extra NMOS control devices. They also degrade the signal integrity of the element.

8. The n-voltage controlled delay element can change its delay value by altering the controlling voltage, $V_n$. However, its delay producing properties are significantly diminished in this mode.

9. Like its n-voltage controlled counterpart, the p-voltage controlled element produces a wide range of delay values, with similar signal integrity and power consumption qualities.

10. The np-voltage controlled element is a combination of the above two designs. Its delay range is narrow, but large compared to the cascaded inverter based elements. It has a very poor signal integrity response, and this deficiency is overcome in the next examined delay element.

11. The np-voltage controlled with Schmitt trigger should only be used in cases where a large, clean delay value is required at any cost. Its power consumption is ten times that of any other

delay element examined here, and it occupies a relatively large amount of area. Its delay and signal integrity qualities are excellent, and unmatched by any other element in this chapter.

12. The thyristor provides a very small amount of delay relative to its size, power requirements and design complexity. Any one of the above mentioned delay elements would be preferable to this one.

| **Transmission Gate Based** | | | | |
|---|---|---|---|---|
| **Delay Element** | **Parameter** | **Range** | **Proportional to** | **Rank** |
| Transmission Gate | Delay (ns) | $0.13 - 0.57$ | $R_{eq}C_L$ | 5 |
| | Signal Integrity (ns) | $0.23 - 1.22$ | $R_{eq}C_L$ | 3 |
| | Power ($\mu$W) | $0.023 - 0.026$ | — | 1 |
| | Area ($\mu$m$^2 10^3$) | $377 - 957$ | — | 1 |
| | $V_{DD}$ (ns) | $0.32 - 0.58$ | — | 1 |
| | Temperature (ns) | $0.20 - 0.44$ | — | 1 |
| Transmission Gate with Schmitt Trigger | Delay (ns) | $0.90 - 2.17$ | $V_{M+}, V_{M-}$ | 4 |
| | Signal Integrity (ns) | $0.46 - 0.51$ | $V_{M+}, V_{M-}$ | 1 |
| | Power ($\mu$W) | $14.8 - 25.2$ | $C_L, V_{DD}$ | 5 |
| | Area ($\mu$m$^2 10^3$) | $3116 - 3936$ | — | 3 |
| | $V_{DD}$ (ns) | $1.38 - 1.67$ | — | 1 |
| | Temperature (ns) | $0.93 - 1.78$ | — | 1 |

Table 4.1: Comparison of the transmission gate based delay elements

| **Thyristor Based** | | | | |
|---|---|---|---|---|
| **Delay Element** | **Parameter** | **Range** | **Proportional to** | **Rank** |
| Thyristor | Delay (ns) | $0.30 - 0.42$ | $C_L, V_{DD}, k_p, k_n$ | 5 |
| | Signal Integrity (ns) | $0.34 - 0.42$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Power ($\mu$W) | 5.0 | $C_L, V_{DD}$ | 4 |
| | Area ($\mu$m$^2 10^3$) | $3116 - 3936$ | — | 3 |
| | $V_{DD}$ (ns) | — | — | — |
| | Temperature (ns) | — | — | — |

Table 4.2: Comparison of the thyristor based delay elements

| Cascaded Inverter Based | | | | |
|---|---|---|---|---|
| **Delay Element** | **Parameter** | **Range** | **Proportional to** | **Rank** |
| Cascaded Inverters | Delay (ns) | $0.20 - 2.91$ | $C_L, V_{DD}, k_p, k_n$ | 4 |
| | Signal Integrity (ns) | $0.11 - 1.53$ | $C_L, V_{DD}, k_p, k_n$ | 3 |
| | Power ($\mu$W) | $2.33 - 5.80$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $812 - 2030$ | $4m$ | 2 |
| | $V_{DD}$ (ns) | $1.25 - 5.07$ | — | 4 |
| | Temperature (ns) | $0.81 - 1.74$ | — | 2 |
| m-Transistor Cascaded Inverters | Delay (ns) | $0.30 - 3.63$ | $C_L, V_{DD}, k_p, k_n$ | 3 |
| | Signal Integrity (ns) | $0.20 - 1.96$ | $C_L, V_{DD}, k_p, k_n$ | 3 |
| | Power ($\mu$W) | $2.98 - 6.53$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $1736 - 4340$ | $m$ | 4 |
| | $V_{DD}$ (ns) | 1.73 | — | 2 |
| | Temperature (ns) | $1.69 - 2.50$ | — | 2 |
| Current Starved Cascaded Inverters | Delay (ns) | $1.27 - 4.07$ | $C_L, V_{DD}, k_p, k_n$ | 2 |
| | Signal Integrity (ns) | $2.96 - 4.16$ | $C_L, V_{DD}, k_p, k_n$ | 5 |
| | Power ($\mu$W) | $2.55 - 5.72$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $3128 - 5848$ | — | 4 |
| | $V_{DD}$ (ns) | $2.45 - 5.74$ | — | 4 |
| | Temperature (ns) | $1.99 - 3.40$ | — | 4 |
| Staged Cascaded Inverters | Delay (ns) | $0.76 - 1.62$ | $C_L, V_{DD}, k_p, k_n$ | 4 |
| | Signal Integrity (ns) | $0.69 - 0.86$ | $C_L, V_{DD}, k_p, k_n$ | 2 |
| | Power ($\mu$W) | $5.33 - 5.74$ | $C_L, V_{DD}$ | 4 |
| | Area ($\mu$m$^2 10^3$) | $5133 - 6960$ | — | 5 |
| | $V_{DD}$ (ns) | $1.25 - 4.61$ | — | 5 |
| | Temperature (ns) | $1.02 - 1.50$ | — | 2 |

Table 4.3: Comparison of the cascaded inverter based delay elements

## 4.7 Conclusions

This chapter discussed twelve different types of delay elements, each one with a particular quality. Each of these delay elements were simulated using a 2.5V voltage supply. Naturally, if a smaller value of supply voltage were used, the power consumption would be much less. The delay elements are listed in Tables 4.1, 4.2, 4.3, and 4.4, providing an overview of the data ranges for each performance parameter. The best choice for a delay element would be the cascaded inverters. It has the best signal integrity in the 1ns to 3ns range and relatively minimal power and area overheads. The

| Voltage Controlled Based | | | | |
|---|---|---|---|---|
| **Delay Element** | **Parameter** | **Range** | **Proportional to** | **Rank** |
| n-Voltage Controlled ($V_n = 0.75$V, Alter Length $L$) | Delay (ns) | $1.43 - 6.13$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Signal Integrity (ns) | $1.85 - 3.45$ | $C_L, V_{DD}, k_p, k_n$ | 4 |
| | Power ($\mu$W) | $3.11 - 4.81$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $1530 - 3570$ | — | 3 |
| | $V_{DD}$ (ns) | $3.7 - 6.64$ | — | 3 |
| | Temperature (ns) | $2.9 - 5.0$ | — | 2 |
| n-Voltage Controlled (Alter Voltage $V_n$) | Delay (ns) | $0.30 - 1.43$ | $C_L, V_{DD}, k_p, k_n$ | 4 |
| | Signal Integrity (ns) | $0.20 - 1.85$ | $C_L, V_{DD}, k_p, k_n$ | 3 |
| | Power ($\mu$W) | $2.46 - 3.11$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | 1530 | — | 2 |
| | $V_{DD}$ (ns) | $1.43 - 1.69$ | — | 1 |
| | Temperature (ns) | $1.08 - 1.97$ | — | 1 |
| p-Voltage Controlled ($V_p = 1.75$V, Alter Length $L$) | Delay (ns) | $1.43 - 6.13$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Signal Integrity (ns) | $1.85 - 3.45$ | $C_L, V_{DD}, k_p, k_n$ | 4 |
| | Power ($\mu$W) | $3.11 - 4.81$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $1530 - 3570$ | — | 3 |
| | $V_{DD}$ (ns) | $3.7 - 6.64$ | — | 3 |
| | Temperature (ns) | $2.9 - 5.0$ | — | 2 |
| np-Voltage Controlled ($V_n = 0.75$V, ($V_p = 1.75$V, Alter Length $L$) | Delay (ns) | $4.10 - 6.39$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Signal Integrity (ns) | $5.25 - 6.20$ | $C_L, V_{DD}, k_p, k_n$ | 5 |
| | Power ($\mu$W) | $3.54 - 4.05$ | $C_L, V_{DD}$ | 3 |
| | Area ($\mu$m$^2 10^3$) | $2010 - 3350$ | — | 3 |
| | $V_{DD}$ (ns) | 6.39 | — | 2 |
| | Temperature (ns) | $5.69 - 7.62$ | — | 3 |
| np-Voltage Controlled ($V_n = 0.75$V, ($V_p = 1.75$V, Alter Length $L$) with Schmitt T. | Delay (ns) | $7.43 - 10.9$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Signal Integrity (ns) | $0.43 - 0.48$ | $C_L, V_{DD}, k_p, k_n$ | 1 |
| | Power ($\mu$W) | $50.5 - 65.0$ | $C_L, V_{DD}$ | 5 |
| | Area ($\mu$m$^2 10^3$) | $5896 - 7236$ | — | 5 |
| | $V_{DD}$ (ns) | 10.97 | — | 4 |
| | Temperature (ns) | $9.52 - 11.69$ | — | 3 |

Table 4.4: Comparison of the voltage controlled based delay elements

next best element would be the n-voltage controlled delay element. Its performance is only slightly worse than that of the cascaded inverters. The transmission gate with Schmitt trigger would be the next logical choice after that, especially for delays greater than 3ns. The other elements would be best used in specialized scenarios. For instance, the transmission gate works best when power must be minimized, and signal integrity is not an issue.

# Chapter 5

# Application of the ckt4 Glitch Minimization Technique on Arithmetic Units

## 5.1 Experimental Method

### 5.1.1 The fundamental static adder cell

#### 5.1.1.1 Design

The first step was determining the exact logical description of the static adder cell, in terms of a truth table and boolean equations, namely sum-of-products (SOP) logical descriptions. The logical descriptions were then translated into the transistor expression of the complex logic block. The CAD layout editor MAGIC was used to draw the circuit under a given set of design rules. Within this program, wire routing problems were solved and design rules were checked. The NMOS and

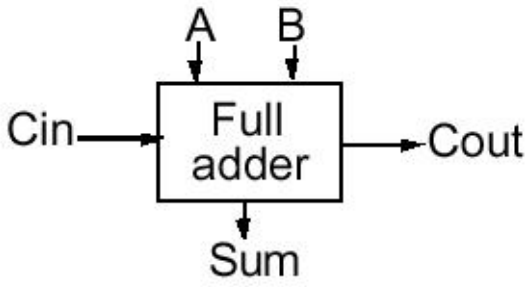



Figure 5.1: The full adder

| A | B | Cin | Sum | Cout | Carry Status |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | delete |
| 0 | 0 | 1 | 1 | 0 | delete |
| 0 | 1 | 0 | 1 | 0 | propagate |
| 0 | 1 | 1 | 0 | 1 | propagate |
| 1 | 0 | 0 | 1 | 0 | propagate |
| 1 | 0 | 1 | 0 | 1 | propagate |
| 1 | 1 | 0 | 0 | 1 | generate |
| 1 | 1 | 1 | 1 | 1 | generate |

Table 5.1: Truth table for the full adder

PMOS devices had to be properly sized in order to ensure equal rise and fall times. I implemented a $W_p/W_n$ ratio of 3.

Boolean equations for the full adder:

$$S = A \oplus B \oplus Cin$$

$$S = A\overline{BCin} + \overline{A}B\overline{Cin} + \overline{AB}Cin + ABCin \tag{5.1}$$

$$Cout = AB + BCin + ACin \tag{5.2}$$

#### 5.1.1.2 Test vectors

A series of twenty-five, 8-bit random test vectors were generated using the RANDOM function on a MS Excel spreadsheet program. They were used as the basis for forming the 4, 16 and 32 bit test vectors. In this manner, power simulations of different bit widths could be compared to each other

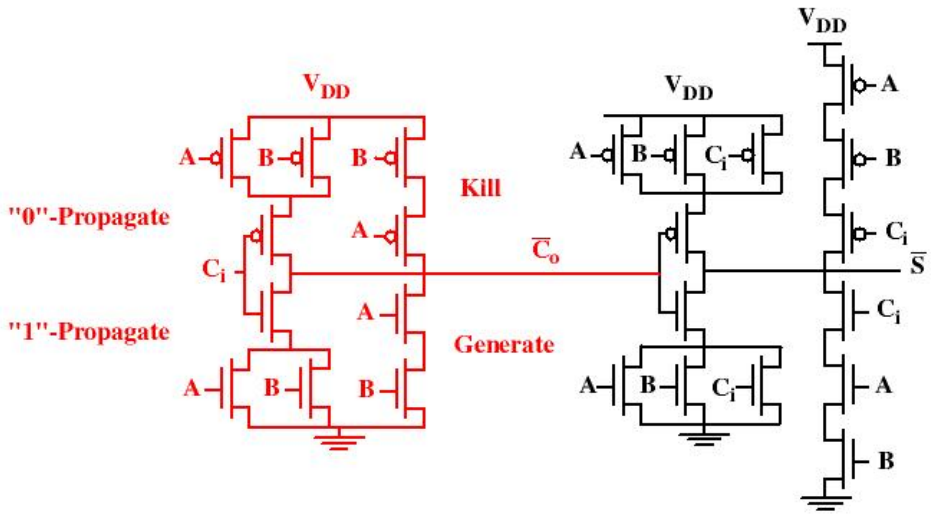



Figure 5.2: Schematic of the mirror adder

without introducing another random variable.

#### 5.1.1.3 Power simulation

Once the circuit was constructed, it was extracted into a description suitable for SPICE 3 simulation, where the transistors were listed as device models. A timing simulation was run, and this verified the amount of time required for an input signal to span one bit, as well as the entire bit-width of the circuit (worst-case delay). A series of test vectors were generated, and they were placed in the circuit's SPICE 3 input file. A SPICE 3 simulation was run, and every output was examined and compared with the test vectors. If the circuit conformed to them, then a power simulation was run under SPICE 3. The power was recorded along with other circuit qualities such as number of inputs, outputs and gates.

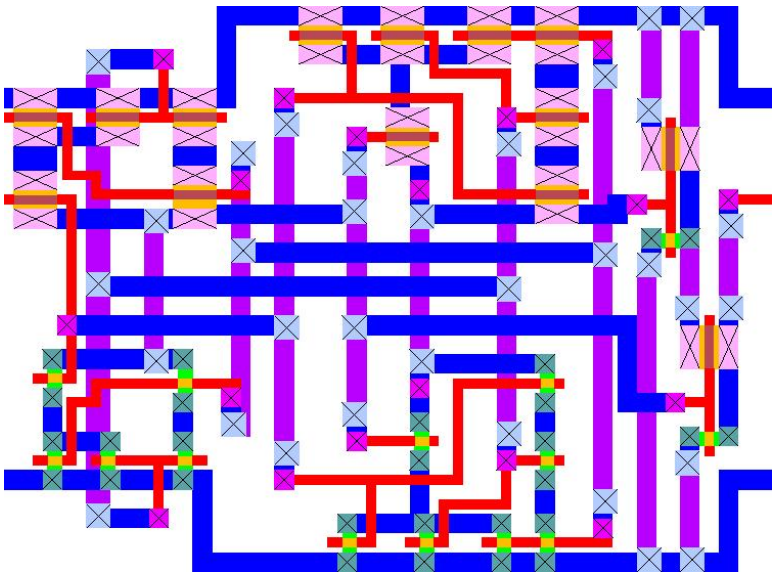


Figure 5.3: MAGIC layout of the mirror adder

### 5.1.2 Appending the ckt4 technique to the fundamental static adder cell

#### 5.1.2.1 Design

The first step was to examine the static adder cell and decide which gates were large enough to justify the added control logic overhead. I reasoned that smaller gates, such as inverters, did not need control logic because their propagation delay values compared to the overall cell would be negligible. Also, single-input gates such as inverters didn't create glitches, so placing control logic on them would be a wasteful expense. A signal analysis was performed on the original circuit to determine the gates that were on the critical path. This analysis also told us which gates needed to be triggered simultaneously, so we could amortize the effect of a single control transistor over several gates. Then, the original circuit was modified by adding control transistors to each of these critical path gates.

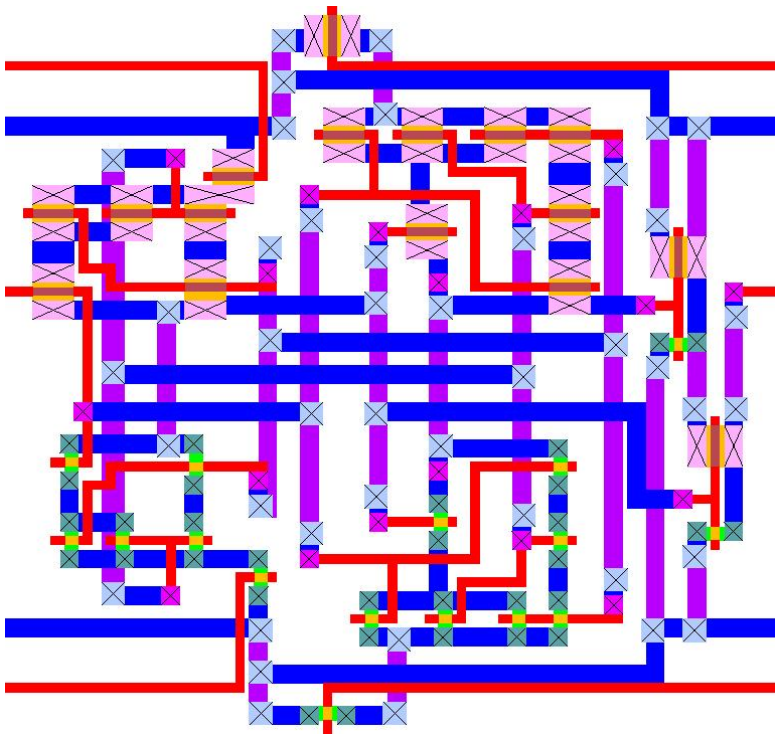


Figure 5.4: MAGIC layout of the mirror adder with the ckt4 technique

#### 5.1.2.2 Simulating the effect of a delay chain

The critical path nodes in the original circuit were examined and the time at which each node settled was noted. The effect of a delay chain was simulated by placing activating signal pulses directly on the control transistors. A pulse was applied to each control transistor so the gate was activated just after all the inputs have arrived and settled. The pulse activated each gate for roughly 2ns at a time – a window just big enough for the gate to switch on, evaluate its inputs, and switch off.

#### 5.1.2.3 Power simulation

A SPICE 3 simulation was performed, and every gate output was examined and compared to the test vectors to ensure that all the glitches have been removed. The power value was recorded. Since we didn't use delay chains in this step, the power due to the overhead came from the control transistors.

The total power reduced was computed by the following equation:

$$\text{Total (net) power reduced} = \frac{\text{(power in original)} - \text{(power in ckt4)}}{\text{(power in original)}} \times 100 \tag{5.3}$$

### 5.1.3 Determining the power consumed by the circuit with ckt4 and a delay chain

The delay chain implementation has been unsuccessful due to difficulties in synchronizing the control signal with the controlled gates.

## 5.2 Analysis: The Ripple Carry Adder

### 5.2.1 Design

The first approach was to lay out the functionality of the circuit at the highest level of abstraction: the module level. The circuit was described as a series of complex logic blocks such as adder units, which were connected together. The modules were laid out in the manner described in Figure 5.5.

### 5.2.2 Transition density

A transition density analysis was necessary to obtain an estimate of the power consumed due to glitching. A Verilog simulation was performed at the transistor level. The output of this simulation was parsed into an MS Excel spreadsheet program for further analysis. My goal was to extract the number of power consuming transitions and glitches that the circuit produced. Table 5.2 outlines the four transistion instances between two successive digits that are possible in binary logic.

In order to determine whether or not a transition occurred, I subtracted cell $X(n)$ from cell $X(n+1)$, where $n$ is an integer. If the result was a zero, I knew that both cells were equal and no transition occurred. If the result was one, I knew that a power consuming transition occurred, and

| Binary digits | Instance |
|---|---|
| 0 to 0 | no transition |
| 0 to 1 | one power consuming transition |
| 1 to 0 | one (non-power consuming) transition |
| 1 to 1 | no transition |

Table 5.2: Transistion instances between two successive binary digits

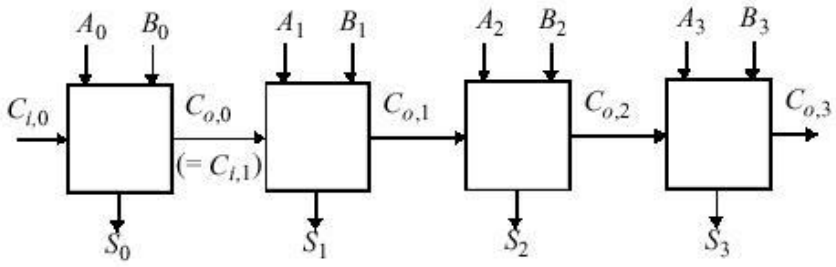



Figure 5.5: The ripple carry adder

those were the ones I wanted to count. Therefore, I used the following $IF$ statement to determine a Power Consuming Transition (PCT):

$$IF(X(n+1)-X(n)=1,1,0) \tag{5.4}$$

$$PCT = SUM(X(n):X(n+y)) \tag{5.5}$$

From the matrix of power consuming transitions, I wanted to extract the ones that were due to glitching. I used the following two assumptions. If the initial value equaled the final value, then the number of glitchy transitions equals the number of total transitions. If the initial value did not equal the final value, then the number of glitchy transitions was equal to the number of total transitions, minus 1. This led to the following IF statement:

$$IF(X(n+y)-X(n)=1,(PCT-1),PCT) \tag{5.6}$$

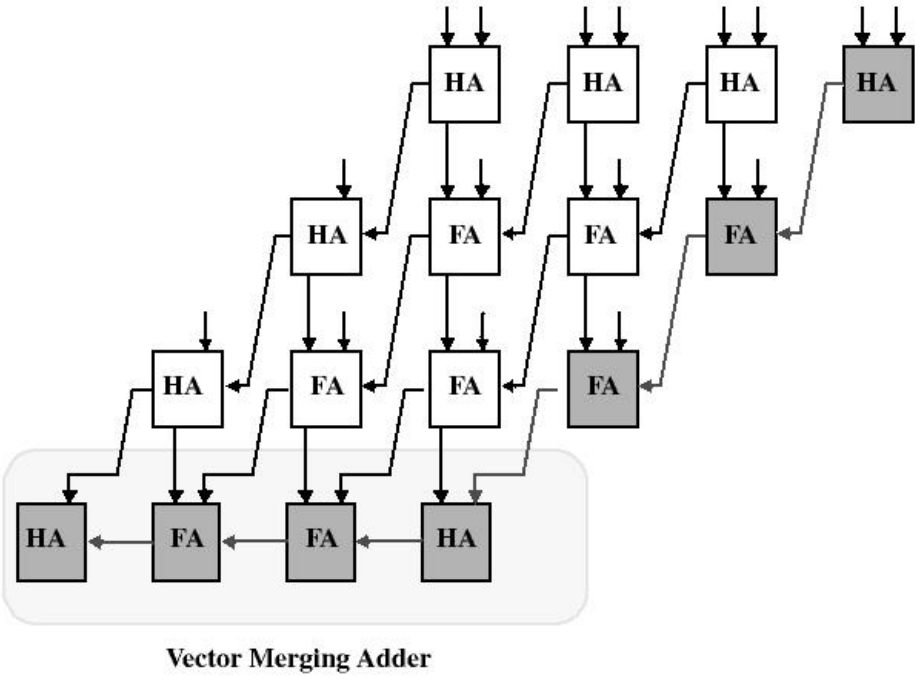



Figure 5.6: The carry save multiplier

## 5.3 Analysis: The Carry Save Multiplier

### 5.3.1 Design

The first approach was to lay out the functionality of the circuit at the highest level of abstraction: the module level. The circuit was described as a series of complex logic blocks such as adder units, which were connected together in a 2-dimensional matrix as outlined in Figure 5.6.

### 5.3.2 Transition density

An analysis was performed using Verilog at the module level to determine the power due to glitching. The analysis theory was similar to that used for the ripple carry adder, described in Section 5.2.2.

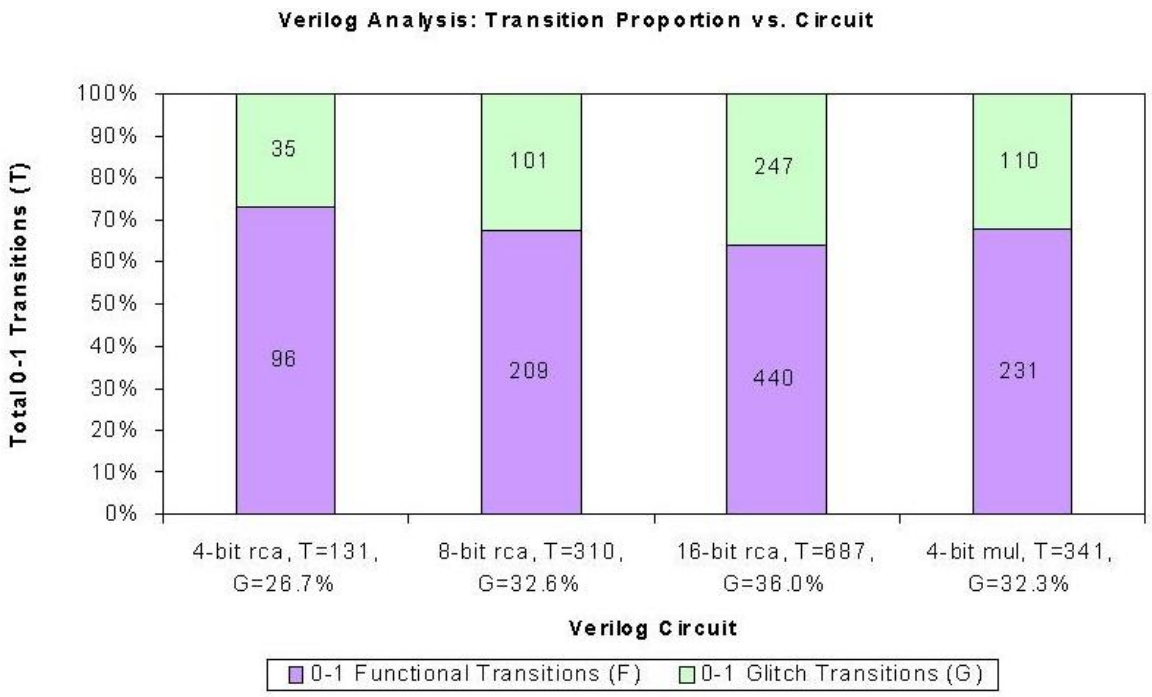



Figure 5.7: Verilog analysis for the rca and mul

## 5.4 Experimental Results

### 5.4.1 Verilog analysis

Figure 5.7 is a chart of transition densities for the ripple carry adder and carry save multiplier.

