DESIGN AND IMPLEMENTATION OF A METASTABILITY TOLERANT LATCH

A Thesis

by

KINSHUK SHARMA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Sunil P. Khatri
Committee Members,	Peng Li
	Eun Jung Kim
Head of Department,	Miroslav M. Begovic

August 2016

Major Subject: Computer Engineering

ABSTRACT

Metastability causes unpredictable behavior in circuits, and can cause circuit failure. Any binary valued circuit element that holds state is vulnerable to metastability. Although the possibility of metastability cannot be completely eliminated in a circuit, the goal is to reduce it as much as possible. In this thesis, we discuss the design of a latch that effectively reduces metastability in circuits.

In today's SoC designs, different clock domains are often used for different functional units. If the clock domains are not synchronous, synchronizers are required for data crossing clock domains. A traditional synchronizer consists of 2 regular flip-flops and is not suited for high frequency operation. In this thesis, we present a new synchronizer design for high performance applications. The master as well as slave latches in the first flip-flop of this synchronizer use the metastability reducing latch. This latch has independent paths for the pull-up and pull-down transitions, thereby minimizing the possibility of metastability. Experimental results demonstrate a significant improvement in signal integrity compared to the traditional synchronizer. Our synchronizer also achieves an improvement in the worst case clock-to-output delay.

Metastability in asynchronous designs has not been given significant attention and metastability resolution is assumed to be handled by the handshaking protocol. However, metastability might manifest (at the electrical level) in various asynchronous circuit elements. One such asynchronous circuit element susceptible to metastability is the C-element. The C-element is vulnerable to metastability conditions at its output, for a short overlap in the input values. In this thesis, a robust design of a C-element is proposed based on our metastability reducing latch. Three popular

circuit topologies for a C-element have been studied and modified with the proposed approach. Experimental results show significant improvements in signal integrity, with up to 9× improvement in the metastability window.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Sunil Khatri for his support and guidance throughout my masters program. He has been a sublime mentor in the process and I feel lucky to have worked with him. My heartfelt thanks to him for helping me grow as a better student as well as a better professional. I would also take this opportunity to thank Dr. Li and Dr. Kim to have agreed to serve as members in my committee and for their guidance during my discussions with them.

I was also blessed with a very friendly and co-operative research group and would like to thank all the members of the group to have helped me during my research in one way or the other. All those thought provoking discussions that we had and the projects we did together helped me hone my technical skills. I would like to extend a special thanks to Mr. Monther Abusultan who helped me learn about HSPICE Simulations, Perl scripts and many other elementary things and was always ready to help with all my queries.

The ECE Department staff, specially Ms Carda has been very co-operative throughout my stay at Texas A&M University. I would like to thank them for all the help with the logistical issues.

I would like to thank my friends who have always motivated me to work for my research and have helped me get rid of my exasperation multiple times. Lastly, I would like to thank my family for helping me grow as a better individual each day. You guys have always made sure that I stay ambitious throughout my academic career. Mom, Dad and Anuashka, whatever I achieve in life, I owe it to you guys!

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In digital circuits, metastability is defined as the condition in which a circuit node holds an unstable state (between logic 0 and logic 1) for an unbounded duration. A metastable state is considered as an erroneous state because digital circuits are designed for binary valued logic and do not have a notion of mid-rail voltage value. Thus, metastability should be avoided at all costs in any digital circuit design. The reason why metastability is undesirable is because a) it causes an unpredictable logical behavior in the circuit, and b) may result in large current surges which can result in a system failure in certain cases.

Any binary valued circuit element that holds state (logic 0 or logic 1) is vulnerable to metastability while transitioning from one state to the other. Synchronous circuits use flip-flops or latches to hold state, and are clocked to implement control. In synchronous designs, timing violations (primarily setup and hold time violations) that can cause metastability can be avoided by design. However, at clock domain boundaries, the threat of metastability exists. *Synchronizers* are used to synchronize data that is transferred from one clock domain to the other, and minimize the likelihood of metastability. A significant amount of research effort has gone into avoiding metastability in synchronous circuits. Asynchronous circuits use state holding elements that operate on events and transitions (instead of clocks). For asynchronous circuits it is assumed that metastability can be avoided at the protocol level, by using handshake mechanism between different events. As a result metastability has not been heavily addressed at the electrical level in asynchronous designs.

In this thesis, we address the metastability issues in clock domain boundaries of synchronous designs as well in asynchronous circuits. We propose a novel circuit-level

design of a latch for avoiding metastability in digital circuits. We then demonstrate the use of this latch in a synchronizer design, which caters to the need of synchronization at high frequencies. In addition, we also use the proposed latch based approach in designing a robust C-element with improved metastability performance, to be used in asynchronous circuit design. The design of a synchronizer for high performance applications is presented in Chapter 2 while a robust C-element design with enhanced metastability performance is discussed in Chapter 3.

2. DESIGN OF A ROBUST SYNCHRONIZER FOR HIGH PERFORMANCE APPLICATIONS

2.1 Overview

As discussed in Chapter 1, the threat of metastability exists in synchronous circuits at clock domain boundaries and synchronizers are used for any signal that crosses clock domains. The conventional technique to synchronize data uses two flip-flops in cascade, which provides enough time for a signal to settle to a non-metastable state before it is passed to the receiving clock domain logic. In this chapter, we discuss the design of a synchronizer circuit which is based on the use of a specialized flip-flop. This specialized flip-flop uses the latch proposed in the thesis to avoid metastability. We begin the discussion with an introduction about clock domains and synchronizers, which is followed by a discussion on the previous work done to address metastability in synchronizers. We then describe our proposed latch based flip-flop design with a detailed description of experiments conducted to test our synchronizer.

2.2 Introduction

As VLSI designs become more complex, with diverse functionality integrated on the same System-on-Chip (SoC), the number of related (or unrelated) clock signals in the system increases. As a result, several clock "islands" (or *clock domains*) exist on the SoC, and sometimes within a single functional unit in the SoC. The clocks of these domains may be synchronous, mesochronous[1], plesiochronous[2] or asynchronous.

If the clocks are synchronous, synchronization of data requires a simple flip-flop,

[1]Mesochronous clocks have the same frequency, but constant unknown phase offset
[2]Plesiochronous clocks have "similar" frequency and non-constant, slowly varying phase

and the need for a special synchronizer does not exist. The slightly harder case is when the clocks are mesochronous, since it requires the use of a delay line to correct for the constant phase offset between the clock domains. The harder case is plesiochronous timing, with its slowly drifting clocks. Existing solution techniques are similar to those of mesochronous timing, but require periodic resynchronization. Asynchronous timing is hardest, since nothing can be assumed about the phase relationship between the two clocks. If an efficient technique is found for asynchronous timing, it can be used for mesochronous and plesiochronous timing as well. This motivates us to study synchronizers for asynchronous systems in our work.

Assume we have data that is driven from a driver in clock domain $CLK1$ and needs to be sampled by a receiver in clock domain $CLK2$ (without loss of generality, we assume that the driver (receiver) drives (samples) on the rising edge of $CLK1$ ($CLK2$). Whenever the rising edges of $CLK1$ and $CLK2$ are such that the D_2 signal arrives around the *setup time* (t_{su}) of the sampling flip-flop $FF2$, a non-rail value can be sampled in the receiving clock domain (Q_2), possibly resulting in *metastability*. This is illustrated in Figure 2.1. The goal of this thesis is to minimize the likelihood of metastability when such a signal crosses asynchronous clock domains.

A traditional synchronizer consists of two flip-flops connected back-to-back (as shown in Figure 2.2). This synchronizer would be used in place of $FF2$ of Figure 2.1.

There are several comments that can be made about synchronizers, at this point of the discussion:

- The reason it is undesirable to drive a non-rail value to downstream logic is that it could cause undesirable power spikes, potentially causing significant damage to the IC.

- The key idea behind using two flip-flops in this configuration is that the data

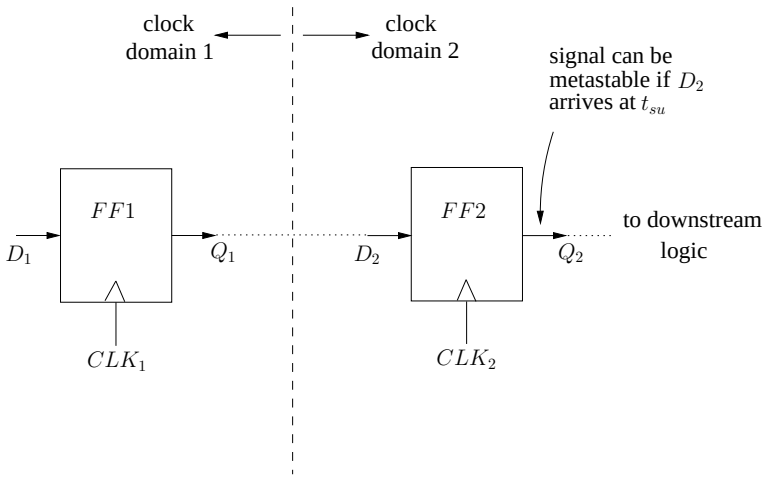

Figure 2.1: Synchronizing Data Across Clock Domains

that is driven to the downstream logic in the $CLK2$ clock domain is less likely to be metastable, since it goes through 2 back-to-back flip-flops before driving the downstream logic. This is because even if the first flip-flop produces a metastable (non-rail) value, the second flip-flop would likely resolve the value to a rail signal.

- The use of a specialized flip-flop as the first flip-flop in a two flip-flop synchronizer has reportedly been adopted in the industry. However, no literature was found which discusses the design of such a specialized flip-flop.

- Although the absence of metastability cannot be guaranteed by this (or any other) synchronizer, the goal of the synchronizer is to present rail values to the downstream logic, and thereby ensure that the sampled data has a stable value. The downside of a synchronizer is that it does not guarantee correct sampling of the signal that crosses clock domains (note that our design does not guarantee this either).

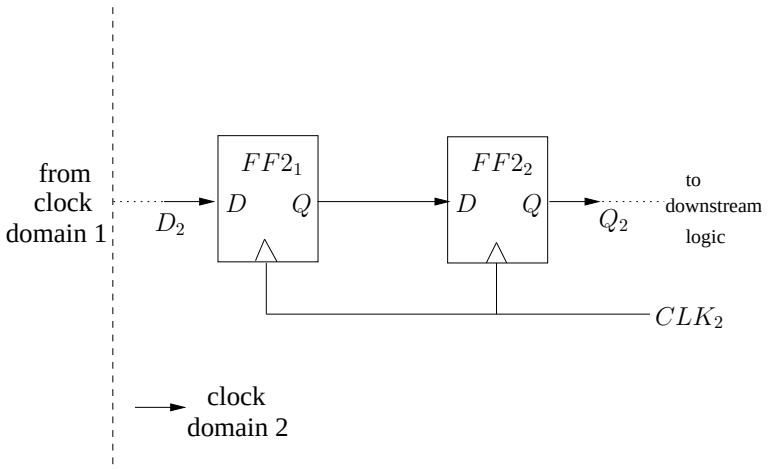

Figure 2.2: Traditional Synchronizer (Using Two Flip-flops)

The drawback of the traditional two flip-flop synchronizer is that it is not suited for high performance applications where the synchronization happens between two clocks, at least one of which operates at very high frequencies. This is because at high frequencies, the Q output of the first flip-flop of the synchronizer ($FF2_1$ in Figure 2.2) gets a very short time to settle to a rail value before the next clock edge, at which $FF2_2$ samples the output of $FF2_1$. It is thus important that the Q output of $FF2_1$ settles to a rail value quickly, when the clock $CLK2$ of Figure 2.2 operates at a high frequency. There are a wide variety of applications where this condition would occur in practice.

One such application is for the high speed data links that connect different processors in a multi-core microprocessor. Intensive research is being done to speed up the Network on Chip (NoC) in a Chip Multiprocessor (CMP). In some approaches, the interconnection network operates at a much higher frequency than the individual processors. The authors in [19] have proposed the design of a ring-based NoC based on the principle of resonant clocking. The data links in their design operate at a

frequency of about 14 GHz using a resonant standing-wave clock scheme. This is roughly 7× the frequency of the individual cores (processors) in the CMP. Figure 2.3 shows the design of a ring based NoC. Note that the routers handle the data communication between different cores and the rings. Since each core and the ring operate at different frequencies, and their clocks are not synchronous with each other, any data that goes from the ring clock domain to the core clock domain, and vice versa, has to be synchronized using a synchronizer embedded in an asynchronous FIFO.

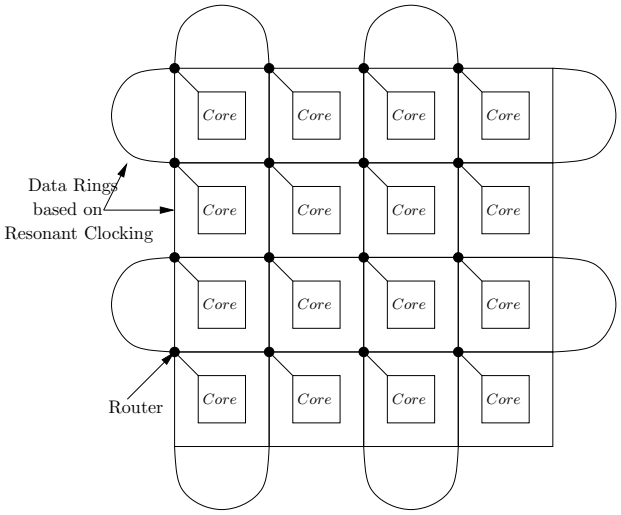

Figure 2.3: A 4x4 Ring Based Network on Chip

Figure 2.4 shows a block diagram of an asynchronous FIFO. Note that in the block diagram, the data is written to the FIFO in the ring clock domain (R_clk) and is read from the FIFO in the core's clock domain (C_clk). Thus, the read pointer is clocked by C_clk and write pointer is clocked by R_clk. The generation of FIFO empty and FIFO full flags requires the read and write pointers to cross clock domains. The use

of synchronizers is thus required. In Figure 2.4, $S1$ and $S2$ synchronize data from C_clk to R_clk and vice versa respectively. Note that $S2$ is clocked by C_clk (which is a slower clock) and hence $S2$ can be realized using two regular flip-flops. However, $S1$ is clocked by R_clk which is a high speed ring (network) clock. Thus, using the traditional synchronizer for $S1$ could result in poor metastability performance.

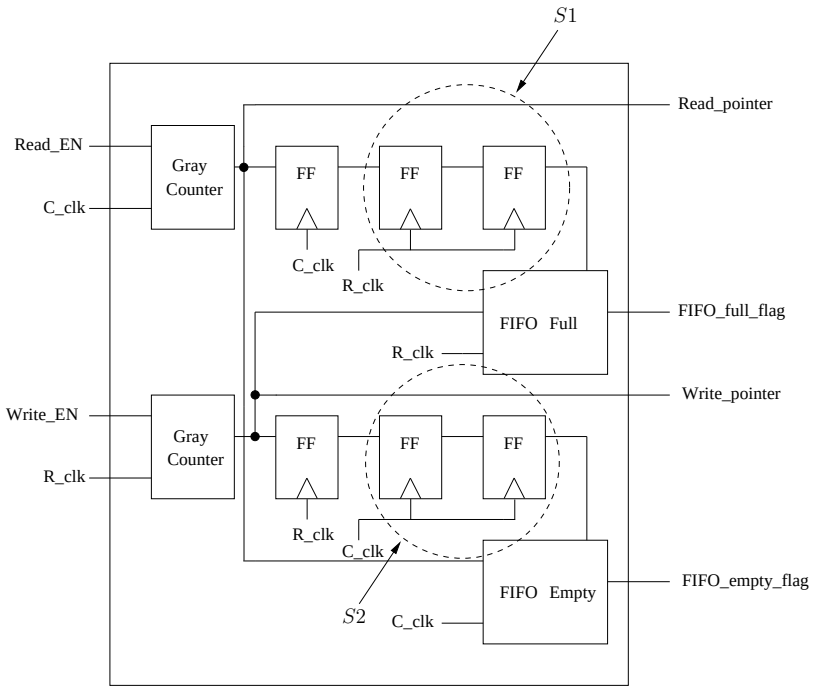

Figure 2.4: Block Diagram of an Asynchronous FIFO

For such applications, there is a need to improve the design of a traditional synchronizer by using specially designed flip-flop cells, such that the synchronizer can sample non-metastable data at higher frequencies.

