FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Digital Metastability Model for VLSI Circuits

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der technischen Wissenschaften

by

Thomas Polzer
Registration Number 0325077

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Andreas Steininger

The dissertation has been reviewed by:

(Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Andreas Steininger)	(Prof. Alex Yakovlev, PhD, DSc)

Wien, 07.10.2013

(Thomas Polzer)

Erklärung zur Verfassung der Arbeit

Thomas Polzer
Römerstraße 75-77/2.3, 2320 Mannswörth

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 07.10.2013

(Ort, Datum)

(Unterschrift Thomas Polzer)

Acknowledgements

I would like to thank Prof. Andreas Steininger for the opportunity to write this thesis under his supervision and for his continuous support. My thanks go to Prof. Ulrich Schmid for his excellent lecture in distributed algorithms which sparked my interest into scientific work. I am grateful to both of them for providing such an inspiring environment for my work.

My thanks go to my colleagues at the Embedded Computing Systems Group, especially to Jakob Lechner and Robert Najvirt. I really enjoyed working with you and the inspiring discussions we had.

Without the support of my family it would not have been possible to create this work. I am very grateful and want to express my gratitude to them.

This work was partly supported by the Austrian Science Fund (FWF) through the Project FATAL (project number: P21694).

Kurzfassung

Diese Dissertation entwickelt ein digitales Model zur Vorhersage von Fehlerraten die durch das zeitlich unvorteilhafte ansteuern von CMOS Speicherelementen, der sogenannten Metastabilität, hervorgerufen werden. Um ein grundlegendes Model entwickeln zu können werden verschiedene Speicherelemente in einer industriellen 90nm Technologie simuliert. Die dominanten Charakteristika des dadurch entstehenden Antwortverhaltens der Elemente werden aus den so erhaltenen Ergebnissen extrahiert. Diese Resultate werden mittels Messungen an einem FPGA basierenden Prototypen überprüft. Die Messschaltungen die dem Stand der Technik entsprechen können den dafür benötigten Detailierungsgrad nicht liefern und müssen deshalb weiterentwickelt und verbessert werden. Die Hauptvorteile unserer Schaltung sind eine signifikant höhere Zeitauflösung und die Möglichkeit eine zustandsabhängige Analyse durchführen zu können. Basierend auf dieser Fallunterscheidung ist es nun möglich eine Messschaltung für Muller C-Elemente und RS-Latches zu entwickeln. Dies ist notwendig da bestehenden Lösungen die asynchrone Elemente mittels der Erkennung von verzögerten Transitionen messen bestenfalls rudimentär sind.

Um die Gültigkeit des neu entwickelten Metastabilitätsmodells zu überprüfen wird ein Vergleich zwischen einer digitalen und einer analogen Simulation durchgeführt. Als Zielobjekt dient eine Schaltung aus zwei D-Latches. Das Diagramm der so erhaltenen Fehlergraten zeigt nur eine geringe Abweichung zwischen den zwei Simulation. Sie ist wesentlich kleiner als der Einfluss von Temperatur-, Spannungs- und Prozessvarianzen.

Weiters wird analysiert wie kurze transiente Pulse sich in einer elastischen Pipeline fortbewegen können ohne gespeichert zu werden. Solche Pulse können zum Beispiel bei einem Treffer durch ein geladenes Partikel entstehen. Dafür werden zuerst die Grundbausteine der Pipeline (die Muller C-Elemente) mittels analogen Simulationen untersucht. Aus den Ergebnissen wird eine Methode entwickelt um nicht digitale Effekte, die durch diese Pulse entstehen können, im Muller C-Element zu kapseln. Basierend auf diesen Erkenntnissen wird die gesamte Pipeline simuliert. Dabei werden die Ausgangstreiber der einzelnen Stufen variiert. Die Ergebnisse weisen auf einen starken Zusammenhang zwischen der vorhandenen Ausgangsstufe und der Eigenschaft umgespeicherte Pulse weiterzuleiten hin.

Abstract

This thesis develops a digital model for predicting failure rates caused by marginal triggering, so called metastability, of CMOS storage elements. To derive the underlying model, various storage elements are simulated in an industrial 90nm technology. The main characteristics of the responses of those elements are extracted from the results. The simulation findings are verified in hardware using measurements on an FPGA prototype. To achieve the required level of detail, the state of the art measurement circuits are not sufficient and are therefore extended. The main novelty for measuring D-flip flops is the possibility to perform a state-dependent response analysis and a significantly increased temporal resolution. Based on the case separation technique of the late transition detector for D-flip flops, a measurement infrastructure for Muller C-elements and RS-latches is developed as solutions for measuring asynchronous components using late transition detection were very basic before.

To verify the functionality of our newly developed metastability model, a comparison between a digital and an analogue simulation of a circuit comprising two D-latches is performed and the resulting failure rate plots demonstrate that the differences between the simulations are much smaller than the deviation caused by temperature, voltage and process variations.

Additionally an analysis on the propagation of short transient pulses in elastic pipelines, as caused by e.g. ionized particle hits, is performed. Therefore the constituting Muller C-elements are subjected to analogue simulation first and a method for containing non-digital output values within the element is derived. Based on those results, the elastic pipeline is simulated using different output stages for the Muller C-elements. The results indicate that the property of containing the propagation of unlatched pulses within the pipeline heavily depends on the used output stage.

Contents

1 Introduction **1**

2 Background **5**
2.1 Timing Models . . . 5
2.1.1 Synchronous Circuits . . . 5
2.1.2 Asynchronous Circuits . . . 6
2.1.3 Globally Asynchronous Locally Synchronous (GALS) Circuits . . . 7
2.2 Basic Storage Elements . . . 7
2.2.1 D-Latch . . . 7
2.2.2 D-Flip Flop . . . 9
2.2.3 Muller C-element . . . 10
2.2.4 RS-Latch . . . 11
2.3 Timing Violations . . . 14
2.3.1 Storage Element Model . . . 14
2.3.2 Setup- and Hold-Time . . . 16
2.3.3 Failure Estimation . . . 17
2.3.4 Metastability Mitigation . . . 18

3 Methodology and Contributions **21**

4 Metastability Simulation **25**
4.1 State of the Art . . . 25
4.2 Simulation Algorithm . . . 26
4.3 Basic Elements . . . 27
4.3.1 Metastability Behavior of D-Latches . . . 27
4.3.2 Metastability Behavior of Mutexes . . . 29
4.3.3 Metastability Behavior of Muller C-Elements . . . 29
4.4 Metastability Containment for Muller C-Elements . . . 31
4.4.1 Low- and High-Threshold Inverters . . . 33
4.4.2 Schmitt-Trigger . . . 33
4.5 Summary . . . 36

5 Metastability Propagation in Micropipelines **37**
5.1 State of the Art . . . 37
5.2 Spice Simulation Setup . . . 39
5.3 Simulation Results . . . 41
5.4 Evaluation . . . 43
5.5 Summary . . . 48

6 Metastability Measurement **49**
6.1 State of the Art . . . 49
6.1.1 Metastability Generation . . . 49
6.1.2 Metastability Detection . . . 50
6.1.3 Metastability Detection in FPGAs . . . 51
6.2 Late Transition Detection for Flip Flops . . . 52
6.2.1 Overall Design Concept . . . 52
6.2.2 Measurement Circuit . . . 55
6.2.3 Validation Results . . . 58
6.2.4 Constraints . . . 62
6.3 Late Transition Detector for Muller C-Elements . . . 64
6.3.1 Initial Circuit Design . . . 64
6.3.2 Prototype Implementation . . . 69
6.3.3 Measurement Procedure . . . 72
6.3.4 Preliminary Results . . . 72
6.3.5 Circuit Refinement . . . 75
6.3.6 Refined Measurement Results . . . 78
6.4 Late Transition Detection for RS-Latches and Mutexes . . . 81
6.4.1 Circuit Design . . . 81
6.4.2 Prototype Implementation . . . 84
6.4.3 Results . . . 85
6.5 Summary . . . 85

7 Merging Results from Simulation and Measurement **89**
7.1 Simulation Model . . . 89
7.2 Simulation Algorithm . . . 90
7.3 Observed Phenomena . . . 91
7.3.1 Late Transitions . . . 91
7.3.2 Pulses . . . 92
7.4 LTD Measurement Result Matching . . . 92
7.4.1 D-Latch . . . 92
7.4.2 D-Flip Flop . . . 95
7.4.3 Remarks . . . 96
7.5 Summary . . . 98

8 Digital Metastability Simulation **99**
8.1 Digital Metastability Simulation Model . . . 100
8.2 Simulation Algorithm and Model Refinement . . . 102
8.3 Case Study . . . 103
8.4 Mathematical Model . . . 108
8.4.1 Delay model . . . 108
8.4.2 Pulse Length Model . . . 112
8.4.3 Enable Delay Model . . . 113
8.4.4 Pulse Propagation Model . . . 113
8.4.5 Example . . . 114
8.4.6 Simulation Using the Mathematical Model . . . 122
8.5 Summary and Limitations . . . 124

9 Conclusion and Future Work **127**

A Additional Simulation Results **129**
A.1 van-Berkel Muller C-Element . . . 129
A.2 Conventional Muller C-Element . . . 140
A.3 Weak Feedback Muller C-Element . . . 150

B Digital Simulation Model - Slave Latch Fitting **161**

List of Figures

2.1 Timing models 6
2.2 D-latch 8
2.3 D-flip flop 9
2.4 Muller C-element 12
2.5 Elastic pipeline implementation circuit 13
2.6 NAND based RS-latch 13
2.7 Storage loop model and visualization of timing-violations (D-latch) 15
2.8 Storage element model for RS-latches 16
2.9 Failure rate plot 19

4.1 Simulation result for a D-latch 28
4.2 Simulation result for a mutex element 30
4.3 Metastability conditions of a Muller C-element 31
4.4 Simulation result for a van Berkel Muller C-element 32
4.5 Simulation result for the low threshold inverter 34
4.6 Simulation result for the Schmitt-Trigger output stage 35

5.1 Driver circuit and SET model 40
5.2 SET propagation in a micropipeline (latching/extinction of SET in third stage, van-Berkel implementation, $V_{th} = 0.355V$) 42
5.3 Micropipeline stage in empty state 43
5.4 Number of stages seeing SET propagation 45
5.5 Charge window for SET propagation (1-0-1 pulses) 45
5.6 No SET propagation with Schmitt-Trigger output stage 47

6.1 Basic metastability detection circuit concept 49
6.2 Digital metastability detection circuit 51
6.3 Metastability and clock generation circuit 56
6.4 Adapted LTD circuit 57
6.5 Measurement result for rising edges with different clock pulse widths 59
6.6 Measurement result for falling edges with different clock pulse widths 60

6.7 Measurement result for $1 \rightarrow 0 \rightarrow 1$ pulses with different clock pulse widths . . 60
6.8 Measurement result for cases starting from 1 61
6.9 LTD circuit for Muller C-elements 66
6.10 Timing diagram of LTD operation 67
6.11 Effect of masking non-critical overlaps, schematic depiction 69
6.12 Detector selection 70
6.13 Muller C-element FPGA implementation 71
6.14 Measurement result of first implementation 73
6.15 Measurement result of second implementation 74
6.16 Input dependence of τ 76
6.17 Problematic cases 76
6.18 Refined measurement circuit for one reference signal 78
6.19 Repeated measurement result of first implementation 79
6.20 Repeated Measurement result of second implementation 80
6.21 NAND-based RS-latch with a dedicated output driver 81
6.22 Input conditions leading to metastability 82
6.23 Late transition detection circuit for RS-latches 83
6.24 RS latch FPGA implementations 84
6.25 Measurement result 86
6.26 Measurement result 87

7.1 Simulation model 90
7.2 LTD emulation in the simulation 91
7.3 Metastable response of a D-latch (matched output inverter threshold) 93
7.4 Metastable response of a D-latch (high output inverter threshold) 94
7.5 Correspondence of metastability time and output delay 95
7.6 Correspondence of metastability time and pulse length 96
7.7 Correspondence of metastability time and LTD results for a D-flip flop 97

8.1 Comparison of the simulation models 101
8.2 Basic simulation concepts 102
8.3 Additional characteristics of the D-latch (90nm bulk CMOS) 104
8.4 Digital metastability simulation case study 105
8.5 Digital metastability simulation case study results (1) 106
8.6 Digital metastability simulation case study results (2) 107
8.7 Failure rate vs. overlap 108
8.8 Failure rate vs. overlap for different PVT corners 109
8.9 Storage loop model 109
8.10 Models for input voltage slope θ 111
8.11 Plot of simulated pulse lengths for one of the sample latches 113
8.12 Model-fitting master latch, rising edges (1) 114
8.13 Model-fitting master latch, rising edges (2) 115
8.14 Model-fitting master latch, falling edges (1) 116
8.15 Model-fitting master latch, falling edges (2) 117

8.16 Model-fitting master latch, falling edges (pulse model) 119
8.17 Model-fitting master latch, rising edges (enable delay) 120
8.18 Model-fitting master latch, falling edges (enable delay) 121
8.19 Model-fitting master latch, rising edges (pulse propagation) 123
8.20 Model-fitting master latch, falling edges (pulse propagation) 123
8.21 Comparison of failure rate vs. overlap plot between analogue simulation and mathematical model based simulation 124
8.22 Failure rate vs. overlap for different PVT corners (mathematical model) 125

A.1 Results of the van-Berkel implementation with a matched output inverter (1) . . 129
A.2 Results of the van-Berkel implementation with a low threshold output inverter (1) 130
A.3 Results of the van-Berkel implementation with a high threshold output inverter (1) 130
A.4 Results of the van-Berkel implementation with a Schmitt-trigger output (1) . . 131
A.5 Results of the van-Berkel implementation with a matched output inverter (2) . . 132
A.6 Results of the van-Berkel implementation with a matched output inverter (3) . . 133
A.7 Results of the van-Berkel implementation with a low threshold output inverter (2) 134
A.8 Results of the van-Berkel implementation with a low threshold output inverter (3) 135
A.9 Results of the van-Berkel implementation with a high threshold output inverter (2) 136
A.10 Results of the van-Berkel implementation with a high threshold output inverter (3) 137
A.11 Results of the van-Berkel implementation with a Schmitt-trigger output (2) . . 138
A.12 Results of the van-Berkel implementation with a Schmitt-trigger output (3) . . 139
A.13 Results of the conventional implementation with a matched output inverter (1) . 140
A.14 Results of the conventional implementation with a low threshold output inverter (1) 140
A.15 Results of the conventional implementation with a high threshold output inverter (1) 141
A.16 Results of the conventional implementation with a Schmitt-trigger output (1) . . 141
A.17 Results of the conventional implementation with a matched output inverter (2) . 142
A.18 Results of the conventional implementation with a matched output inverter (3) . 143
A.19 Results of the conventional implementation with a low threshold output inverter (2) 144
A.20 Results of the conventional implementation with a low threshold output inverter (3) 145
A.21 Results of the conventional implementation with a high threshold output inverter (2) 146
A.22 Results of the conventional implementation with a high threshold output inverter (3) 147
A.23 Results of the conventional implementation with a Schmitt-trigger output (2) . . 148
A.24 Results of the conventional implementation with a Schmitt-trigger output (3) . . 149
A.25 Results of the weak feedback implementation with a matched output inverter (1) 150
A.26 Results of the weak feedback implementation with a low threshold output inverter (1) 150

A.27 Results of the weak feedback implementation with a high threshold output inverter (1) . . . 151
A.28 Results of the weak feedback implementation with a Schmitt-trigger output (1) 151
A.29 Results of the weak feedback implementation with a matched output inverter (2) 152
A.30 Results of the weak feedback implementation with a matched output inverter (3) 153
A.31 Results of the weak feedback implementation with a low threshold output inverter (2) . . . 154
A.32 Results of the weak feedback implementation with a low threshold output inverter (3) . . . 155
A.33 Results of the weak feedback implementation with a high threshold output inverter (2) . . . 156
A.34 Results of the weak feedback implementation with a high threshold output inverter (3) . . . 157
A.35 Results of the weak feedback implementation with a Schmitt-trigger output (2) 158
A.36 Results of the weak feedback implementation with a Schmitt-trigger output (3) 159

B.1 Model-fitting slave latch, rising edges . . . 163
B.2 Model-fitting slave latch, rising edges (pulse model) . . . 164
B.3 Model-fitting slave latch, falling edges . . . 165
B.4 Model-fitting slave latch, rising edges (enable delay) . . . 166
B.5 Model-fitting slave latch, falling edges (enable delay) . . . 167
B.6 Model-fitting slave latch, rising edges (pulse propagation) . . . 168
B.7 Model-fitting slave latch, falling edges (pulse propagation) . . . 168

List of Tables

4.1 Comparison of the metastability mitigation implementations 33

6.1 Metastable cases that can be distinguished 54

8.1 Delay model parameters for master latch 118
8.2 Pulse length model parameters for master latch 118
8.3 Enable model parameter for master latch 122
8.4 Pulse propagation model parameters for the transparent master latch 122

B.1 Delay model parameters for slave latch (rising edges) 161
B.2 Pulse length model parameters for slave latch 161
B.3 Delay model parameters for slave latch (falling edges) 162
B.4 Enable model parameter for slave latch 162
B.5 Pulse propagation model parameters for the transparent slave latch 162

1 Introduction

With each new technology cycle, very large scale integrated (VLSI) circuits are becoming more and more complex. Their integration density increases exponentially, following Moore's law which states that the number of components per chip is doubled every two years [Moo75]. To be able to accommodate such an increasing number of transistors on the same area, their size must shrink accordingly. As the channel length (feature size) of the transistor is becoming smaller, the time electrons need to traverse the channel is getting shorter and therefore the speed of the device is increased. Additionally, the smaller feature sizes also have an impact on the internal capacitances of the devices (which shrink quadratically with the feature size [DW11]), further speeding up their switching operation.

Timing closure To control the interaction within the circuit and to coordinate the data processing, a rigorous timing closure must be established. The most prominent schemes are:

- Synchronous circuits
- Asynchronous circuits
- GALS systems

Historically the fully **synchronous** [FH90, Sei79] variant is used in digital VLSI chips. Due to its high abstraction properties, the design of synchronous chips is largely simplified. In current technology nodes, however, more and more problems arise from the assumptions required to implement it.

As devices are becoming faster, the maximum achievable clock frequency is also increased. The time required for distributing data signals on the chip is not decreasing and therefore the gap between the required and the available time becomes larger. Therefore the signal propagation time limits the achievable clock frequency and it is no longer possible to operate synchronous circuits with the theoretically achievable maximum clock frequency. Additionally the distribution of the clock signal itself is no longer straight forward but has become quite involved[Fri01] as special topologies (as e.g. balanced clock trees) become mandatory.

Apart from the problem of data- and clock-signal distribution, parameter variations have a high impact on the performance of modern VLSI circuits. Today CMOS (complementary metal oxide semiconductor) design has nearly reached the limits of physics, as the dimensions of e.g. the oxide layer in the MOSFETs (metal oxide semiconductor field effect transistors) have shrunk to only a few atom layers [TBC+97]. Devices with such small dimensions are subjected to quantum effects, like tunneling currents [YHHW99]. The variability of the devices is significantly increased as small variations in the process may have a huge impact on the resulting device characteristics. Therefore the differences between best-case and worst-case timings become larger [BDM02]. To guarantee the circuit operation under all expected conditions, worst case assumptions have to be used. Therefore the most adverse conditions (concerning process tolerances as well as voltage- and temperature-variations, also shortly called PVT-variations) are assumed when designing the circuit. Therefore the circuit will operate correctly within the whole operation range at the cost of a high performance impact for the average case. This waste of performance has become a major problem.

The cause of this behavior is that the timing closure is achieved in an open loop fashion. The clock signal is calculated at design time and is not adapted to the current operation conditions. To circumvent this shortcoming, a timing closure scheme using a closed loop control was developed. Circuits built based on these scheme are called **asynchronous** [Hau95, FB96]. This protocol removes the assumption of the forward delay required by the synchronous approach by acknowledging the reception of data explicitly. Furthermore no global clock signal is used but the availability of data is signaled locally. In contrast to the synchronous paradigm, where the longest path on the whole chip determines the overall clock speed, the local handshaking only depends on the delays of the corresponding link. Additionally, as asynchronous circuits make no assumptions on the delays, the speed is automatically adapted to the current operation conditions. The system will work at full speed under best case operation conditions and slowing down enough to accommodate for the increased delays under worst case conditions, which makes asynchronous circuits increasingly attractive.

Unfortunately the design of asynchronous circuits is much more involved than the design of synchronous circuits. Furthermore the tool support is not as mature as for synchronous circuits. Therefore, in industrial designs, they are only used for specialized circuits, where, e.g., very high speed or extremely low power consumption is required.

To combine the advantages of the synchronous and asynchronous schemes without inheriting all their disadvantages, a third, basic model for timing closure was developed. In this model the circuit is split into small modules which are designed synchronously. Therefore the available powerful toolsets can be used. Between the modules the communication is executed in an asynchronous fashion. Such systems are called globally asynchronous, locally synchronous (**GALS**) systems [Cha84].

Metastability As GALS modules are very compact, the time required for signal distribution is normally quite small and can be handled efficiently by the synchronous tools. As, however, multiple independent clock sources are used, clock domain crossings arise at the module boundaries and the timing assumptions required for a safe operation of the circuit break down at those interfaces. This may lead to out of spec behavior called metastable upsets [Gin11]. They manifest

themselves as non-digital signal values for extended times, delayed transitions or pulses at the boundary elements of the modules. To circumvent adverse effects within the circuit, their rate of occurrence must be kept low. This is achieved by specialized circuits, so called synchronizers [Kin07]. They must be designed rigorously to get a acceptable trade-off between failure rate and circuit performance. Furthermore a high number of pitfalls exist [Gin03, Kin07], voiding their benefits or even increasing the overall failure rate of the system. This fact makes it apparent that the design of synchronization circuits is of topmost importance in modern systems.

At a first glance, the problem of metastability seems to be only relevant for GALS circuits with their multiple clock domains. Unfortunately this problem also arises for fully synchronous systems when handling external data. Even in asynchronous circuits external data may cause metastability, as its source is not necessarily part of the system's timing closure and therefore does not adhere to the handshake protocol. The circuit's boundary elements may therefore experience metastable upsets as well.

External faults Another source of metastability of digital circuits may be external faults. As such faults will definitively not adhere to the underlying timing closure (independent of its type), upsets can not be ruled out. An important fault mechanism are single event transients (SETs) which may be caused by hits of ionized particles. Unfortunately shrinking feature sizes lead to shrinking internal charges which increases the susceptibility of the circuits to external faults. As the effects of SETs heavily depend on the internal charge of the circuit nodes (called critical charge in this case) [DW11], the energy required from the hitting particles to upset the circuit also decreases. The number of critical particles, however, increases with falling energy [DW11], which has a direct impact on the predicted failure rates. As these transients manifest themselves as short pulses, the probability of metastable upsets caused by SETs also increases and since they are uncorrelated to the underlying timing assumptions metastable upsets may occur.

Metastability effects The effects of metastability are diverse and often hard to debug. As, e.g., data may be inconsistently latched [Kin07] into the circuit (which is outside of any specification) and the occurrence of these failures may be at arbitrary times and non reproducible, the debugging of the resulting errors becomes very challenging. Therefore it is very important to sufficiently prevent their occurrences in the first place.

Unfortunately metastability is able to overcome conventional error containment boundaries. Therefore all reasoning and all proofs on the correctness of a circuit may become void in its presence. The root for this behavior is that typically proofs are based on the specified functional description of logic gates. Metastability on the other hand causes out-of-specification operation of these gates and is hence normally ignored in these descriptions. Therefore it is very important to understand how metastability affects different kinds of circuit elements. This knowledge is the foundation for devising a functioning containment mechanism.

Mean time between failure To be able to calculate the expected mean time between failure (MTBF), a mathematical model has been created [Vee80]. Using this model it is possible to predict the MTBF based on the circuit characteristics (like clock- and data-frequencies as well as component parameters). This model, however, has several issues for practical circuits:

- The model expects that the component parameters are precisely known. In practice, however, these parameters often may only be roughly estimated, as no datasheet values exist.
- PVT variation have a major impact on the performance in the metastable regime. Therefore the failure rate may either be calculated for worst case conditions, leading to a vast overhead in nearly all practical cases, or for the typical case. If the latter is true, the failure rate may, however, be too high for adverse operation conditions.
- In most state of the art circuits the different clock domains are not completely independent but derived from a common reference clock using, e.g., phase locked loops (PLLs). In this case the mathematical model is not valid and may predict much too low failure rates [BG13].

Based on these calculations, the minimum number of required synchronizer stages can be estimated. Unfortunately, many designers work with some rules of thumb (like use two synchronization stages for normal circuits and three for high dependability ones). These rules may apply to many situations but may not be used as a general design methodology. Dependent on the boundary conditions, the resulting MTBF may be too small (increased error rate) or too high (unnecessary circuit latency).

As it has been shown in theory that metastability is unavoidable [KC87b, Mar81] in the first place and the fact that it may affect virtually all electronic circuits manufactured today, makes analyzing its effects and devising countermeasures, such as local containment and mitigation, a vital requirement.

With this motivation this thesis is dedicated to analyze the metastability behavior of selected, basic building block used in constructing synchronous and asynchronous circuits by means of simulations and measurements. The goal is to develop a purely digital simulation model which is able to predict their response to marginal triggering.

2 Background

To start with we want to outline some of the basic concepts required by this thesis, which are different timing models for VLSI circuits, the most commonly used VLSI storage elements and an overview on metastability.

2.1 Timing Models

The correct functioning of sequential digital circuits relies on a rigorous control of the underlying storage element. This coordinating of the information flow between the elements is necessary to achieve a well defined circuit behavior and is also called the timing closure of the system. In the following we will introduce tree different methods for achieving timing closure [Pol09].

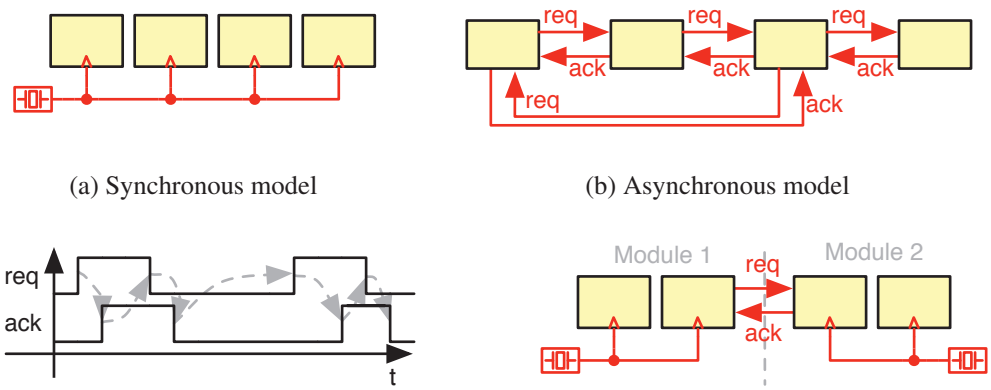
2.1.1 Synchronous Circuits

Most industrially designed and produces circuits today are built using the synchronous design paradigm. It is in use for several decades now and therefore very mature and well supported by design tools.

The timing closure in synchronous circuits is achieved using a single, central clock signal[FH90, Sei79] (see Figure 2.1a). Each storage element in the system is controlled by this clock and therefore, in theory, all storage operations are executed in perfect lockstep.

To facilitate a reliable operation of the system in practice, the frequency of the clock signal must be calculated rigorously [BC09]. Using a so called static timing analysis [BC09], the maximum path delay between any two storage elements is calculated for the worst case operation condition. This results in the longest delay possible within the circuit and can therefore be used to determine the maximum safe clock frequency.

As appealing as the presence of an ideal global clock is, as problematic the system analysis can get. Increasing clock- and signal frequencies [Con03], therefore decreasing timing safety margins, and increasing gate count tend to make the static timing analysis more and more challenging. Thanks to the high degree of automation in the design process and the very sophisticated tool support, these problems can still be handled.



(a) Synchronous model

(b) Asynchronous model

(c) Asynchronous handshake protocol

(d) GALS model

Figure 2.1: Timing models

A highly problematic and challenging task, especially in high performance systems, is the design of the clock network. To keep the synchronous abstraction valid, the differences in the delays from the clock source to the different storage elements (the skew within the clock network) must be rigorously controlled [Fri01]. This is especially true for large clock networks. The usage of special topologies, like e.g. forks and trees, nevertheless lead to acceptable results, but render clock routing an art of its own.

2.1.2 Asynchronous Circuits

A completely different approach are asynchronous circuits [MB59, Huf57, Sei79]. Instead of using a centralized coordination, timing closure is achieved by coordinating the storage elements locally (by a so called handshake protocol). The source uses a request (req) signal (either an explicit signal line or implicitly encoded into the data) to advertise the availability of new data. The data is kept constant until the receiver has confirmed their successful reception using an explicit acknowledge (ack) signal. Figure 2.1b shows an example of an asynchronous circuit using an explicit request and acknowledge signal, while Figure 2.1c depicts the principle of a handshake protocol. Each rising edge of the request signal (req) must be confirmed by a rising edge on the acknowledge signal (ack) and each falling edge on request with a falling edge on acknowledge, respectively. The alternation of req and ack events defined by the handshake protocol governs a cyclic sequence of operation for the circuit. It can be safely implemented in a closed environment which then forms an independent timing domain.

Due to the absence of any timing assumptions (recall that, in contrast, synchronous circuits make extensive use of worst case assumption) conceptually very appealing, it suffers on the lack of tool support and is therefore commercially scarcely used. Furthermore delay insensitive solutions introduce a significant implementation overhead (like null convention logic (NCL) [FB96]) and/or a timing overhead (a handshake cycles requires twice the time as a push-forward solution). Nevertheless there are already working industrial implementation examples available

(e.g. [SFGP09]).

2.1.3 Globally Asynchronous Locally Synchronous (GALS) Circuits

An approach to mitigate the restrictions of the synchronous paradigm is the globally asynchronous locally synchronous (GALS) model [Cha84]. The system is split into multiple modules and the timing closure within these modules is achieved by using the synchronous paradigm. Therefore each module has its own, local clock source. These clock sources may be loosely coordinated [DY07], but do not have to be. The timing closure of the inter-module communication is, however, achieved using the asynchronous approach (see Figure 2.1d).

An advantage of this scheme is that each module can be implemented as synchronous circuit utilizing the existing powerful toolsets. Due to the small size of the modules, their design and analysis is much easier and faster than the analysis of a large fully synchronous system. A big drawback is the limited communication speed between the modules. Due to the lack of synchronization flow control is needed to enable a secure data transfer between the modules. This limits the communication throughput [TGL07] significantly. Furthermore, in case of uncoordinated local oscillators, no global timing information is available. To circumvent these drawbacks, several approaches are based on loosely synchronized clock sources for the different modules [TGL07].

2.2 Basic Storage Elements

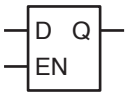
Nearly all circuits designs today are sequential, meaning they have an internal state. To be able to persist this state, dedicated storage elements are required. In this section we will introduce the most important storage element implementations.

2.2.1 D-Latch

The D-latch is a very simple and straight forward implementation of a storage element. Its input stack consists only of a transmission gate, and the the storage loop is built from two cross coupled inverters [Kae08]. The D-latch circuit symbol and its truth table are shown in Figure 2.2a and Table 2.2b, respectively, while Figure 2.2c shows one possible implementation.

If the input stack is transparent ($EN = 1$), the transistors $M1$ and $M2$ of the transmission gate actively drive the current value of D into the storage loop (node z). The forward inverter ($M3$, $M4$) is used to generate the signal on the inverted loop node zb and the output inverter ($M5$, $M6$) drives the output Q. The loopback inverter ($M7$, $M8$, $M9$ and $M10$) is switched off in this operation mode and does not participate in the operation.

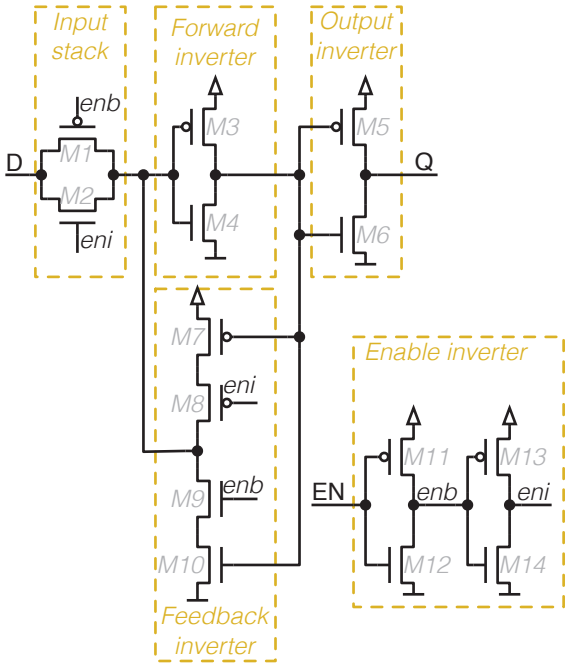
When entering the storage (or opaque) mode ($EN = 0$), the input transmission gate is disabled and therefore the input D is decoupled from the storage loop. At the same time, the loopback inverter is switched on and retains the current value of node z, as its input is fixed to the opposite value (node zb). The nodes z and zb form the storage loop required for retaining the state of the D-latch while it it is opaque. Figure 2.2d illustrates the functionality of the D-latch. It can be seen that as long as the latch is transparent, each input change is directly propagated to its output (indicated by the gray arrows). If, on the other hand, the latch is in the opaque



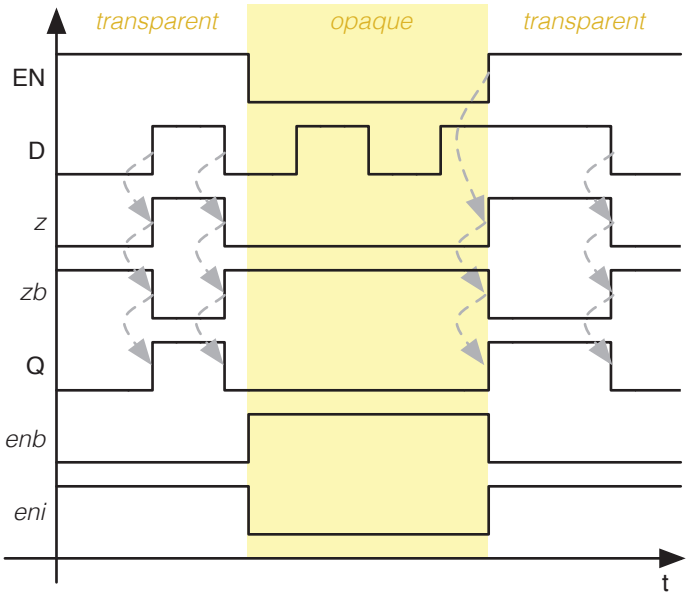
(a) Symbol

D	EN	Q_{n+1}
x	0	Q_n
0	1	0
1	1	1

(b) Truth table

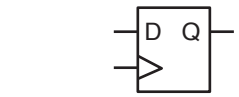


(c) Implementation circuit



(d) D-latch idealized timing

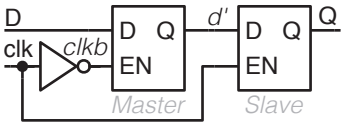
Figure 2.2: D-latch



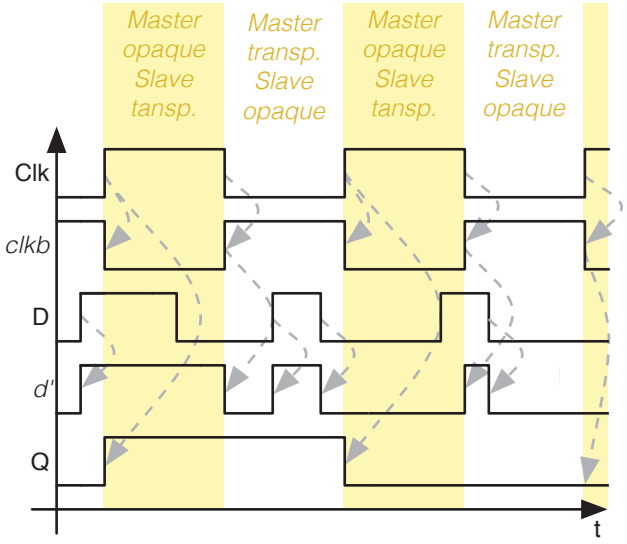
(a) Symbol

D	Clk	Q_{n+1}
x	0	Q_n
x	1	Q_n
x	$\downarrow$	Q_n
0	$\uparrow$	0
1	$\uparrow$	1

(b) Truth table



(c) Implementation circuit



(d) Idealized timing

Figure 2.3: D-flip flop

mode, the input changes are no longer mimicked in the storage loop and therefore also not at the output. When the latch changes to the transparent state again and if the input is different from the internal state, a state change is triggered by the transition of the EN signal, and the output value is flipped accordingly.

The two additional inverters ($M11$, $M12$ and $M13$, $M14$, respectively) are used to generate the signals required for controlling the input transmission gate and the loopback inverter.

There are multiple other ways to implement D-latches (e.g. based on RS-latches [Kae08]). A commonly used optimization moves the storage loop out of the critical path (see [JYG09] for a flip flop circuit using such a latch topology). Therefore the data signal can directly propagate from the input to the output, while the storage loop inverters can be sized minimally. Using such a topology can increase the speed of the latch while, however, the storage loop characteristics may be somewhat degraded. If, on the other hand, the characteristic of the storage loop is important, specialized latch topologies exist (like [ZKRY06], e.g.).

2.2.2 D-Flip Flop

The D-flip flop is the simplest edge triggered storage element [Kae08] and is therefore also the most important element for building synchronous VLSI design. The input signal is only captured, when an active edge (in most cases a rising edge) occurs at the Clk input. Figure 2.3 shows the schematic symbol and the truth table of the D-flip flop.

While there are multiple different possibilities for implementing the D-flip flop, the basic D-flip flop implementation is a chain of two D-latches [Kae08, AKY09] (see Figure 2.3c). The enable signals of the two latches are connected in a way that, if the first latch is transparent the second one is opaque and vice versa. This enables the circuit to only capture a new value at the active edge of the clock signal (when the master is switching from transparent to opaque, the rising edge for the one depicted in the figure). Therefore the primary responsibility to capture a new value lies with the first latch (therefore called the master latch), while the second latch is only required to hold the old value while the master latch is transparent (called slave latch). This implementation version is called the master-slave D-flip flop.

The basic operation of the circuit is shown in Figure 2.3d. It can be seen that, as long as the master is transparent, the input changes are directly copied to the intermediate signal d'. As the slave is, however, opaque in this operation mode, the change is not visible at the output. If an active clock edge occurs, the master switches to opaque, freezing the current value of d' by decoupling it from the input. The slave latch becomes transparent and forwards the frozen value of d' to the output Q. Please note that for the circuit to function correctly, the inertia of forwarding a value by the latch must be higher than the time to switch from transparent to opaque. Otherwise the decoupling of d' would not be fast enough and an undesired value may propagate from the input directly to the output. The timing diagram also shows that there is no direct combinational path from the input to the output. One of the latches is always opaque and therefore input changes are only captured at the active (rising in the example) clock edge.

Most data path flip flops use this implementation variant. As the slave latch is only used for buffering the result it is often built different from the master latch (smaller transistors, changed topology) to increase the overall speed of the flip flop [JYG09]. For specialized functions (like synchronization, e.g.), other flip flop implementations are possible [YJG11, AKY09]. When using an FPGA to implement the circuit these optimized versions are, however, not available.

2.2.3 Muller C-element

Muller C-elements [Sei79] are the basic building blocks of the control paths in most asynchronous circuits [SF01, Sut89]. Basically a Muller C-element is a conjunction for signal transitions: Its output will only go high, if a rising edge was detected on both of its inputs and will only go low, if a falling edge has occurred on both. The symbol for the Muller C-element and its truth table are shown in Figure 2.4a and Table 2.4b, respectively. Note that in case of non-matching inputs the previous output is retained, which causes the need for storage capability.

The basic operation of a Muller C-element is shown in Figure 2.4f. If both input signals are the same, the output is set to the corresponding value, if they differ, the Muller C-element enters the storage mode, keeping the inner state and therefore the output unchanged. A very interesting fact is that in this mode, any transition at the input will be immediately captured and stored. The Muller C-element in storage mode displays a high eagerness to capture new input transitions.

There are several ways of implementing a Muller C-elements in CMOS. The three most important variants are [SEE96]:

- Conventional Implementation

- Weak Feedback Implementation
- Van-Berkel Implementation

In the *weak feedback-implementation* (according to Martin, see Figure 2.4c) an input stage made up by a p-stack ($M1$, $M2$) and an n-stack ($M3$, $M4$) is responsible for switching the inner node zb to high or low, respectively, when the inputs A and B match. As the stacks are not complementary, they will not actively drive zb for non-matching inputs. In this situation a "keeper element" formed by a loop of two inverters ($M5$, $M6$ and $M9$, $M10$) will retain the current state of zb (semi-static implementation). The feedback inverter ($M9$, $M10$) must be weak and is (temporarily) overridden by the input stage when matching inputs force the circuit into the other logic state. This will finally flip the keeper stage into the appropriate state as well. Normally the forward inverter ($M5$, $M6$) does not directly drive the output, but some kind of output buffering is provided ($M7$, $M8$).

The *conventional implementation* (according to Sutherland, see Figure 2.4d) proposed by Sutherland basically works in the same way, but transistors $M11$-$M14$ disconnect the feedback inverter in the keeper element from the respective supply rail when the input stage is active, thus avoiding the transient driver conflict of the above solution (static implementation).

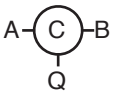
The *van-Berkel implementation* (see Figure 2.4e) consists of an input stage formed by $M1$-$M4$ (p-stack) and $M6$-$M9$ (n-stack), the output inverter ($M13$, $M14$) and a loop-back path formed by the inverter ($M11$, $M12$) and transistors ($M5$, $M10$). Again the input stack is responsible for changing the state in case of matching inputs. If the two inputs are different, one feedback transistor ($M5$ if z is low or $M10$ is z is high) is conductive. Together with the conducting transistors of its associated stack it forms a path to VDD or GND, respectively, which stabilizes the current value of zb like the keeper element did in the above solutions. For the proper control of $M5$ and $M10$ an inverted image of zb is required that is provided by the loop-back inverter ($M11$, $M12$). Therefore this implementation is also fully static. A separate output inverter ($M13$, $M14$) is employed to decouple the feedback loop from the load connected at the output Q.

To achieve a proper operation and to meet the circuit's performance requirements, the semi-static CMOS implementation must be sized rigorously. This is done by selecting an appropriate width for all transistors of the gate. A good overview on the sizing rules for the Muller C-elements as well as details on the other implementations is presented in [SEE96].

The local handshaking that is characteristic for the asynchronous approach is often implemented by means of elastic pipelines [Sut89]. Basically these are chains of Muller C-elements. The basic circuit of the elastic pipeline is shown in Figure 2.5. Coordination between the different stages is done by the local handshaking signals (request/acknowledge protocol, recall Figure 2.1c), where one of the inputs of the Muller C-element functions as request (*req*) and the other as acknowledge (*ack*) input. The request is connected to the output of the predecessor stage (upstream), while the acknowledge is the inverted output of the successor stage (downstream).

2.2.4 RS-Latch

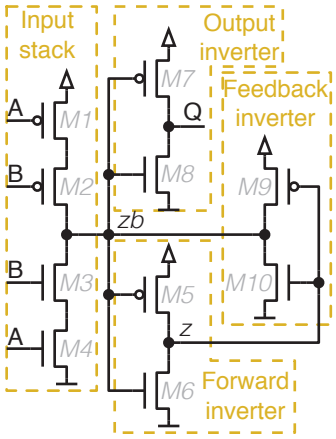
RS-latches are commonly used when designing asynchronous sequential networks[Mur07]. In contrast to a D-latch, where one input represents the data and the other a validity signal, RS-



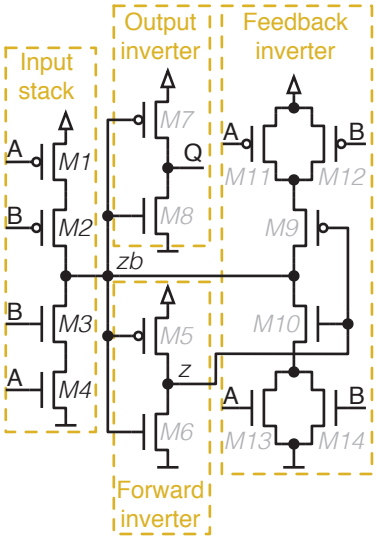
(a) Symbol

A	B	Q_{n+1}
0	0	0
0	1	Q_n
1	0	Q_n
1	1	1

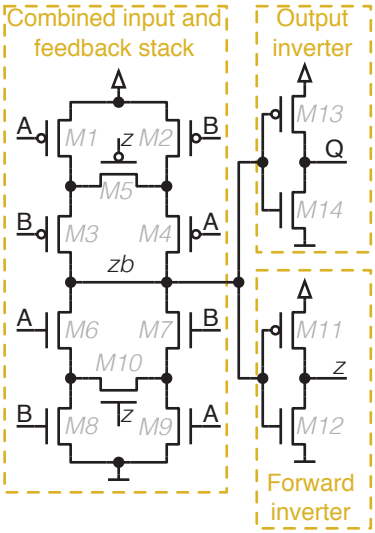
(b) Truth table



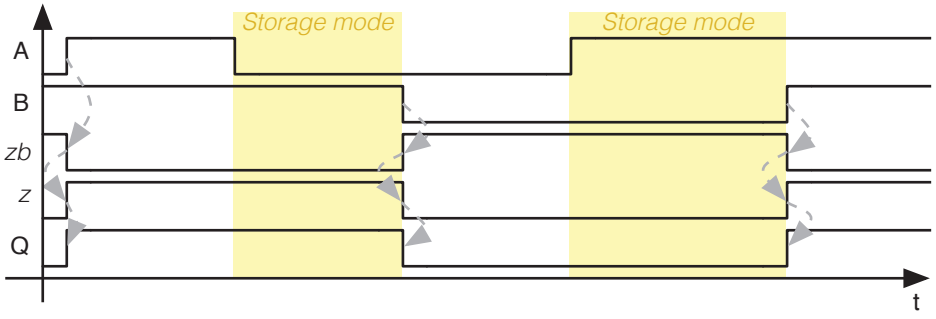
(c) Weak-feedback implementation



(d) Conventional implementation



(e) van-Berkel implementation



(f) Idealized timing

Figure 2.4: Muller C-element

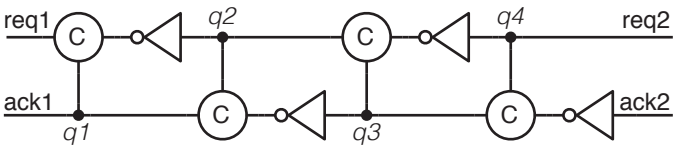
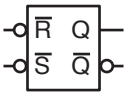


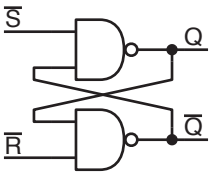
Figure 2.5: Elastic pipeline implementation circuit



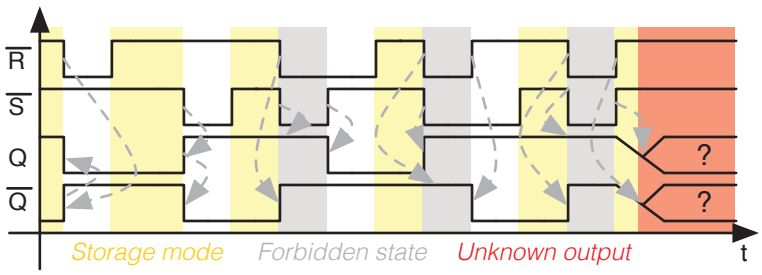
(a) Symbol

$\overline{R}$	$\overline{S}$	Q_{n+1}	$\overline{Q_{n+1}}$
1	1	Q_n	$\overline{Q_n}$
0	1	0	1
1	0	1	0
0	0	1	1

(b) Truth table



(c) Implementation circuit



(d) NAND based RS-latch idealized timing

Figure 2.6: NAND based RS-latch

latches have two inputs, each one combing both of these functionalities. The first one is a reset input, setting the latch to low, while the other one is responsible to set the latch to high. Basically there are two variants of RS-latches, namely NAND and NOR based latches. Both kinds are built from cross coupled gates, but they differ in the active polarity of the input signals. While the NOR based implementation is high active (a high level at the input activates its function), the NAND latch is low active or the inputs must be additionally inverted, otherwise their functionality is the same [Mur07]. In the remainder, we will concentrate on the NAND based version. Figure 2.6 shows the circuit symbol, the truth table and the basic implementation of a NAND based RS-latch.

Figure 2.6d shows the idealized timing of the NAND based RS-latch. It can be seen that, if one of the inputs is low (active) the output executes the corresponding operation (set and reset, respectively), if both inputs are low (the forbidden state), both outputs are high. If only one of the is released, the outputs will perform correctly, but if both are released at the same time, a race condition will occur and it is unclear what will happen to the output, as the state of the

storage loop is inconsistent and must settle before the unpredictable end value of the output can be reached.

The problem of the unknown state arises from the fact that, if both inputs are inactive (high), the NAND-gates are reduced to simple inverters. Therefore, for a stable state, Q must be the inverse of $\overline{Q}$. If this is not the case, the circuit will require an extended time to settle (one output must stay high, while the other one must go to low). In the next section we will have a more in depth look on this race condition.

2.3 Timing Violations

In the previous section we introduced the basic storage elements. The description was based on the assumption that the timing closure is valid and the input and clock signals only change at appropriate times. This assumptions can, however, not be guaranteed, if multiple timing domains interface with each other (e.g. in GALS systems) or if external signals have to be captured. In this section we therefore will analyze what happens, if these timing assumptions are violated.

We were able to get a first glimpse into the problem while looking at the RS-latch. If, for any reason, both inputs are active, the storage loop is in an inconsistent state and, if the storage mode is directly entered after the forbidden state, the end values of Q and $\overline{Q}$ are not predictable.

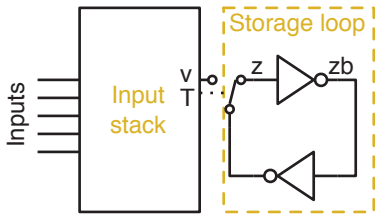
For a more in depth analysis, we will continue with introducing a basic model for the storage loop and using that model to describe the behavior, if the timing assumptions are violated.

2.3.1 Storage Element Model

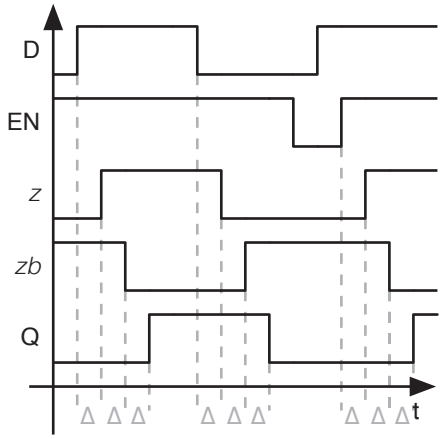
When comparing all the storage elements shown in Section 2.2, all these circuits can basically be reduced to an input stack, responsible for setting new values, and a storage loop responsible for retaining the current value in case the input stack does not drive a new value (see Figure 2.7a). As the input stack is disconnected when a storage operation is in progress, it is normally not considered in the models [Vee80, KBY02]. We will use this model throughout the thesis.

The input stack processes the input signals and generates a corresponding value signal v which should be retained in the storage loop. Additionally a triggering condition T is calculated, which is used to switch between storage and transparent mode. The storage loop can be reduced to two cross coupled inverters. These inverters have the ability to memorize an input bit, if the trigger switch is in storage position. The upper inverter takes the value v and inverts it once, while the lower one outputs the original value again. As long as $z = \overline{zb}$ holds, the storage loop is stable. If, however, the trigger impulse T and a change of the new value v occur at approximately the same time, this condition may be violated. As long as it is assumed that the storage loop has zero delay, it is always fast enough to capture all input changes, no matter how close they are to the trigger condition. If, however, a realistic model is assumed (inverter propagation delay > 0), this assumption may no longer hold (for an example see Figure 2.7c).

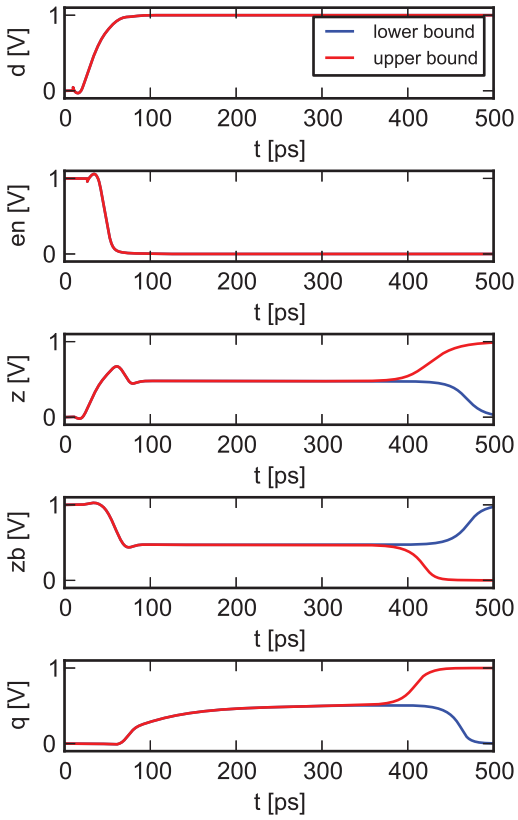
The naming used in the figure is for the case of a D-latch. However, other elements an easily be mapped to the model as follows. For D-flip flops, the model only considers the master latch. If the slave latch is required in the analysis, a second model instance has to be used for it. For the Muller C-element implementation mapping, the nodes z and zb must be exchanged. In case



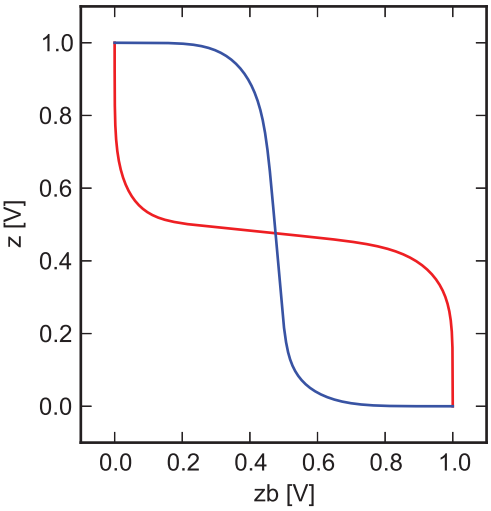
(a) Storage element model



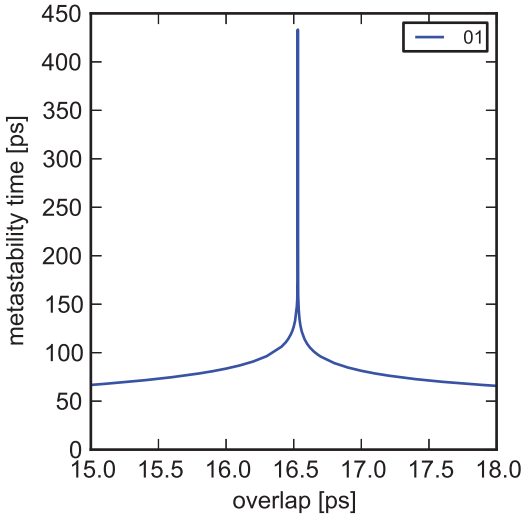
(b) Storage loop delay visualization



(c) Transient trace in the timing domain



(d) Superposition of inverter characteristics



(e) Response time vs. input overlap trace

Figure 2.7: Storage loop model and visualization of timing-violations (D-latch)

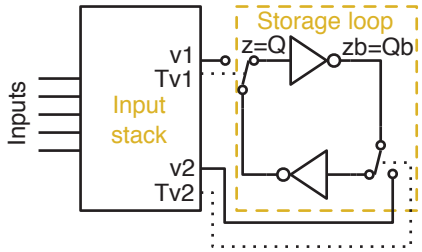


Figure 2.8: Storage element model for RS-latches

of the weak feedback implementation there is no dedicated switching between storage and non-storage mode but the weak feedback inverter is overruled. For the van-Berkel implementation the feedback inverter is part of the input stack.