### 5.4.2 SPICE 3 analysis

Table 5.3 outlines a power analysis performed using SPICE 3 for the ripple carry adder and carry save multiplier.

The following are a series of graphs for the input and output signals of the 8 bit ripple carry adder and the 8 bit carry save multiplier. The output signals give the reader a chance to visually compare the amount of glitching present in the regular implementation, and the amount reduced in the ckt4 implementation of each circuit.

| Circuit | Bit width | Primary inputs | Primary outputs | Gate count | Gates controlled | (orig) Power (a)($\mu$W) | (ckt4) Power (b)($\mu$W) | % Power* reduced |
|---|---|---|---|---|---|---|---|---|
| Ripple Carry Adder | 4 | 9 | 5 | 16 | 7 | 0.060841 | 0.063191 | −3.86 |
| | 8 | 17 | 9 | 32 | 15 | 0.152238 | 0.160513 | −5.43 |
| | 16 | 33 | 17 | 64 | 31 | 0.334317 | 0.393969 | −17.84 |
| | 32 | 65 | 33 | 128 | 63 | 0.349688 | 0.549482 | −57.13 |
| Carry Save Mult. | 4 | 8 | 8 | 50 | 17 | 0.213227 | 0.228765 | −7.28 |
| | 8 | 16 | 16 | 230 | 55 | 1.62812 | 1.27712 | 21.56 |

Table 5.3: SPICE 3 power analysis for the rca and mul. *Negative values indicate a power increase.

| Parameter | Value |
|---|---|
| circuit style | static CMOS |
| power supply voltage ($V_{DD}$) | 2.5V |
| lambda | 0.12 (0.25$\mu$m) |
| functional unit | static full-adder circuit |
| $W_p/W_n$ ratio | 3 |
| architectures | ripple-carry adder, carry-save multiplier |
| ripple-carry adder bit widths | 4, 8, 16 and 32 bits |
| carry-save multiplier bit widths | 4 and 8 bits |
| glitch minimization technique | ckt4 |
| number of random test vectors | 25 |

Table 5.4: Parameters used in the MAGIC and SPICE 3 simulations

#### 5.4.2.1 8-bit Ripple Carry Adder

The following is the signal label information used in the MAGIC layouts for the regular(Figure 5.8) and ckt4(Figure 5.9) ripple carry adders.

inputs: $a0 \ldots a7$, $b0 \ldots b7$, $cin$

outputs: $s0 \ldots s7$, $cout$

adds: $(a0 \ldots a7) + (b0 \ldots b7) + cin = (s0 \ldots s7) + cout$

input data: 25 random test vectors

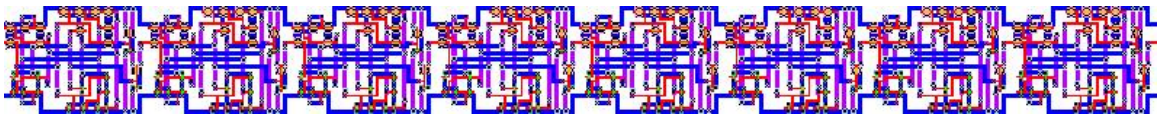


Figure 5.8: rca(a): regular ripple carry adder

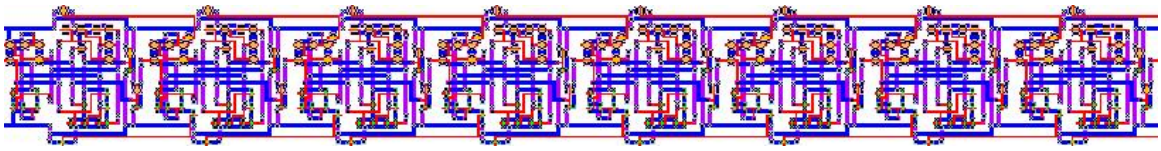


Figure 5.9: rca(b): ckt4 ripple carry adder

The waveforms for the input signals $a0 \ldots a7$ are described in Figures 5.10 and 5.11. Similarly, the input signals applied to $b0 \ldots b7$ are presented in Figures 5.12 and 5.13. Following these graphs are the results of the addition of these input signals for the regular ripple carry adder, rca(a), and the ckt4 ripple carry adder, rca(b). The addition results for the regular ripple carry adder are outlined in Figures 5.14 and 5.15, namely $s0 \ldots s7$. Also, the solution results of the ckt4 ripple carry adder are presented here, in Figures 5.16 and 5.17. Note that the same output signal labels are used, $s0 \ldots s7$. The following is a specific breakdown of the various signals and their corresponding Figure allocations.

Figure 5.10: $a0 \ldots a3$

Figure 5.11: $a4 \ldots a7$

Figure 5.12: $b0 \ldots b3$

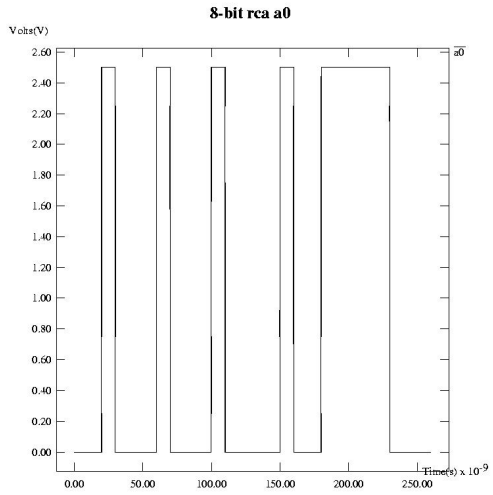
Figure 5.13: $b4 \ldots b7$

Figure 5.14, rca(a): $s0 \ldots s3$

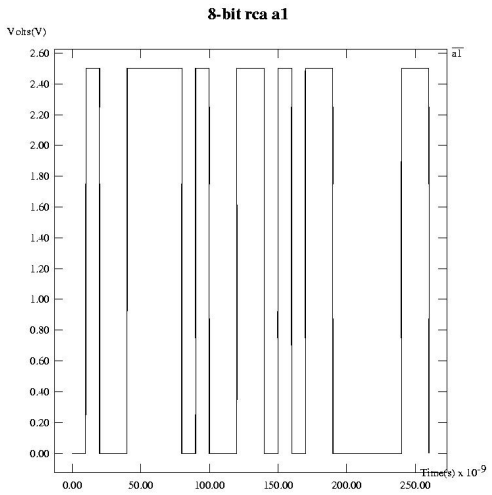
Figure 5.15, rca(a): $s4 \ldots s7$

Figure 5.16, rca(b): $s0 \ldots s3$

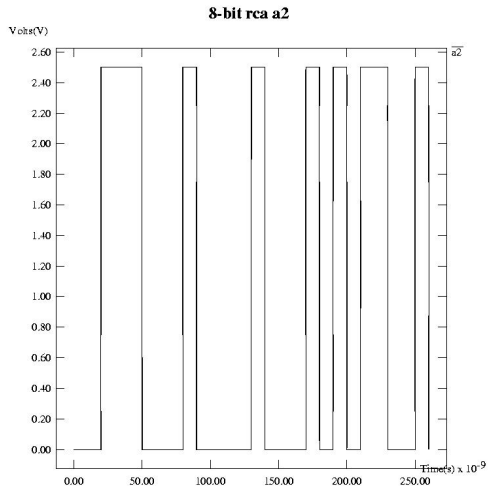
Figure 5.17, rca(b): $s4 \ldots s7$



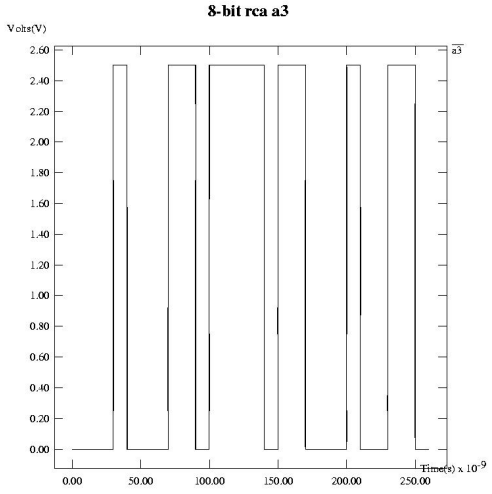

(a)




(b)

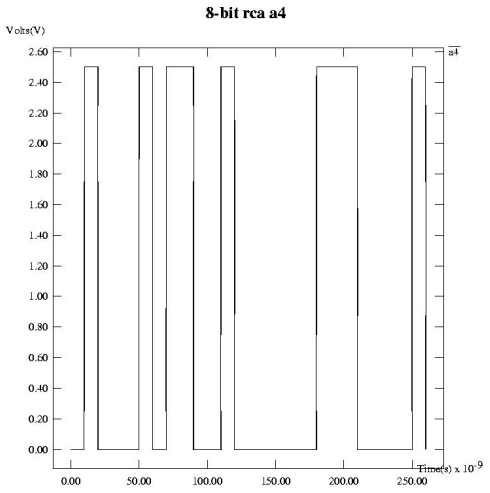



(c)

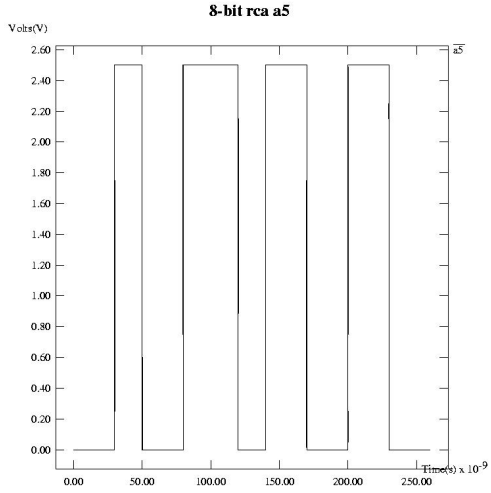



(d)

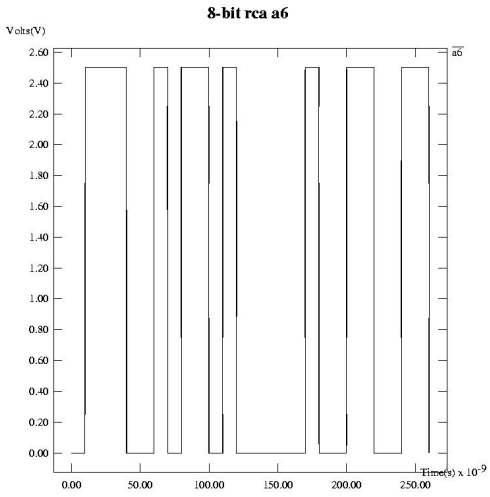
Figure 5.10: 8-bit ripple carry adder inputs: (a) $a0$ (b) $a1$ (c) $a2$ (d) $a3$



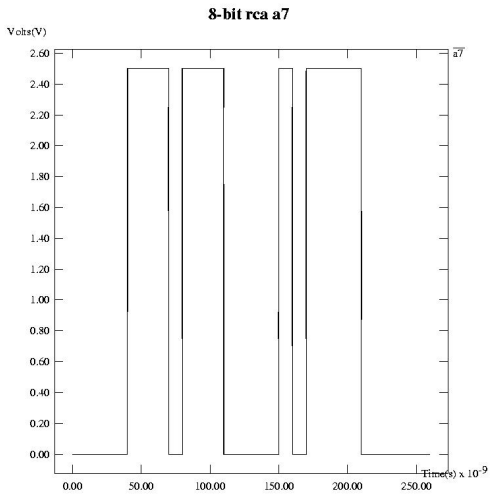

(a)




(b)




(c)




(d)

Figure 5.11: 8-bit ripple carry adder inputs: (a) $a4$ (b) $a5$ (c) $a6$ (d) $a7$

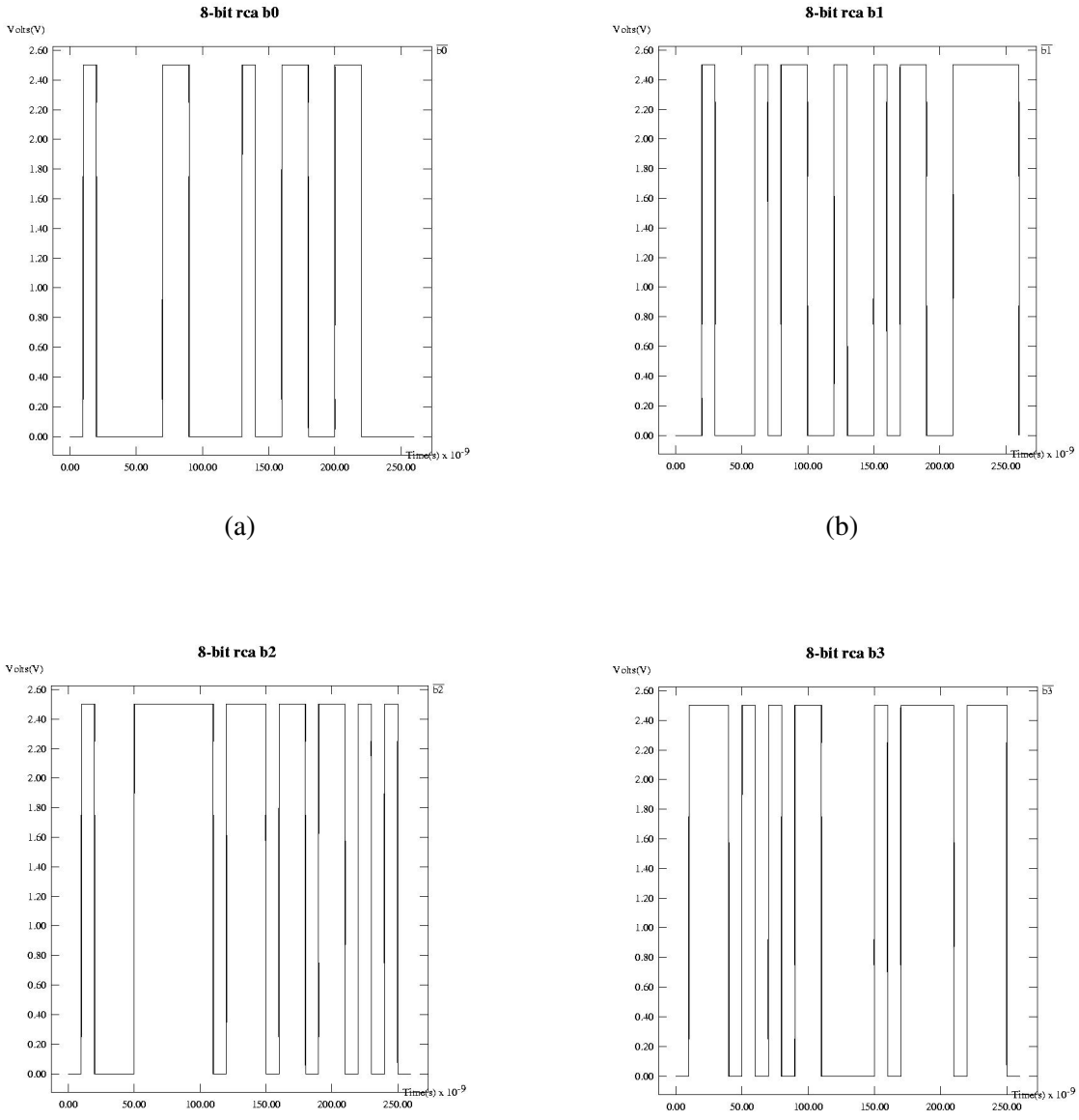
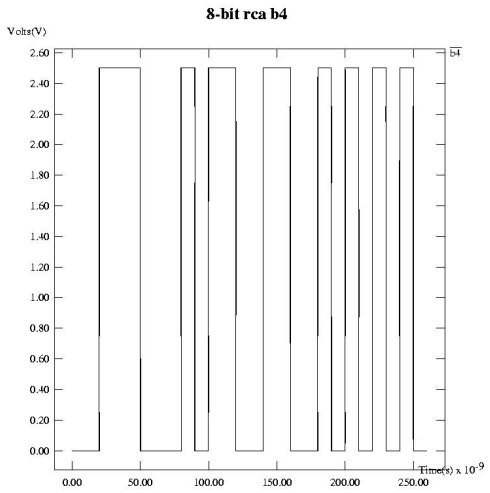
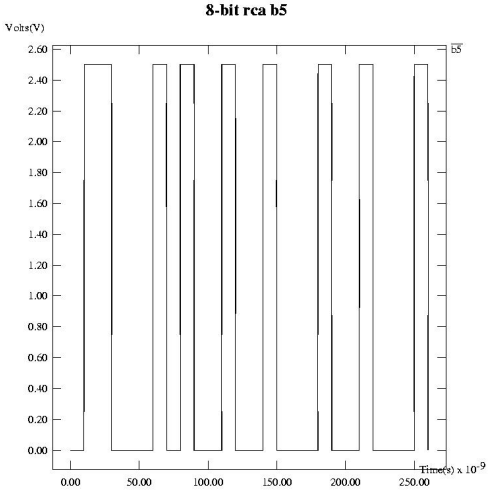


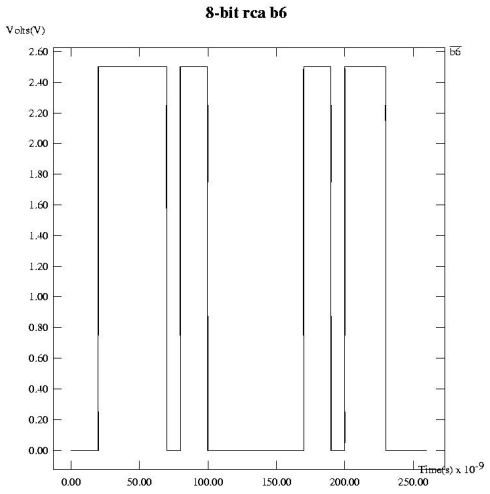

Figure 5.12: 8-bit ripple carry adder inputs: (a) $b0$ (b) $b1$ (c) $b2$ (d) $b3$



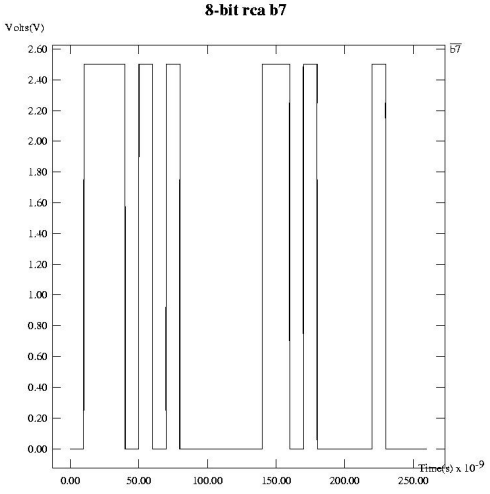

(a)




(b)

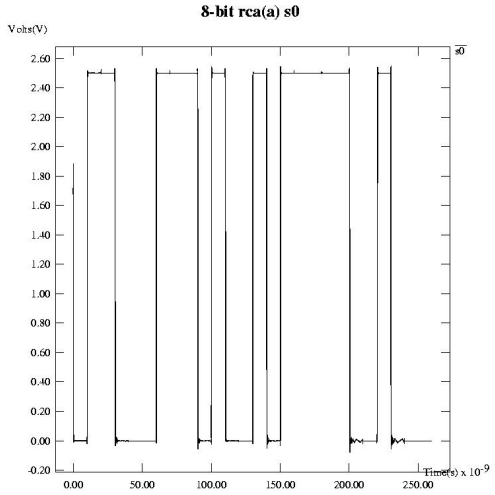



(c)

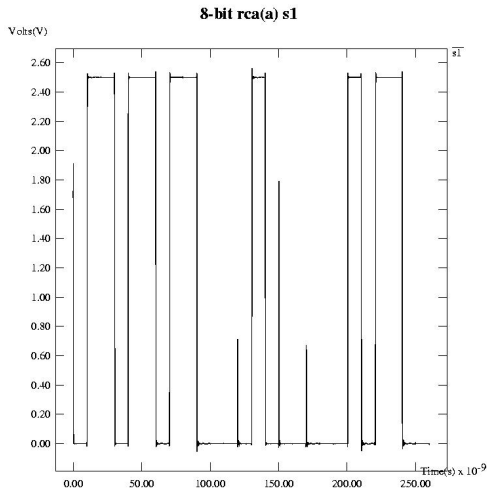



(d)

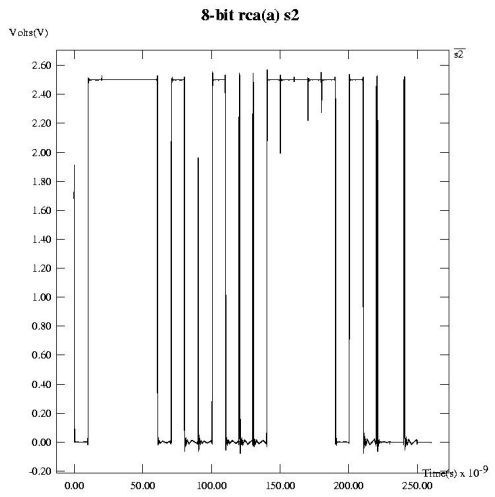
Figure 5.13: 8-bit ripple carry adder inputs: (a) $b4$ (b) $b5$ (c) $b6$ (d) $b7$



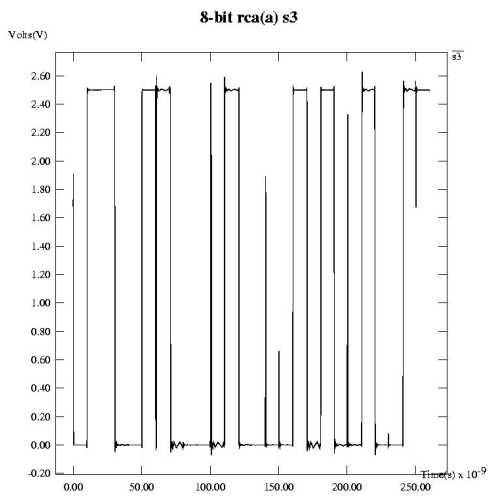

(a)




(b)




(c)




(d)

Figure 5.14: 8-bit regular ripple carry adder outputs: (a) $s0$ (b) $s1$ (c) $s2$ (d) $s3$

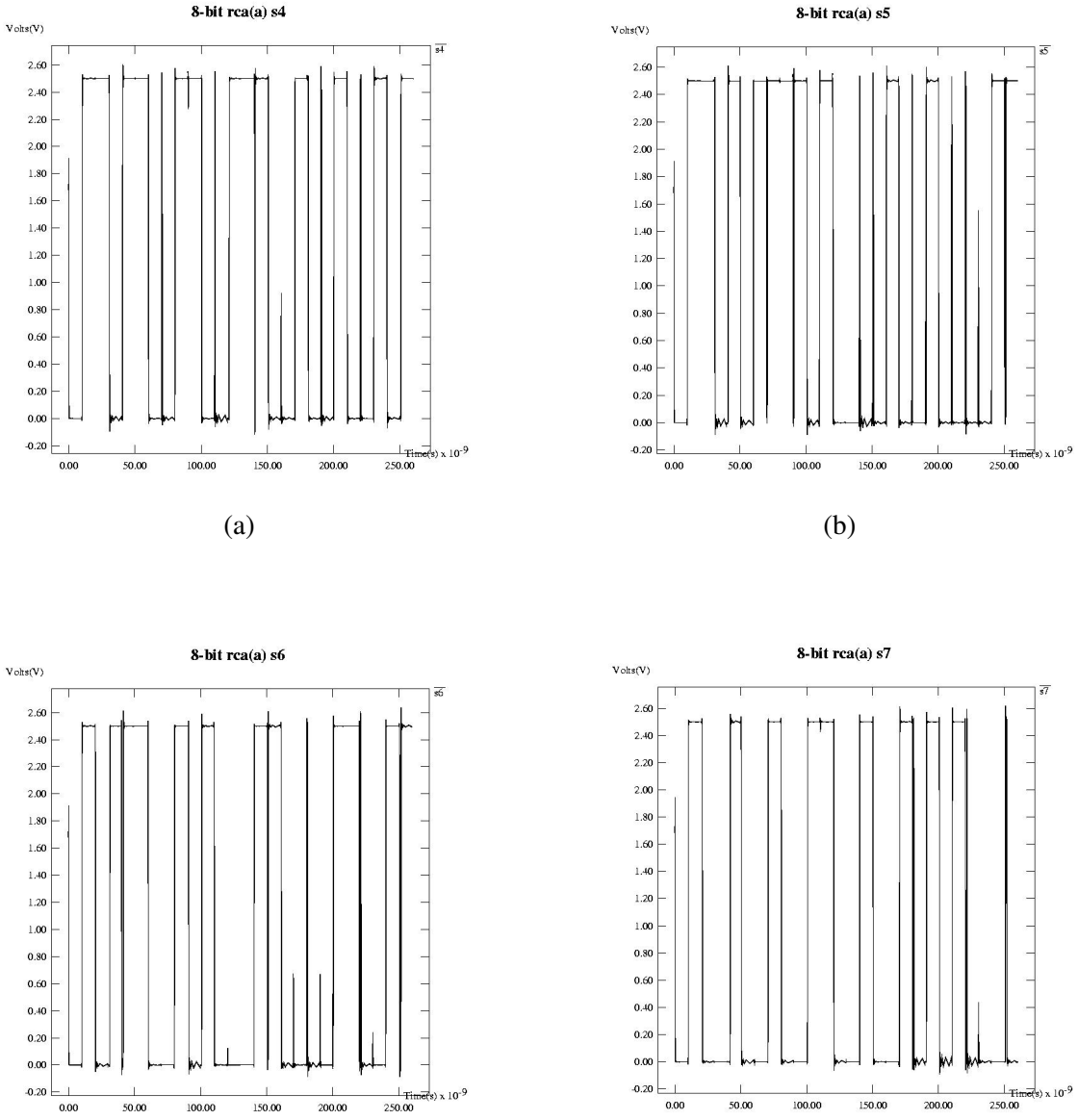
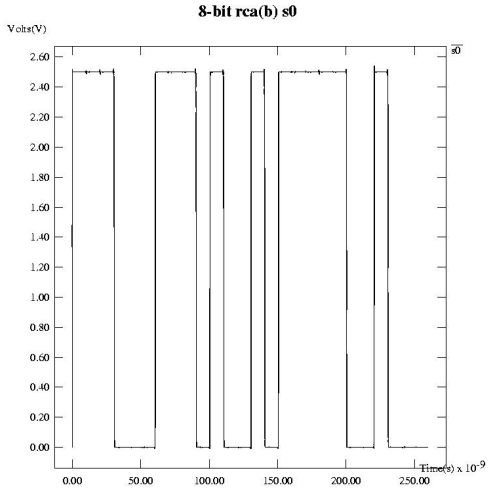
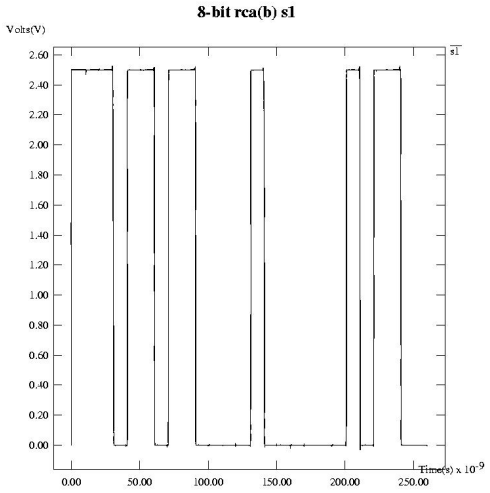