Accordingly, the goals of this work are as follows:

- To design a custom synchronizer, which has a lower probability of producing a metastable output compared with the traditional synchronizer mentioned above.

- The benefit of this synchronizer would be that it will cater to the need of synchronization between clocks with very high frequencies, used in high performance applications. The metastability performance of the traditional two flip-flop synchronizer is not good at high frequencies, as discussed above.

The key contributions of this chapter are:

- A novel synchronizer design is presented. This synchronizer consists of two flip-flops. The master as well as slave latches in the first flip-flop both use independent paths for the pull-up and pull-down transitions, thereby reducing the possibility of metastability at its output, which is the input to a regular flip-flop (the second flip-flop of new synchronizer design).

- The proposed synchronizer includes a custom designed flip-flop that uses a specialized sampling circuit in both the master and slave latches, reducing the likelihood of metastability.

- Our experiments show that when compared to the traditional synchronizer (which uses two regular flip-flops and is commonly used in modern VLSI design practice), our synchronizer has significantly better signal integrity (metastability performance), with reduced clock-to-output delay (by up to 14%).

The remainder of this chapter is organized as follows. Section 2.3 discusses some previous work in the area of synchronizer design. Section 2.4 describes our new synchronizer design, while Section 2.5 reports the results of experiments we performed to compare our design with the existing synchronizer (which is based on the use

of two regular flip-flops). Conclusions are drawn in Section 2.6. In Section 2.7 we present the summary of the work presented in this chapter.

2.3 Previous Work

Metastability in synchronizers has been studied extensively over the years, and several theoretical papers have attempted to create a framework to understand and model it [28, 5, 3, 11]. These papers present a sound theoretical framework for the metastability problem, and provide mathematical models to estimate the MTBF in metastable systems.

Several practical circuit approaches have been presented to mitigate metastability in synchronizers. US Patents [13, 4] present a level sensing circuit to detect a metastable condition inside a sampling circuit, allowing a stable value to be propagated in the event of metastability. Our circuit avoids the use of level sensing, since this is a sensitive analog operation, and may be hard to complete within a clock period of a high-speed digital IC. The use of a Schmitt Trigger circuit in the forward path of a master latch in a flip-flop was presented in [24, 8]. The metastable points for the master and the slave are different. In contrast, our approach uses two inverters (one with a high switchpoint, and the other with a low switchpoint) in the forward path of both the master and slave latches of the specialized (first) flip-flop of our synchronizer. Both these latches are topologically identical in our case, unlike in [24, 8]. More esoteric approaches appear in [22, 12]. In [22], the authors use a 4-latch synchronizer, with complex internal feedback to resolve metastability. The author of [12], on the other hand, presents a control circuit which shifts at least one of the clocks of the two clock domains and generates new clocking signals when metastability is detected. Both these approaches are intricate in their design, while our approach focuses on a simple, practical and robust synchronizer design using

level-shifted inverters, which are commonly employed in custom IC design.

In [16], the focus is on integrating a jamb latch with a capacitor based charge pump. This technique is sensitive (on account of the charge pump, which is susceptible to capacitive coupling from noise in the digital IC), and also requires a metastability detector. Our approach avoids these circuits. In [29], the authors sample the signal using multiple flip-flops and use a multiplexer to select one of the outputs of these flip-flops. Our approach avoids such redundancy.

In a digital system with a data signal that traverses two clock domains, the clocks of these two domains can be synchronous, mesochronous, plesiochronous or asynchronous. Synchronization for these scenarios is discussed in [7]. In the first case, synchronization of data across the two domains is trivial, and just requires a simple flip-flop. As such, the synchronization problem does not exist since the two domains are synchronous. When the clocks are mesochronous, the situation is harder, since it requires the use of a delay line to correct for the constant phase offset between the clocks. Another approach uses a two-register synchronizer or a FIFO-based synchronizer [6]. Synchronizers for plesiochronous timing, with its slowly varying clocks, are harder to design to. Existing techniques in practice are similar to those of mesochronous timing, but require periodic resynchronization [6]. Asynchronous timing is hardest for a synchronizer, since nothing can be assumed about the phase relationship between the two clocks. If an efficient technique is found for asynchronous timing, it can be used for mesochronous and plesiochronous timing as well. This motivates us to study synchronizers for asynchronous systems in our work.

The authors in [19] focus on solving the problem of designing a fast scheme to communicate between different processing elements in the NoC of a multi-core system. The authors use a high speed clock to design a source-synchronous NoC.

An important component in this NoC design is the synchronizer that synchronizes data to the high speed clock domain. When operating at such high speeds, the metastability performance of the traditional synchronizer is not good, as we will demonstrate. For a variety of similar high performance applications, the need for a special synchronizer is paramount. Hence there is a motivation to come up with a synchronizer design that reliably samples a signal value when synchronizing at high frequencies, while minimizing the likelihood of metastability.

In this chapter, we propose a novel synchronizer based on the use of a special flip-flop. The key idea behind our synchronizer is that we sample the incoming signal by a specialized flip-flop which uses two inverters – one of which is a high switchpoint (HSP) inverter, while the other is a low switchpoint (LSP) inverter. The outputs of the HSP (LSP) inverter drives a NMOS (PMOS) driver stage, resulting in a situation where only one of the MOSFETs of the driver is conducting, thereby significantly reducing the likelihood of a metastable value being sampled. The output of this specialized flip-flop is then passed to the second flip-flop of our synchronizer, which is a traditional flip-flop. This style of synchronizer results in reduced clock-to-output delay, mainly because the sampled signals are significantly closer to rail values in our design compared to a traditional synchronizer.

2.4 Our Approach

In this section, we begin the discussion with a brief introduction of the figures of merit of a good synchronizer. Next, we briefly discuss the traditional synchronizer design, followed by a description of our synchronizer.

2.4.1 Figures of Merit

The key performance parameters of a sequential element such as a flip-flop are its setup time (T_{su}), hold time (T_h) and clock-to-output delay (T_{cq}). However in

synchronizers, since the data is synchronized between two clock domains, the clocks of which are asynchronous, the setup time and hold time of a synchronizer flip-flop are not relevant. The clock-to-output delay of a synchronizer is of importance because it determines the fastest a clock can operate in a particular clock domain. Another figure of merit of a synchronizer is the setup margin for the second flip-flop in a synchronizer. In Figure 2.2, the output of flip-flop $FF2_1$ sets up to the D input of flip-flop $FF2_2$. Now, if the output of $FF2_1$ is metastable, it might not settle to a stable value before the setup time (t_{su}) of $FF2_2$. This could cause the output of $FF2_2$ to be metastable also, or could cause variations in the clock-to-output delay of $FF2_2$. It it thus important that the output of $FF2_1$ provides a comfortable setup margin for $FF2_2$, and hence the setup margin of the second flip-flop of the synchronizer is important.

In our experiments, therefore, we will compare the clock-to-output delay of both our synchronizer and the traditional synchronizer. We will also compare the setup margins for the second flip-flop in both the synchronizers. Other parameters that will be compared in our analysis will be area utilization and power consumption. Additionally, we will also qualitatively analyze the waveforms of critical nodes in the synchronizer and compare their signal integrity.

2.4.2 Traditional Synchronizer

A traditional synchronizer consists of two flip-flops, each of which uses the topology shown in Figure 2.5. The operating principle is relatively simple. Since the output of the first flip-flop ($FF2_1$ in Figure 2.2) has the likelihood of being in a metastable state, its output is driven to the D input of a second flip-flop ($FF2_2$). The output of the second flip-flop has a lower likelihood of being in a metastable state, since the Miller capacitance associated with the pass-gates of the master latch

of $FF2_2$ (which pass the D input of $FF2_2$ into its master latch) would inject sufficient charge into the D and N signals of the master latch of $FF2_2$ when it was in a metastable state, resulting in an increased likelihood of a 0 or 1 being sampled and transmitted by the synchronizer output. Another way this can be explained (in an arguably less sophisticated manner from a circuit point of view) is to say that the likelihood of the final output of the synchronizer being metastable is based on the likelihood of the same metastable value being sampled for *two* clock periods (as opposed to one clock period for a single flip-flop). The use of two flip-flops reduces this likelihood significantly. The circuit for a traditional two flip-flop synchronizer has been shown in Figure 2.6. Note that RegFF in the figure refers to the circuit shown in Figure 2.5.

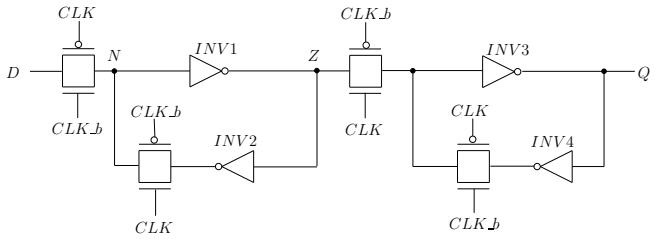

Figure 2.5: The Flip-flop Used in the Traditional Synchronizer (Referred as RegFF)

For RegFF, consider the signals N and Z shown in Figure 2.5. The signal N is the node that samples the value of the D input of the synchronizer during the low phase of the CLK signal. This signal is inverted by inverter $INV1$, the output of which we refer to as Z. The key idea is that as we vary the setup time of the D signal to the CLK rising edge, we would ideally like Z to never have a metastable value. If this is ensured, then the Q value can be guaranteed never to be metastable.

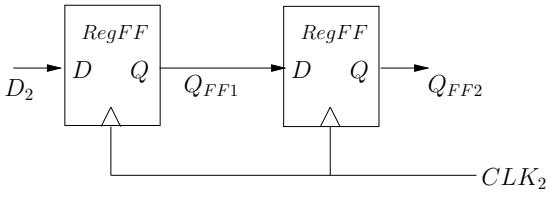

Figure 2.6: Traditional Synchronizer

In practice, however, this cannot be guaranteed. This is the reason we use two flip-flops back-to-back so that the possibility of metastability at the output of the second flip-flop is minimized.

The fundamental idea here is to: **a)** simulate a sweep of D inputs such that the separation in time domain of the D input and the active clock edge of the flip-flop is varied in each sweep **b)** compare metastability characteristics of the output of the synchronizer, when the D input switches at a time close to the setup time of the synchronizer cell.

It is useful to restate here that the main reason for the use of synchronizers is that the data being driven to the downstream logic, for a CMOS design, must always be driven to rail (logic 0 or logic 1) values in order to guarantee that power spikes that could be damaging to the design, are averted at all costs.

2.4.3 Our Synchronizer

In this chapter, we propose a novel synchronizer based on the use of a special flip-flop. The circuit of our synchronizer is shown in Figure 2.7. As opposed to using two regular flip-flop cells (RegFF) in a traditional synchronizer, our synchronizer uses a special flip-flop (SyncFF) followed by a regular flip-flop (RegFF). The circuit of our special synchronizer flip-flop is shown in Figure 2.8.

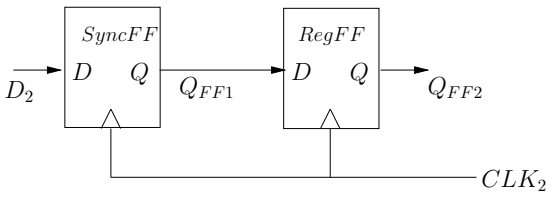

Figure 2.7: Our Synchronizer

The key idea behind our synchronizer flip-flop (SyncFF) is that instead of using just one inverter ($INV1$ in each of the latches of the traditional flip-flop of Figure 2.5) to sample the input when the master (or slave) latch of the flip-flop is transparent, we use two inverters which separately sample the master (or slave) latch input. The two inverters are chosen such that their values will be different, for a metastable value of the sampled latch input. One of these inverters is a high switchpoint (HSP) inverter $INV2$, while the other is a low switchpoint (LSP) inverter $INV1$ as shown in Figure 2.8. The output of the LSP and HSP inverters are inputs to a driver stage consisting of transistors $M1$ and $M2$. The HSP (LSP) inverter drives a NMOS $M2$ (PMOS $M1$). For the same non-rail value of the N signal, the HSP and LSP inverters have different output values on account of their different switchpoints, and as a result, the output of the driver stage (node Z) is highly unlikely to be in a metastable condition, since either of $M1$ or $M2$ is conducting at any time. Thus the likelihood of a metastable value being sampled is significantly reduced.

Pull-down Circuit

To understand the need of the extra pull-down structure in Figure 2.8 at the output of $INV2$, refer to the situation described in Figure 2.9. Initially, when the node N is falling slowly, the HSP inverter switches first while the LSP inverter has still not switched. This results in a situation where the LSP inverter output is low

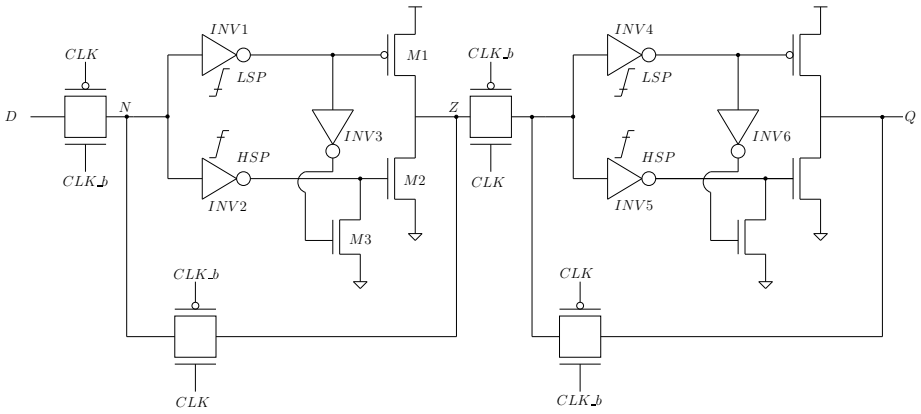

Figure 2.8: Our Modified Flip-flop Design (Referred as SyncFF)

while that of HSP inverter is high. This causes transistors $M1$ and $M2$ to turn ON together momentarily. When the node N is rising slowly, the output of the LSP inverter falls first while the HSP inverter output falls later. This results in a similar situation as described in the first case, and both transistors $M1$ and $M2$ are turned ON together momentarily. Such a situation will result in a steep momentary increase in power consumption, which needs to be avoided. To avoid both devices turning ON together we can either pull up the output of $INV1$ (to turn off $M1$) or pull-down the output of $INV2$ (to turn off $M2$). Our simulation results with both the options showed that the pull-down structure results in better response.