The mapping of the RS-latch is more involved. The model has to be slightly adopted (see Figure 2.8), as the input stack is connected to both storage loop nodes. Therefore it is possible to set both nodes to the same value in this case. The NAND gates of the latch are replaced by two inverters, each with a switch at the input. To connect these switches, the input stack now has two output values ($v1$ and $v2$) and two trigger ($Tv1$ and $Tv2$) controlling the positions of the switches. Using this representation of the RS-latch, its operation in the metastable regime can be modeled the same way as for the other storage elements (both switches are connected such that only the two loop inverters are active).

2.3.2 Setup- and Hold-Time

When assuming finite delays in the storage loop, an input change will need a certain time until it is safely captured by the storage loop, as a change of value v needs some time to propagate to node z and even more time to reach node zb safely (see Figure 2.7b). If, however, a storage operation is preempted, by disabling the latching condition T, before the operation has finished, the storage loop may be in an intermediate state, with non-digital node voltages (on z and zb) making the digital abstraction of the loop inverter void. Figure 2.7c shows an example of what may happen, if the storage operation is prematurely interrupted. It can be seen that the nodes z and zb indeed need an extended amount of time to reach a digital level. Furthermore while the timing of the input signals is virtually indistinguishable (differences in the sub-ps range) the end value of the output is different in the two depicted cases. This behavior is called metastability and was already mentioned in 1966 by I. Catt [Cat66].

For analyzing the behavior of the storage loop in the metastable region, the simple digital inverter model is insufficient and the full analogue transfer function is required. By superimposing the transfer function of both loop inverters [Vee80] (see Figure 2.7d), besides the two known, stable points (high and low), a third semi-stable point can be seen. This third point, called the metastable point, represents a semi-stable equilibrium. As the smallest deviation from this point (e.g. caused by noise), however, results in an amplification of this difference by the loop inverters, the voltages of the nodes will drift apart and eventually the equilibrium will no longer hold.

Therefore, eventually, one of the known, stable points will be reached. The time the loop will stay in this semi-stable state is not predictable and, at least in theory, it may be infinitely long.

It has been theoretically proved [Mar81, KC87b] that it is impossible to avoid the metastable point and any circuit with storage capability will experience metastability. The problem is based on the fact that it is impossible to map a continuous space cleanly to a discrete one. For the case of two distinct states there must always exist one point in the middle between the states with the same resolution probability for both states. Therefore this point will never be resolved to one of the states.

To visualize how the preemption of the storage operation influences the response time of the element, plotting the response time in dependence of the overlap of the input signals is normally used [Fol96]. Figure 2.7e shows an example for a D-latch. The overlap specifies the time between the corresponding data transition and the disabling of the enable signal. As can be seen, if the data- and the enable-transition are in a critical window, the response time of the latch increases exponentially. To prevent this phenomenon in practice, a so called setup-hold-window is introduced around the disabling point of the latch. Within this window it is, by definition, forbidden for the data signal to change. In the figure, an output delay bound of 100 ps was chosen. To be able to guarantee this bound, no data transition is allowed in the interval of [10.34,10.8] ps before the latch is disabled. If a data edge occurs earlier than this interval, it will be safely captured by the latch (right area of the graph), if it occurs after this interval, it will be safely ignored (left area of the graph). The setup-/hold-window can easily be guaranteed within timing closure of a closed circuit (achieved by using static timing analysis). For interfacing of independent timing domains, however, the setup-hold-window may be violated and metastability is unavoidable in these cases.

2.3.3 Failure Estimation

Based on the simple model introduced in Section 2.3.1, an equation for estimating the rate of upsets of a D-latch has been developed [Vee80]. In an electronic circuit a failure based on metastability occurs, if the metastable state has not resolved before it is evaluated by a successor stage. Therefore a resolution time t_{res} is introduced, measuring the maximum available time for a storage element to resolve an internal metastable state before the first successor stage will evaluate it. The model is also widely used for D-flip flops, however, ignoring the (maybe) different metastability characteristics of the slave latch [JYG09, BCCZ13].

Based on this resolution time and some characteristic values of the storage element and the operating conditions, it is possible to predict the mean time between failure. The equation developed in [Vee80] is:

$$MTBF = \frac{1}{f_{clk}\lambda_{dat}T_W}\mathrm{e}^{\frac{t_{res}}{\tau}} \tag{2.1}$$

In Equation (2.1), MTBF stands for mean time between failure and defines a statistical measure for the mean time between two successive failure occurrences based on metastability. The parameter f_{clk} is the clock frequency of the underlying system and λ_{dat} is the change rate of the data input. Two additional, flip flop dependent parameters are also required for this estimation: τ specifies the exponential decay of metastability over time, while T_W is a measure for the width of the critical input window (setup-/hold-window).

Based on the model the two important characteristics of a storage element are its parameters τ and T_W. Throughout the thesis we will see methods for determining approximate values for those parameters for a given storage element. In literature it has been shown that the metastability behavior may get worse with changing temperature as well as decreasing supply voltage [KGD07, BGC+13]. Nevertheless current metastability estimation models (like Equation 2.1) do not take this into account and assume constant metastability parameters (like τ and T_W). Furthermore the process variation is completely ignored (see [Alf08] for an example from the industry).

While some publications predict an increase of failure rates in the future due to shrinking feature sizes [BGP+10], other authors claim that the observed effect is due to badly chosen flip flop architecture and that specialized synchronizer flip flops will have decreasing error rates for future technologies [YJG11].

The estimated MTBF is, due to assumptions when deriving the formula, only valid for independent clocks. If both, data- and clock-signal, are derived from the same source, e.g. by different outputs of the same phase locked loop (PLL), the actual error rate may be significantly higher or lower than the predicted one [BGDW13]. As the two signals are related, an increased number of setup-/hold-violations may occur. In the (theoretical) worst case, each data transition may occur within the setup-/hold-window, while in the best case none would.

To visualize the failure rate in dependence of the available resolution time, commonly the failure rate (FR, inverse of the MTBF) is plotted over the resolution time [Kin07] (see Figure 2.9). The mathematical model (Equation (2.1) results in a straight line (exponential dependence in a semilogarithmic scale). The real latch, however, will have an area of constant failure rate for negative resolution times (available time less than the nominal output delay). An idealized version of this dependence is also shown in the figure.

2.3.4 Metastability Mitigation

To mitigate metastability specialized circuits are used. In synchronous and GALS circuits this is done using synchronizers while arbiters are the circuits of choice in asynchronous circuits.

Synchronizers

The simplest implementation of a synchronizer is a chain of flip flops [Gin03, Gin11, Kin07]. The goal of a synchronizer is to increase the resolution time to a value such that the resulting $MTBF$ is within acceptable bounds. With each additional stage, the resolution time is roughly extended by one clock cycle [BCCZ13]. The number of stages used therefore depends on the required failure probability. It is important to note that the number of stages is a trade off between reliability on the one and performance and area overhead on the other hand. Each stage increases reliability but also latency and area.

Aside from this brute force approach, more elaborate synchronization techniques are known [PG07, DT10, Gin11]. For these techniques to perform better than the brute force implementation or in some cases even to work, assumptions on the underlying clock relations must be made. As in many GALS systems, the clocks of the different modules may be related (same source but different phase and/or frequency), in many cases these assumptions may be safely

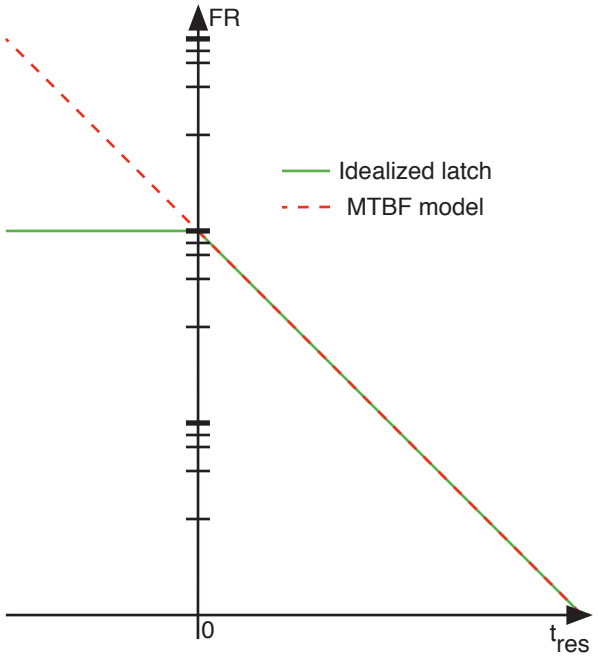


Figure 2.9: Failure rate plot

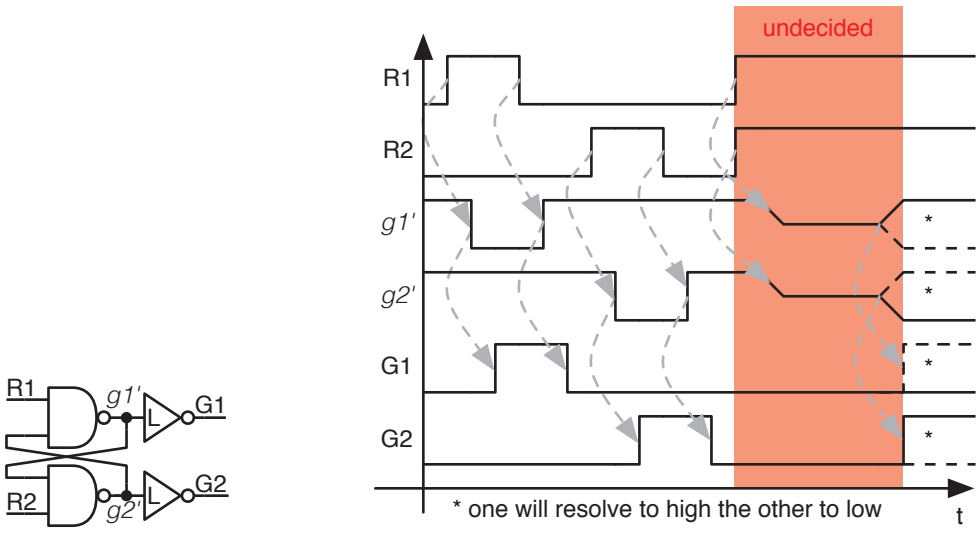
made. These advanced synchronization schemes reduce the required performance overhead significantly. Nevertheless we will concentrate on the brute force method in this thesis as our goal is to better characterize the underlying storage elements in the general case and refine their models. These models may also be used later to refine the more advanced synchronizers as well.

Mutex

In asynchronous circuits, if a resource is shared between two independent clients, the access must be coordinated, as the resource may only be used by one of them at any given point in time. This coordination is performed by a mutual exclusion element [Kae08, Kin07] (shortly mutex).

An additional application of mutexes is the processing of external data. As this data do not not adhere to the circuits request-acknowledge protocol, it must be decided in which handshake cycle the data had arrived. This decision becomes problematic, if the time between the start of a new cycle and the data arrival are approximately at the same time. In contrast to the synchronous system, the available decision time is not bounded by a clock period but may be dynamically stretched by delaying the next handshake cycle until the current one has finished. Therefore, however, it must be guaranteed that the metastable state is encapsulated within the mutex element and the output is only changed after a definite decision has been made.

In CMOS a mutex element can be built from a simple RS-latch consisting of two cross coupled NAND-gates [Kae08, Kin07] (see Figure 2.10a). In the idle state, both inputs are at



(a) Implementation

(b) Timing

zero, which is the restricted state of the RS-latch and therefore both outputs are high ($g1' = \overline{g2'}$). To generate valid grant signals, one output inverter is used for each output. If a request signal changes from low to high, the corresponding grant signal ($G1$ or $G2$, respectively) will go high, eventually. A timing diagram describing the mutex operation can be found in Figure 2.10b.

If both request signals are asserted roughly at the same time, the RS-latch will enter the metastable state (recall Figure 2.6d), and its outputs may become non-digital. Therefore the output inverters must have a low input threshold (in fact often an analogue metastability filter circuit is used [Sei79, KBY02]) and any non-digital output value of the RS-latch will not propagate to the output of the mutex (see Figure 2.10b). Only after a decision is made (the output of the RS-latch have left the metastable region), the output of the mutex changes. This ensures that always a clean, final grant output is created.

3

Methodology and Contributions

The goal of this thesis is to develop a purely digital but still accurate metastability simulation model. Our approach to reach this goal is:

- Investigation of the behavior of relevant components in analogue simulations and FPGA measurements.
- Creation of a digital simulation model based on mathematical derivation.
- Fitting of this model to the results of the analogue simulations.

As we have outlined in Chapter 2, the topic of metastability analysis and mitigation is already widely researched for synchronous circuits. On the other hand its analysis in context of asynchronous circuits is very limited. Therefore we will focus on analyzing the metastability behavior of the basic asynchronous elements (namely the RS-latch and mutex, the D-latch as well as the Muller C-element). Our methodology to achieve this goal will be to use analogue simulation (Spice) to characterize the elements, and to verify the observed phenomena using measurements on physical circuits. In this context the following challenges arise in both areas:

- There exist multiple issues with metastability simulations and their accuracy.
- No adequate measurement infrastructure is currently known to perform an in depth metastability characterization for synchronous elements.
- Measurement concepts for asynchronous components are virtually non-existing.

As no adequate measurement equipment currently exists, we have to develop a custom measurement solution.

Main contributions

- In depth metastability simulations of Muller C-elements [PSL13]:
  The three main implementation variants of Muller C-elements will be throughly simulated using Spice and their reaction to marginal triggering (metastability) will be recorded.

- Metastability containment for Muller C-elements using a Schmitt-trigger [PSL13]:
  Based on the simulation results for the Muller C-elements, we develop a containment mechanism (using a shifted output inverter threshold) to avoid the propagation of the analogue metastability value out of the element. A refinement of this approach, using a Schmitt-trigger instead of the output inverter, will even has the ability to avoid pulses at the output.

- An analysis of the propagation of SETs within an elastic pipeline [PS13d]:
  We will study the propagation of metastability in an elastic pipeline caused by SETs using Spice simulations. To be able to estimate the probability of the propagation, we determine the window of required charge at the input of the pipeline such that the SET reaches a given pipeline stage.

- A precise metastability measurement circuit for in-depth characterization of D-flip flops on FPGAs (without the need for elaborate measurement equipment) [PS13a]:
  To be able to develop the measurement infrastructure, we will first expand a state of the art solution for synchronous circuits. By carefully redesigning a known measurement architecture we are able to achieve a highly increased accuracy for the metastability measurement of FPGAs and additionally analyze the results in more detail (namely enabling the state dependent analysis of metastable operation).

- Digital metastability measurement circuits for Muller C-elements [PS13c] as well as RS-latches and mutexes:
  The additional data derived from the new measurement mechanism for D-flip flops is the foundation for developing the measurement solutions for the asynchronous components. By carefully analyzing the input- and output-conditions for detected metastability cases, a correct interpretation of those cases is possible and a valid measurement infrastructure can be built.

- An analysis matching phenomena observed in measurements to corresponding simulation:
  As the state dependent measurement additionally increases the level of detail available for further analysis, we become able to create a mapping between the simulation and measurement results. By emulating the behavior of the measurement infrastructure in simulations, we show which phenomena are visible in the measurement results and how to map them to characteristics of the underlying circuit found in simulations.

- A purely digital metastability characterization model for D-latches based on late transition and pulse generation [PS13b]:
  Due to the fact that simulation and measurement times for metastability are quite high and the accuracy of the simulators is insufficient for simulating extended metastability events, we will develop a purely digital simulator for studying synchronizers. It will be based on the characteristics of the basic elements (D-latches) gained by virtue of Spice simulations and will employ a purely digital model of metastability. The goal is that the simulation times for the same synchronizer become much shorter using this new model compared to using Spice. Additionally the representation of small input overlaps will not be limited by

the double precision floating point implementation used by the Spice simulation, as we intend to use arbitrary precision decimal numbers to represent simulation time. This will enable us to handle small input overlaps even after extended simulation durations.

Structure The thesis is structured as follows: Chapter 4 is devoted to the characterization of the basic storage elements using Spice simulations, while the simulation of metastability propagation in a micropipeline is described in Chapter 5. In Chapter 6 our measurement solutions are presented and their results are outlined. The matching of measurement results to accompanying simulations is presented in Chapter 7. Chapter 8 shows the digital metastability simulation model and compares its results with Spice simulations, while Chapter 9 wraps up the thesis and gives an overview of future work.

4 Metastability Simulation

There are several different methods to analyze metastability. A very common and widely used one is the method of simulating the basics element using an analogue circuit simulator like Spice (see [YG07, TY10], e.g.).

4.1 State of the Art

In contrast to measurements, the advantage of a simulation method is that all internal nodes of the element can be probed without adding additional load, e.g. by using measurement amplifiers, to the circuit. Therefore the results are not disturbed by the evaluation electronics.

On the downside, however, the used simulators are not designed for this kind of analysis and therefore several problems arise:

- The circuit models are not developed for operation in the metastable ares (out of spec operation).
- Numerical stability of the used numerical solvers
- Exponential behavior when leaving the metastable state may fool the solvers (solver step size).
- Size of the search space

Out of spec operation A simulation results can only be as accurate as the underlying models. For current technologies comprehensive models are available. Using the BSIM standard [CVK+12], the behavior of the underlying transistors is captured quite well. Nevertheless for simulating corner cases, which metastability simulation unfortunately is, the accuracy of the models may not be guaranteed. Therefore simulation results should always be taken with care and sanity checked by accompanying measurement runs.

Numerical solvers As the BSIM model is not analytical, the underlying simulator must use numerical solvers [CVK+12]. This leads to another known problem as the accuracy of the simulators is not sufficient to get meaningful results out of the box. For extended metastability durations, highly precise simulations are necessary. Typical input overlaps are in the ps-range (recall from Section 2.3.1 that the overlap may be around 10 ps) while for deep metastability input overlap precisions down to the zepto second ($10^{-21}s$) range, or even smaller, are required for finding deep metastability. As Spice simulators use double-precision floating point arithmetic in the best case, the accuracy of the input overlap becomes problematic [YG07].

Solver step size In any case, the maximum step size of the underlying simulator must be rigorously controlled [Kin07]. It has to be set low enough to get meaningful results while it must be kept as high as possible to get feasible simulation times. The problem is that in the metastable case, no signals in the circuit change and the simulators therefore tend to increase the calculation step size to its maximum. Unfortunately the exponential behavior of metastability leads to sudden, significant changes in the node voltages. If the step size is too big, however, the simulator will step over the critical point and erroneously increase or shorten the metastability time. Too small step sized, however, will introduce countless, empty simulation cycles outside the critical area. As the simulator is not allowed to exceed this maximum step size, it will also use it in areas where no signal changes occur.

Huge search space Furthermore using simple, uncoordinated simulations alone it will take a long time to find the metastable point. Specialized techniques, such as bisection were introduced to speed up the time required for finding critical input overlaps [YG07]. Bisection uses an upper and a lower bound for the input overlap and refines these bounds by repeated simulations until the critical overlap is found with the desired precision. The repetitive refinement of both bounds has an enormous speed advantage over a simple, linear search algorithm.

Improvements Newer simulation methodologies try to circumvent some of these problems and reduce the simulation time by implementing advanced techniques. In [BG13], e.g., the metastable voltage of the loop nodes (z and zb) is determined by a current compensation analysis (a set of Spice DC analyses) or a set of transient bisection analyses. After the difference voltage between the two nodes is found, a single transient analysis can predict the metastable behavior. This method improves the simulation speed.

4.2 Simulation Algorithm

The storage element models were subjected to a Spice DC analysis to determine the metastable voltage values for the inner nodes of the feedback loop. By simulating the forward and backward paths of the loop separately and overlaying them afterwards using Python, we were able to extract the metastable point (recall Section 2.3.2) from the resulting plots.

A Spice transient analysis was then used to determine the metastable response over time. We created voltage traces of the storage loop as well as for the output nodes. We drove the storage

element into metastability by shortly overlapping matching input signals, that would require the internal state to flip, for a very short time. To be able to also generate the output delay vs. overlap diagrams, we used a linear sweep covering the interesting area. Using adaptive stepping (coarse step size while the time in the metastable state only changes slightly, fine step size in the critical areas), we were able to achieve manageable simulation times. To get meaningful results, we clamped the Spice transient analysis maximum step size to 10 fs. All simulations are based on transistor models of an industrial 90 nm technology with a nominal VDD of 1 V. Our models are pre-layout, therefore ignoring the interconnect characteristics.

4.3 Basic Elements

We start with the investigation on the basic storage elements and will show how they can cope with metastability.

As all storage element implementations comprise a storage loop, setup and hold constraints need to be respected for a proper function. In particular short phases in which the inputs drive the storage loop actively to a value contradicting the output must normally be avoided as this represents the borderline case between clear “switch” and clear “hold”. This violation of the timing constraints will result in an increased output delay. If the output signal is read before that delay has elapsed, a metastable upset has occurred. This involves the risk of reading an ambiguous intermediate voltage.

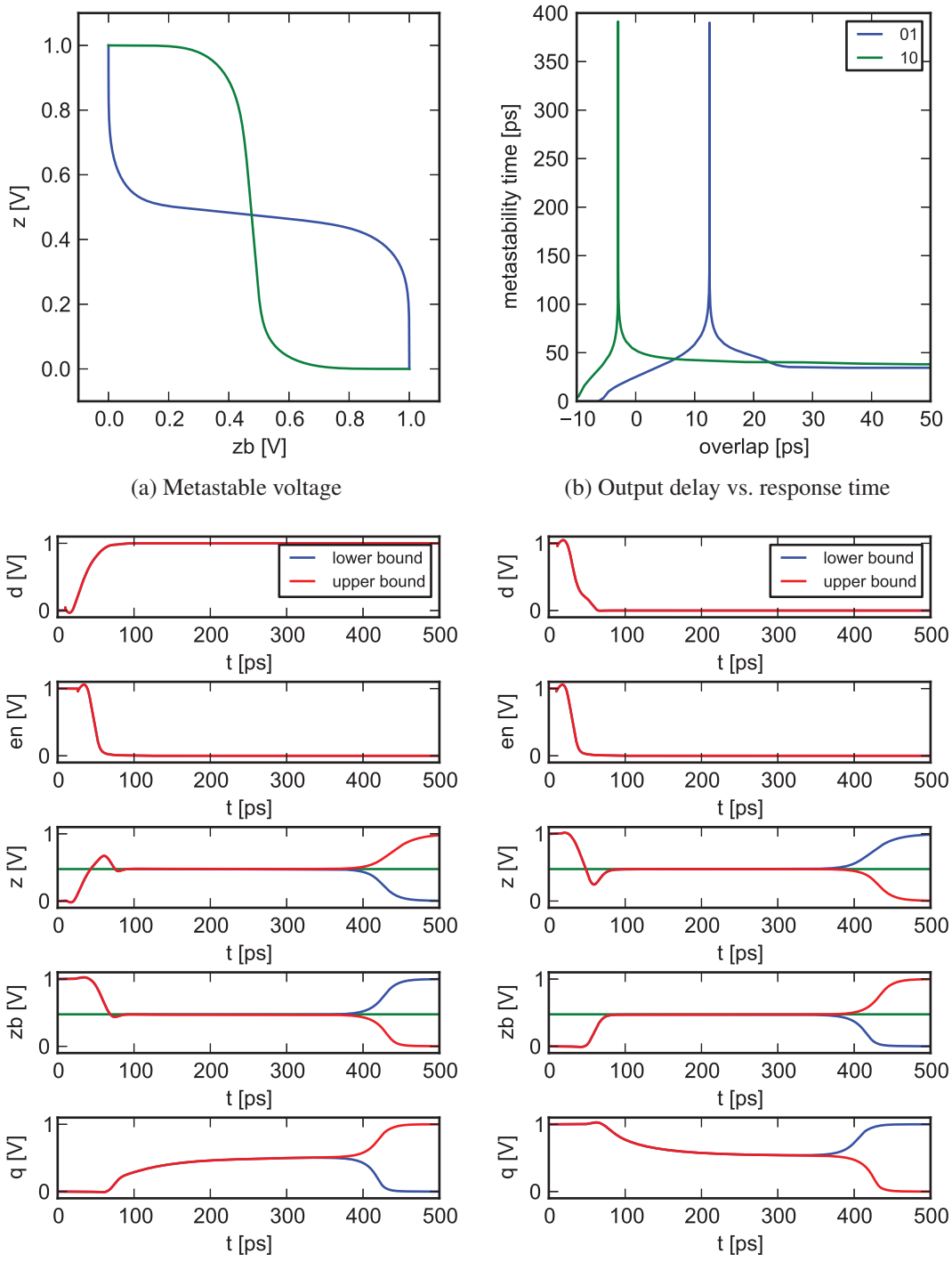
Our simulations were done for a single transistor sizing and are intended as a showcase for our approach. Although the results will quantitatively depend on the sizing, the qualitative message holds for all practical ones.

4.3.1 Metastability Behavior of D-Latches

The metastability characteristics of D-Latches and D-flip flops are widely known [BG13, KD90, Pec76, YJG11]. Nevertheless we start our analysis with these two elements to ensure that our simulation and analysis environment is sufficiently accurate and precise. A D-latch may be driven into metastability, if the input changes in close proximity to the disabling of the latch. In such a case the time for the storage loop to switch may become insufficient.

The simulation was performed on the basic D-latch implementation already shown in Figure 2.2c. As predicted, the storage loop of the D-latch indeed experiences non-digital values, matching the result of the DC analysis (green line), and the exponential dependence of the output delay on the input overlap could also be shown (see Figure 4.1b). Furthermore the results indicate that the behavior for rising and falling edges is non-symmetrical (shift of the increased output delay peak in Figure 4.1b), suggesting the necessity of a state-dependent metastability behavior analysis.

An analogue value may also be seen at the output of the D-latch. Figure 4.1c visualizes this behavior for a rising, Figure 4.1d for a falling edge on the input. The cause for this behavior is that we have used an output inverter with a threshold that is perfectly matched to the loop inverters.



(a) Metastable voltage

(b) Output delay vs. response time

(c) Signal trace (rising edges)

(d) Signal trace (falling edges)

Figure 4.1: Simulation result for a D-latch

4.3.2 Metastability Behavior of Mutexes

In asynchronous circuits it is assumed that all signals adhere to the underlying handshake protocol. If external signals must be processed, mutexes are used to synchronize them with the internal handshake. Therefore the state of the art metastability research in asynchronous circuits focuses on mutexes (see [Kin07, KBY02], e.g.).

Again, the storage loop will experience non-digital values (see Figure 4.2c). In contrast to the other elements, however, the critical case is limited to rising edges on both inputs. This is caused as only in this case the internal RS-latch leaves the forbidden area in an uncontrolled way. It can be seen, however, that the output is not experiencing any non-digital values. The low-threshold output inverters typical for mutex implementations can effectively contain the metastable voltage within the storage loop, as the threshold is only crossed after the metastable state is resolved [Kin07]. The design of the low-threshold inverter can be based on the voltage level found in the DC sweep, as it matches result of the transient analysis (green line in Figure 4.2c).

4.3.3 Metastability Behavior of Muller C-Elements

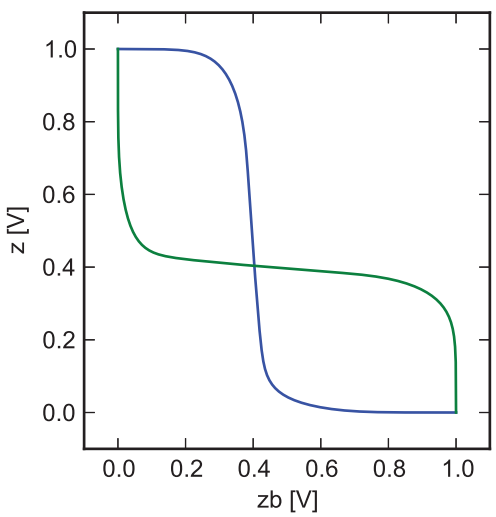
In asynchronous circuits mutexes are normally used to resolve any input conflicts. They themselves may therefore get metastable, but no metastability is seen in the rest of the circuit. This assumption, however, only holds as long as no external errors, such as e.g. single event transients, occur. As these external errors do not adhere to the handshaking protocol, they may upset the circuit. Therefore it is important to also investigate the metastability behavior of other circuit elements, like the Muller C-element.

The truth table of the Muller C-element describes the static behavior only. If a state change of the output is triggered, but this enabling condition is invalidated shortly afterwards, the output may become metastable in response to this marginal trigger. As shown in Figure 4.3, there are two possible scenarios for this: (I) A short pulse on one input with the polarity of the other (stable) input (cases P1 ... P4), and (II) a transition on one input triggering a state change, followed by an opposite (disabling) transition on the other input in close temporal proximity (cases O1 ... O4).

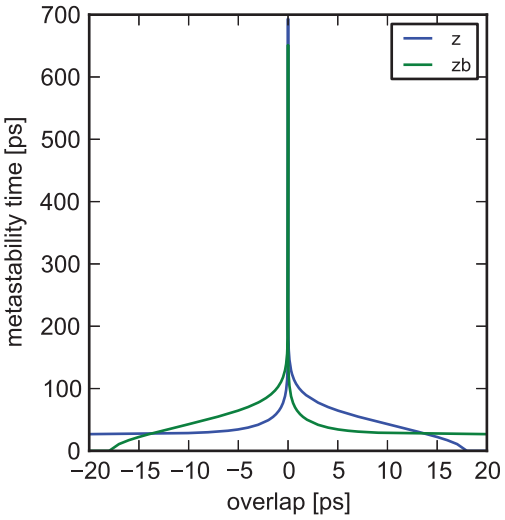
The simulations were done for all three Muller C-element implementations shown in Section 2.2.3. The simulation results for the van Berkel implementation is shown in Figure 4.4. The figure confirms that, as for the D-latch and mutexes, the inner nodes (z, zb) go to a non digital level while the element is metastable. The straight green lines in the figure represent the result of the DC analysis. It is apparent that the analogue voltage level found in the transient analysis again complies nicely with the result of the DC analysis. Figure 4.4b shows the response time for all cases introduced in Figure 4.3. It can be seen that this implementation is symmetric with respect to the inputs (the traces for O2, O4, P2 and P4 as well as O1, O3, P1 and P3 are nearly the same). As we have seen for the D-latch, the critical overlap is, however, dependent on the state of the element.

As the results for the other input conditions and implementations vary only in the length of the metastable state but not in its existence, they were omitted here but can be found in Appendix A.

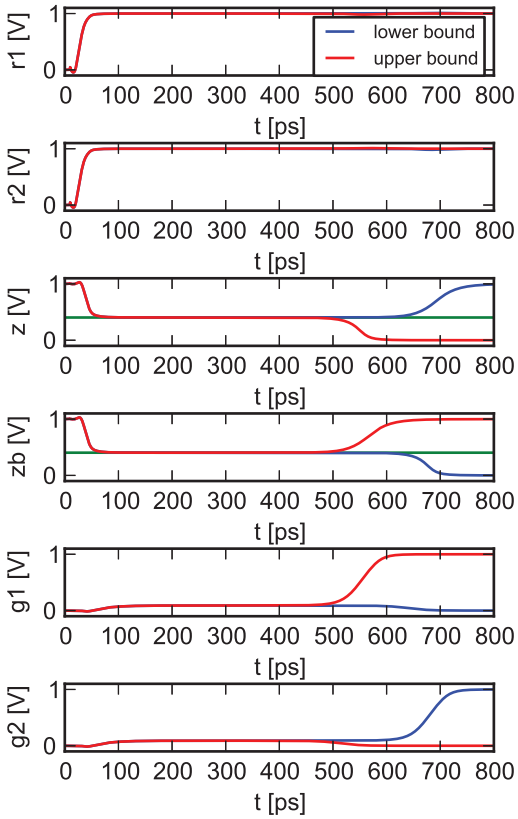
It is also visible in the figures that the analogue voltage level again propagates from the



(a) Metastable voltage



(b) Output delay vs. response time



(c) Signal trace (rising edges)

Figure 4.2: Simulation result for a mutex element

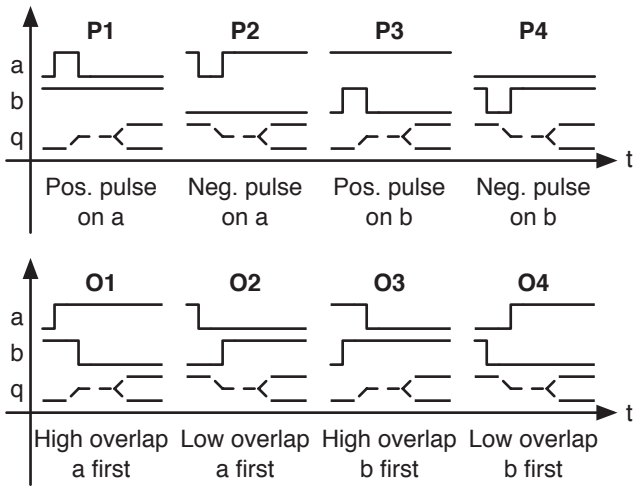


Figure 4.3: Metastability conditions of a Muller C-element

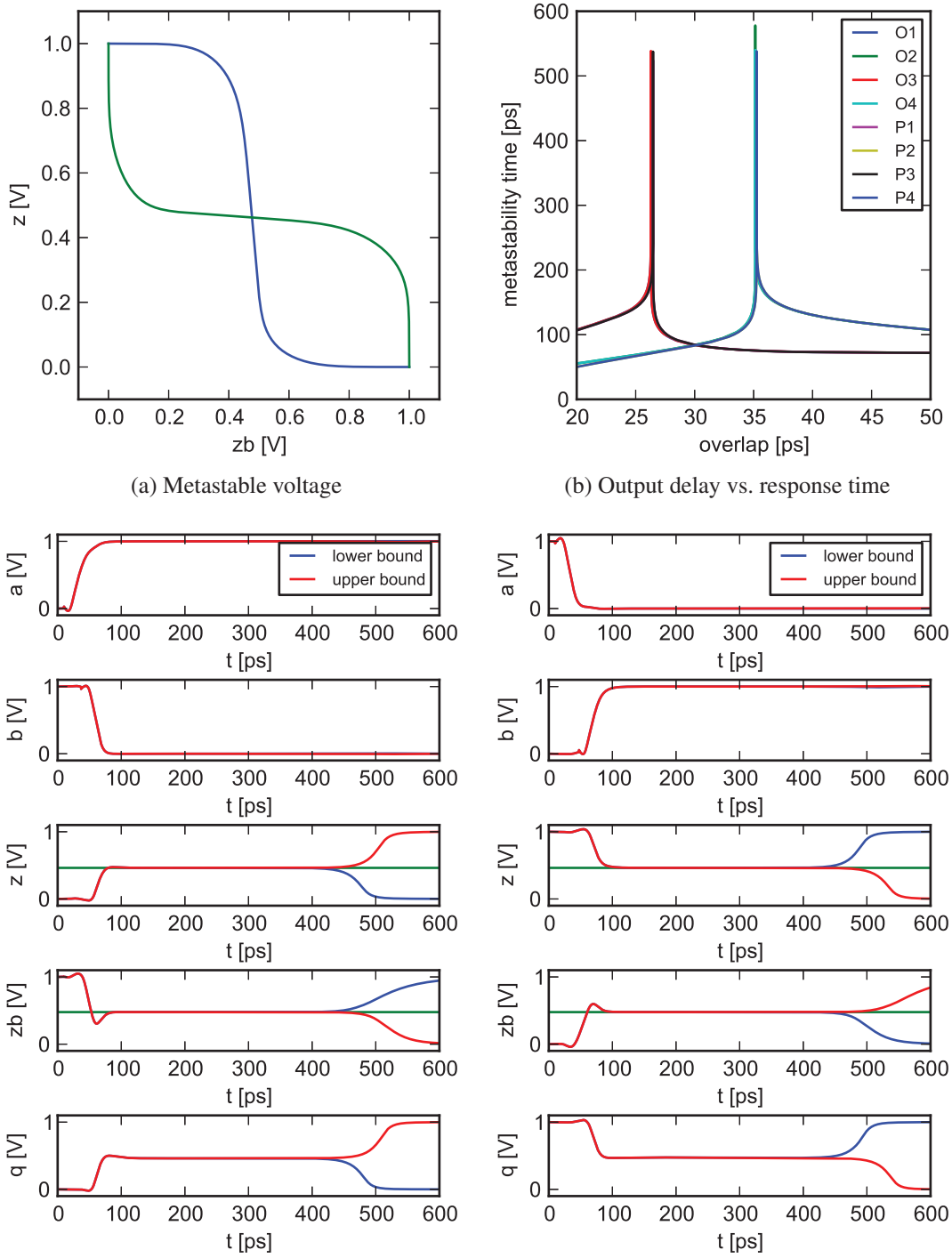
inner node to the output of the Muller C-element. This is because in our simulations we had chosen the same threshold voltages for all inverters, and thus the metastable voltage matches the threshold voltage of the output inverter quite well. In contrast to a synchronous system, however, the analogue voltage may cause severe problems in asynchronous circuits. As no clock signal is present, no temporal masking of the metastable state is present here. An analogue level may directly propagate to the next stage. Since it has been proven impossible to prevent metastability in the first place [Mar81, KC87b], we at least want to confine the metastable voltage to the inner node to mitigate its propagation.

We will therefore continue with analyzing the behavior of the output inverter and try to redesign it so as to contain metastability within the element. We base our efforts on the technique already in use for the mutex (see Section 4.3.2) and will analyze if this techniques may lead to a better metastability containment in case of the Muller C-element and how to adapt it to achieve even better results.

4.4 Metastability Containment for Muller C-Elements

As the metastable voltage can be efficiently determined by a DC analysis in advance, it is possible to devise the metastability counter-measures that are based on its knowledge. More specifically we can select the threshold voltage of the output inverter in such a way that the metastable voltage is uniquely regarded as a high or low value. In this way the output inverter will safely convert the intermediate voltage level into a defined low or high value at the Muller C-elements output. This is the idea behind the approaches presented in this section.

The solutions were tested using a set of Spice transient analyses. To speed up the simulation, the duration of the critical overlap was, in contrast to the previous simulations, extracted using a binary search on the input interval. We implemented the binary search in Python and used again



(a) Metastable voltage

(b) Output delay vs. response time

(c) Signal trace (rising edges, condition O1)

(d) Signal trace (falling edges, condition O2)

Figure 4.4: Simulation result for a van Berkel Muller C-element

Table 4.1: Comparison of the metastability mitigation implementations

	low → high	high → low	low → low	high → high
Low threshold inverter	OK (late)	OK	OK (suppressed)	glitch
High threshold inverter	OK	OK (late)	glitch	OK (suppressed)
Schmitt-Trigger	OK (late)	OK (late)	OK (suppressed)	OK (suppressed)

hSpice as simulation back-end. When the binary search was finished, the upper and lower bound traces were plotted and used to analyze the metastability behavior.

4.4.1 Low- and High-Threshold Inverters

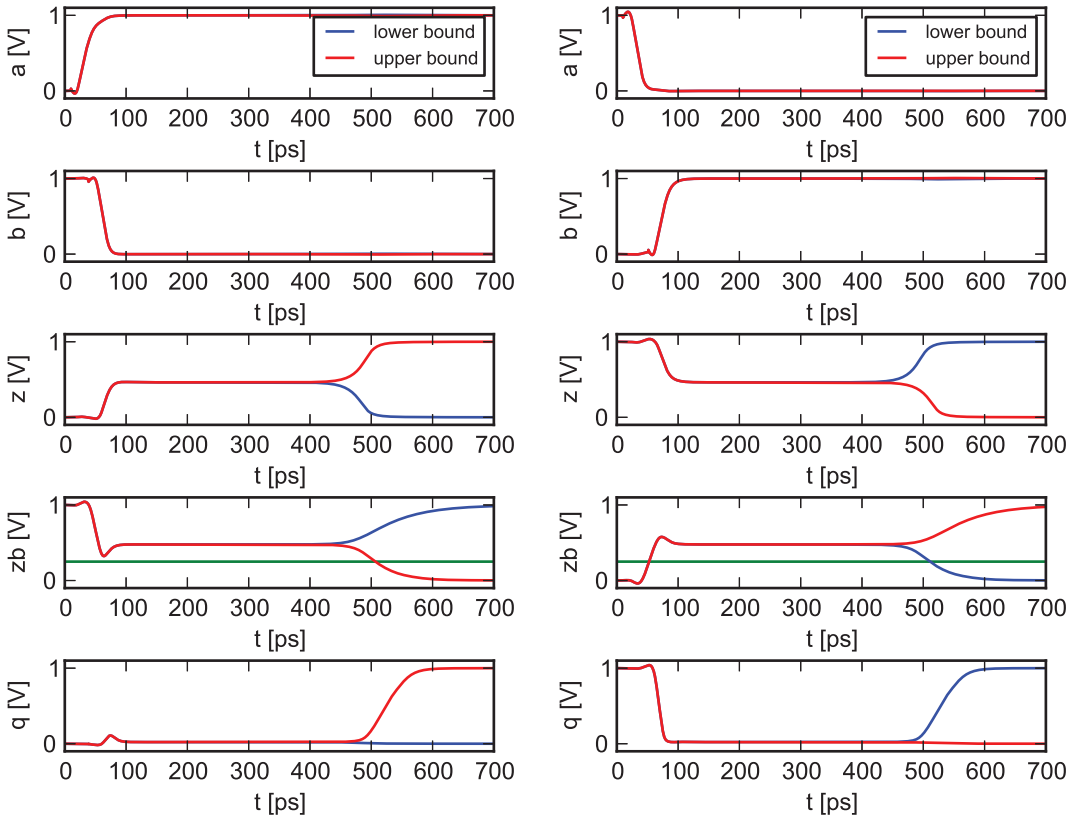
As already mentioned, for mutual exclusion elements low-threshold output inverters are used to delay the output for the low-high transition until the RS-latch has left the metastable area [Kin07]. When using the solution for the Muller C-element, the low-high transition works as expected and is delayed until the metastable state has been resolved (see Figure 4.5a, the low threshold is indicated by the green line). Unfortunately for the high-low transition a glitch is created: When the element is entering the metastable state, the voltage level at the internal node is crossing the threshold and the output switches from high to low immediately. If the metastability then resolves to the original state of the element, the voltage on the internal node again crosses the threshold and the output signal switches back to high (see Figure 4.5b). This glitch in the high-low transition is not a problem in the mutual exclusion element, since the critical phase of the arbitration process always starts in the low state. That is why this approach works fine there. We expect from the containment circuit, however, that the state is cleanly switched in both directions. Especially in asynchronous control circuits the above mentioned glitch may propagate and be the cause for unexpected behavior.

When using a high-threshold inverter instead, a dual problem can be observed. In this case the high-low transition works as expected but the low-high transition creates a pulse. In general we can distinguish four cases that must be considered when designing the containment circuit. Table 4.1 shows how the different cases are handled by the measures shown in this thesis.

Based on these observations it becomes clear that an element with an adaptive threshold is required to contain metastability within the element. The threshold must be raised when the output is high and lowered when the output is low, so that it is above the metastable voltage for high-low and below the metastable voltage for low-high transitions, respectively. The natural choice for such an element is an inverting Schmitt-Trigger.

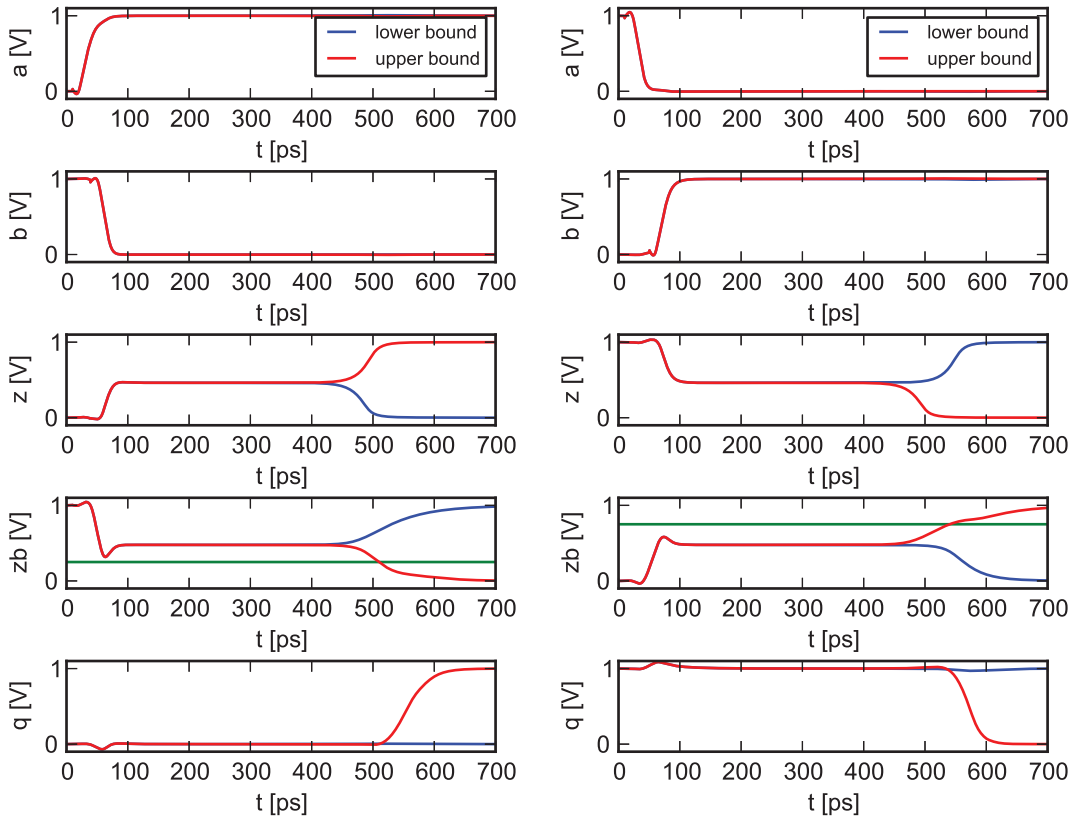
4.4.2 Schmitt-Trigger

Figure 4.6 confirms that when using a Schmitt-Trigger as output stage of the Muller C-element, both the low-high and high-low transitions work properly (the adaptive threshold is indicated by the green line). Due to the hysteresis, the switching threshold of the Schmitt-Trigger is adapted in such a way that for both transitions just moving to the metastable state is not sufficient for the



(a) Case O1 (b) Case O2

Figure 4.5: Simulation result for the low threshold inverter



(a) Case O1

(b) Case O2

Figure 4.6: Simulation result for the Schmitt-Trigger output stage

analogue voltage to reach the threshold, and therefore the output signal does not change its state. Only if the element definitely changes its state, the threshold is crossed and the output signal switches as well.

One might wonder whether this mechanism is not applied for synchronous elements too. It is important to note that the boundary conditions for synchronous and asynchronous metastability containment are quite different. Notice that this result is essentially different from what [KC87a] claimed for synchronous circuits: There the additional delay introduced by the Schmitt-Trigger consumes so much of the available resolution time that the overall upset probability is even increased. The essential problem here is that in synchronous circuits the resolution time for metastability is globally determined by the fixed clock frequency in an open-loop fashion [KC87a]. Quite in contrast to that, in the asynchronous case the resolution time is naturally extended as required, as the handshaking forms a closed loop timing control. In this way the output can be delayed without increasing the upset rate. The only constraint is a clean transition from one output level to the other without staying at an intermediate voltage and without glitches – and this is what the Schmitt-Trigger can effectively provide.

Nevertheless it is important to note that the additional delay may cause the loss of signal transitions at the timing domain boundaries. As no handshaking is applied at those interfaces, the source may issue the next transition too early (while the receiving Muller C-element still decides on the previous one). In this case both transitions may get lost as the element is driven back to its previous state by the new transition. In fact this is the same mechanism as known from clock domain boundaries in the synchronous case: If the sender clock is faster than the receiver clock, the receiver may loose some transitions.

After having shown the applicability of the Schmitt-Trigger approach for metastability containment in Muller C-elements, let us briefly analyze its penalties:

1. *Area Overhead:* The standard van-Berkel Muller C-element implementation consists of fourteen transistors (including the output inverter). The proposed version with the Schmitt-Trigger output stage uses 18 transistors, as the Schmitt-Trigger requires four additional transistors compared to the inverter circuit. This yields an area overhead of 28.57%.

2. *Delay Penalty:* The nominal output delay of the standard Muller C-element in the used technology is 28 ps. The Schmitt-Trigger increases this delay to 60 ps (214%), leading to an overhead of 32 ps in the fault-free case.

While the area overhead is quite moderate, the delay penalty is considerable. However, protection against metastability has its price in the synchronous domain, as well. At timing domain boundaries this reduction in performance will limit the sustainable data transfer rate.

4.5 Summary

The simulations performed in this chapter have shown that the metastability behavior of all the investigated storage elements is similar. The length of the metastable state may vary but the existence and the non-digital behavior of the storage loop remain unchanged by the implementation. An interesting result is that the behavior for the triggering conditions (rising, falling edges for D-latches, cases O1-O4 and P1-P4 for Muller-C elements) different critical overlaps arise which is not covered by current metastability models.

As the simulations have shown, an analogue value at the output is highly improbable, as the matching of the output threshold and the threshold of the loop inverters must be very precise in this case. Furthermore the inertia of the signals requires the metastable state to be present for a prolonged time to settle to the critical voltage. Only very deep metastable state will create such a condition. In most practical cases, however, the output will be either a late transition or a pulse.

Furthermore we have seen that a Schmitt-Trigger is able to contain the non-digital voltage in the storage loop and safely converts it into a variable output delay. In asynchronous circuits, due to their stretchable handshake pattern, this behavior can be exploited to safely capture external control signals. In the synchronous case, however, the additional delay of the Schmitt-Trigger decreases the available resolution time and may lead to even worse dependability results (see [KC87a]). The problem is that the resolution time is based on the strictly periodic, non stretchable clock signal.

5 Metastability Propagation in Micropipelines

While the propagation behavior of metastability in flip flop chains (so called synchronizers) is widely researched [JYG09, Gin03, BCCZ13], there are only a few results for elastic pipelines known. In this chapter we want to show how metastability may propagate through an elastic pipeline. To use a more realistic setting, metastability will be triggered by an SET hit at the input of the pipeline.

5.1 State of the Art

As already mentioned, elastic pipelines are very convenient and fundamental building blocks in asynchronous design. Their classical application (as proposed in [Sut89]) is flow control in data pipelines. While in the meantime more efficient data pipelining schemes have been proposed, the original elastic pipeline scheme is still very useful for FIFO buffering of transitions or the implementation of fast asynchronous up/down counters [VPSS12] and many more. By careful interleaving of their constituent Muller C-elements (see Section 2.2.3) this scheme achieves a rigorous timing closure such that a transition at one stage is acknowledged right after it has been safely captured by the next downstream stage. That is why it seems evident that short fault pulses on the input will (1) either, if they are long enough, be appropriately captured, thus turning them into extra transitions that are stored in the pipeline, or (2) they will, if they are too short, simply not be acknowledged and hence masked. The latter case is of course the more desired one in context with fault pulses, while the former one can normally still be handled by suitable error mitigation techniques on the data level.

This simplified block-level view of the pipeline operation, however, is somewhat optimistic, as it does not consider the non-idealities of the Muller C-element's internal implementation (see Sections 2.2.3 and 4.3.3). Therefore, when exposed to marginal pulses whose amplitude is somewhere between the well defined logic levels, the behavior of the elastic pipeline is not evident. The focus of this chapter will be an exploration of this very situation. We will, by means of Spice models, identify the class of pulses that can trigger such an undefined behavior, estimate the relative probability of occurrence of these critical pulses (i.e. how exactly must the

parameters be chosen to yield a critical pulse) and identify the relevant properties of the Muller C-element implementation in this context.

The importance of characterizing this non-ideal behavior is based on the increasing number of radiation effects being reported for newer technology nodes [KHP04, RDK+08, DW11]. Since the charge used for storing information quadratically decreases with the feature size of the chip [KHP04], the soft error rate (SER) starts to exceed acceptable limits.

While bit flips in storage cells due to particle hits (single-event upsets, SEUs) are widely researched[KHP04, DW11], considerably less work has been done on transient pulses (single-event transients, SETs) caused by particles hitting combinational gates. Especially the SET behavior of asynchronous components, like Muller C-elements, has not received much attention.

Three types of masking mechanisms are distinguished in the literature that may prevent an SET from causing a failure [Bau05]: *electrical masking* comprises mechanisms that degrade width and amplitude of the pulse in the analog domain (like RC filtering), *logical masking* occurs when a pulse is blocked by a logic gate whose other input(s) already determine the output, and *temporal masking* is due to the fact that storage elements like flip flops typically do not consider their input level during certain phases. Obviously all these masking effects depend on many details and conditions. One very pronounced difference between synchronous and asynchronous logic concerns the temporal masking: In a synchronous circuit temporal masking is relatively easy to model, due to the periodic, independent nature of the clock as well as the strict decoupling between a flip flop's input and output. Consequently SET propagation through a shift register is not an issue – in the worst case the SET causes a metastable upset in the first stage, which is still thoroughly covered by respective modeling approaches. In asynchronous logic, however, the masking window is causally and temporally correlated with the SET event, and a transparent latch (or Muller C-element in the storage state) may, in principle, propagate a pulse. Hence the modeling becomes much more intricate, as already mentioned above. Still, it is important to have suitable models at hand, since one cannot simply rely on a fault tolerance concept (like logical masking by a voter), if not all possible behaviors of the circuit are considered or even understood. Even if the undefined behavior should turn out sufficiently improbable to be ignored in retrospective, this decision can only be made after having assessed the respective probability. Note that, in contrast to existing literature (like [MRL05, GYB07, KZYD10, PCV12]), we do not aim at devising novel radiation tolerance or hardening methods – our aim is to thoroughly study the effects of SETs in an elastic pipeline in different settings, and to reveal potential unexpected behavior.