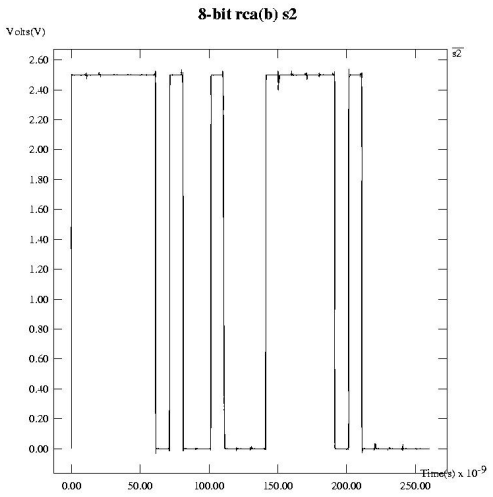

Figure 5.15: 8-bit regular ripple carry adder outputs: (a) $s4$ (b) $s5$ (c) $s6$ (d) $s7$



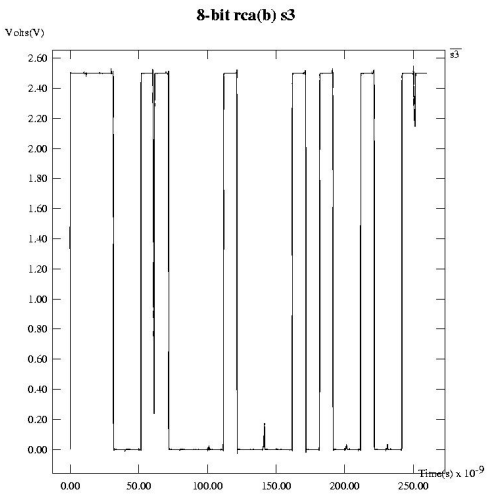

(a)




(b)




(c)




(d)

Figure 5.16: 8-bit ckt4 ripple carry adder outputs: (a) *s*0 (b) *s*1 (c) *s*2 (d) *s*3

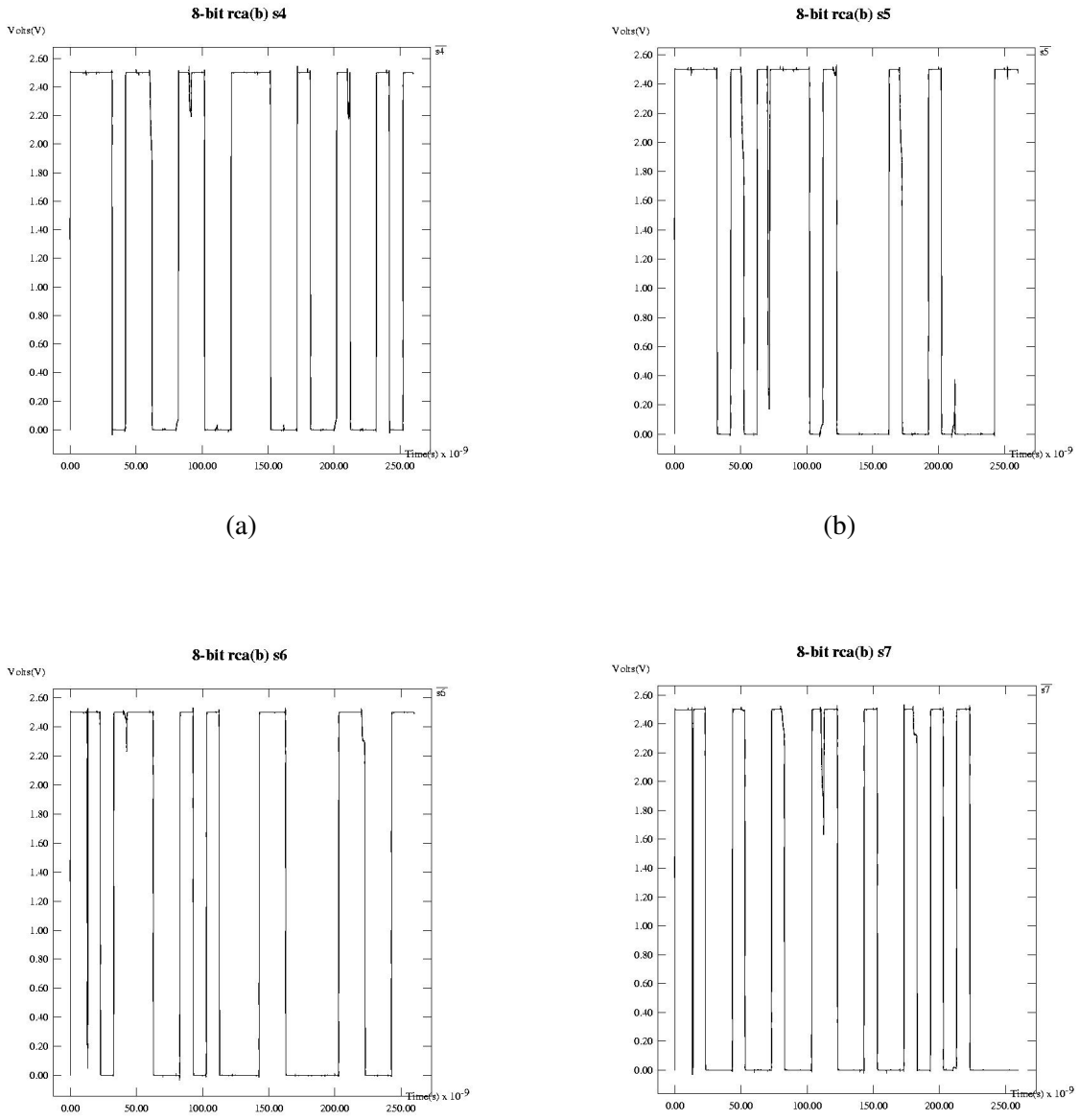



Figure 5.17: 8-bit ckt4 ripple carry adder outputs: (a) *s*4 (b) *s*5 (c) *s*6 (d) *s*7

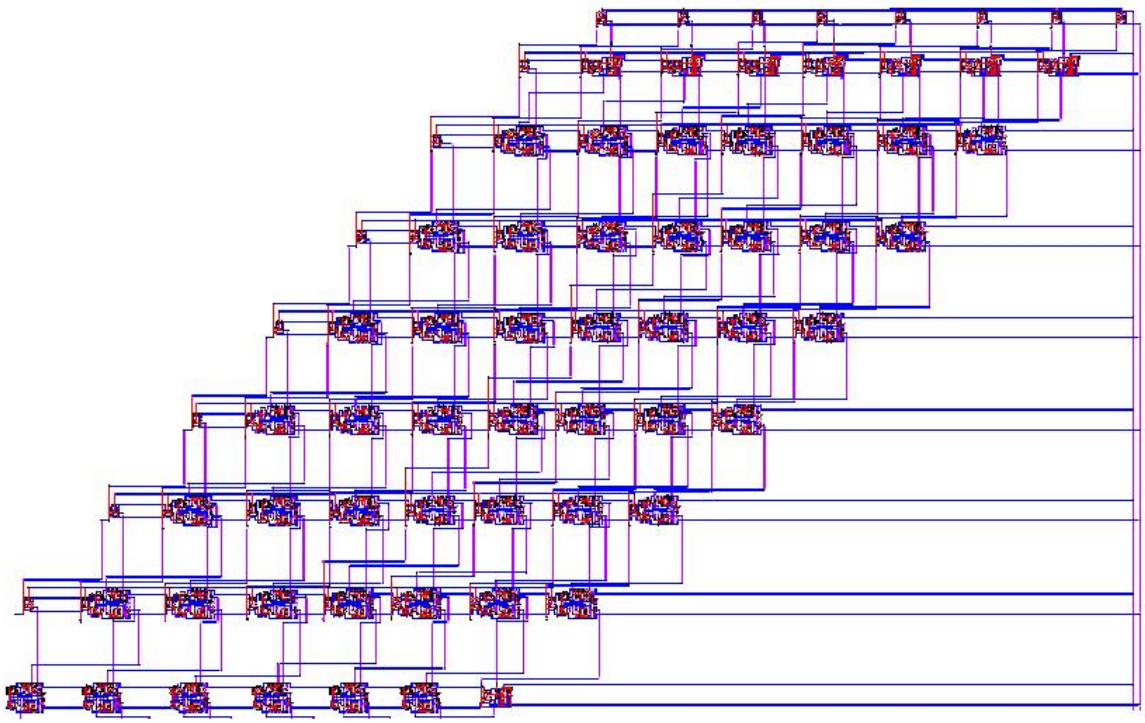


Figure 5.18: mul(a): regular carry save multiplier

#### 5.4.2.2 8-bit Carry Save Multiplier

The following is the signal label information used in the MAGIC layouts for the regular(Figure 5.18) and ckt4(Figure 5.19) carry save multipliers.

inputs: $a0 \ldots a7$, $b0 \ldots b7$

outputs: $s0 \ldots s15$

adds: $(a0 \ldots a7) \times (b0 \ldots b7) = (s0 \ldots s15)$

input data: 25 random test vectors

The waveforms for the input signals $a0 \ldots a7$ are described in Figures 5.20 and 5.21. Similarly, the input signals applied to $b0 \ldots b7$ are presented in Figures 5.22 and 5.23. Following these graphs are the results of the multiplication of these input signals for the regular carry save multiplier, mul(a), and the ckt4 carry save multiplier, mul(b). The multiplication results for the regular carry

save multiplier are outlined in Figures 5.24, 5.25, 5.26 and 5.27, namely $s0 \ldots s15$. Also, the solution results of the ckt4 carry save multiplier are presented here, in Figures 5.28, 5.29, 5.30 and 5.31. Note that the same output signal labels are used, $s0 \ldots s15$. The following is a specific breakdown of the various signals and their corresponding Figure allocations.

Figure 5.20: $a0 \ldots a3$

Figure 5.21: $a4 \ldots a7$

Figure 5.22: $b0 \ldots b3$

Figure 5.23: $b4 \ldots b7$

Figure 5.24: mul(a): $s0 \ldots s3$

Figure 5.25: mul(a): $s4 \ldots s7$

Figure 5.26: mul(a): $s8 \ldots s11$

Figure 5.27: mul(a): $s12 \ldots s15$

Figure 5.28: mul(b): $s0 \ldots s3$

Figure 5.29: mul(b): $s4 \ldots s7$

Figure 5.30: mul(b): $s8 \ldots s11$

Figure 5.31: mul(b): $s12 \ldots s15$

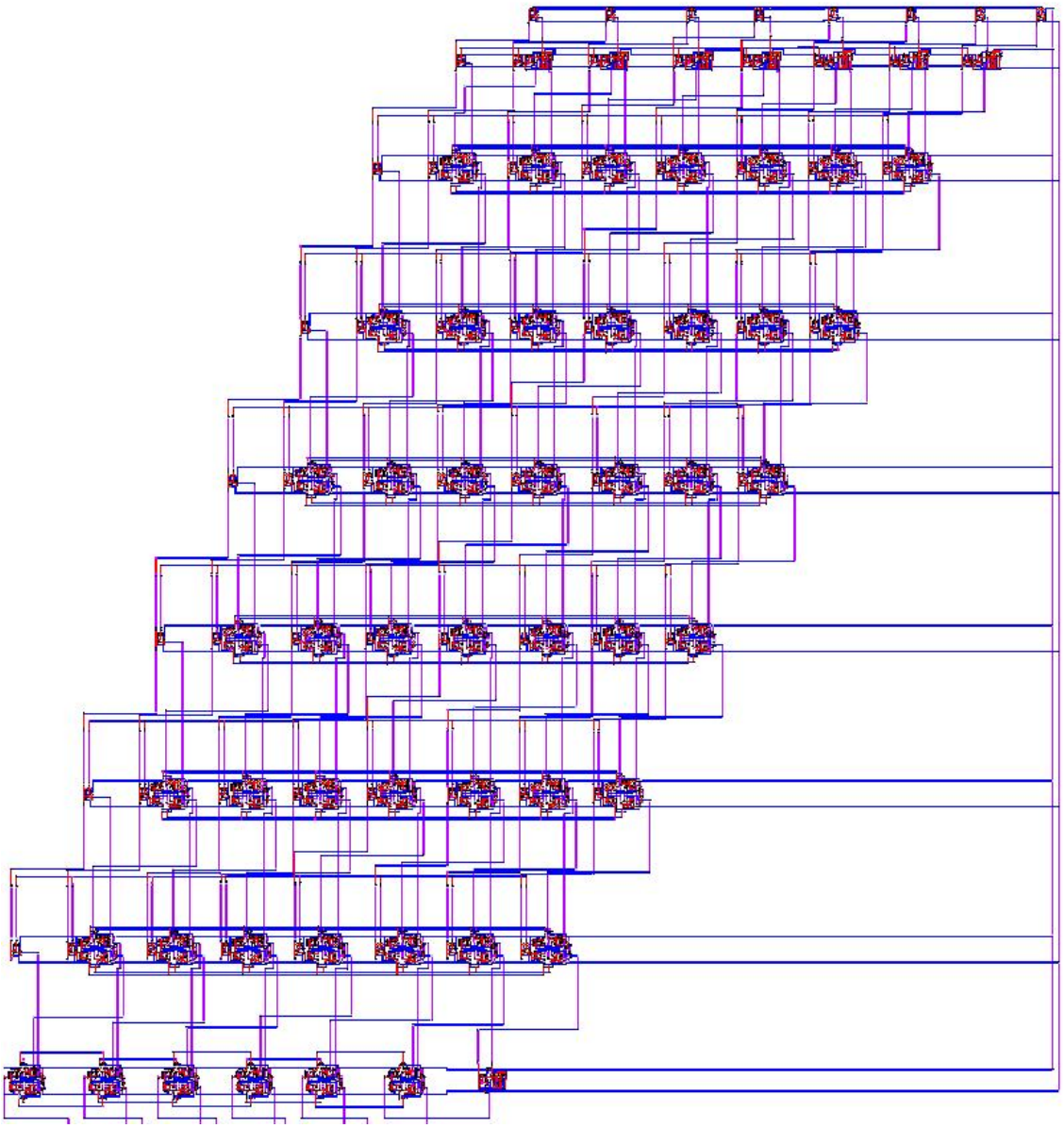


Figure 5.19: mul(b): ckt4 carry save multiplier

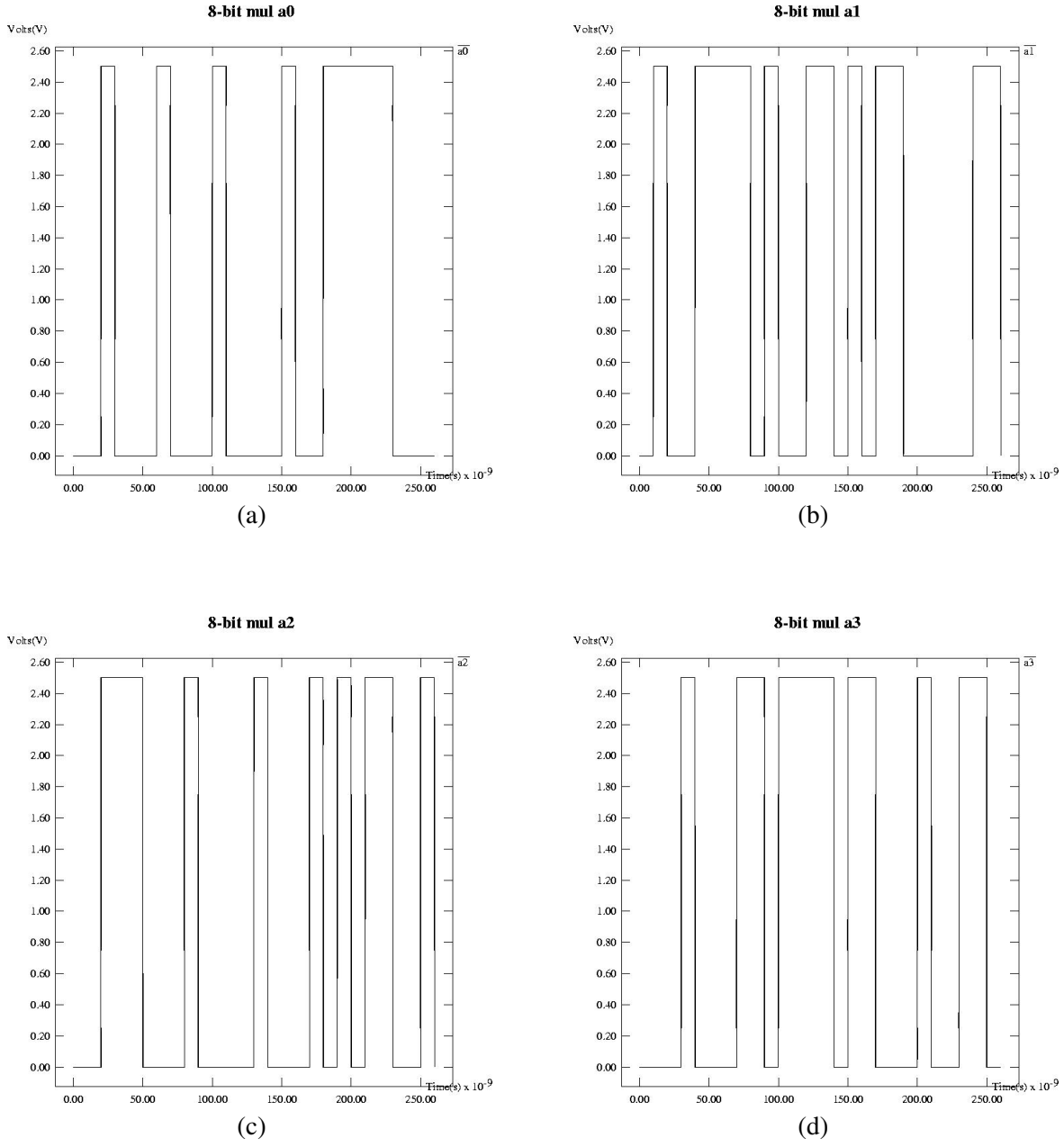



Figure 5.20: 8-bit carry save multiplier inputs: (a) $a0$ (b) $a1$ (c) $a2$ (d) $a3$

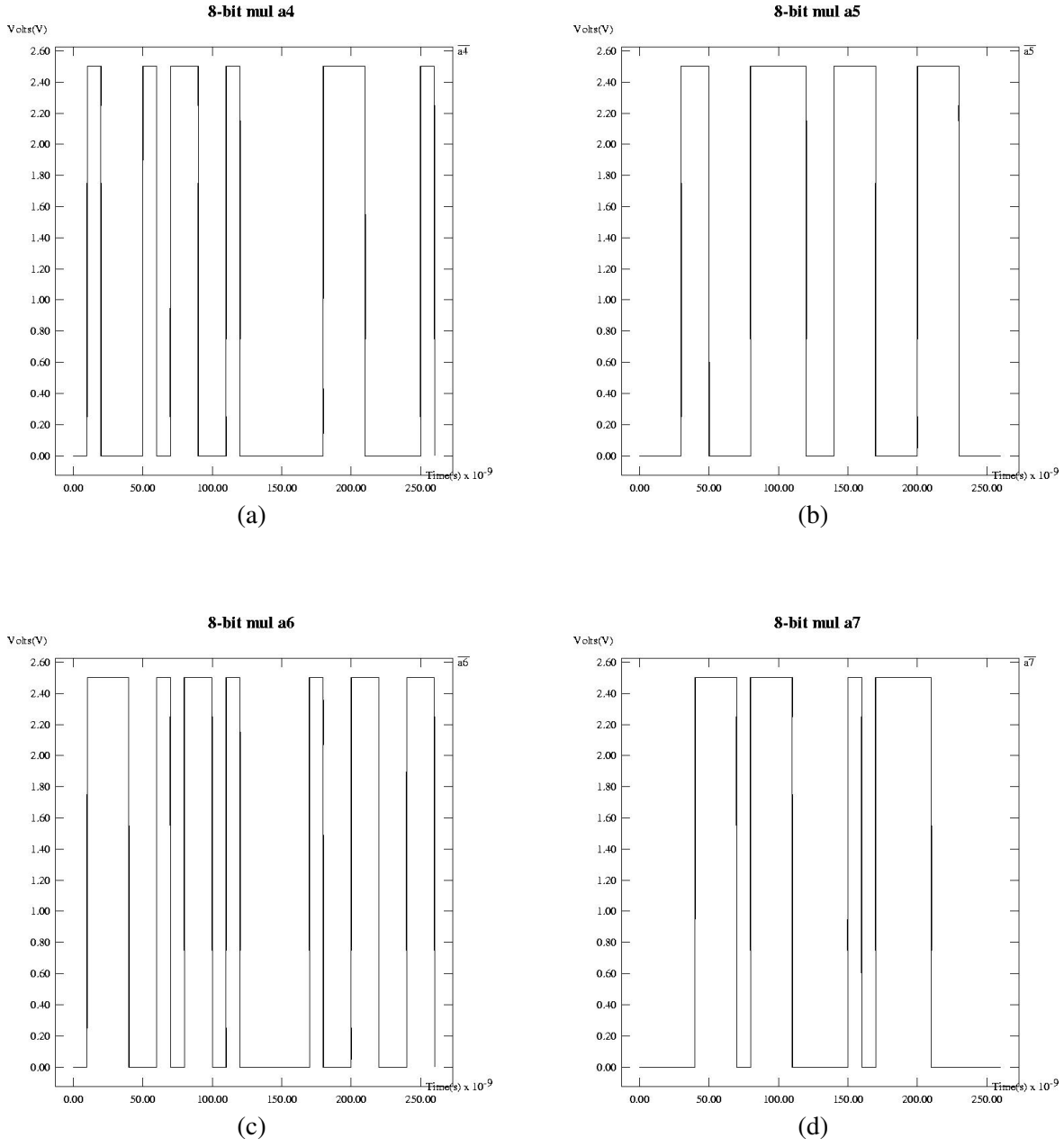



Figure 5.21: 8-bit carry save multiplier inputs: (a) $a4$ (b) $a5$ (c) $a6$ (d) $a7$

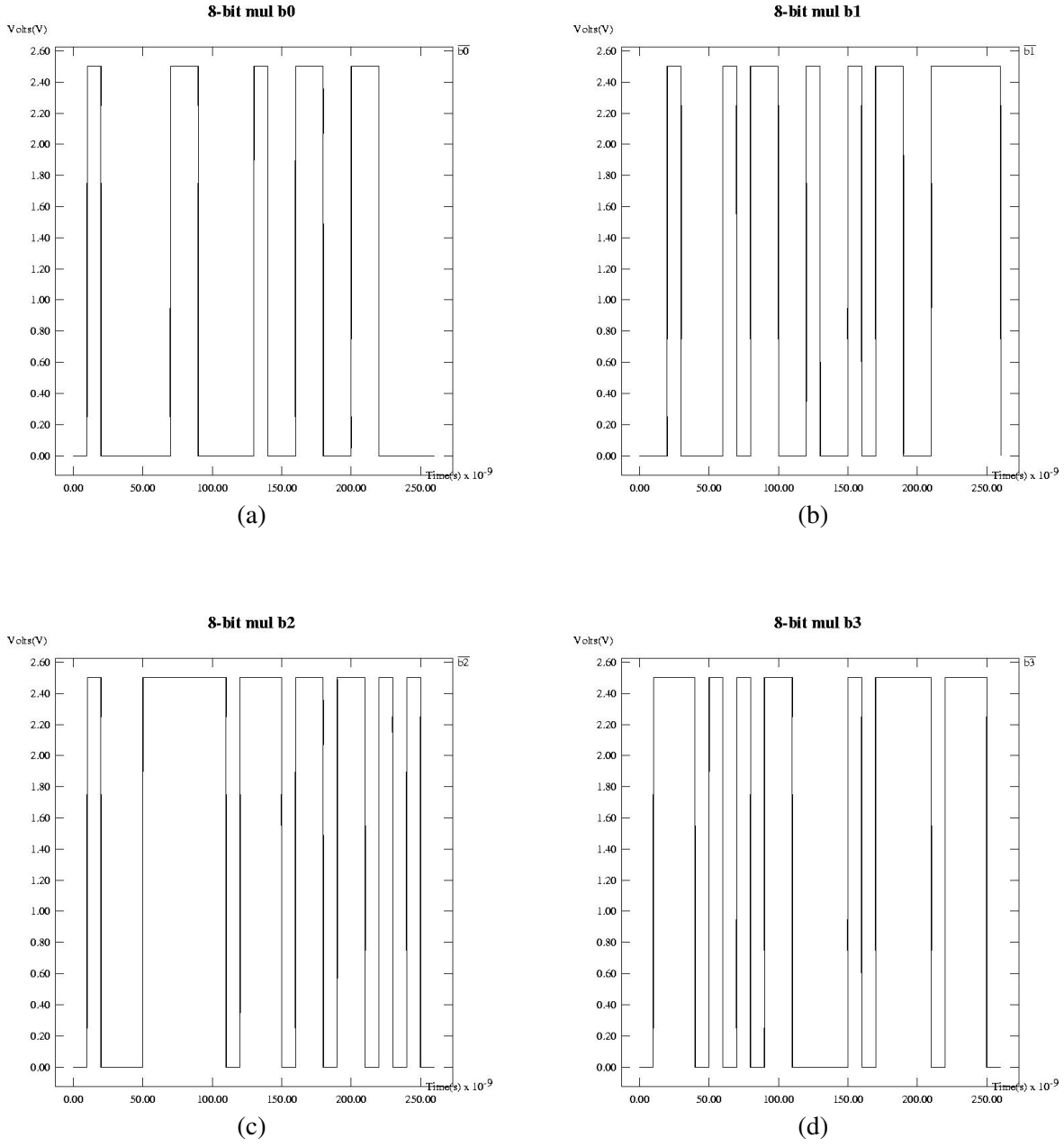



Figure 5.22: 8-bit carry save multiplier inputs: (a) $b0$ (b) $b1$ (c) $b2$ (d) $b3$