High Switchpoint and Low Switchpoint Inverters

The switch point of an inverter is given by $V_{sw} = \frac{V_{dd} - |V_{tp}| + V_{tn}\sqrt{\frac{\beta_n}{\beta_p}}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}}$. For an inverter with the NMOS and the PMOS having equal threshold voltages (in magnitude), the switchpoint of the inverter depends on the ratio $\frac{\beta_p}{\beta_n}$. If this ratio is more than 1, the switchpoint would be more than $\frac{V_{dd}}{2}$. If this ratio is less than 1, the switchpoint would be less than $\frac{V_{dd}}{2}$. If the ratio is equal to 1, then the switchpoint

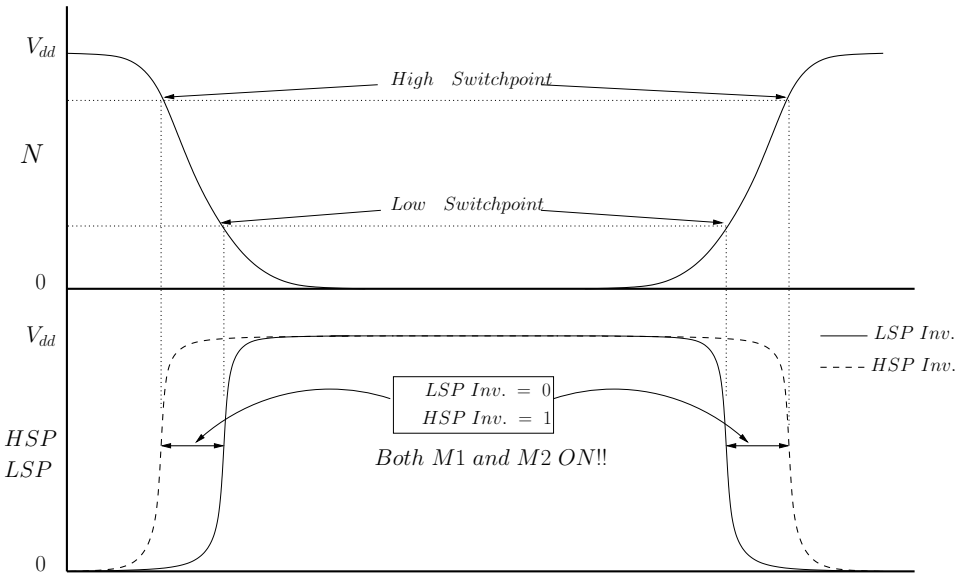

Figure 2.9: Response of HSP and LSP Inverter for Switching N Input

of the inverter is $\frac{V_{dd}}{2}$. These β values for an NMOS and a PMOS are defined as $\beta_n = \frac{\mu_N \epsilon}{t_{ox}} (\frac{W_N}{L_N})$ and $\beta_p = \frac{\mu_P \epsilon}{t_{ox}} (\frac{W_P}{L_P})$, (where ϵ is the permittivity of the dielectric, t_{ox} is the oxide thickness, μ_N and μ_P are the mobilities of charge carriers in the NMOS and PMOS respectively and W_N, L_N and W_P, L_P are the width and length of the NMOS and PMOS respectively). This has been further explained in Figure 2.10. The figure shows different voltage transfer characteristics of an inverter with varying β ratios. Thus, by varying the width and length of the devices in an inverter, we can tune the switchpoint of an inverter above or below $\frac{V_{dd}}{2}$. When the switchpoint is above (below) $\frac{V_{dd}}{2}$, we call the inverter an HSP (LSP). In our simulations, we used an HSP ($INV2$ in Figure 2.8) which had a switchpoint $30mV$ above $\frac{V_{dd}}{2}$ and an LSP ($INV1$ in Figure 2.8) which had a switchpoint $30mV$ below $\frac{V_{dd}}{2}$.

In our experiments, we will study the electrical characteristics of the Q_{FF1} and Q_{FF2} signals of both synchronizers (see Figures 2.6 and 2.7), and favor the design which exhibits fewer metastable conditions on these nodes as the setup time of D to

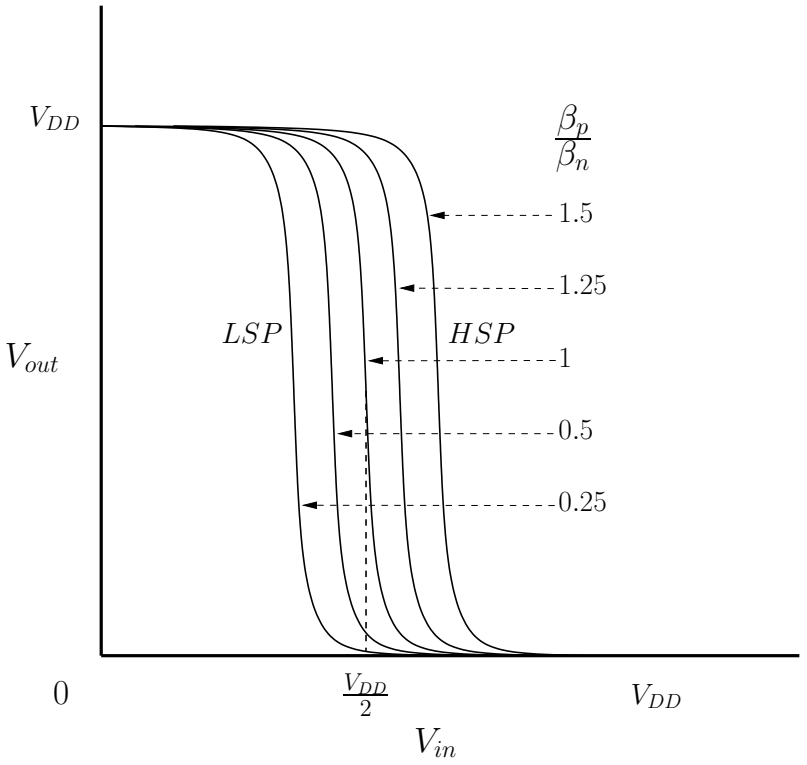

Figure 2.10: Inverter Voltage Transfer Characteristics for Different Beta Ratios

the rising edge of CLK is varied in very small increments.

2.5 Experiments

We simulated the proposed synchronizer as well as the traditional synchronizer in HSPICE [26], using a 14nm PTM [27] model card. Extensive sweeps were conducted to realize the device sizing that was used in the final design for our synchronizer. Based on these sweeps, the HSP inverter had a switchpoint which was 30 mV above the nominal, and the LSP inverter had a switchpoint which was 30 mV below the nominal. The difference in their switchpoint was 7.5% of the circuit supply voltage of 0.8V.

For each synchronizer, we tested its metastability performance by evaluating its ability to correctly capture a D signal. Assume that this D signal rose (or fell) T

time units before the rising (sampling) edge of the CLK. When the value of T is less than t_{su}, the synchronizer would fail to correctly capture the D signal. We sweep the value of T in extremely fine increments (we used 0.2 fs increments in our simulations). For values of T that are very close to t_{su}, we qualitatively as well as quantitatively evaluated the metastability of the synchronizer internal and output signals. Based on the reference example of the resonant clock based NoC discussed in [19], the clock (CLK) had a frequency of 14 GHz in our simulations.

We claim that the output node Q_{FF2} of the synchronizer, and the node Q_{FF1} of the first flip-flop of the synchronizer are critical signals in terms of metastability performance. Clearly, Q_{FF2} is critical since it is the output of the synchronizer, and hence its metastability characteristics are critically important. In addition, Q_{FF1} (in both our synchronizer as well as the traditional synchronizer, see Figures 2.7 and 2.6) is a signal which is crucial to the metastability performance of the synchronizer. This is because Q_{FF1} directly interfaces with the second flip-flop, and hence any metastability problems on Q_{FF1} could propagate further towards the synchronizer output, Q_{FF2}.

In Figures 2.11, 2.12, 2.13 and 2.14 we plot the overlay of the Q and D waveforms obtained over all our HSPICE sweeps of the value of T. These sweeps were conducted in 0.2 fs increments. We chose a window of 3ps around the setup time of the first flip-flop of both the traditional synchronizer (RegFF) and our synchronizer (SyncFF). We then swept the arrival of input signal D in 0.2 fs increments in that window. Thus, each sweep had 15000 simulations, each corresponding to a different D input signal. The window size, number of simulations and the sweep increments were kept same for both the synchronizers.

Figures 2.11 and 2.12 correspond to a falling D input being captured by the first flip-flop of the synchronizer. Figure 2.11 shows the overlay of waveforms for our

synchronizer and Figure 2.12 shows the overlay of waveforms for the traditional synchronizer. Each figure illustrates (from top to bottom) the voltage waveform of the CLK signal (CLK), the D input, the output of the first flip-flop of the synchronizer (Q FF1), and the output of the synchronizer (Q FF2).

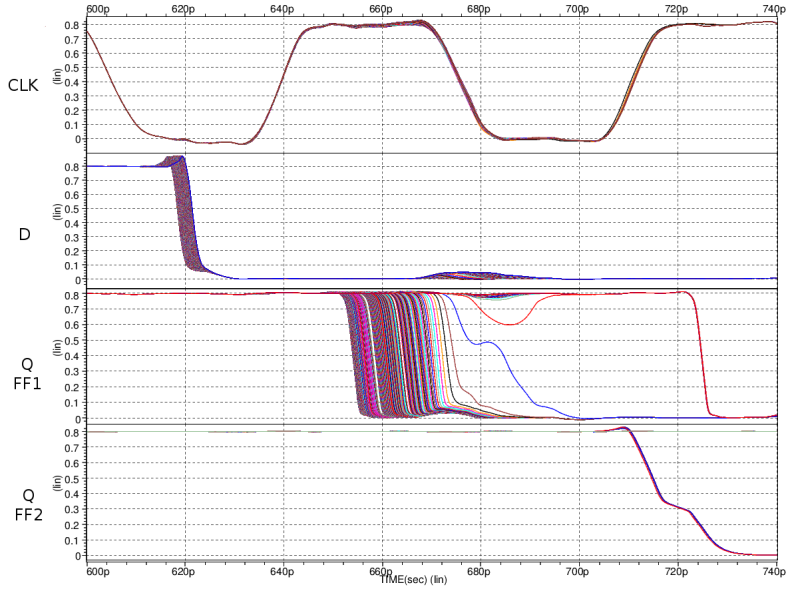

Figure 2.11: Overlay of 15000 Waveforms for Our Synchronizer for Falling D

Note that our synchronizer shows a significantly tighter band of Q_{FF1} signals in the sweep, with all signals demonstrating a very high slewrate. The traditional synchronizer has a much wider band of Q_{FF1} signals, many with extremely poor output slewrates. For this reason, the possibility of the second flip-flop in the traditional synchronizer sampling a metastable value at its input is high. This can result in high variation in the clock-to-output delay of the second flip-flop.

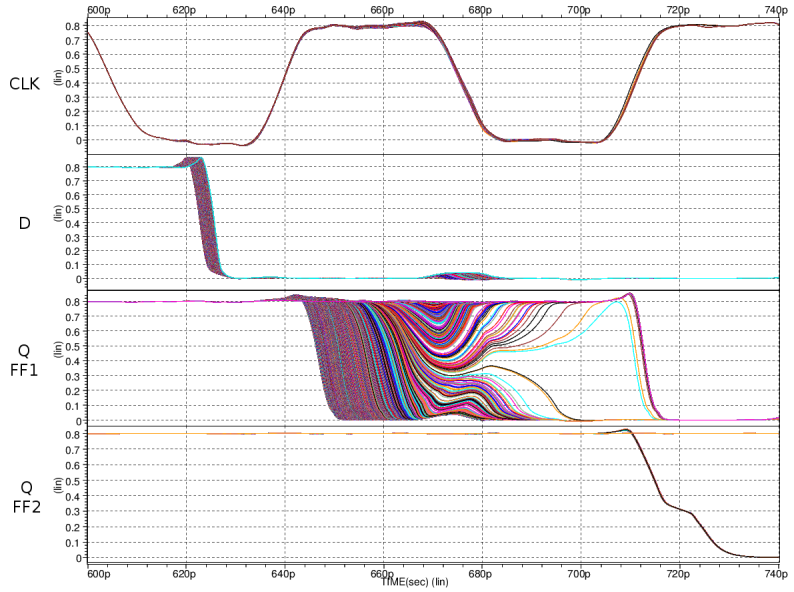

Figure 2.12: Overlay of 15000 Waveforms for Traditional Synchronizer for Falling D

Figures 2.13 and 2.14 correspond to a rising D input being captured by the first flip-flop of the synchronizer. Figure 2.13 shows the overlay of waveforms for our synchronizer and Figure 2.14 shows the overlay of waveforms for the traditional synchronizer. Again, each figure illustrates (from top to bottom) the voltage waveform of the CLK signal (CLK), the D input, the output of the first flip-flop of the synchronizer (Q FF1), and the output of the synchronizer (Q FF2).

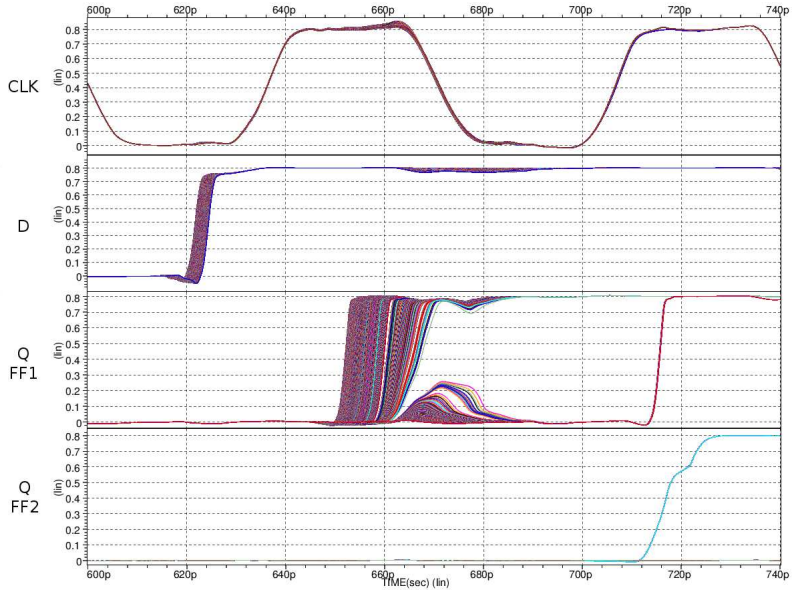

Figure 2.13: Overlay of 15000 Waveforms for Our Synchronizer for Rising D

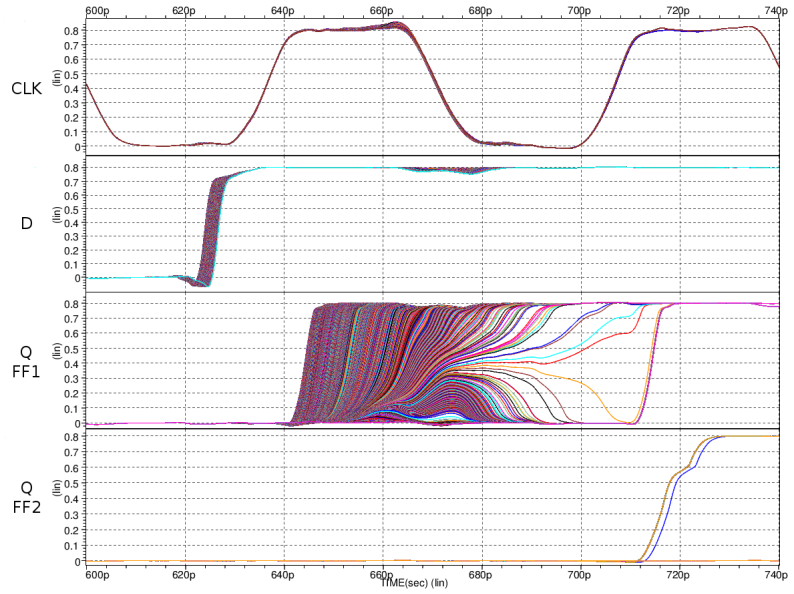

Figure 2.14: Overlay of 15000 Waveforms for Traditional Synchronizer for Rising D

We can again observe that for our synchronizer, the Q_{FF1} signal settles much before the rising edge of the CLK signal, compared to the traditional design, which shows several examples in which the Q_{FF1} signal stays at a metastable value, for a significant duration. The second flip-flop in our synchronizer, in contrast, samples a stable value for all the sweeps. Note the variation in clock-to-output delay of the second flip-flop of the traditional synchronizer (Figure 2.14).

Note that in all the figures discussed above, both synchronizers sometimes sample "wrong" values, which is because when the D input undergoes a transition very close to the setup time of the first flip-flop of the synchronizer, it can get sampled either as a logic 0 or a logic 1. No synchronizer can avoid this situation. The key requirement, however, is that a synchronizer needs to ensure that its output is close to the power rails.

In the six tables that follow, we report a quantitative comparison of the clock-to-output delay, the setup margin (for the second flip-flop of the synchronizer) and the average and peak power consumptions of the traditional synchronizer and our proposed synchronizer. Note that the absolute values reported in all these tables are specific to the sweep parameters we chose in our simulations (sweep of D input in a window of 3ps in 0.2fs increments). These numbers will change when the sweep increment is different from 0.2fs. The key thing is that even with the change in the sweep increment value, the trend of improvement in metastability performance of the proposed design is maintained, as we show later.

Table 2.1 reports, over all our sweep simulations, the worst case clock-to-output delay for both the traditional synchronizer and our synchronizer. We note that our synchronizer is faster (by about 0.1% and 14% when the D signal that is being captured is falling and rising respectively). This is because the input to the second flip-flop in our synchronizer is stable much before the setup time of the second flip-

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	5.56 ps	5.57 ps	0.18%
Setup to 1	10.21 ps	11.90 ps	14.20%

Table 2.1: Clock-to-Output Delay (Worst Case) Comparison

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	4.19 ps	-8.94 ps	146.9%
Setup to 1	16.57 ps	-15.23 ps	208.8%

Table 2.2: Setup Margin (for $FF2_2$) Comparison

flop. The second flip-flop of the traditional synchronizer samples many non-rail values, which increase the worst case delay of the traditional synchronizer.