If an energy particle strikes a sensitive site in a circuit stage, it creates electron/hole pairs along its propagation path. This extra charge induces an undesired current from the n-type diffusion to the p-type diffusion through a pn junction that affects the operation of the circuit. Note that technology scaling relies on reducing nodal capacitance and voltages, thus also reducing the critical charge (Q_{crit}) required to upset a circuit node. This makes advanced sub-micron technologies more susceptible to soft errors [Bau05]. To understand the effect on the circuit consider an inverter with a high output: A radiation-induced current pulse at the NMOS transistor (that should be off in this state) can cause a low-pulse at the output, if it is stronger than the current driven by the PMOS transistor that counteracts it. This voltage change (SET) will then be propagated to the next stage(s). It causes an observable effect, a so-called soft error, only if it finally

becomes visible at a primary output, or if it is latched into a memory element of the circuit causing a so called SEU. This is where the masking mechanisms outlined above may become effective. Although these masking mechanisms reduce the rate of soft errors, it has been observed that their impact is lessening across technologies. Alternatively, the particle may as well directly hit a storage element in a memory or sequential logic and thus cause an SEU without any masking being involved.

The physical effects of radiation on digital circuits are widely researched. These studies normally expose specialized target circuits, mostly inverter chains, to radiation (see [NP03, BB97, AAW+11, GAN+11] ,e.g.). In other approaches the effects of radiation are simulated on the physical level using specialized software suites ([ME07, RDK+08]).

However, for studying SET propagation in circuits comprising more than a handful of transistors, analog-level simulators such as Spice have become the method of choice. A number of current models have been proposed in the literature over the years [RCI94, BB97, CRL+03, ME07], which model the charge-injection effect of particle hits by current injection.

The shapes of SET pulses depend on operating conditions like supply voltage or temperature and therefore it is hard to find a simple, yet accurate model. So, although more complex models [Kle09] do exist, the state of the art technique for mimicking the actual charge deposition mechanism of a particle strike is to use a current source connected in parallel to the channel of the hit transistor [ME07] for injecting a double exponential current pulse according to Equation (5.1).

$$i\left(t\right) = \frac{Q}{t_f - t_r} \cdot \left(e^{-\frac{t}{t_f}} - e^{-\frac{t}{t_r}} \right) \tag{5.1}$$

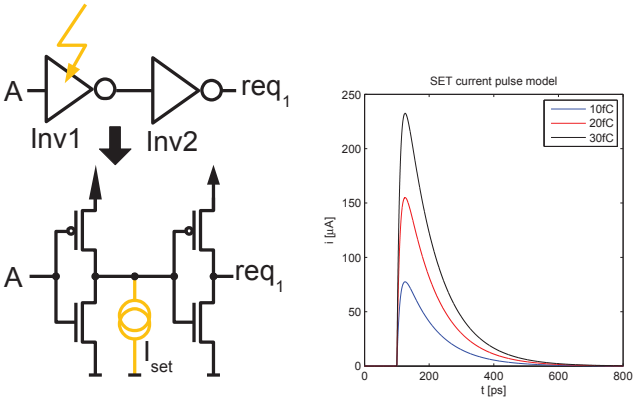
Herein, i is the transient current pulse, Q is the injected charge, t_f is the decay time of the current pulse, and t_r is the time constant for initially establishing the ion track. Although, as already mentioned, the actual SET pulse shape very much depends on many conditions, a usual choice which can be found in the literature [ME07] is $t_r = 10ps$ and $t_f = 100ps$.

There are several papers (like [MRL05, GYB07]) studying SET susceptibility of asynchronous circuits on the logical level. As our aim is to find corner cases in the pulse propagation, the logical level is, however, not suitable for our analysis.

Pulse propagation in micropipelines on the electrical level was already investigated in [PM05, KZYD10, FFS09]. The first paper performs a thorough Spice pulse injection campaign varying several parameters (like VDD or sizing). Unfortunately the used step size for the pulse width is quite coarse (10 ps) and therefore no pulse propagation after the first stage was detected. SET impacts onto pipelines are studied in the second paper. Its main goal is the hardening of the circuit and not the propagation analysis of unlatched pulses. The third one describes the micropipeline as kind of synchronizer for pulses mitigating the effects of short input pulses with each pipeline stage. Only the basic analysis was done using Spice, while the main analysis, however, was performed using a MATLAB model.

5.2 Spice Simulation Setup

Our evaluation was done using hSpice. The test circuit consists of the SET target circuit (inverter, see Figure 5.1a) and the micropipeline itself (recall Figure 2.5).



(a) Pipeline driver circuit (b) Current model

Figure 5.1: Driver circuit and SET model

We decided for a pipeline with fifteen stages to make sure that unlatched pulses do not reach the end of the pipeline in any of the simulations. For this length we can safely assume that the pulses either die out or are latched within the pipeline before reaching the end.

The assumption for the simulations is that an SET is injected into the circuit driving the input of the pipeline. Therefore a pulse will be created at the input of the pipeline. We choose this strategy because we are interested in the SET (metastability) propagation properties of the pipeline and not its SET generation properties, and for this purpose all stages upstream of the injection point become irrelevant. For the same reason we consider the empty state of the pipeline only: From a functional analysis of the elastic pipeline it is straight forward to conclude that a transition or pulse at the input (req) will not propagate beyond a full pipeline stage – the Muller C-element's input is simply blocked by the missing acknowledge[1].

The simulations were performed on a $90nm$ industrial CMOS process specification with a nominal VDD of $1V$. We used a basic p-to-n ratio of $960/480nm$. The weak feedback inverter was chosen to have a width of one fourth of the nominal one. The output stages were also implemented using the same basic sizing. The threshold of the output inverter was set by adjusting its VDD in the range between $0.75V$ and $1.2V$, leading to a threshold range of $0.355V$ to $0.593V$, while the threshold of the loop inverters is $0.487V$.

An inverter was chosen as SET target as its behavior under such conditions is well researched [ME07]. Depending on the desired pulse direction ($0-1-0$ or $1-0-1$) the input of the inverter must either be connected to ground or VDD, respectively. To get more realistic pulse shapes on the input of the pipeline, we use a second inverter to simulate the pulse propagation and shaping present in the real driver circuit. The complete driver circuit can be found in Figure 5.1a.

As already mentioned in Section 5.1, the SET itself is modeled as a double exponential current pulse (see Figure 5.1b) with a rise time (t_r) of $10ps$ and a fall time (t_f) of $100ps$. These values were taken from preliminary experiments and literature [ME07].

[1] A full pipeline would allow studying the propagation of an SET along the ACK path, i.e. from output to input. For symmetry reasons the behavior will be the same as for req, so we focus on the latter here.

To simulate hits of different intensity and position within the target inverter, we varied the injected charge Q, and therefore the injected peak current. Using this scheme, we are able to determine the critical charge necessary for an SET to propagate through the pipeline.

First the critical charge for creating an SEU in each individual stage (latching point) is determined using a simple binary search. Starting with a charge which surely creates an SEU and one which surely does not, the distance between upper and lower bound is cut in half. If the newly simulated charge creates an SEU, it is used as new upper bound, otherwise it is used as new lower bound. The process is continued until the difference between upper and lower bound is in the order of $10^{-25}C$. Afterwards, the lower charge limit for SET propagation (versus the pulse dying out) is determined by separate simulations for each pipeline stage.

5.3 Simulation Results

The simulations have shown a potential for SET propagation (without being converted into an SEU) within the pipeline. One of the observed cases can be seen in Figure 5.2. It shows the behavior for two nearly identical SETs at the input (charge difference of $0.004fC$). For each of the SETs a separate simulation run was executed. As the analysis will show, the two SETs are indistinguishable in the first stage of the pipeline but with each stage the two SETs will become more and more different, since the charges were selected to be near the balance point between pulse amplification and extinction.

- First stage: The input pulses on req_1 propagates into the first stage of the pipeline and show up in the storage loop (z_1 and z_{b_1}). The pulses are not long enough to be latched and disappears from the storage loop even before an acknowledgment from the next stage is received (rising edge on q_{b_2}, marked by the dashed line).

  Even though the pulses are not latched, they cross the low threshold of the output inverter ($0.355V$ in this case) and propagate to the output of the first stage (q_1).

- Second stage: In the second stage the pulses also show up in the corresponding storage loop (z_2 and z_{b_2}), but are already distinguishable. The differences in the pulse charges are big enough such that the characteristic of the first stage and the input stack of the second one cause an amplification of the upper, and an attenuation the lower pulse. Otherwise the behavior in the second stage is exactly the same as the one of the first stage.

- Third stage: The behavior of the third stage is somewhat different. As the two pulses have been sufficiently amplified and attenuated, respectively, the third stage is able to latch the upper pulse, creating an SEU, while the lower pulse is too small to propagate any further and dies out.

- Fourth stage: We observe pronounced logic levels only.

This example shows that an SET can, indeed, propagate through a micropipeline without being latched. In the following we will analyze this behavior and, for our simulation circuits, give a window of critical charge necessary for SET propagation to a certain stage in the pipeline.

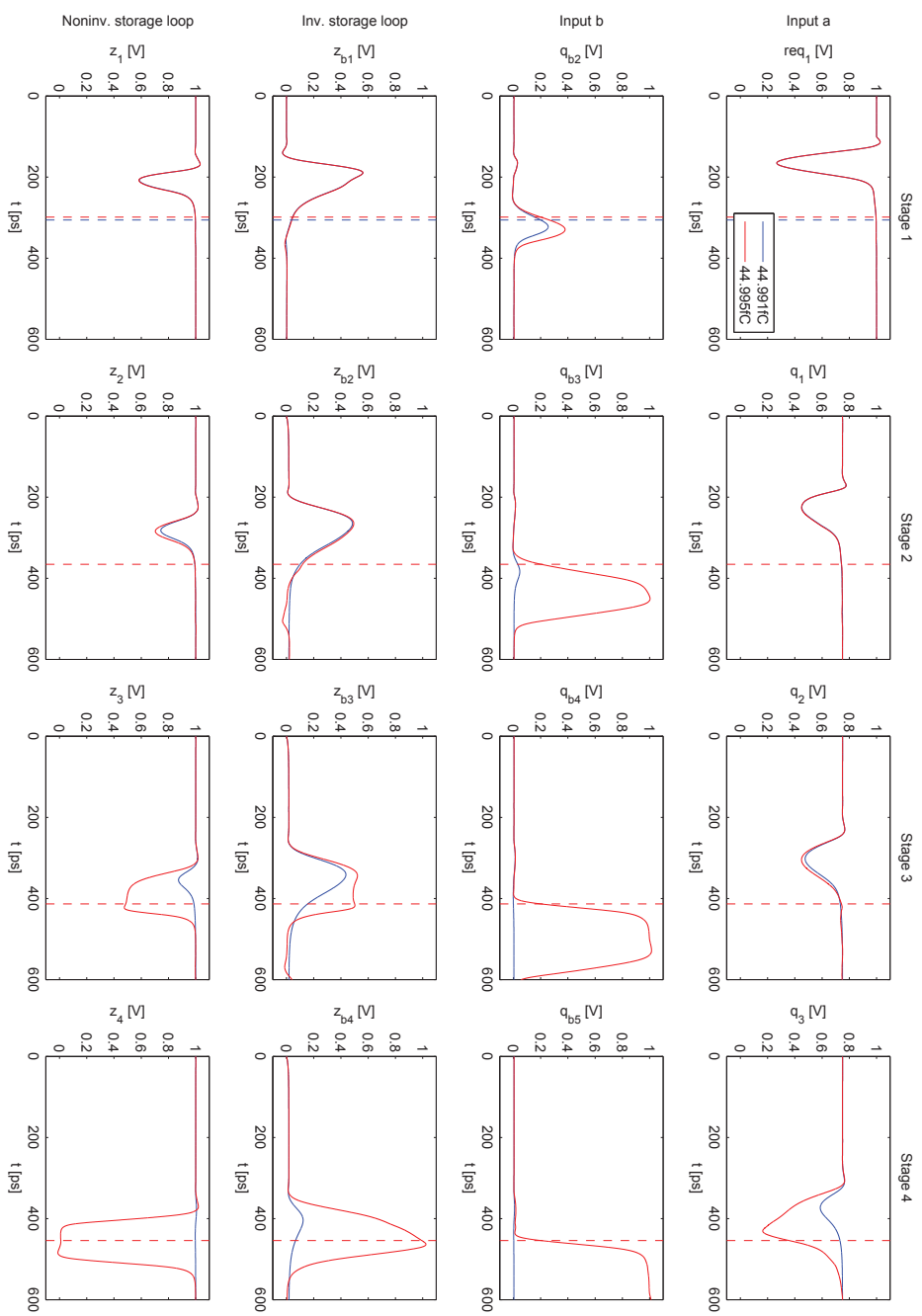


Figure 5.2: SET propagation in a micropipeline (latching/extinction of SET in third stage, van-Berkel implementation, $V_{th} = 0.355V$)

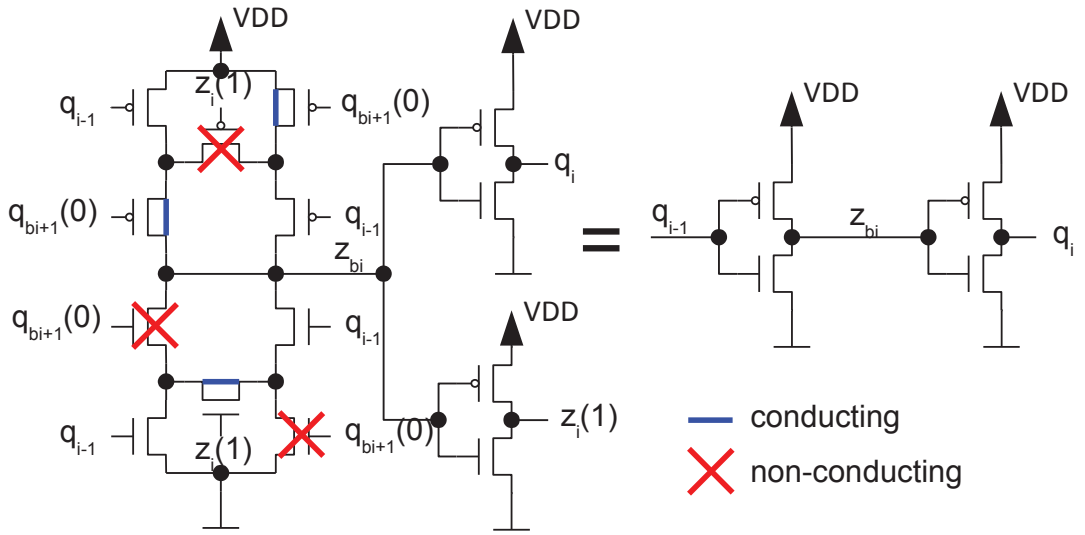


Figure 5.3: Micropipeline stage in empty state

This window will give us an estimation on how probable the propagation of an SET over multiple pipeline stages really is.

5.4 Evaluation

As the results have shown, it is possible for an SET to pass a pipeline stage without getting latched. Now we want to further analyze the conditions required for this behavior. The analysis will be performed for the van-Berkel implementation and for $1-0-1$ pulses. We have done the simulations for all mentioned implementation options and for the $0-1-0$ pulse as well. As the results from the experiments lead to the same qualitative results and only differ quantitatively, we have chosen to concentrate on this one case.

As motivated in Section 5.2 the pipeline was in the empty state ($q_i = z_i = 1$, $q_{b_i} = z_{b_i} = 0$). Therefore the local handshakes are set such that a new input transition will be accepted by the pipeline stage immediately. Figure 5.3 shows the equivalent transistor level circuit of a pipeline stage in this state.

As can be seen in the figure, the pipeline stages are in a “quasi-combinational” mode in this case. In fact, the pipeline forms an inverter chain. As long as its storage loop is not flipped, the pipeline will stay in this mode. This assumption holds because in the analyzed cases, the threshold of the output inverter is lower than the threshold of the storage loop. Together with the inertia of the loop, the Muller C-element will not switch into storage mode. This model looses its correctness only for those cases where the thresholds nearly match.

If now a short pulse is present at the input of the pipeline, the first stage will forward the pulse to its inner node z_{b_1}. If the energy of the pulse is not sufficient to flip the storage loop (beneath the latching point), the pipeline stage will stay in the inverter chain configuration.

Assuming that the pulse is indeed too small to be latched, two different cases are possible:

- The pulse does not cross the output inverter threshold: In this case the pulse will not be propagated to the output.
- The pulse does cross the output inverter threshold: The pulse will be forwarded to the output. The characteristics of the pulse will determine, whether the forwarded pulse will be amplified or attenuated.

The same process is performed for the output of the first stage by the second one and repeated until the pulse either becomes large enough to be latched, or is no longer registered (dies out). If the pulse is latched, the corresponding stage switches from the inverter-chain mode back to the expected micropipeline behavior.

As the simulations have shown, the acknowledgment signal is normally too slow to interfere with the pulse propagation process, as the critical pulses are quite short (see simulation results above) and are not latched. Only if the pulse is near the latching point, the decision whether the pulse is stored or not may take an extended time (metastability) and will be preempted by the acknowledge signal.

The difference to synchronous circuits is that flip flop chains do not have a combinational forward path. Every other latch is in storage mode and therefore has a decoupled input. The latches exchange this state with half the clock period (master transparent, slave opaque to master opaque, slave transparent or vice versa). Therefore the pulse is halted at these barriers until the next switching occurs and is therefore not free to propagate through the chain. In most cases this time is enough for the storage loop to capture the pulse or for the pulse to die out (temporal masking). As our analysis has shown, micropipelines in their empty state, however, do not have any temporal masking ability as they behave like inverter chains.

Figure 5.5a shows an example of the critical window for the injected charge to enable SET propagation[2] to the output (q_i) of a certain pipeline stage before the pulse is latched or extinct.

Please note that, for a pulse to propagate beyond a certain stage, the equivalent charge is always below the latching point of all previous stages. Otherwise the pulse would already have been converted into an SEU in one of them.

To better visualize the pipeline stage dependence of the window, we have plotted the critical window *size* for some interesting cases (Figure 5.5b) in semilogarithmic scale. The window size required for an SET to travel to a certain stage decreases exponentially with the depth of the stage in the pipeline. The window size for the first stage is nearly the same for all cases, while the decrease of the window size for deeper stages heavily depends on the threshold of the output inverter.

The window sizes observed are quite small and therefore the probability of seeing SET propagation in a pipeline is not high. However, the eagerness of an empty pipeline to accept a new pulse definitely introduces the possibility of SET propagation (and not only SEU propagation as usually assumed).

The dependence between the threshold of the output inverter and the pulse propagation behavior in the pipeline is of vital importance, as the mismatch between the threshold of the storage

[2] In the $1-0-1$ case, a pulse is present, if the output voltage is below $0.8 \cdot VDD$ for longer than $25ps$.

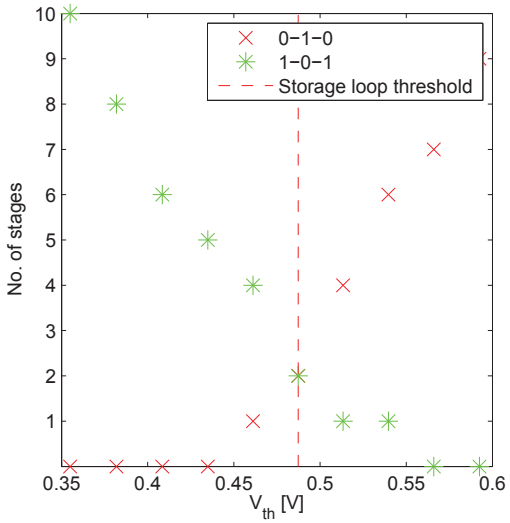
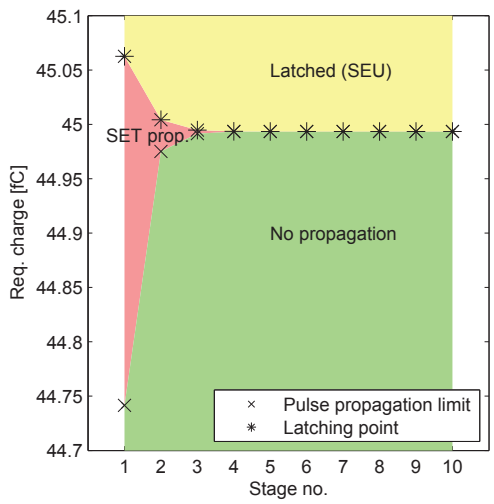
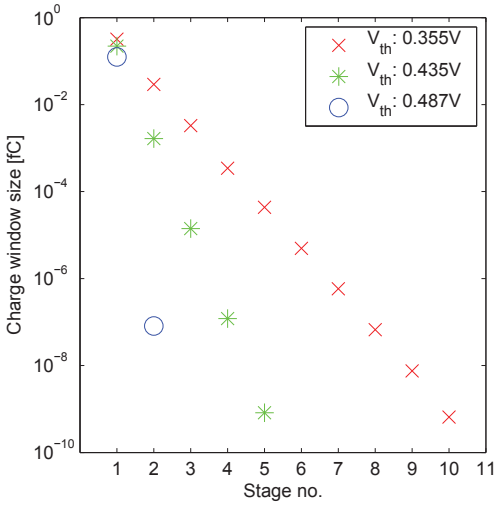


Figure 5.4: Number of stages seeing SET propagation



(a) Latch point and lower limit ($V_{th} = 0.355V$)



(b) Window size

Figure 5.5: Charge window for SET propagation (1-0-1 pulses)

loop and the output inverter facilitates the pulse propagation. Figure 5.4 shows the number of stages a pulse was able to propagate in dependence of the output inverter's threshold.

The figure shows the values for both cases ($0-1-0$[3] and $1-0-1$ pulses) and shows that, the bigger the mismatch between the storage loop and the output inverter threshold is, the farther pulses may propagate. This figure was created with a very precise injected charge value (accuracy in the order of $10^{-25}C$). Therefore, in real world circuits, the propagation may not be as intense but the figure gives a good overview on the dependence between threshold mismatch and pulse propagation.

It can be further seen that it is a unidirectional problem. If the threshold of the output inverter is sufficiently higher than the one of the storage loop, no more $0-1-0$ pulse propagation is possible. However, as both pulse directions are present, this property can not be exploited directly.

Based on this observation, we concluded that a Schmitt-Trigger may effectively eliminate the propagation of intermediate pulses. In contrast to other works, we do not use the Schmitt-Trigger in the storage loop but as output stage.

Further simulations have confirmed that this is indeed the case. We again simulated the circuit to find the upper and lower limit for SET propagation. Even with the high charge accuracy used we were unable to generate any SET causing an unlatched pulse at the output of the first Muller C-element. Figure 5.6 summarizes the results for the Schmitt-Trigger output stage. The left column shows the behavior for $1-0-1$ pulses, while the right column shows it for $0-1-0$ pulses. The upper and lower traces in the figure correspond to SETs with critical charge. Even though the window size is smaller than $10^{-25}fC$, the lower limit does not have any effect on the output of the first stage (q_1), while the upper limit already latches the SET safely in the first stage. Based on this result we can conclude that a Schmitt-Trigger has the potential to limit the SET propagation in micropipelines.

The implementation overhead, however, is significant. The Schmitt-Trigger output stage needs 4 additional transistors (compared to the normal inverter output). This is an overhead of 25% for a pipeline stage if the van-Berkel implementation is used, while the overhead for the other implementations is even higher. Furthermore the performance in the SET free case is also decreased. The nominal input to output delay of our van-Berkel implementation is $32.9ps$. This delay is increased to $70.9ps$, if a Schmitt-Trigger output stage is used ($+115.5\%$).

To summarize our evaluation, we can conclude that the propagation probability highly depends on the threshold differences between the storage loop and the output element. The more these are apart from each other, the higher the probability of SET propagation is. The only way to prevent SET propagation in the first place is to use adaptive thresholds in the output elements. Therefore the output of the pipeline stage only updates for pulses that are safely captured by the storage loop as well.

[3] In the $0-1-0$ case, a pulse is present, if the output voltage exceeds $0.2 \cdot VDD$ for longer than $25ps$.

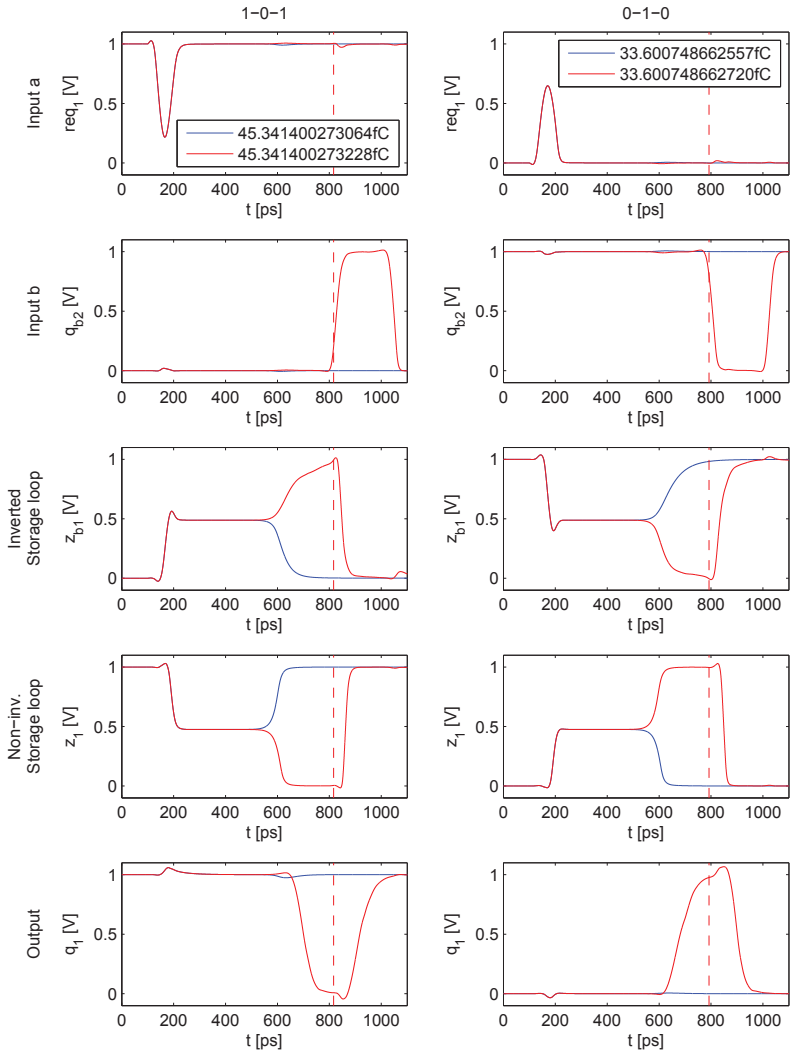


Figure 5.6: No SET propagation with Schmitt-Trigger output stage

5.5 Summary

Our simulations have shown the potential of SET propagation through several pipeline stages before the pulse was latched or extinct. The micropipeline indeed shows the expected mitigation behavior against short pulses (exponential decrease of critical charge window size with depth of the pipeline) but the effect strongly depends on its implementation. As the results show, an important factor is the output stage of the C-elements. If an inverter with non-matched threshold (to the storage loop) is used, the probability of SET propagation is increased. The overall probability, however, is quite small as the critical window for the deposited charge is narrow.

The crucial difference to synchronous circuits is the lack of barriers in the chain. While FF chains always have latches in storage mode within the chain (every other latch in fact), which enable its temporal masking capability, an empty micropipeline behaves like a simple inverter chain, lacking any temporal masking.

With additional simulations we could show the potential of a Schmitt-Trigger to prohibit SET propagation and identify its adaptive threshold as the key element for this behavior. The Schmitt-Trigger, however, introduces a high implementation overhead.

6 Metastability Measurement

After analyzing the basic behavior of the storage elements and the propagation of metastability through a pipeline, we will now concentrate on characterizing and verifying the observed metastability behavior using measurements.

6.1 State of the Art

All measurements circuits are, naturally, built around a device under test (or short DUT). The two major parts of the measurement electronics are:

- A mechanism to create metastable upsets within the DUT.
- A mechanism to evaluate some characteristics (length, count or shape) of the metastable upsets.

6.1.1 Metastability Generation

Metastability generation involves producing a pair of critical transitions at the input of the storage element. For a D-flip flop, e.g., this would require to create a data transition in the critical time window around the active clock edge. This can be done either randomly or deterministically.

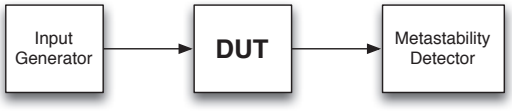


Figure 6.1: Basic metastability detection circuit concept

In the *random approach* two free running, independent oscillators are employed for creating the input signals of the DUT. Based on their independence, the phase difference of these two signals is uniformly distributed over their period [SG03]. Therefore from time to time a transition pair violating the critical window will occur and a metastable state of the DUT can be observed. The uniform phase distribution perfectly matches the assumption made for Equation (2.1) (and often the situation anticipated in the application as well), which makes this method very attractive for metastability characterization. Its drawback is that, since the critical window is typically considerably smaller than the period of the input signals, chances of producing a metastable upset are low, which causes the need for increased measurement times.

In the *deterministic approach* the phase shift between the input signals is carefully controlled to produce a maximum number of metastable upsets. Due to the influence of jitter, noise, voltage and parameter variations, etc. this is a very delicate task that requires special provisions like e.g. using a delay locked loop [KHR06, ZKD+08].

Based on the current upset rate, the phase difference is either increased or decreased such that the upset rate is increased. This process must be executed very carefully, as the required time difference, which is directly related to the phase shift, must be in the sub-atto-seconds-area. It is impossible to create such precise, stable phase shifts. Therefore, again, statistics are used. Thereby the mean value of the phase distribution is changed. Assuming a normally distributed phase shift, most of them will be in the interesting region.

The obvious advantage of this method is to generate a relatively high yield of (even deep) metastable upsets, which allows their detailed study. The correlation between the input signal transitions established by the phase control, however, rules out a characterization based on the application of Equation (2.1).

6.1.2 Metastability Detection

Again two basic methods exist here. For the *oscilloscope based approach* [CM73, SG03, KHR06, RSS+10] the DUT output signal is routed to a pin, using a (preferably analogue) output buffer. The signal is recorded by an oscilloscope and analyzed in the analogue domain. While this method allows the most detailed observation of the metastable behavior, it suffers from the probing effect caused by the extra load the buffer puts on the DUT output. Furthermore, since an oscilloscope does not allow seamless monitoring of the output signal, careful triggering is required to avoid missing the (deep) metastable events of interest in the myriad of less interesting ones.

The *late transition detection method* works in the digital domain. As outlined in Section 4.5 one common manifestation of metastability is the delay of the output transition, which is exploited in this approach. Figure 6.2 shows the principle: The output of the DUT is sampled (i) by a detector flip flop (DET)[1] after a well controlled delay $var\Delta$ that determines the resolution time t_{res} and (ii) by another flip flop (REF) that samples the DUT output after (ideally) infinite delay ($\Delta \rightarrow \infty$). Since the latter sample represents a reference for the correct, stable DUT output, a comparison of the two samples indicates whether the DUT output had already

[1] Note that in principle DET can also get metastable if the DUT output transition coincides with its critical window, which is why it is safer to append another stage.

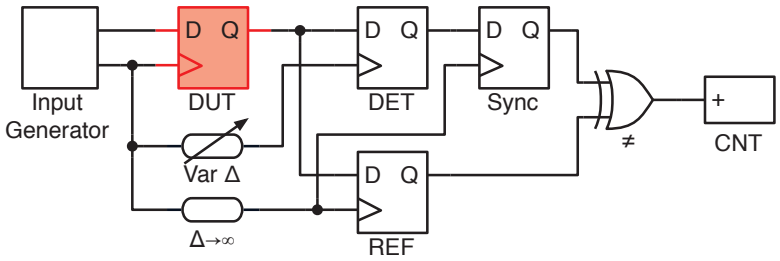


Figure 6.2: Digital metastability detection circuit

changed before being sampled by DET after $var\Delta$. Of course, infinite delay for REF is not feasible in practice; a full or a half period of the clock is often used instead, which is sufficient with a carefully chosen clock frequency. Finally a counter can be used to determine the number of mismatches experienced throughout a given measurement interval.

A variation seen sometimes is to delay the data output of the DUT instead of the detection clock signal [KJ04]. On the first glance this seems to be a good idea as it is easy to implement without requiring a non standard clock tree, but the extra inverters in the data path change the slope and shape of the signal and may thus influence the result. After all, the charm of the original late transition detection method is that the actual threshold of a circuit element from the same technology is used to classify the DUT response as being high or low, and not an artificial one as in the case of an oscilloscope measurement.

Another advantage of the late transition detection method is that *all* events are registered and that there is no blind time interval in the measurement period. However, it does neither allow a detailed study of individual upsets nor the direct measurement of signal shapes or actual resolution times. In fact one can only tell how often the actual resolution time has been larger than the one allowed by the choice of $var\Delta$. By varying $var\Delta$ for repeated runs of the experiment a cumulative plot of upsets over $var\Delta$ can be generated.

As this circuit is very compact and does, unlike the oscilloscope based approach, not require an analogue amplifier, it is often used in ASICs and FPGAs [KJ04, Alf05, ZKRY08, KW76]. A challenging problem, however, is the implementation of the variable delay $var\Delta$. Obviously the precision of this delay (in terms of accuracy and jitter) is crucial to the accuracy of the measured parameters. On an ASIC variable delays can be implemented in various ways, e.g. by analogue delay lines [RC82, BKD08], delay-locked loops [KHR06, ZKD$^+$08], tapped delay lines or starved inverters [RD93]. However, the integration of those functions into an ASIC with the required precision into an otherwise fully digital ASIC is still troublesome and costly.

To circumvent this problem, engineers have proposed alternatives like using the falling clock edge for DET, which allows a variation of t_{res} by virtue of the duty cycle.

6.1.3 Metastability Detection in FPGAs

In FPGAs the oscilloscope based method is infeasible, thus the late transition detection approach must be adopted. The central problem here is again the implementation of the variable delay, but

this time not even the above mentioned techniques can be applied. One reason is that analogue delay lines and starved inverters (or separate supply voltages for different LUTs, respectively) are simply not available in FPGAs. Another reason is the notorious uncertainty of routing delays within FPGAs. Not only are the delays of different paths within the FPGA subjected to PVT variations, also the place and route tools use heuristics to generate the routed design. Therefore after each synthesis run, the delay relationship of different paths may have changed, even if the primitives are locked to a certain place. This makes the application of tapped delay lines at best problematic, while in addition to that the fact that in FPGAs even simple inverters are mapped to a LUT with considerable propagation delay severely limits the attainable temporal resolution to the ns-range. Finally, solutions based on variable clock or duty cycle become ultimately problematic in the face of the unknown routing delays and the need to control the duty cycle with high precision as well.

Another solution used by [ZKRY08], is to take advantage of differences in routing delays by providing paths of different length between the clock generator and the clock input of the FF. The path selection is done by a multiplexer. However, since there is no dominant deterministic delay element involved any more, the uncertainty in the routing delay (concerning the PVT corners) is very large and the actual delay of the different routes is hard to predict. Moreover, useful delay differences are only achievable using manual routing, which render the process very work intense and impossible to automate.

Newer approaches have utilized the PLLs built into modern FPGAs to create the phase shifts. As PLLs contain delay locked loops, this approach is very promising. The resolution reported, however, was quite coarse [BGP+10].

6.2 Late Transition Detection for Flip Flops

The first step towards a metastability characterization circuit for all introduced storage elements was to extend the well known late transition measurement scheme. Our aim was to devise a setup for metastability characterization of an FPGA that does not require sophisticated external equipment (just a PC for the data processing), but still attains good accuracy for all relevant parameters and effects within reasonable measurement time. Furthermore, as we have seen different metastability behavior depending on the current storage element state, our circuit must be able to facilitate a state dependent analysis of the metastable state.

6.2.1 Overall Design Concept

The given boundary conditions determined the choice of the methods: For metastability generation we use random clock generation, as we do not want do depend on a two channel clock generator with precise phase alignment (to be useful for the purpose the phase resolution must be in the sub-ps range). To save the need for a high bandwidth oscilloscope along with the availability of analogue output pins on the target, we decided for late transition detection (LTD). Based on these decisions it is possible to host the complete infrastructure on chip; this has been shown in the literature already. There is still a lot of conceptual work and tuning required to

attain sufficient accuracy and observe all effects of interest. Before going into detail with our respective circuit design, we will briefly present the cornerstones of our approach:

Consideration of Slave Metastability

State of the art FPGA measurement circuits normally measure the response of the master latch only [WMZ+09, BGP+10]. In modern flip flops, however, τ of the slave latch may be significantly worse than the one of the master latch [JYG09]. If synchronizers are designed based on the master τ only, the calculated MTBF may be too optimistic. Therefore we consider the measurement of the slave latch of vital importance. The only approach we know of, that characterizes the slave latch for FPGAs is [CSC+10]. However, in this approach, the latches are characterized using a simulation model and later it is tried to correlate these result to the real hardware using an LTD to measure the metastable response of the DUT's master latch. Our approach, in contrast, directly measures the slave's metastable response in hardware.

For an efficient measurement of slave metastability we need to vary the resolution time of the master latch towards very low values. This makes it less likely for a metastable master output to resolve before being captured by the slave upon the falling clock edge. When using a clock with 50% duty cycle, this might be achieved by increasing the clock frequency. This, however, at the same time reduces the resolution time for the reference signal as well as requires the measurement electronics to work faster. As in most cases the speed of the measurement circuit is already at its limits, this is not an option. Instead we propose a scheme to create shorter clock pulses out of ordinary 50%-duty cycle clock signals, such that the clock period stays as long as necessary to avoid the above issues, while still attaining low and variable resolution time for the master latch. A respective pulse generation circuit that allows well controlled variation of the clock high-time will be detailed in Section 6.2.2.

Case Separation

CMOS circuits are known to exhibit significantly different timing properties for rising and falling edges, so one can expect the same for metastability parameters. Furthermore, one possible manifestation of metastability that occurs jointly with late transitions is the occurrence of glitches. The LTD schemes proposed so far do not account for these facts – they only record the overall metastable behavior without further distinction. The characterization obtained that way is sufficient for standard synchronization problems. Note, however, that knowledge about different parameter values for rising and falling edges may be beneficial, if a data event of interest is always associated with a certain polarity (like, e.g., an arriving interrupt), while the other edge is not relevant or becomes masked. In the same way knowledge about the existence and polarity of glitches at a synchronizer output may be relevant, if the subsequent logic cannot properly handle those.

In fact it is relatively easy to implement additional bookkeeping that allows to distinguish the cases listed in Table 6.1. In addition to detecting a mismatch between detector and reference we just need to keep track of the current and the previous reference value. This enables a much more detailed analysis of the metastability behavior.

case	start from (ref_last)	resolved to (ref_cur)	possible metastable effect
overall	any	any	overall # of metastable upsets
$from\_0$	low	any	late rising edge, positive glitch
$from\_1$	high	any	late falling edge, negative glitch
to_0	any	low	late falling edge, negative glitch
to_1	any	high	late rising edge, positive glitch
0_to_1	low	high	late rising edge
1_to_0	high	low	late falling edge
0_to_0	low	low	positive glitch
1_to_1	high	high	negative glitch

Table 6.1: Metastable cases that can be distinguished

For a better understanding of Table 6.1 consider the case 0_to_0. If the current as well as the previous reference value indicate a low, while the detector flip flop indicates that the DUT (at the sampling point given by the resolution time) issued a high, this means the DUT output has exhibited a positive glitch ($0 \rightarrow 1 \rightarrow 0$). Such a glitch may well be seen in reality, if the output buffer (inverter) of the DUT has a high threshold (similarly, for low input threshold a negative glitch will be seen at the inverter output, recall Section 4.4.1). In the conventional setup, when scanning with different resolution times, such a glitch will first be regarded as correct result and then, as resolution time is increased, as late transition and for even higher resolution times as correct again, thus distorting the results (see Section 6.2.3).

Calibration Runs

The accuracy of the measurement is, among other factors, limited by the accuracy of the variable delay elements as well as by the delays along some selected routing paths. Unfortunately interconnect delays of FPGAs are specifically pronounced and hard to control or determine. To tackle this problem we propose a calibration run in which we make the DUT behave in a well defined way (i.e. with no metastability in the game) and calibrate our on-chip measurement infrastructure so as to properly reflect this behavior. For details see Section 6.2.2.

Averaging over Results

Since the time resolution we are aiming at is right at the limit of what can be attained with the available infrastructure, we have to consider several sources of error, partly of systematic and partly of statistical nature[2]. We will take care of some systematic errors in Section 6.2.4. For statistical errors averaging is known to be an effective remedy. Since the required measurement circuit is typically much smaller than what the modern target FPGAs can accommodate, we

[2]We consider an effect systematic, if the change it causes on the results can be reproduced in subsequent measurements, and statistical, if the changes caused by it significantly vary throughout multiple measurements.

propose to replicate circuitry, such that multiple measurements can be performed in parallel and thus without a penalty on measurement time.

6.2.2 Measurement Circuit

In the this section we will discuss the components constituting our setup in more detail. Please note that the circuits shown subsequently are simplified to the basic concepts, while details like non essential synchronizers, pipeline flip flops to increase the achievable clock frequency and special purpose buffers (like clock buffers) are omitted to improve readability.

Delay Generation

Before discussing the basic blocks of our setup, we will first focus on a function block that is used in several of them, namely a delay element. As already outlined in Section 6.1.3 it is not trivial to implement a sufficiently accurate adjustable delay within an FPGA. Our solution relies on delay locked loops (DLLs), which are readily available on most modern FPGAs. They accept a reference clock at their input and supply an output signal that is delayed against that input by an adjustable phase angle. The adjustment is possible in steps, under the control of increment and decrement ports. The digital clock manager (DCM) of our Xilinx Virtex-4 target FPGA provides such a function, with a dynamic range of two clock cycles and a resolution of 256 steps per cycle. With a reference clock of 400MHz this results in a step size of $9.77\ ps$.

Metastability Generation

In accordance with our decisions from Section 6.2.1 we use two unrelated clock sources for generating data and clock inputs of the DUT as shown in Figure 6.3. The clock input is not directly connected but through a pulse generation unit that controls the duty cycle. This implements our requirement from Section 6.2.1 aiming at efficient measurement of slave metastability. As visible in Figure 6.3 (block *clock pulse generation*) we perform a logical AND of the original clock with a delayed version (using the DCM). This solution allows us to adjust the high-time of the clock in steps of $9.77\ ps$. For investigating slave metastability we need to determine the minimum clock pulse width that still allows safe operation of the measurement circuit. As we do not know the involved delays (interconnect, AND gate), the best option is to perform a calibration measurement. To this end we add a counter to the clock generation component that counts the clock pulses issued by the AND gate. By periodically reading the count value, a host PC can check whether the pulses are long enough to facilitate proper counting; we further call this a "heartbeat" of the counter. The calibration procedure works as follows

- Reset the circuit and set phase shift to zero.
- Decrease phase shift until heartbeat stops.
- Increase phase shift until heartbeat is detected again.
- The minimum pulse length for measurements is found.

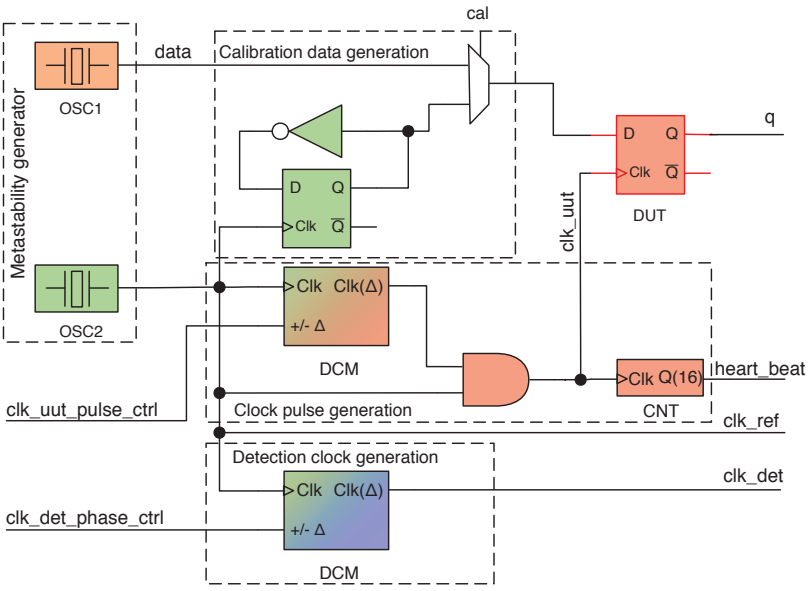


Figure 6.3: Metastability and clock generation circuit

- For clean measurements a safety margin of ten steps should be added (roughly $100\ ps$).

Note that when starting with zero delay "decrease" actually means applying *negative* delay (which the DCM can easily accommodate; it is just a phase alignment), so the delayed edges occur before the reference ones. Therefore the rising edge of the reference always determines the start of the pulse and hence the reference and the DUT clock stay in phase sync.

Late Transition Detection

We use the standard LTD detector found in numerous publications (e.g. [RC82, WMZ+09, ZKRY08]). It is shown in Figure 6.4. The output of the DUT is fed into two flip flops, namely the detector flip flop and the reference flip flop. The reference flip flop has a whole clock period available for resolving metastability. The clock of the detector is shifted, such that its resolution time is variable. Please note the additional flip flop on the detection rail. It is necessary to re-synchronize the detection data to the reference clock. As the rising edge of the detection clock happens before the reference clock, the data at the XOR gate would otherwise be out of sync.

The block labeled detection clock generation in Figure 6.3 shows how the digital clock manager (DCM) is used to generate the variable delay. This is basically the same solution as used in [BGP+10] but with a much smaller step size. We have introduced a calibration mechanism for the clock phase alignment between detection clock and reference clock, i.e. for the variable delay, since again unknown interconnect delays are involved. Our calibration algorithm works as follows:

- Reset the circuit.

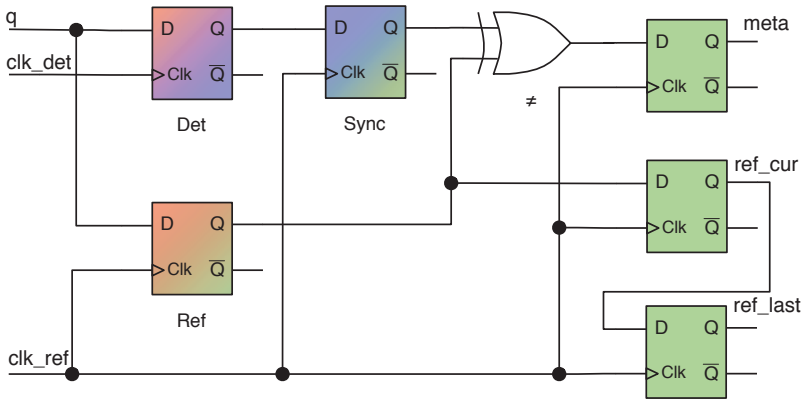


Figure 6.4: Adapted LTD circuit

- Switch the DUT to synchronous operation, using a toggle flip flop driven by the reference clock (rather than the uncorrelated data source), as shown in Figure 6.3, block *calibration data generation*. In this mode of operation we can be sure the DUT will operate metastability-free.

- Decrease the variable delay until a mismatch at the XOR gate is seen. Since we can be sure that no late transitions can occur, this mismatch must be due to sampling the DUT output slightly before its nominal clock-to-output delay.

- Increase the phase shift again until no more upsets are detected. Now sampling occurs right after the nominal clock-to-output delay, leaving zero resolution time for metastable upsets (corresponding to the definition of T_W from [Kin07]).

- The calibration value for the detection clock phase shift has been found. We can now increment the resolution time, in our implementation in steps of $9.77\ ps$.

Detection Counters

We have slightly modified the LTD by introducing two additional flip flops (Figure 6.4 right). These maintain the reference values of the current and the previous cycle, such that the counter has a consistent triple available, comprising the following information: $meta$: mismatch between detector and reference detected; ref_cur: associated reference value; ref_last: previous reference value, which is the starting value of the current period. Note that it is important to have this information available in a *consistent* manner and evaluate the complete triple, rather than just counting events on all outputs. Therefore we need to provide a counter for each of the cases listed in Table 6.1 that is incremented every time a triple corresponding to its respective case is seen.

Controller

The on-chip controller is a combination of a simple state machine and a register bank. The register bank can be accessed by the host PC to set up measurements, calibrate the circuit and read back measurement results (counter values).

The state machine allows to precisely control the duration of a measurement: The host PC sets up the number of clock cycles the measurement shall last and sends the start signal. The controller counts the elapsed number of cycles and stops the measurement accordingly. The controller works with a slower clock than the measurement and uses synchronizers for the control and status signals. The result counters need not to be synchronized as they are only read after the measurement was stopped (no more counter activity).

6.2.3 Validation Results

To validate our concept we have implemented it on a Xilinx Virtex-4 FPGA. As already mentioned we can achieve a time resolution of $9.77\ ps$ for our delay elements there, using as a reference a $100\ MHz$ on-board crystal clock that we multiplied by four with the DCM. Figure 6.5 shows the results we obtained for rising edges. The figure represents an overlay of measurements with six different pulse widths – ranging from minimal pulse width $+97.66\ ps$ to $+781.25\ ps$ – for the clock (shown in different colors), therefore the slave metastability becomes effective at different points (lower right ends of the curves). The dashed lines represent the straight lines that best approximate the measured curves and hence allow to estimate the value of τ, which is in the relatively narrow range of $87 - 92\ ps$. This confirms that our approach for measuring slave metastability works.

The pronounced slope left of the slave metastability is characterized by the master metastability. Here the respective slopes (solid lines) are in the range of $46 - 48\ ps$. This matches the value published by Xilinx [Alf08] (resulting in $\tau = 41ps$) quite well. At the same time it can be clearly seen that the τ of the master is much better than that of the slave, nearly by a factor of two. So it is definitely optimistic to rely on the τ of the master latch alone (which is much easier to measure).

One can also observe that our setup allows to cover a considerable range of resolution times with very small step size, yielding a dense curve. The resolution time could be further enlarged without problems, but at the cost of considerable measurement time (see Section 6.2.4).

Plots of the raw measurement data have shown that the delay steps of the DCM are not completely uniform; there seems to be a larger step after a period of smaller ones. To minimize the effects of this behavior, we used three parallel detectors for the same flip flop for the measurement. Since the routing delays of the detectors were slightly different, the effective resolution times ultimately differed. This delay mismatch was too small to be calibrated, so we simply aligned the measured curves. Consequently the large steps ended up at different positions on the time axis. Therefore calculating the mean of the logarithm of the data points not only removed the statistical effects (like clock jitter), but also the DCM step mismatches, yielding the relatively smooth curves shown in the figure. These curves are reproducible, even for different routing.

The equivalent results for the falling edges are shown in Figure 6.6. Here the τ for the master is in the range of $39 - 42\ ps$, which is clearly better than for the rising edge but in the same order.