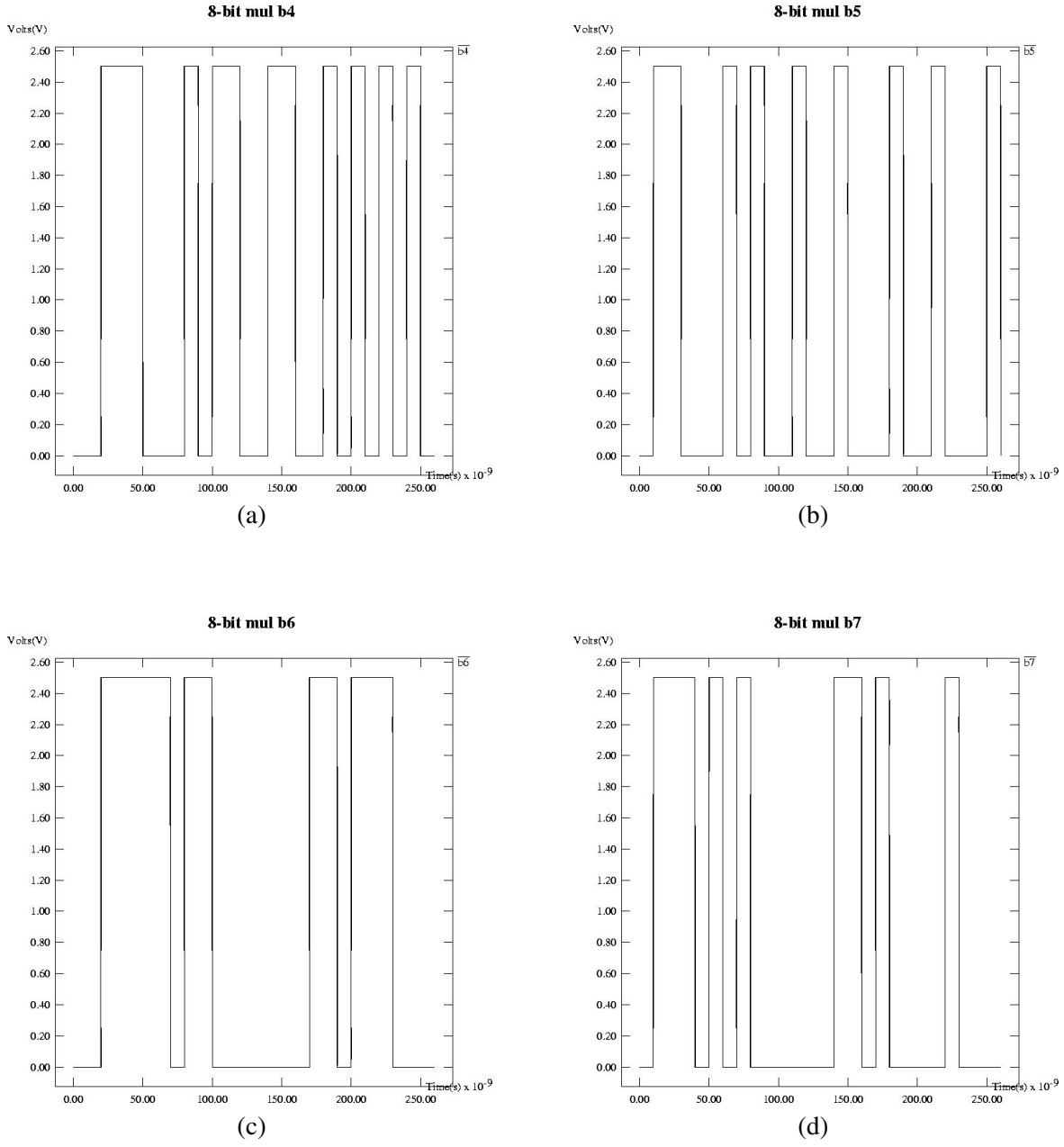



Figure 5.23: 8-bit carry save multiplier inputs: (a) $b4$ (b) $b5$ (c) $b6$ (d) $b7$

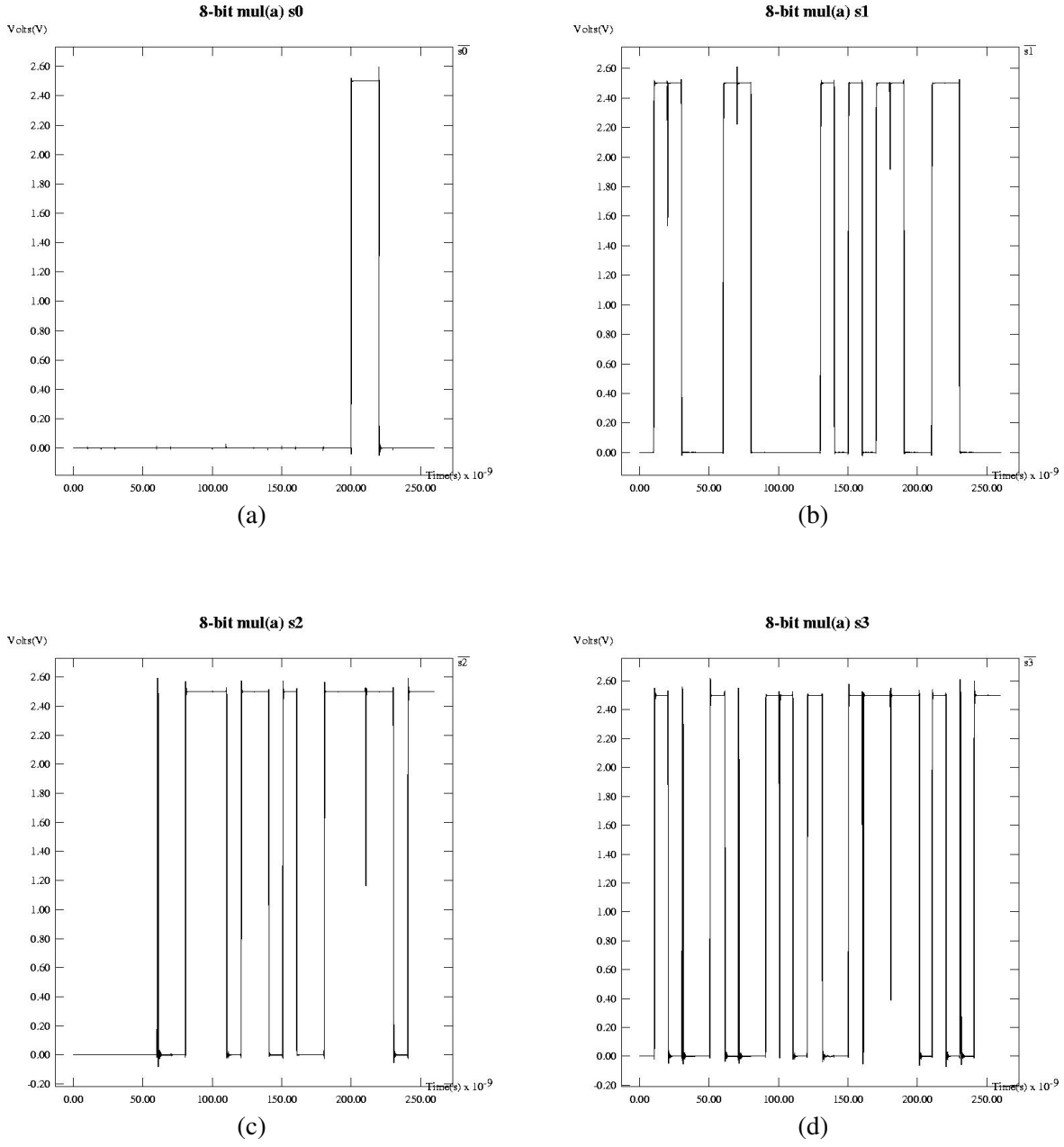



Figure 5.24: 8-bit regular carry save multiplier outputs: (a) $s0$ (b) $s1$ (c) $s2$ (d) $s3$

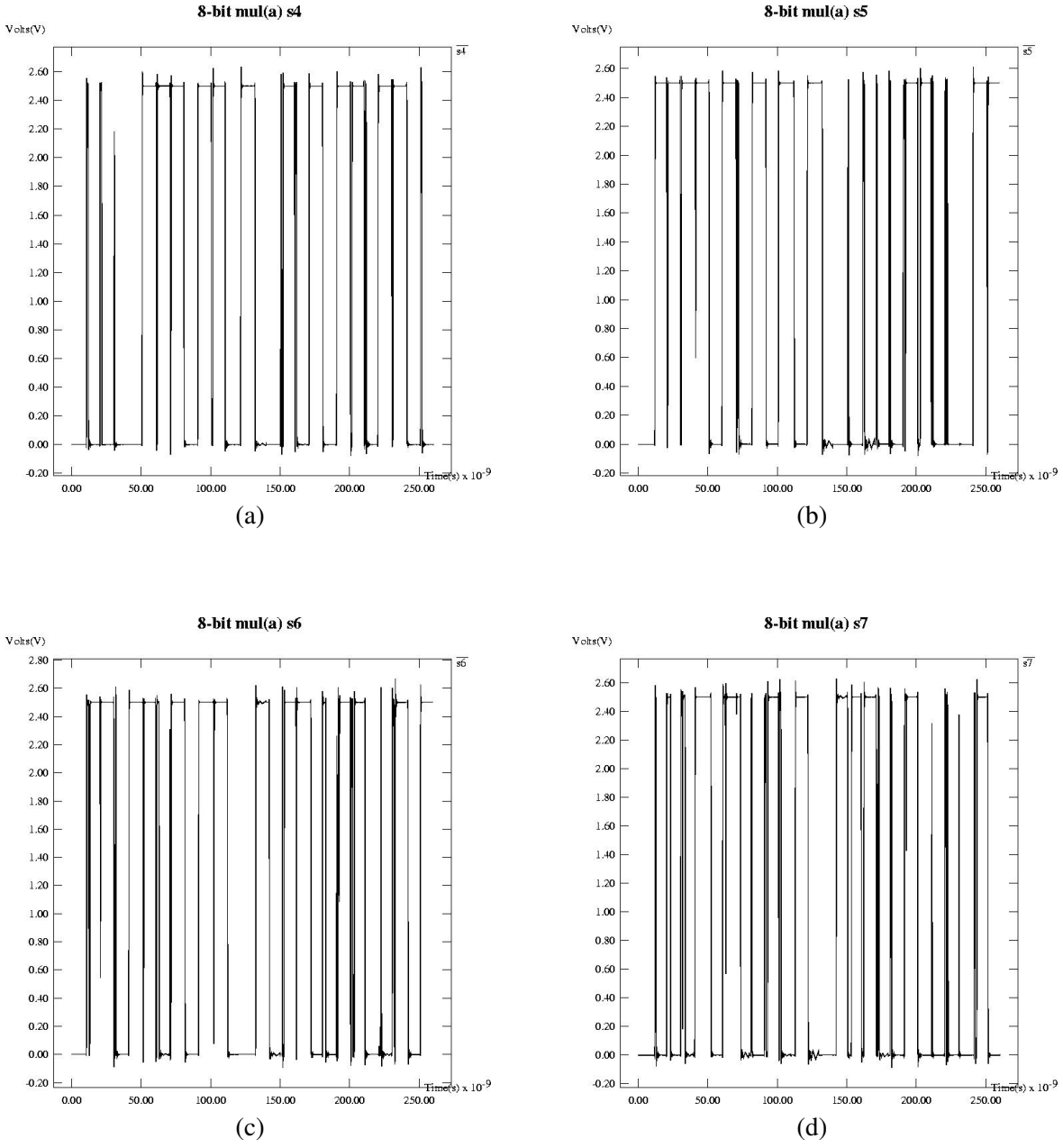



Figure 5.25: 8-bit regular carry save multiplier outputs: (a) $s4$ (b) $s5$ (c) $s6$ (d) $s7$

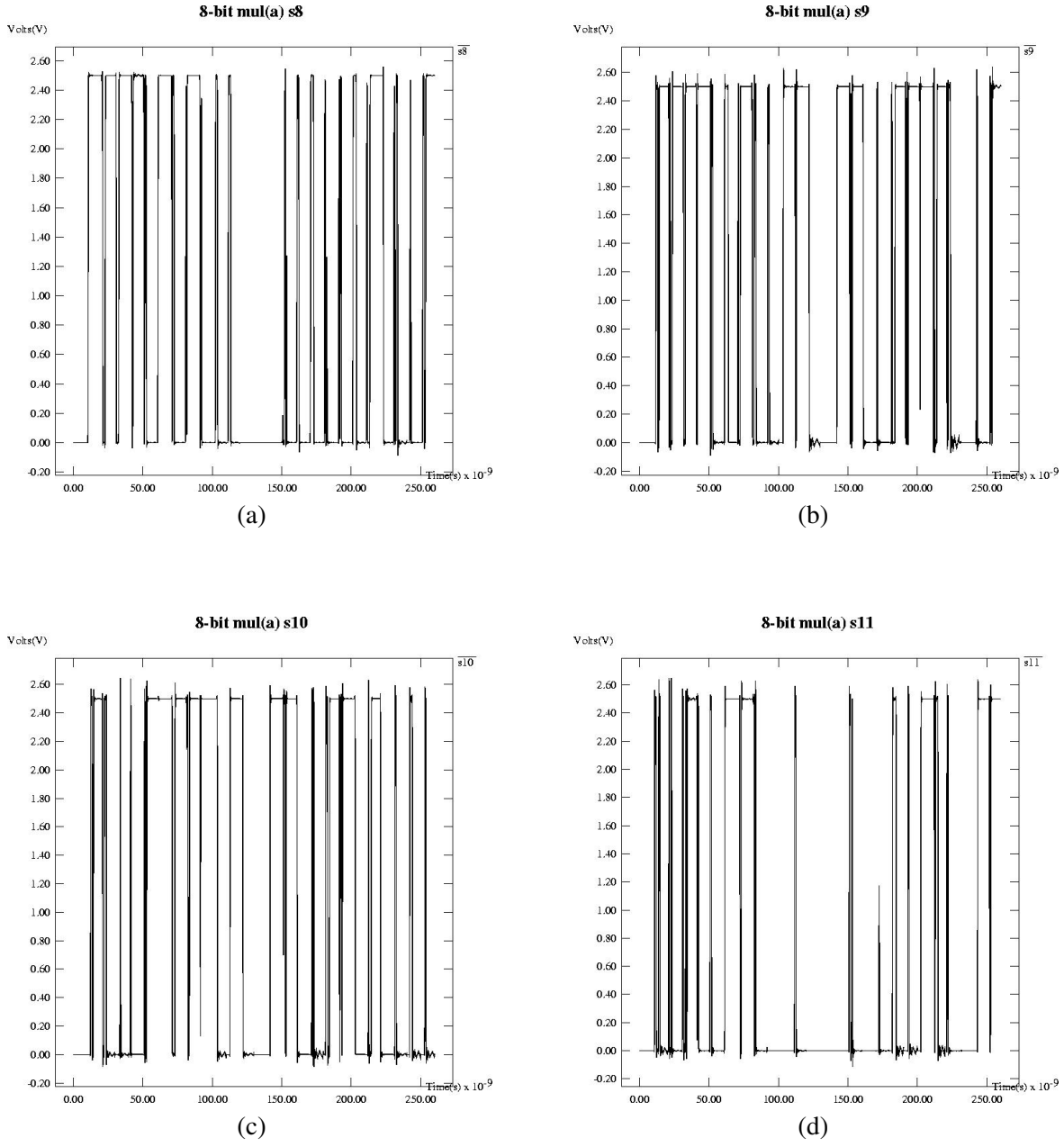



Figure 5.26: 8-bit regular carry save multiplier outputs: (a) $s8$ (b) $s9$ (c) $s10$ (d) $s11$

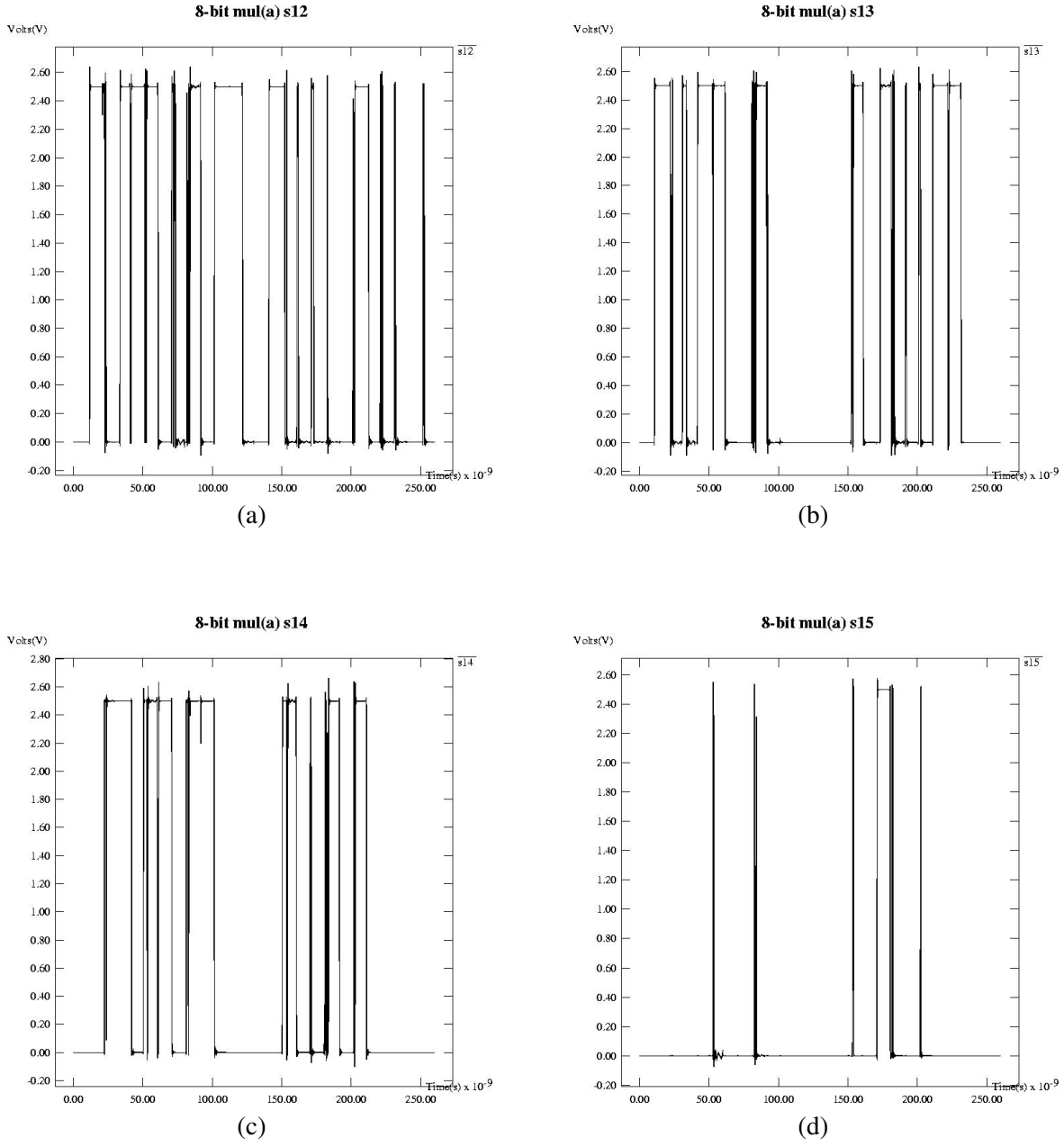



Figure 5.27: 8-bit regular carry save multiplier outputs: (a) $s12$ (b) $s13$ (c) $s14$ (d) $s15$

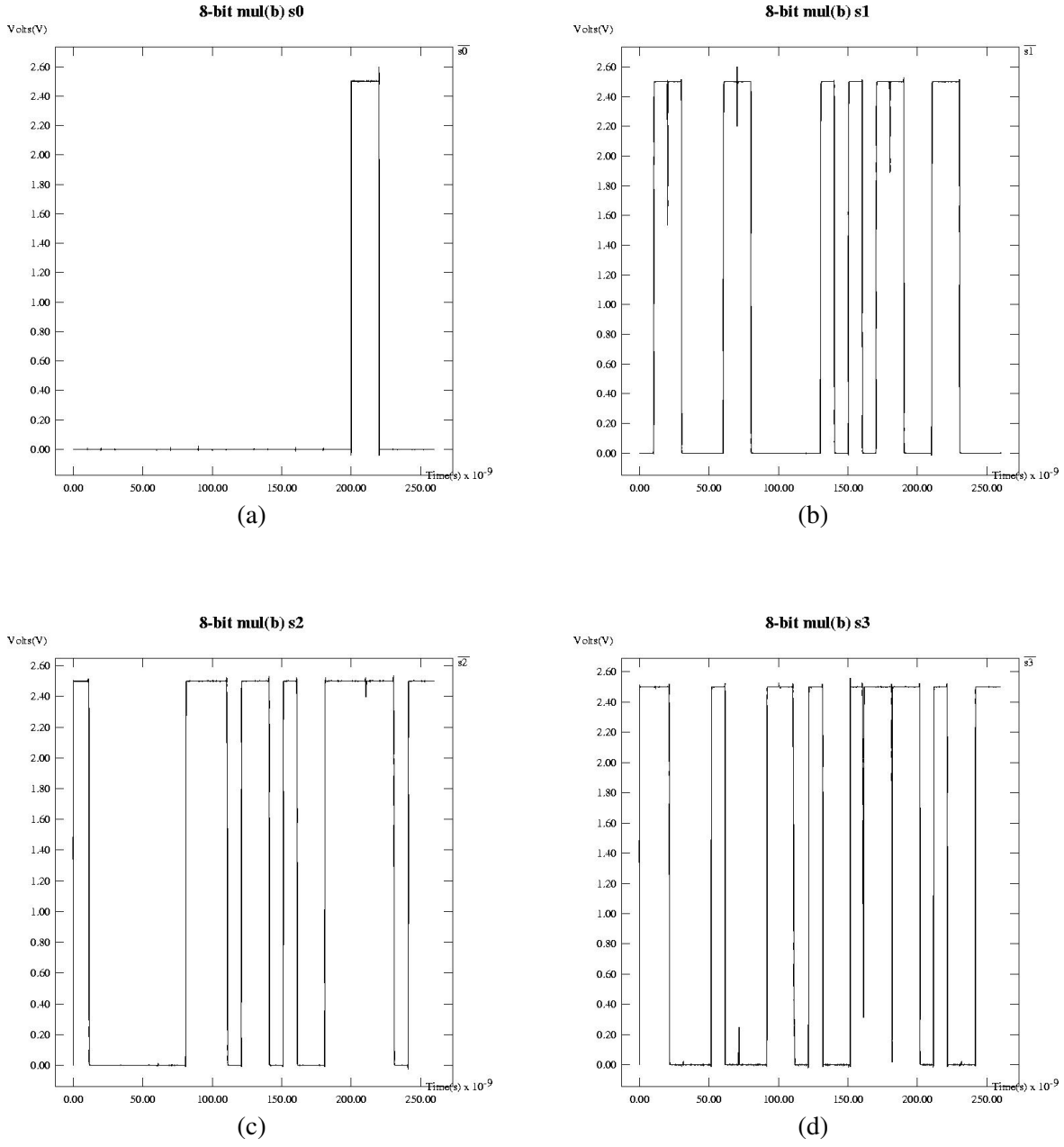



Figure 5.28: 8-bit ckt4 carry save multiplier outputs: (a) $s0$ (b) $s1$ (c) $s2$ (d) $s3$