Table 2.2 reports, over all our sweep simulations, the setup margin for the second flip-flop of the traditional synchronizer and our synchronizer. The setup margin is defined as the difference of the setup time of the second flip-flop of the synchronizer and the time τ, which is the latest time that any of the 15000 Q_{FF1} signals crosses 90% of VDD (if it is rising) or 10% of VDD (if it is falling). As discussed in the beginning of Section 2.4, the setup margin for the second flip-flop of a synchronizer is a good figure of merit for a synchronizer. We note that our synchronizer is much better in terms of setup margin which ensures the second flip-flop of our synchronizer comfortably samples a logic 0 or a logic 1 value. Whereas for the traditional synchronizer, the setup margins are negative which poses the threat of capturing a non-rail value by the second flip-flop of the traditional synchronizer.

As mentioned earlier, the absolute values of setup margin are specific to our experimental setup. Since setup margin is the metric that best represents the improvement of the proposed design of synchronizer over the traditional design, we also

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	15.50 ps	-5.85 ps	364.9%
Setup to 1	17.10 ps	-10.30 ps	266.0%

Table 2.3: Setup Margin (for $FF2_2$) Comparison for 1fs Sweeps

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	13.66 μW	6.58 μW	-107.6%
Setup to 1	13.38 μW	8.68 μW	-54.1%

Table 2.4: Average Power Consumption over One Cycle Averaged over all Sweeps

computed the value of setup margin for a different sweep increment (1fs). We report these numbers in Table 2.3. It must be noted that all the other simulation parameters except the sweep increment were kept same as that of the original experimental setup. Note that even for a sweep increment of 1fs, our scheme shows a significant improvement in setup margin.

We next present four tables which report power consumption comparisons. Note that for each of these tables, we compare the power consumption of our synchronizer with the traditional synchronizer.

To present the results of Table 2.4, we first compute, for each of the sweeps, the averaged circuit power P over one clock cycle. Now, this P value is averaged across all the sweep simulations, and reported in Table 2.4, for both the traditional and our synchronizer. We note that the averaged P value is more for our synchronizer (by about $2\times$ and $1.5\times$) when the D signal that is being captured is falling and rising respectively.

To present the results of Table 2.5, we first compute, for each of the sweeps, the averaged circuit power P over one clock cycle. Now, the maximum value P_{max} across

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	25.19 μW	9.34 μW	-169.7%
Setup to 1	20.26 μW	11.02 μW	-83.8%

Table 2.5: Average Power Consumption over One Cycle Maximum over all Sweeps

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	99.39 μW	38.21 μW	-160.1%
Setup to 1	78.02 μW	34.58 μW	-125.6%

Table 2.6: Peak Power Consumption over One Cycle Averaged over all Sweeps

all the sweep simulations is reported in Table 2.5, for both the traditional and our synchronizer. We note that for our synchronizer, the value of P_{max} is more (by about 2.7× and 1.8× when the D input is falling and rising respectively) when compared to the traditional synchronizer.

From Tables 2.4 and 2.5, we can see that our design consumes more power. This is because our design uses extra transistors in the first flip-flop of the synchronizer, which contribute to the increased average power consumption when the circuit nodes are switching.

To present the results of Table 2.6, we first compute, for each of the sweeps, the peak circuit power R over one clock cycle. Now, the average R value is found across all the sweep simulations, and reported in Table 2.6, for both the traditional and our synchronizer. We note that the averaged R value is more for our synchronizer (by about 2.6× and 2.2× when the D signal that is being captured is falling and rising respectively).

To present the results of Table 2.7, we first compute, for each of the sweeps, the peak circuit power R over one clock cycle. Now, the maximum value R_{max} is found

	Our Synchronizer	Traditional Synchronizer	Impr.
Setup to 0	100.5 μW	42.05 μW	-139.0%
Setup to 1	89.99 μW	39.11 μW	-130.0%

Table 2.7: Peak Power Consumption over One Cycle Maximum over all Sweeps

across all the sweep simulations, and reported in Table 2.7, for both the traditional and our synchronizer. Again, the value of R_{max} is more for our synchronizer (by about 2.4× and 2.3× when the D input is falling and rising respectively).

The peak power consumption in our circuit is more because as discussed earlier in Section 2.4, the transistors $M1$ and $M2$ are on together for a very short time. During this time, the power consumption is momentarily high.

We also generated layouts of both our modified synchronizer flip-flop (Figure 2.16) and the regular flip-flop (Figure 2.15). These layouts were generated using the TSMC 250nm Bulk CMOS process design rules. Based on the layouts shown, the area utilization of our synchronizer is higher than that of the traditional synchronizer by 26%, owing to the extra devices we use in the special flip-flop of our synchronizer.

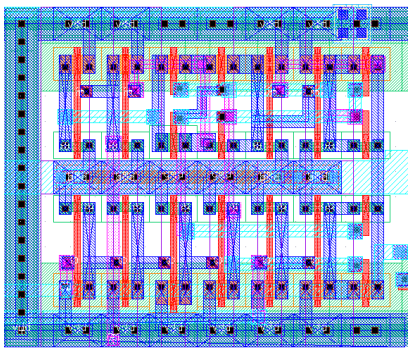

Figure 2.15: Layout of Regular Flip-flop

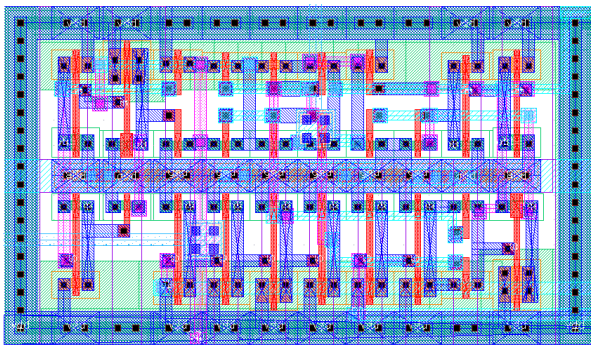

Figure 2.16: Layout of Our Synchronizer Flip-flop

As reported in this section, our proposed synchronizer circuit uses extra area and consumes extra power as compared to the traditional synchronizer. However, the metastability performance of our synchronizer is significantly superior, as seen from the waveforms and the results for setup margin and clock-to-output delay. Considering the importance of capturing a non-metastable value by a synchronizer, and given that there are a relatively small number of synchronizer cells in an SoC, the chip-level area and power penalties are lower. The superior metastability performance far outweighs these penalties for high-speed designs.

2.6 Conclusion

In today's IC designs, there are many instances where data signals cross clock domains. The most challenging clock domain crossing is one in which the two clocks are asynchronous. The traditional approach to synchronizing data in such a case uses two flip-flops. However it has been seen that the metastability performance of a traditional synchronizer is not good when the data has to be synchronized to a clock domain operating at a very high frequency. In this chapter we present a novel flip-flop based synchronizer. The circuit topology of our flip-flop uses two independent sampling paths for the data signal, so as to minimize the likelihood of

the output being metastable. Our synchronizer flip-flop design ensures that the data being sampled settles to a rail value quickly, allowing the synchronizer to operate at high frequencies. Such a design is suitable for a variety of high performance applications. Our synchronizer exhibits a significantly better setup margin (to the second flip-flop) in comparison to the traditional synchronizer (4.19 ps (16.57 ps) vs -8.94 ps (-15.23 ps) for a setup to 0(1)). The maximum clock-to-output delay is reduced with our synchronizer (by up to 14%), mainly because the sampled signals are significantly closer to rail values in our design when compared to a traditional synchronizer. Our synchronizer achieves better metastability performance with an area and power consumption overhead.

2.7 Summary

In this chapter, we describe the design of a novel synchronizer based on a specialized metastability reducing latch, which is used in the first flip-flop of a two flip-flop synchronizer. The design of this specialized flip-flop leverages the idea of using different switchpoint inverters in the latch, to avoid metastability. Our experiments show significant improvement in metastability performance with our synchronizer, with a nominal area and power overhead. In the next chapter, we will use the proposed metastability reducing latch to design a metastability tolerant C-element.

3. A ROBUST C-ELEMENT DESIGN WITH ENHANCED METASTABILITY PERFORMANCE

3.1 Overview

In this chapter, we discuss metastability issues that exist in the C-element, which is a state holding element in asynchronous circuit design. As mentioned in Chapter 1, any binary valued state holding element is vulnerable to metastability when transitioning between states. We propose a circuit-level scheme (based on the approach presented in Chapter 2) to improve the metastability performance (at the electrical level), of three popular C-element implementations. The discussion begins by a description of the C-element, and different existing implementations to realize its functionality. An in-depth analysis of the metastability performance of three traditional designs and our modified designs of C-element follows.

3.2 Introduction

The asynchronous circuit design style is often preferred for high performance and low power applications. This is because the signals in an asynchronous circuit do not need to wait for a clock edge in order to propagate, making it a faster design style. Also, asynchronous circuits consume lower power compared to synchronous circuits because in synchronous circuits, a clock has to toggle all or most of the time, which consumes power. Clocking often comprises 45-70% of the total power consumption of contemporary synchronous designs [14]. In recent times, globally asynchronous and locally synchronous (GALS) [15] designs have gained popularity because of the benefits of asynchronous circuits.

The C-element is a popular state holding element which is heavily used in asynchronous circuits. The C-element is a logic element that computes the AND function

on the events at its inputs. Here, an event is classified as a logic 0-1 or a logic 1-0 transition. The behavior of a C-element is such that the output matches the inputs if all the inputs are in the same logic state. If the states of the inputs are different the output holds its previous state. The behavior of a 2-input C-element is represented in Table 3.1.

Input a	Input b	Output c_{n+1}
0	0	0
0	1	c_n
1	0	c_n
1	1	1

Table 3.1: C-element Truth Table

As discussed earlier in Chapter 1, any state holding element is vulnerable to metastability while transitioning from one state to another. Hence, the output of C-element is also subject to a metastable condition. In the case of a very small overlap in the input values, the output of the C-element can begin to flip its state, but may not complete the transition to the new state. This is the primary electrical cause of metastability in the C-element. One set of input transitions that could result in metastability in a C-element is shown in Figure 3.1.

In the figure, initially the input a was a stable logic 0 and the input b was a stable logic 1. The output c was holding its last state which happened to be a logic 0. Now, a transitions to a logic 1 and the output c starts to rise (because both the inputs are logic 1). However, after a short period, and before the output completely transitions to a stable high value, the input b falls. The output c can now get stuck in a metastable state. Several other combinations of initial output state and input

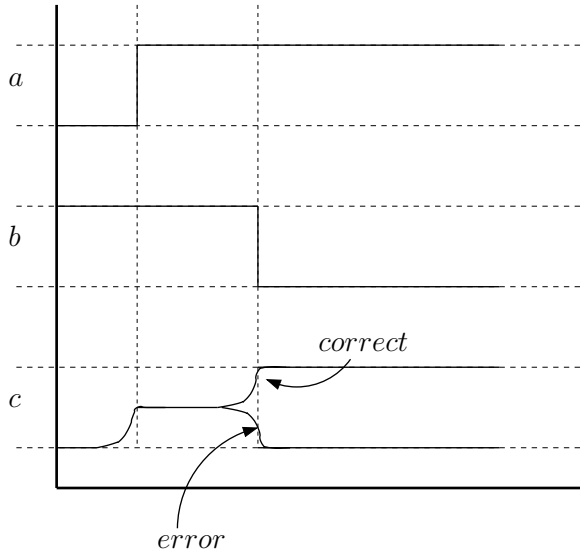

Figure 3.1: Input Transitions and Metastability in a C-element

transitions can cause metastability in the C-element. As a result, there is a need for a technique that avoids metastability in the C-element.

Over the years, various circuit-level implementations have been proposed to realize the functionality of the C-element. Three most popular implementations are the Static C-element (Figure 3.2), the Sutherland C-element (Figure 3.3) and the Van Berkel C-element (Figure 3.4). There are other gate-level implementations of the C-element that use standard cell based design to implement a C-element as well. In our work, we will focus on the circuit-level implementations of the Static, Sutherland and Van Berkel C-elements. A detailed discussion on the traditional and proposed circuits for these C-element implementations follows in Section 3.4.

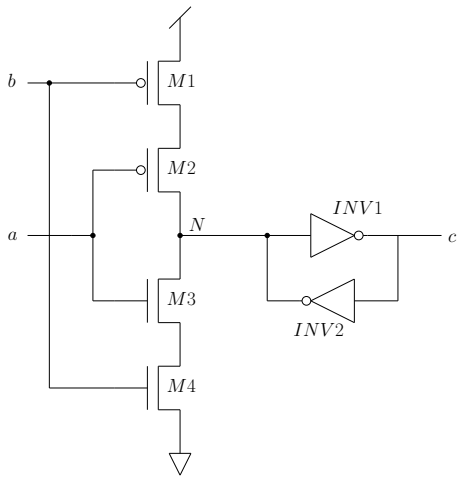

Figure 3.2: Traditional Static C-element

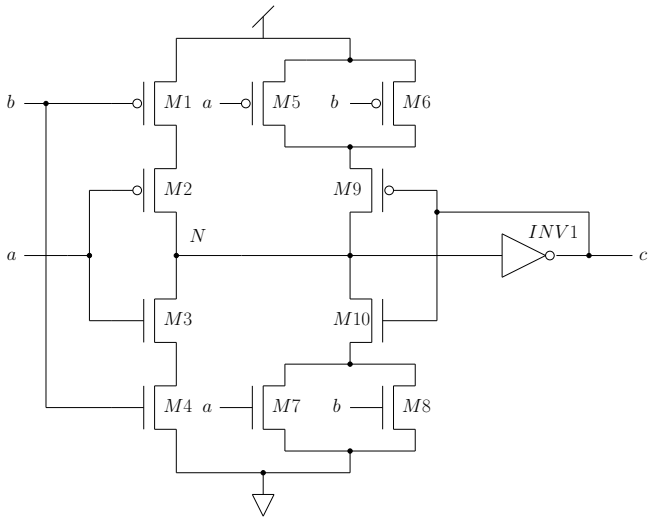

Figure 3.3: Traditional Sutherland C-element

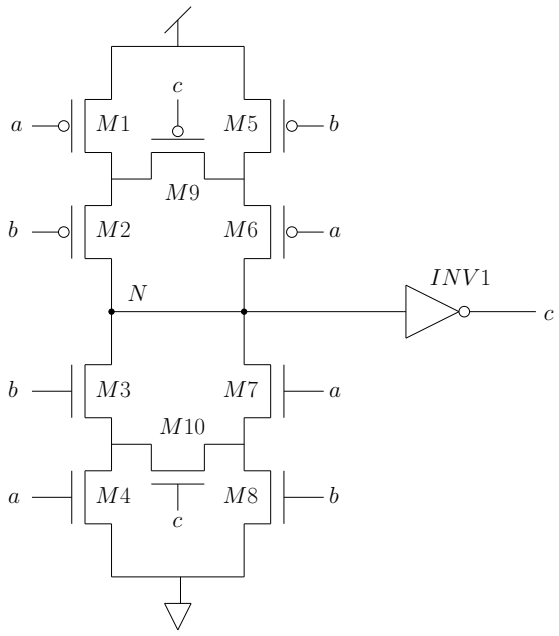

Figure 3.4: Traditional Van Berkel C-element

C-elements are used in self-timed circuits where every functional block produces an event (a transition on a signal) upon completion of its task. As discussed earlier in this section, the C-element can go into a metastable state for certain timing conditions on its inputs. In asynchronous designs, it is hard in general to coordinate events in such a way that they do not overlap briefly. The key thing to note here is that none of the C-element implementations mentioned above address the issue of metastability electrically. Additionally, no literature is found that presents an effective scheme to avoid metastability at the output of these C-elements. The importance of C-elements in asynchronous designs demands a deeper look into the circuit-level metastability behavior of C-elements.

Accordingly, the goals of our work are as follows:

- To compare the metastability performance of three popular circuits to realize C-elements, namely the Static, Sutherland and Van Berkel C-elements.

- Propose a new circuit-level approach which can be adopted for all the implementations of the C-element mentioned above, to reduce the possibility of metastability at the output of the C-element.