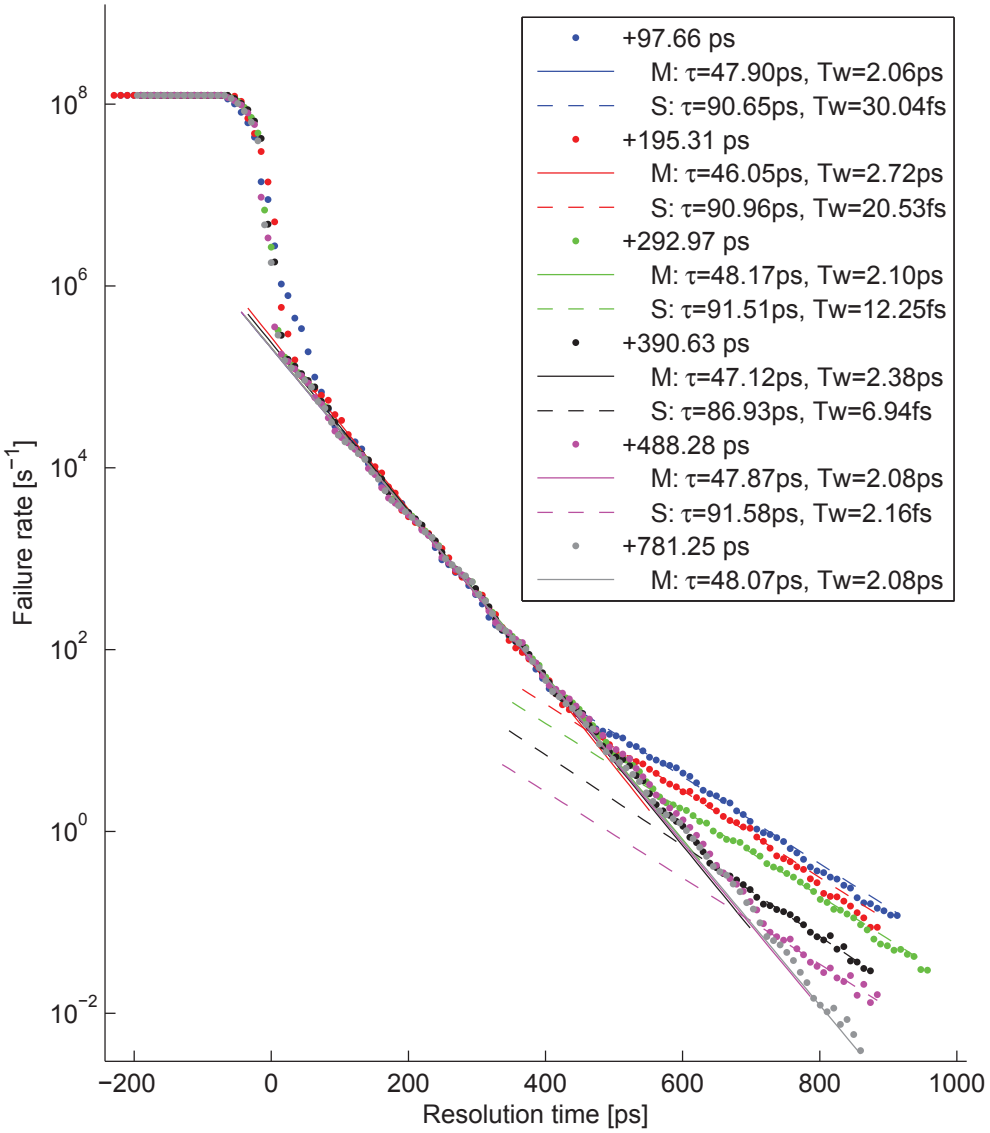


Figure 6.5: Measurement result for rising edges with different clock pulse widths

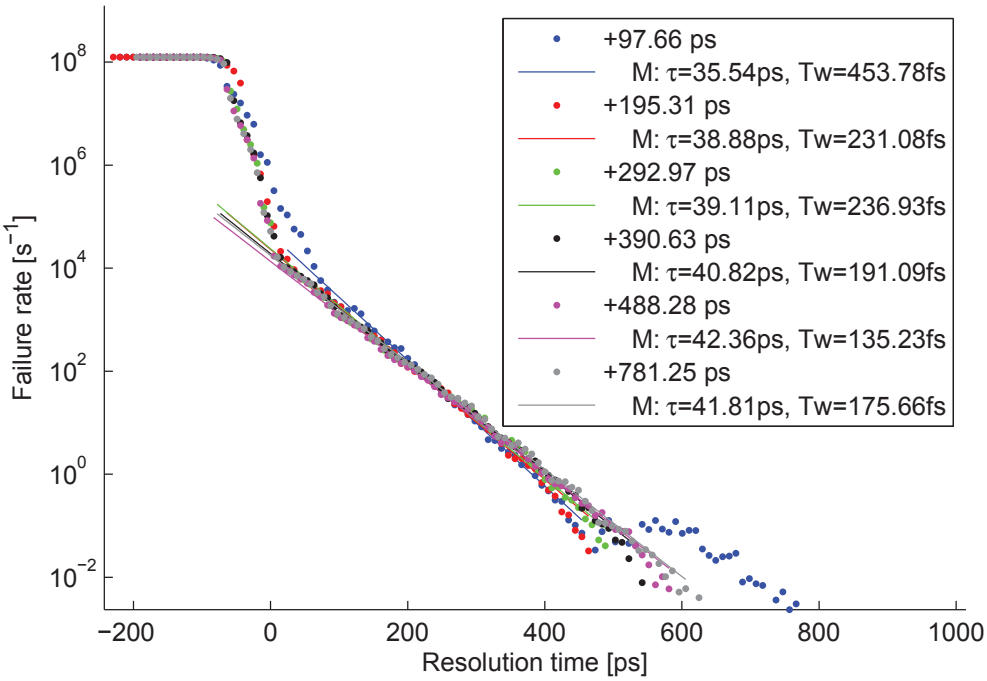


Figure 6.6: Measurement result for falling edges with different clock pulse widths

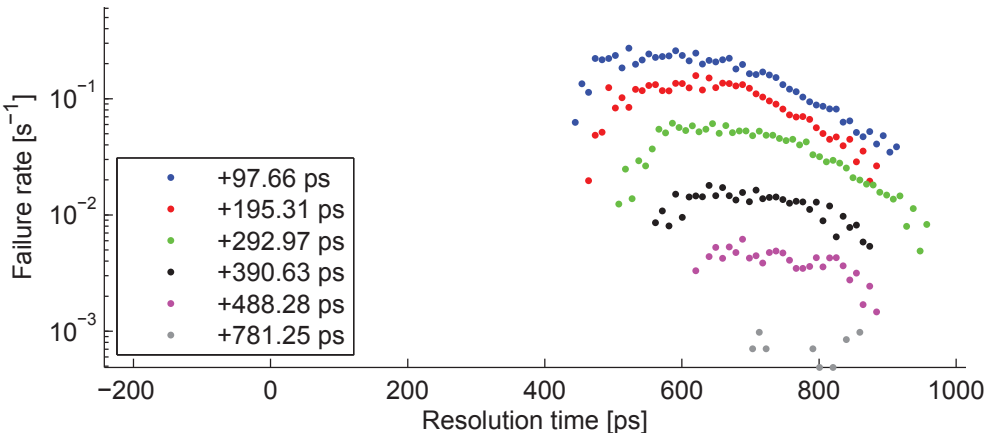


Figure 6.7: Measurement result for $1 \rightarrow 0 \rightarrow 1$ pulses with different clock pulse widths

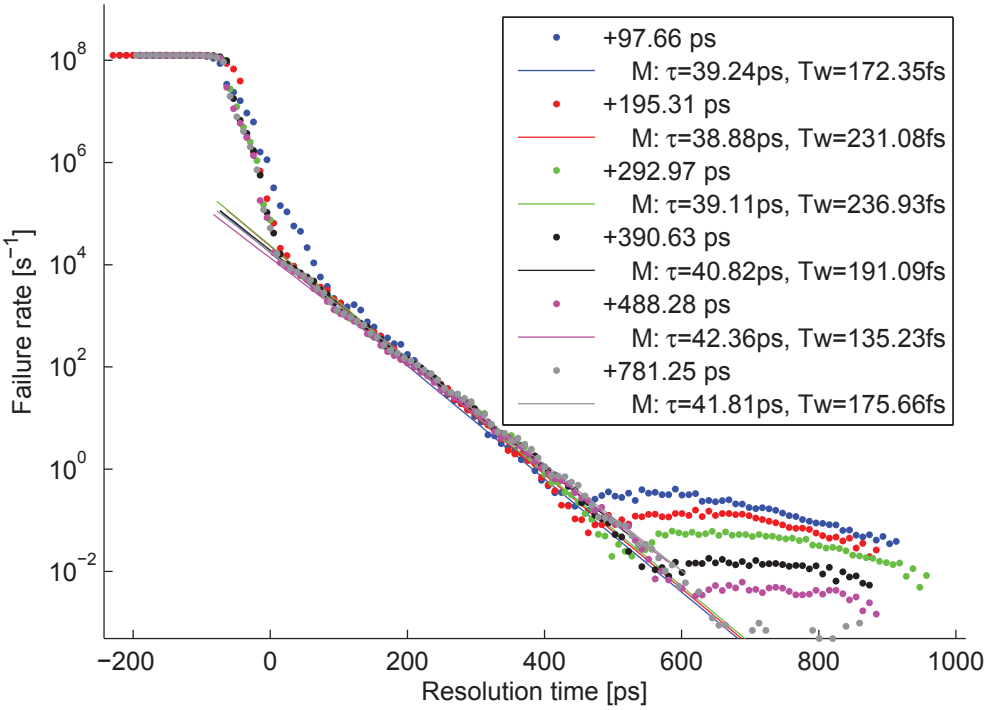


Figure 6.8: Measurement result for cases starting from 1

For the slave metastability we cannot make a useful observation here. This is because the output inverter has a low threshold, and so, if the slave gets metastable on a falling edge (i.e. the inverter input is rising), it will immediately cross that threshold just by going metastable, and no late transition will ever be observed. For falling edges the probability for metastable upsets is greatly decreased when the slave gets metastable. Unfortunately this is not a sign of a better resolution capability but the output flips prematurely to the new value and these upsets are masked. This becomes clearly visible when comparing to the $1 \to 0 \to 1$ pulses in Figure 6.7. For resolution times where we had observed slave metastability on the rising edges, now pulses start to appear. As with the slave metastability before, shorter clock pulse widths increase their probability and move their peak as well as their point of first occurrence towards higher resolution times. These graphs indicate the low threshold of the output inverter: When its input goes from low (i.e. output high) to the metastable state, the threshold is already crossed, causing a low output, but then when metastability resolves back to low, the threshold is crossed once again, bringing the output back to high. With a similar argument we can explain why we do not see any $0 \to 1 \to 0$ pulses for our DUT.

Figure 6.8 shows the plot for all metastable occurrences starting from high, which is the sum of the occurrences from Figures 6.6 and 6.7. Without the preceding analysis one would be tempted here to misinterpret the rightmost parts of the graphs as late transitions due to slave metastability.

While for the determination of τ only differences of delays are relevant, T_W requires their absolute values. These are difficult to determine or control, and in fact we are not aware of any paper detailing an LTD scheme for accurately measuring T_W. Our calibration at least represents a step in that direction. It is, however, limited by the attainable step size which turned out way too coarse for a reliable measurement. So the values of T_W given in the figures must be seen as rough estimates.

6.2.4 Constraints

When building a circuit that is as time critical as the late transition detection, one must adhere to numerous constraints and add annotations to make the tool chain produce a suitable implementation. The most important ones for the LTD are described in this section.

Critical paths

The clock domain crossings in the LTD design are of vital importance to the correctness of the circuit. Unfavorable delays decrease the available resolution time for the reference and can have a negative effect on the dynamic measurement range as well.

The available resolution time for the reference is:

$$t_{res}^{REF} = \frac{1}{f_{clk}} - \Delta_C^{REF,DUT} - \delta^{DUT,REF} - t_{co_{nom}}^{DUT} - t_{su}^{REF}$$

$\Delta_C^{REF,DUT}$ is the clock skew between detection and reference clock, $\delta^{DUT,REF}$ the data path delay between the two, $t_{co_{nom}}^{DUT}$ the nominal clock to output time of the DUT and t_{su}^{REF} the setup time of the reference. All these values shorten the available resolution time of $\frac{1}{f_{clk}}$. The only two

parameters controllable by the designer are $\Delta_C^{REF,DUT}$ and $\delta^{DUT,REF}$. We tried to minimize $\Delta_C^{REF,DUT}$ by matching the paths through the clock pulse generation circuit. Additionally we manually placed the DUT and the reference to also minimize $\delta^{DUT,REF}$. The penalty of a short resolution time for the reference is an increased probability that its value will not be resolved within the available time, yielding an incorrect reference.

For the detector (DET) the situation is somewhat more complex. As the detector must read the value from the DUT (at different times) and must also safely deliver its measured value to the reference clock domain (via the sync FF), a second set of constraints is required:

$$t_{res}^{DET} = \frac{1}{f_{clk}} - \Delta_C^{DET,DUT} - \delta^{DUT,DET} - t_{co_{nom}}^{DUT} - t_{su}^{DET} - \\ - \Delta_C^{sync,DET} - \delta^{DET,sync} - t_{co_{nom}}^{DET} - t_{su}^{sync}$$

Therefore the maximum measurable resolution time t_{res}^{DET} is, on one hand, constrained by the path from the DUT to the Detector (clock skew $\Delta_C^{DET,DUT}$, routing delay $\delta^{DUT,DET}$ as well as setup- and clock-to-output time of the corresponding flip flops. On the other hand, the delivery of the detected value to the reference clock domain requires the clock skew between them ($\Delta_C^{sync,DET}$), the routing delay between the detector and the sync FF ($\delta^{DET,sync}$ as well as the corresponding setup- and clock-to-output delays again. Again we minimized the penalty of the data path delay by manually placing the flip flops. In addition, we introduced some pipeline stages to simplify the placing of the critical flip flops. To compensate for the clock skew we introduced the calibration mode. Therefore the skew between the two clocks can be bounded and is of little consequence to the measurements.

Nominal output delay

To be able to calculate T_W, the nominal output delay of the DUT is required. In principle, the calibration algorithm proposed in this paper can achieve this. A fundamental limitation here is, however, that the nominal output delay of the flip flop is not a single value but has a distribution, so finding its maximum is not possible within finite calibration time. A more practical issue was that the calibration granularity was too coarse.

Measurement considerations

There are many parameters to consider for achieving a fast and reliable measurement. The measurement time must be large enough to minimize the statistical errors, but still tractable. To achieve this we dynamically adapted the measurement duration. If for one resolution time the event count dropped below a certain limit (we used 500), we doubled the measurement period for the next (larger) resolution time. By implementing this regime, the region with high failure rates can be measured fast, while that with low failure rates still attains adequate quality. For the results presented in this paper, the measurement of the first $200ps$ of resolution time took approximately three minutes, while the last $200ps$ took 10.8 hours. As one would expect, the measurement time grows exponentially.

For a good resolution the length of a time step is essential. For the DCM, the number of achievable steps per clock period and the step size itself greatly depend on the input clock frequency. For a good time resolution, the clock frequency should be as high as possible. Therefore, in the proof of concept, we used the maximum frequency supported by the Virtex-4 DCM. The non-ideal differential linearity of the step sizes has already been addressed in Section 6.2.3.

Another potential source of uncertainty is the time interval measurement. We employed a counter on the target device to count the clock cycles within a measurement interval. As the time interval is now given in clock cycles, its duration in terms of time has no effect on the calculation of τ as it cancels out in the corresponding equation.

Finally, PVT variations of both, target and measurement circuits are a notorious source of uncertainty. Our approach does not solve this problem, but part of its charm is to allow a measurement of the concrete device at hand in its concrete environment rather than having to rely on abstract data sheet values that ultimately require substantial safety margins to safely cover the specific case.

6.3 Late Transition Detector for Muller C-Elements

Our aim is to adapt our LTD circuit for D-flip flops (see Section 6.2) to make it applicable for characterizing a Muller C-element.

6.3.1 Initial Circuit Design

The key problem is that, while in case of the flip flop the clock edge can be safely taken as a reference for the output delay, this is not so easy for the Muller C-element: The appropriate reference must be determined based on which input actually triggered the output change. As this is not possible on the fly, our solution is to measure both cases (using each of the inputs as a reference) in parallel and seed out the inappropriate one later on. For the latter we need to know the relative position of the input edges and the type of the output transition (rising/falling/glitch). Our circuit will therefore comprise (i) an LTD that uses input A as a reference (topmost part in Figure 6.9, (ii) an LTD that uses input B as a reference (middle of Figure 6.9, and (iii) an overlap detection circuit (bottom part in the figure) to determine the relative sequence. Note that in principle we could come along with the one LTD that uses B as a reference alone. In that case we would simply miss the cases that require A as a reference, but still get valid results. However, as we cannot be sure that the DUT behaves equally for these two scenarios, we want to capture the cases with reference A as well.

We focus on the case of opposing input edges (i.e. disregard the case of glitches on a single input), as this relieves us from generating glitches with high precision, and is more similar to the flip flop case. Therefore we will, like in the standard LTD, use independent clocks of approximately equal frequency for generating the inputs A and B. A key strategy will be to structure the period of each of these clocks into segments to allow for a better analysis of the different phases. That is why we generate the inputs A and B by dividing higher frequency clock signals clk_a and clk_b by eight. The blocks labeled by "$/8$" in Figure 6.9 are responsible for this clock division. A counter keeps track of the phase (further called state) of the inputs in both cases. For

both a rising edge occurs from state eight to state one, and a falling edge from state four to state five.

Critical Overlap Detection

Recall that metastability only occurs when the edges arrive in close proximity, so this is the only relevant case to consider. Unfortunately in this case the relative arrival time of the edges is most difficult to determine. In essence this would again require a mutex element to prevent metastability, but even then the mutex might decide for a certain sequence, while, due to routing asymmetries, the DUT (i.e. the Muller C-element) actually experiences a different sequence (we need precision in the sub-ps range). Moreover, the decision may take unbounded time [Kin07]. Therefore the only appropriate solution is to include the output behavior of the DUT in the decision. We will come back to this later.

For the moment our strategy will be to sample input A with clk_b (recall that we have approximately eight samples per period) and identify the position of the edges relative to those of B. The problem here is that in the interesting area of nearly coincident edges the sampling will also tend to deliver metastable results. However, this is only true for the sample right at the edge; one sample before the edge of A and one after we will get stable results. These are sufficient to identify whether a transition (and in which direction) has occurred within the interval of interest[3]. This is essentially what the overlap detector does, and the signals of interest are a_b_last, a_b and a_b_next (sequential series of samples of A), as well as $valid\_a\_01$ and $valid\_a\_10$, which are the outputs indicating the type of transition on A, if any (the remaining flip flops in this block are just responsible for an appropriate temporal alignment in terms of clock cycles). One might argue that for unfavorable phase relations of A and B the signals a_b_last and a_b_next might become metastable in the same way as a_b. While this is true in principle, it only occurs outside the window of interest, i.e., only when the transitions are sufficiently (namely one period of clk_b or more) separated in time such that the DUT will not get metastable anyway. In this way we have an overlap detector available that safely avoids metastability in the interesting window.

Figure 6.10 shows the temporal relation of all signals for the unfavorable case that the two input clocks are close to synchronous. As can be seen, the sample a_b is undetermined in this case, as it may become metastable. Nevertheless, the signals a_b_next and a_b_last are well defined, as intended. Furthermore, the construction of the circuit ensures that these two signals stay well defined for nearly one clock cycle shift between clk_a and clk_b in either direction and therefore are valid for all critical overlaps.

As already mentioned we will also use the phase state of the inputs to draw our conclusions (block “valid” in Figure 6.9). Here we will use the state of B to identify those cases where A and B overlap with the same polarity, spanning a short pulse (as opposed to those with different polarity), as only those trigger metastability (cases O1 ... O4 in Figure 4.3). This state information is temporally aligned through a chain of flip flops as well.

[3] Since the frequency of clk_b is about eight times that of A there cannot be more than one transition

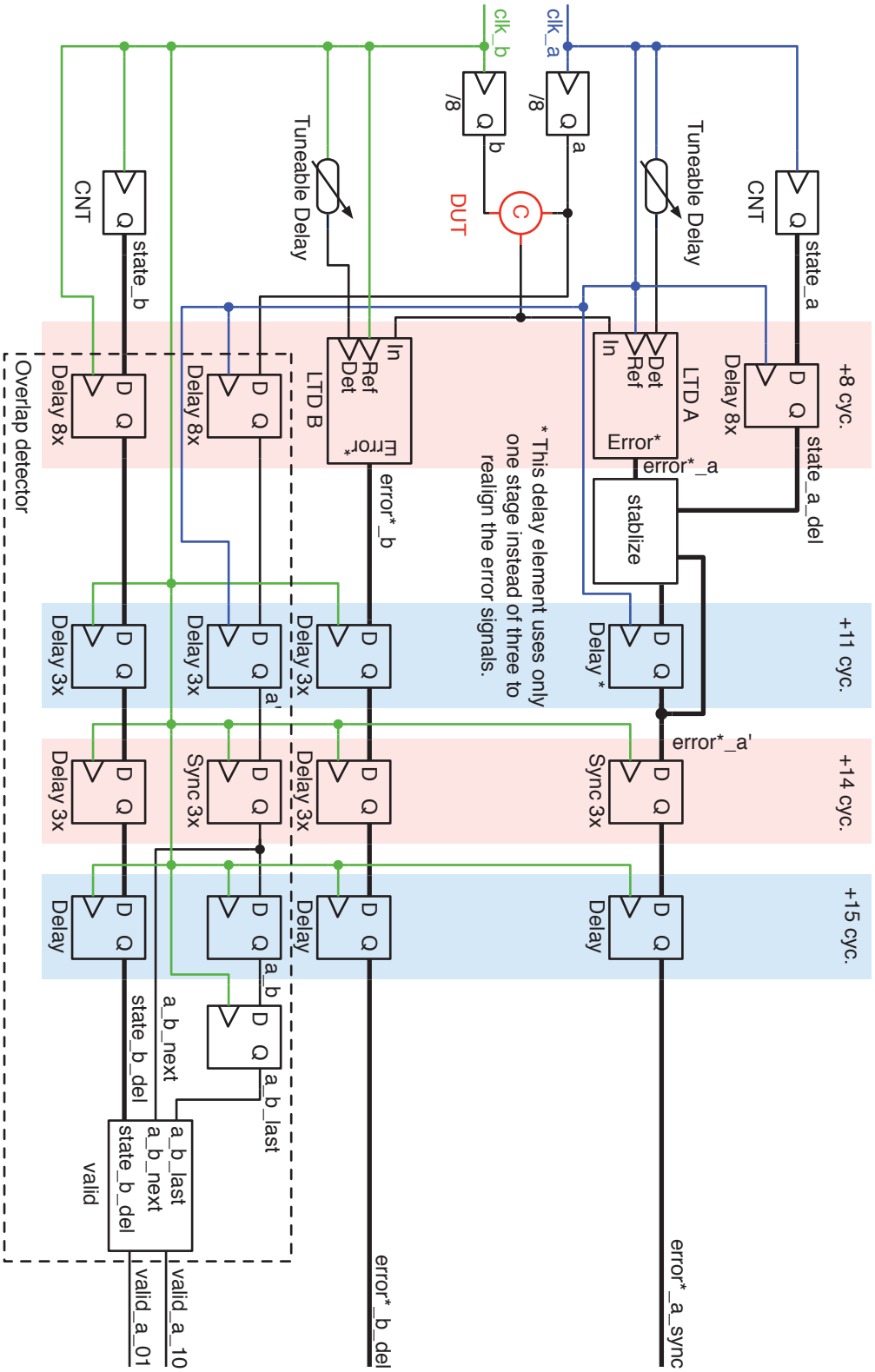


Figure 6.9: LTD circuit for Muller C-elements

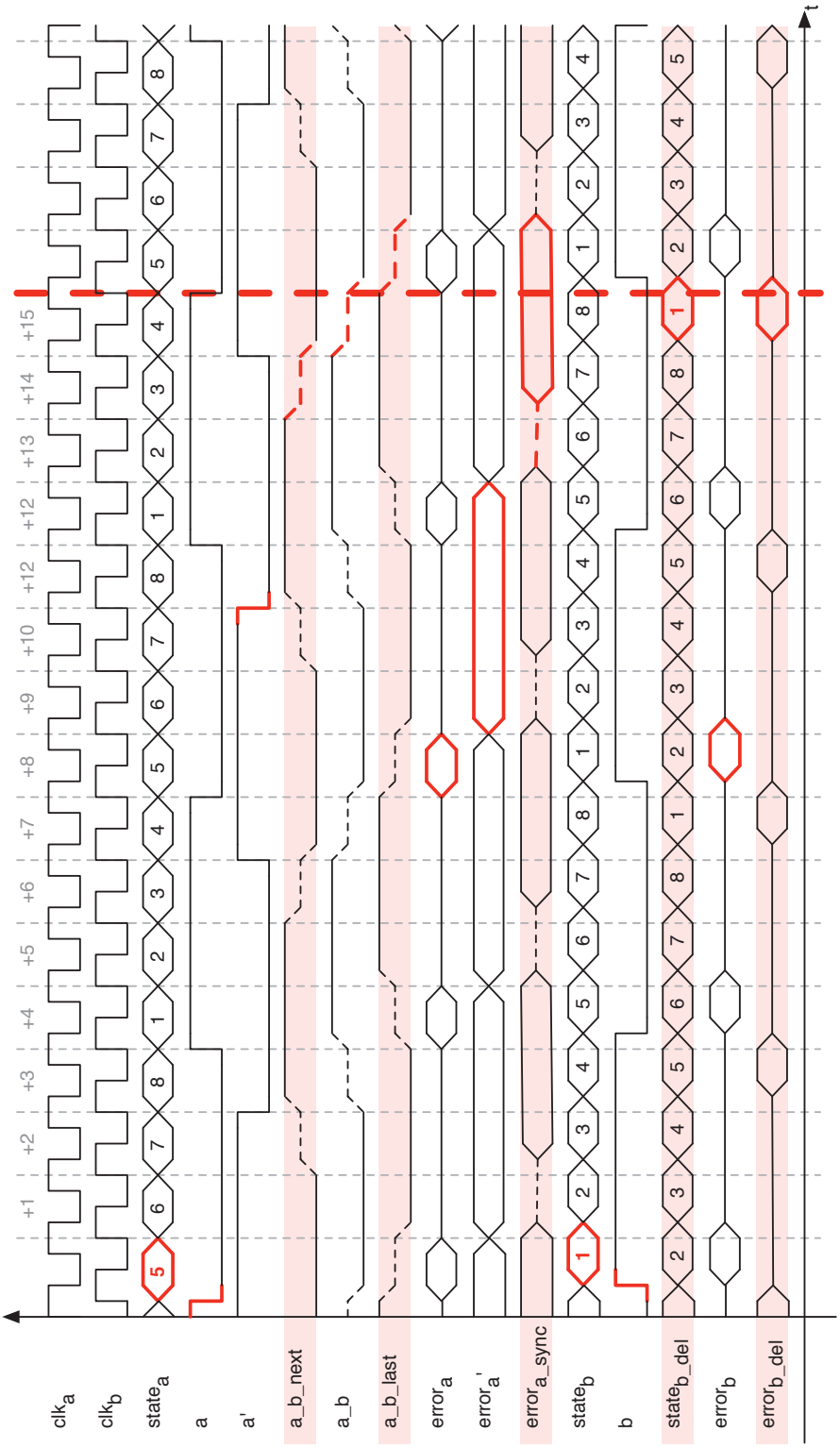


Figure 6.10: Timing diagram of LTD operation

Output Delay Measurement

As already mentioned, the LTD scheme is used to measure the output delay, and two instances are used, between which the decision is made later on. For this decision the standard LTD is not sufficient, the measured cases must rather be split according to different scenarios at the output, namely late rising edges, late falling edges, short $0 \rightarrow 1 \rightarrow 0$ and $1 \rightarrow 0 \rightarrow 1$ pulses. Consequently the output of the detector consists of four error signals ($error01$, $error10$, $error010$ and $error101$, respectively). For brevity, we will call all these signals $error*$ in the following. We already developed such an LTD scheme for D-flip flops providing this case distinction with only a couple of extra flip flops in Section 6.2.

LTD Result Synchronization

As the whole analysis is performed in the clock domain of clk_b, the results (i.e. the four error flags $error*$) of the LTD for clk_a must be safely transferred. Like before we do not want to use a brute force synchronizer (inferring unavoidable residual upset probability) here, but safely avoid metastability for the interesting cases. Here we take a two-step strategy: First we capture the flags within the domain of clk_a. Recall that the inputs A and B have been generated by dividing the respective clk_* signals by eight, and the transitions occur from state eight to one and from four to five. So we can safely latch the detector outputs $error*$ one cycle later (i.e., one/two and five/six, see Figure 6.10) and hold them constant for the next four cycles. This task is performed by the block "stabilize" in Figure 6.9.

Finally, however, we need the results in the domain of clk_b, which is our reference domain for all evaluations in general. As clk_a and clk_b are uncorrelated (recall that they need to be), we cannot avoid metastability in general. However, we know that the case of interest is when the edges of the input signals (A, B) are close to each other. So for exactly this situation we have extra information on the relative position of the signal transitions, and we can leverage this by sampling with $state\_b\_del$ transitions one/two and five/six (recall that A and B are aligned in the cases of interest). This is illustrated in Figure 6.10.

Again one may argue that there is still a potential for metastability for phases outside the window of interest, but these cases are masked out by the overlap detection circuit. These events, however, are all associated with delay values nearly identical to the nominal output delay of the DUT. So these measurements would only contribute to the area in the far left of the result diagram. Masking them out only decreases the constant part at the beginning but has no effect on the exponentially decaying area (see Figure 6.11 for a schematic illustration) which is the only relevant portion for determining the characteristic τ value.

Detector Selection

So far we have collected error signals with two LTDs in parallel, also including scenarios that do not lead to DUT metastability, and, even worse, may involve metastability in the detector. We have, however, taken care that all results of interest are correct, moving all undesired effects to the other phases. So at this point it is important to pick out the relevant results and discard all others. We have two types of information available for this purpose: (i) Our LTDs provide us

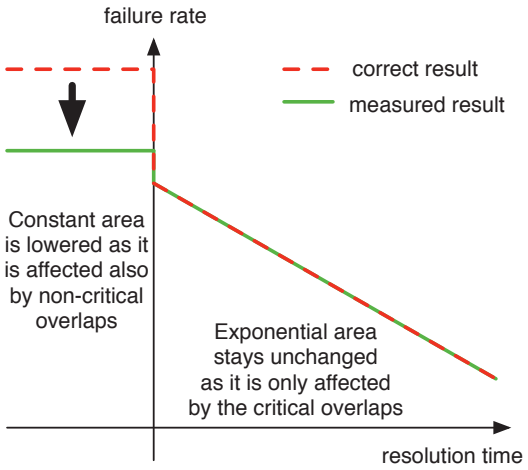


Figure 6.11: Effect of masking non-critical overlaps, schematic depiction

information not only on the occurrence of an upset but also on the type of transition (error flags $error*$), so we know the output behavior of the DUT in each single case. Concerning the inputs we get the required information from our overlap detection circuit. This allows us to exactly identify the cases of interest as shown in Figure 6.12 (Q_l stands for an output inverter with low threshold, Q_h for one with high threshold).

The selection exploits the fact that two error signals starting from a different reference value are mutually exclusive. As in each of the two groups in the table, the error signals for the two clock domains are based on different reference values ($error1x\_a$ vs. $error0x\_b$ and $error0x\_a$ vs. $error1x\_b$, respectively), a simple conjugation of the error signals with the valid signals selects the correct detector. Therefore the decision of A before B is indirectly solved by the Muller C-element itself, as intended.

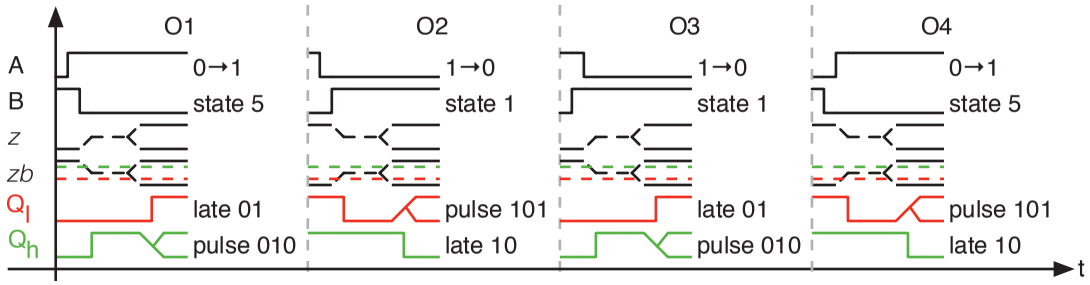
Based on the criteria from Figure 6.12 several counters can be controlled to accumulate the number of certain types of upsets for the currently selected resolution time. These values can than be used to create the failure rate vs. resolution time plot, as shown in the Section 6.3.4.

6.3.2 Prototype Implementation

To demonstrate the viability of our design, we created a prototype implementation within an FPGA. As measurement target we used a D-Latch which we converted into a Muller C-element by implementing the corresponding set- and reset-circuits (setting when $A = B = 1$, reseting when $A = B = 0$, see Figure 6.13 for details). The figure furthermore shows that we have chosen two different implementations for the Muller C-element.

An advantage of using the FPGA prototype is that we have already measured the τ values for the D-flip flops of this FPGA series (in fact the τ of the master latch must be used) in Section 6.2. Therefore we are able to compare the results of the Muller C-element with them.

The two delay lines for generating the detection delay can not be implemented directly in an FPGA fabric, as no analogue components are available and the resolution of inverter chains

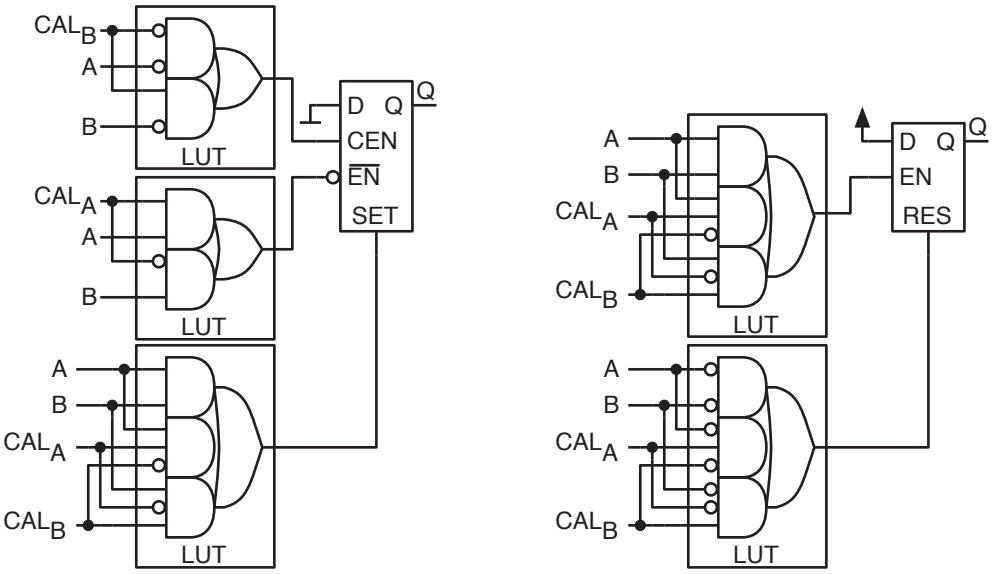


(a) Error cases

input signals			calc. valid signals		result			
state_b_del	*a_b_last*	*a_b_next*	*valid_a_01*	*valid_a_10*	transitions	possible overlaps	valid error signals	case
1	1	0	0	1	falling on A and rising on B	low overlap, A first	$error10\_a$, $error101\_a$	O2
						high overlap, B first	$error01\_b$, $error010\_b$	O3
5	0	1	1	0	rising on A and falling on B	high overlap, A first	$error01\_a$, $error010\_a$	O1
						low overlap, B first	$error10\_b$, $error101\_b$	O4

(b) Matrix

Figure 6.12: Detector selection



(a) First implementation (b) Second implementation

Figure 6.13: Muller C-element FPGA implementation

would be much too coarse. Therefore we again utilized the clock managers of the FPGA, more precisely their built-in delay locked loops (DLLs). With an input frequency of 250 MHz, a step size for the delay of 15.6 ps has been achieved.

As already mentioned in the previous section, the delays between the DLLs and the detection flip flops may vary significantly. Therefore a calibration circuit to compensate for these delays was required. In fact, the importance of calibration is even higher in this case than for the solution with the flip flop target from the previous section, as we use two independent delay lines. If these delay lines are off too much, the required measurement durations for the two rails (namely $error * \_a$ and $error * \_b$) become too different. If the delay is set for the one with the lower failure count, the overall measurement time becomes too high, while if it is trimmed to the one with the higher failure rate, no useful values would be detected on the second rail.

The calibration mode for the Muller C-element is straight forward. Using two multiplexers, one on A and one on B, both inputs are set to the same value. When the upper delay line is being calibrated, both inputs are connected to A, while in the other case, both inputs are connected to B. As both inputs of the Muller C-element are now operated in tandem, no marginal triggering will occur and the output delay of the Muller C-element will be nominal. By trimming the DLL, the resolution time can be varied to a point where no more errors will be detected, while when shortening the delay by one step, errors will occur. Therefore this point is at the border of the nominal output delay and marks zero resolution time. Please note that the calibration logic was directly integrated within the LUTs responsible for calculating the set and reset function of the Muller C-element (signals CAL_A and CAL_B in Figure 6.13). The precision of the calibration

is limited by the step size of the DCM, so there will always be a slight shift between the two detectors. Nevertheless, as the results for the two rails are counted separately, they can be manually aligned offline after the measurement has finished. It is important to note that, due to the independent calibration of both rails, any differences in the output delay caused by a possible asymmetry between the inputs A and B are removed. The calibration circuit will artificially set the nominal output delay for both cases to zero and therefore hide any differences.

In addition to the circuit, a simple measurement controller was implemented, and a serial port for communicating with a monitoring PC was added.

6.3.3 Measurement Procedure

The measurement procedure used to characterize the Muller C-element metastability response is as follows: First the two delay lines are calibrated to find the nominal output delay. The measurement is then started with a delay that is several steps shorter, to also characterize the transition between the nominal output delay and the area with increased response time. The measurement is started with a measurement period of one second. The high failure rate when measuring within the nominal output delay ensures that there are enough errors detected for a meaningful statistical analysis.

After a point is successfully measured, all error counts are transferred to the monitoring PC, the resolution time is increased by one step and the next point is measured. If the detected failure count for the currently measured resolution time drops below 250, the measurement duration for the next resolution time is doubled. This ensures that the resulting failure counts are always high enough for a good statistical analysis, while keeping the required overall measurement time to a minimum. When all resolution times have been measured, the results are analyzed on the monitoring PC. It is then also possible, to manually re-align the result curves for the two measurement rails (A and B).

6.3.4 Preliminary Results

Using the procedure described in the previous section, we measured the metastability response for our Virtex-4 FPGA. The results are shown in Figure 6.14. The calibration routine used for these results is a combination of the automatic mechanism described above and some manual realignment.

As can be seen, the τ value of the Muller C-element is 46.335 ps, which nicely matches the values reported in the last section (46 ps - 48 ps).

The results show a comparable number of upsets for both operation cases (A before B and B before A). It can therefore be concluded that the metastability behavior of the tested Muller C-element is not dependent on the input order, i.e. the circuit structure is symmetric. However, the metastability behavior of the Muller C-element seems to be heavily dependent on the direction of the output change. While the number of upsets for rising edges is quite high, the number for falling edges is negligible.

To show that this is not an artifact of our measurement circuit, we have performed a second measurement campaign using a different implementation of the Muller C-element (set and reset control changed). As the result of this run shows (Figure 6.15), the falling edges are correctly

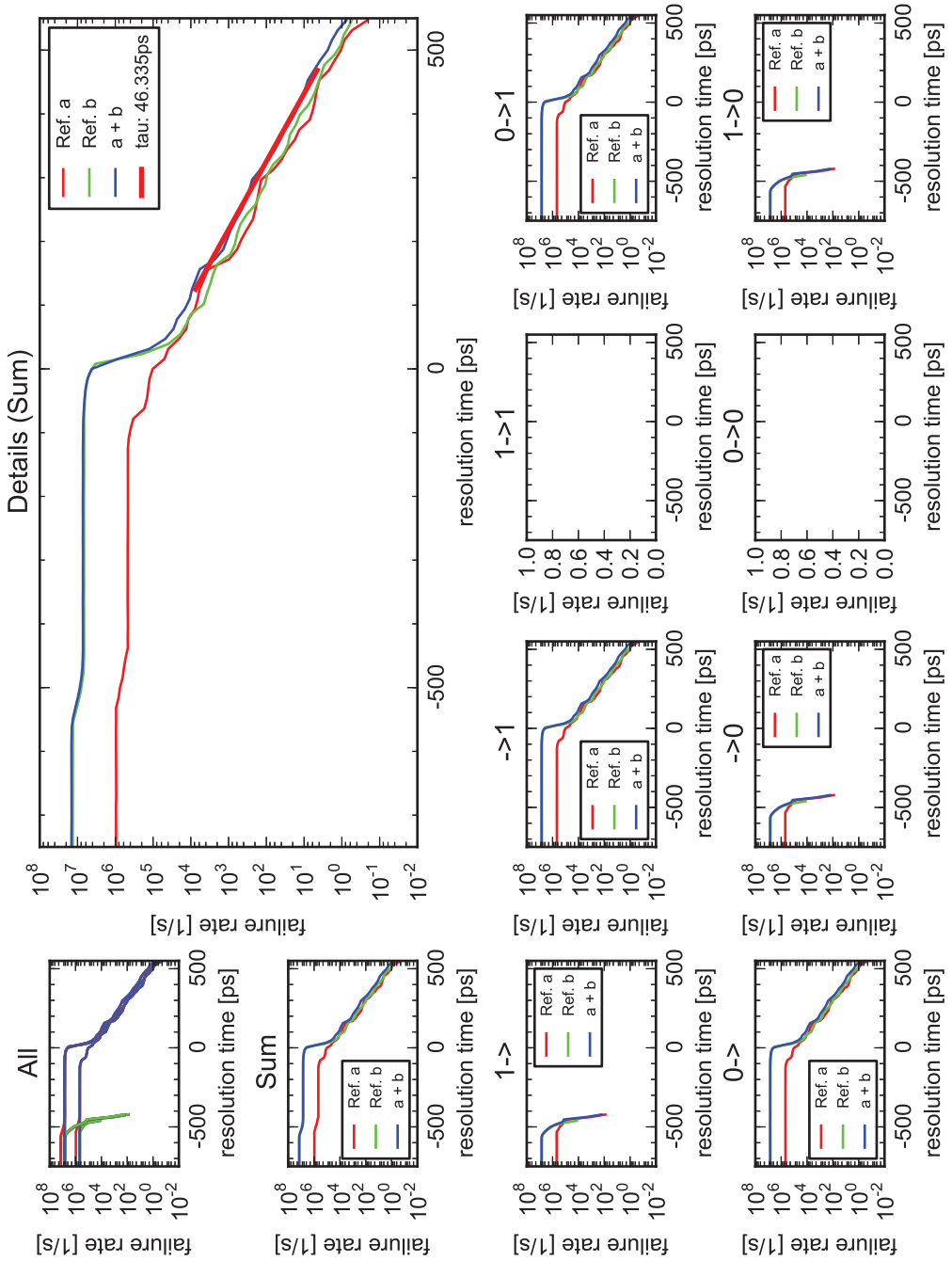


Figure 6.14: Measurement result of first implementation

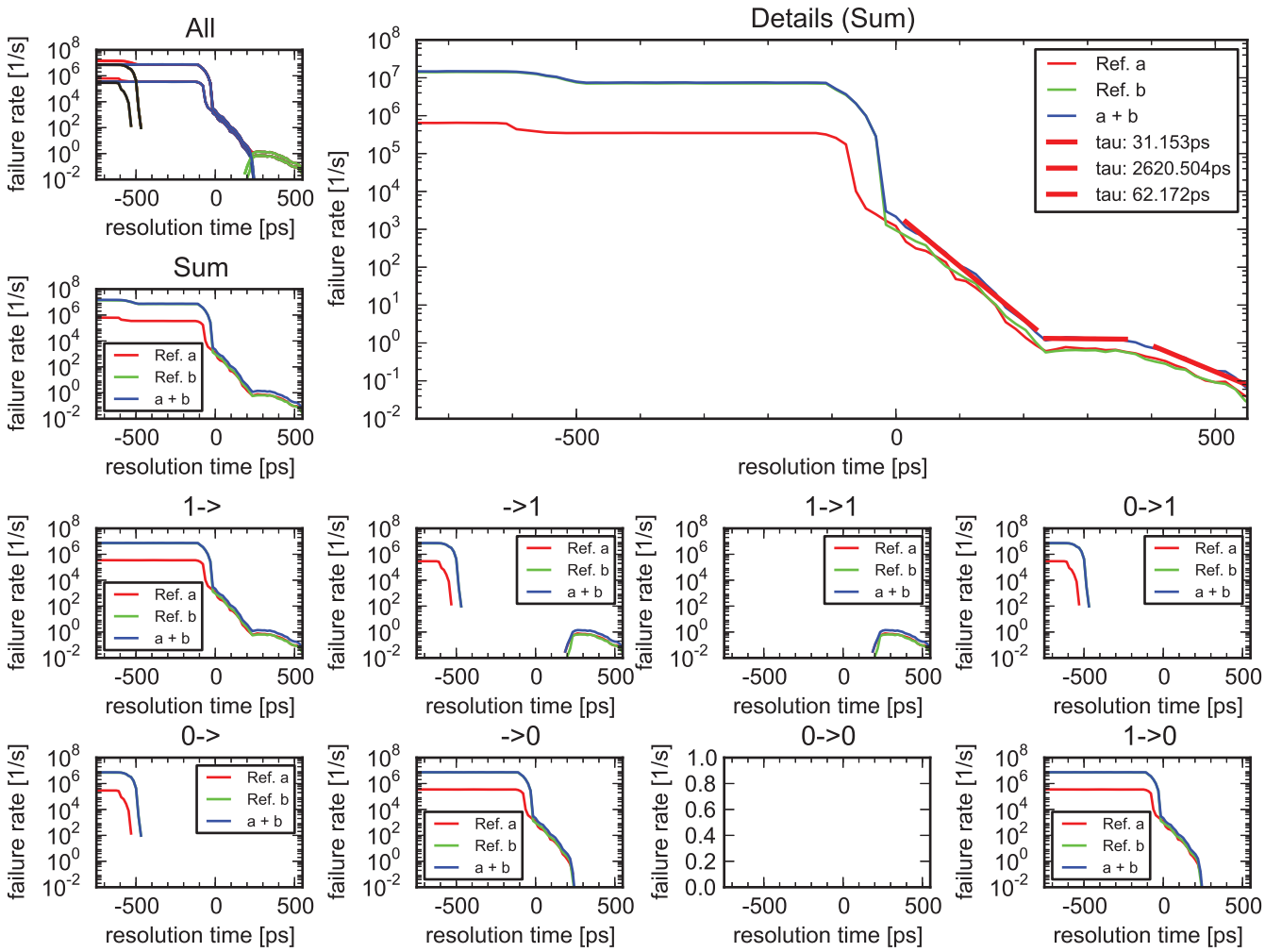


Figure 6.15: Measurement result of second implementation

recorded by our circuit. Even more, pulses may be seen at the output of the Muller C-element in this implementation.

6.3.5 Circuit Refinement

The circuit introduced in Section 6.3.1 works correctly but its complexity is quite high. In this section we will revisit the important characteristics measured by the circuit and try to derive a simplified version of the circuit. We will now investigate several options for simplifying the circuit and state the conditions necessary for them to be applicable.

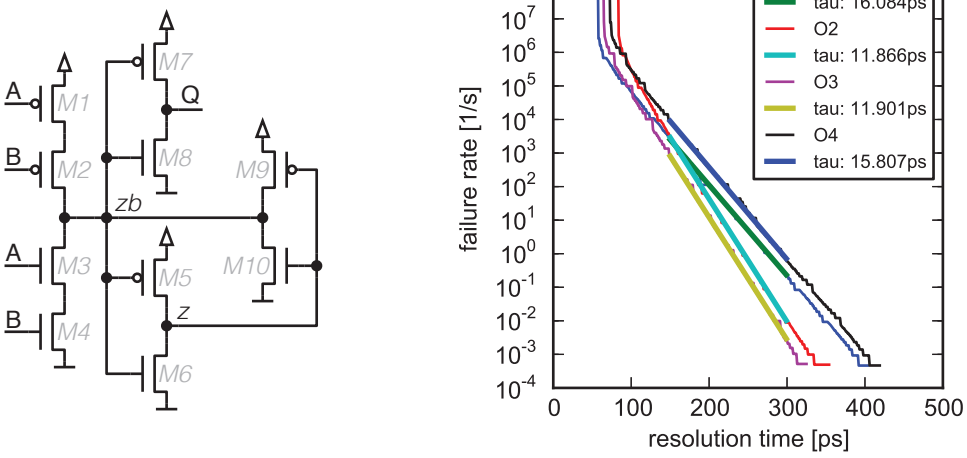
Detection of the Reference Signal Direction

In case of a Muller C-element each output polarity change may be triggered by two different but symmetrical input conditions as either of its inputs may be the first to change. Therefore, e.g., a late rising transition at the output may be initiated by a rising edge on input a followed by a falling edge on input b, or a rising edge on input b followed by a falling edge on input a. Depending on the implementation of the Muller C-element, these two symmetrical input conditions may, however, lead to an unsymmetrical response at the output. To analyze such a case further, we simulated the circuit shown in Figure 6.16a using a linear input sweep in hSpice. We then calculated the corresponding failure rate curves from the simulation results (see Figure 6.16b). As can be seen, the τ-value heavily depends on the switching direction of input b (Cases O1 and O4 for falling edges on b with a high τ value, cases O2 and O3 for rising edges on b, lower τ value). This behavior is caused by the fact that, in case b is low and a is high, both input stack transistors directly connected to the storage loop (M2 and M3) are conductive and therefore a high capacitive load is connected to the storage loop, while in the opposite case the two transistors are non-conductive and the resulting load on the storage loop is decreased.

If only the worst case resolution behavior is of interest, the detection of the reference signal may not be necessary. If, however, the different behavior is of interest, a distinction of the input polarity is necessary. Its detection can be solely based on an analysis of the reference signal's polarity. (in critical cases the polarity of the second input is the inverse of the reference signal's one).

If, as in case of the flip flop, only the output polarity would be used to distinguish between the cases, the superposition of the two triggering conditions may introduce artifacts into the results. Figure 6.17a shows such a case. As already mentioned, a late rising edge on the output may be either triggered by a rising edge on a followed by a falling edge on b (case O1) or a rising edge on b followed by a falling edge on a (case O3). As the design of the simulated Muller C-element, however, reacts asymmetrically to these cases, the corresponding τ values are different (the stack introduces a different capacitive load to the storage loop).

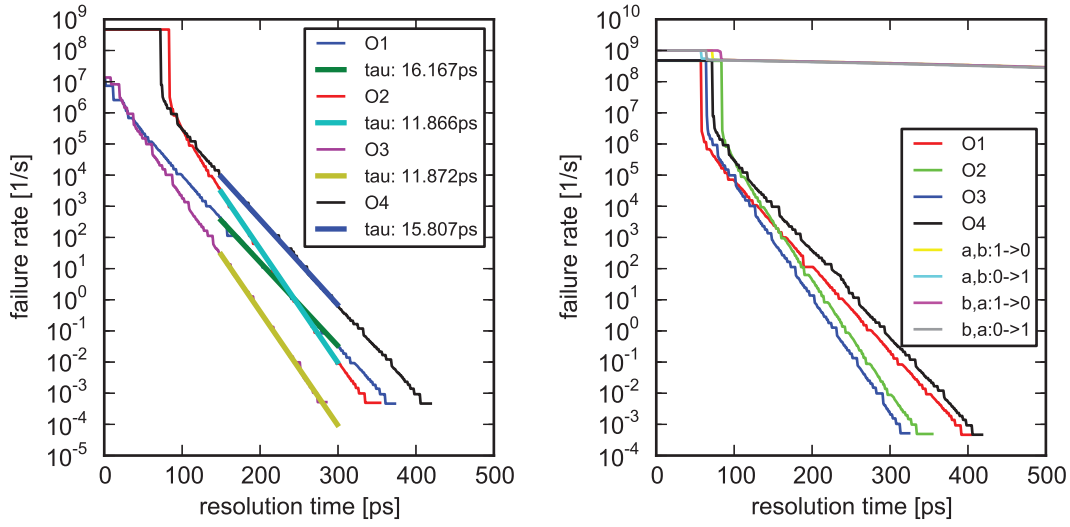
Assuming that we use b as a reference for the measurement without separating the result by b's polarity, the result of the measurement would be a sum of the curves O1 and O3. Due to the missing masking instead of ignoring the cases triggered by a (O1 and O2), their delays will be shortened by the critical overlap (cases $O1$ and $O2$). For cases outside the critical window, the result will be distorted as the overlap increases faster than the delay (far right of the figure).



(a) Demonstration circuit

(b) Simulation result

Figure 6.16: Input dependence of τ



(a) Superposition of trigger conditions

(b) Unfiltered measurement

Figure 6.17: Problematic cases

It can be seen that, for smaller resolution times, the correct result with b as a reference is dominating (case O3), while for increasing resolution time, the results which were triggered by a dominate the result (case O1). Their delays, however, are incorrectly shortened by the critical overlap. The results for falling edges are not affected, as the τ value of the erroneous trace (triggered by a, case O2) is smaller than for the correct one (case O4) and therefore the two traces will not intersect. To be able to mask the cases correctly, only the results where b has triggered the state change (O3 and O4) may be used. This can be achieved by keeping track of the polarity of the state change of b and combining this information with the polarity of the output change.

If signal a is used as a reference, a dual problem will occur for falling edges, while rising edges may be correctly measured. As the used Muller C-element has an asymmetric behavior, it is not sufficient to use the measurements for rising edges with a as reference and falling edges with b. Such a measurement will only show the worst case for both transition types, while it will hide the cases with smaller τ.

Detecting Critical Cases

The circuit introduced in Section 6.3.1 is designed for detecting critical input cases. It is therefore possible to mask out the non-critical ones which otherwise would dominate the results (see Figure 6.17b). As it may be assumed that the non-critical cases are equally distributed over all resolution times, it should be possible to subtract the expected number of these cases from the result after the measurement and the critical overlap detection circuit may be removed.

This solution, however, has a severe drawback: The width of the counters must be increased by several stages. For measuring long resolution times, measurement intervals of an hour or more are required. While only a few hundred interesting events may occur in this time period, a huge number of non-critical cases will be measured (for an input clock frequency of 300MHz and a clock divider of eight, more than 10^8 non-critical events may occur per second!). Therefore counter sizes of 36-40 bits will be necessary, while in the first case we used 10-bits[4]. The overlap detection circuit itself has a size of approximately ten flip flops. As four counters are necessary to keep track of all possible output changes, an implementation using the critical overlap detector is much more efficient in terms of area.

The influence on the measurement result by the overlap detector is minimal. Please recall Figure 6.11 for the effects of masking (only the constant part of the figure in the far right is impacted). As the overlap detector is only connected to the input of the Muller C-element and not the critical output node, it does not influence the metastability behavior of the target.

Saving the Synchronization

To measure both references, the circuit in Figure 6.9 used an elaborate synchronization mechanism. After further analysis it became clear that the assumption that the decision which of the inputs to use as reference must be made in one clock domain can be relaxed. As only the critical

[4]In the demonstration circuit we used 32-bits to be able to resolve the constant area for delays smaller than the nominal output delay.

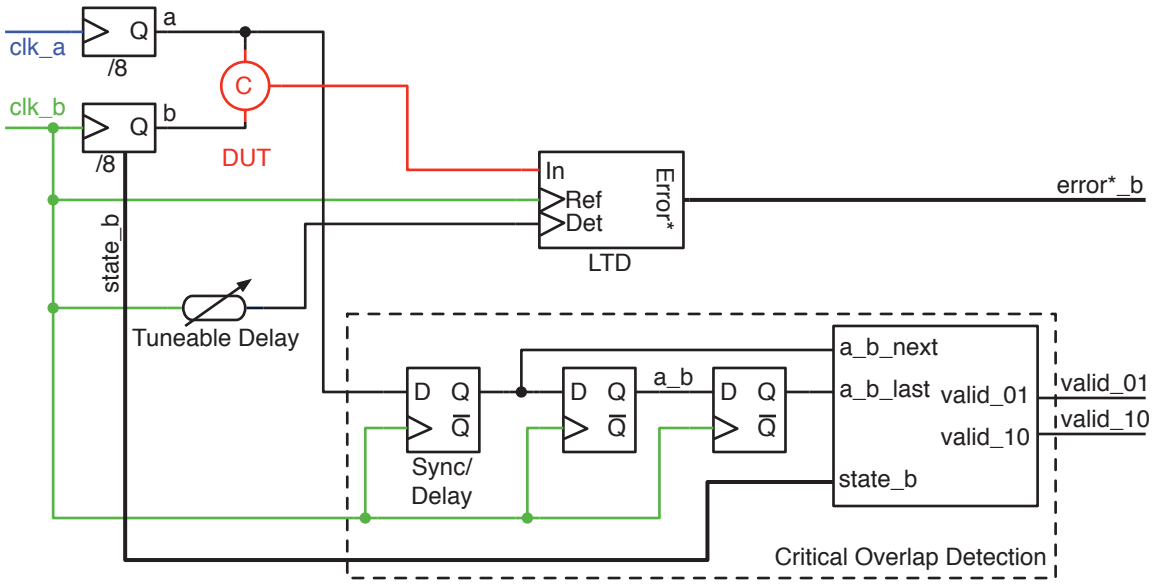


Figure 6.18: Refined measurement circuit for one reference signal

overlap conditions and not the precedence $a \rightarrow b$ is detected, this can be done in both clock domains simultaneously without losing correctness. The simplified circuit for one reference signal can be found in Figure 6.18.