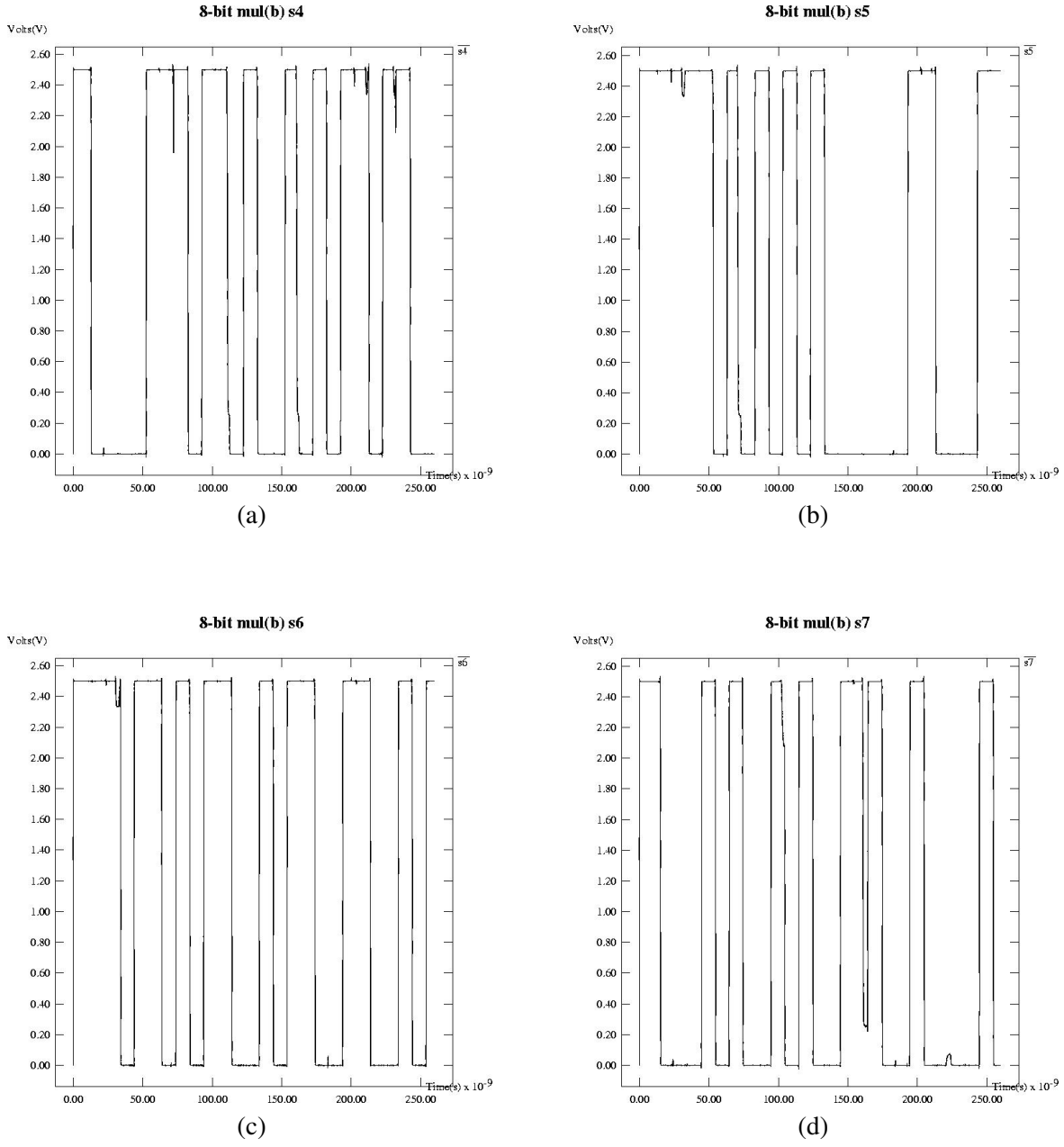



Figure 5.29: 8-bit ckt4 carry save multiplier outputs: (a) *s*4 (b) *s*5 (c) *s*6 (d) *s*7

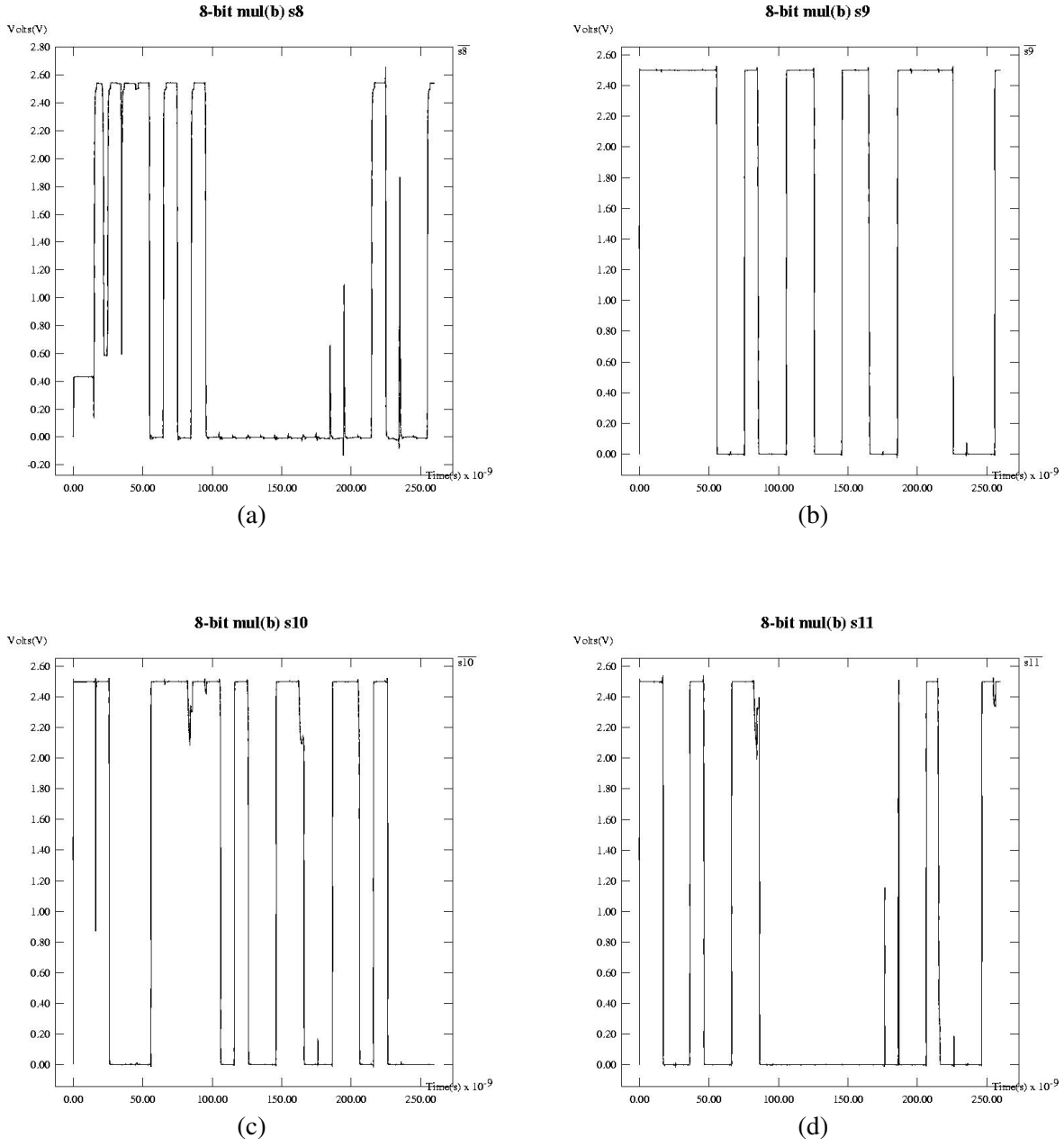



Figure 5.30: 8-bit ckt4 carry save multiplier outputs: (a) *s*8 (b) *s*9 (c) *s*10 (d) *s*11

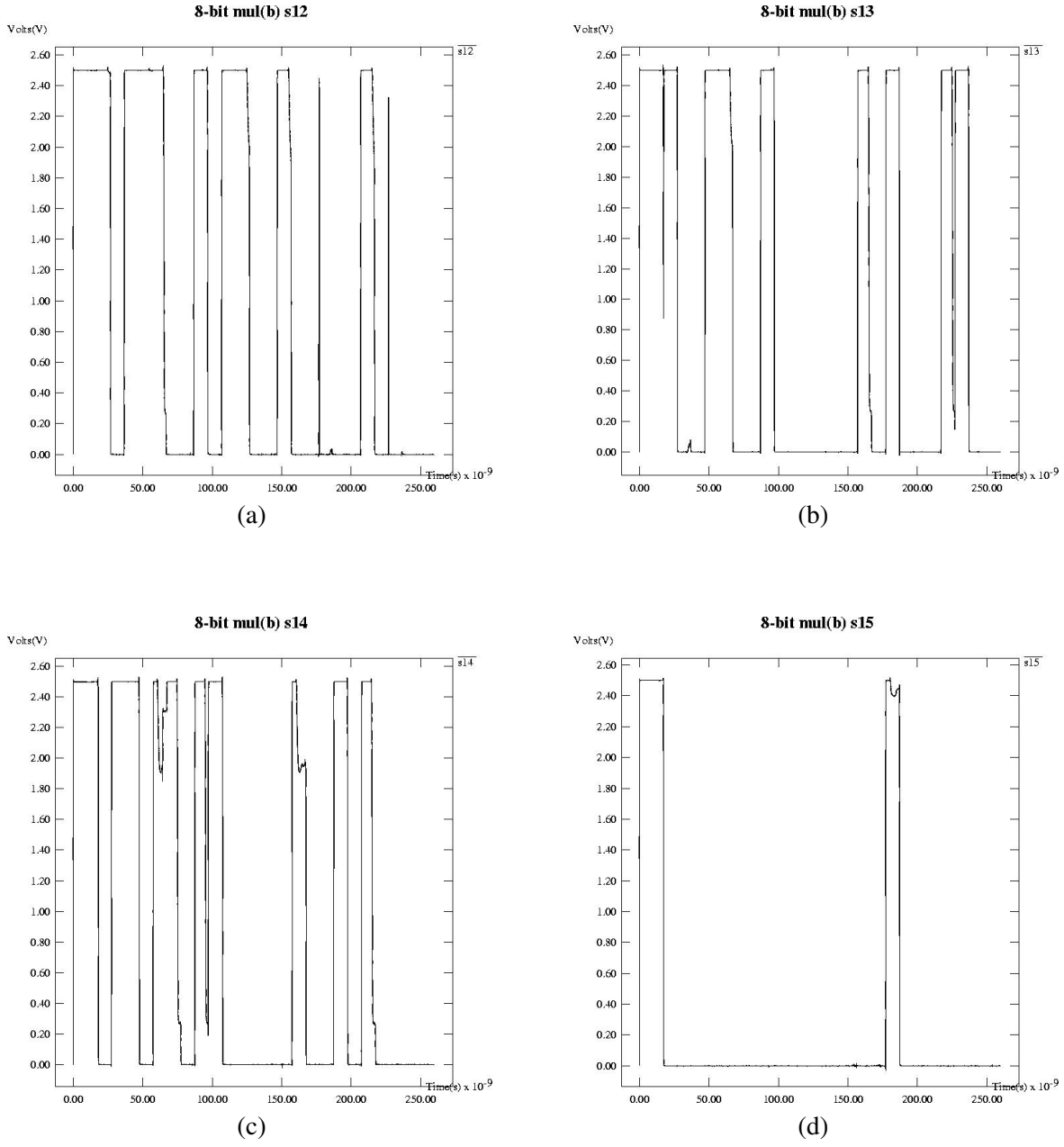



Figure 5.31: 8-bit ckt4 carry save multiplier outputs: (a) $s12$ (b) $s13$ (c) $s14$ (d) $s15$

# Chapter 6

# Comparisons, Conclusions and Future Work

## 6.1 Comparing the ckt4 Technique with Current Glitch Reduction Techniques

Our research only encompassed the use of the ckt4 glitch minimization technique on two arithmetic unit architectures. For the ripple carry adder, the results were less than stellar - as the bit width of the adder was increased, we saw a marked increase in the amount of power it consumed, and the application of a glitch minimization technique only exasperated this effect. This was because while the power consumption was increasing exponentially, the power suppressing component was only increasing linearly. The carry save multiplier was a more interesting case. The 4 bit implementation showed a small power increase when compared to the original circuit. However, when the 8 bit implementation was simulated, it showed a large power savings of 21.56%. Extrapolating from this result, I have reason to believe that an even larger implementation, such as 16 bits, would show

an even greater improvement in power savings. These results show that the ckt4 technique is most successful when applied to large, complex circuits. Other researchers mentioned in this thesis use benchmark circuits for their testing purposes, which weren't available to me during this research phase. Application of the ckt4 technique to these circuits would not only provide better results, but would allow it to compete at the same standard as the other glitch reduction techniques.

### 6.1.1 Circuit Level

Of the techniques discussed in Chapter 2, three provided explicit power simulation results as evidence of their glitch reduction theories. Their results are listed in the following tables. In Table 6.1, the effect of targeted gate resizing on a number of benchmark circuits is investigated. Table 6.2 demonstrates the validity of the guarded evaluation technique on several arithmetic circuits. Then, the results of the RTL technique are shown in Table 6.3, applied to a series of circuits commonly used in industrial applications. Each of these glitch reduction techniques are compared with each other and the ckt4 technique in Table 6.4.

| **Circuit** | **# of Gates** | **Power Reduction (%)** |
|---|---|---|
| C3540 | 525 | 10.0 |
| C5315 | 721 | 15.7 |
| Apex3 | 750 | 14.2 |
| Pair | 841 | 4.1 |
| Alu4 | 889 | 14.4 |
| Ex5p | 986 | 31.2 |
| C7552 | 1098 | 7.5 |
| I10 | 1165 | 7.7 |
| Misex3 | 1165 | 16.2 |
| Apex2 | 1254 | 14.7 |
| Seq | 1374 | 16.1 |
| Des | 1608 | 4.0 |
| **Average** | **-** | **13.0** |

Table 6.1: Gate resizing

| Circuit | Power Reduction (%) |
|---|---|
| Mux | -22.1 |
| X3 | -8.0 |
| Frg1 | -7.4 |
| Pcler8 | -0.8 |
| Frg2 | 1.6 |
| Des | 5.3 |
| Term1 | 8.1 |
| Pcle | 8.3 |
| Misex3 | 11.2 |
| Duke2 | 15.9 |
| Dalu | 20.8 |
| TooLarge | 26.5 |
| Sao2 | 37.6 |
| Sao2-hdl | 51.4 |
| **Average** | **10.6** |

Table 6.2: Guarded evaluation

| Circuit | Power Reduction (%) |
|---|---|
| GCD | 17.27 |
| Barcode | 17.42 |
| UAV | 26.25 |
| Vendor | 26.22 |
| Graphics | 30.06 |
| X.25 | 19.36 |
| DotProd | 21.23 |
| **Average** | **22.54** |

Table 6.3: RTL technique

## 6.2 Conclusions

### 6.2.1 Glitch minimization techniques

The ckt4 technique provided a substantial amount of glitch power reduction with one of the least invasive application methodologies. These qualities made it suitable for testing in an CAD application scenario. The results show that it provides the practical glitch reduction in an arithmetic unit as I had theoretically predicted. Another suitable candidate for further testing would be the ckt2

| Comparison Parameters(Rank) | Gate resizing | Guarded evaluation | RTL technique | ckt4 technique |
|---|---|---|---|---|
| Area overhead | 1 | 3 | 2 | 4 |
| Effect on delay/performance | 4 | 2 | 3 | 1 |
| Power savings | 2 | 3 | 1 | 4 |
| Ease of application | 1 | 2 | 4 | 3 |

Table 6.4: Comparison with other glitch minimization techniques

technique. It has similar attributes to ckt4, and is even less invasive, making use of a single control transistor. However, this tradeoff may be exposed in a practical setting, because it has a fundamental quality of allowing 0 to 1 glitches to pass through.

### 6.2.2 Delay elements

The best delay element depends largely on the target application scenario. Each delay element has its own range of strengths and weaknesses. For general delay requirements, the n-voltage controlled would be the most suitable choice. It can provide a wide range of delay values, but it suffers from a poor signal integrity quotient. If a large delay is required at any cost, the np-voltage controlled with Schmitt trigger delay element provides a substantial amount with an excellent signal integrity. However, this excellence comes at a cost - it's also the most expensive to implement in terms of power requirements.

### 6.2.3 Application of the ckt4 glitch minimization technique on arithmetic units

When applied to a ripple carry adder, the glitch minimization structure didn't produce any significant power savings. This is primarily due to the structure of the ripple carry adder. It's a linear circuit, so there's limited opportunities for glitches to form. Also, the amount of power and area devoted to the glitch minimization overhead is high, when compared to the rest of the circuit. The carry

save multiplier is a much better candidate for application of glitch reducing methods. Its structure is array-based, meaning that there are many different signal paths of varying lengths. They provide sufficient breeding grounds for glitch production. Another advantage is that the effect of one control gate can be spread out over several gates that need to be switched simultaneously. This keeps the glitch reducing overhead low, and that savings is reflected in the overall power budget.

## 6.3 Future Work

The first logical step is to expand the bit-scope of the carry save multiplier, that is, to implement it in larger bit widths such as 16 and 32 bits. The practical implementation of these larger circuits would be more easily facilitated with better CAD tools, since there are nodal limitations in SPICE 3. The next step is to research and build different types of circuits that are good candidates for application of the glitch minimization technique. The application methodology would be the same as that outlined in this thesis, where an original circuit is compared to a ckt4-modified circuit to determine power savings. In order to determine more accurate power simulations, we need to include actual delay elements that regulate the control signals in the experimental circuits.

# Appendix A

# Verilog

## A.1 Verilog code used to simulate the 4-bit ripple carry adder (rca4borig.v)

```
module rcadd(sum,cout,w1,w6,a,b,cin);
input a,b,cin;
output sum,cout,w1,w6;
supply1 vdd;
supply0 gnd;
trior w0,w3,w7,w9,w10;
triand w2,w4,w8,w11,w12;
wire w1a,w1,w6,w6a,w13,w14,sum,cout;

tranif0 p0(vdd,w0,a);
tranif0 p1(vdd,w0,b);
tranif0 p2(w0,w1a,cin);
buf #2 b1(w1,w1a);
tranif0 p3(vdd,w3,b);
tranif0 p4(w3,w1a,a);
tranif0 p5(vdd,w7,a);
tranif0 p6(vdd,w7,b);
tranif0 p7(vdd,w7,cin);
tranif0 p8(w7,w6,w1);
tranif0 p9(vdd,w10,a);
tranif0 p10(w10,w9,b);
tranif0 p11(w9,w6,cin);
buf #2 b2(w6a,w6);
```

```
tranif0 p12(vdd,sum,w6a);
tranif0 p13(vdd,cout,w1);

tranif1 n0(w2,gnd,a);
tranif1 n1(w2,gnd,b);
tranif1 n2(w1a,w2,cin);
tranif1 n3(w4,gnd,b);
tranif1 n4(w1a,w4,a);
tranif1 n5(w8,gnd,a);
tranif1 n6(w8,gnd,b);
tranif1 n7(w8,gnd,cin);
tranif1 n8(w6,w8,w1);
tranif1 n9(w12,gnd,a);
tranif1 n10(w11,w12,b);
tranif1 n11(w6,w11,cin);
tranif1 n12(sum,gnd,w6a);
tranif1 n13(cout,gnd,w1);
endmodule

module rcastim;
reg[3:0] a,b;
reg ci;

wire[3:0] sum;
wire co;

rcadd bit1(sum[0],c0,b1w1,b1w6,a[0],b[0],ci);
rcadd bit2(sum[1],c1,b2w1,b2w6,a[1],b[1],c0);
rcadd bit3(sum[2],c2,b3w1,b3w6,a[2],b[2],c1);
rcadd bit4(sum[3],co,b4w1,b4w6,a[3],b[3],c2);

initial
  begin
$display("                Time ci a        b        co sum
w1 w6 c0 w1 w6 c1 w1 w6 c2 w1 w6");
$monitor("%d %b  %b %b %b  %b %b %b %b  %b  %b  %b  %b  %b
%b  %b  %b  %b  %b  %b",
$time,ci,a,b,co,sum[0],sum[1],sum[2],sum[3],b1w1,b1w6,c0,b2
w1,b2w6,c1,b3w1,b3w6,c2,b4w1,b4w6);
  end

initial
  begin
a=8'd 82 ; b=8'd 173; ci=1'b0 ;#100 $display($time);
a=8'd 69 ; b=8'd 250; ci=1'b0 ;#100 $display($time);
a=8'd 108; b=8'd 216; ci=1'b0 ;#100 $display($time);
```

```
a=8'd 166; b=8'd 80 ; ci=1'b0 ;#100 $display($time);
a=8'd 146; b=8'd 204; ci=1'b0 ;#100 $display($time);
a=8'd 195; b=8'd 102; ci=1'b0 ;#100 $display($time);
a=8'd 26 ; b=8'd 141; ci=1'b0 ;#100 $display($time);
a=8'd 252; b=8'd 119; ci=1'b0 ;#100 $display($time);
a=8'd 226; b=8'd 78 ; ci=1'b0 ;#100 $display($time);
a=8'd 169; b=8'd 28 ; ci=1'b0 ;#100 $display($time);
a=8'd 120; b=8'd 48 ; ci=1'b0 ;#100 $display($time);
a=8'd 10 ; b=8'd 6  ; ci=1'b0 ;#100 $display($time);
a=8'd 14 ; b=8'd 5  ; ci=1'b0 ;#100 $display($time);
a=8'd 32 ; b=8'd 180; ci=1'b0 ;#100 $display($time);
a=8'd 171; b=8'd 154; ci=1'b0 ;#100 $display($time);
a=8'd 40 ; b=8'd 5  ; ci=1'b0 ;#100 $display($time);
a=8'd 198; b=8'd 207; ci=1'b0 ;#100 $display($time);
a=8'd 147; b=8'd 122; ci=1'b0 ;#100 $display($time);
a=8'd 149; b=8'd 12 ; ci=1'b0 ;#100 $display($time);
a=8'd 249; b=8'd 93 ; ci=1'b0 ;#100 $display($time);
a=8'd 101; b=8'd 99 ; ci=1'b0 ;#100 $display($time);
a=8'd 37 ; b=8'd 222; ci=1'b0 ;#100 $display($time);
a=8'd 8  ; b=8'd 10 ; ci=1'b0 ;#100 $display($time);
a=8'd 74 ; b=8'd 30 ; ci=1'b0 ;#100 $display($time);
a=8'd 86 ; b=8'd 34 ; ci=1'b0 ;#100 $display($time);
  end
endmodule
```

## A.2 Verilog code used to simulate the 4-bit carry-save multiplier (mul4borig.v)

```
module fulladder(a, b, cin, sum, cout, n1, n2, n3);
//declare fulladder
input a, b, cin; //declare inputs
output sum, cout, n1, n2, n3; //declare outputs
wire sum, cout, n1, n2, n3;
//declare data type of outputs sum,cout
xor #1(n1, a, b); //n1=a xor b
xor #1(sum, n1, cin); //sum=n1 xor cin
and #1(n2, a, b); //n2=a.b
and #1(n3, n1, cin); //n3=n1.cin
or #1(cout, n2, n3);//cout=n2 + n3
endmodule //end of the fulladder module

module fulladdera(a, x, y, cin, sum, cout, n1, n2, n3, b);
//declare fulladdera
```

```
input a, x, y, cin; //declare inputs
output sum, cout, n1, n2, n3, b; //declare outputs
wire sum, cout, n1, n2, n3, b;
//declare data type of outputs sum,cout
xor #1(n1, a, b); //n1=a xor b
xor #1(sum, n1, cin); //sum=n1 xor cin
and #1(b, x, y); //b=x.y
and #1(n2, a, b); //n2=a.b
and #1(n3, n1, cin); //n3=n1.cin
or #1(cout, n2, n3);//cout=n2 + n3
endmodule //end of the fulladdera module

module halfadder(a, b, sum, cout);
//declare halfadder as the module name
input a, b; //declare inputs
output sum, cout; //declare outputs
wire sum, cout;
//declare data type of outputs sum,cout
xor #1(sum, a, b); //sum=a xor b
and #1(cout, a, b); //cout=a.b
endmodule //end of the halfadder module

module halfaddera(a, x, y, sum, cout, b);
//declare halfaddera
input a, x, y; //declare inputs
output sum, cout, b; //declare outputs
wire sum, cout, b;
//declare data type of outputs sum,cout
xor #1(sum, a, b); //sum=a xor b
and #1(b, x, y); //b=x.y
and #1(cout, a, b); //cout=a.b
endmodule //end of the halfaddera module

module multstim01; //declare stimulus01 as the module name
reg[3:0] x, y; //declare x, y as 4 bit wide variables
wire[7:0] z; //declare data type of 8 bit wide output z

//level 0
and #1(z[0], x[0], y[0]);
and #1(s01, x[1], y[0]);
and #1(s02, x[2], y[0]);
and #1(s03, x[3], y[0]);
//the modules halfadder(), halfaddera(),
//fulladder(), and fulladdera()
//are used to form instances
//ha, haa, fa, faa of the multiplier
```

```
//level 1
halfaddera haa01(s01, x[0], y[1], z[1], c10, b01);
halfaddera haa11(s02, x[1], y[1], s11, c11, b11);
halfaddera haa21(s03, x[2], y[1], s12, c12, b21);
and #1(s13, x[3], y[1]);

//level 2
fulladdera
faa02(s11,x[0],y[2],c10,z[2],c20,f02n1,f02n2,f02n3,b02);
fulladdera
faa12(s12,x[1],y[2],c11,s21,c21,f12n1,f12n2,f12n3,b12);
fulladdera
faa22(s13,x[2],y[2],c12,s22,c22,f22n1,f22n2,f22n3,b22);
and #1(s23, x[3], y[2]);

//level 3
fulladdera
faa03(s21,x[0],y[3],c20,z[3],c30,f03n1,f03n2,f03n3,b03);
fulladdera
faa13(s22,x[1],y[3],c21,s31,c31,f13n1,f13n2,f13n3,b13);
fulladdera
faa23(s23,x[2],y[3],c22,s32,c32,f23n1,f23n2,f23n3,b23);
and #1(s33, x[3], y[3]);

//level 4
//fulladder fa04(s31, c30, 0, z[4], c40, f04n1, f04n2, f04n3);
halfadder ha04(s31, c30, z[4], c40);
fulladder fa14(s32, c31, c40, z[5], c41, f14n1, f14n2, f14n3);
fulladder fa24(s33, c32, c41, z[6], z[7], f24n1, f24n2, f24n3);

initial
  begin //beginning of the initial block
//display a row of headings for the time and gate values
$display("               Time x    y    z
s01 b01 c10 s02 b11 c11 s03 b21 c12 s11 b02 n1 n2 n3 c20 s12
b12 n1 n2 n3 c21 s13 b22 n1 n2 n3 c22 s21 b03 n1 n2 n3 c30 s22
b13 n1 n2 n3 c31 s23 b23 n1 n2 n3 c32 s31 c40 s32 c41 s33");

//this statement monitors the values of each gate constantly
and displays a result whenever a value change occurs
$monitor("%d %b %b %b %b   %b   %b   %b   %b   %b   %b
%b   %b   %b   %b   %b  %b  %b  %b   %b   %b   %b  %b
%b  %b   %b   %b   %b  %b  %b  %b   %b   %b   %b  %b
%b  %b   %b   %b   %b  %b  %b  %b   %b   %b   %b  %b
%b  %b   %b   %b   %b   %b %b",
```

```
$time,x,y,z,s01,b01,c10,s02,b11,c11,s03,b21,c12,s11,b02,
f02n1,f02n2,f02n3,c20,s12,b12,f12n1,f12n2,f12n3,c21,s13,
b22,f22n1,f22n2,f22n3,c22,s21,b03,f03n1,f03n2,f03n3,c30,
s22,b13,f13n1,f13n2,f13n3,c31,s23,b23,f23n1,f23n2,f23n3,
c32,s31,c40,s32,c41,s33);
  end //end of the initial block

initial
  begin //beginning of the initial block
x=4'd 2 ; y=4'd 13 ;#100 $display($time);
x=4'd 5 ; y=4'd 10 ;#100 $display($time);
x=4'd 12 ; y=4'd 8 ;#100 $display($time);
x=4'd 6 ; y=4'd 0 ;#100 $display($time);
x=4'd 2 ; y=4'd 12 ;#100 $display($time);
x=4'd 3 ; y=4'd 6 ;#100 $display($time);
x=4'd 10 ; y=4'd 13 ;#100 $display($time);
x=4'd 12 ; y=4'd 7 ;#100 $display($time);
x=4'd 2 ; y=4'd 14 ;#100 $display($time);
x=4'd 9 ; y=4'd 12 ;#100 $display($time);
x=4'd 8 ; y=4'd 0 ;#100 $display($time);
x=4'd 10 ; y=4'd 6 ;#100 $display($time);
x=4'd 14 ; y=4'd 5 ;#100 $display($time);
x=4'd 0 ; y=4'd 4 ;#100 $display($time);
x=4'd 11 ; y=4'd 10 ;#100 $display($time);
x=4'd 8 ; y=4'd 5 ;#100 $display($time);
x=4'd 6 ; y=4'd 15 ;#100 $display($time);
x=4'd 3 ; y=4'd 10 ;#100 $display($time);
x=4'd 5 ; y=4'd 12 ;#100 $display($time);
x=4'd 9 ; y=4'd 13 ;#100 $display($time);
x=4'd 5 ; y=4'd 3 ;#100 $display($time);
x=4'd 5 ; y=4'd 14 ;#100 $display($time);
x=4'd 8 ; y=4'd 10 ;#100 $display($time);
x=4'd 10 ; y=4'd 14 ;#100 $display($time);
x=4'd 6 ; y=4'd 2 ;#100 $display($time);
  end //end of the initial block
endmodule //end of the multstim01 module
```