- Qualitatively as well as quantitatively analyze the metastability behavior of the proposed solutions, and compare it with the traditional designs.

- It is important to note that our design does not guarantee the absence of metastability (no other design guarantees this either). The goal is to minimize the likelihood of metastability as far as possible.

In this chapter we propose a circuit-level approach that minimizes the metastability conditions in a C-element. Our approach is based on the circuit of the latch described in Section 2.4 which uses two independent paths to sample a potential metastable signal with two inverters that have different switchpoints. Based on our simulations, the proposed approach achieves up to 9× improvement in the *metastability window*, and up to 6× improvement in the worst case delay of a C-element.

The remainder of this chapter is organized as follows. Section 3.3 discusses previous research in C-elements and metastability in asynchronous designs. Section 3.4 explains our approach in designing a metastability tolerant C-element, while Section 3.5 describes our experiment setup and reports the results of our experiments we conducted to compare our modified C-element implementations with the existing traditional C-element implementations. Conclusions are drawn in Section 3.6 while Section 3.7 presents a summary of the chapter.

3.3 Previous Work

Metastability conditions in synchronous circuits have been extensively addressed. However, metastability has not received similar attention in asynchronous circuits. Asynchronous circuits, with their handshake based control flow, are assumed to not suffer from metastable conditions. However, this is only true at the protocol level. In the work presented in this chapter, we address the metastability issues in Muller C-element, which is a fundamental building block in asynchronous circuit design. The authors of [21], elaborate eight possible combinations of transitions on the inputs of a two input C-element that could cause metastability. The effort in their work is to characterize metastability in a C-element using a late transition detection scheme. The authors of [9] show, with an example of a fault-tolerant clock generation scheme, that metastability can exist in asynchronous systems in the presence of a few fault effects.

Different circuit-level implementations have been proposed to realize the functionality of the C-element. A comparison of three most popular implementations has been presented in [23]. The authors compare the performance of a Static (referred to as Martin's), Sutherland and Van Berkel C-element in terms of their propagation delay, energy consumption and area utilization. In our work, we also choose these three implementations to analyze and compare the metastability performance of different C-elements. However, our analysis is focused on the metastability behavior of different C-element implementations. The design of a latch based speed independent C-element is proposed in [20]. Different circuit implementations of C-elements are compared in [1] on the basis of their robustness to the effects of Single Event Upsets (SEUs). A comparison is made between C-elements implemented in Bulk CMOS technology and SOI technology. The key parameters compared are the

delay, and the static and dynamic power of the different C-elements. However, the work in [20, 1] does not address metastability issues in C-elements. We compare the robustness of different C-elements under metastability-prone conditions at the electrical/timing level and then propose a circuit-level approach to design a more robust version of these C-elements. The authors of [30] propose a design for a multi-input C-element which has fewer restrictions on the number of inputs as compared to the traditional implementations. The work in [18] is focused on further improving the design presented in [30]. The authors of [18] discuss the possible race conditions at the output of the C-element when the inputs to the C-element arrive close to one another. In our work, we show how such situations can result in metastability at the output. The authors in [2] provide a quantitative estimate of the probability of error at the output of C-element depending on the error probabilities at the inputs.

A key observation is that metastability in C-elements has been acknowledged, studied and characterized. However, a circuit-level effort to reduce metastability in C-elements has not been made. Sutherland, in his discussion about micropipelines, lists various modules that provide the logical combination of transition events [25]. The C-element is one such module that provides the logical AND of transition events at its inputs. Its heavy use in asynchronous pipeline control motivates us to study C-element and propose a reliable design that is metastability tolerant.

3.4 Our Approach

In this section, we discuss the circuit-level implementations of 3 C-elements, and describe our approach to design a robust version of these C-elements with enhanced metastability performance. We first compare 3 traditional implementations of C-element and then discuss how our approach minimizes metastability in all these implementations.

3.4.1 Traditional C-element

The three popular traditional circuit-level implementations for a C-element that we will analyze are shown in Figures 3.2, 3.3 and 3.4. Let us begin with a brief explanation of the simplest C-element implementation, the Static C-element (shown in Figure 3.2). Assume both the inputs a and b are at logic 0. In this case, the transistors $M1$ and $M2$ are turned on and the node N is pulled up to logic 1. This drives the output to a logic 0 (same as both the input values). Now, without loss of generality, if b transitions to a 1, the transistors $M2$ and $M4$ are on, and the node N stays at its last value (logic 1). The output state is maintained through the feedback inverter $INV2$. The output changes state only when both the inputs transition to a logic 1, in which case transistors $M3$ and $M4$ turn on and the node N is pulled down, which drives the output high.

The other two implementations of the C-element (Sutherland and Van Berkel) work in a similar way. The difference between these two implementations and the Static C-element is the way the feedback structure is implemented. In the Static C-element, the feedback inverter $INV2$ is a weak inverter (implemented with long channel devices), which allows the transistors $M1$-$M4$ to drive the node N when the output is changing state. Depending on the timing of a and b this type of feedback can result in a race condition, when the weak feedback inverter $INV2$ and the input transistors ($M1$-$M4$) try to drive conflicting values to node N. Thus, the relative strength of the feedback inverter has to be carefully selected. The Sutherland and the Van Berkel implementations of the C-element avoid such a situation by designing their feedback structure differently.

Consider the operation of the Sutherland C-element (Figure 3.3). Assume a and b are logic 1, which causes N to be driven to logic 0, and c to be driven to logic 1.

Now, transistor $M10$ turns on as a result of c being at logic 1. As long as one of $M7$ or $M8$ are turned on (i.e as long as either a or b stay at logic 1), the node N is kept at logic 0 (through the transistor $M10$ and one or both of $M7$ and $M8$). This implements the "hold" functionality of the C-element. Only when *both* a and b are driven low, is this feedback broken. The analysis for the condition when c is logic 0 is symmetrical.

Finally, let us consider the operation of the Van Berkel C-element. When a and b are logic 1, the node N is driven to logic 0, and c is driven to logic 1. Now, as long as a stays at logic 1, the node N stays driven to logic 0 through the devices $M7$-$M10$-$M4$. Similarly, if b stays at logic 1, the node N stays driven to logic 0 through $M3$-$M10$-$M8$. Only when both a and b are driven to logic 0, is this feedback broken. The analysis for the case when c is logic 0 is symmetrical.

In the traditional C-element implementations, when the output is at a metastable value, a mechanism to rapidly drive the output to a rail value does not exist. Assume the output c in Figure 3.2 is stuck at a metastable value. This will cause both the devices (NMOS and PMOS) of $INV2$ to turn on and the voltage at N will be a mid rail value (if the inputs a and b have different values). This in turn causes both devices of $INV1$ to be on. Thus, the metastable voltage loops through the feedback path, and the time to resolve the metastable voltage is unknown. A similar problem exists in the Sutherland and Van Berkel C-element implementations. However, the superior feedback structure in these two implementations makes them less prone to metastability. In our experiments, we show that when the overlap in the value of inputs is small, all the traditional C-element implementations are liable to get stuck at a metastable state which is resolved after an unpredictable amount of time.

3.4.2 Modified C-element

In this work, we present a circuit-level approach which works with all the 3 implementations of the C-elements discussed above, and minimizes the possibility of metastability at the output of the modified C-element. Our proposed modification for the 3 implementations of the C-elements is shown in Figures 3.5, 3.6 and 3.7. For the sake of discussion, let us consider the design shown in Figure 3.5 (the modified Static C-element). As we can see, our design replaces $INV1$ of the Static C-element of Figure 3.2 with a structure that uses independent paths for falling and rising transitions on the output of the C-element. The working of the metastability reducing structure is similar to the latch used in our synchronizer design (Chapter 2), which is discussed in detail in Section 2.4.

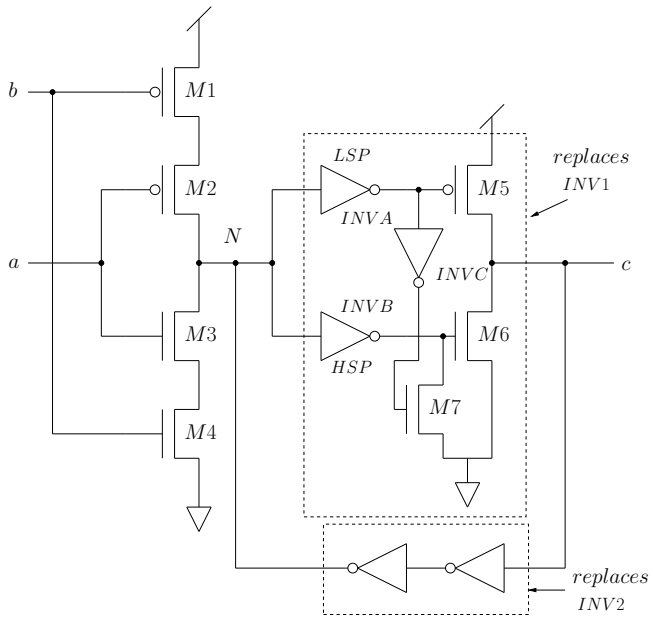

Figure 3.5: Modified Static C-element

With the use of our metastability reducing latch, a finite resolution time is achieved in our design depending on the delay of the pull-down structure. This cannot be ensured in a traditional design since there is no mechanism to actively resolve a metastable value and pull it to a stable high or low rail value. Since our metastability reducing latch is non-inverting, the feedback structure (which replaces $INV2$ of the Static C-element of Figure 3.2) consists of a pair of regular inverters.

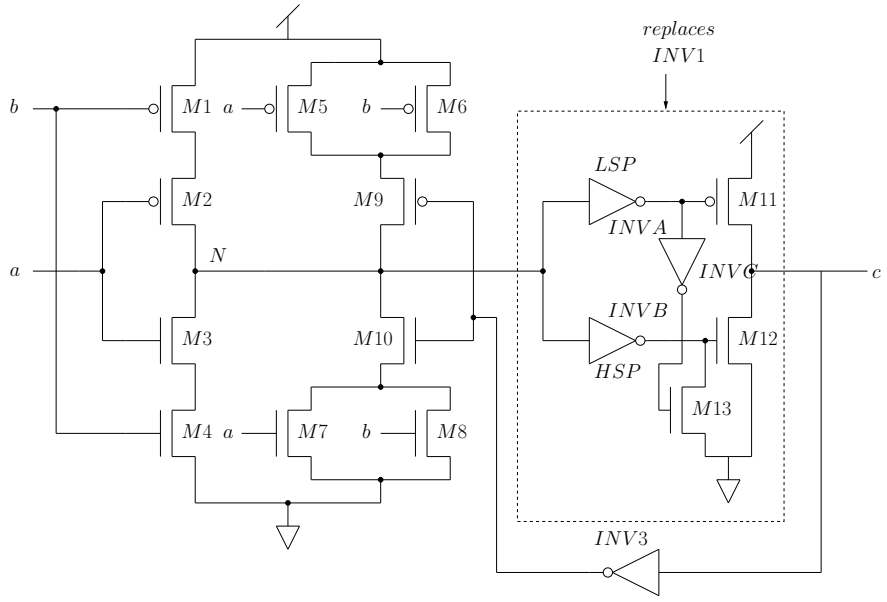

Figure 3.6: Modified Sutherland C-element

The metastability reducing latch discussed for the Static C-element (Figure 3.5) is effective for the other two C-elements as well. We use a similar structure in the Sutherland C-element (Figure 3.6) and the Van Berkel C-element (Figure 3.7).

In the Sutherland C-element, we replace $INV1$ (see Figure 3.3) with our metastability reducing latch. Since this latch is non-inverting, an inverter $INV3$ is intro-

duced in the feedback path, as shown in Figure 3.6. In the Van Berkel C-element, our metastability reducing latch replaces $INV1$ (see Figure 3.4). Our experimental results, which are discussed in the next section, show that our modified Sutherland and Van Berkel C-element resolves metastability much faster than the traditional design.

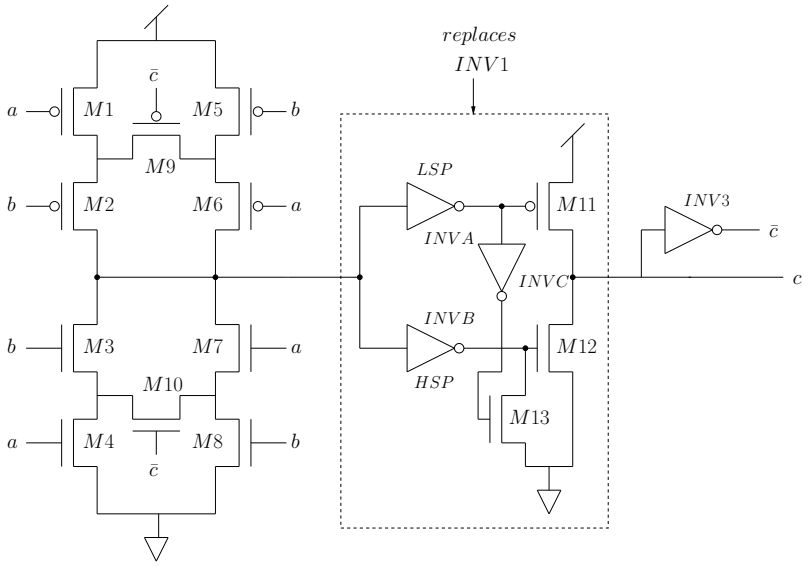

Figure 3.7: Modified Van Berkel C-element

As discussed in Section 2.4, we achieve different switchpoints for inverters by varying the width and length of the PMOS and the NMOS transistors in an inverter. In our design of C-element, we used an HSP inverter which had a switchpoint 30 mV above $VDD/2$, and an LSP inverter which had a switchpoint 30 mV below $VDD/2$.

3.4.3 Figures of Merit

We now discuss a few figures of merit which are key metrics to evaluate the performance of a C-element. We will compare our modified designs with the 3 traditional designs of a C-element based on these metrics. The common performance parameter for any combinational or sequential circuit element is its worst case delay from the input to the output. We will compare the *worst case delay* of our designs with the traditional designs. The other common parameters that we compare are the *area utilization* and the *power consumption* of the circuits.

In addition to these traditional metrics, we define another metric that quantifies the metastability performance of a C-element. This new metric, the *metastability window*, quantitatively measures the metastability performance of any C-element. A representative picture has been shown in Figure 3.8. In this figure, the input a goes through a transition from a high state to a low state at a fixed time. The input b goes through an opposite transition and the time of transition is swept. The output c, which happened to be at the high state initially, either transitions to a low state or stays high, depending on when the transition on b happens, relative to the transition on a. This has been shown in the figure. The metastability window is defined as the time from which the outputs in our sweep start a transition and reach 50% of the V_{dd} value, to the time when all the outputs from the sweep have settled within 10% of the power rail values. The metastability window for Figure 3.8, as defined above, would be either W_1 or W_2, whichever is larger. The metastability window is a representation of how long the output of the C-element stays in a metastable condition.

Other than the above metrics, we will also qualitatively analyze the signal integrity of the output of the C-element in our simulations of the circuit. Based on

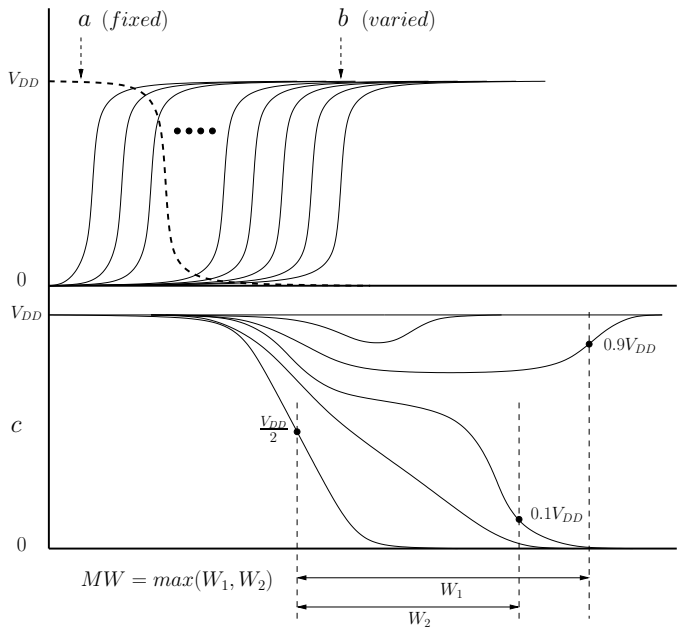

Figure 3.8: A Representation of Metastability Window

the qualitative and quantitative analysis, we will choose the better design.