Using two overlap detectors, one for each rail, the decision whether an overlap is critical or not can be done locally for both clock domains. Therefore it is sufficient to duplicate the LTD, the overlap detector and the counters without introducing any additional circuitry.

This solution, however, still adds extra load of two flip flops to the critical output signal of the Muller C-element. If load is a critical issue in a given measurement problem, only one LTD may be used and it may be swapped between the inputs of the Muller C-element (only two additional multiplexers are required at the inputs). In this case, however, it is not possible to measure the results for both reference signals at the same time, but two distinct measurement runs are necessary, doubling the measurement time.

6.3.6 Refined Measurement Results

Using the simplified circuit with two LTDs, the measurements previously presented were repeated. Figures 6.19 and 6.20 summarize the result. It can be seen that the results are identical (within reasonable measurement accuracy). Please note that due to the symmetric nature of the Muller C-element implementation in the FPGA it would be sufficient to use only one of the reference signals in this measurement but this is not true for the general case (as argued above).

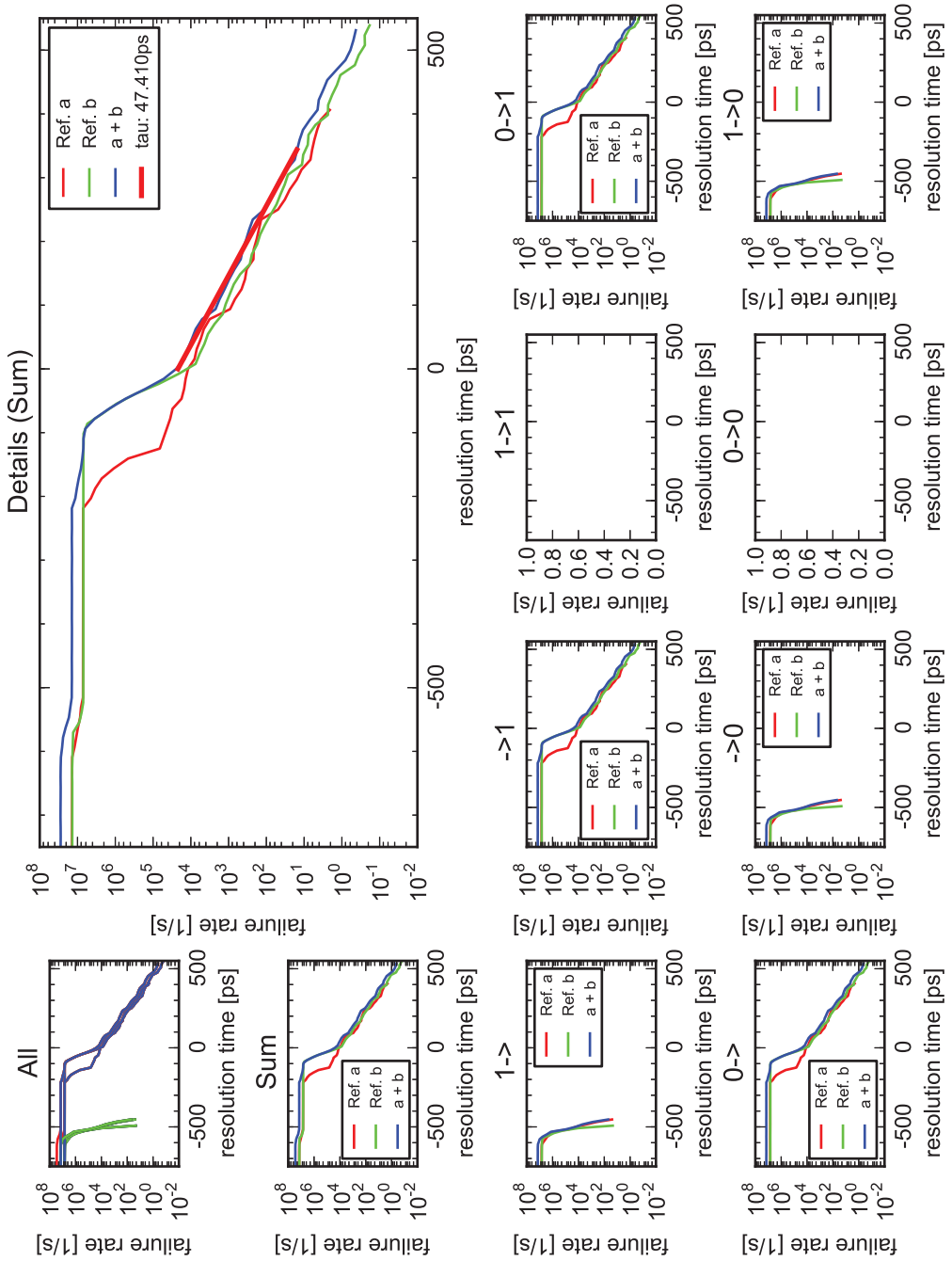


Figure 6.19: Repeated measurement result of first implementation

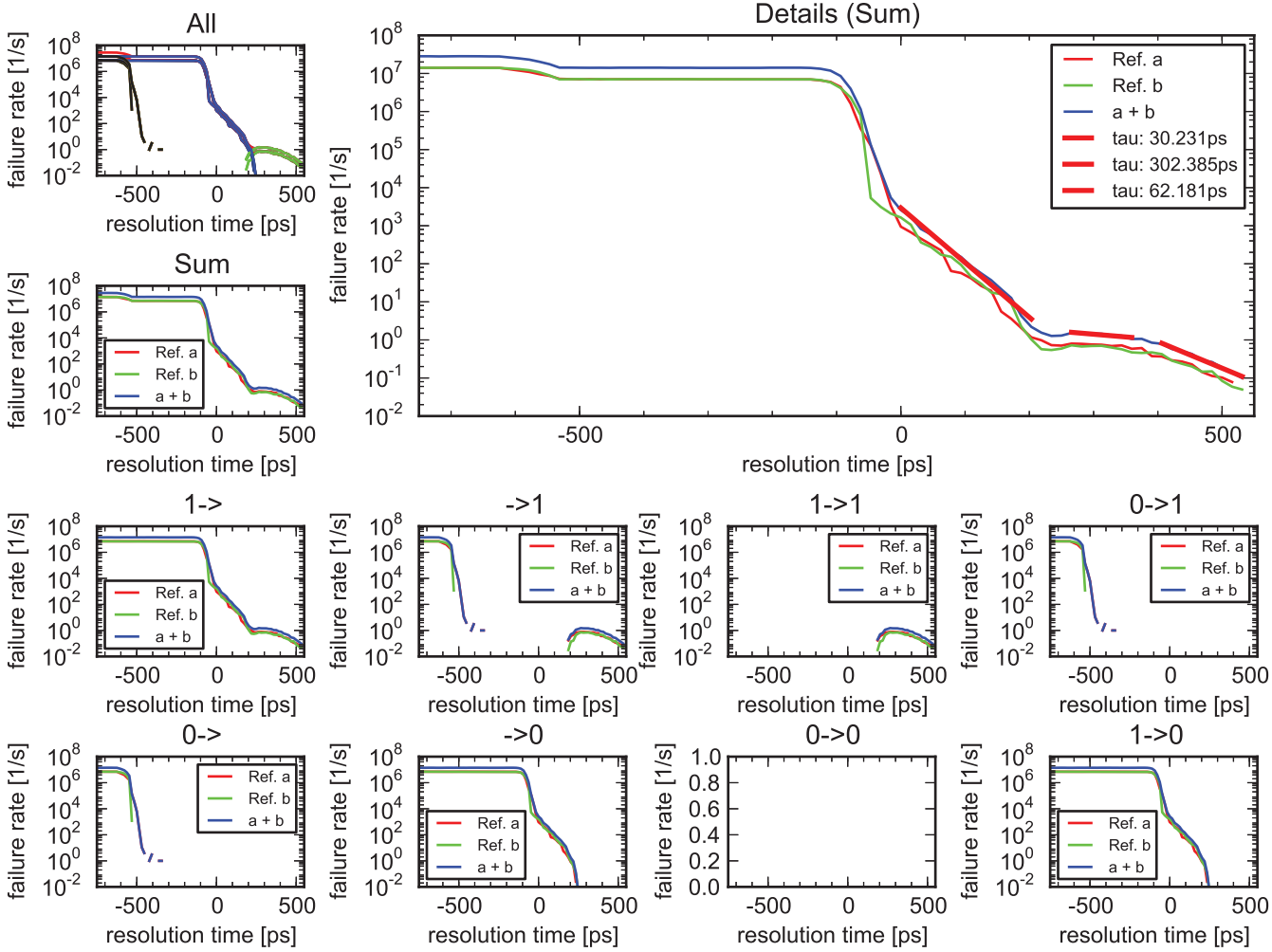


Figure 6.20: Repeated Measurement result of second implementation

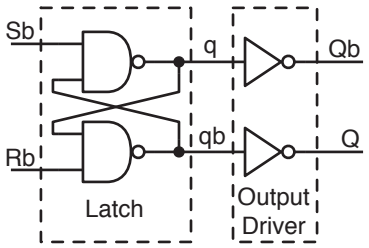


Figure 6.21: NAND-based RS-latch with a dedicated output driver

6.4 Late Transition Detection for RS-Latches and Mutexes

As the mutex is the most commonly used element for resolving input conflicts in asynchronous circuits [Kin07], the next step is to adapt our measurement infrastructure to be able to handle mutexes as well.

Historically the RS-latch, which is the core element of a mutex, was one of the first elements measured by means of a late transition detector [RC82]. The latch was driven into metastability by using the deterministic approach (recall Section 6.1). However, current research mainly focuses on D-flip flops and synchronizers as most modern circuits use the synchronous design paradigm. Unfortunately all other storage elements, including the RS-latch, seem to be neglected. Therefore no up-to-date data is available for these storage elements.

6.4.1 Circuit Design

For a late transition detector to function correctly, the output signals of the measured element must be buffered sufficiently to avoid non-digital values at its input. We therefore assume in our design that the measured RS-latch has a dedicated output diver (like an inverter as shown in Figure 6.21).

Our approach again focuses on an unbiased measurement of the element and therefore the random input generation approach (independent oscillators for both inputs) is used. As in the case of the Muller C-element, the selection of the right reference signal for the measurements is of vital importance. The number of interesting cases is, however, smaller in this case as only input overlaps leaving the forbidden state may lead to metastability. Figure 6.22 shows the interesting input overlaps for NAND based RS latches (as are used in mutexes). Please note that we are only focusing on input overlaps, as pulses do not occur in the normal operation of a mutex element.

For a low-threshold inverter only late transitions will occur. As, however, the late transition on the non-inverted output Q is triggered by releasing the reset input (Rb) first and the one on the inverting output Qb by releasing set (Sb) first, measurements using two different reference signals (Rb, Sb) are necessary to capture the full behavior of the element. Furthermore the figure shows that, if a high threshold inverter is used, pulses may occur. The reference signal for late transition and pulses are, however, different, as the opposite input must have been released

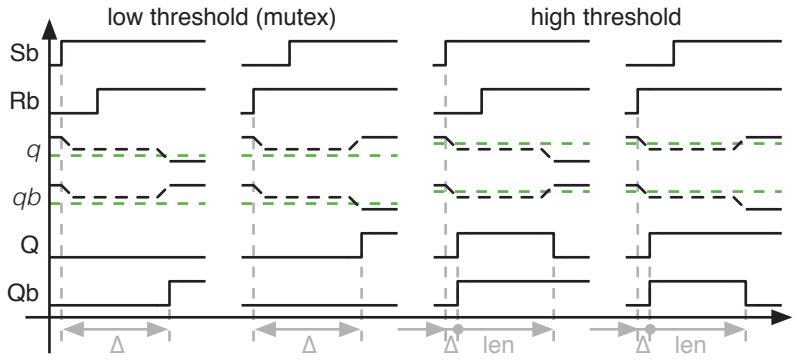


Figure 6.22: Input conditions leading to metastability

first. For a correct measurement, again, different reference signals are required. Without a priori knowledge of the threshold, this would lead to a total of four LTDs (LTD on Q with Rb as reference, LTD on Q with Sb as reference, LTD on Qb with Sb as reference and LTD on Qb with Rb as reference).

Unfortunately, for the LTDs measuring the pulses no time reference can be found for calibrating the output delay, as in normal operation no pulses are created. Therefore the output of the detector will always be zero outside the critical window. We therefore decided to omit them and only detect the occurrence of pulses using the second detector on the corresponding output. As the pulses only occur in the critical window, the delay of the pulses will be shifted by the critical overlap between Rb and Sb but no distortions will occur (differences of the input overlaps for all occurring pulses in the sub-ps range). Therefore we will, nevertheless, be able to show the existence of pulses and even to determine the corresponding τ value correctly.

It is even possible to scale down the measurement infrastructure further in some cases: If the element has a symmetric behavior in regard to the inputs the measurement of one of the outputs will be sufficient. Therefore one of the two remaining LTDs can be saved as well.

Figure 6.23 shows the late transition detection circuit for the general RS-latch. The circuit is based on the implementation for Muller C-elements. In the following we will discuss the different parts of the circuit.

Critical Overlap Detection

The detection of critical overlaps is necessary to correctly measure the output delay of the circuit. As Figure 6.22 shows critical overlap conditions (for NAND-based RS-latches) arise when both inputs have a rising edge at roughly the same time. Therefore the Muller C-element overlap detection circuit is slightly adapted to accommodate for this difference. A valid overlap on the Q-rail, e.g., is detected, if the last sample of signal Rb (Rb_last) is low and the next sample (Rb_next) is high and the state of the set rail ($state\_s\_del$) is one (indicating a rising edge on Sb). For NOR-based latches the condition would be: $R\_last = 1$, $R\_next = 0$ and $state\_s\_del = 5$ (falling edges on both inputs).

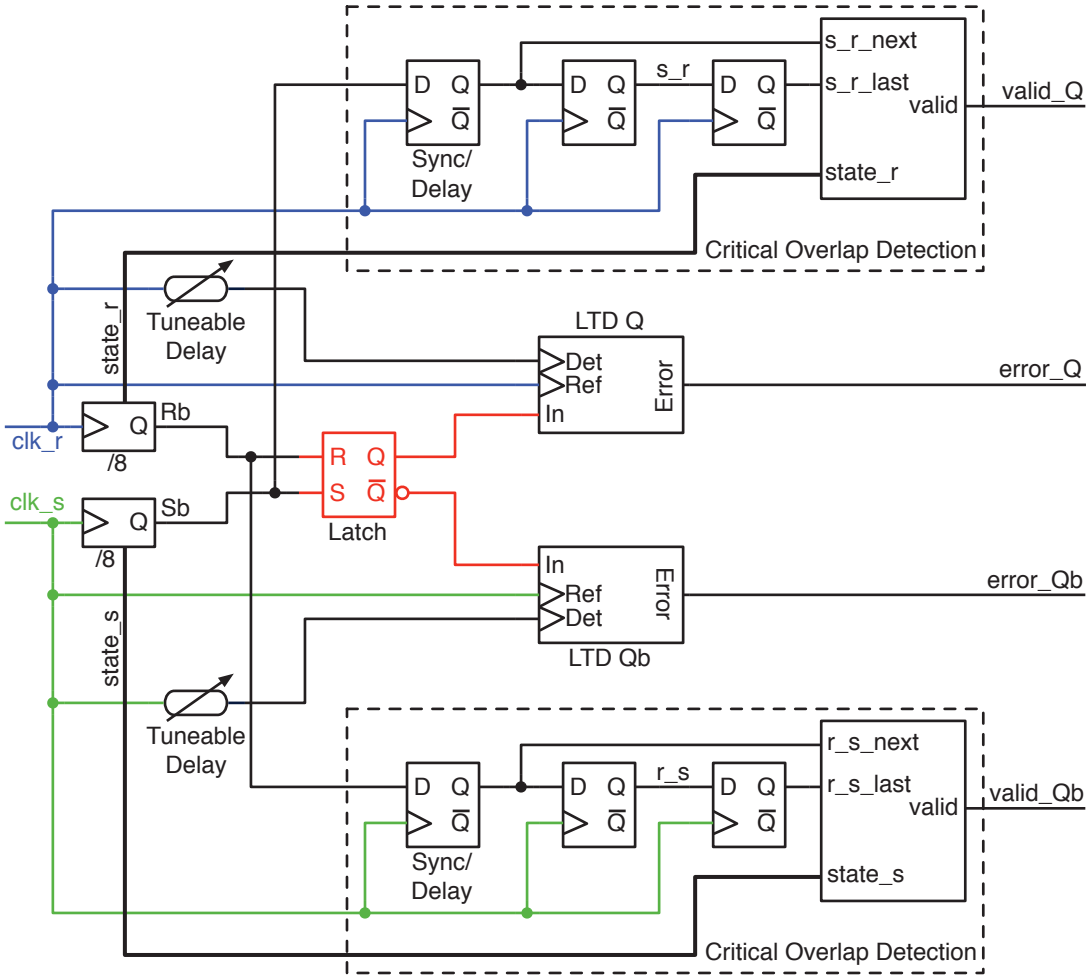


Figure 6.23: Late transition detection circuit for RS-latches

(a) First implementation

(b) Second implementation

Figure 6.24: RS latch FPGA implementations

The state signal $state\_s$ is again derived using a counter and will have a value between one and eight, as the signals Rb and Sb are created from their respective clocks by dividing them by eight. The samples for the Rb rail are taken in the same manner as the sample of the A rail in case of the Muller C-element.

Output Delay Measurement

The output delay is measured by the two late transition detectors. Based on the chosen output and reference signal only one late transition condition from Figure 6.22 can be evaluated per detector.

Calibration Circuit

We again use a calibration circuit to find the nominal input to output delay of the RS-latch. As we now use two LTDs, it is of vital importance that the phase shift between them is very small, otherwise the measurement would not be feasible. If they are too far apart, the corresponding count values would differ quite extensively (recall the exponential dependence of the error count and the resolution time). This would lead to either an significantly increased measurement time or some of the detectors would not be able to collect enough data samples to get meaningful results.

For the calibration of the RS-latch, the inputs are connected such that Sb is always the inverse of Rb. Using this scheme, the outputs of the RS-latch will toggle for each input change which is the optimal condition for the delay calibration.

6.4.2 Prototype Implementation

The prototype system was implemented on a Xilinx Virtex-4 FPGA, as we already know the τ values for its storage elements. Implementing the RS-latch measurement circuit there was, however, a slight problem. The storage elements have only one single output (Q) and do not support the direct measurement of the inverted one (Qb). Therefore only the measurement circuit for symmetric latches could be built in the prototype. Figure 6.24 shows the implementation used for the RS-latch.

Measurement Procedure

The measurement procedure is similar to the other LTD implementations already shown in the thesis. First we use the calibration mode to trim the delay line such that it mimics the nominal output delay. After finding this point, we leave the calibration mode and start with a measurement duration of one second and a detection delay several steps less as the nominal output delay. This ensures that the error count at the start is high enough even as we use such a short measurement time. The detection delay is increased step by step and the measurement is repeated for each detection delay value. If the detected failure count falls below 250, we double the measurement time for the next output delay. After recording the failure counts for all output delay values by the monitoring PC, the calibration value is manually fine tuned and a failure rate vs. resolution time plot can be generated.

6.4.3 Results

The results are similar to the ones we already obtained for the Muller C-element. As we use the same basic configuration of the D-latch this is not surprising. Figure 6.25 depicts the result plots. This τ value is again in the same order as for the other measurements ($\tau = 47.787\ ps$). This confirms that our measurement approach indeed works for RS-latches as well.

The measurements have only shown late rising transitions ($0 \rightarrow 1$). The reason is that the underlying D-latch, if it starts from 0, is only able to produce late transitions (see our previous results).

The second RS-latch implementations differs significantly in its behavior. If both inputs are active, the first implementation (Figure 6.24a) will output a low value, while the second implementation (Figure 6.24b) will generate a high output. Therefore the used reference signals have to be swapped. Figure 6.26 shows the measurement results for the second implementation. Since falling edges are produced when leaving the critical state, this implementation is, as we have seen in our other measurements, prone to pulses for our target technology.

6.5 Summary

Our measurements have shown that it is possible to create detailed metastability characterizations using digital equipment only. Our extension of the standard LTD scheme has enabled us to perform state dependent analyses and even measure the response of the slave latch. The low cost technique utilized by our approach makes these data available to all designers. There is no need for expensive lab equipment (like oscilloscopes and high precision pulse generators) to characterize the metastability behavior of FPGA flip flops.

The extension of the D-flip flop solution to other storage element variants, namely Muller C-elements and RS-latches, was only possible because of the previously developed state dependent analysis. For these elements a correct measurement heavily depends on knowledge about the output transition type and must be, in case of the Muller C-element, also correlated to the input transition polarities. Therefore the results of the state dependent LTD were crucial for the measurement circuit design. One of the main challenges for these circuits was to safely detect critical overlap conditions to be able to efficiently process the results of the LTDs.

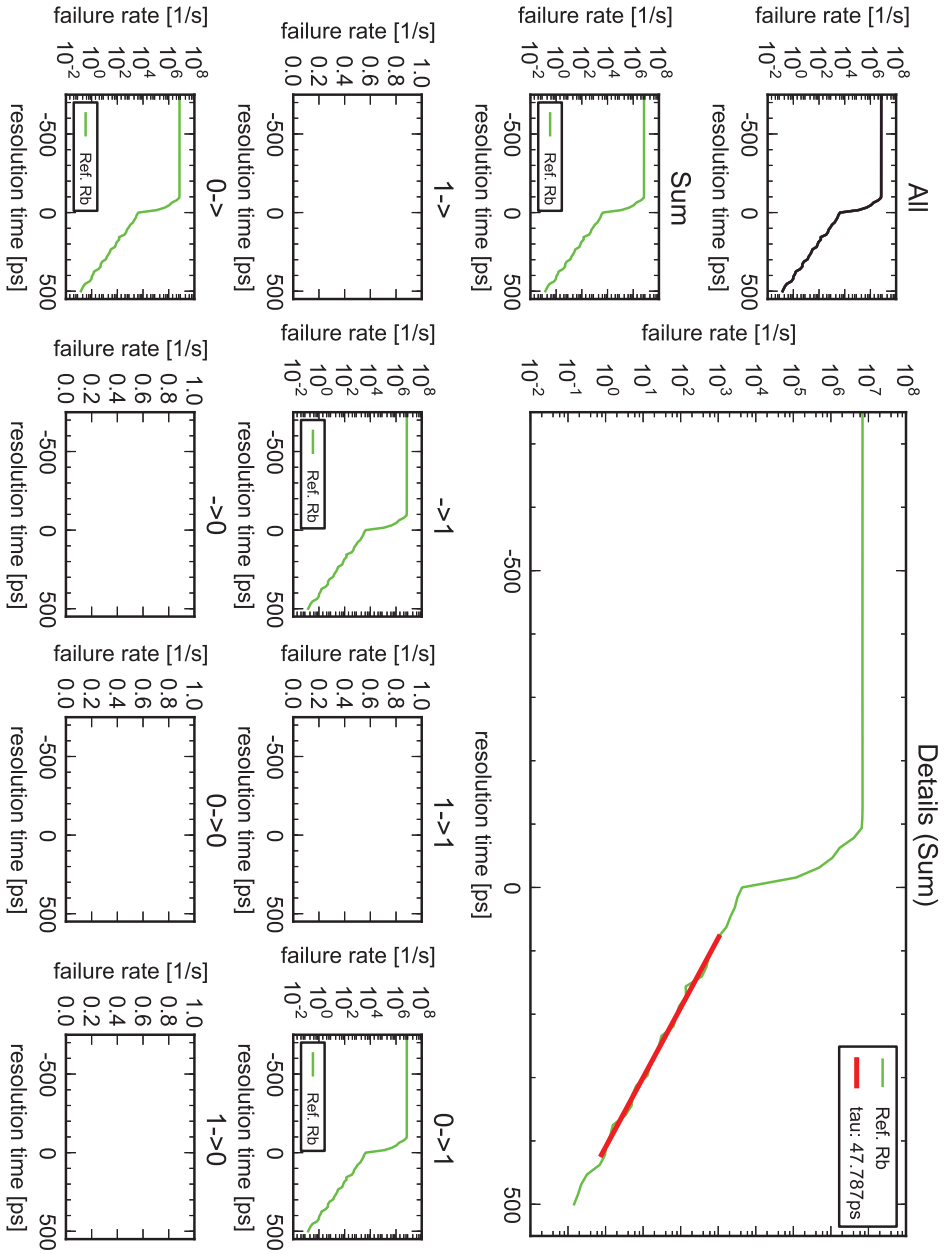


Figure 6.25: Measurement result

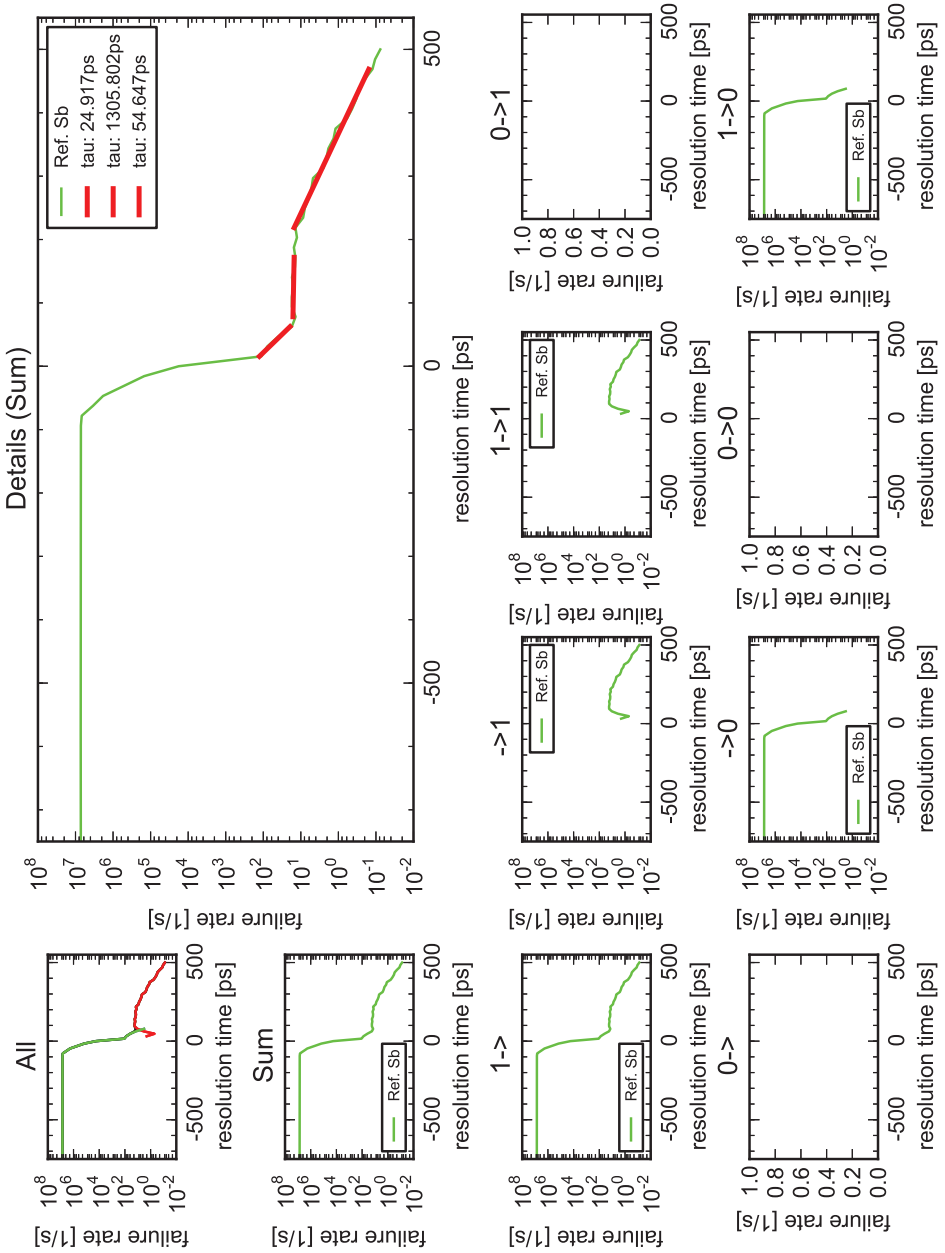


Figure 6.26: Measurement result

As all measurement prototypes were implemented on the same FPGA board, the resulting metastability characteristics were expected to match in all measurements. The results show that, within reasonable error margins, this is indeed the case. Therefore we can conclude that our measurement infrastructure is fit for characterizing not only D-flip flops but also Muller C-elements and RS-latches (including mutexes).

The results of the measurements are, however, not directly comparable to the simulation results as our simulations have, until now, not encompassed failure rate plots. In the next section we will therefore introduce a methodology to create failure rate plots out of our simulations and compare them to the measured results to get a more detailed idea of how the storage elements behave in the metastable state.

7 Merging Results from Simulation and Measurement

When designing a measurement architecture for metastability, the main goal is to get as detailed signal traces as possible. The target architectures, however, do not always support the necessary components for analogue signal measurement (e.g. when using FPGAs as target architecture), or the required internal nodes of the storage elements are not accessible to the designer (e.g. in case of a standard cell ASIC, where the elements are normally black boxes). Furthermore the analogue probing of internal signals may add extensive additional load to these nodes completely changing the metastability behavior. Therefore it is often required to limit the measurement efforts to digital only solutions as e.g. the late transition detection scheme already shown in this thesis.

Nevertheless for a thorough analysis the analogue traces of the internal nodes are essential. Straight forward simulation alone, on the other hand, is problematic because the simulation models of the transistors are not designed for such corner cases. We will therefore show how to qualitatively match the observed behavior of our measurements to analogue simulations. These results will enable us to extract a set of internal conditions leading to the observed output behavior.

7.1 Simulation Model

As models for FPGA cells and interconnect are not available, we will model the FPGA primitives using custom cells in a industrial 90nm technology library. Of course such a model will have quite different quantitative behavior but the qualitative one will be similar as the technology model and the FPGA have the same feature size. For this analysis we will concentrate on a D-latch model (as all measured components on the FPGA were built using D-latches). We implemented the circuit shown in Figure 2.2c as custom cell on the transistor level.

The input stage of the detection flip flop and the interconnect of the FPGA are modeled by adding two extra inverters to the output of the D-latch (see Figure 7.1). This model for the interconnect and the input stage of the flip flop is very basic but should suffice for our purpose.

The inputs D_in and EN_in are driven using very steep trapezoid signals (rise- and fall-times

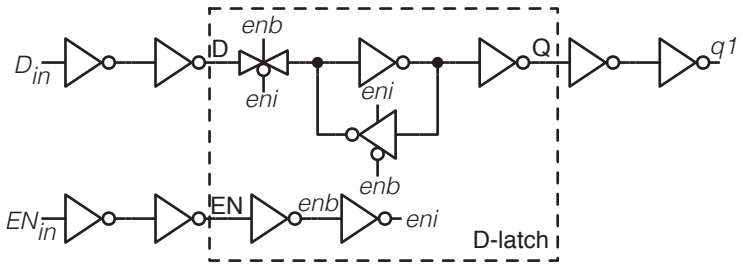


Figure 7.1: Simulation model

are 10^{-15}s). To achieve a more realistic behavior of the latch these signals are, however, fed into two inverters each. The inverters will convert the steep rise- and fall-times such that they comply to the nominal values of the target technology.

7.2 Simulation Algorithm

As in Chapter 4, we use a linear sweep with adaptive stepping to perform the simulations. The main difference is that, instead of the metastability time, the input- to output delay is measured (from the D-input to the output of the inverter chain $q1$). The delay is measured at a 50% level of VDD from the first crossing of the input to the first crossing of the output. Furthermore, if the output exhibits a second level crossing, the length of this pulse is also recorded (again at a 50% level of VDD).

The adaption of the step size is based on the differences in delay and pulse length between two successive simulations. If the values for the two simulations are further apart than a certain threshold, the step size is divided by ten. If, on the other hand the differences are very small (a tenth of the threshold), it is tried to increase the step size again. If the new simulation indeed leads to a delay (and pulse) length difference within the desired margins, the new step size is kept. Otherwise the old step size is kept. The simulations are performed for both input transition types (rising and falling transitions) independently.

To be able to compare the simulation results to the measurements, the same result plots have to be generated. To emulate the late transition detection in the simulation, without the necessity to simulate the whole detection circuit, we digitize and rasterize the analogue output signal $q1$ (result of the spice simulation) with the desired temporal resolution and digital threshold using Python (50% VDD and 10 ps are used throughout this chapter). Afterwards the beginning of the output trace is trimmed such that it is now exactly at the time of input change (all data before that point is discarded). The error trace is generated by comparing each sample with its end value (which emulates the comparison to the reference value in the LTD measurement). All values different from the end value have exhibit an error and the corresponding entry in the error trace is therefore one. The error trace is then weighted (multiplied) by the current step size (as it is assumed that all, not simulated, sub-steps within the current step will lead to the same error trace). By repeating these steps for all simulations and summing up all those error traces, the

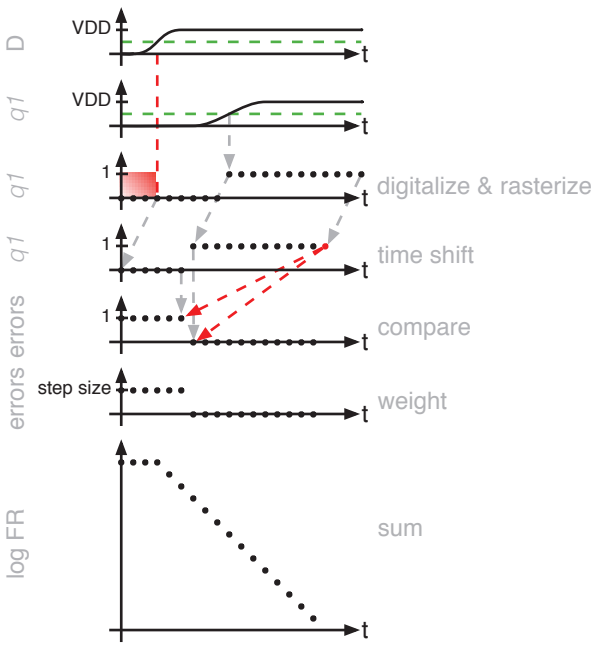


Figure 7.2: LTD emulation in the simulation

failure rate vs. resolution time plot is created. Figure 7.2 visualizes the process.

7.3 Observed Phenomena

We now want to recreate some of the phenomena identified in the measurements. This way we can get possible analogue signal shapes leading to the same observations.

7.3.1 Late Transitions

As we have seen in our measurements, late transitions lead to a exponentially decaying behavior of the failure rate plot. The same behavior is visible in the simulation results (see Figure 7.3b). A corresponding analogue simulation signal trace for the critical case (the longest delay of a late transition with the chosen overlap accuracy) can be seen in Figure 7.3d. Please note that the output of the latch exhibits analogue behavior in the figure but the interconnect and the input stage of the detector flip flop convert the analogue value into a late transition. As the thresholds of the inverters within the chain will never match perfectly, this conversion phenomenon will always occur. The better the thresholds are matched, the further the analogue signal will be able to propagate before it is converted, however in practical cases, it will eventually become a late transition (or pulse).

Another possibility for converting the analogue level into a late transition is, as already discussed, a low- or high-threshold inverter. Figure 7.4d shows the simulated signal traces for

this case. Please note that in this case the output of the latch already is at digital levels (due to the shifted threshold of the output inverter). The resulting failure rate plot (see Figure 7.4b) is quite similar to the one of the matched inverter. Therefore the late transition detection method can not distinguish between a late transition created by a high-(low-)threshold output inverter and a late transition created by the FPGA fabric.

7.3.2 Pulses

The results of the late transition detection measurements also showed the occurrence of supposed pulses. The simulations have shown that pulses indeed create the plots with the same basic shape (see Figure 7.3a, green plot). The corresponding, simulated, analogue traces for a matched output inverter are again shown in Figure 7.3c. The output is at analogue level but the FPGA fabric and the detector flip flop will convert it to a pulse instead of a late transition. Due to the single threshold of the involved components, if one transition direction is converted to a late transition, the other must be eventually converted to a pulse. The reason is that, to create a late transition the threshold must be on the other side of the analogue output level. If, however, the signal starts in the opposite logic state, the threshold must be crossed to reach the analogue level and if the transition is not finished but the end result is the same logic state as the start value, the threshold must be crossed a second time leading to a pulse.

If, again, a low- or high-threshold output inverter is used, the conversion already happens at the output inverter and no signal forming is done by the interconnect. The simulated analogue trace can be seen in Figure 7.4c. The resulting failure rate plot again shows a high similarity to the one of the matched threshold output inverter (see Figure 7.4a), but with an increased probability for pulses as the conversation process is more efficient than in the first case. This similarity, however, makes the two cases indistinguishable for the late transition detection measurements. One may argue that, if the signal forming is done by the fabric, the observed results are artifacts of the measurement circuit. As, however, the same fabric is used for the circuits built on the FPGA, the same signal forming will be present in those and therefore the measured results are valid.

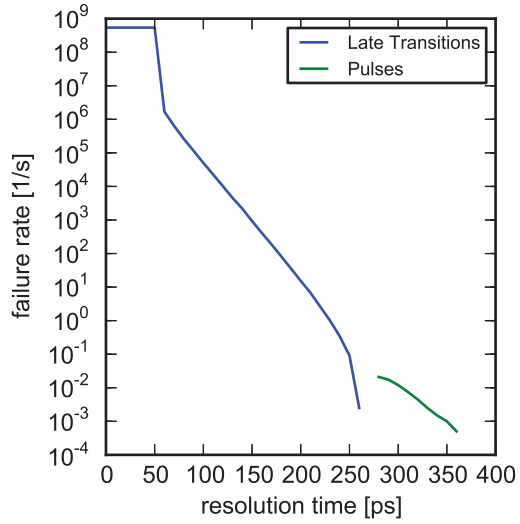
7.4 LTD Measurement Result Matching

What we are actually interested in is the time the inner storage loop is metastable and not some variable output delay. In this section we will analyze how these two characteristics are connected. To illustrate the dependence, we have simulated a D-latch and plotted the traces of the output delay and the metastability time into the same axes.

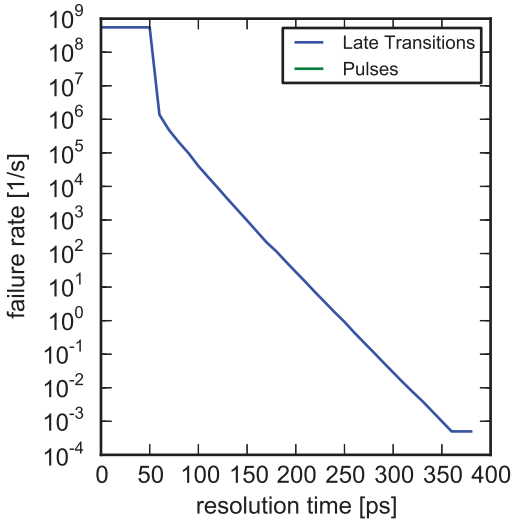
7.4.1 D-Latch

Depending on the threshold of the output inverter there are basically two possible behaviors, namely the creation of late transitions or a combined creation of late transition and pulses[1].

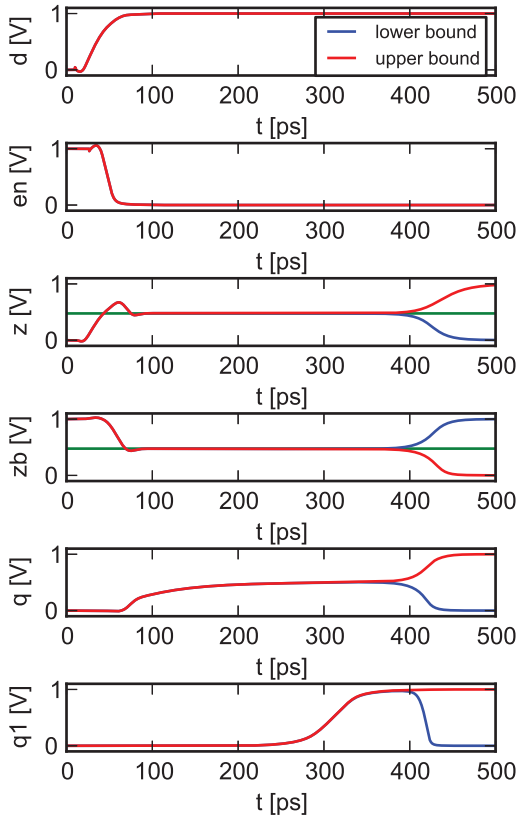
[1] We assume that any analogue effects will be converted into digital signals by the output inverter and the interconnect



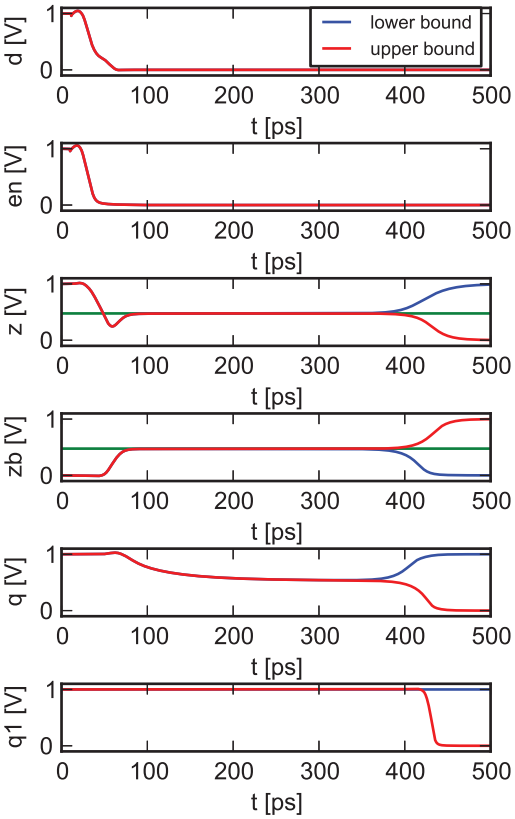
(a) Failure rate (rising edges)



(b) Failure rate (falling edges)

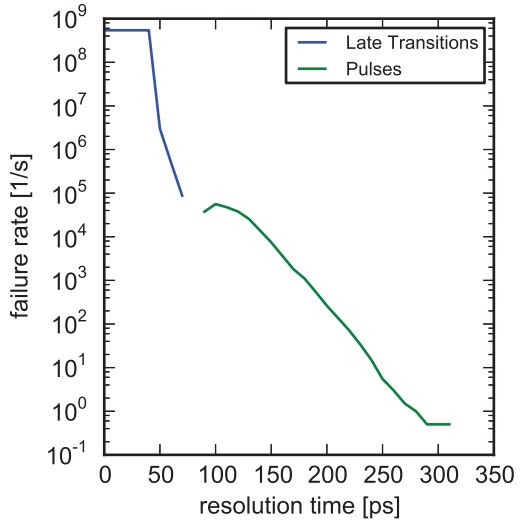


(c) Critical case (rising edges)

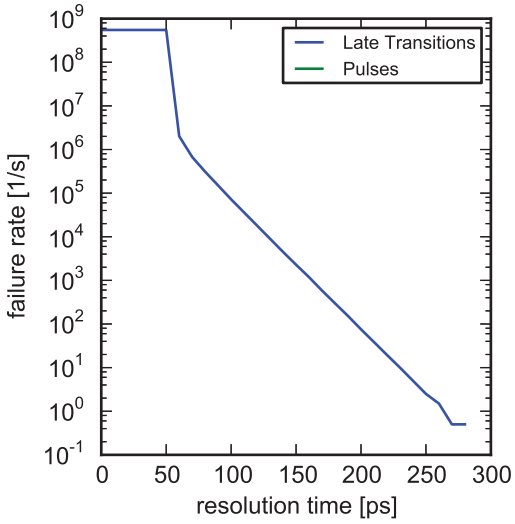


(d) Critical case (falling edges)

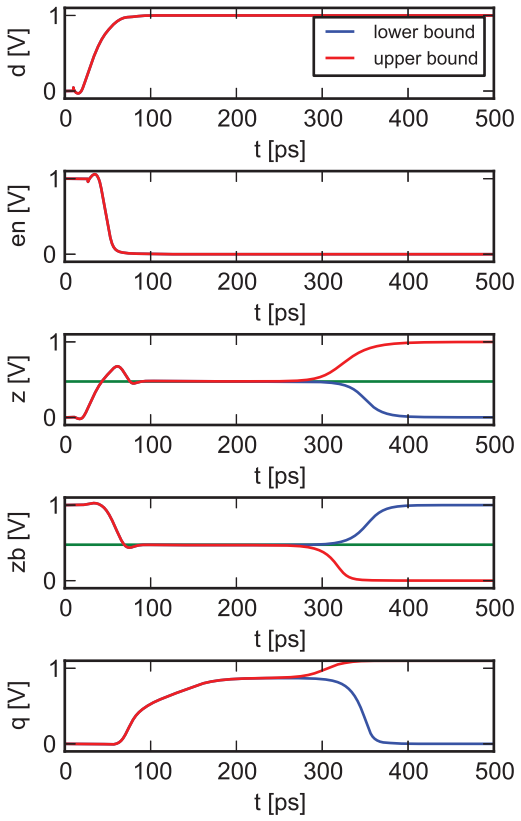
Figure 7.3: Metastable response of a D-latch (matched output inverter threshold)



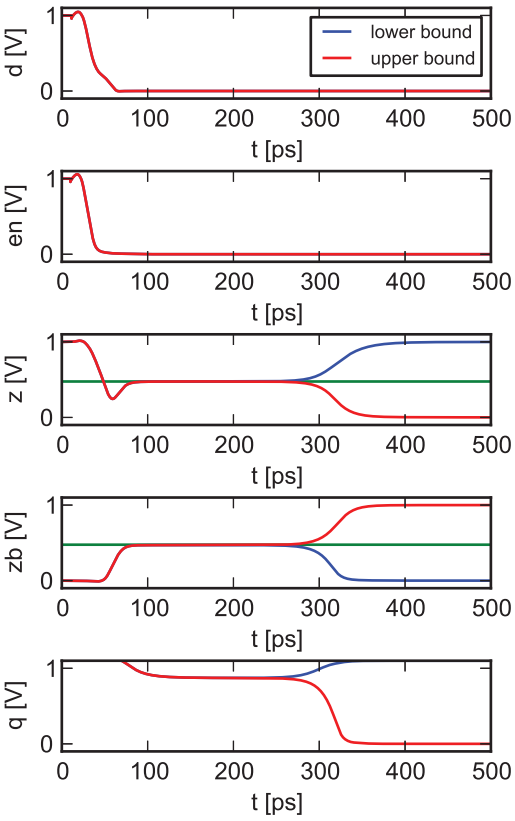
(a) Failure rate (rising edges)



(b) Failure rate (falling edges)



(c) Critical case (rising edges)



(d) Critical case (falling edges)

Figure 7.4: Metastable response of a D-latch (high output inverter threshold)

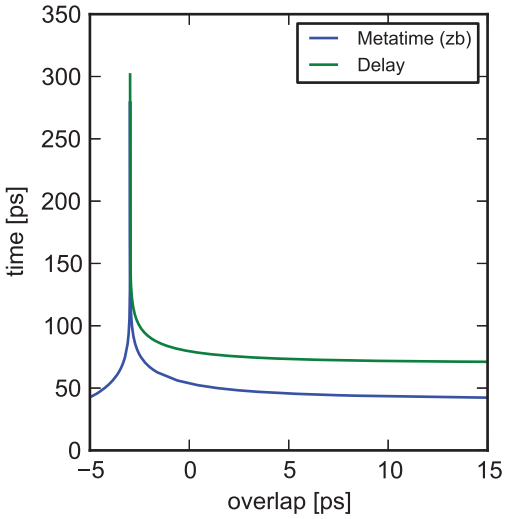


Figure 7.5: Correspondence of metastability time and output delay

If the latch only produces late transitions the resulting delay has the same characteristics as the internal metastable time (see Figure 7.5). Both curves have the same curvature and therefore the same rate of change. As the output delay includes the nominal propagation delay as well, the curves have a constant offset. The late transitions, however, only cover the metastability times after the balance point (state switch).

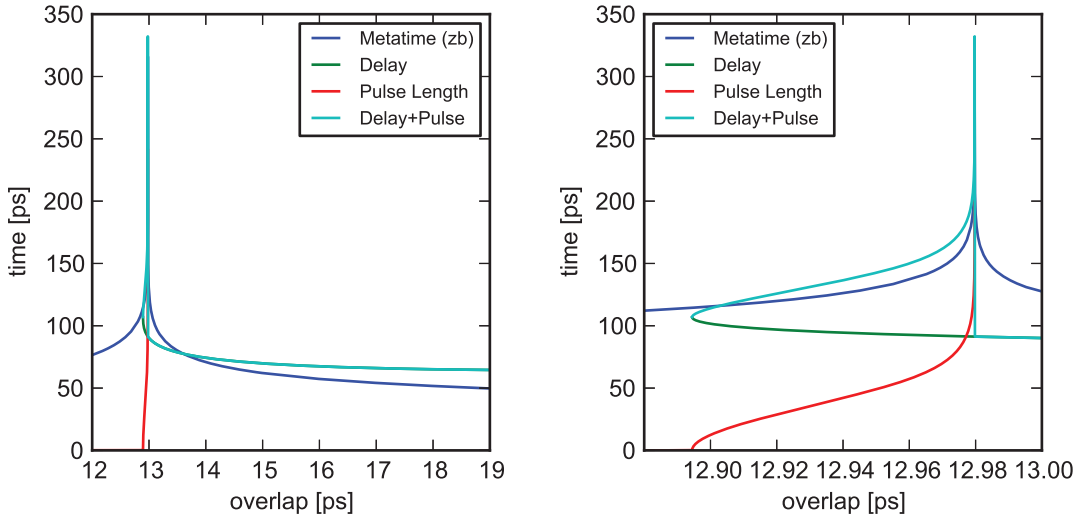
If pulses occur, the output delay for large overlaps follows the metastability time, but the smaller the overlap becomes, the more the curves deviate. This part of the curve is created by late transitions (in this area no pulses occur) and it expresses the behavior after the state change (see Figure 7.6a). The area of the curve where pulses are experienced (see Figure 7.6b), however, shows a different behavior. The pulse lengths follow the curvature of the metastability time quite nicely but represent the part before the state switch. If the overlap becomes smaller, the curves start to deviate again until the pulses disappear. This behavior is due to the fact that the metastability time becomes too small for the output to react.

7.4.2 D-Flip Flop

In case of the D-Flip Flop three different effects can be distinguished:

- Late transitions of the master latch
- Late transitions of the slave latch
- Pulses created by the slave latch

If the master latch creates a late transition on its output while the slave latch is transparent, the late transition output of the master is copied by the slave. The curvature of the delay curve matches the one of the master metastability time (Figure 7.7a). However, some distortions by the



(a) Late transition delay area

(b) Pulse area

Figure 7.6: Correspondence of metastability time and pulse length

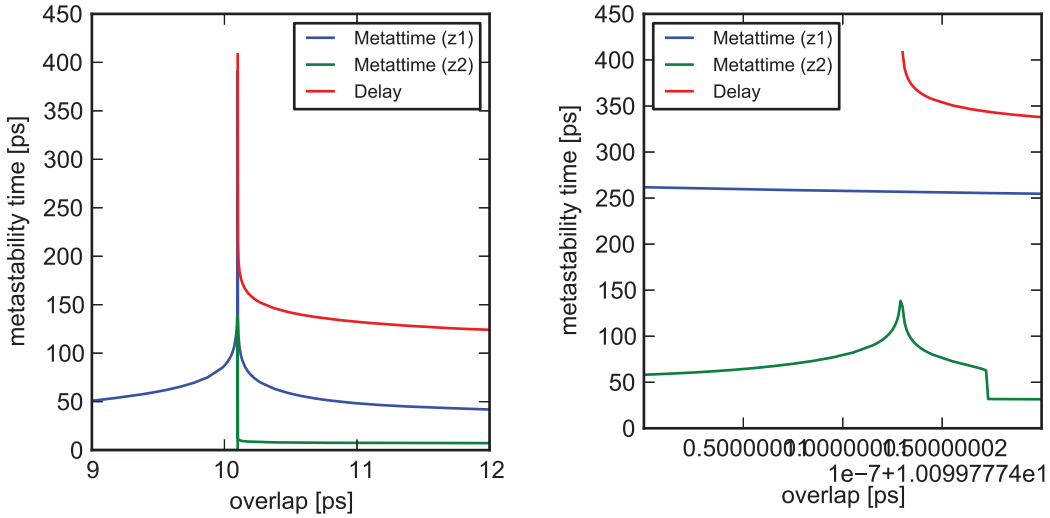
slave are possible. It is interesting to note that the maximum output delay does not correspond to the, theoretical, maximum of the master metastability time, because the resolution process of the master is preempted when the slave becomes opaque.

If the master does not resolve until the slave becomes opaque, the slave itself gets metastable. One possibility is that the slave latch also creates a late transition (see Figure 7.7b). In this area the output delay matches the curvature of the slave latch metastability time.

Another possibility is that the slave produces pulses. For large overlaps, the output delay of the flip flop reflects the metastability time of the master well (Figure 7.7c). For smaller overlaps, however, the slave latch starts to form pulses at its output. Figure 7.7d visualizes that the pulse length matches the metastability time of the slave latch nicely. For short metastability times, however, the time is not enough to form a full output pulse and the pulse length is attenuated until the pulses disappear.

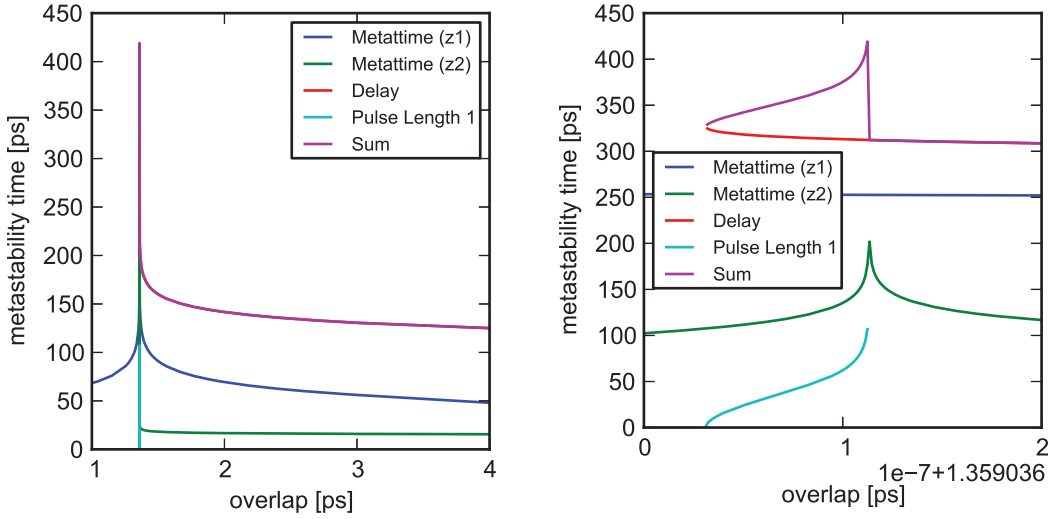
7.4.3 Remarks

The analysis of the matching of LTD results and metastability time was done for a single implementation of a D-latch and a D-flip flop. The behavior may vary from case to case. The used flip flop's master latch has a strong output inverter and therefore the signal propagated from master to slave can be assumed as being digital (late transition or pulse). Also the combination of master and slave late transitions and pulses may vary. Therefore the results in this section should be considered as case study and not as generally valid.