# Appendix B

# Spice

## B.1 Spice code used to perform power simulations (psim3)

```
Power Simulation
.control
                let watts = (vdd1#branch)*(vdd)
                let len = length(time)-1
                let watts1 = watts[ 0 , len - 1 ]
                let watts2 = watts[ 1 , len ]
                let time1 = time[ 0 , len - 1 ]
                let time2 = time[ 1 , len ]
                let power3 = (watts2+watts1)*(time2-time1)/2
                let power = mean(power3)*(len)/time[len+0]
                print power
.endcontrol
```

## B.2 Level 8 Spice MOSFET model parameters for a 0.25$\mu$m CMOS process (level25.8)

```
* LOT: n94s WAF: 08
* Temperature_parameters=Default
.MODEL nfet NMOS ( LEVEL = 8
+VERSION = 3.1 TNOM = 27 TOX = 5.8E-9
+XJ = 1E-7 NCH = 2.3549E17 VTH0 = 0.4308936
+K1 = 0.3519429 K2 = 0.0298493 K3 = 1E-3
```

```
+K3B = 0.0592323 W0 = 1E-5 NLX = 1.465901E-7
+DVT0W = 0 DVT1W = 0 DVT2W = 0
+DVT0 = 0.0183405 DVT1 = 4.897584E-3 DVT2 = -0.0252658
+U0 = 455.3033362 UA = 5.223592E-10 UB = 1.104713E-18
+UC = 3.287888E-11 VSAT = 1.050993E5 A0 = 1.2318623
+AGS = 0.3043334 B0 = 6.67749E-8 B1 = 5E-6
+KETA = 8.518046E-4 A1 = 0 A2 = 1
+RDSW = 509.5675851 PRWG = 0.0227558 PRWB = -1E-3
+WR = 1 WINT = 2.126497E-9 LINT = 4.393474E-9
+DWG = -3.409033E-9
+DWB = 2.794842E-9 VOFF = -0.1026054 NFACTOR = 0.1344887
+CIT = 0 CDSC = 1.527511E-3 CDSCD = 0
+CDSCB = 0 ETA0 = 3.48737E-3 ETAB = 4.557986E-4
+DSUB = 3.045473E-3 PCLM = 1.0446257 PDIBLC1 = 0.1441952
+PDIBLC2 = 4.513382E-4 PDIBLCB = -2.816756E-5 DROUT = 0.4698725
+PSCBE1 = 1.761109E10 PSCBE2 = 3.772916E-9 PVAG = 0.0361824
+DELTA = 0.01 MOBMOD = 1 PRT = 0
+UTE = -1.5 KT1 = -0.11 KT1L = 0
+KT2 = 0.022 UA1 = 4.31E-9 UB1 = -7.61E-18
+UC1 = -5.6E-11 AT = 3.3E4 WL = 0
+WLN = 1 WW = 0 WWN = 1
+WWL = 0 LL = 0 LLN = 1
+LW = 0 LWN = 1 LWL = 0
+CAPMOD = 2 XPART = 0.4 CGDO = 6.27E-10
+CGSO = 6.27E-10 CGBO = 0 CJ = 1.918655E-3
+PB = 0.9784049 MJ = 0.4721729 CJSW = 4.441595E-10
+PBSW = 0.9419636 MJSW = 0.2871118 PVTH0 = 1.342985E-3
+PRDSW = -61.8357222 PK2 = -3.140724E-3 WKETA = 7.512693E-4
+LKETA = -6.144062E-3 )
*

.MODEL pfet PMOS ( LEVEL = 8
+VERSION = 3.1 TNOM = 27 TOX = 5.8E-9
+XJ = 1E-7 NCH = 4.1589E17 VTH0 = -0.6158735
+K1 = 0.4598379 K2 = 0.0399415 K3 = 0
+K3B = 8.7410723 W0 = 1E-6 NLX = 1E-9
+DVT0W = 0 DVT1W = 0 DVT2W = 0
+DVT0 = 0.6249485 DVT1 = 0.203296 DVT2 = -0.0513763
+U0 = 158.67524 UA = 2.200024E-10 UB = 4.457415E-18
+UC = 1.02138E-10 VSAT = 1.85064E5 A0 = 1.3826397
+AGS = 0.4192977 B0 = 2.844099E-6 B1 = 5E-6
+KETA = 0.0208695 A1 = 0 A2 = 1
+RDSW = 968.5463 PRWG = -0.1026483 PRWB = -0.325
+WR = 1 WINT = 2.748811E-8 LINT = 8.71907E-9
+DWG = -4.087585E-8
+DWB = 2.032008E-8 VOFF = -0.15 NFACTOR = 1.5460516
```

```
+CIT = 0 CDSC = 1.413317E-4 CDSCD = 0
+CDSCB = 0 ETA0 = 0.3241245 ETAB = -0.1842
+DSUB = 1.0287138 PCLM = 5.2654567 PDIBLC1 = 4.228338E-3
+PDIBLC2 = 1.204519E-3 PDIBLCB = 2.37525E-3 DROUT = 0
+PSCBE1 = 3.011456E10 PSCBE2 = 3.037042E-7 PVAG = 8.9564294
+DELTA = 0.01 MOBMOD = 1 PRT = 0
+UTE = -1.5 KT1 = -0.11 KT1L = 0
+KT2 = 0.022 UA1 = 4.31E-9 UB1 = -7.61E-18
+UC1 = -5.6E-11 AT = 3.3E4 WL = 0
+WLN = 1 WW = 0 WWN = 1
+WWL = 0 LL = 0 LLN = 1
+LW = 0 LWN = 1 LWL = 0
+CAPMOD = 2 XPART = 0.4 CGDO = 5.59E-10
+CGSO = 5.59E-10 CGBO = 0 CJ = 1.882857E-3
+PB = 0.9891317 MJ = 0.4679789 CJSW = 3.67186E-10
+PBSW = 0.9884654 MJSW = 0.3562128 PVTH0 = 3.923756E-3
+PRDSW = 15.3953053 PK2 = 2.061759E-3 WKETA = 4.10049E-3
+LKETA = -0.0232426 LVSAT = 1.257E5 )
*
```

## B.3 Twenty-five 8-bit random test vectors used in the Spice simulations

| decimal a | decimal b | binary a a7 to a0 | binary b b7 to b0 | carry out | binary sum s7 to s0 |
|---|---|---|---|---|---|
| 68 | 88 | 01000100 | 01011000 | 0 | 10011100 |
| 93 | 76 | 01011101 | 01001100 | 0 | 10101001 |
| 190 | 196 | 10111110 | 11000100 | 1 | 10000010 |
| 12 | 193 | 00001100 | 11000001 | 0 | 11001101 |
| 121 | 190 | 01111001 | 10111110 | 1 | 00110111 |
| 104 | 56 | 01101000 | 00111000 | 0 | 10100000 |
| 121 | 203 | 01111001 | 11001011 | 1 | 01000100 |
| 186 | 97 | 10111010 | 01100001 | 1 | 00011011 |
| 98 | 6 | 01100010 | 00000110 | 0 | 01101000 |
| 11 | 22 | 00001011 | 00010110 | 0 | 00100001 |
| 148 | 229 | 10010100 | 11100101 | 1 | 01111001 |
| 228 | 118 | 11100100 | 01110110 | 1 | 01011010 |
| 121 | 216 | 01111001 | 11011000 | 1 | 01010001 |
| 57 | 77 | 00111001 | 01001101 | 0 | 10000110 |
| 246 | 245 | 11110110 | 11110101 | 1 | 11101011 |
| 240 | 154 | 11110000 | 10011010 | 1 | 10001010 |
| 14 | 184 | 00001110 | 10111000 | 0 | 11000110 |
| 210 | 189 | 11010010 | 10111101 | 1 | 10001111 |
| 27 | 131 | 00011011 | 10000011 | 0 | 10011110 |
| 190 | 37 | 10111110 | 00100101 | 0 | 11100011 |
| 113 | 67 | 01110001 | 01000011 | 0 | 10110100 |
| 59 | 165 | 00111011 | 10100101 | 0 | 11100000 |
| 233 | 148 | 11101001 | 10010100 | 1 | 01111101 |
| 249 | 38 | 11111001 | 00100110 | 1 | 00011111 |
| 114 | 70 | 01110010 | 01000110 | 0 | 10111000 |

## B.4 Spice code used to simulate the 4-bit ripple carry adder (rca8bitorig.spn) and carry-save multiplier (mul8bitorig.spn)

```
vdd1 Vdd 0 dc 2.5
vgnd1 GND 0 dc 0
vci ci 0 dc 0

va3 a3 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 0 29.9n 0 30n 2.5
39.9n 2.5 40n 0 49.9n 0 50n 0 59.9n 0 60n 0 69.9n 0 70n 2.5
79.9n 2.5 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 2.5 109.9n 2.5
```

```
110n 2.5 119.9n 2.5 120n 2.5 129.9n 2.5 130n 2.5 139.9n 2.5
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 2.5 169.9n 2.5 170n 0
179.9n 0 180n 0 189.9n 0 190n 0 199.9n 0 200n 2.5 209.9n 2.5
210n 0 219.9n 0 220n 0 229.9n 0 230n 2.5 239.9n 2.5 240n 2.5
249.9n 2.5 250n 0 259.9n 0 260n 0 )

va2 a2 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 2.5 39.9n 2.5 40n 2.5 49.9n 2.5 50n 0 59.9n 0 60n 0
69.9n 0 70n 0 79.9n 0 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 0
109.9n 0 110n 0 119.9n 0 120n 0 129.9n 0 130n 2.5 139.9n 2.5
140n 0 149.9n 0 150n 0 159.9n 0 160n 0 169.9n 0 170n 2.5
179.9n 2.5 180n 0 189.9n 0 190n 2.5 199.9n 2.5 200n 0 209.9n 0
210n 2.5 219.9n 2.5 220n 2.5 229.9n 2.5 230n 0 239.9n 0 240n 0
249.9n 0 250n 2.5 259.9n 2.5 260n 0 )

va1 a1 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 2.5 49.9n 2.5 50n 2.5 59.9n 2.5 60n 2.5
69.9n 2.5 70n 2.5 79.9n 2.5 80n 0 89.9n 0 90n 2.5 99.9n 2.5
100n 0 109.9n 0 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 2.5
139.9n 2.5 140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0
170n 2.5 179.9n 2.5 180n 2.5 189.9n 2.5 190n 0 199.9n 0 200n 0
209.9n 0 210n 0 219.9n 0 220n 0 229.9n 0 230n 0 239.9n 0
240n 2.5 249.9n 2.5 250n 2.5 259.9n 2.5 260n 0 )

va0 a0 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 2.5 69.9n 2.5
70n 0 79.9n 0 80n 0 89.9n 0 90n 0 99.9n 0 100n 2.5 109.9n 2.5
110n 0 119.9n 0 120n 0 129.9n 0 130n 0 139.9n 0 140n 0 149.9n 0
150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 0 179.9n 0 180n 2.5
189.9n 2.5 190n 2.5 199.9n 2.5 200n 2.5 209.9n 2.5 210n 2.5
219.9n 2.5 220n 2.5 229.9n 2.5 230n 0 239.9n 0 240n 0 249.9n 0
250n 0 259.9n 0 260n 0 )

vb3 b3 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 2.5 29.9n 2.5
30n 2.5 39.9n 2.5 40n 0 49.9n 0 50n 2.5 59.9n 2.5 60n 0 69.9n 0
70n 2.5 79.9n 2.5 80n 0 89.9n 0 90n 2.5 99.9n 2.5 100n 2.5
109.9n 2.5 110n 0 119.9n 0 120n 0 129.9n 0 130n 0 139.9n 0
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 2.5
179.9n 2.5 180n 2.5 189.9n 2.5 190n 2.5 199.9n 2.5 200n 2.5
209.9n 2.5 210n 0 219.9n 0 220n 2.5 229.9n 2.5 230n 2.5
239.9n 2.5 240n 2.5 249.9n 2.5 250n 0 259.9n 0 260n 0 )

vb2 b2 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 0 49.9n 0 50n 2.5 59.9n 2.5 60n 2.5 69.9n 2.5
70n 2.5 79.9n 2.5 80n 2.5 89.9n 2.5 90n 2.5 99.9n 2.5 100n 2.5
109.9n 2.5 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 2.5
```

```
139.9n 2.5 140n 2.5 149.9n 2.5 150n 0 159.9n 0 160n 2.5
169.9n 2.5 170n 2.5 179.9n 2.5 180n 0 189.9n 0 190n 2.5
199.9n 2.5 200n 2.5 209.9n 2.5 210n 0 219.9n 0 220n 2.5
229.9n 2.5 230n 0 239.9n 0 240n 2.5 249.9n 2.5 250n 0 259.9n 0
260n 0 )

vb1 b1 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 2.5 69.9n 2.5
70n 0 79.9n 0 80n 2.5 89.9n 2.5 90n 2.5 99.9n 2.5 100n 0
109.9n 0 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 0 139.9n 0
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 2.5
179.9n 2.5 180n 2.5 189.9n 2.5 190n 0 199.9n 0 200n 0 209.9n 0
210n 2.5 219.9n 2.5 220n 2.5 229.9n 2.5 230n 2.5 239.9n 2.5
240n 2.5 249.9n 2.5 250n 2.5 259.9n 2.5 260n 0 )

vb0 b0 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 0 69.9n 0 70n 2.5
79.9n 2.5 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 0 109.9n 0
110n 0 119.9n 0 120n 0 129.9n 0 130n 2.5 139.9n 2.5 140n 0
149.9n 0 150n 0 159.9n 0 160n 2.5 169.9n 2.5 170n 2.5 179.9n 2.5
180n 0 189.9n 0 190n 0 199.9n 0 200n 2.5 209.9n 2.5 210n 2.5
219.9n 2.5 220n 0 229.9n 0 230n 0 239.9n 0 240n 0 249.9n 0
250n 0 259.9n 0 260n 0 )

.tran 1n 260n
.options itl4=100
.IC V(w0)=0
.IC V(w1)=0
.IC V(w2)=0
.IC V(w3)=0
.IC V(w4)=0
.IC V(w5)=0
.IC V(w6)=0
.IC V(w7)=0
.IC V(c0)=0
.IC V(c1)=0
.IC V(c2)=0
.IC V(c3)=0
.IC V(s0)=0
.IC V(s1)=0
.IC V(s2)=0
.IC V(s3)=0
.end
```