3.5 Experiments

To compare the metastability performance of the 3 traditional C-element implementations and the 3 modified implementations proposed, we simulated the proposed designs as well as the traditional designs in HSPICE [26], using a 14nm PTM [27] model card.

To simulate the metastable condition in the C-element, we fixed the time of transition of one of the inputs (a) and swept the time of transition of the second input (b). This caused a short overlap in the input values (as shown in Figure 3.1). To observe metastability, sweeps of the b signal were conducted (as illustrated in Figure 3.8) in very fine increments of $1fs$. This input waveform set was connected to all the inputs of all the (6) implementations (3 traditional C-elements and 3 of

our modified C-elements). The signal integrity at the output of each C-element was then compared. We simulated two cases:

- The output c was initially set to 0 and the transitions on the inputs marginally triggered the output to rise
- The output c was initially set to 1 and the transitions on the inputs marginally triggered the output to fall

To sweep the transition of input b while the time of transition of input a is fixed, we chose a window of $140ps$ around the time of transition of a. For some of these sweeps, the output held its previous state, for other sweeps, the output went through a complete transition, flipping to the opposite stable state, and for the remaining sweeps, the output got stuck at a metastable value. A total of 140,000 sweeps were performed.

Figures 3.9 and 3.10 show the overlay of all the swept waveforms of the inputs a and b, the output of the traditional Static C-element and our modified Static C-element. Figure 3.9 shows the case when the output c was initially set to 0 and Figure 3.10 shows the case when c was initially set to 1. Note that since our design is inverting, the output is the compliment of a traditional C-element in all cases.

Each figure shows, from top to bottom, the input a, the input b, the output of a traditional C-element (c reg), and the output of our proposed C-element (c mod).

We see from Figures 3.9 and 3.10 that the output resolves to a rail value much faster in our design as compared to the traditional Static C-Element for both the cases. The metastability window is smaller by a factor of about $3\times$ and $9\times$ for our design. The traditional Static C-element has the risk of the output being stuck at a metastable value for a long time (up to $460ps$ in our experiments), which could cause unpredictable behavior in the combinational logic that follows the C-element.

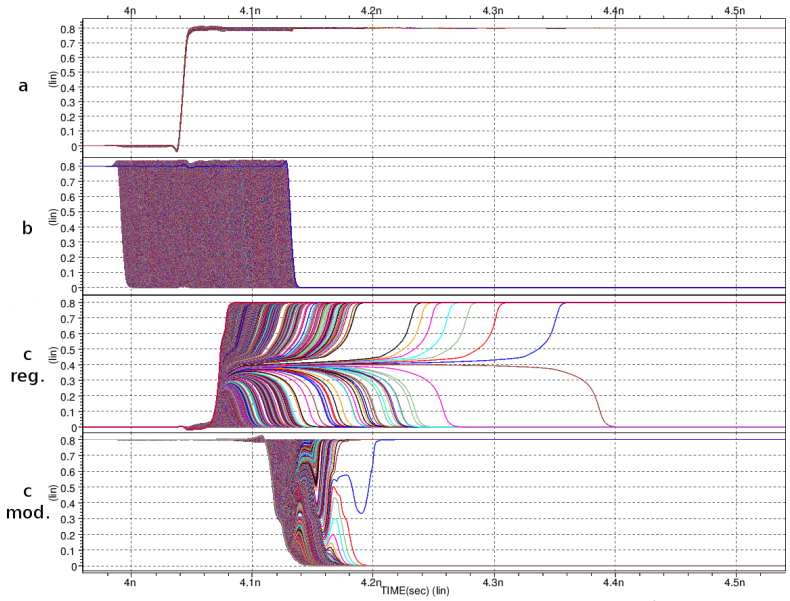

Figure 3.9: Overlay of c Waveform for Static C-element with c Initially Set to 0

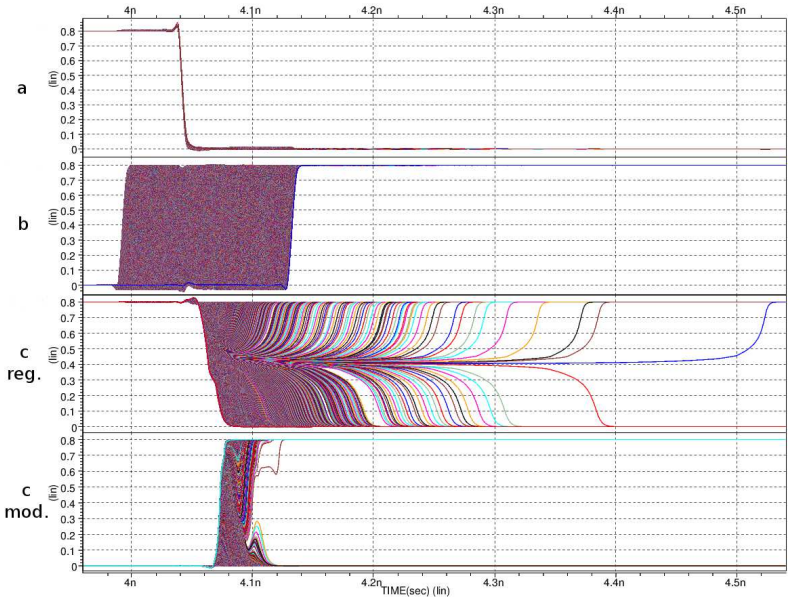

Figure 3.10: Overlay of c Waveform for Static C-element with c Initially Set to 1

Figures 3.11 and 3.12 show the overlay of waveforms of the inputs a and b, and the output of the traditional Sutherland C-element and our modified Sutherland C-element, for all our sweeps. Each figure shows, from top to bottom, the input a, the input b, the output of a traditional C-element (c reg), and the output of our proposed C-element (c mod).

We see that the traditional Sutherland C-element has much better metastability performance than the traditional Static C-element. This is because of the difference in the feedback structure between the two C-elements. In the Static C-element, the feedback and the input drivers may be active at the same time, trying to drive conflicting values at node N (see Figure 3.2). The Sutherland C-element avoids such a condition, which prevents a conflicting value being driven from two separate drivers. This makes the output much less metastable.

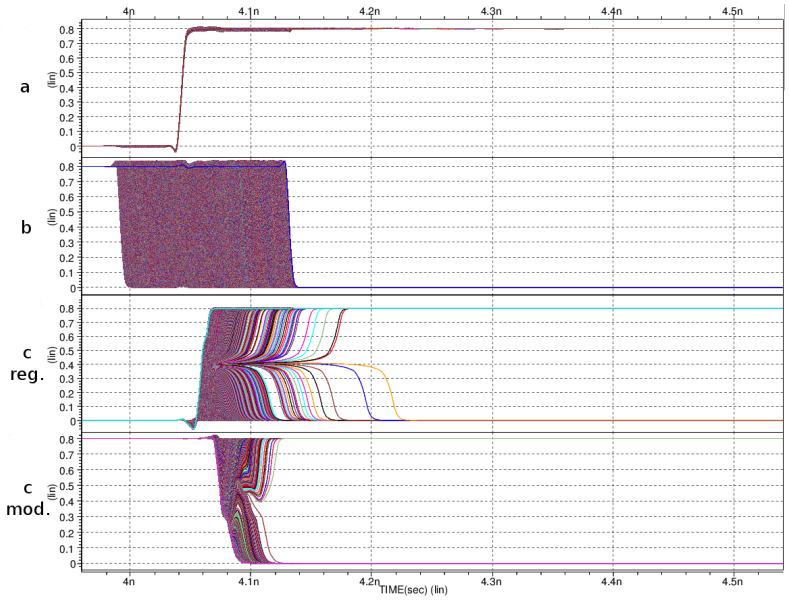

Figure 3.11: Overlay of c Waveform for Sutherland C-element with c Initially Set to 0

However, the traditional Sutherland C-element, when compared to our modified version, is worse in terms of resolving metastability. We see that our design of Sutherland C-element reduces the metastability window by about 3× and 2× as compared to the traditional Sutherland C-element.

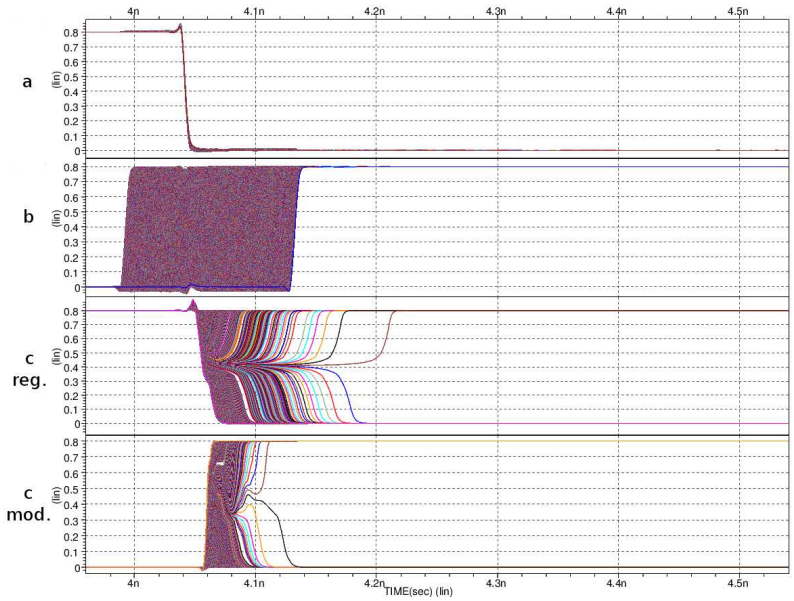

Figure 3.12: Overlay of c Waveform for Sutherland C-element with c Initially Set to 1

Figures 3.13 and 3.14 show the overlay of waveforms of the inputs a and b, and the output c of the traditional Van Berkel C-element and our modified Van Berkel C-element, for all our sweeps. Each figure shows, from top to bottom, the input a, the input b, the output of a traditional C-element (c-reg), and the output of our proposed C-element (c-mod).

As can be seen, the metastability performance of a Van Berkel C-element is similar to that of a Sutherland C-element and much better than a Static C-element. The reason for this is similar to the one given for Sutherland C-element.

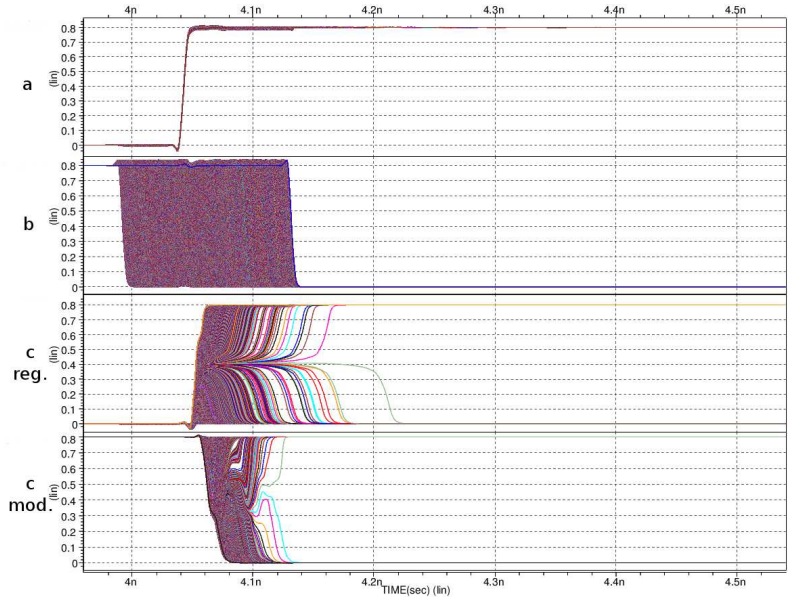

Figure 3.13: Overlay of c Waveform for Van Berkel C-element with c Initially Set to 0

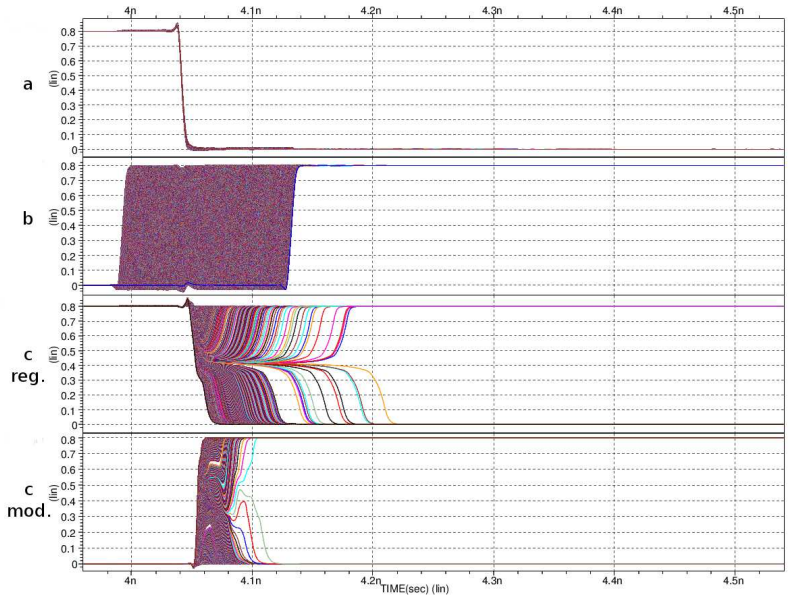

Figure 3.14: Overlay of c Waveform for Van Berkel C-element with c Initially Set to 1

Additionally, we can also see from Figures 3.13 and 3.14 that our design for Van

Berkel C-element is better than the traditional Van Berkel C-element in resolving metastability. We achieve about 2× and 3× reduction in the metastability window.

The key merit of our proposed design is the fast resolution of metastability at the output of the C-element. As opposed to the traditional design, where the time needed to resolve the metastable output to a logic 0 or 1 is large, our design makes sure that the output is resolved to a stable rail value quickly whenever it is metastable. This can be seen in Table 3.2 where we report, across all our sweeps, the metastability window for different circuit implementations of C-elements for both, the traditional designs and our modified designs.

Recall from the discussion in Section 3.4 that the metastability window is measured from the time when the output starts to switch and reaches 50% of the rail value, to the time when all the outputs from our sweeps have settled within 10% of the rail values. As seen in Table 3.2, our design resolves the metastable output much quicker as compared to the traditional C-element implementations. We achieve improvement in the metastability window (by 2.3× to 9×) compared to the traditional implementations. This makes our design significantly more robust. We see that our design is consistently better than the traditional design for all the three styles of C-elements studied in our work.

As discussed above, the key merit of our proposed design for different C-element implementations is the fast resolution of metastability at the output. This merit is quantified by the metastability window which we reported in Table 3.2 for the experiment setup we used. To demonstrate that the improvement in metastability window follows the same trend for a different sweep increment (of input b), in Table 3.3, we additionally report the metastability window for all the C-element implementations for a different simulation where we chose the sweep increments to be 5fs. All the other simulation parameters were kept same as that of the original simulation setup.

Static C-element			
	Our Design	Traditional Design	Impr.
c = 0	82.5 ps	315.5 ps	3.8×
c = 1	49.4 ps	460.5 ps	9.3×
Sutherland C-element			
	Our Design	Traditional Design	Impr.
c = 0	46.5 ps	160.5 ps	3.4×
c = 1	65.9 ps	154.8 ps	2.3×
Van Berkel C-element			
	Our Design	Traditional Design	Impr.
c = 0	63.2 ps	161.4 ps	2.5×
c = 1	56.9 ps	157.7 ps	2.8×

Table 3.2: Metastability Window Comparison

Static C-element			
	Our Design	Traditional Design	Impr.
c = 0	57.8 ps	315.5 ps	5.4×
c = 1	34.5 ps	308.6 ps	8.9×
Sutherland C-element			
	Our Design	Traditional Design	Impr.
c = 0	37.7 ps	100.8 ps	2.7×
c = 1	40.2 ps	102.9 ps	2.6×
Van Berkel C-element			
	Our Design	Traditional Design	Impr.
c = 0	63.2 ps	98.7 ps	1.6×
c = 1	39.4 ps	123.8 ps	3.1×

Table 3.3: Metastability Window Comparison for 5fs Sweeps

We notice that even with the change in the sweep increment, our proposed designs have a significantly smaller metastability window as compared to the traditional implementations.