(a) Late transition of the master latch

(b) Late transitions of the slave latch



(c) Late transition of the master latch

(d) Pulse area of the slave latch

Figure 7.7: Correspondence of metastability time and LTD results for a D-flip flop

7.5 Summary

The late transition detection method can be augmented with additional, analogue data by matching simulation results to the measurements. The phenomena seen in the measurements, namely late transitions and pulses, could be recreated in the simulations and matching analogue traces could be stated. They are, however, not unique as the same phenomena can be created in multiple ways. We have showed two possibilities, shifted threshold output inverters and conversion by the FPGA fabric and detector flip flop input. The simulation results match the measurements quite well, even though we used another technology library (but with the same feature size) and a custom, transistor level cell for the latch. This was necessary as no spice data for FPGAs is available.

8

Digital Metastability Simulation

Current digital timing simulators (like Post-Layout simulations using VHDL or Verilog models, e.g.) require that the timing constraints of the underlying memory elements are always adhered. If in any case these are violated, the simulator will generate an error state on the corresponding signal (X for a IEEE standard logic [IEE93] based simulation, e.g.) and may even, based on its configuration, abort the simulation completely. Even if the simulation does not abort, the error state on the signal will be spread through the circuit and no mitigation is applied by the simulator making it impossible to, for example, characterize synchronizer circuits. One may argue that logical and timing masking may nevertheless be applied to the signals (the error state will only be read by a flip flop at the next active clock edge and may be masked by a combinational gate, if the other inputs solely determine the output). While this is true in the general case, when simulating synchronizers, however, no combinational gates are present between the flip flops, and the error state is kept stable until the next active clock edge and therefore will be surely propagated to the output of the next stage.

Furthermore, the timing parameters of the underlying models are fixed and specifically do not consider different input overlap times. Therefore the results of these timing simulations will have a constant input- to output-delay for all input overlaps outside the setup-/hold-window and will generate an error inside the window and no study of the timing variations for different input situations is possible.

The only possibility currently available to simulate synchronizer circuits is purely on the analogue level. Using this simulation approach, the properties of the used gates are characterized in more detail and the therefore the delays calculated by the underlying simulator are much better matched to the physical circuit. Unfortunately these simulations have a much higher requirement on computing power and suffer from limited resolution and numerical problems of the underlying mathematical models [YG07]. To be able to simulate a multi-stage (in fact even two-stage) synchronizer, accuracies in the input overlaps exceeding the double precision floating point resolution normally utilized by Spice simulators are required.

8.1 Digital Metastability Simulation Model

To overcome the problems of the currently available simulators, we have developed a new simulation methodology for metastable operation. The main purpose of this new methodology is the simulation of multi-stage synchronizers under varying input overlaps.

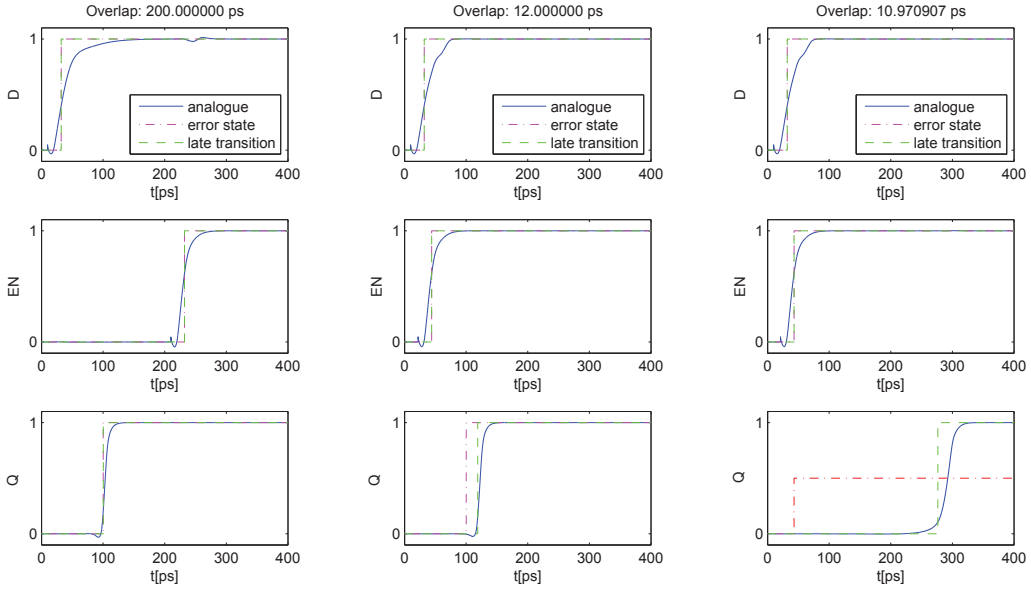
The model is based on the assumption that outside of the basic storage elements (like latches, e.g.) only digital values are present, i.e. the metastable state is sufficiently well converted by their output driver into a late transitions or pulses. As we have already shown, this behavior may indeed be observed in CMOS gates. To be able to perform the simulation, these basic elements must be characterized sufficiently well. As we have already shown in Chapter 4 the dependency of the output delay (Δ) and the output pulse length ($l(p)$), if any pulse occurs, on the input overlap can be determined using an adaptive, linear sweep Spice simulation. We therefore base our simulation model on these two characteristics.

The ability of the new model to capture the input- to output-delay behavior of arbitrary input overlaps can be exploited to generate a digital metastability simulator. In contrast to state of the art timing simulation (e.g. executed with a VHDL simulator), the input- to output-delay of the gates is not fixed any more but can be calculated depending on the input overlap. The output transition is scheduled after the resulting delay. This mechanism faithfully emulates late transitions in such a way that the gate keeps the old output value until the input- to output-delay has elapsed. At that point in time, the output switches its state cleanly. The metastability is therefore converted from the value into the timing domain. For most cases this captures the circuit behavior sufficiently well and is definitely an advance to the normally employed error state model.

Furthermore the determination of the output delay may be based on additional parameters (as e.g. the current state of the gate) capturing gates with different behaviors for different operation conditions (e.g. rising and falling edges). For each such condition set a different parameter set is then used. The modeling of metastability as late transition has even more advantages over the error state model. Two important ones are the possibility of simulating synchronizers and non-isochronous forks.

In the late transition model, a synchronizer stage only gets metastable, if the output transition of the previous stage creates a transition within its setup-/hold-window. If the transition is outside, metastability is resolved. When using the error state model, the error state of the previous stage (if the simulation was not already aborted) is copied through the synchronizer and no metastability resolution is performed. The advantage of the new method can clearly be seen here as it allows for metastability mitigation and interfacing of independent timing domains within digital timing simulations.

Non-isochronous forks have different delays on different paths of the fork. A late output transition may arrive within the setup-/hold-window of one of the successor stages on one path, driving it into metastability, while on the other path the transition arrives outside the window, enabling normal operation of another successor. Therefore inconsistent timing behaviors can be naturally modeled and simulated enabling debugging of hard-to-detect inconsistent system behavior. If, on the other hand, the error state method would be used, the critical paths would just enter the error state, while all other paths would generate valid signals.



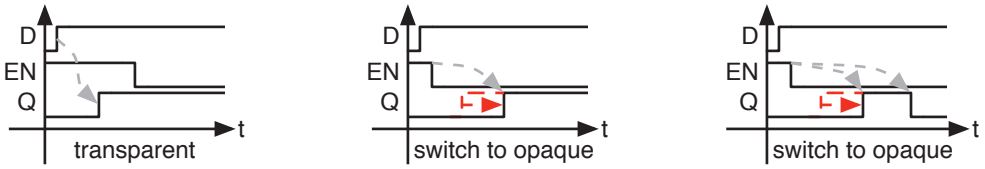
(a) Large overlap (b) Overlap near s/h-window (c) Overlap in metastable window

Figure 8.1: Comparison of the simulation models

The principle of our simulation algorithm can be found in Figure 8.1. The three traces in this figure where created by a Spice simulation of a D-latch, a manually calculated run of the proposed simulation algorithm as well as a manually calculated simulation using a fixed output delay in conjunction with an error state. It can clearly be seen that our approach tracks the delay of the analogue simulation much better than the digital simulation using the fixed delay: It faithfully reflects the increased delay already seen near the setup-/hold-window (Figure 8.1b), and it manages to approximate the position of the output transition even in the case of a setup-/hold-violation (Figure 8.1c).

To ensure the accuracy of our model it is essential that the output signals of the element are indeed at digital levels and no (major) analogue effects occur when coupling the elements. This decoupling may be achieved by the output inverter of the basic blocks. The amplification of the inverters is used to shape the output signals accordingly. Another important property is the precise handling of small overlaps even if the simulation time is long (e.g. for simulation times of several milli-seconds, time differences between two signals in the range of 10^{-21} seconds would no longer be within the dynamic range of a double precision floating point value). Therefore, all values are stored as arbitrary precision decimal numbers. The necessity of handling such small time difference additionally rules out the usage of a fixed time grid for the simulations as normally applied by digital simulators. Instead the full time stamp of the events is used and managed in an ordered list which is then processed in ascending order.

The simulator uses two different approaches to store the model values. The first, and easier approach is to directly store the output delays and pulse widths determined by the Spice simula-



(a) Output transition scheduling (b) Output tran. rescheduling (c) Creation of an output pulse

Figure 8.2: Basic simulation concepts

tions as a table and linearly interpolate between the entries. This model also allows extrapolation of the model outside the critical window but is not suitable for extrapolations of deeper metastability. The second approach involves the mathematical modeling of the storage element behavior and fitting a continuous mathematical function to the measured data. Using this model also extrapolations within the metastable state become possible and simulations of critical overlaps not covered by the underlying data can be performed.

We will continue with describing the simulation algorithm and present first simulation results of the table based approach. Afterwards, the mathematical modeling will be shown and the case study is repeated using a simulation based on the mathematical model.

8.2 Simulation Algorithm and Model Refinement

Based on the characteristics described in the previous section, the simulation algorithm works as follows: If any input of the element changes, the new output behavior is calculated. In case the element is transparent, the nominal output delay of the element is used to schedule the new output transition (Figure 8.2a). If, on the other hand, the element is opaque, the input transition is ignored. A special case is the situation when it has just been switched from transparent to opaque. In this case, the input overlap is calculated based on the previously mentioned two characteristics. This may changed the delay of an already scheduled output transition (Figure 8.2b). In this case the delay of this transition is updated. Furthermore additional transitions may be generated (if the output experiences a pulse, Figure 8.2c). These transitions are also scheduled accordingly.

Our first simulations using a basic D-latch only were very promising and the calculated output responses matched the Spice simulation perfectly (which was not a big surprise as the characteristics are matched). When simulating a flip flop built by two of these latches, however, the simulation results differed substantially. One of the reasons turned out to be that the behavior of the transparent slave latch was not taken into account when calculating the output response (as only the nominal delay is used for transparent latches). Furthermore no pulse shaping was done by the slave latch and very short pulses could propagate through the slave latch unchanged instead of being attenuated correctly.

We therefore add two additional Spice characterization steps to the model. First, the simulation of a latch switching from opaque to transparent is simulated using different overlaps

between the D- and the enable input. This step makes it possible for us to estimate the input- to output-delay depending on how long the latch was already transparent (Δ_{en}). Figure 8.3a shows an example for this characteristic. On the left side of the figure, the latch is transparent and input changes are propagated with the nominal delay to the output. If, however, the enabling of the latch and the input change occur in proximity (middle of the figure), the delay slightly increases. If, on the other hand, the latch is opaque when the input change happens, the delay linearly increases with the time the latch stays opaque after the input change.

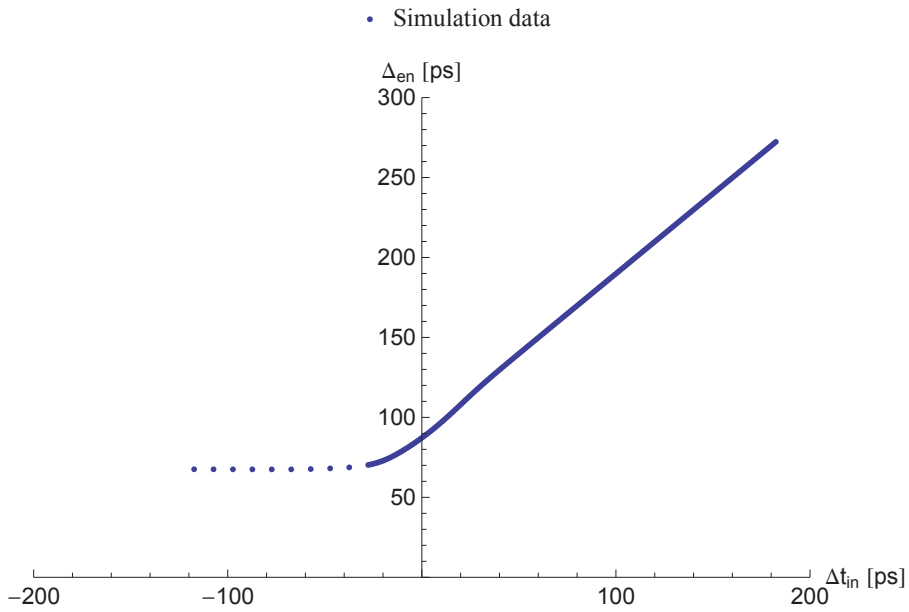
The second characteristic is the pulse forming behavior of a transparent latch. We therefore subject a transparent latch (enable tied to active) to pulses of various lengths and record the corresponding ration between output- and input-pulse length ($\frac{l(p_{out})}{l(p_{in})}$) . As can be seen in Figure 8.3b, short pulses (left side of the figure) are not forwarded but removed by the latch. As the pulse length increases, the pulses start to propagate but are still attenuated significantly. Only larger pulses (greater $75\ ps$ in this example) are forwarded nearly unchanged. Whether they are slightly attenuated or amplified depends on the characteristics of the rise and fall times of the latch.

The resulting characteristics are used to replace the nominal output delay when calculating the response of a transparent latch. This adaption increased the accuracy of our model considerably.

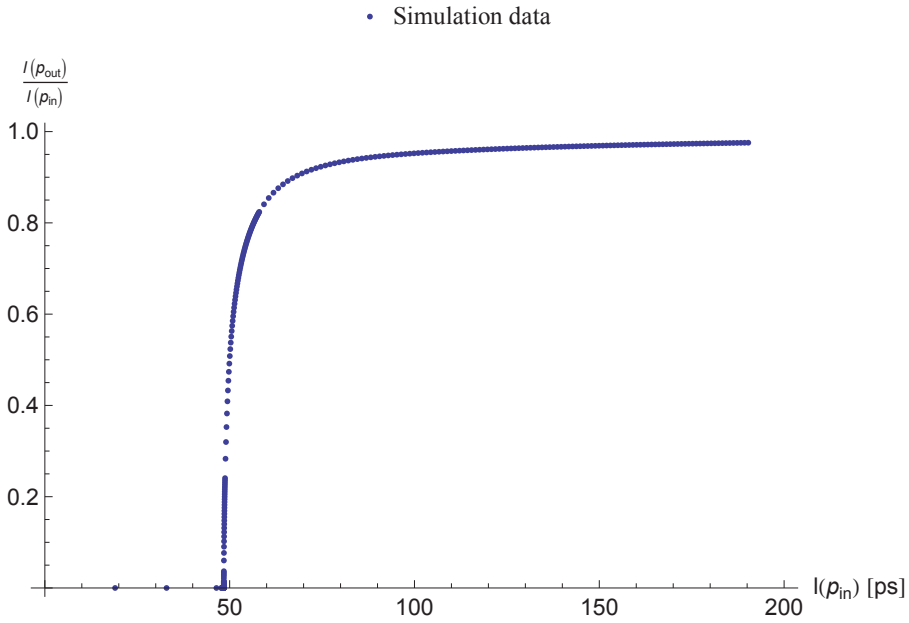
8.3 Case Study

To show the viability of our approach, we implemented two D-latches on the transistor level in a 90nm industrial library and built a flip flop out of the two latches (see Figure 8.4). First we simulated both latches independently to generate the required characteristics introduced in the last section. The additional inverters at the in- and outputs are used to simulate the load of a predecessor/successor stage. This is required to make the resulting characteristics more realistic. When building the flip flop, we have maintained the same structure as when simulating the latches individually. Therefore the enable signals of the two latches are not connected to the same internal inverters but a small clock tree is built. This makes the matching of the simulation results easier.

We simulated the flip flop once using Spice and a second time using our simulation framework. Figures 8.5 and 8.6 show a comparison of the output delays and the pulse lengths generated. It can be seen that the delay predictions match very well and also the pulse lengths are predicted nicely but have a small deviation for short ones. The critical overlaps for pulses are slightly different (femto second range). A comparison plot of the calculated error rates can be found in Figure 8.7. Therefore the simulated output signal of the flip flop is evaluated after a certain resolution time. If this value is the same as the value at the end of the simulation, no error has occurred, otherwise an error was detected. By evaluating various resolution times in parallel, an emulation of a late transition detection scheme is performed. It can be seen that the error rates of the master and the slave latch match between both simulations. Differences in the calculated error rates can, however, be seen at the points where pulses start to occur. In areas where a new output pulse appears, the failure rate curves are slightly off. These deviations are nevertheless much smaller than the differences caused by PVT variations (for worst case, typical

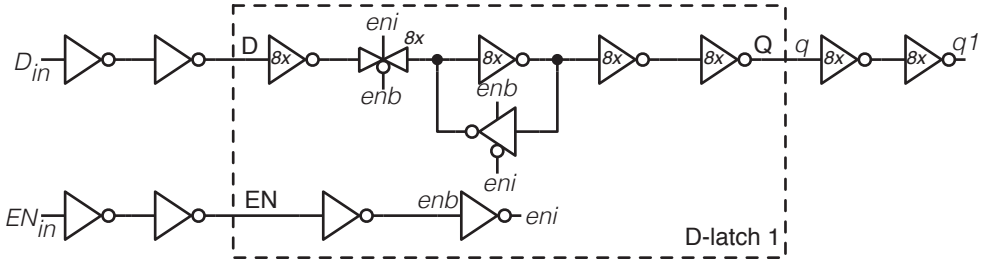


(a) Dependence of output delay and enabling time

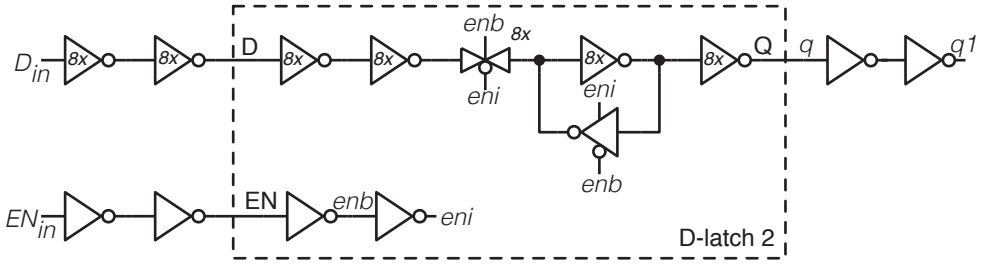


(b) Pulse propagation characteristic of a transparent latch

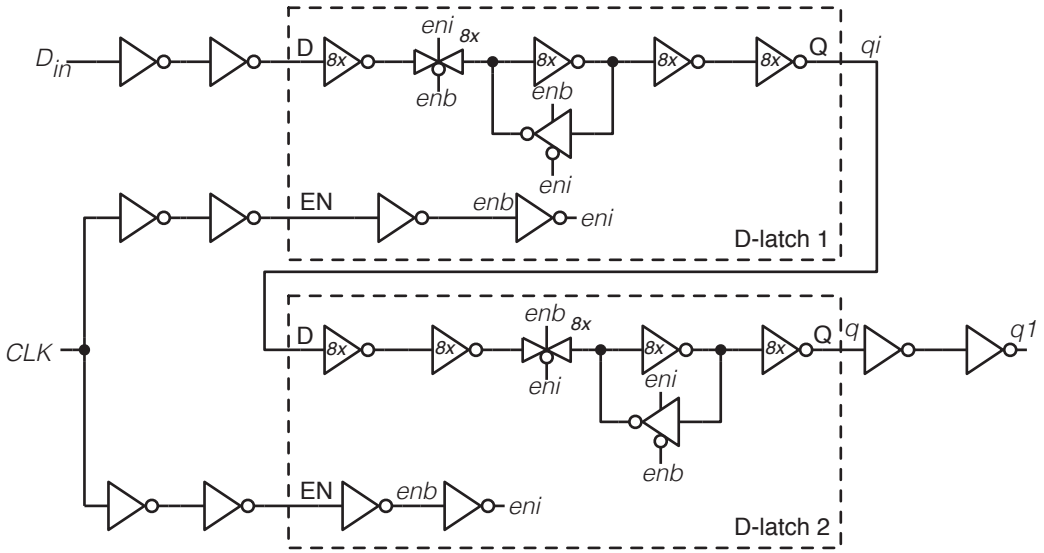
Figure 8.3: Additional characteristics of the D-latch (90nm bulk CMOS)



(a) Analogue simulation model of the first latch

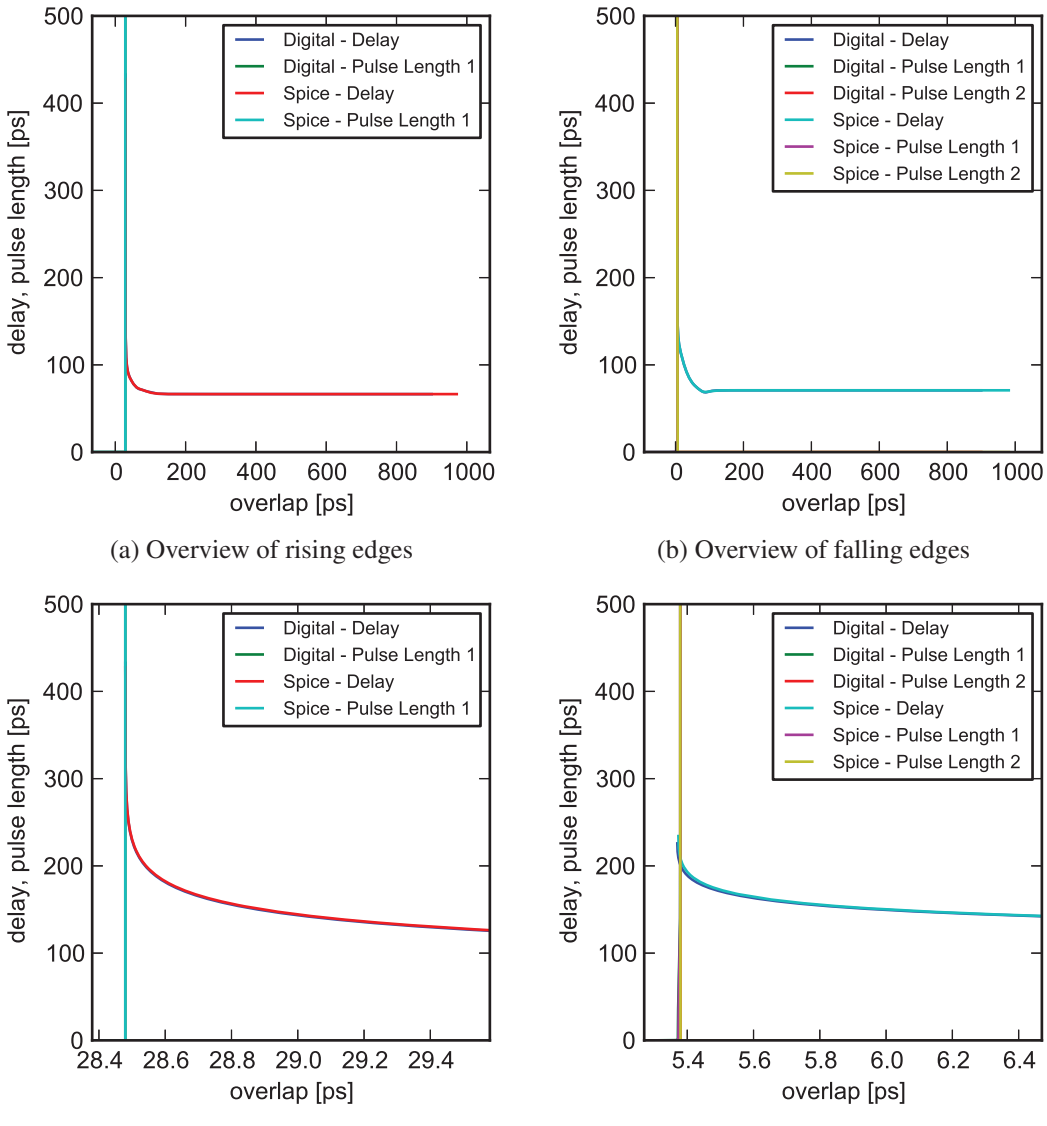


(b) Analogue simulation model of the second latch



(c) Analogue simulation model of the whole flip flop

Figure 8.4: Digital metastability simulation case study



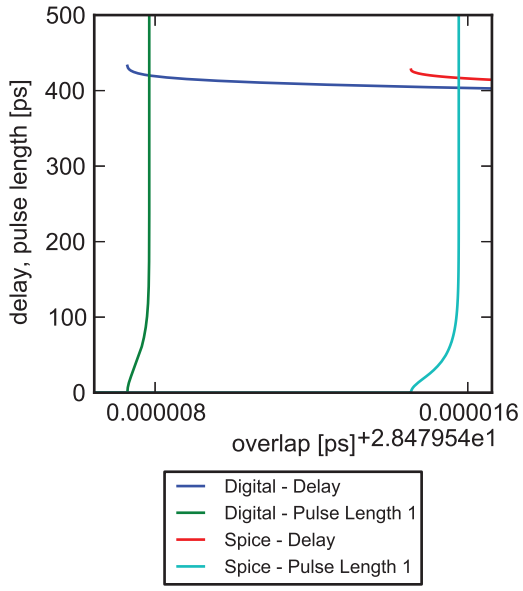
(a) Overview of rising edges

(b) Overview of falling edges

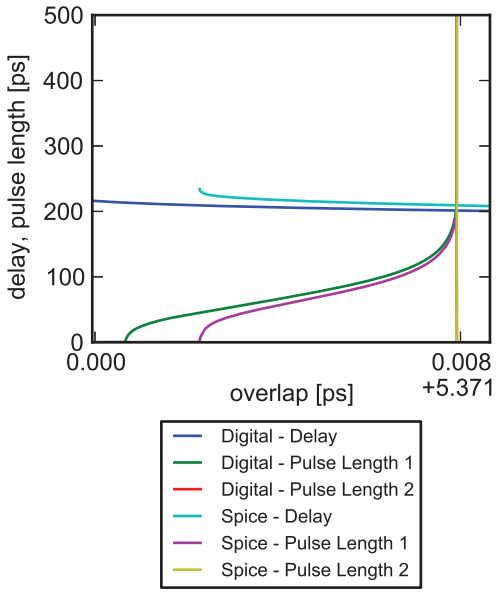
(c) Zoomed view of rising edges

(d) Zoomed view of falling edges

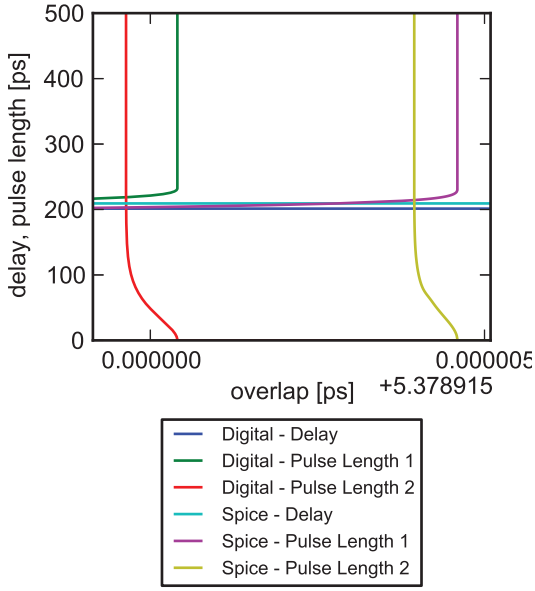
Figure 8.5: Digital metastability simulation case study results (1)



(a) Pulse occurring on rising input edges

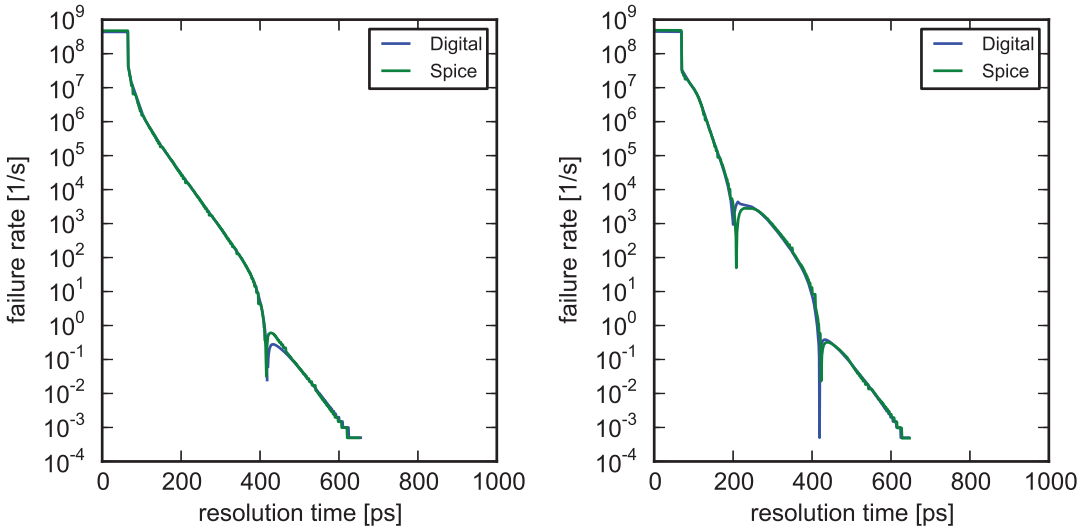


(b) Pulse occurring on falling input edges



(c) Second Pulse occurring on rising input edges

Figure 8.6: Digital metastability simulation case study results (2)



(a) Rising edges (b) Falling edges

Figure 8.7: Failure rate vs. overlap

and best case Spice results of the flip flop see Figure 8.8), making our simulation methodology an attractive choice.

8.4 Mathematical Model

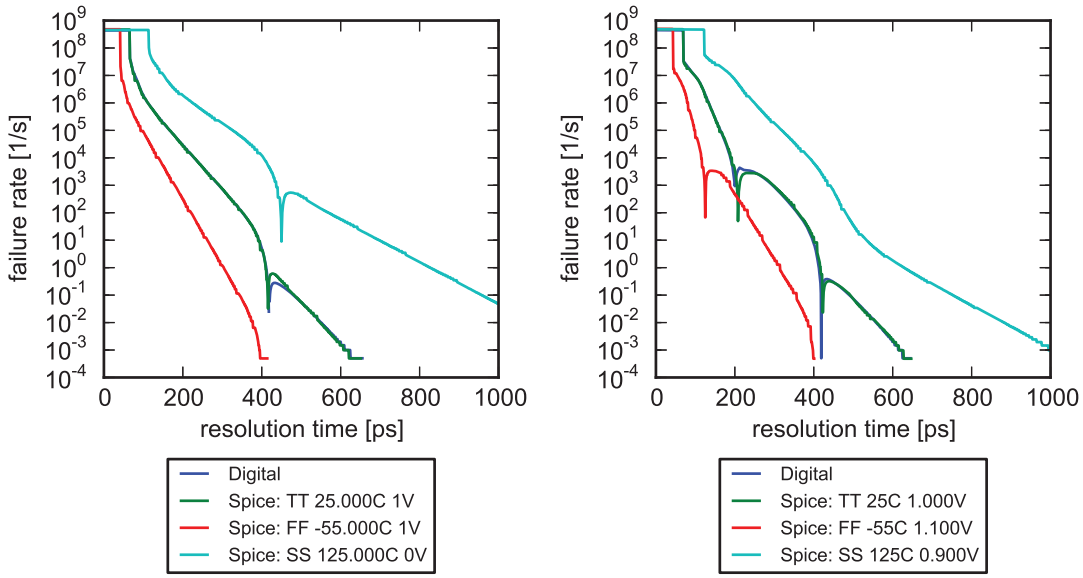
After we have shown the applicability of the new simulation methodology we now want to extend it to be based on a mathematical description by a continuous function rather than a discrete mapping table. To this end we need to express all four characteristics used in our refined model, namely delay, pulse width, enabling delay and pulse propagation.

8.4.1 Delay model

For deriving our delay model we will start with the classical model from [Vee80] that is based on the CMOS circuit already shown in Figure 2.2c.

The storage loop model is composed of the two loop inverters. The inverters are modeled as an amplifier each, followed by an RC-element. The amplifier captures the gain of the inverters, while the RC-element is used to approximate its timing behavior. The additional circuitry of the latch does not influence its basic metastability behavior; its influence can be considered by changing the RC constant.

Figure 8.9 shows the resulting model of the storage loop. In this model v_1 and v_2 are the time dependent voltages at the corresponding nodes, forming the output and input of the storage element, respectively. We will call their initial values v_{1_0} and v_{2_0}. R and C capture the delay parameters, while A models the inverter gain.



(a) Rising edges (b) Falling edges

Figure 8.8: Failure rate vs. overlap for different PVT corners

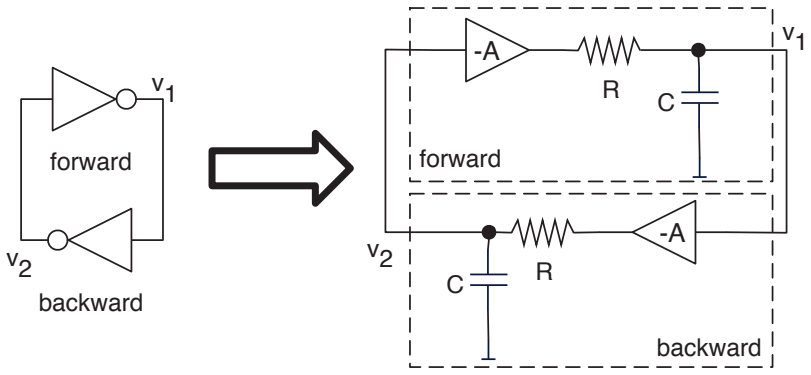


Figure 8.9: Storage loop model

For this model circuit, the differential equations can be stated and a simplification leads to the following equation for the node voltage v_1:

$$v_1 = \frac{v_{1_0} - v_{2_0}}{2} \exp\left(\frac{A-1}{RC}t\right) + \frac{v_{1_0} + v_{2_0}}{2} \exp\left(-\frac{A+1}{RC}t\right)$$

For brevity we only give the resulting equation and not the whole derivation.

As already stated in [Vee80], the second exponential (the decaying one) is only interesting for very small times t and can therefore be ignored in the rest of the analysis. The resulting equation therefore reads:

$$v_1 = \frac{v_{1_0} - v_{2_0}}{2} \exp\left(\frac{A-1}{RC}t\right)$$

It can be further simplified by assuming symmetry within the circuit, meaning that the difference of v_{1_0} and v_{2_0} from the balance point ($V_{dd}/2$) has the same magnitude (ΔV) but different sign. Furthermore we can rename the parameter of the exponential to:

$$\frac{RC}{A-1} = \tau$$

Using these two properties we get:

$$v_1 = \Delta V \exp\left(\frac{t}{\tau}\right) \tag{8.1}$$

At this point we want to express the voltage offset ΔV as a function of the temporal distance of a data transition and the disabling of the latch, which we call input overlap Δt_{in} in the following. This can be done based on the slope θ of the input voltage and the critical overlap Δt_{in_0} in the following way:

$$\Delta V = \theta\left(\Delta t_{in} - \Delta t_{in_0}\right)$$

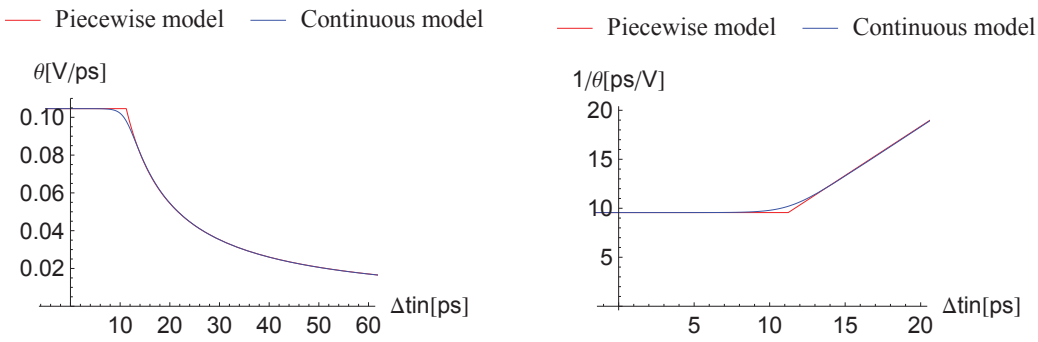
The classical models (like [Vee80]) assume a linear dependence (θ constant) between Δt_{in} and ΔV. This is useful for investigating the behavior around the balance point, but obviously for increasing Δt_{in}, saturation effects are encountered. This is exactly where our approach extends these models: We use a constant θ for very small overlaps Δt_{in} as well, while for larger overlaps a $1/\Delta t_{in}$ dependence is assumed. This allows us to cover arbitrary values for Δt_{in}. The resulting relation reads:

$$\theta \propto \begin{cases} K & \Delta t_{in} \leq \text{limit} \\ \frac{K}{\Delta t_{in}} & else \end{cases}$$

The red plot in Figure 8.10a visualizes this relation. Up to a certain threshold (20 ps in the example), θ is constant and afterward decaying with $1/\Delta t_{in}$. As physical systems do not have abrupt state changes, we smooth the transition between the two areas of the model (blue curve in Figure 8.10a). This additionally helps us with the mathematical modeling, as case separations are always quite unhandy.

For finding a closed expression, we plotted $1/\theta$ (see Figure 8.10b). Curves of this type are known from control and filter theory where they result from plotting terms like

$$\frac{K}{1 + a\Delta t_{in}}$$



(a) Model for parameter θ (b) Model for $1/\theta$

Figure 8.10: Models for input voltage slope θ

on a double logarithmic scale. Compensating for the logarithmic axes and introducing a time shift Δt_{in_0} and an additional scaling factor c, we can express our model for θ as:

$$-\frac{1}{\theta} = \frac{1}{c} \ln\left(\frac{K}{1 + a \exp\left(b\left(\Delta t_{in} - \Delta t_{in_0}\right)\right)}\right) \Rightarrow$$

$$\theta = -\frac{c}{\ln\left(\frac{K}{1+a \exp\left(b\left(\Delta t_{in} - \Delta t_{in_0}\right)\right)}\right)}$$

where K, a, b and c are parameters which are used to fit the result to actual characteristics of a latch. These need to be determined by measurement or simulation. For the complete equation of the node voltage v_1 we insert this expression for θ into Equation (8.1):

$$v_1 = -\frac{c\left(\Delta t_{in} - \Delta t_{in_0}\right)}{\ln\left(\frac{K}{1+a \exp\left(b\left(\Delta t_{in} - \Delta t_{in_0}\right)\right)}\right)} \exp\left(\frac{t}{\tau}\right)$$

This equation is stating the time dependence of the node voltage v_1 parametrized by the input overlap Δt_{in}. To be able to get the output delay Δ' of the latch, we need to define a threshold voltage V_{th} as a reference for delay measurement. Therefore we replace v_1 by V_{th} and t by Δ'. Solving the resulting equation for Δ' leads to:

$$\Delta' = \tau \ln\left(-V_{th} \frac{\ln\left(\frac{K}{1+a \exp\left(b\left(\Delta t_{in} - \Delta t_{in_0}\right)\right)}\right)}{c\left(\Delta t_{in} - \Delta t_{in_0}\right)}\right)$$

The above equation measures Δ' from the start of the metastable state until the output reaches V_{th}. For an end to end delay in a circuit (data-input change to output change) this is still not adequate. To account for the time from the input change until the metastable voltage is reached

an additional constant t_0 is introduced. The equation now reads:

$$\Delta = t_0 + \tau \ln \left(-V_{th} \frac{\ln \left(\frac{K}{1+a \exp\left(b\left(\Delta t_{in} - \Delta t_{in_0}\right)\right)} \right)}{c\left(\Delta t_{in} - \Delta t_{in_0}\right)} \right) \tag{8.2}$$

Equation (8.2) now represents the desired delay model for the latch for all input overlaps Δt_{in}. It extends the state of the art metastability models which are only capable of capturing small input overlaps, and thus facilitates handling late transitions.

Parameter Fitting

When fitting Equation (8.2) to the results of measurements or analogue simulations a two step approach can be used. First the classical, linear part of the model can be fitted to the data. In this first step the characteristics for cases of deep metastability are fixed. As the defining equation for θ is constant for small overlaps Δt_{in}, the behavior near the critical overlap is fully defined by this first step also for the newly proposed model. In a second step, the remaining parameters a, b and c are fixed using the extended Equation (8.2) in the process. This second step handles the non-linear behavior of θ for large overlaps.

8.4.2 Pulse Length Model

As the pulse length $l(p)$ (at least for long pulses) directly corresponds to the metastability time (recall Section 7.4), [Vee80] can be used as basis for this model again:

$$l'(p) = t_0 + \tau \log \left(\frac{v_1}{\left(\Delta t_{in} - \Delta t_{in_0}\right) \theta} \right) \tag{8.3}$$

For short overlaps, however, the pulses die out (see Figure 8.11 for an example) and therefore we need to attenuate short pulses in our model. This is achieved by weighting the metastability time by a square root function leading to our pulse length model:

$$l(p) = a \sqrt{b\left(\Delta t_{in} - \Delta t_{in_{02}}\right)} \left(t_0 + \tau \log \left(\frac{v_1}{\left(\Delta t_{in} - \Delta t_{in_0}\right) \theta} \right) \right) \tag{8.4}$$

The selection of the square root function was by visual inspection. The parameters a and b are used to scale the square root function, while the parameter $\Delta t_{in_{02}}$ is used to shift it to correspond with the start of the pulse occurrences.

Please note that the pulse length has to be set to zero for overlaps smaller than $\Delta_{tin_{02}}$, as the model will not return a valid result (the square root function is not defined – negative argument – in the domain of the real numbers).

The fitting operation first the classical model is matched to the long pulses. As a second step, the whole dataset is used to fit the complete model (Equation (8.4)).

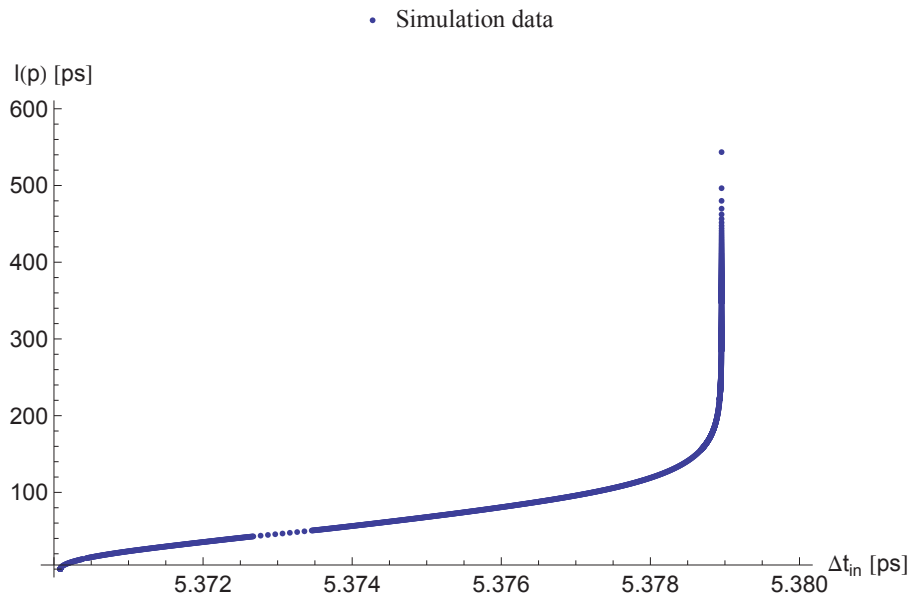


Figure 8.11: Plot of simulated pulse lengths for one of the sample latches

8.4.3 Enable Delay Model

As for the pulse length model we again derived the mathematical representation of the enable delay Δ_{en} by visual inspection of the data (Figure 8.3a shows an example). The data has the same overall shape as parameter θ in the case of the delay model and therefore we can state that the data corresponds to a term of the form

$$\frac{K}{1 + a\Delta t_{in}}$$

but in linear instead of double logarithmic scale. Compensating for the different scale and adding scaling parameters (a, b, c, d and f) as well as a shift along the x axis (Δt_{in_0}), the model can be written as:

$$\Delta_{en} = f + d \cdot log\left(\frac{c}{1 + ae^{b\cdot(\Delta t_{in} - \Delta t_{in_0})}}\right) \tag{8.5}$$

The fitting of the enable model is straight forward without any additional sub steps.

8.4.4 Pulse Propagation Model

For the pulse propagation model of the transparent latch we again derived a mathematical model for the ratio between output- and input-pulses ($\frac{l(p_{out})}{l(p_{in})}$) by inspecting the resulting data from the simulations (see Figure 8.3b). As the ratio follows a $1 - e^{-x}$ function, we derived the following model by adding scaling (a and b) and shifting (c) parameters. Additionally the model is clamped

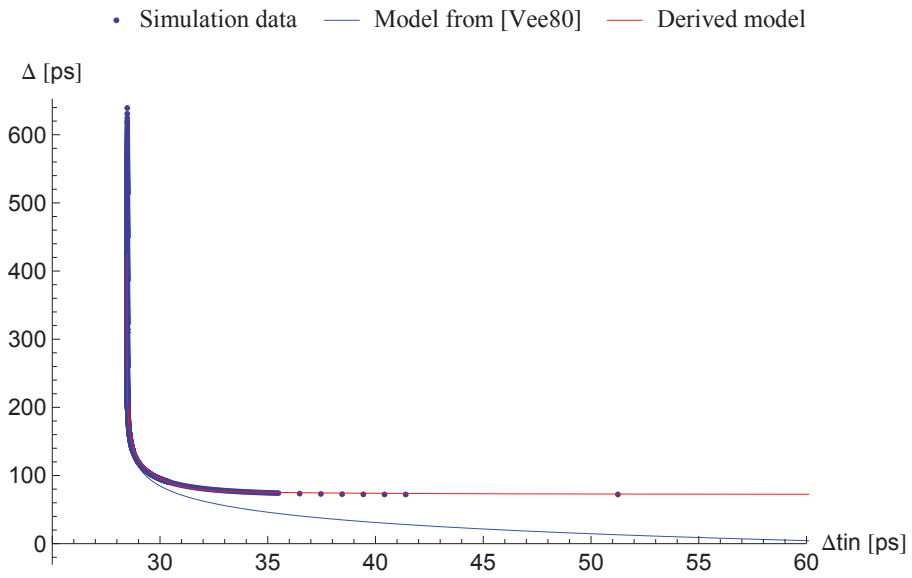


Figure 8.12: Model-fitting master latch, rising edges (1)

to zero to avoid pulses of negative length (as they are physically impossible). The model reads:

$$\frac{l(p_{out})}{l(p_{in})} = \max\left(0, b\left(1 - e^{-a(l(p_{in})-c)}\right)\right) \tag{8.6}$$

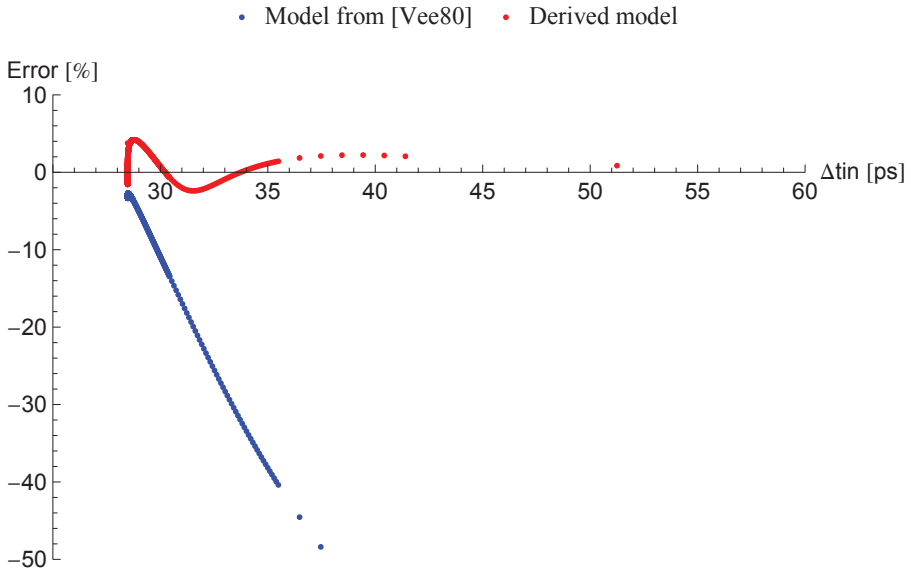
For fitting the model to the data, first the parameter b is set to the maximum ratio of the output pulse lengths with their corresponding input pulse. Afterward a standard fitting operation is used to determine the remaining parameters.

8.4.5 Example

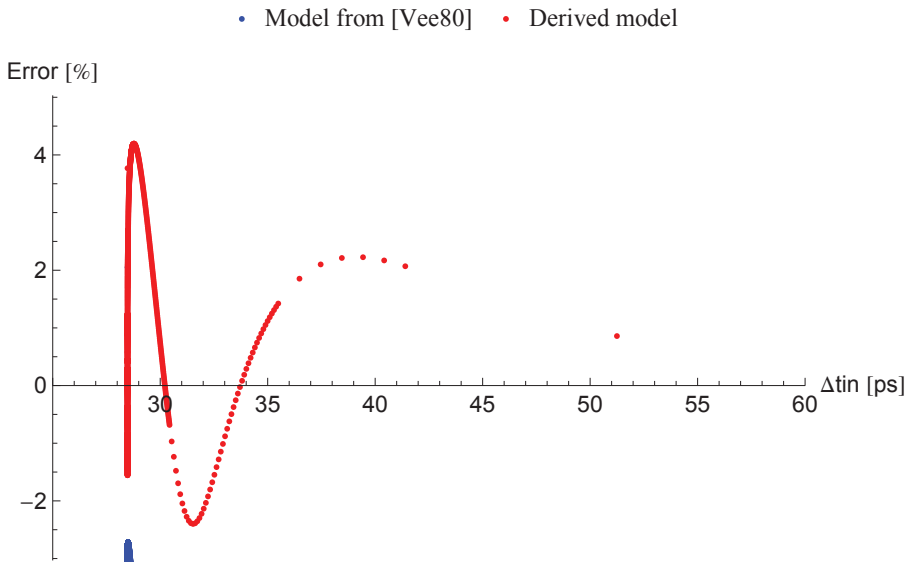
To illustrate the viability of our approach, we have fitted the simulation results of the latches we used to build the flip flop implementation in Section 8.3.

Master latch - rising edges: The fitting is done on the delay between the D-input of the latch and its output qb (before the additional inverters). We then used Mathematica to fit the results to the model from [Vee80]. The resulting parameters can be found in Table 8.1. Figure 8.12 shows the resulting curve (blue trace) and the simulated data.

Using τ and Δt_{in_0} from the previous fitting operation and setting c to $1V/ps$, we were able to fit the additional parameters (see Table 8.1) leading to the following result: The relative error compared to the simulation result is between -2.4% and 4.2%. Refitting t_0 was necessary as the curve shape is quite different to the first model. Figure 8.12 shows the fitting result (red trace), while Figure 8.13 shows the relative error of both models (classical model: blue trace, derived model: red trace).



(a) Relative error for fitted model



(b) Relative error for fitted model (zoomed)

Figure 8.13: Model-fitting master latch, rising edges (2)

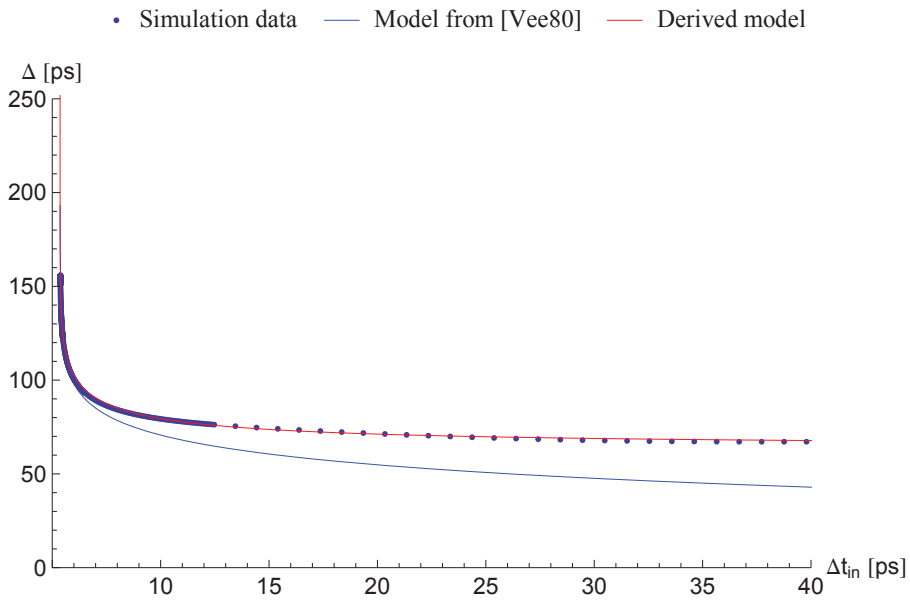


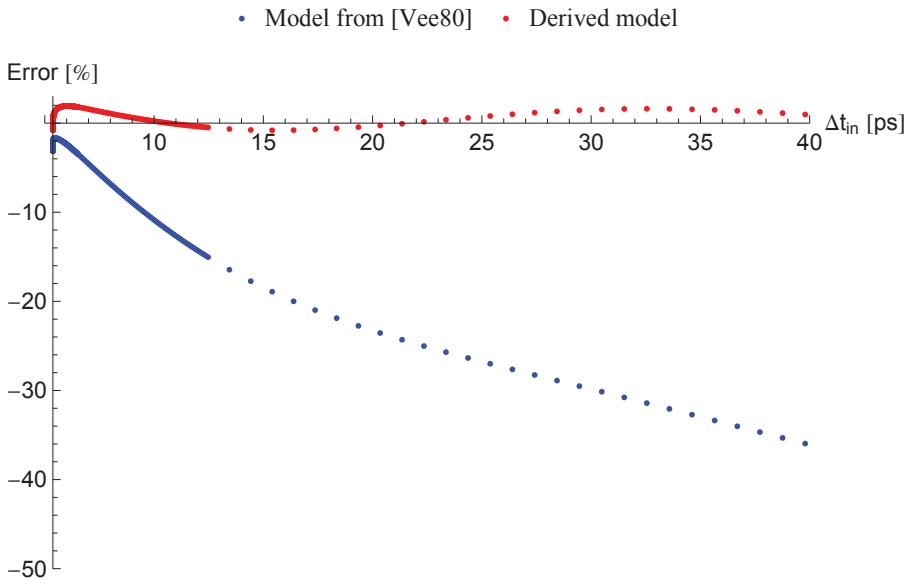
Figure 8.14: Model-fitting master latch, falling edges (1)

Master latch - falling edges: The fitting operation of the output delay is the same as for the rising edges. Table 8.1 summarizes the parameters found for the two models while Figures 8.14 and 8.15 show the resulting traces (Model from [Vee80] in blue, derived model in red). The relative error of the derived model is within [-4.2%, 1.93%]. It is important to note that the delay behavior in case of the falling edges does not adhere to the normal metastable operation (recall Section 7.4), as pulses occur at the output. Nevertheless the model fits quite nicely.