## B.5 Spice code used to simulate the 4-bit ripple carry adder with ckt4 control transistors connected to the gates (rca4bitckt4.spn)

```
*Similar to the code used to control the 4-bit carry save
*multiplier with ckt4 control transistors (mul4bitckt4.spn)

vdd1 Vdd 0 dc 2.5
vgnd1 GND 0 dc 0
vci ci 0 dc 0

va3 a3 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 0 29.9n 0 30n 2.5
39.9n 2.5 40n 0 49.9n 0 50n 0 59.9n 0 60n 0 69.9n 0 70n 2.5
79.9n 2.5 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 2.5 109.9n 2.5
110n 2.5 119.9n 2.5 120n 2.5 129.9n 2.5 130n 2.5 139.9n 2.5
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 2.5 169.9n 2.5 170n 0
179.9n 0 180n 0 189.9n 0 190n 0 199.9n 0 200n 2.5 209.9n 2.5
210n 0 219.9n 0 220n 0 229.9n 0 230n 2.5 239.9n 2.5 240n 2.5
249.9n 2.5 250n 0 259.9n 0 260n 0 )

va2 a2 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 2.5 39.9n 2.5 40n 2.5 49.9n 2.5 50n 0 59.9n 0 60n 0 69.9n 0
70n 0 79.9n 0 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 0 109.9n 0
110n 0 119.9n 0 120n 0 129.9n 0 130n 2.5 139.9n 2.5 140n 0
149.9n 0 150n 0 159.9n 0 160n 0 169.9n 0 170n 2.5 179.9n 2.5
180n 0 189.9n 0 190n 2.5 199.9n 2.5 200n 0 209.9n 0 210n 2.5
219.9n 2.5 220n 2.5 229.9n 2.5 230n 0 239.9n 0 240n 0 249.9n 0
250n 2.5 259.9n 2.5 260n 0 )

va1 a1 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 2.5 49.9n 2.5 50n 2.5 59.9n 2.5 60n 2.5
69.9n 2.5 70n 2.5 79.9n 2.5 80n 0 89.9n 0 90n 2.5 99.9n 2.5
100n 0 109.9n 0 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 2.5
139.9n 2.5 140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0
170n 2.5 179.9n 2.5 180n 2.5 189.9n 2.5 190n 0 199.9n 0 200n 0
209.9n 0 210n 0 219.9n 0 220n 0 229.9n 0 230n 0 239.9n 0
240n 2.5 249.9n 2.5 250n 2.5 259.9n 2.5 260n 0 )

va0 a0 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 2.5 69.9n 2.5
70n 0 79.9n 0 80n 0 89.9n 0 90n 0 99.9n 0 100n 2.5 109.9n 2.5
110n 0 119.9n 0 120n 0 129.9n 0 130n 0 139.9n 0 140n 0 149.9n 0
150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 0 179.9n 0 180n 2.5
189.9n 2.5 190n 2.5 199.9n 2.5 200n 2.5 209.9n 2.5 210n 2.5
219.9n 2.5 220n 2.5 229.9n 2.5 230n 0 239.9n 0 240n 0 249.9n 0
```

```
250n 0 259.9n 0 260n 0 )

vb3 b3 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 2.5 29.9n 2.5
30n 2.5 39.9n 2.5 40n 0 49.9n 0 50n 2.5 59.9n 2.5 60n 0 69.9n 0
70n 2.5 79.9n 2.5 80n 0 89.9n 0 90n 2.5 99.9n 2.5 100n 2.5
109.9n 2.5 110n 0 119.9n 0 120n 0 129.9n 0 130n 0 139.9n 0
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 2.5
179.9n 2.5 180n 2.5 189.9n 2.5 190n 2.5 199.9n 2.5 200n 2.5
209.9n 2.5 210n 0 219.9n 0 220n 2.5 229.9n 2.5 230n 2.5
239.9n 2.5 240n 2.5 249.9n 2.5 250n 0 259.9n 0 260n 0 )

vb2 b2 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 0 49.9n 0 50n 2.5 59.9n 2.5 60n 2.5 69.9n 2.5
70n 2.5 79.9n 2.5 80n 2.5 89.9n 2.5 90n 2.5 99.9n 2.5 100n 2.5
109.9n 2.5 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 2.5
139.9n 2.5 140n 2.5 149.9n 2.5 150n 0 159.9n 0 160n 2.5
169.9n 2.5 170n 2.5 179.9n 2.5 180n 0 189.9n 0 190n 2.5
199.9n 2.5 200n 2.5 209.9n 2.5 210n 0 219.9n 0 220n 2.5
229.9n 2.5 230n 0 239.9n 0 240n 2.5 249.9n 2.5 250n 0 259.9n 0
260n 0 )

vb1 b1 0 pwl ( 0n 0 9.9n 0 10n 0 19.9n 0 20n 2.5 29.9n 2.5
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 2.5 69.9n 2.5
70n 0 79.9n 0 80n 2.5 89.9n 2.5 90n 2.5 99.9n 2.5 100n 0
109.9n 0 110n 0 119.9n 0 120n 2.5 129.9n 2.5 130n 0 139.9n 0
140n 0 149.9n 0 150n 2.5 159.9n 2.5 160n 0 169.9n 0 170n 2.5
179.9n 2.5 180n 2.5 189.9n 2.5 190n 0 199.9n 0 200n 0 209.9n 0
210n 2.5 219.9n 2.5 220n 2.5 229.9n 2.5 230n 2.5 239.9n 2.5
240n 2.5 249.9n 2.5 250n 2.5 259.9n 2.5 260n 0 )

vb0 b0 0 pwl ( 0n 0 9.9n 0 10n 2.5 19.9n 2.5 20n 0 29.9n 0
30n 0 39.9n 0 40n 0 49.9n 0 50n 0 59.9n 0 60n 0 69.9n 0 70n 2.5
79.9n 2.5 80n 2.5 89.9n 2.5 90n 0 99.9n 0 100n 0 109.9n 0
110n 0 119.9n 0 120n 0 129.9n 0 130n 2.5 139.9n 2.5 140n 0
149.9n 0 150n 0 159.9n 0 160n 2.5 169.9n 2.5 170n 2.5
179.9n 2.5 180n 0 189.9n 0 190n 0 199.9n 0 200n 2.5 209.9n 2.5
210n 2.5 219.9n 2.5 220n 0 229.9n 0 230n 0 239.9n 0 240n 0
249.9n 0 250n 0 259.9n 0 260n 0 )

vx0 x0 0 pwl ( 0n 0 10.3n 0 10.4n 2.5 12.3n 2.5 12.4n 0
20.3n 0 20.4n 2.5 22.3n 2.5 22.4n 0 30.3n 0 30.4n 2.5 32.3n 2.5
32.4n 0 40.3n 0 40.4n 2.5 42.3n 2.5 42.4n 0 50.3n 0 50.4n 2.5
52.3n 2.5 52.4n 0 60.3n 0 60.4n 2.5 62.3n 2.5 62.4n 0 70.3n 0
70.4n 2.5 72.3n 2.5 72.4n 0 80.3n 0 80.4n 2.5 82.3n 2.5 82.4n 0
90.3n 0 90.4n 2.5 92.3n 2.5 92.4n 0 100.3n 0 100.4n 2.5
102.3n 2.5 102.4n 0 110.3n 0 110.4n 2.5 112.3n 2.5 112.4n 0
```

```
120.3n 0 120.4n 2.5 122.3n 2.5 122.4n 0 130.3n 0 130.4n 2.5
132.3n 2.5 132.4n 0 140.3n 0 140.4n 2.5 142.3n 2.5 142.4n 0
150.3n 0 150.4n 2.5 152.3n 2.5 152.4n 0 160.3n 0 160.4n 2.5
162.3n 2.5 162.4n 0 170.3n 0 170.4n 2.5 172.3n 2.5 172.4n 0
180.3n 0 180.4n 2.5 182.3n 2.5 182.4n 0 190.3n 0 190.4n 2.5
192.3n 2.5 192.4n 0 200.3n 0 200.4n 2.5 202.3n 2.5 202.4n 0
210.3n 0 210.4n 2.5 212.3n 2.5 212.4n 0 220.3n 0 220.4n 2.5
222.3n 2.5 222.4n 0 230.3n 0 230.4n 2.5 232.3n 2.5 232.4n 0
240.3n 0 240.4n 2.5 242.3n 2.5 242.4n 0 250.3n 0 250.4n 2.5
252.3n 2.5 252.4n 0 260n 0 )

vx1 x1 0 pwl ( 0n 0 10.7n 0 10.8n 2.5 12.7n 2.5 12.8n 0 20.7n 0
20.8n 2.5 22.7n 2.5 22.8n 0 30.7n 0 30.8n 2.5 32.7n 2.5 32.8n 0
40.7n 0 40.8n 2.5 42.7n 2.5 42.8n 0 50.7n 0 50.8n 2.5 52.7n 2.5
52.8n 0 60.7n 0 60.8n 2.5 62.7n 2.5 62.8n 0 70.7n 0 70.8n 2.5
72.7n 2.5 72.8n 0 80.7n 0 80.8n 2.5 82.7n 2.5 82.8n 0 90.7n 0
90.8n 2.5 92.7n 2.5 92.8n 0 100.7n 0 100.8n 2.5 102.7n 2.5
102.8n 0 110.7n 0 110.8n 2.5 112.7n 2.5 112.8n 0 120.7n 0
120.8n 2.5 122.7n 2.5 122.8n 0 130.7n 0 130.8n 2.5 132.7n 2.5
132.8n 0 140.7n 0 140.8n 2.5 142.7n 2.5 142.8n 0 150.7n 0
150.8n 2.5 152.7n 2.5 152.8n 0 160.7n 0 160.8n 2.5 162.7n 2.5
162.8n 0 170.7n 0 170.8n 2.5 172.7n 2.5 172.8n 0 180.7n 0
180.8n 2.5 182.7n 2.5 182.8n 0 190.7n 0 190.8n 2.5 192.7n 2.5
192.8n 0 200.7n 0 200.8n 2.5 202.7n 2.5 202.8n 0 210.7n 0
210.8n 2.5 212.7n 2.5 212.8n 0 220.7n 0 220.8n 2.5 222.7n 2.5
222.8n 0 230.7n 0 230.8n 2.5 232.7n 2.5 232.8n 0 240.7n 0
240.8n 2.5 242.7n 2.5 242.8n 0 250.7n 0 250.8n 2.5 252.7n 2.5
252.8n 0 260n 0 )

vx2 x2 0 pwl ( 0n 0 11n 0 11.1n 2.5 13n 2.5 13.1n 0 21n 0
21.1n 2.5 23n 2.5 23.1n 0 31n 0 31.1n 2.5 33n 2.5 33.1n 0
41n 0 41.1n 2.5 43n 2.5 43.1n 0 51n 0 51.1n 2.5 53n 2.5 53.1n 0
61n 0 61.1n 2.5 63n 2.5 63.1n 0 71n 0 71.1n 2.5 73n 2.5 73.1n 0
81n 0 81.1n 2.5 83n 2.5 83.1n 0 91n 0 91.1n 2.5 93n 2.5 93.1n 0
101n 0 101.1n 2.5 103n 2.5 103.1n 0 111n 0 111.1n 2.5 113n 2.5
113.1n 0 121n 0 121.1n 2.5 123n 2.5 123.1n 0 131n 0 131.1n 2.5
133n 2.5 133.1n 0 141n 0 141.1n 2.5 143n 2.5 143.1n 0 151n 0
151.1n 2.5 153n 2.5 153.1n 0 161n 0 161.1n 2.5 163n 2.5
163.1n 0 171n 0 171.1n 2.5 173n 2.5 173.1n 0 181n 0 181.1n 2.5
183n 2.5 183.1n 0 191n 0 191.1n 2.5 193n 2.5 193.1n 0 201n 0
201.1n 2.5 203n 2.5 203.1n 0 211n 0 211.1n 2.5 213n 2.5 213.1n 0
221n 0 221.1n 2.5 223n 2.5 223.1n 0 231n 0 231.1n 2.5 233n 2.5
233.1n 0 241n 0 241.1n 2.5 243n 2.5 243.1n 0 251n 0 251.1n 2.5
253n 2.5 253.1n 0 260n 0 )

vx3 x3 0 pwl ( 0n 0 11.4n 0 11.5n 2.5 13.4n 2.5 13.5n 0 21.4n 0
```

```
21.5n 2.5 23.4n 2.5 23.5n 0 31.4n 0 31.5n 2.5 33.4n 2.5 33.5n 0
41.4n 0 41.5n 2.5 43.4n 2.5 43.5n 0 51.4n 0 51.5n 2.5 53.4n 2.5
53.5n 0 61.4n 0 61.5n 2.5 63.4n 2.5 63.5n 0 71.4n 0 71.5n 2.5
73.4n 2.5 73.5n 0 81.4n 0 81.5n 2.5 83.4n 2.5 83.5n 0 91.4n 0
91.5n 2.5 93.4n 2.5 93.5n 0 101.4n 0 101.5n 2.5 103.4n 2.5
103.5n 0 111.4n 0 111.5n 2.5 113.4n 2.5 113.5n 0 121.4n 0
121.5n 2.5 123.4n 2.5 123.5n 0 131.4n 0 131.5n 2.5 133.4n 2.5
133.5n 0 141.4n 0 141.5n 2.5 143.4n 2.5 143.5n 0 151.4n 0
151.5n 2.5 153.4n 2.5 153.5n 0 161.4n 0 161.5n 2.5 163.4n 2.5
163.5n 0 171.4n 0 171.5n 2.5 173.4n 2.5 173.5n 0 181.4n 0
181.5n 2.5 183.4n 2.5 183.5n 0 191.4n 0 191.5n 2.5 193.4n 2.5
193.5n 0 201.4n 0 201.5n 2.5 203.4n 2.5 203.5n 0 211.4n 0
211.5n 2.5 213.4n 2.5 213.5n 0 221.4n 0 221.5n 2.5 223.4n 2.5
223.5n 0 231.4n 0 231.5n 2.5 233.4n 2.5 233.5n 0 241.4n 0
241.5n 2.5 243.4n 2.5 243.5n 0 251.4n 0 251.5n 2.5 253.4n 2.5
253.5n 0 260n 0 )

vy0 y0 0 pwl ( 0n 2.5 10.3n 2.5 10.4n 0 12.3n 0 12.4n 2.5
20.3n 2.5 20.4n 0 22.3n 0 22.4n 2.5 30.3n 2.5 30.4n 0 32.3n 0
32.4n 2.5 40.3n 2.5 40.4n 0 42.3n 0 42.4n 2.5 50.3n 2.5 50.4n 0
52.3n 0 52.4n 2.5 60.3n 2.5 60.4n 0 62.3n 0 62.4n 2.5 70.3n 2.5
70.4n 0 72.3n 0 72.4n 2.5 80.3n 2.5 80.4n 0 82.3n 0 82.4n 2.5
90.3n 2.5 90.4n 0 92.3n 0 92.4n 2.5 100.3n 2.5 100.4n 0
102.3n 0 102.4n 2.5 110.3n 2.5 110.4n 0 112.3n 0 112.4n 2.5
120.3n 2.5 120.4n 0 122.3n 0 122.4n 2.5 130.3n 2.5 130.4n 0
132.3n 0 132.4n 2.5 140.3n 2.5 140.4n 0 142.3n 0 142.4n 2.5
150.3n 2.5 150.4n 0 152.3n 0 152.4n 2.5 160.3n 2.5 160.4n 0
162.3n 0 162.4n 2.5 170.3n 2.5 170.4n 0 172.3n 0 172.4n 2.5
180.3n 2.5 180.4n 0 182.3n 0 182.4n 2.5 190.3n 2.5 190.4n 0
192.3n 0 192.4n 2.5 200.3n 2.5 200.4n 0 202.3n 0 202.4n 2.5
210.3n 2.5 210.4n 0 212.3n 0 212.4n 2.5 220.3n 2.5 220.4n 0
222.3n 0 222.4n 2.5 230.3n 2.5 230.4n 0 232.3n 0 232.4n 2.5
240.3n 2.5 240.4n 0 242.3n 0 242.4n 2.5 250.3n 2.5 250.4n 0
252.3n 0 252.4n 2.5 260n 2.5 )

vy1 y1 0 pwl ( 0n 2.5 10.7n 2.5 10.8n 0 12.7n 0 12.8n 2.5
20.7n 2.5 20.8n 0 22.7n 0 22.8n 2.5 30.7n 2.5 30.8n 0 32.7n 0
32.8n 2.5 40.7n 2.5 40.8n 0 42.7n 0 42.8n 2.5 50.7n 2.5 50.8n 0
52.7n 0 52.8n 2.5 60.7n 2.5 60.8n 0 62.7n 0 62.8n 2.5 70.7n 2.5
70.8n 0 72.7n 0 72.8n 2.5 80.7n 2.5 80.8n 0 82.7n 0 82.8n 2.5
90.7n 2.5 90.8n 0 92.7n 0 92.8n 2.5 100.7n 2.5 100.8n 0 102.7n 0
102.8n 2.5 110.7n 2.5 110.8n 0 112.7n 0 112.8n 2.5 120.7n 2.5
120.8n 0 122.7n 0 122.8n 2.5 130.7n 2.5 130.8n 0 132.7n 0
132.8n 2.5 140.7n 2.5 140.8n 0 142.7n 0 142.8n 2.5 150.7n 2.5
150.8n 0 152.7n 0 152.8n 2.5 160.7n 2.5 160.8n 0 162.7n 0
162.8n 2.5 170.7n 2.5 170.8n 0 172.7n 0 172.8n 2.5 180.7n 2.5
```

```
180.8n 0 182.7n 0 182.8n 2.5 190.7n 2.5 190.8n 0 192.7n 0
192.8n 2.5 200.7n 2.5 200.8n 0 202.7n 0 202.8n 2.5 210.7n 2.5
210.8n 0 212.7n 0 212.8n 2.5 220.7n 2.5 220.8n 0 222.7n 0
222.8n 2.5 230.7n 2.5 230.8n 0 232.7n 0 232.8n 2.5 240.7n 2.5
240.8n 0 242.7n 0 242.8n 2.5 250.7n 2.5 250.8n 0 252.7n 0
252.8n 2.5 260n 2.5 )

vy2 y2 0 pwl ( 0n 2.5 11n 2.5 11.1n 0 13n 0 13.1n 2.5 21n 2.5
21.1n 0 23n 0 23.1n 2.5 31n 2.5 31.1n 0 33n 0 33.1n 2.5
41n 2.5 41.1n 0 43n 0 43.1n 2.5 51n 2.5 51.1n 0 53n 0
53.1n 2.5 61n 2.5 61.1n 0 63n 0 63.1n 2.5 71n 2.5 71.1n 0
73n 0 73.1n 2.5 81n 2.5 81.1n 0 83n 0 83.1n 2.5 91n 2.5
91.1n 0 93n 0 93.1n 2.5 101n 2.5 101.1n 0 103n 0 103.1n 2.5
111n 2.5 111.1n 0 113n 0 113.1n 2.5 121n 2.5 121.1n 0 123n 0
123.1n 2.5 131n 2.5 131.1n 0 133n 0 133.1n 2.5 141n 2.5
141.1n 0 143n 0 143.1n 2.5 151n 2.5 151.1n 0 153n 0 153.1n 2.5
161n 2.5 161.1n 0 163n 0 163.1n 2.5 171n 2.5 171.1n 0 173n 0
173.1n 2.5 181n 2.5 181.1n 0 183n 0 183.1n 2.5 191n 2.5
191.1n 0 193n 0 193.1n 2.5 201n 2.5 201.1n 0 203n 0 203.1n 2.5
211n 2.5 211.1n 0 213n 0 213.1n 2.5 221n 2.5 221.1n 0 223n 0
223.1n 2.5 231n 2.5 231.1n 0 233n 0 233.1n 2.5 241n 2.5
241.1n 0 243n 0 243.1n 2.5 251n 2.5 251.1n 0 253n 0 253.1n 2.5
260n 2.5 )

vy3 y3 0 pwl ( 0n 2.5 11.4n 2.5 11.5n 0 13.4n 0 13.5n 2.5
21.4n 2.5 21.5n 0 23.4n 0 23.5n 2.5 31.4n 2.5 31.5n 0 33.4n 0
33.5n 2.5 41.4n 2.5 41.5n 0 43.4n 0 43.5n 2.5 51.4n 2.5
51.5n 0 53.4n 0 53.5n 2.5 61.4n 2.5 61.5n 0 63.4n 0 63.5n 2.5
71.4n 2.5 71.5n 0 73.4n 0 73.5n 2.5 81.4n 2.5 81.5n 0 83.4n 0
83.5n 2.5 91.4n 2.5 91.5n 0 93.4n 0 93.5n 2.5 101.4n 2.5
101.5n 0 103.4n 0 103.5n 2.5 111.4n 2.5 111.5n 0 113.4n 0
113.5n 2.5 121.4n 2.5 121.5n 0 123.4n 0 123.5n 2.5 131.4n 2.5
131.5n 0 133.4n 0 133.5n 2.5 141.4n 2.5 141.5n 0 143.4n 0
143.5n 2.5 151.4n 2.5 151.5n 0 153.4n 0 153.5n 2.5 161.4n 2.5
161.5n 0 163.4n 0 163.5n 2.5 171.4n 2.5 171.5n 0 173.4n 0
173.5n 2.5 181.4n 2.5 181.5n 0 183.4n 0 183.5n 2.5 191.4n 2.5
191.5n 0 193.4n 0 193.5n 2.5 201.4n 2.5 201.5n 0 203.4n 0
203.5n 2.5 211.4n 2.5 211.5n 0 213.4n 0 213.5n 2.5 221.4n 2.5
221.5n 0 223.4n 0 223.5n 2.5 231.4n 2.5 231.5n 0 233.4n 0
233.5n 2.5 241.4n 2.5 241.5n 0 243.4n 0 243.5n 2.5 251.4n 2.5
251.5n 0 253.4n 0 253.5n 2.5 260n 2.5 )

.tran 1n 260n
.options itl4=100
.IC V(w0)=0
.IC V(w1)=0
```

```
.IC V(w2)=0
.IC V(w3)=0
.IC V(w4)=0
.IC V(w5)=0
.IC V(w6)=0
.IC V(w7)=0
.IC V(c0)=0
.IC V(c1)=0
.IC V(c2)=0
.IC V(c3)=0
.IC V(s0)=0
.IC V(s1)=0
.IC V(s2)=0
.IC V(s3)=0
.end
```

# Appendix C

# Boolean Algebra

## C.1 Mathematical Systems

### C.1.1 Boolean definitions

A mathematical system consists of a:

1. Set of elements
2. Set of operations
3. Set of postulates

The following are a series of boolean algebra definitions.

**Operation** A binary operation $\circ$ on a set $S$ of elements is a rule that assigns to pairs of elements from $S$ a unique element from $S$. This is written as $c = a \circ b$.

**Closed set** A set $S$ is closed with respect to a binary operation if, for every pair of elements of $S$, the binary operation specifies a rule for obtaining a unique element of $S$.

**Commutative** A binary operation $\circ$ on a set $S$ is said to be commutative if and only if for every $a$ and $b$ in $S$: $a \circ b = b \circ a$.

**Associative** A binary operation $\circ$ on a set $S$ is said to be associative if and only if for every $a$, $b$ and $c$ in $S$: $(a \circ b) \circ c = a \circ (b \circ c)$.

**Identity** An element $e$ in the set $S$ is an identity element for the binary operation $\circ$ on $S$ if and only if for every element $a$ in $S$: $a \circ e = e \circ a = a$.

**Distributive** If $\circ$ and $*$ are 2 binary operations on the set $S$, the operation $\circ$ is distributive over $*$ if and only if for every $a$, $b$ and $c$ in $S$: $a \circ (b * c) = (a \circ b) * (a \circ c)$.

### C.1.2 Boolean postulates

A mathematical system consisting of a set of elements $B$ and 2 binary operations $(+)$ and $(\cdot)$ is called a Boolean algebra if and only if the following postulates hold:

**P1** The set $B$ is closed with respect to the operations $(+)$ and $(\cdot)$.

**P2** The operations $(+)$ and $(\cdot)$ are commutative, i. e. $x+y=y+x$ and $x\cdot y=y\cdot x$.

**P3** Each operation $(+)$ and $(\cdot)$ is distributive over the other, i. e.

$$\begin{aligned} x\cdot(y+z) &= (x\cdot y)+(x\cdot z) \\ x+(y\cdot z) &= (x+y)\cdot(x+z) \end{aligned}$$

**P4** There exists in $B$ distinct identity elements 0 and 1 relative to the operations $(+)$ and $(\cdot)$, respectively, i. e.

$$\begin{aligned} 0+x &= x+0 = x \\ 1\cdot x &= x\cdot 1 = x \end{aligned}$$

**P5** For every element $x$ in $B$ there exists an element $x'$ in $B$, called the complement or negation of $x$ such that $x+x'=1$ and $x\cdot x'=0$.

**P6** There exists at least 2 elements $x, y \varepsilon B$ such that $x \neq y$.

### C.1.3 Principle of duality

Every statement or algebraic identity deducible from the postulates of a Boolean algebra remains valid if the operations $(+)$ and $(\cdot)$, and the identity elements 0 and 1 are interchanged throughout. i. e.

$$\begin{aligned} (x\cdot y)+(x\cdot z) &= (xy)+(xz) \\ &= xy+xz \end{aligned}$$

**Theorem:** The set consisting of the elements 0 and 1, together with the operations defined by the following tables, is a Boolean algebra. $B=\{0,1\}$

| $+$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| $\cdot$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

If a postulate is valid for every single possible case, then it must be valid. This is known as proof by exhaustion. Consider the statement $x+yz=(x+y)(x+z)$. It can be proven in full with the following truth table.

| $x$ | $y$ | $z$ | $x+y$ | $x+z$ | $(x+y)(x+z)$ | $yz$ | $x+yz$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Since the columns for $(x+y)(x+z)$ and $x+yz$ are identical, this is an example of *perfect induction*. For example, consider the truth table for the confirmation of postulate 5:

| $x$ | $x'$ | $x+x'$ | $x\cdot x'$ |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |

### C.1.4 Two-valued Boolean algebra

A two-valued Boolean algebra is defined on a set of two elements, $B=\{0,1\}$, with rules for the two binary operators $+$ and $\cdot$ as shown in the following truth tables.

Or

| $+$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

And

| $\cdot$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

Not

| $x$ | $x'$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

### C.1.5 Boolean identities

| | | | |
|---|---|---|---|
| 1a) | $0'=1$ | 1b) | $1'=0$ | |
| 2a) | $x+0=x$ | 2b) | $x\cdot 1=x$ | |
| 3a) | $x+1=1$ | 3b) | $x\cdot 0=0$ | |
| 4a) | $x+x=x$ | 4b) | $x\cdot x=x$ | Idempotent law |
| 5a) | $x+x'=1$ | 5b) | $x\cdot x'=0$ | |
| 6 | $(x')'=x$ | | | Involution law |
| 7a) | $x+y=y+x$ | 7b) | $x\cdot y=y\cdot x$ | Commutative law |
| 8a) | $x+xy=x$ | 8b) | $x(x+y)=x$ | Absorbtion law |
| 9a) | $x+x'y=x+y$ | 9b) | $x(x'+y)=xy$ | |
| 10a) | $(x+y)'=x'y'$ | 10b) | $(xy)'=x'+y'$ | DeMorgan's law |
| 11a) | $(x+y)+z=x+(y+z)$ $=x+y+z$ | 11b) | $(xy)z=x(yz)$ $=xyz$ | Associative law |
| 12a) | $x+yz=(x+y)(x+z)$ | 12b) | $x(y+z)=xy+xz$ | Distributive law |

**Note:**

$$\begin{aligned} x+x &= (x+x)\cdot 1 && \text{(P4)}\\ &= (x+x)(x+x') && \text{(P5)}\\ &= x+xx' && \text{(P3)}\\ &= x+0 && \text{(P5)}\\ &= x && \text{(P4)} \end{aligned}$$

**Generalized DeMorgan's law**

$$(w+x+\ldots+y+z)' = w'\cdot x'\cdot\ldots\cdot y'\cdot z'$$

also

$$(w\cdot x\cdot\ldots\cdot y\cdot z)' = w'+x'+\ldots+y'+z'$$

## C.2 Boolean Formulas and Functions

### C.2.1 Boolean functions

An $n-$variable (complete) *Boolean function* $f(x_1,x_2,\ldots,x_n)$ is a mapping that assigns a unique value, called the value of the function, for every combination of values of the $n$ independent variables in which all values are limited to the set $\{0,1\}$. The following is known as a Truth Table or Table of Combinations. There are $2^{2^n}$ functions in all.

$n+1$ columns

| | $x_1$ | $x_2$ | $\ldots$ | $x_n$ | $f$ |
|---|---|---|---|---|---|
| $2^n$rows | 0 | 0 | $\ldots$ | 0 | $f(0,0,\ldots,0)$ |
| | 0 | 0 | $\ldots$ | 1 | $f(0,0,\ldots,1)$ |
| | | | $\vdots$ | | $\vdots$ |
| | 1 | 1 | $\ldots$ | 1 | $f(1,1,\ldots,1)$ |

**Example:** Consider the function $f = x'y + y'z + yz$.

| $x$ | $y$ | $z$ | $x'y$ | $y'z$ | $yz$ | $f$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |

**Equivalence** Two formulas are said to be equivalent if and only if they describe the same function, i. e. $A = B$.

**Literal** A literal is the occurrence of a completed or uncompleted variable in a Boolean expression, e. g. $f = x'y + y'z + yz$. This expression contains 6 literals.

**Product term** This is either a literal or a product of literals, e. g. $w + y'z + w'x$ (dnf).

**Disjunctive normal formula (dnf)** This is a Boolean formula which is written as a single product term or as a sum of product terms.

**Sum term** This is a literal or a sum of literals, e. g. $w'$ or $(y+z)$.

**Conjunctive normal formula (cnf)** This is a Boolean formula which is written as a single sum term or as a product of sum terms, e. g. $w'(x+z)(w'+y)$.

Consider the following: $w'(x+z)+y'z+xz$. It is not a dnf. Similary, $(x'y+z)(x+z')$ is not a cnf.

**Example:** Consider the function $f = (a' + a'b)(a + bc)$. This has 6 literals. We want to simplify or manipulate this expression into another form.

$$\begin{array}{rrcl} \text{since} & x + xy & = & x \\ \text{then} & a' + a'b & = & a' \\ \text{therefore} & f & = & a'(a + bc) \\ \text{also} & x(x' + y) & = & xy \\ \text{then} & a'(a + bc) & = & a'bc \\ \text{therefore} & f & = & a'bc \end{array}$$

This equation has 3 literals, so it's much simpler.

**Example:** Simplify the function $f = w'y' + wz + y'z + xyz$. Note that sometimes you must add more literals in order to get rid of more later on.

$$\begin{array}{rrcll} & f & = & w'y' + wz + y'z + xyz & \\ & f & = & w'y' + wz + z(y' + xy) & \\ \text{now} & x + x'y & = & x + y & \\ \text{then} & y' + yx & = & y' + x & \\ & f & = & w'y' + wz + z(y' + x) & \\ & f & = & w'y' + wz + y'z + xz & \\ & f & = & w'y' + wz + (w + w')y'z + xz & \\ & f & = & w'y' + wz + wy'z + w'y'z + xz & \\ & f & = & w'y'(1 + z) + wz(1 + y') + xz & \\ & f & = & w'y' + wz + xz & \text{(6 literals)} \end{array}$$

**Complimentation:** Given the function $f = (w'x + yz)y + z'$, find the compliment $f'$. Note the following postulates: $(x + y)' = x'y'$ and $(xy)' = x' + y'$.

$$\begin{array}{rcl} f & = & (w'x + yz)y + z' \\ f' & = & \left[(w'x + yz)y + z'\right]' \\ f' & = & \left[(w'x + yz)y\right]' z \\ f' & = & \left[\dfrac{(w'x + yz)}{x} \cdot \dfrac{y}{x}\right]' z \\ f' & = & \left[(w'x + yz)' + y'\right] z \\ f' & = & \left[(w'x)'(yz)' + y'\right] z \\ f' & = & \left[(w + x')(y' + z') + y'\right] z \end{array}$$

### C.2.1.1 Implementing functions with gates

When a Boolean function is implemented with logic gates, each literal in the function designates an input to a gate, and each term is implemented with a gate. The minimization of the number of literals and the number of terms results in a circuit with less equipment.

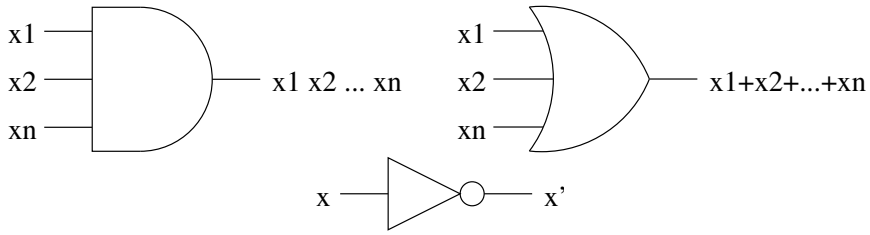



Figure C.1: AND gate, OR gate, NOT gate

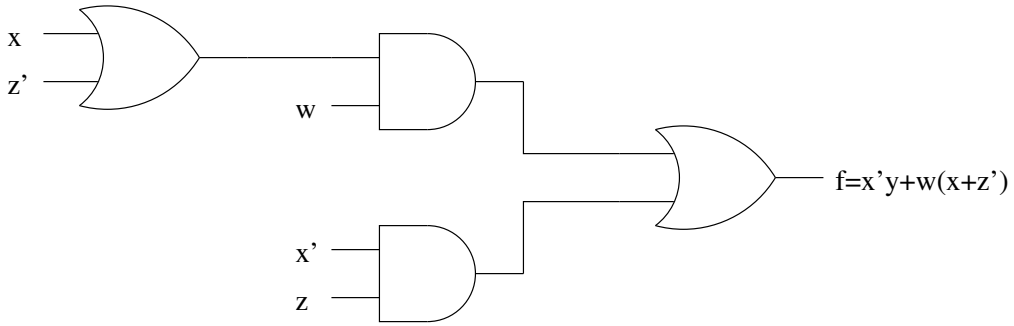



Figure C.2: Realization of $f(w,x,y,z) = x'y + w(x+z')$

## C.2.2 Minterm canonical formulae

Consider an expression that consists of the sum of these three terms: $f = x'y'z' + x'y'z + xyz'$. This is an example of a *minterm canonical formula*.

| $xyz$ | $f$ | $x'y'z'$ | $x'y'z$ | $xyz'$ | $f$ |
|---|---|---|---|---|---|
| 000 | 1 | 1 | 0 | 0 | 1 |
| 001 | 1 | 0 | 1 | 0 | 1 |
| 010 | 0 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 |
| 110 | 1 | 0 | 0 | 1 | 1 |
| 111 | 0 | 0 | 0 | 0 | 0 |

**Minterm** (Standard product) This is a product term consisting of exactly one of each of the variables of a function in its complemented or uncomplemented form.

**Minterm canonical formula** (Standard sum-of-products, Disjunctive canonical formula) This is a Boolean expression consisting of a sum of just minterms.