The worst case delay of a C-element is critical in asynchronous circuit design. In Table 3.4, we report the worst case delay for all the implementations of the C-element, both for our modifications and the traditional designs. In all our sweeps, we

simulate the C-elements in such a way that always the transition on the output c is caused by a transition on the input a. Thus we measure the worst case delay from the transition on the input a to the transition in the output c. This has been illustrated in Figure 3.15. Note that we compute the delay from the time when the transition on a crosses 50% of V_{dd} to the time when the output c has settled down within 10% of the rail values. We do not use the conventional definition of delay (50% value of a to 50% value of c) because when the output c is metastable, it reaches 50% of VDD quickly, but often stays there for a long time. In such cases, the conventional definition of the delay would not be a true representation of the actual delay of the C-element.

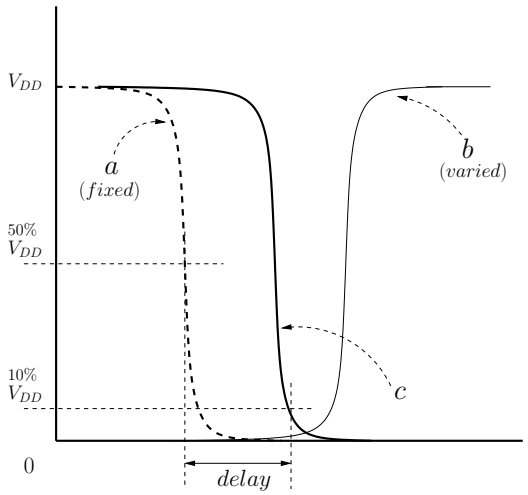

Figure 3.15: Description of the Delay of the C-element

We can see from Table 3.4 that the worst case delay is better for our modified designs of a C-element (by 2× to 6×) compared to the traditional designs. The worst case delay is better for our designs for all the implementations of the C-element. This

Static C-element			
	Our Design	Traditional Design	Impr.
c = 0	157.8 ps	345.7 ps	2.2×
c = 1	80.2 ps	481.8 ps	6.0×
Sutherland C-element			
	Our Design	Traditional Design	Impr.
c = 0	78.8 ps	176.8 ps	2.2×
c = 1	83.7 ps	168.8 ps	2.0×
Van Berkel C-element			
	Our Design	Traditional Design	Impr.
c = 0	83.7 ps	171.4 ps	2.0×
c = 1	69.4 ps	168.9 ps	2.4×

Table 3.4: Worst Case Delay Comparison

enables designers to design faster circuits using our design of a C-element.

In the next four tables we report the power consumption of different circuit implementations for C-elements, both for the traditional designs and our modified designs. To compute power, we define a time window (PW), which is shown in Figure 3.16. We chose the time window in which the output (c) of a C-element goes through two different transitions. The first transition is the one where c is initially set to 1 and the sweep of transitions on one of the inputs (b) causes the output to transition to 0 for some sweeps and stay at 1 for the other sweeps. In all these sweeps we witness many cases when the output is stuck at a metastable state. The other transition on the output in our window PW is the opposite of the first transition we just described. Thus, we include both the possible metastability cases while computing power consumed by a C-element. The size of PW window for our simulations was 4ns. Figure 3.16 is a representative figure of our simulation setup for computing power.

To present the results of Tables 3.5 and 3.6, we first compute the average power (A) consumed by different C-element implementations being compared, for each

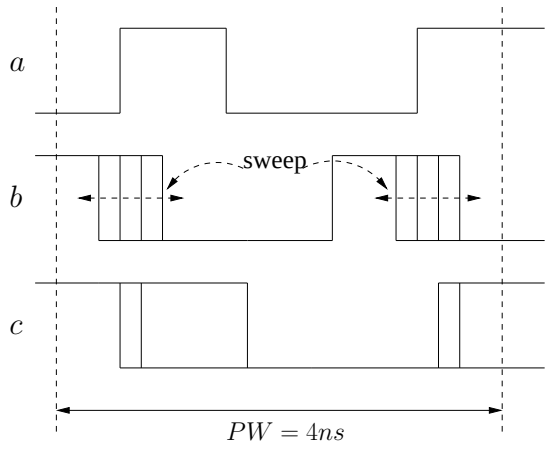

Figure 3.16: Time Window for Computing Power

Static C-element		
Our Design	Traditional Design	Impr.
0.765 μW	0.317 μW	$-2.4\times$
Sutherland C-element		
Our Design	Traditional Design	Impr.
0.365 μW	0.155 μW	$-2.3\times$
Van Berkel C-element		
Our Design	Traditional Design	Impr.
0.41 μW	0.178 μW	$-2.3\times$

Table 3.5: Comparison of Average Power Consumption Averaged over Sweeps

sweep. This average power is measured over the time window PW described above.

In Table 3.5, we report, across all our sweeps, the average value of A. It is to be noted that the average value of A for our proposed designs is more than the traditional designs (by up to 2.4×) for all the implementations of the C-element.

In Table 3.6, we report, across all our sweeps the maximum value (A_{max}) of all the A values. Again, A_{max} is larger for our designs as compared to the traditional C-element (by up to 4×), over all the implementations of the C-element.

Static C-element		
Our Design	Traditional Design	Impr.
2.077 μW	0.513 μW	$-4.0\times$
Sutherland C-element		
Our Design	Traditional Design	Impr.
0.808 μW	0.263 μW	$-3.0\times$
Van Berkel C-element		
Our Design	Traditional Design	Impr.
0.999 μW	0.281 μW	$-3.6\times$

Table 3.6: Comparison of Average Power Consumption Maximum over Sweeps

The average power consumption for our design is more than the traditional design of a C-element as seen in Tables 3.5 and 3.6. This is because our modifications of the C-element have extra devices that contribute in resolving metastability at the output. The power consumed by these extra devices adds to the average power consumption.

To report the results of Tables 3.7 and 3.8, we first compute the peak power P (for each sweep) consumed by the different implementations of the C-element over the time window PW described earlier in this section.

In Table 3.7, the average value of P is reported across all our sweeps. We see that the average P value is more for our design (by up to $3.2\times$) compared to the traditional design.

To report the numbers in Table 3.8, we compute the maximum value P_{max} of P from all our sweeps. We see that the maximum P value is more for our design (by up to $2.8\times$) compared to the traditional design.

The peak power consumed by our design is more than the traditional design for all the implementations of the C-element because of the fact that in our design, the NMOS and PMOS of the output driver are momentarily on together. During that time, the peak power consumption is high.

Static C-element		
Our Design	Traditional Design	Impr.
107.05 μW	41.43 μW	$-2.6\times$
Sutherland C-element		
Our Design	Traditional Design	Impr.
113.28 μW	35.67 μW	$-3.2\times$
Van Berkel C-element		
Our Design	Traditional Design	Impr.
131.31 μW	42.11 μW	$-3.1\times$

Table 3.7: Comparison of Peak Power Consumption Averaged over Sweeps

Static C-element		
Our Design	Traditional Design	Impr.
110.68 μW	48.20 μW	$-2.3\times$
Sutherland C-element		
Our Design	Traditional Design	Impr.
116.17 μW	41.68 μW	$-2.8\times$
Van Berkel C-element		
Our Design	Traditional Design	Impr.
135.02 μW	53.80 μW	$-2.5\times$

Table 3.8: Comparison of Peak Power Consumption Maximum over Sweeps

An important point to note about all the absolute numbers for metastability window, delay and power comparisons presented in this section is that all these values are obtained using our simulation setup. These numbers will change with the change in simulation parameters (sweep window and sweep increment value). However, the improvement in metastability performance of our proposed design of C-element is maintained for different values of sweep increments.

The product of width (W) and length (L) of a transistor is a representative number for the on-chip area it occupies. In Table 3.9 we report the summation of the product (W*L) for all the devices ($\sum W_i L_i$) used in implementing different circuit implementations for the traditional and our C-elements. Is is to be noted that our

Static C-element		
Our Design	Traditional Design	Impr.
0.0284 μm^2	0.0157 μm^2	$-1.8\times$
Sutherland C-element		
Our Design	Traditional Design	Impr.
0.0259 μm^2	0.0132 μm^2	$-1.9\times$
Van Berkel C-element		
Our Design	Traditional Design	Impr.
0.0259 μm^2	0.0132 μm^2	$-1.9\times$

Table 3.9: Comparison of $\sum W_i L_i$ for C-elements

C-element design uses more on-chip area (by about 1.9×) when compared to the traditional design.

We observe that our design needs more area and consumes more power as compared to the traditional design. However, these overheads usually do not impact the overall area and power consumption of an IC significantly. This is because the number of C-elements in an IC is usually limited. Robust operation of a C-element is a significantly bigger concern, as it affects the functionality and yield of an IC.

3.5.1 Monte Carlo Simulations for Process Variation

Since our design uses high switchpoint and low switchpoint inverters with fine-tuned switchpoints, it is necessary to check the robustness of the design against process variations, to ensure that the metastability performance is not affected. To validate the metastability performance of our circuit with process variations taken into account, we performed Monte Carlo simulations in HSPICE.

To simulate the circuit behavior with process variation effects, it is necessary to first identify the set of variation sources. The authors in [32] show that the threshold voltage (V_{th}) and the channel length (L) are the primary sources of variations in circuit behavior. For 14nm FinFETs, the ratio of standard deviation to mean for

channel length (σ_L) is 1.3% [17]. The variations in threshold voltage are caused by short channel effect and random dopant effect, and the total standard deviation of threshold voltage variation is a summation of the deviation caused by short channel effect and random dopant effect, since both these effects are independent sources of fluctuations [17]. However, in a sub-45nm FinFET device, if the channel doping is smaller than $1e^{17}cm^{-3}$, the variation in V_{th} is immune to random doping concentrations [31]. In our analysis, we thus accounted for variations in V_{th} caused by short channel effect. The ratio of standard deviation to mean is 3.2% for variations in V_{th} caused by short channel effect [17].

We adjusted the V_{th} variation depending on the length and width of the device, as explained in [10]. The authors in [10] explain that the standard deviation of the threshold voltage $\sigma_{V_{th}}$ is a function of the square root of the width of the device. They also state that the standard deviation of the threshold voltage is lower-bounded by half the $\sigma_{V_{th}}$ of a minimum sized device. The relation between $\sigma_{V_{th}}$ and the width W of a device as given in [10] is:

$$\sigma_{V_{th}}(W) = max\{\sigma_{V_{th}}(W_{min})\sqrt{\frac{W_{min}}{W}}, \frac{\sigma_{V_{th}}(W_{min})}{2}\}$$

To take both the length and width of the device into account, we used the following formula to calculate the effective $\sigma_{V_{th}}$ of a device having length (L) and width (W) different from the minimum values (W_{min}, L_{min}):

$$\sigma_{V_{th}}(W, L) = max\{\sigma_{V_{th}}(W_{min}, L_{min})\sqrt{\frac{W_{min}L_{min}}{WL}}, \frac{\sigma_{V_{th}}(W_{min}, L_{min})}{2}\}$$

The 3-sigma deviation in gate length for a FinFET device with gate length of 16nm is shown to be 0.7nm in [17]. However, the absolute deviation in gate length due to process variations remains constant for a given process. Thus, the ratio of standard deviation to mean (σ_L) for devices with gate length other than 16nm is different than the σ_L for a device with gate length 16nm. Therefore, we also adjusted the standard deviation in length (σ_L) for a device with channel length (L) using the

following formula:

$\sigma_L(L) = \sigma_L(16nm)\{\frac{16nm}{L}\}$

Due to run time limitation of simulations, we varied the channel length and threshold voltages of selected devices in the traditional and our modified C-elements. We took process variations into account in the devices that contributed in the latching action in a C-element, since the latching mechanism significantly affects the metastability performance. As a result, the devices that were *not* varied in the Monte Carlo simulations are as follows:

- For the Traditional Static C-element (Figure 3.2) and the Traditional Sutherland C-element (Figure 3.3), the devices $M1$, $M2$, $M3$ and $M4$ were not varied.
- For the Modified Static C-element (Figure 3.5), the devices $M1$ to $M7$ and the devices in $INVC$ were not varied.
- For the Modified Sutherland C-element (Figure 3.6), the devices $M1$ to $M4$, $M11$, $M12$, $M13$ and the devices in $INVC$ were not varied.
- For the Modified Van Berkel C-element (Figure 3.7), the devices $M11$, $M12$, $M13$ and the devices in $INVC$ were not varied.

In this section, we present the results of our Monte Carlo simulations for process variations, in the form of superimposed waveforms of inputs a and b, and the output c of the traditional C-element and our modified C-element, for all our Monte Carlo simulations, similar to the waveforms presented earlier in this section. These waveforms were obtained from a fine sweep of the transition time of b (refer to the beginning of this section) in increments of 40 fs in each sweep. For each unique b signal, we ran 100 Monte Carlo sweeps with different values of channel length (L) and threshold voltage (V_{th}) for the devices chosen for variation in our Monte Carlo

simulations. The variation in values of L and V_{th} was taken to be $\pm 3\sigma$ from the mean value. We chose a window of 140 ps around the transition time of input a of the C-element, and swept the transition time of input b in that window in 40fs increments. Thus, each set of sweeps had 350,000 simulations (3500 unique b signals and 100 Monte Carlo sweeps for each for each unique b signal). The run time for simulations held us from doing more Monte Carlo simulations with a sweep increment finer than 40fs.

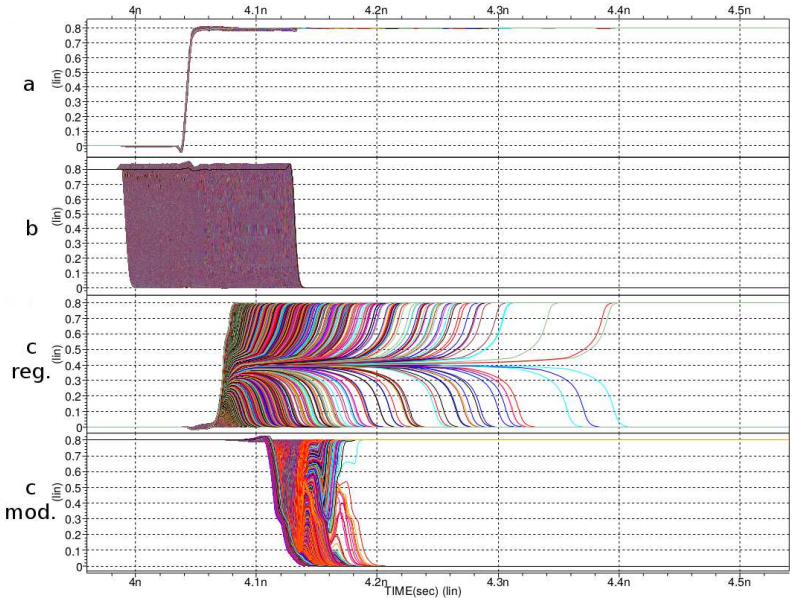

Figure 3.17: Overlay of Monte Carlo Waveforms for Static C-element with c Initially Set to 0

Figures 3.17 and 3.18 show the overlay of 350,000 waveforms for the Static C-element for the case when the output c is respectively set to 0 and 1 initially. Figures 3.19 and 3.20 show similar waveforms for the Sutherland C-element and Figures 3.21 and 3.22 show the overlay of waveforms for the Van-Berkel C-element. All

of these figures show, from top to bottom, the input a, the input b, the output c of the traditional C-element (c reg.) and the output c of our modified C-element (c mod.).