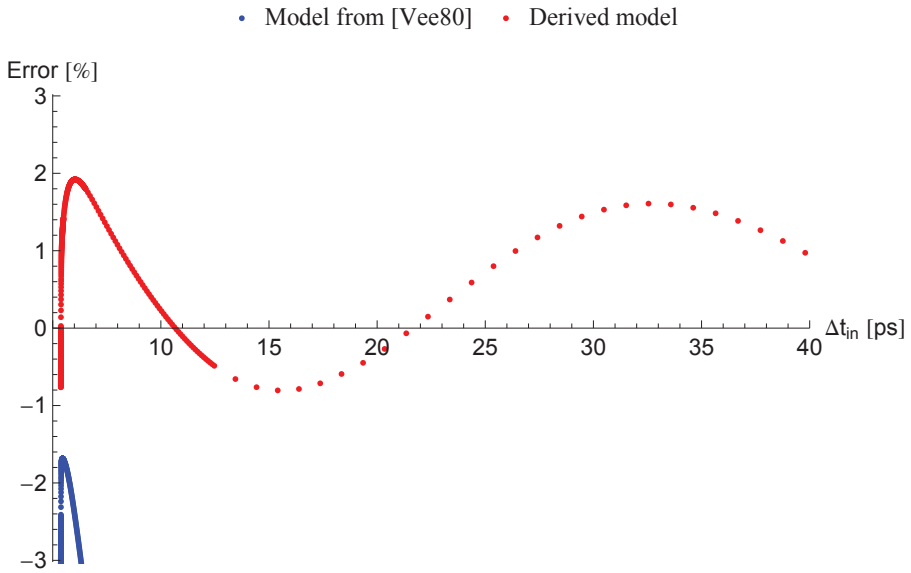
An additional step is required for the falling edges though. As in this case pulses may occur at the output also the pulse length model must be fitted. First only the model from [Vee80] is processed. To get a better results, only the data for the longer pulses is used in the fitting operation. In a second step, using the whole dataset, the remaining parameters are fixed to the complete model. The result is visualized in Figure 8.16 and the resulting parameters can be found in Table 8.2. Please note that the parameter θ is negative, mirroring the exponential around the y-axis.

The error made by the fitting operation is in the interval of [-18.65%, 32.49%]. The large relative errors are, however, for very small pulse lengths (see Figure 8.16b) and therefore the corresponding absolute errors are rather small (between $-7.4ps$ and $3.97ps$).

Master latch - enable delay: The fitting of the enable delay is straight forward. We used Mathematica to fit the simulation results with the model. The resulting parameters can be found in Table 8.3. The results are shown in Figures 8.17 and 8.18. The relative error is between -3.02% and 1.82% for rising edges and -0.81% and 1.81% for falling edges.



(a) Relative error for fitted model



(b) Relative error for fitted model (zoomed)

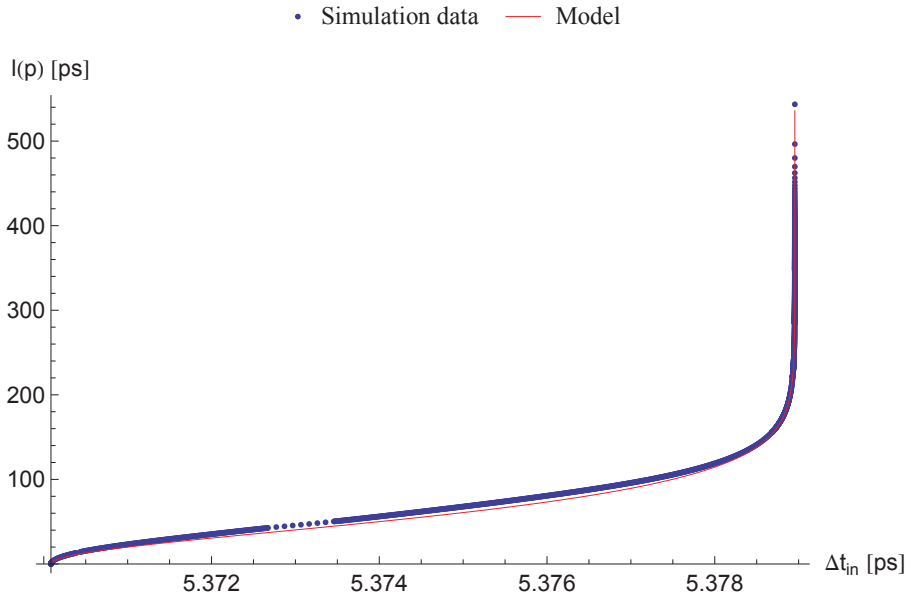
Figure 8.15: Model-fitting master latch, falling edges (2)

Model	Parameters
Rising edges, model from [Vee80]	$\tau = 26.47801329502695ps$, $\theta = 0.4253293290559774V/ps$, $t_0 = 91.4993771550824ps$, $\Delta t_{in_0} = 28.47951857742057ps$
Rising edges, derived model	$\tau = 26.47801329502695ps$, $\Delta t_{in_0} = 28.47951857742057ps$, $c = 1V/ps$, $K = 0.001362104611316363$, $a = 0.00813718301624549$, $b = 1.777348213901242/ps$, $t_0 = 74.79068606930909ps$
Model from [Vee80]	$\tau = 13.81707488022531ps$, $\theta = 0.00091794172244458 6V/ps$, $t_0 = 4.851570307567965ps$, $\Delta t_{in_0} = 5.361518447368466ps$
Derived model	$\tau = 13.81707488022531ps$, $\Delta t_{in_0} = 5.361518447368466ps$, $c = 1V/ps$, $K = 0.0002243168874663145$, $a = 1.955973312492632$, $b = 1.009571829265592/ps$, $t_0 = 74.04984003416182ps$

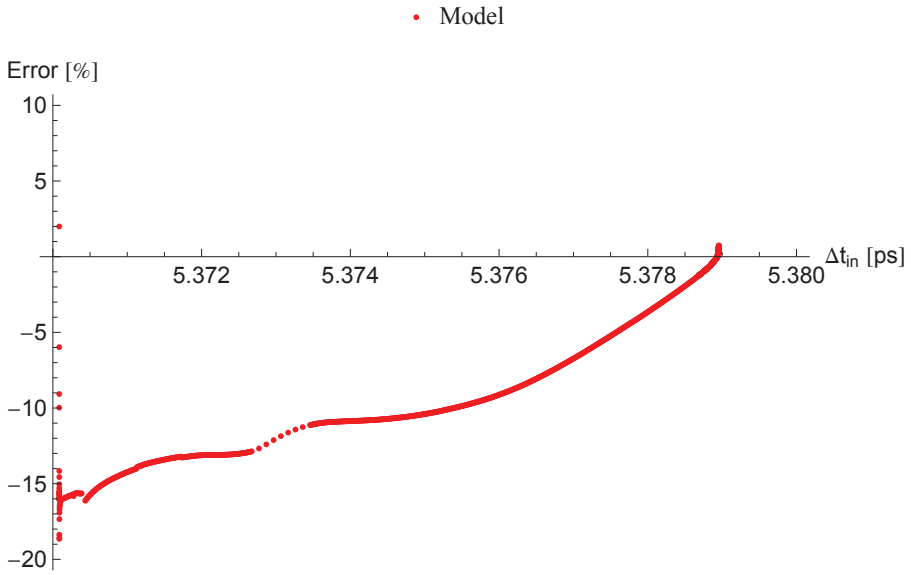
Table 8.1: Delay model parameters for master latch

Model	Parameters
Model from [Vee80]	$\tau = 27.74611752417537ps$, $\theta = -1.007690926775137V/ps$, $t_0 = -52.80599300197188ps$, $\Delta t_{in_0} = 5.378954923610285ps$
Derived model	$a = 10.444777562988$, $b = 1.04323697683022/ps$, $\Delta t_{in_{02}} = 5.370085372778102ps$

Table 8.2: Pulse length model parameters for master latch

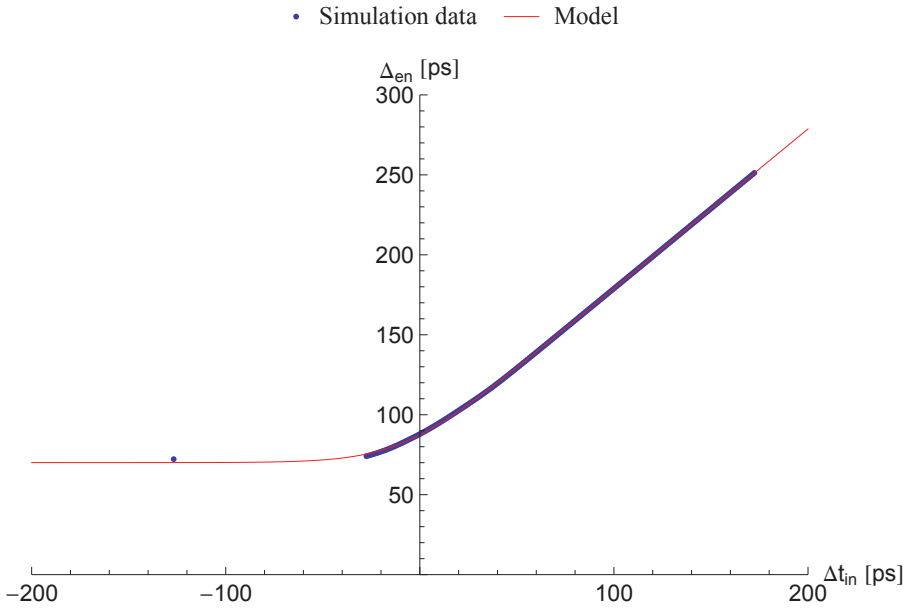


(a) Simulation result and fitted model

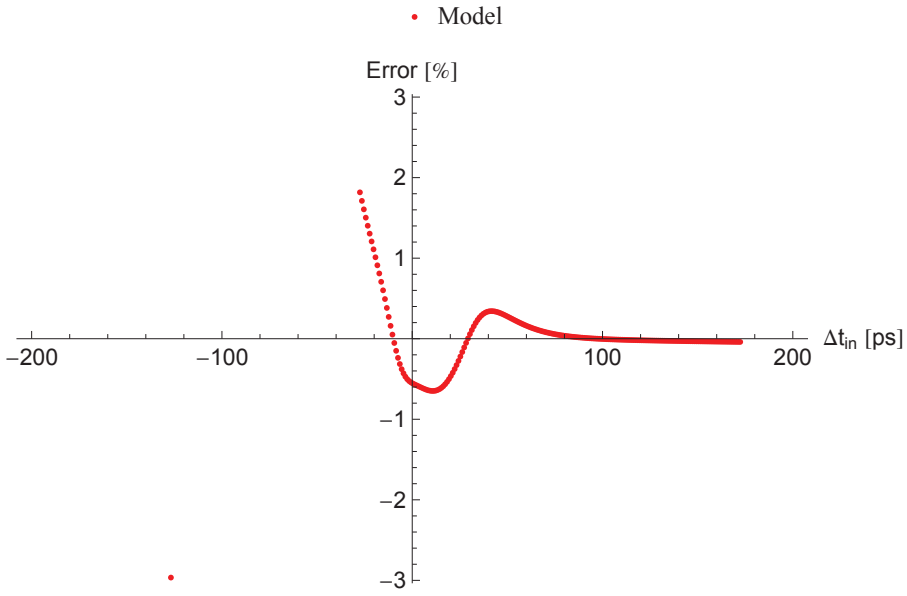


(b) Relative error for fitted model

Figure 8.16: Model-fitting master latch, falling edges (pulse model)

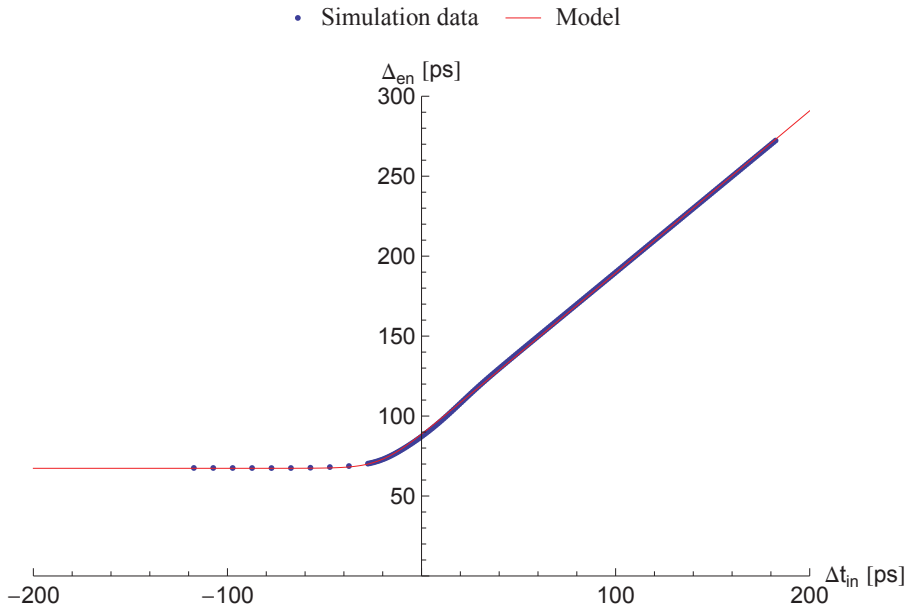


(a) Simulation result and fitted model

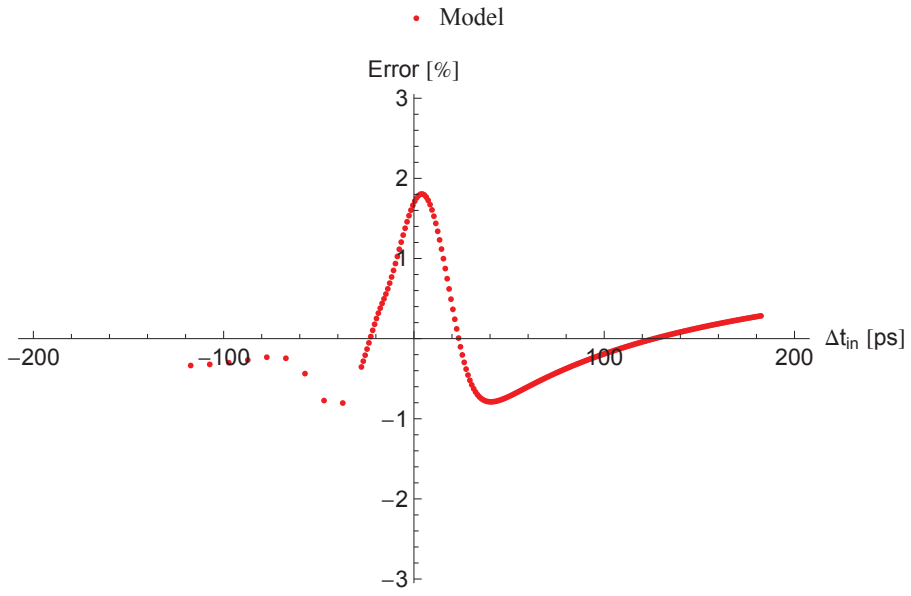


(b) Relative error for fitted model

Figure 8.17: Model-fitting master latch, rising edges (enable delay)



(a) Simulation result and fitted model



(b) Relative error for fitted model

Figure 8.18: Model-fitting master latch, falling edges (enable delay)

Model	Parameters
Rising edges	$a = 0.06681037672938762$, $b = 0.05606164944588496/ps$, $c = 0.1185656052101724$, $d = -17.8146447189506ps$, $\Delta t_{in_0} = -57.19904147922834ps$, $f = 32.06966829664684ps$
Falling edges	$a = 5.545778746195794$, $b = 0.1283730787357248/ps$, $c = 1.363951081917175$, $d = -7.897430206467983ps$, $\Delta t_{in_0} = -7.17654789859909ps$, $f = 69.77596748105465ps$

Table 8.3: Enable model parameter for master latch

Model	Parameters
High polarity	$a = -0.2896820384240702/ps$, $b = 0.975616068022728$, $d = 48.27944906417747ps$
Low polarity	$a = -0.3014428445897257/ps$, $b = 1.038321515382784$, $d = 44.93527690304164ps$

Table 8.4: Pulse propagation model parameters for the transparent master latch

Master latch - pulse propagation for the transparent latch: The parameters for the master latch can be found in Table 8.4, while the results are visualized in Figures 8.19 and 8.20.

Slave latch: The fitting operation for the slave latch is the same as for the master latch. For brevity reasons it is omitted here, but can be found in Appendix B.

8.4.6 Simulation Using the Mathematical Model

The ability of the new model to capture the input- to output-delay behavior of arbitrary input overlaps can be exploited as a replacement for the data table in the simulator. In contrast to the interpolation method the input- to output-delay of the gates is not determined by linear interpolation between two neighboring simulation results but calculated depending on the input overlap using a continuous function. A repetition of the simulations from Section 8.3 using the mathematical model leads to the failure rate plots in Figure 8.21.

The digital simulation was performed with a minimum step size of $10^{-27}s$, while the analogue simulation uses a minimum step size of $10^{-21}s$. The difference is clearly visible in the results when looking at overlaps bigger than $600ps$. Therefore it is possible to predict metasta-

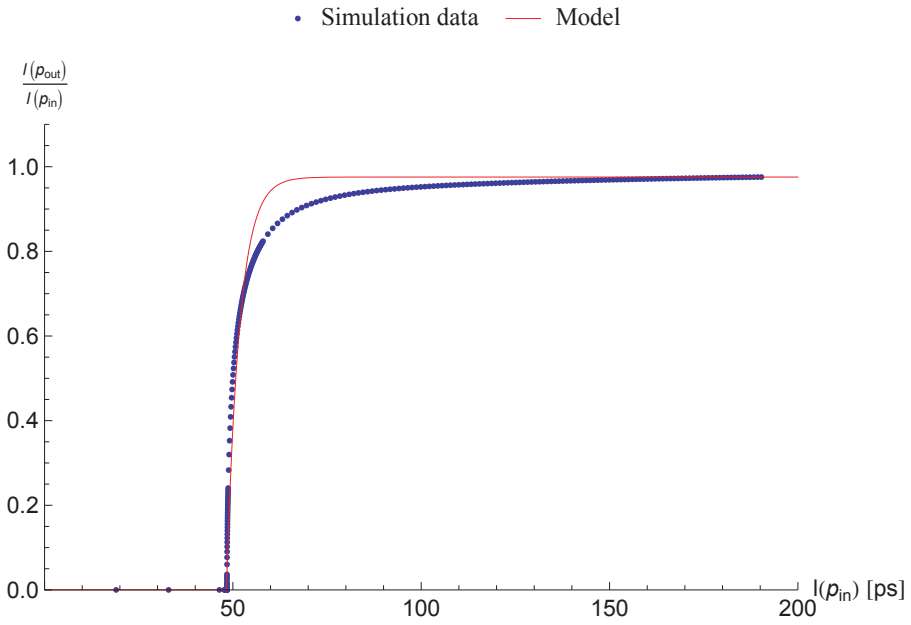


Figure 8.19: Model-fitting master latch, rising edges (pulse propagation)

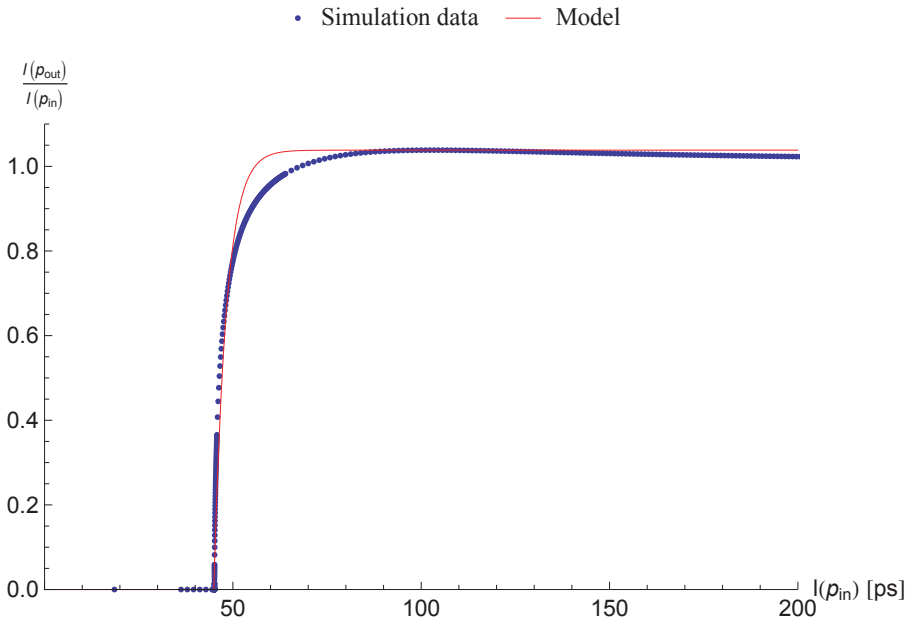
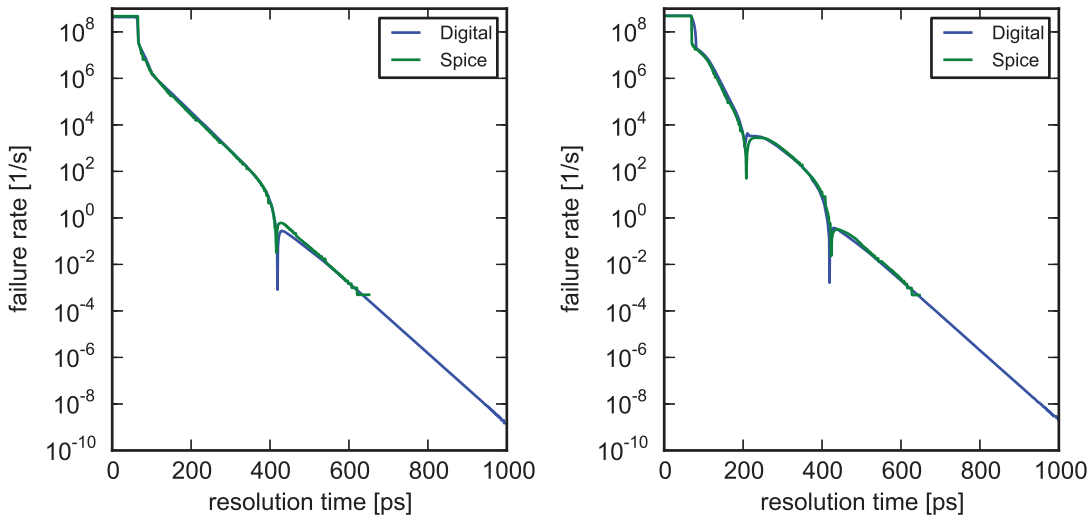


Figure 8.20: Model-fitting master latch, falling edges (pulse propagation)



(a) Rising edges (b) Falling edges

Figure 8.21: Comparison of failure rate vs. overlap plot between analogue simulation and mathematical model based simulation

bility which is deeper as in the underlying dataset.

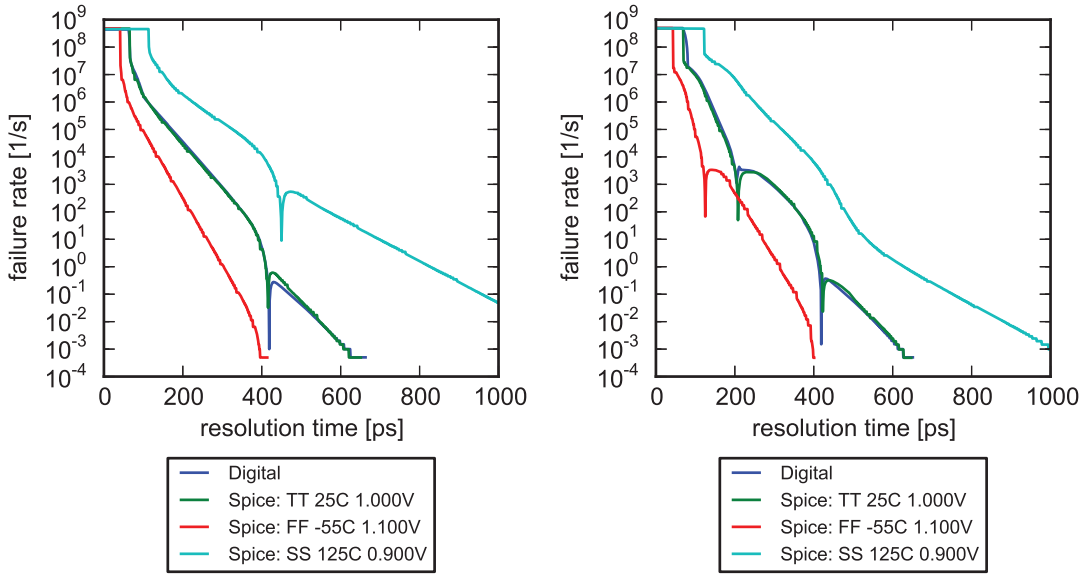
To be able to assess the deviations between the analogue and the digital, mathematical model based simulation, we have again compared the results with the PVT corners (Figure 8.22) using a minimal step size of $10^{-21}s$ for all simulations. We can conclude that the deviations of our simulation model are much smaller than the ones introduces by PVT variations.

8.5 Summary and Limitations

We have seen that in many cases metastability can be sufficiently well expressed in a digital model giving a big advantage over the error state simulation model. Depending on the input overlap, different input to output delays are generated in the simulation. These delays match the results of the analogue simulations quite well. Due to the models ability to interpolate, the characterization of the element parameters in Spice can be done in relatively coarse steps. Nevertheless the mapping of an analogue problem into the digital domain does not come for free.

The deviations of the models are small compared to the PVT variations and therefore the behavior of the flip flop is captured quite well by our approach. The mathematical model enables the extrapolation of the metastable state and supports simulation of very small overlaps (an example using a minimum step size of $10^{-27}s$ has been shown). Due to the increased resolution of the simulation time and the higher speed compared to the analogue Spice simulations, the investigation of very precise overlaps becomes viable.

Most importantly, metastability of the inner storage loop is only converted into late transition or pulses, if the threshold voltage of the output buffer is sufficiently different from the metastable



(a) Rising edges

(b) Falling edges

Figure 8.22: Failure rate vs. overlap for different PVT corners (mathematical model)

voltage. Otherwise the buffer will more or less convey the undefined voltage to its output. This behavior (as e.g. seen in Chapter 5) is of course not correctly captured in our model that is digital only. However, if low- or high-threshold inverters are used (which is often the case for synchronizing flip-flops), such analogue outputs can be safely avoided.

Currently the simulator is optimized for handling synchronizers. However, it is also possible to simulate bigger, more complex circuits. For such implementations it is important to keep the error propagation in mind. As, in the metastable state, any difference in input overlap is amplified exponentially to the output, a small error at the input of a metastable circuit element will be extended significantly at the output. To be able to achieve meaningful results, the number of metastable stages connected in series must therefore be limited. Nevertheless it is important to note that the variations in the production process will outweigh these imprecisions in the predictions of the model significantly, which was already shown by our measurements (the τ of different latches on the same chip may vary significantly) and simulations.

Additionally, in the current implementation, it is not possible to simulate the occurrence of short pulses on the enable input of a latch. Furthermore the occurrence of small pulses exactly at the enable or disable event may cause undesired effects. These borderline conditions can not be handled with the current model data (overlap vs. delay and pulse length, transparent pulse forming and enable delay data). To be able to incorporate these effects, the underlying model must be extended and additional characterizations using Spice or measurements must be performed.

In spite of all these limitations the proposed model, even in its current shape, is definitely

a step forward compared to the available simulation methodologies: While, as we have shown, available digital simulators cannot properly handle setup-/hold-violations at all, Spice simulations are not applicable to circuits comprising more than a hand full of logic components, due to the mere explosion of computational effort. Here our approach seems to represent a first viable solution.

9

Conclusion and Future Work

The digital simulation model developed in this thesis can be used to simulate the failure rate of digital circuits with high accuracy while being much faster than traditional, analogue simulations. The model captures the metastability behavior of a component and is calibrated to a specific technology by an initial Spice simulation.

We have shown that the basic metastability characteristic of several digital storage elements, namely D-latches, D-flip flops, C-elements and RS-latches) are qualitatively the same. By running elaborate Spice simulations the dependence of the overlap of the input signals to the output response has been captured. These simulations have shown that in most cases, the metastable response of these elements can be modeled as late transitions and pulses. This model has its limits, if output inverters matched to the storage loop are used and the interconnect and input stage of the next cell have a too low gain.

Using digital measurements on D-flip flops, so called late transition detection, we were able to verify the simulation results. By extending the state of the art measurement circuits to capture not only the sum of the failures but by splitting the result into different cases (rising and falling edges as well as low and high polarity pulses), we could map the qualitative behavior recorded by the measurements to accompanying simulation results. Therefore it was possible to get a more detailed understanding of the inner working of the late transition detection and how it maps the characteristic of the internal metastability parameters to the output of the flip flop.

Furthermore we extended the late transition detection scheme to also incorporate the measurement of asynchronous elements. We managed to characterize Muller C-elements and RS-latches without the need of any external, expensive specialized equipment, like high precision pulse generators or high bandwidth oscilloscopes.

The ability of short transient pulses to pass through several stages of an elastic pipeline without being latched was demonstrated by virtue of Spice simulation. As the required window of critical charge is small and decreases exponentially with the stage number, the probability for transient propagation is quite small, however, the possibility exists. Our analysis has shown that the output stage of the Muller C-elements in the pipeline has a major impact on the propagation behavior.

Future work Currently our digital simulation model only works for D-latches. Two instances of this model can be used to implement flip flops in our simulation environment but this characterization is only true, if the coupling of the two latches can be abstracted on the digital level. If analogue phenomena occur at this boundary, the concatenation of two D-latches is not sufficient and a specialized model of the whole D-flip flop is required. We are currently working on such a model and we are also developing a model for Muller C-elements.

Up to now, all our measurements were performed on FPGAs only. In a future project we want to build a specialized ASIC to be able to verify the function of our measurement architecture on this target platform as well. Using an ASIC will also increase the correlation between the simulations and the measurements as we will be able to simulate the exact structure of the circuit. Our current simulations only used models of the same feature size and are therefore only comparable with measurements on a qualitative level.

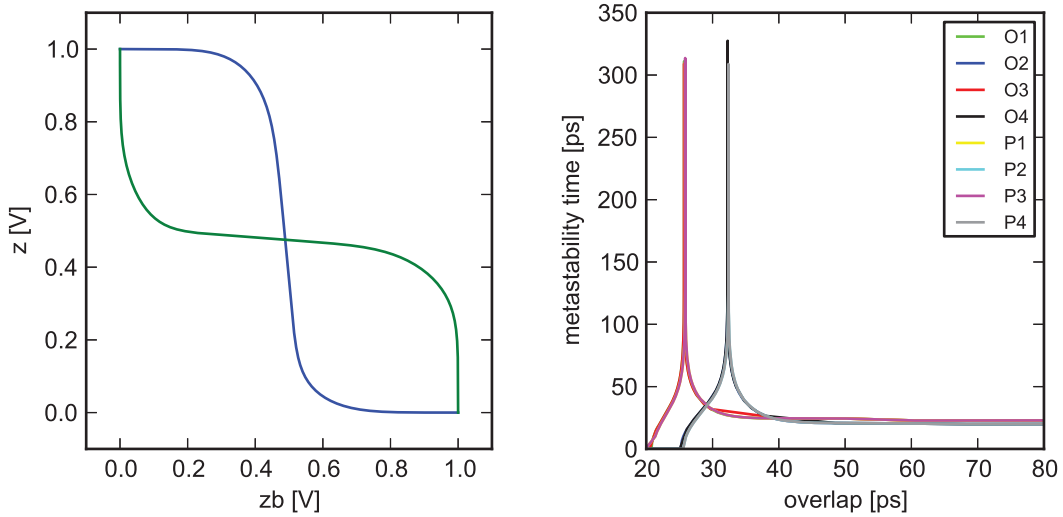
The results of this thesis were done for the 90nm technology node. Future experiments will be performed on devices built in a smaller node, like e.g. 28 nm FPGAs. Preliminary measurement results of flip flop on an FPGA built using this feature size are promising but not yet fully analyzed. We also intend to build the specialized measurement ASICs in multiple technology nodes. This will enable us to study the effects of miniaturization on the metastable parameter.

Our current simulation model was only verified for two D-latch stages. This analysis has to be extended to multiple stages in the future to ensure the validity of the model for multistage synchronizers. Furthermore the current simulation models, as well as state of the art failure prediction models, are completely deterministic. We intend to extend these models by introducing error bound and confidence intervals to input signals and device parameters. The simulations and calculated failure predictions will then incorporate this additional information to give such bounds for their results.

A Additional Simulation Results

This appendix contains the results of additional hSpice simulations performed on the different Muller C-element implementations using an industrial 90 nm technology with a nominal VDD of $1V$. For each implementation several different output stages are used. The result is visualized using the DC characteristic of the element, the dependence of the metastability time on the input overlap and the signal trace of an upper- and lower-bound for the critical overlap.

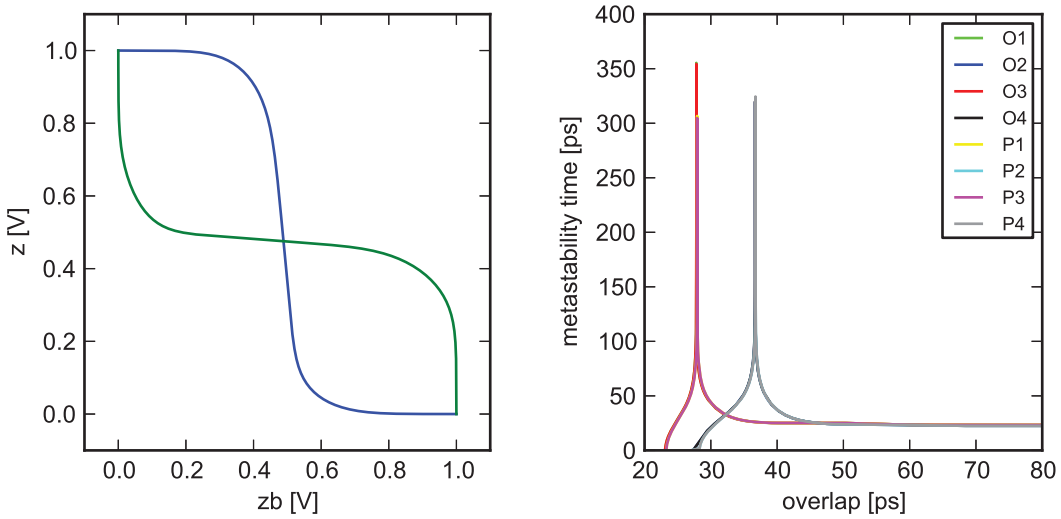
A.1 van-Berkel Muller C-Element



(a) DC characteristic

(b) Response time vs. input overlap

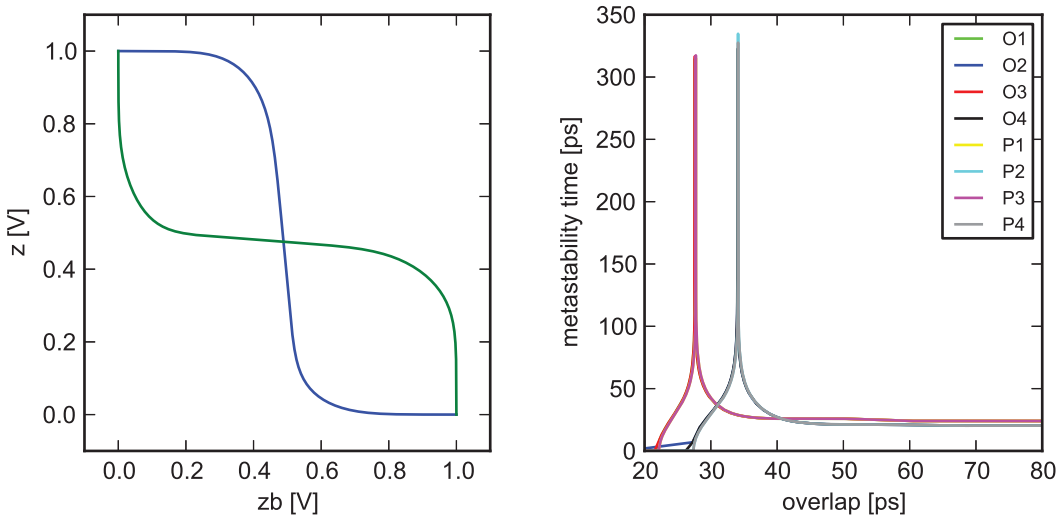
Figure A.1: Results of the van-Berkel implementation with a matched output inverter (1)



(a) DC characteristic

(b) Response time vs. input overlap

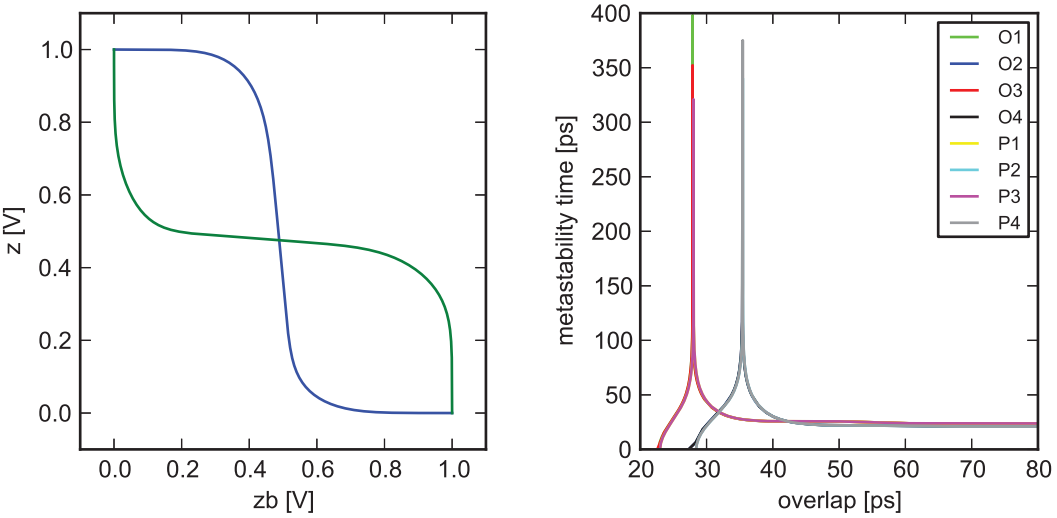
Figure A.2: Results of the van-Berkel implementation with a low threshold output inverter (1)



(a) DC characteristic

(b) Response time vs. input overlap

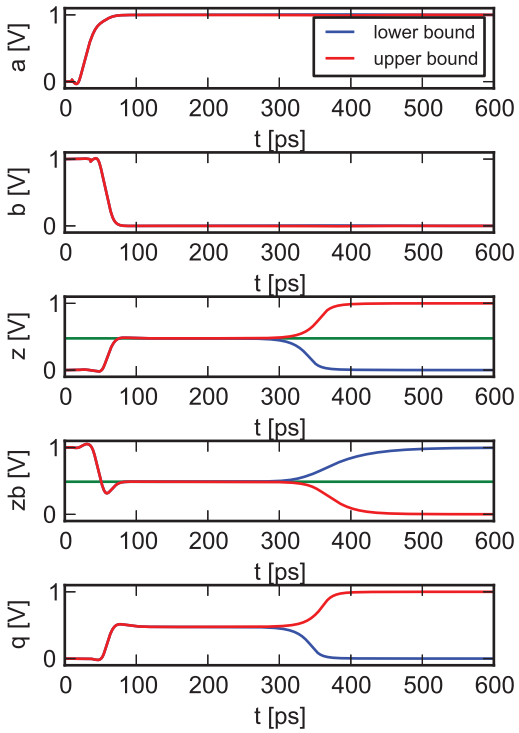
Figure A.3: Results of the van-Berkel implementation with a high threshold output inverter (1)



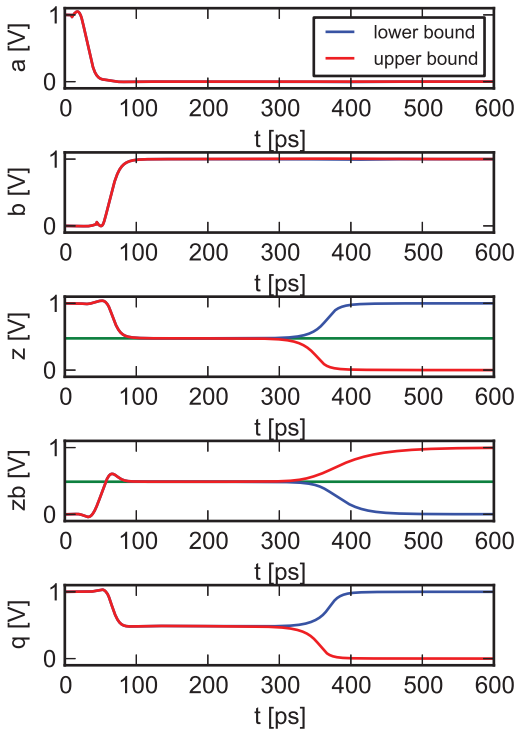
(a) DC characteristic

(b) Response time vs. input overlap

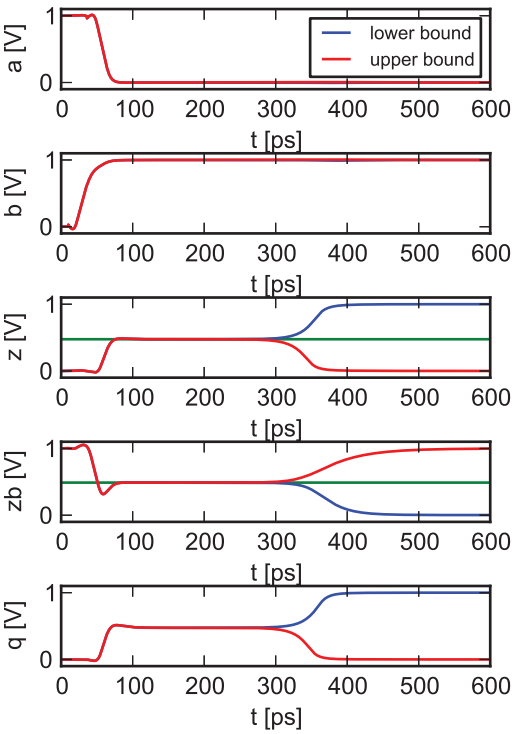
Figure A.4: Results of the van-Berkel implementation with a Schmitt-trigger output (1)



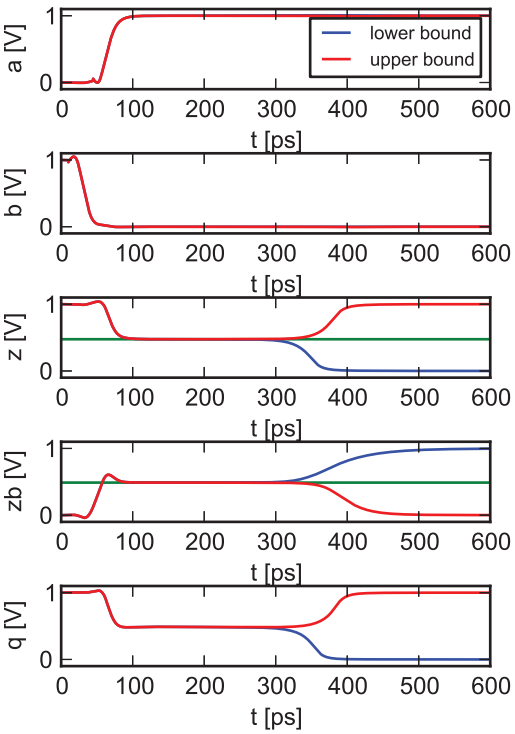
(a) Signal trace (case O1)



(b) Signal trace (case O2)

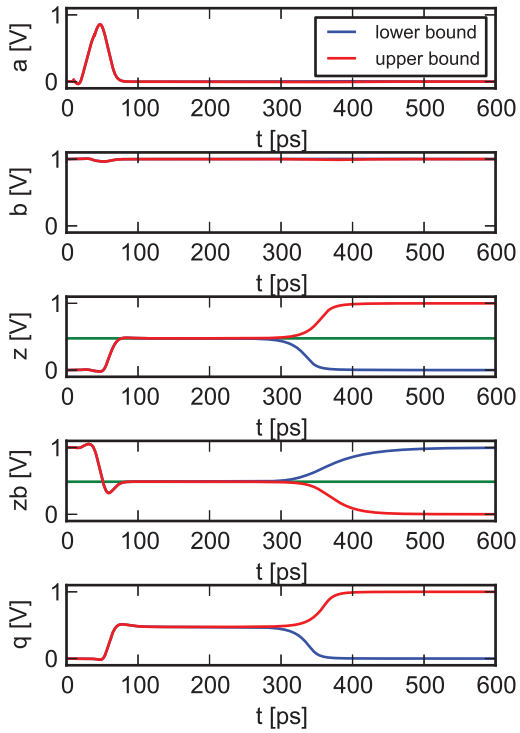


(c) Signal trace (case O3)

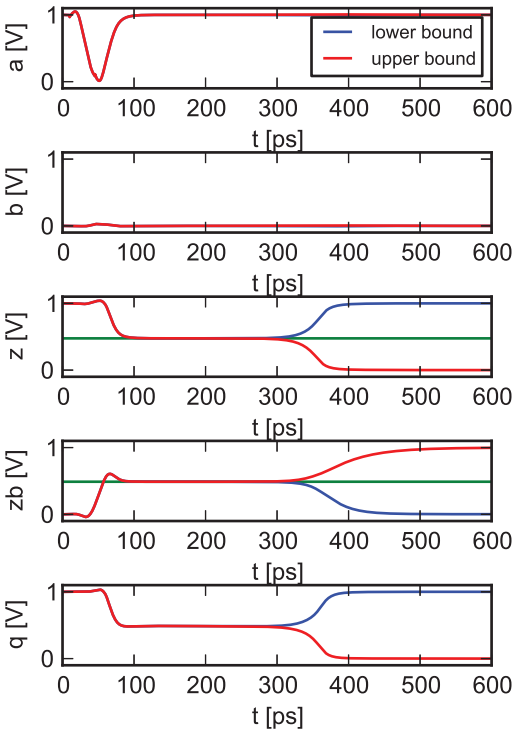


(d) Signal trace (case O4)

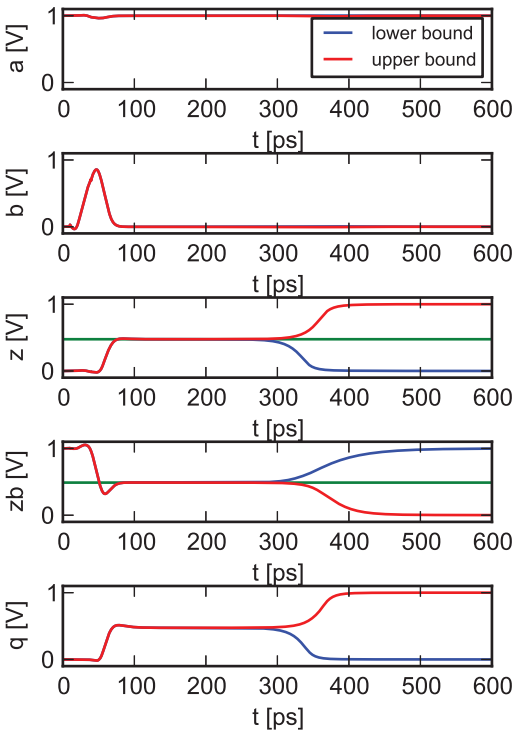
Figure A.5: Results of the van-Berkel implementation with a matched output inverter (2)



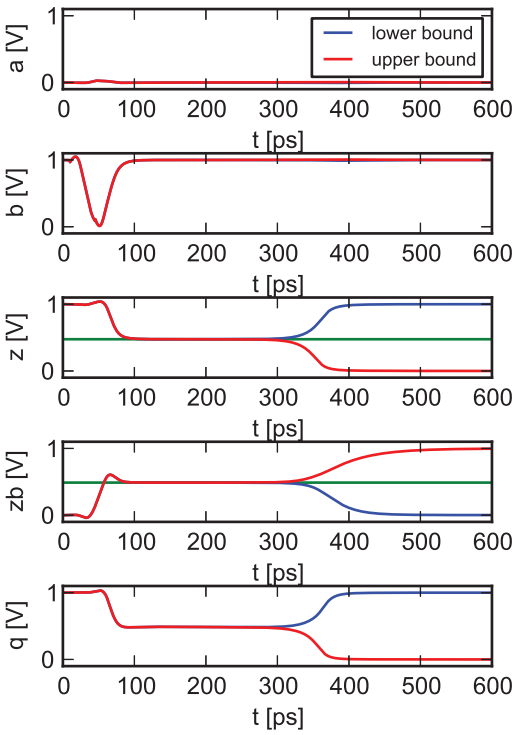
(a) Signal trace (case P1)



(b) Signal trace (case P2)

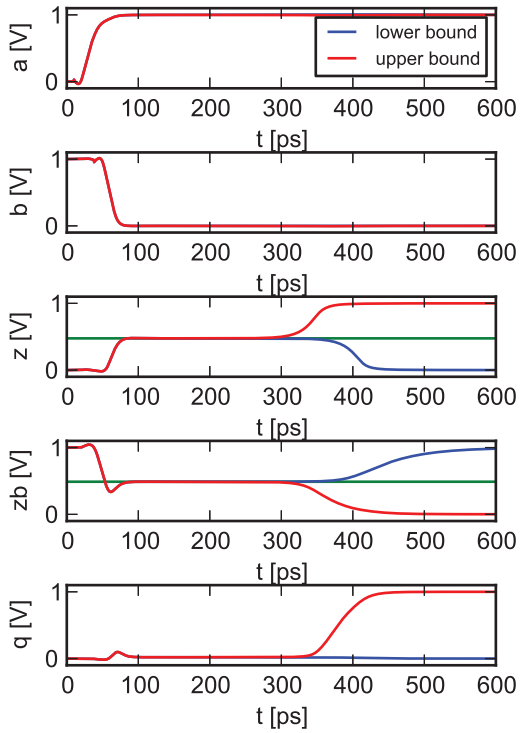


(c) Signal trace (case P3)

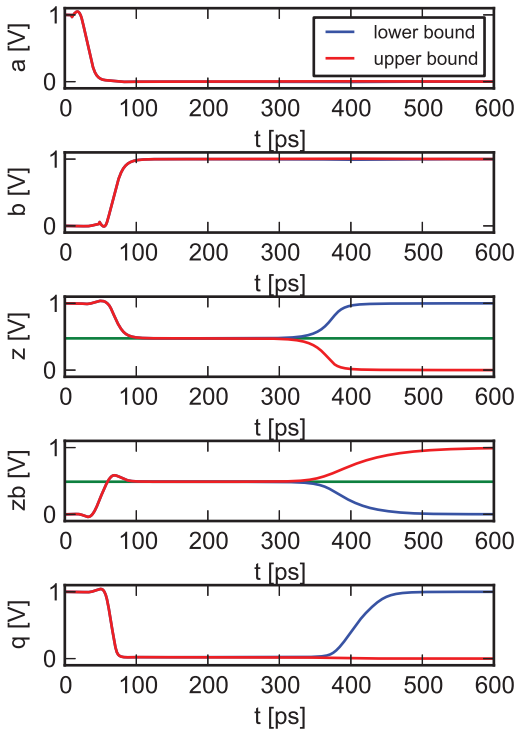


(d) Signal trace (case P4)

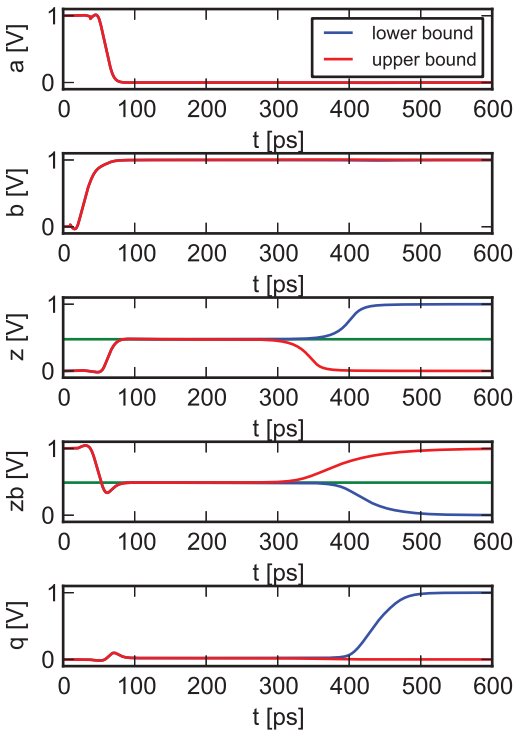
Figure A.6: Results of the van-Berkel implementation with a matched output inverter (3)