**Example:**

$$\begin{aligned} f(x,y,z) &= x' + x(y+z') \\ &= x' + xy + xz' \qquad \text{(dnf)} \\ &= x'(y+y')(z+z') + xy(z+z') + xz'(y+y') \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xyz + xyz' + xyz' + xy'z' \\ &= x'yz + x'yz' + x'y'z + x'y'z' + xyz + xyz' + xy'z' \end{aligned}$$

**Minterm truth table**

| $xyz$ | minterms | decimal equivalent | notation |
|---|---|---|---|
| 000 | $x'y'z'$ | 0 | $m_0$ |
| 001 | $x'y'z$ | 1 | $m_1$ |
| 010 | $x'yz'$ | 2 | $m_2$ |
| 011 | $x'yz$ | 3 | $m_3$ |
| 100 | $xy'z'$ | 4 | $m_4$ |
| 101 | $xy'z$ | 5 | $m_5$ |
| 110 | $xyz'$ | 6 | $m_6$ |
| 111 | $xyz$ | 7 | $m_7$ |

**Example:**

$$\begin{aligned} f(x,y,z) &= x'y'z + xyz' + xyz \\ &= 001 + 110 + 111 \qquad \text{(in binary notation)} \\ &= m_1 + m_6 + m_7 \\ &= \sum m(1,6,7) \end{aligned}$$

**Also:**

$$\begin{aligned} f(x,y,z) &= \sum m(0,1,5) \\ &= 000 + 001 + 101 \\ &= m_0 + m_1 + m_5 \\ &= x'y'z' + x'y'z + xy'z \end{aligned}$$

### C.2.3 Maxterm canonical formulae

Consider the following truth table:

| $xyz$ | $f$ | $f' = g$ | |
|---|---|---|---|
| 000 | 1 | 0 | |
| 001 | 1 | 0 | |
| 010 | 0 | 1 | $\leftarrow$ |
| 011 | 0 | 1 | $\leftarrow$ |
| 100 | 1 | 0 | |
| 101 | 1 | 0 | |
| 110 | 1 | 0 | |
| 111 | 0 | 1 | $\leftarrow$ |

To go from a sum of minterms to a product of maxterms, complement the function.

$$\begin{aligned} f &= x'yz' + x'yz + xyz \\ f' &= (x'yz' + x'yz + xyz)' \\ g &= (x+y'+z)(x+y'+z')(x'+y'+z') \qquad \text{(cnf)} \end{aligned}$$

**Maxterm** (Standard sum) This is a sum term consisting of exactly one of each of the variables of a function in its complemented or uncomplemented form.

**Maxterm canonical formula** (Standard product-of-sums, conjunctive canonical formula) This is a Boolean expression consisting of a product of just maxterms.

**Example:**

$$\begin{aligned} f(x,y,z) &= x'y + xz \qquad \text{NB: } x + yz = (x+y)(x+z) \\ &= (x'y + x)(x'y + z) \\ &= (x+x')(x+y)(x'+z)(y+z) \\ &= (x+y)(x'+z)(y+z) \\ &= (x+y+zz')(x'+z+yy')(xx'+y+z) \\ &= (x+y+z)(x+y+z')(x'+y+z)(x'+y'+z)(x+y+z)(x'+y+z) \\ &= (x+y+z)(x+y+z')(x'+y+z)(x'+y'+z) \end{aligned}$$

**Maxterm truth table**

| $xyz$ | maxterms | decimal equivalent | notation |
|---|---|---|---|
| 000 | $xyz$ | 0 | $M_0$ |
| 001 | $xyz'$ | 1 | $M_1$ |
| 010 | $xy'z$ | 2 | $M_2$ |
| 011 | $xy'z'$ | 3 | $M_3$ |
| 100 | $x'yz$ | 4 | $M_4$ |
| 101 | $x'yz'$ | 5 | $M_5$ |
| 110 | $x'y'z$ | 6 | $M_6$ |
| 111 | $x'y'z'$ | 7 | $M_7$ |

**Example:**

$$\begin{aligned} f(x,y,z) &= (x+y'+z)(x'+y+z)(x'+y+z') \\ &= 010+100+101 \\ &= M_2M_4M_5 \\ &= \prod M(2,4,5) \qquad \text{where } \prod = \text{'the product of'} \end{aligned}$$

**Also:**

$$\begin{aligned} f(x,y,z) &= \prod M(0,1,5,6) \\ &= M_0M_1M_5M_6 \\ &= (x+y+z)(x+y+z')(x'+y+z')(x'+y'+z) \end{aligned}$$

### C.2.4 Complements and conversions

**Minterm complement theorem** The complement of a complete Boolean function represented by a minterm canonical formula can be described by joining the minterms not contained in the expression for the original function with 'or' operators, i. e.

$$\begin{aligned} f(x,y,z) &= m_1+m_2+m_3+m_7 \\ f'(x,y,z) &= m_0+m_4+m_5+m_6 \end{aligned}$$

**Maxterm complement theorem** The complement of a complete Boolean function represented by a maxterm canonical formula can be described by joining the maxterms not contained in the expression for the original function with 'and' operators, i. e.

$$\begin{aligned} f(w,x,y,z) &= M_1M_5M_7M_8M_9M_{12}M_{13}M_{14}M_{15} \\ f'(w,x,y,z) &= M_0M_2M_3M_4M_6M_{10}M_{11} \end{aligned}$$

**Theorem:** $m_i' = M_i$

**and** $M_i' = m_i$ where $i$ is a decimal subscript

The following examples show how to convert between minterms and maxterms.

**Eg. 1**

$$\begin{aligned} m_1 &= x'y'z \\ (m_1)' &= (x'y'z)' \\ &= x+y+z' \\ &= M_1 \end{aligned}$$

**Eg. 2**

$$\begin{aligned} M_1 &= x+y+z' \\ (M_1)' &= (x+y+z')' \\ &= x'y'z \\ &= m_1 \end{aligned}$$

**Eg. 3**

$$\begin{aligned} f(x,y,z) &= m_1+m_2+m_6 \\ f' &= (m_1+m_2+m_6)' \\ &= m_1'm_2'm_6' \\ &= M_1M_2M_6 \\ (f')' &= (M_1M_2M_6)' \\ &= M_0M_3M_4M_5M_7 \\ &= m_1+m_2+m_6 \end{aligned}$$

### C.2.5 Incomplete functions: 'don't care' conditions

**Incomplete Boolean functions** An incomplete Boolean function of $n$ variables is the set of ordered pairs in which the domain of the function is a proper subset of the set of $n-$tuples, $B^n$, and the range of the function is the set $B = \{0,1\}$.

**Example:** Consider the following function $f(x,y,z)$.

| $x$ | $y$ | $z$ | $f$ | $f'$ | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | | |
| 0 | 0 | 0 | – | – | ← | } don't cares (dC) |
| 0 | 1 | 1 | – | – | ← | } don't cares (dC) |
| 0 | 1 | 0 | 1 | 0 | | |
| 1 | 0 | 1 | – | – | ← | } don't cares (dC) |
| 1 | 0 | 0 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | 1 | | |
| 1 | 1 | 0 | 1 | 0 | | |

$$\begin{aligned} f(x,y,z) &= \sum m(0,3,7) + \mathrm{dC}(1,2,4) \\ &= \prod M(5,6) + \mathrm{dC}(1,2,4) \end{aligned}$$

$$\begin{aligned} f'(x,y,z) &= \sum m(5,6) + \mathrm{dC}(1,2,4) \\ &= \prod M(0,3,7) + \mathrm{dC}(1,2,4) \end{aligned}$$

## C.3 Digital Logic Gates

### C.3.1 Exclusive OR (XOR)

**Truth Table**

| $x$ | $y$ | $x \oplus y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Logic Symbol**

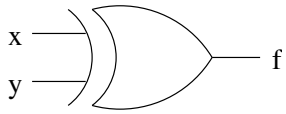



Figure C.3: XOR gate: $f = x \oplus y$

**XOR Identities**

i)
$$\begin{aligned} x \oplus y &= x'y + xy' \\ &= (x+y)(x'+y') \\ (x \oplus y)' &= x'y' + xy \\ &= (x'+y)(x+y') \end{aligned}$$

ii)
$$\begin{aligned} x \oplus 0 &= x \\ x \oplus 1 &= x' \end{aligned}$$

iii)
$$\begin{aligned} x \oplus x &= 0 \\ x \oplus x' &= 1 \end{aligned}$$

iv)
$$\begin{aligned} x' \oplus y' &= x \oplus y \\ x' \oplus y &= x \oplus y' \\ &= (x \oplus y)' \end{aligned}$$

v)
$$x \oplus y = y \oplus x$$

vi)
$$\begin{aligned} x \oplus (y \oplus z) &= (x \oplus y) \oplus z \\ &= x \oplus y \oplus z \end{aligned}$$

vii)
$$x(y \oplus z) = xy \oplus xz$$

viii)
$$x + y = x \oplus y \oplus xy$$

ix)
$$x + y = x \oplus y \quad \text{if and only if} \quad xy \equiv 0$$

x)
$$\text{if} \quad x \oplus y = z, \quad \text{then} \quad y \oplus z = x \quad \text{or} \quad x \oplus z = y$$

### C.3.2 Exclusive NOR (XNOR)

**Truth Table**

| $x$ | $y$ | $x \oplus y$ | $(x \oplus y)'$ | $= x \odot y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | |

**Logic Symbol**

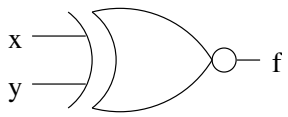



Figure C.4: XNOR gate: $f = x'y' + xy = (x \oplus y)'$

### C.3.3 NAND

**Truth Table**

| $x$ | $y$ | $(xy)'$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Logic Symbol**

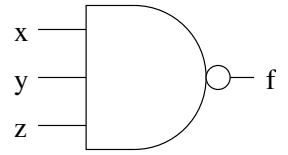



Figure C.5: NAND gate: $f = (xyz)' = x' + y' + z'$

**Equivalent Structure**

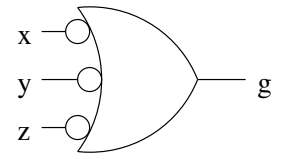



Figure C.6: OR gate (inverted inputs): $g = x' + y' + z' = (xyz)'$

It is possible to implement any Boolean operator using only NAND gates. Consider the following figures.

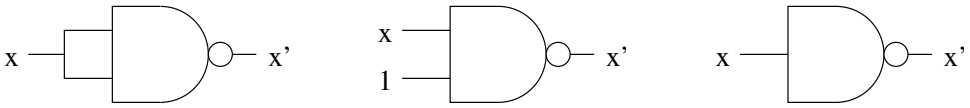



Figure C.7: NOT gate: $x' = (xx)'$, $x' = (1 \cdot x)'$, $x' = \text{NAND}(x)$

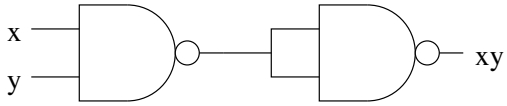



Figure C.8: AND gate: $xy = [(xy)']'$

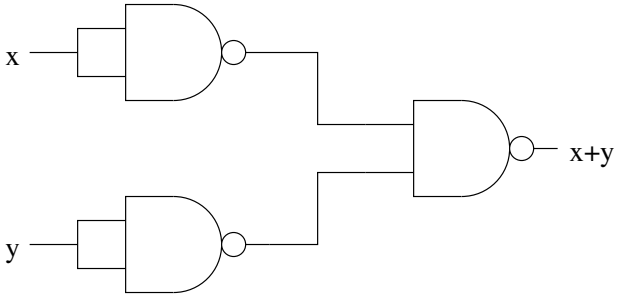



Figure C.9: OR gate: $x + y = (x' \cdot y')'$

**Example:** The function $f = xy'z + x'(w + y + z')$ can be implemented using only NAND gates.

$$\begin{aligned} & & f &= xy'z + x'(w + y + z') \\ \text{NB:} \quad & \text{NAND}(g, h \ldots k) &= [g() \cdot h() \ldots k()]' \\ & & f &= \Big\{ \underbrace{(xy'z)}_{g} \cdot \underbrace{[x'(w + y + z')]'}_{h} \Big\}' \end{aligned}$$

**Note:** Sometimes, when putting a function in the NAND complemented-products format, it is necessary to invert the output. This can be achieved with a single, one-input NAND gate.

$$\begin{aligned} f &= g() \cdot h() \cdots k() \\ f' &= [g() \cdot h() \cdots k()]' \end{aligned}$$

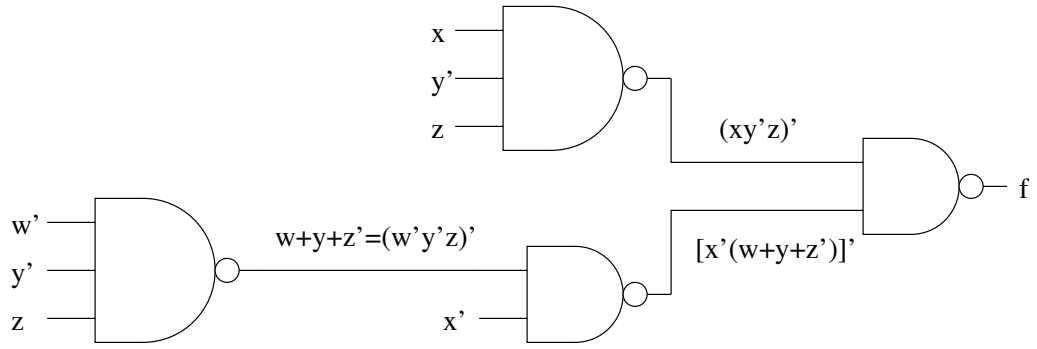



Figure C.10: NAND gate implentation of $f = xy'z + x'(w + y + z')$

### C.3.4 NOR

**Truth Table**

| $x$ | $y$ | $(x+y)'$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Logic Symbol**

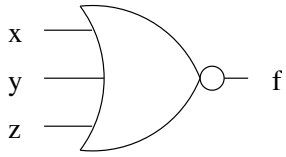



Figure C.11: NOR gate: $f = (x + y + z)' = x'y'z'$

**Equivalent Structure**

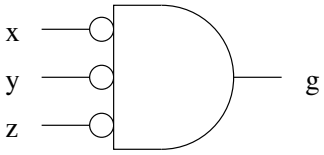



Figure C.12: AND gate (inverted inputs): $g = x'y'z' = (x + y + z)'$

It is possible to implement any Boolean operator using only NOR gates. Consider the following figures.

**Example:** The function $f = xy'z + x'(w + y + z')$ can be implemented using only NOR gates.

$$\text{NB:} \qquad \text{NOR}(g, h \ldots k) \quad = \quad [g() + h() + \ldots + k()]'$$

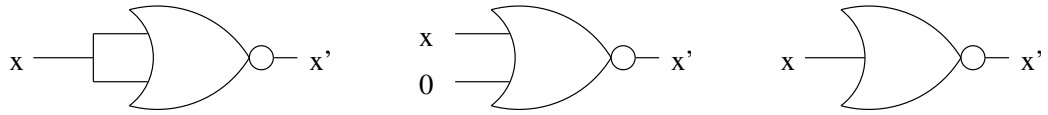



Figure C.13: NOT gate: $x' = (x+x)'$, $x' = (x+0)'$, $x' = \text{NOR}(x)$

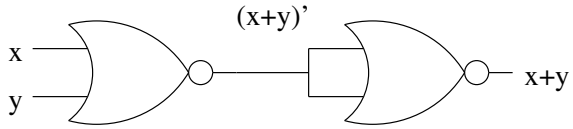



Figure C.14: OR gate: $x+y = [(x+y)']'$

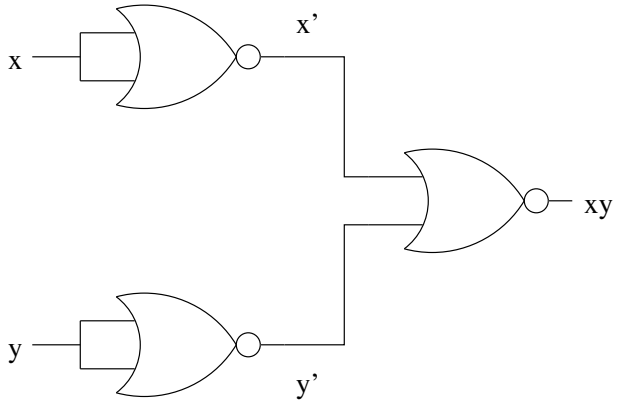



Figure C.15: AND gate: $xy = [(xy)']'$

$$\begin{aligned} f &= xy'z + x'(w+y+z') \\ f' &= \left[xy'z + x'(w+y+z')\right]' \end{aligned}$$

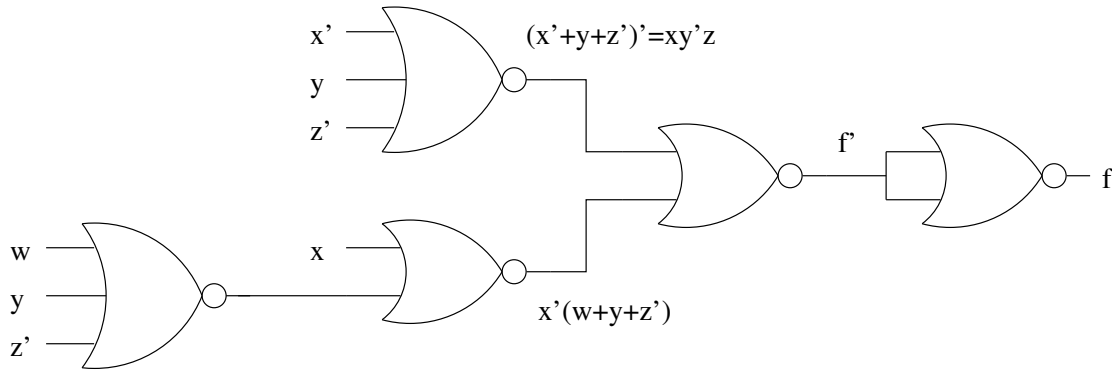



Figure C.16: NOR gate implentation of $f = xy'z + x'(w + y + z')$

# Bibliography


[1] M. Favalli and L. Benini, "Analysis of Glitch Power Dissipation in CMOS ICs," *Proceedings of ISLPED*, pp. 123-128, 1995.

[2] D. Rabe and W. Nebel, "New Approach in Gate-Level Glitch Modelling," *Proceedings of EURO-DAC*, 1996.

[3] R. Panda and F. N. Najm, "Technology-dependent Transformations for Low-power Synthesis," *Proceedings of DAC 97*, Anaheim, CA, 1997.

[4] F. N. Najm and M. Y. Zhang, "Extreme Delay Sensitivity and the Worst-Case Switching Activity in VLSI Circuits," *32nd ACM/IEEE Design Automation Conference*, 1995.

[5] V. Chandramouli and K. A. Sakallah, "Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time," *33rd ACM/IEEE Design Automation Conference*, Las Vegas, NV, Jun. 1996.

[6] Y. J. Lim, K. Son, H. Park and M. Soma, "A Statistical Approach to the Estimation of Delay-dependent Switching Activities in CMOS Combinational Circuits," *33rd ACM/IEEE Design Automation Conference*, Las Vegas, NV, Jun. 1996.

[7] H. Mehta, M. Borah, R. M. Owens and M. J. Irwin, "Accurate Estimation of Combinational Circuit Activity," *32nd ACM/IEEE Design Automation Conference*, 1995.

[8] D. Rabe and W. Nebel, "Short Circuit Power Consumption of Glitches," *Proceedings of ISLPED*, Monterey, CA, 1996.

[9] P. Kudva, G. Gopalakrishnan, H. Jacobson and S. M. Nowick, "Synthesis of Hazard-free Customized CMOS Complex-gate Networks Under Multiple-input Changes," *33rd ACM/IEEE Design Automation Conference*, Las Vegas, NV, Jun. 1996.

[10] M. Theobald, S. M. Nowick and T. Wu, "Espresso-HF: A Heuristic Hazard-free Minimizer for Two-level Logic," *33rd ACM/IEEE Design Automation Conference*, Las Vegas, NV, Jun. 1996.

[11] M. Sawasaki, C. Ykman-Couvreur and B. Lin, "Externally Hazard-free Implementations of Asynchronous Circuits," *32nd ACM/IEEE Design Automation Conference*, 1995.

[12] M. Hashimoto, H. Onodera and K. Tamaru, "A Practical Gate Resizing Technique Considering Glitch Reduction for Low Power Design," *Proceedings of DAC 99*, New Orleans, LA, 1999.

[13] M. Hashimoto, H. Onodera and K. Tamaru, "A Power Optimization Method Considering Glitch Reduction by Gate Sizing," *Proceedings of ISLPED 98*, Monterey, CA, 1998.

[14] A. Raghunathan, S. Dey and N. K. Jha, "Register Transfer Level Power Optimization with Emphasis on Glitch Analysis and Reduction," *IEEE Transactions on CADICS*, Vol. 18, No. 8, pp. 1114-1130, Aug. 1999.

[15] C. W. Moon and R. K. Brayton, "Elimination of Dynamic Hazards by Factoring," *30th ACM/IEEE Design Automation Conference*, 1993.

[16] J.M. Rabaey, "Digital integrated circuits: A design perspective," *Prentice-Hall Book Company*, 1st edition, ISBN #0-13-178609-1, Upper Saddle River, NJ, 1996.

[17] D. Burger and J.R. Goodman, "Billion-transistor architectures," *IEEE Computer*, Vol. 30, No. 9, pp. 46-48, Sep. 1997.

[18] *CPU Info Center*, http://infopad.eecs.berkeley.edu/CIC/summary/local.

[19] A.P. Chandrakasan and R.W. Broderson, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, Vol. 83, No. 4, pp. 498-523, Apr. 1995.

[20] M. Favalli and C. Metra, "The effect of glitches on CMOS buffer optimization," *Proc. PATMOS*, pp. 202-212, Oldenberg, Germany, Oct. 1995.

[21] M. Favalli and L. Benini, "Analysis of glitch power dissipation in CMOS IC's," *Proc. of ISLPED*, pp. 123-128, 1995.

[22] "Gated clocks and hazards," http://erebor.cudenver.edu/courses/ee3651/hazards/hazard.html.

[23] F.S. Hillier and G.J. Lieberman, "Introduction to operations research," *McGraw-Hill*, 1995.

[24] R.H. Katz, "Contemporary logic design," *Addison Wesley Publications*, 1993.

[25] J. Leijten, et al., "Analysis and reduction of glitches in synchronous networks," *Proc. EDAC*, pp. 398-403, 1995.

[26] N.R. Mahapatra and R. Janakiraman, "Gate triggering: A new framework for minimizing glitch power dissipation in static CMOS ICs and its ILP-based optimization," *Proc. IEEE ICCDCS 2000*, Cancun, Mexico, Mar. 15-17, 2000.

[27] N.R. Mahapatra, S.V. Garimella, and A. Tareen, "Efficient techniques for designing static CMOS ICs with very low glitch power dissipation," *ISCAS 2000*, Geneva, Switzerland, May 28-31, 2000.

[28] N.R. Mahapatra, S.V. Garimella, and A. Tareen, "An empirical and analytical comparison of delay elements and a new delay element design," *IEEE WVLSI 2000*, Orlando, FL, Apr. 27-28, 2000.

[29] E.J. McCluskey, "Logic design principles with emphasis on testable semicustom circuits," *Prentice Hall*, Englewood Cliffs, NJ, 1986.

[30] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," *International Conference on Computer-Aided Design*, pp. 398-402, Santa Clara, CA, Nov. 1993.

[31] F.N. Najm, "A survey of of power estimation techniques in VLSI circuits," *IEEE T-VLSI*, Vol. 2, No. 4, Dec. 1994.

[32] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM TODAES*, Vol. 1, No. 1, pp. 3-56, Jan. 1996.

[33] R.A. Powers, "Batteries for low power electronics," *Proc. IEEE 38*, Vol. 4, pp. 687-693, Apr. 1995.

[34] D. Rabe, et al., "Comparison of different gate level glitch models," *Proc. PATMOS'96*, Bologna, Italy, 1996.

[35] A. Raghunathan, S. Dey, and N.K. Jha, "Glitch analysis and reduction in register transfer level power optimization," *Proc. DAC*, pp. 331-336, Jun. 1996.

[36] S.J. Abou-Samra and A. Guyot, "Glitch threshold," *FTFC'97*, Paris, Nov. 1997.

[37] *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, San Jose, CA, 1994.

[38] P.S.K. Siegel, "Automatic technology mapping for asynchronous designs," Technical Report #CSL-TR-95-663, Stanford University.

[39] C. Small, "Shrinking devices put the squeeze on system packaging," *EDN*, pp 41-46, EDN 39, Feb. 1994.

[40] H.J.M.Veendrick, "Short circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid State Circuits*, Vol. 19, pp. 468-473, Aug. 1984.

[41] N.H.E. Weste and K. Eshraghian, "Principles of CMOS VLSI design," *Addison-Wesley*, 1993.

[42] T.-Y. Wu, et al., "A low glitch 10-bit 75-MHz CMOS video D/A converter," *IEEE JSSC*, Vol. 30, No. 1, Jan. 1995.

[43] Y. Watanabe, T. Ohsawa, K. Sakurai, and T. Furuyama, "A new CR-delay circuit technology for high-density and high-speed DRAM's," *IEEE Journal of Solid State Circuits*, Vol. 24, No. 4, pp. 905-910, Aug. 1989.

[44] A. Sekiyama, T. Seki, S. Nagai, A. Iwase, N. Suzuki, and M. Hayasaka, "A 1V operating 256-Kbit FULL CMOS SRAM," *Symposium of VLSI circuits, Digest of Technical Papers*, pp. 53-54, Jun. 1990.

[45] T.H. Lee and J.F. Bulzacchelli, "A 155-MHz clock recovery delay and phase-locked loop," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 12, pp. 1736-1746, Dec. 1992.

[46] A.W. Buchwald, K.W. Martin, A.K. Oki, and K.W. Kobayashi, "A 6GHz integrated phase-locked loop using AlGaAs/GaAs heterojunction bipolar transistors," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 12, pp. 1752-1762, Dec. 1992.

[47] G. Kim, M.-K. Kim, B.-S. Chang, and W. Kim, "A low-voltage, low-power CMOS delay element," *IEEE Journal of Solid State Circuits*, Vol. 31, No. 7, pp. 966-971, Jul. 1996.

[48] Y.-W. Pang, W.-Y. Sit, C.-S. Choy, C.-F. Chan and W.-K. Cham, "An asynchronous cell library for self-timed system designs," *IEICE Transactions on Information and Systems*, Vol. E80-D, No. 3, pp. 296-305, Mar. 1997.

[49] M.F. Aburdene, J. Zheng and R.J. Kozick, "New recursive VLSI architectures for forward and inverse discrete cosine transform," *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 2669, pp. 59-65, 1996.

[50] D.K. Jeong, G. Borriello, D.A. Hodges and R.H. Katz, "Design of PLL-based clock generation circuits," *IEEE Journal of Solid State Circuits*, Vol. 22, No. 2, pp. 255-261, 1987.