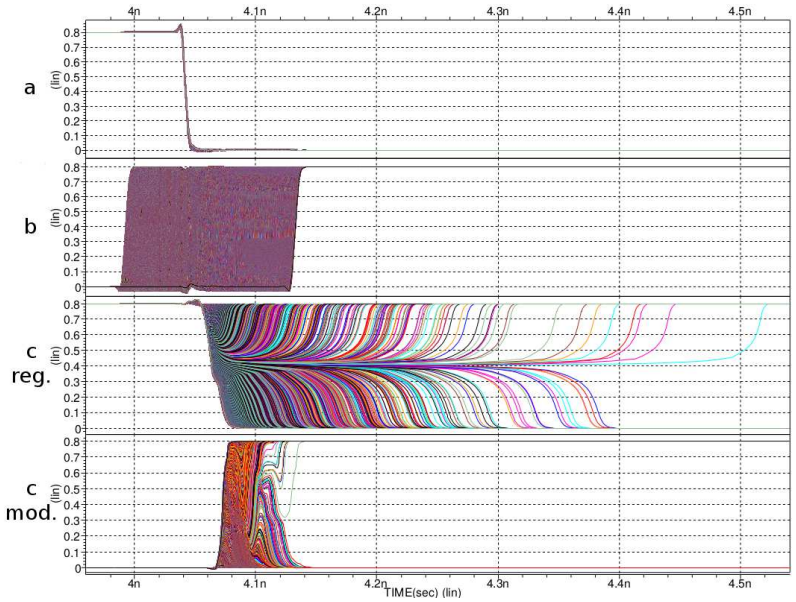

Figure 3.18: Overlay of Monte Carlo Waveforms for Static C-element with c Initially Set to 1

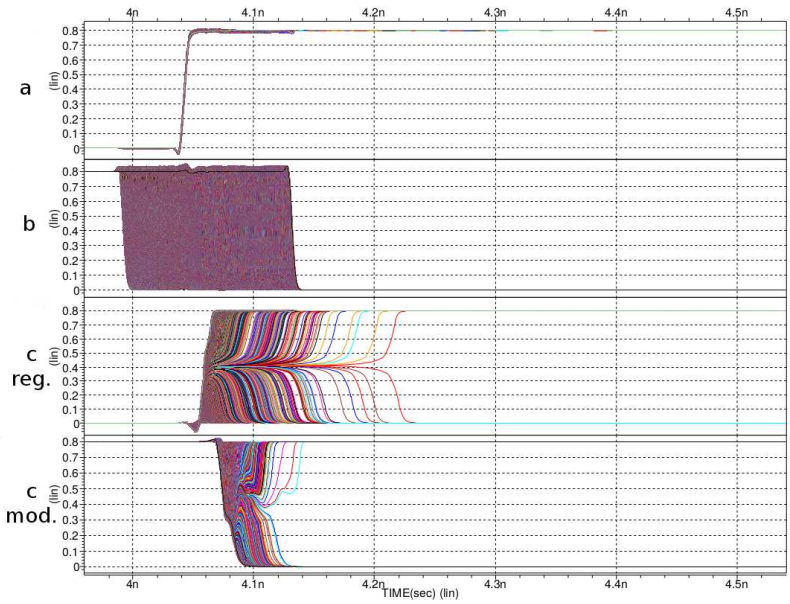

Figure 3.19: Overlay of Monte Carlo Waveforms for Sutherland C-element with c Initially Set to 0

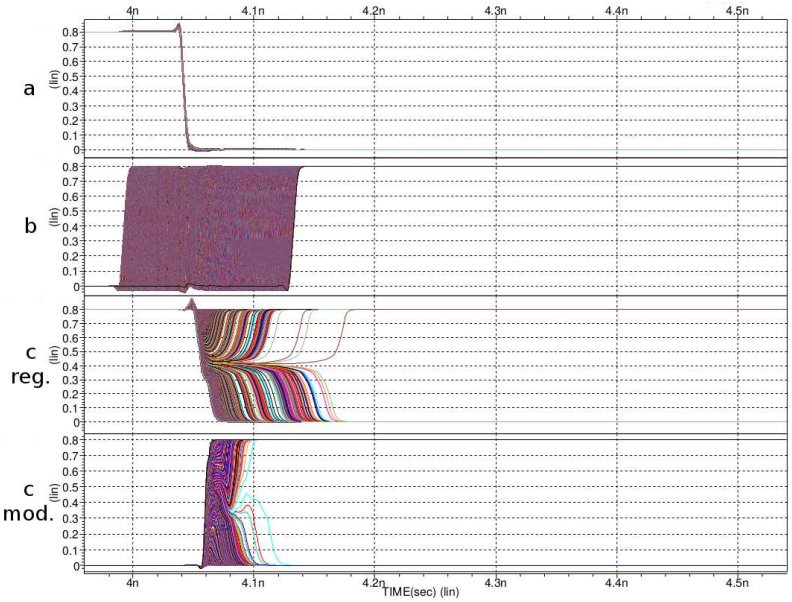

Figure 3.20: Overlay of Monte Carlo Waveforms for Sutherland C-element with c Initially Set to 1

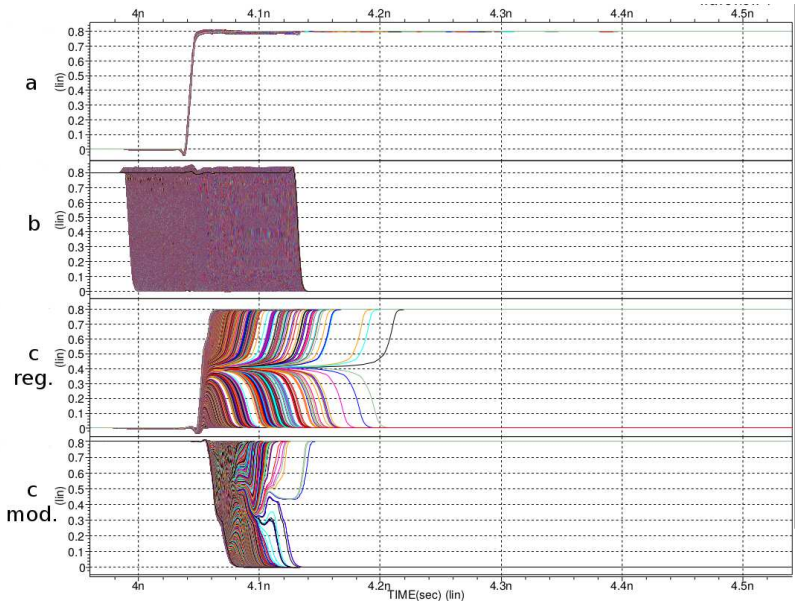

Figure 3.21: Overlay of Monte Carlo Waveforms for Van Berkel C-element with c Initially Set to 0

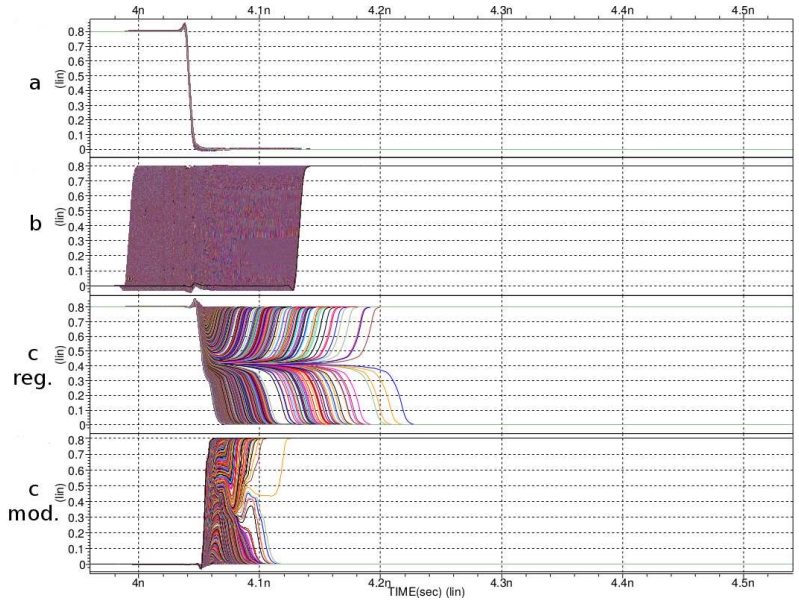

Figure 3.22: Overlay of Monte Carlo Waveforms for Van Berkel C-element with c Initially Set to 0

As can be qualitatively seen from all the figures (Figure 3.17 to 3.22), the modified design of all the C-element implementations shows significantly better metastability performance with process variations taken into account. The output signals in the modified design settle much faster to a rail value as compared to the traditional implementation of the C-element. This shows that the proposed design is robust against process variation effects.

3.6 Conclusion

Asynchronous circuits are preferred for low power and high performance applications. Unlike synchronous circuits, metastability has not been heavily addressed in asynchronous circuit design. In this work, we study the metastability characteristics of the C-element, which is a fundamental building block in asynchronous circuit design. C-elements are used in implementing the control in an asynchronous design. The output of the C-element is prone to metastability when the input values overlap briefly. We compare the metastability response of three popular circuit implementations of the C-element. Additionally, we propose a novel circuit-level design of a robust C-element which drastically reduces the likelihood of a metastable value at the output. Our approach uses independent paths with different switchpoint inverters for rising and falling transitions on the output. Experimental results show that the proposed design achieves a significantly better metastability response as compared to the traditional design of a C-element (with up to 9× improvement in metastability window). With the improved metastability response, the worst case delay for our design is also reduced because the output of our C-element is closer to the rails. The improved metastability response is achieved at the cost of area and power consumption.

3.7 Summary

In this chapter, we compare the metastability performance of three different implementations of C-element (Static, Sutherland and Van Berkel C-element) and propose our metastability reducing latch based scheme to improve the design of all the three C-element implementations being compared. With a nominal overhead of area and power consumption, we drastically reduce the possibility of metastability at the output of the C-element.

4. CONCLUSIONS AND FUTURE WORK

In this thesis, we propose the design of a metastability tolerant latch which uses two independent paths to sample a metastable signal. These independent paths consist of inverters with different switchpoints, such that their response to a metastable signal is different. This ensures that metastability is resolved in a short duration. The use of this special latch based approach is demonstrated in two different circuits; in the design of a synchronizer (which is used at clock domain boundaries in synchronous circuit design) and in a C-element (which is a fundamental building block in asynchronous circuit design). Our experimental simulations demonstrate a significant improvement in the metastability performance of both the circuits mentioned above.

4.1 Future Work on Synchronizer Design

This thesis addresses the issue of synchronization of data between clocks operating at high frequencies. Our proposed design of the synchronizer (with a specialized flip-flop as the first flip-flop of the synchronizer) achieves better metastability performance, providing significantly better setup margin to the second flip-flop of the synchronizer. The use of two specialized flip-flops may be explored in the future, to allow synchronization at a higher frequency than demonstrated in our experiments, with an added area and power overhead.

4.2 Future Work on C-element Design

Metastability issues in asynchronous circuits have also been addressed in this thesis, using the widely used C-element circuit as a reference candidate. Our modified designs for three popular circuit-level implementations of a two input C-element show

notable improvement in the metastability window of the C-element. Future work in this area would be to extend the use of this approach to design multi-input C-elements (with more than two inputs) and achieve similar metastability response.

REFERENCES

[1] Z Al-Tarawneh, G Russell, and A Yakovlev. An analysis of SEU robustness of C-element structures implemented in bulk CMOS and SOI technologies. In *International Conference on Microelectronics*, ICM '10, pages 280–283, Cairo, Egypt, 2010. IEEE.

[2] P Balasubramanian and HR Arabnia. Computation of error resiliency of Muller C-element. In *International Conference on Computational Science and Computational Intelligence*, CSCI '14, pages 179–180, Las Vegas, NV, USA, 2014. IEEE.

[3] S Beer and R Ginosar. A model for supply voltage and temperature variation effects on synchronizer performance. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2461–2472, October 2015.

[4] M Conrad, K Guttag, J Schabowski, D Roskell, J Carey, and B Shore. Synchronizer circuit with dual input. US Patents and Trademarks Office, October 1985. US Patent 4,544,851.

[5] G Couranz and D Wann. Theoretical and experimental behavior of synchronizers operating in the metastable region. *IEEE Transactions on Computers*, 24(6):604–616, June 1975.

[6] W Dally. Synchronizer design. Stanford University, November 1998. http://cva.stanford.edu/books/dig_sys_engr/lectures/l14.pdf.

[7] W Dally and J Poulton. *Digital Systems Engineering*. Cambridge University Press, New York, NY, USA, June 1998.

[8] G Dukes. High speed data synchronizer. US Patents and Trademarks Office, July 1992. US Patent 5,132,990.

[9] G Fuchs, M Fugger, and A Steininger. On the threat of metastability in an asynchronous fault-tolerant clock generation scheme. In *15th IEEE Symposium on Asynchronous Circuits and Systems*, ASYNC '09, pages 127–136, Chapel Hill, NC, USA, 2009. IEEE.

[10] R Garg and SP Khatri. A variation tolerant combinational circuit design approach using parallel gates. In *Analysis and Design of Resilient VLSI Circuits*, pages 153–171. Springer, 2010.

[11] R Ginosar. Fourteen ways to fool your synchronizer. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*, ASYNC '03, pages 89–96, Vancouver, BC, Canada, 2003. IEEE Computer Society.

[12] G Goldrian. Synchronizing logic avoiding metastability. US Patents and Trademarks Office, August 1998. US Patent 5,793,227.

[13] K Guttag and J Carey. Synchronizer circuit. US Patents and Trademarks Office, September 1984. US Patent 4,469,964.

[14] G Keeler. Optical interconnects to silicon CMOS using densely-integrated optoelectronics. *Integrated Optoelectronics: Proceedings of the First International Symposium*, 2002(4):209–236, September 2002.

[15] M Krstić, E Grass, F Gürkaynak, and P Vivet. Globally asynchronous, locally synchronous circuits: Overview and outlook. *IEEE Design & Test of Computers*, 24(5):430–441, September 2007.

[16] Y Li, P Chuang, A Kennings, and M Sachdev. Voltage-boosted synchronizers. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI*, GLSVLSI

'15, pages 307–312, Pittsburgh, Pennsylvania, USA, 2015. ACM.

[17] Y Li, CH Hwang, and HW Cheng. Process-variation-and random-dopants-induced threshold voltage fluctuations in nanoscale planar MOSFET and bulk FinFET devices. *Microelectronic Engineering*, 86(3):277–282, March 2009.

[18] S Lu. Improved design of CMOS multiple-input Muller-C-elements. *Electronics Letters*, 29(19):1680–1682, September 1993.

[19] A Mandal. *Efficient Design and Clocking for a Network-on-Chip*. PhD thesis, Texas A&M University, College Station, TX, USA, May 2013.

[20] JP Murphy. Design of latch-based C-element. *Electronics letters*, 48(19):1190–1191, September 2012.

[21] T Polzer and A Steininger. Metastability characterization for Muller C-elements. In *23rd International Workshop on Power and Timing Modeling, Optimization and Simulation*, PATMOS '13, pages 164–171, Karlsruhe, Germany, 2013. IEEE.

[22] B Sandhu. Synchronizer circuit and method for reducing the occurrence of metastability conditions in digital systems. US Patents and Trademarks Office, April 1996. US Patent 5,510,732.

[23] M Shams, JC Ebergen, and MI Elmasry. A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element. In *International Symposium on Low Power Electronics and Design*, ISLPED '96, pages 93–96, Monterey, CA, 1996. IEEE.

[24] R Sowell and R Pieters. Finite metastable time synchronizer. US Patents and Trademarks Office, April 1989. US Patent 4,820,939.

[25] IE Sutherland. Micropipelines. *Commun. ACM*, 32(6):720–738, June 1989.

[26] Synopsys. HSPICE. http://www.synopsys.com/tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx. Accessed: 10/07/2013.

[27] Arizon State University. Predictive Technology Model. http://ptm.asu.edu. Accessed: 10/21/2013.

[28] H Veendrick. The behaviour of flip-flops used as synchronizers and prediction of their failure rate. *IEEE Journal of Solid-State Circuits*, 15(2):169–176, April 1980.

[29] J Walker and A Cantoni. A new synchronizer design. *IEEE Transactions on Computers*, 45(11):1308–1311, November 1996.

[30] TY Wuu and SBK Vrudhula. A design of a fast and area efficient multi-input Muller C-element. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(2):215–219, June 1993.

[31] W Zhao and Y Cao. Predictive technology model for nano-CMOS design exploration. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3(1):1–17, April 2007.

[32] W Zhao, F Liu, K Agarwal, D Acharyya, SR Nassif, KJ Nowka, and Y Cao. Rigorous extraction of process variations for 65-nm CMOS design. *IEEE Transactions on Semiconductor Manufacturing*, 22(1):196–203, February 2009.