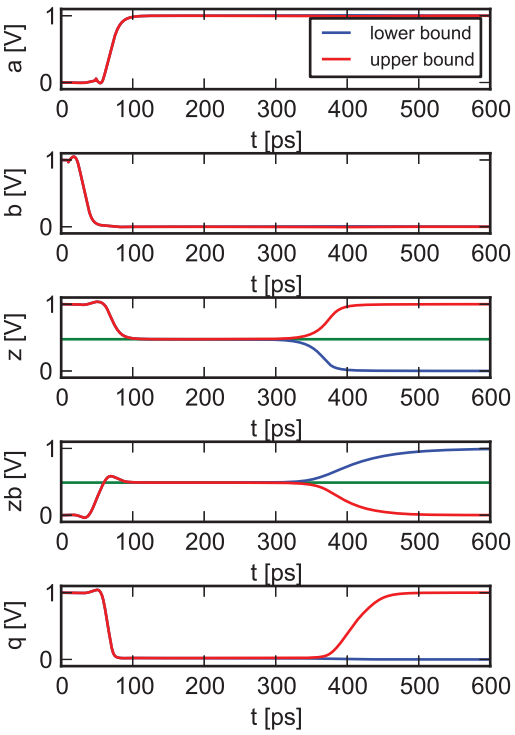
(a) Signal trace (case O1)



(b) Signal trace (case O2)

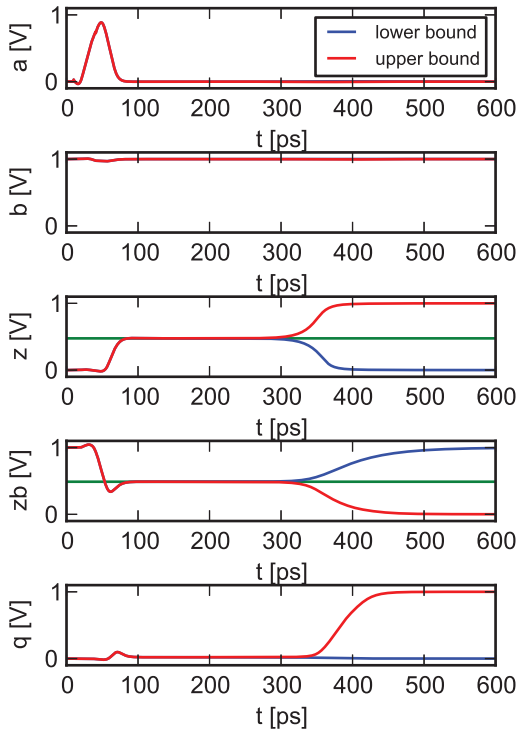


(c) Signal trace (case O3)

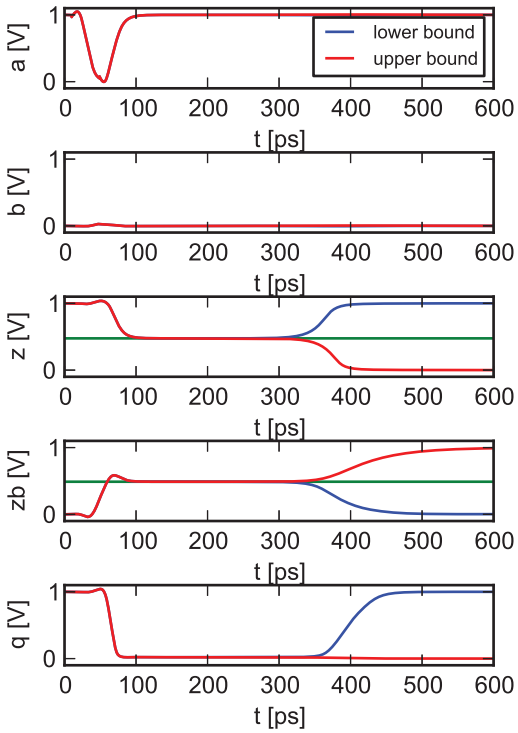


(d) Signal trace (case O4)

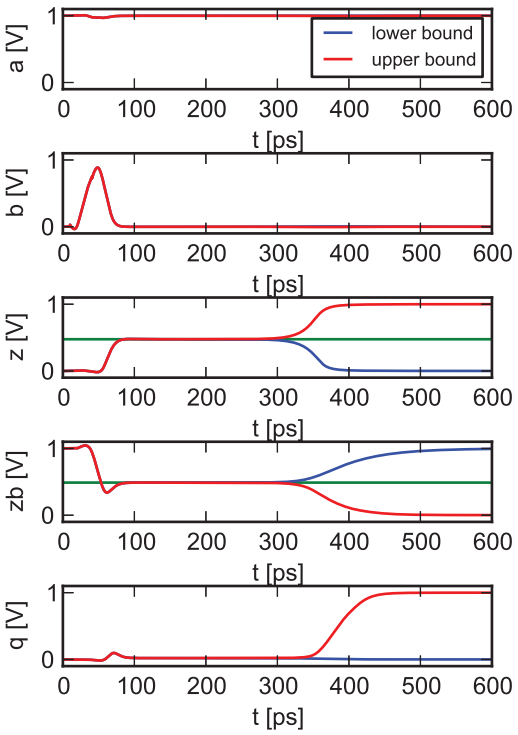
Figure A.7: Results of the van-Berkel implementation with a low threshold output inverter (2)



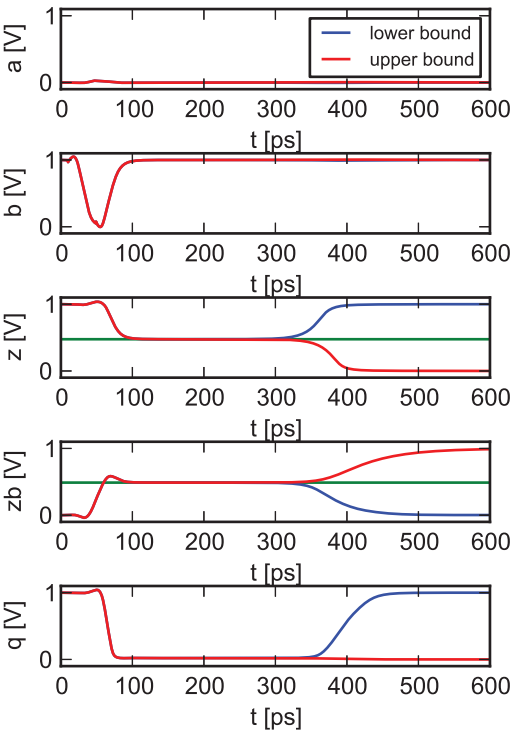
(a) Signal trace (case P1)



(b) Signal trace (case P2)

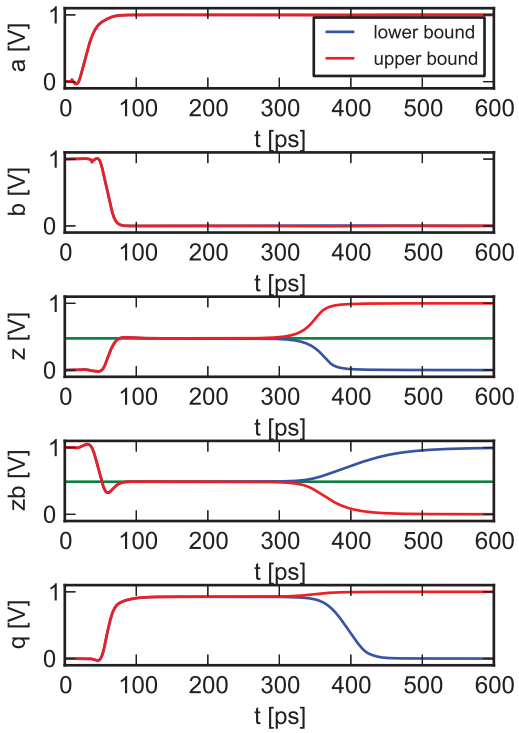


(c) Signal trace (case P3)

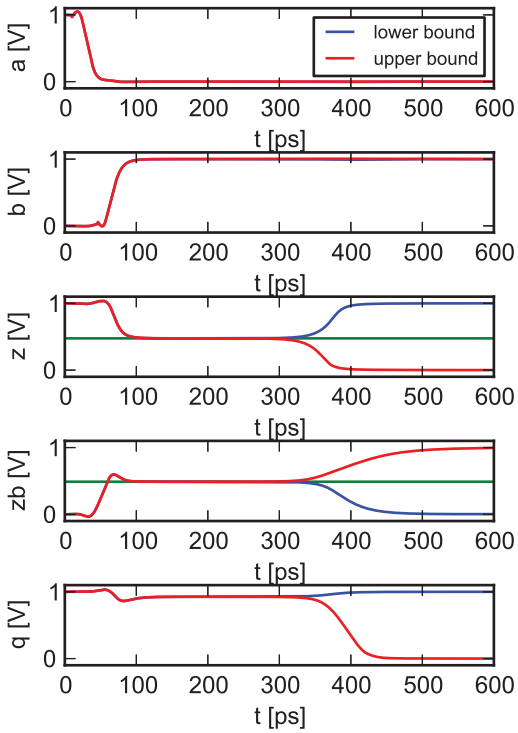


(d) Signal trace (case P4)

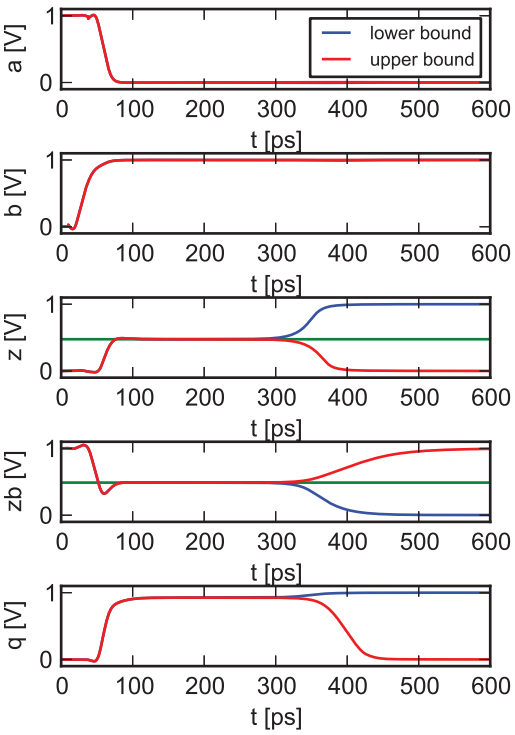
Figure A.8: Results of the van-Berkel implementation with a low threshold output inverter (3)



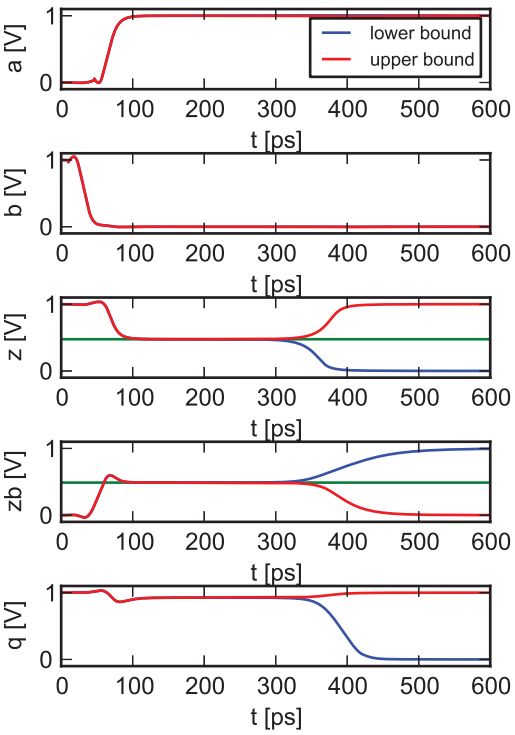
(a) Signal trace (case O1)



(b) Signal trace (case O2)

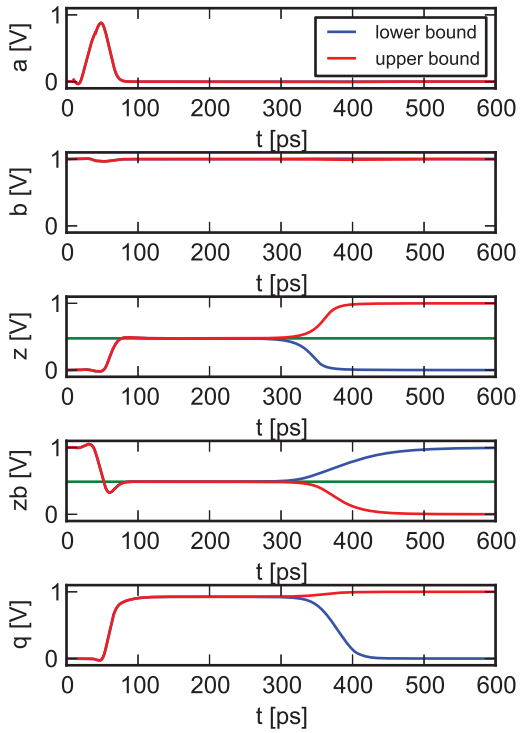


(c) Signal trace (case O3)

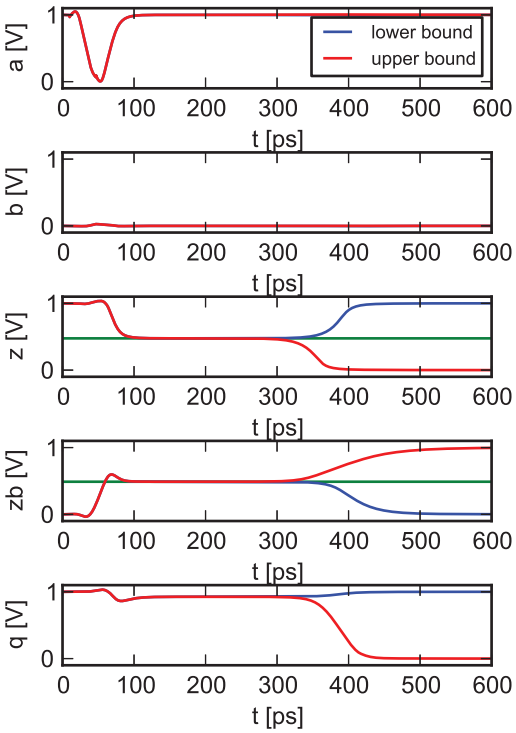


(d) Signal trace (case O4)

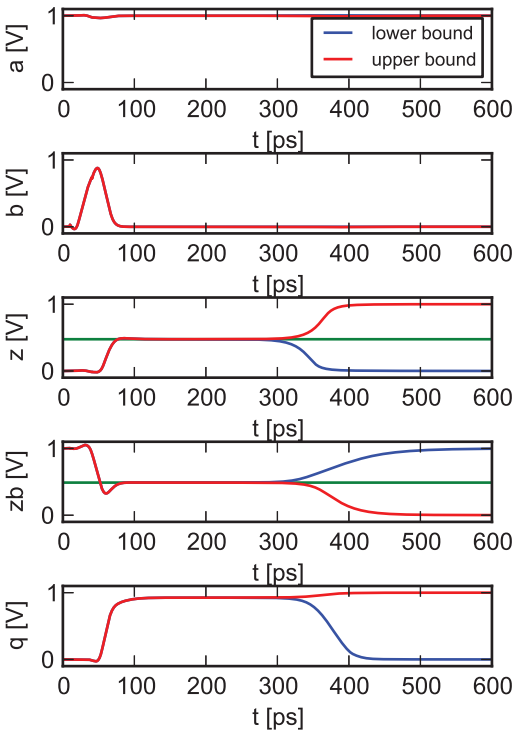
Figure A.9: Results of the van-Berkel implementation with a high threshold output inverter (2)



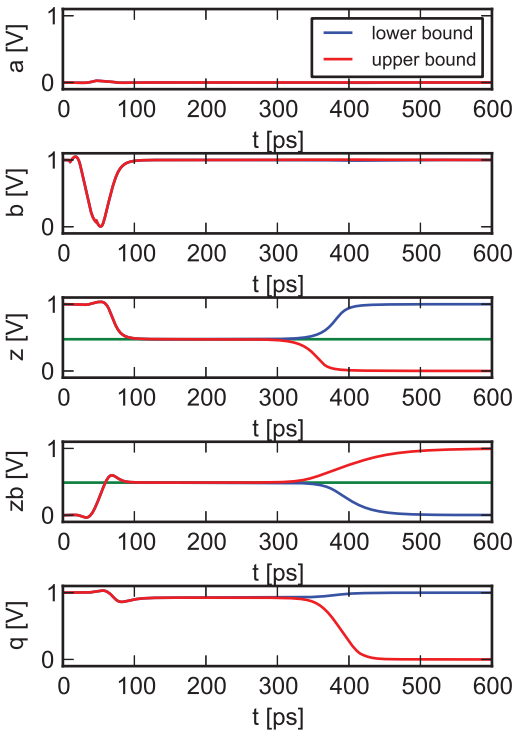
(a) Signal trace (case P1)



(b) Signal trace (case P2)

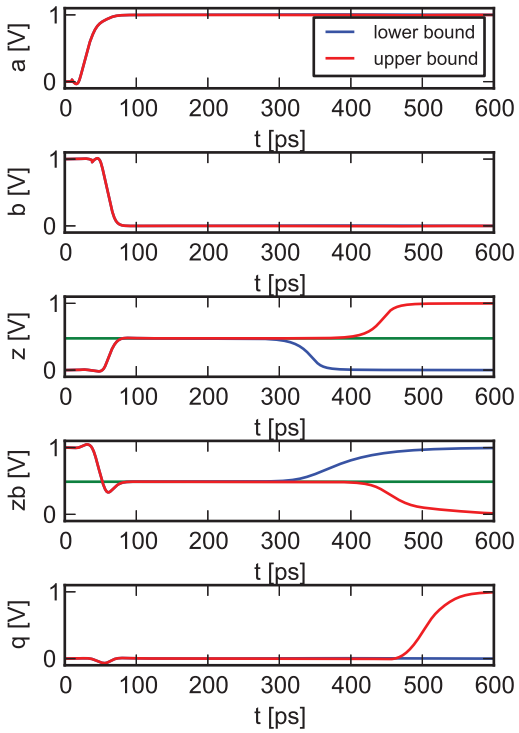


(c) Signal trace (case P3)

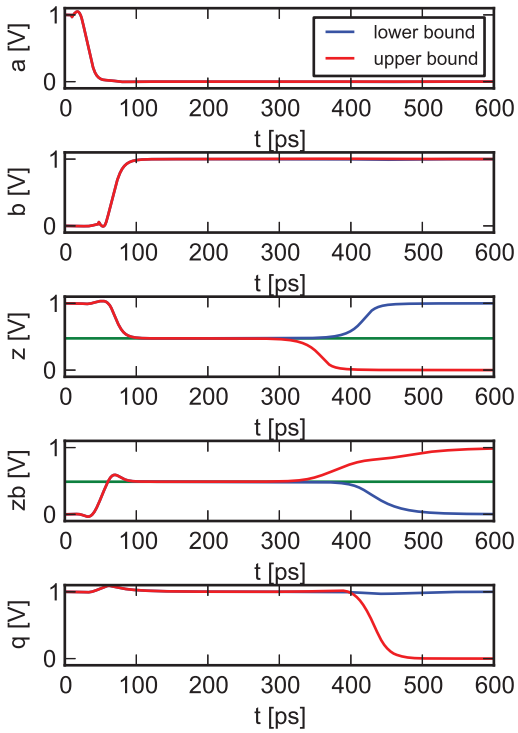


(d) Signal trace (case P4)

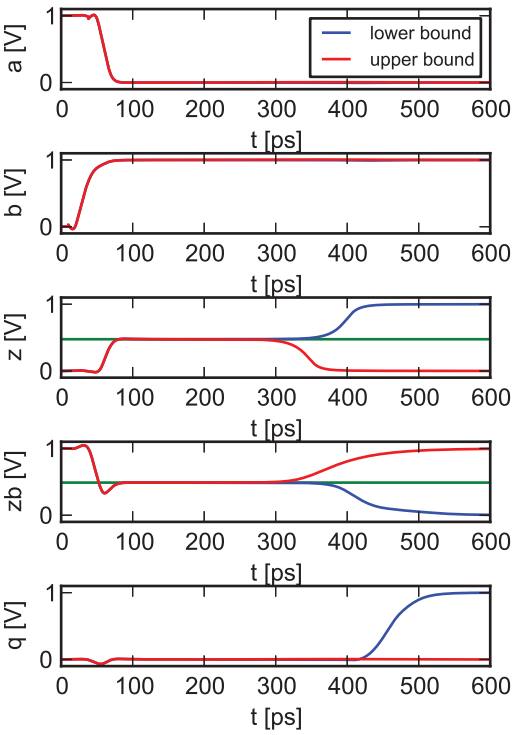
Figure A.10: Results of the van-Berkel implementation with a high threshold output inverter (3)



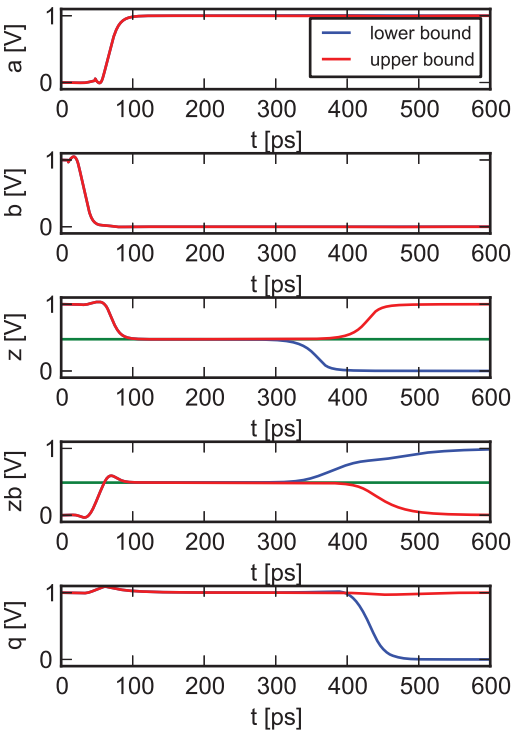
(a) Signal trace (case O1)



(b) Signal trace (case O2)

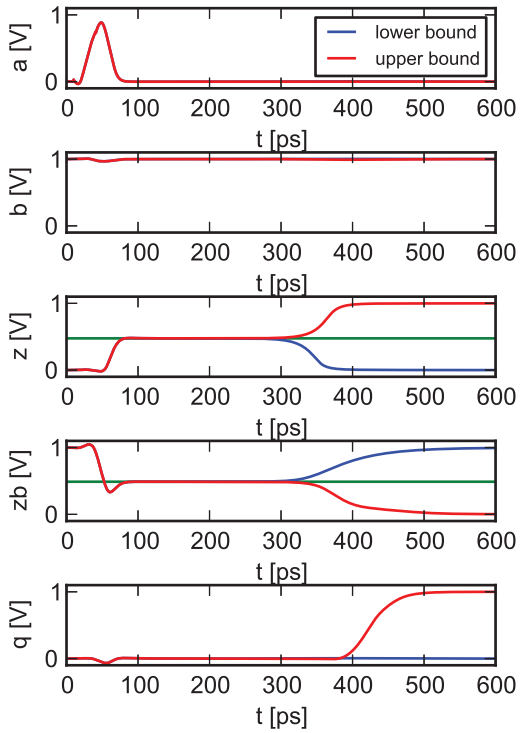


(c) Signal trace (case O3)

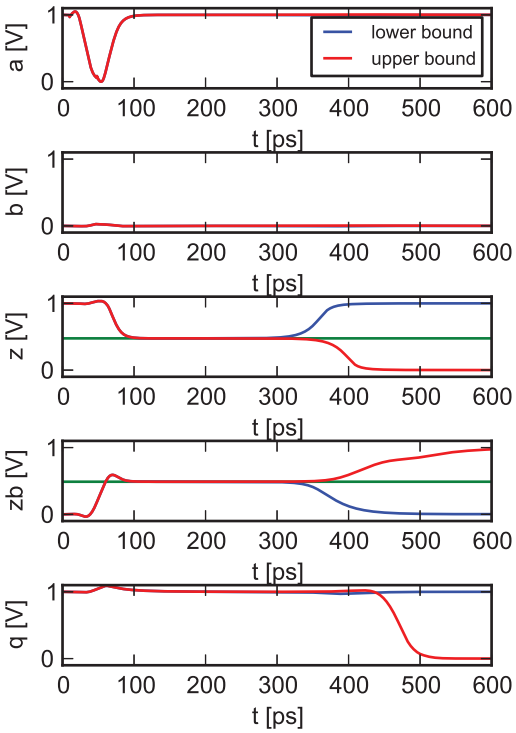


(d) Signal trace (case O4)

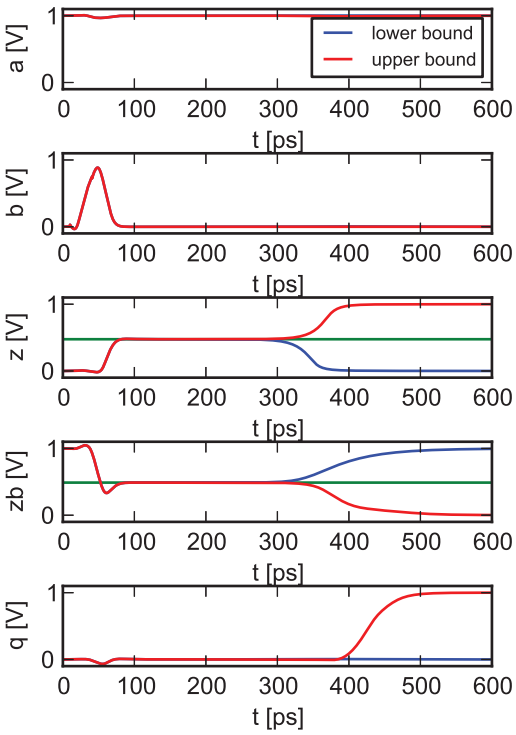
Figure A.11: Results of the van-Berkel implementation with a Schmitt-trigger output (2)



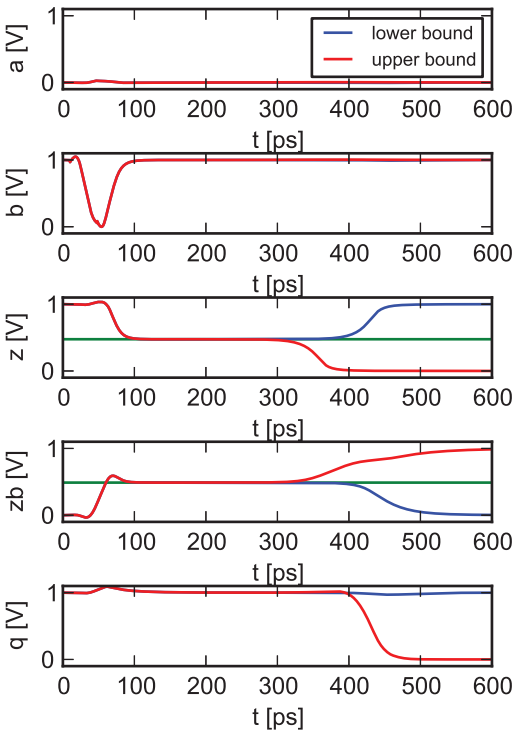
(a) Signal trace (case P1)



(b) Signal trace (case P2)



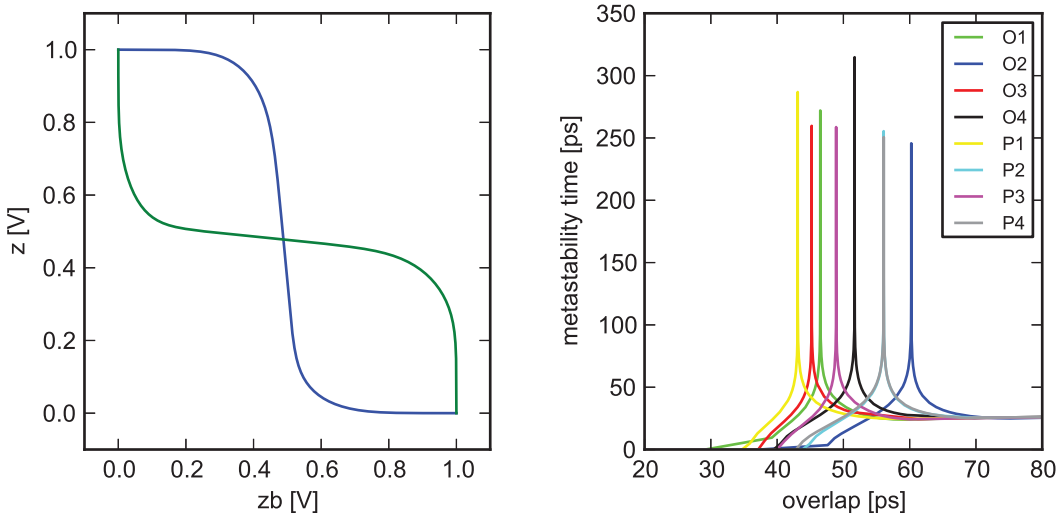
(c) Signal trace (case P3)



(d) Signal trace (case P4)

Figure A.12: Results of the van-Berkel implementation with a Schmitt-trigger output (3)

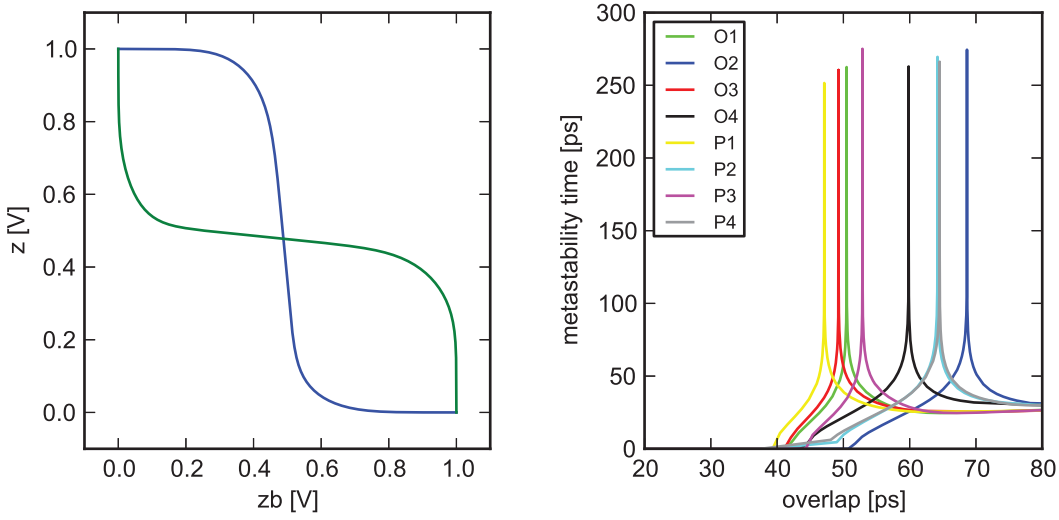
A.2 Conventional Muller C-Element



(a) DC characteristic

(b) Response time vs. input overlap

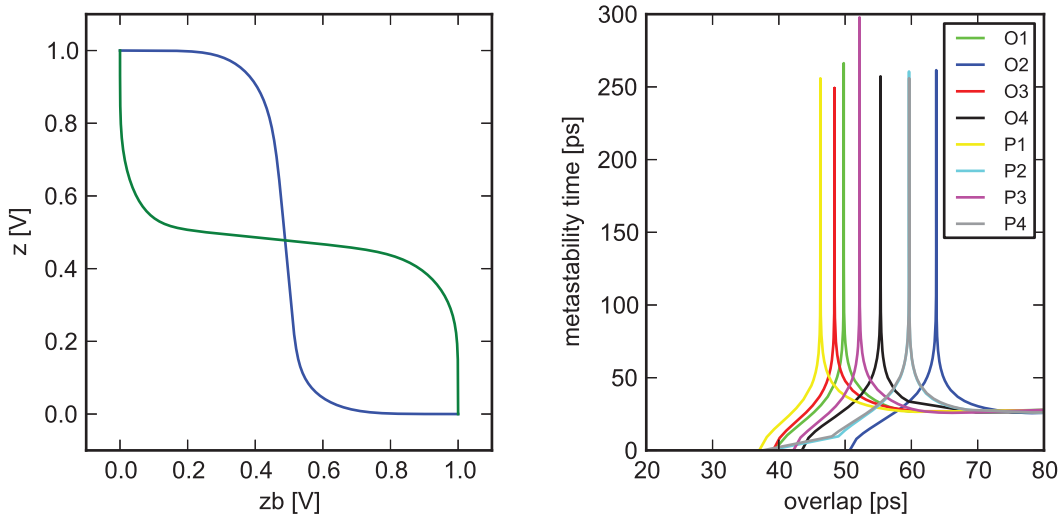
Figure A.13: Results of the conventional implementation with a matched output inverter (1)



(a) DC characteristic

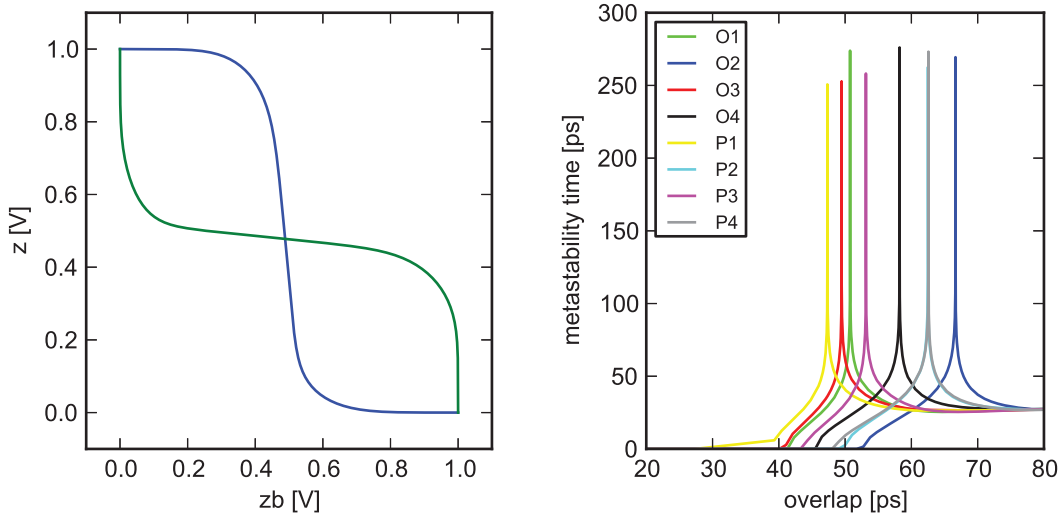
(b) Response time vs. input overlap

Figure A.14: Results of the conventional implementation with a low threshold output inverter (1)



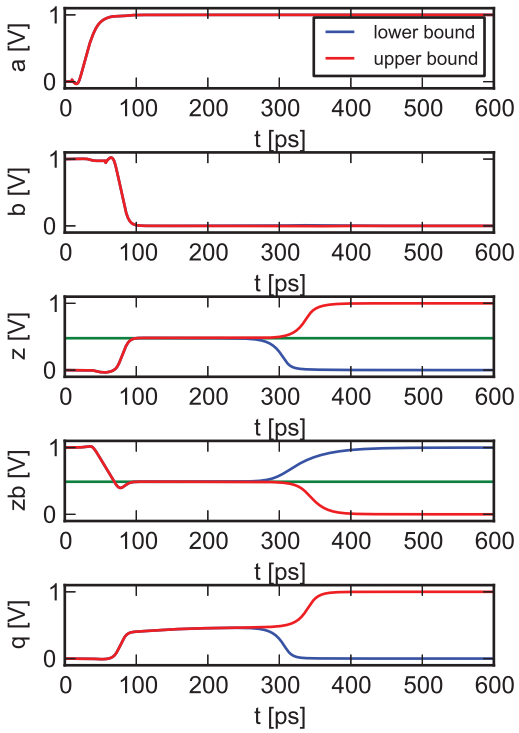
(a) DC characteristic (b) Response time vs. input overlap

Figure A.15: Results of the conventional implementation with a high threshold output inverter (1)

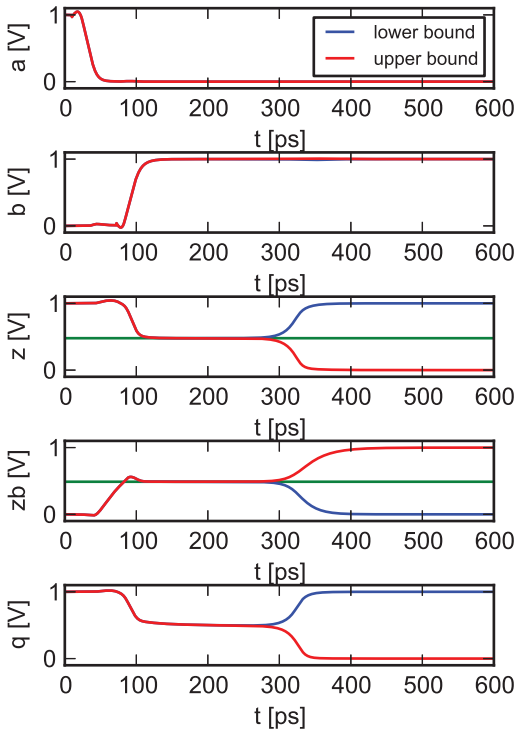


(a) DC characteristic (b) Response time vs. input overlap

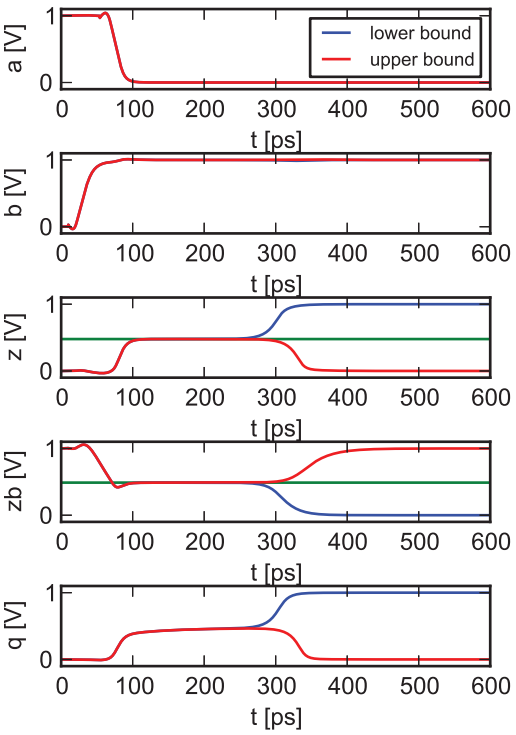
Figure A.16: Results of the conventional implementation with a Schmitt-trigger output (1)



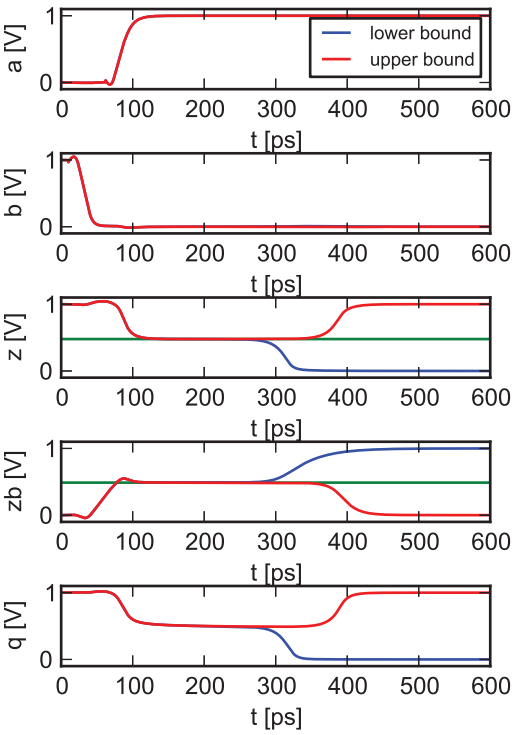
(a) Signal trace (case O1)



(b) Signal trace (case O2)

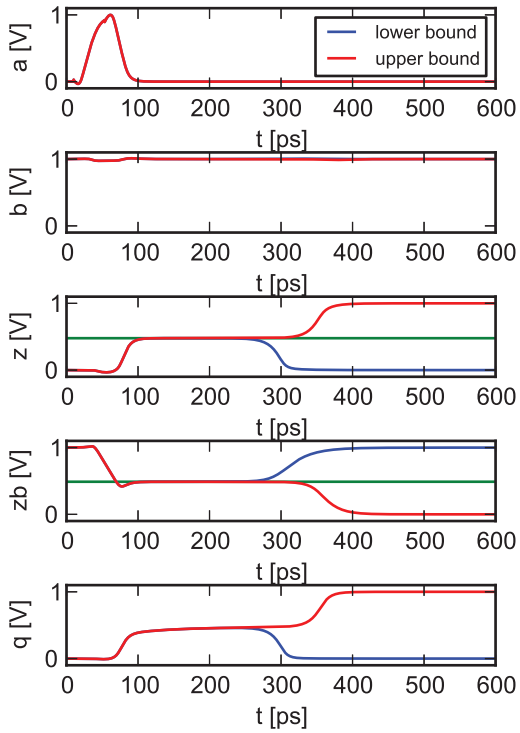


(c) Signal trace (case O3)

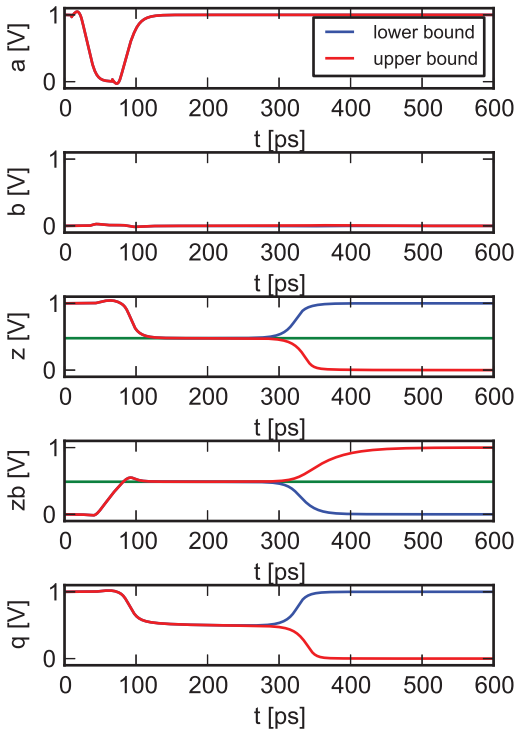


(d) Signal trace (case O4)

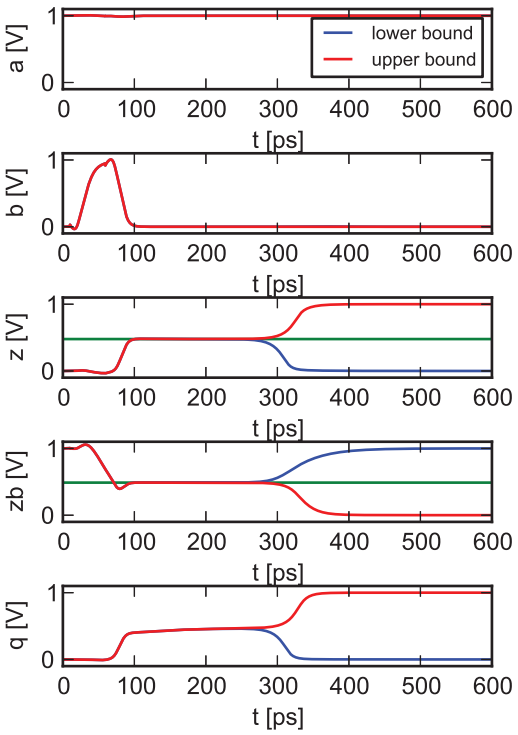
Figure A.17: Results of the conventional implementation with a matched output inverter (2)



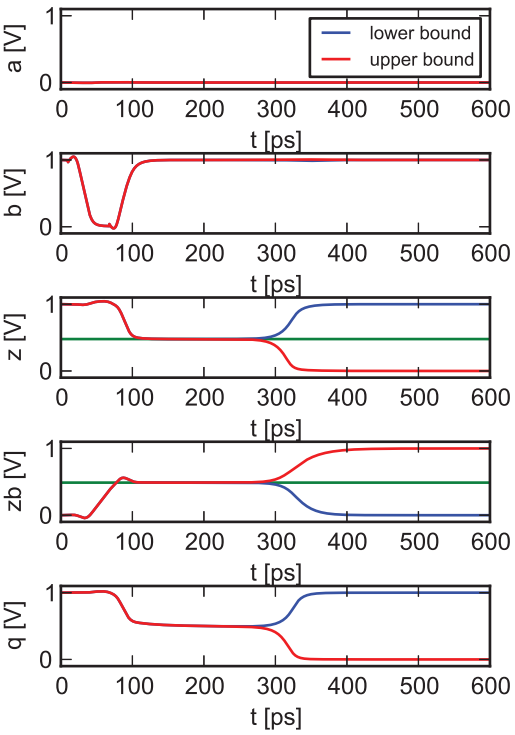
(a) Signal trace (case P1)



(b) Signal trace (case P2)

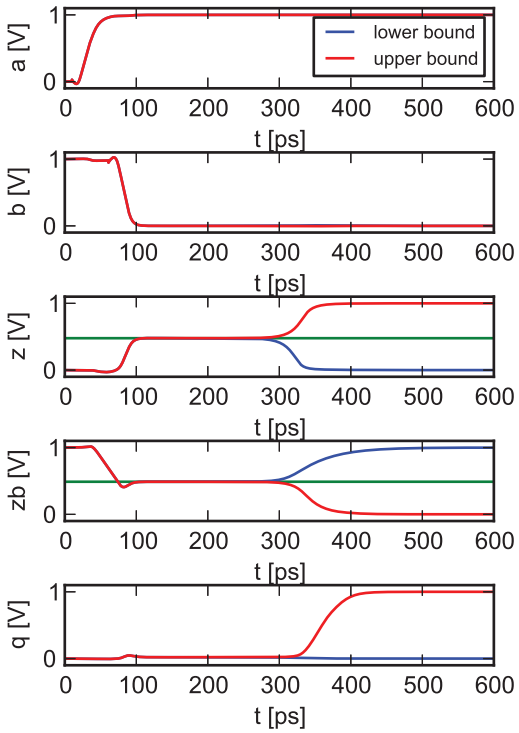


(c) Signal trace (case P3)

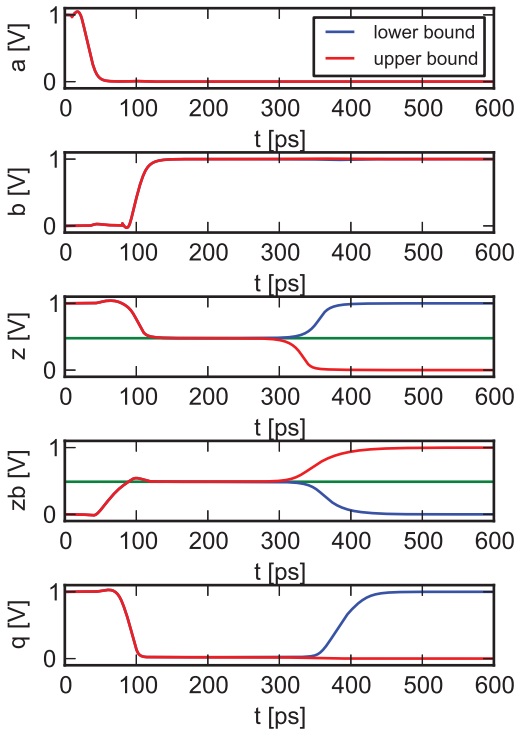


(d) Signal trace (case P4)

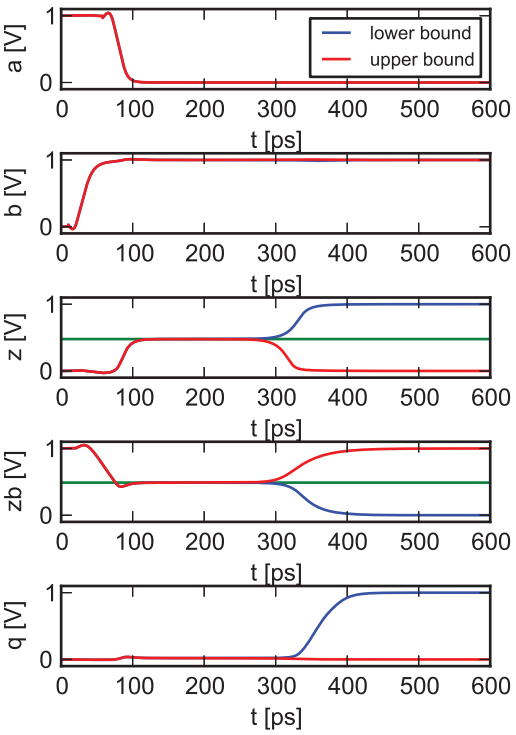
Figure A.18: Results of the conventional implementation with a matched output inverter (3)



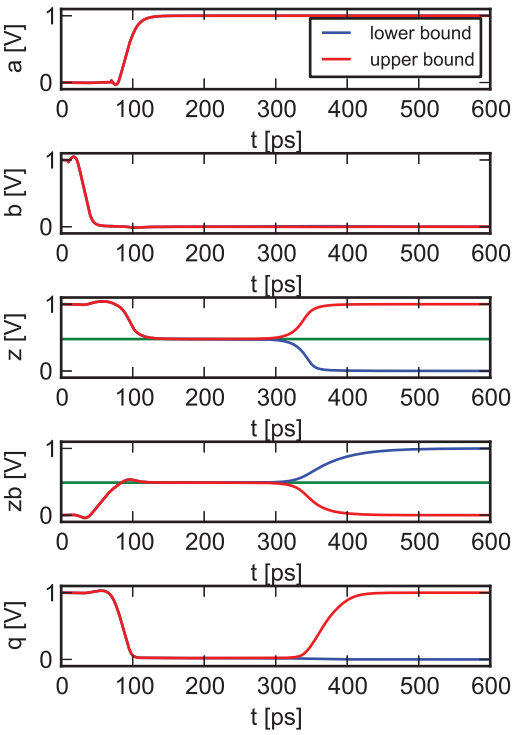
(a) Signal trace (case O1)



(b) Signal trace (case O2)

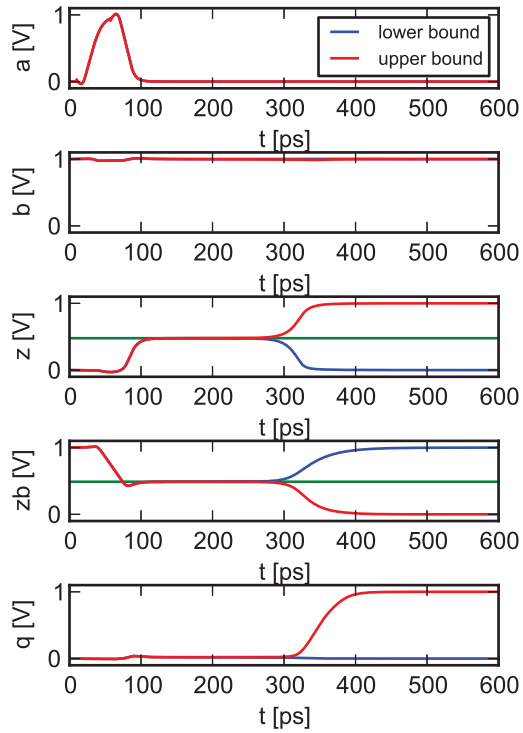


(c) Signal trace (case O3)

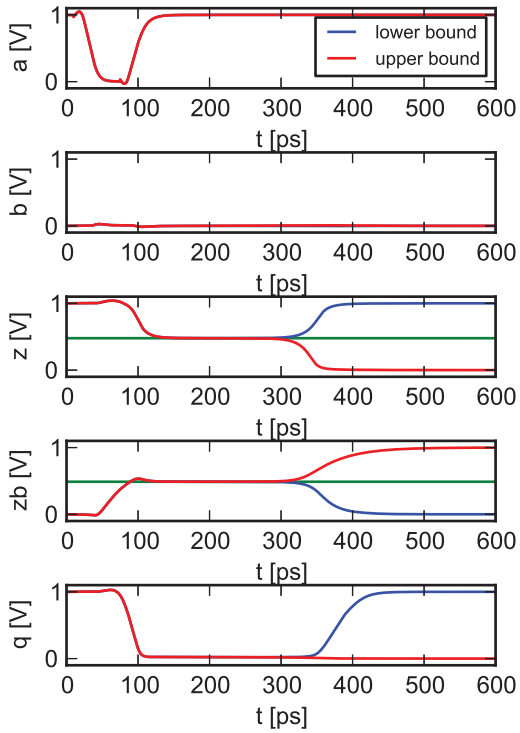


(d) Signal trace (case O4)

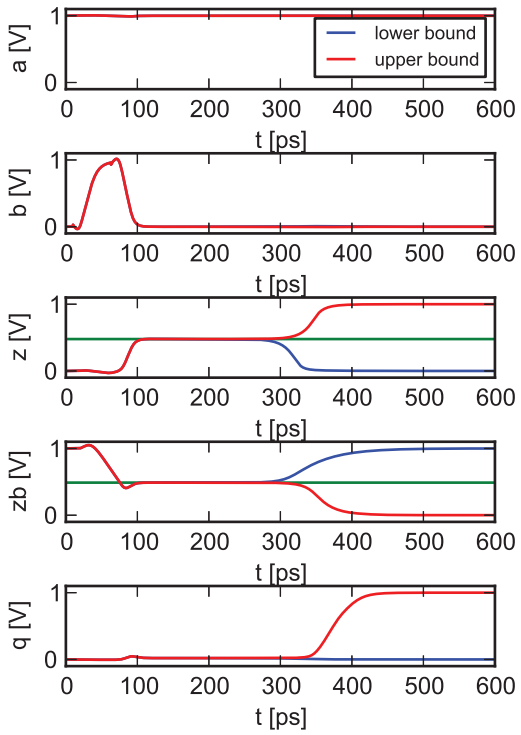
Figure A.19: Results of the conventional implementation with a low threshold output inverter (2)



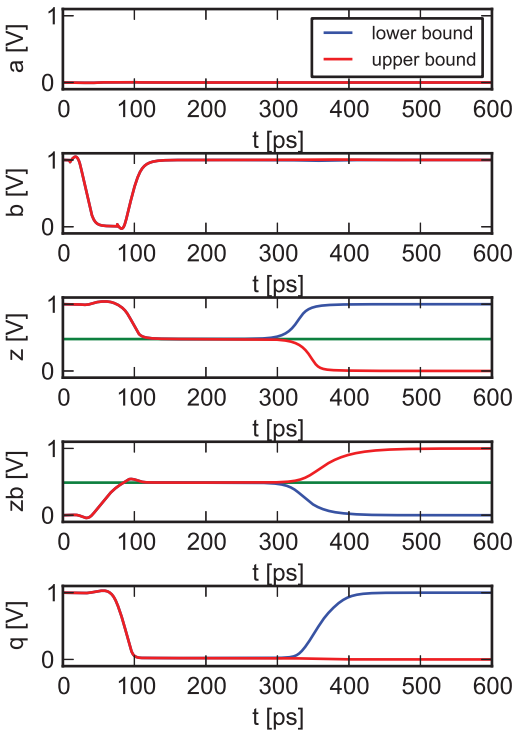
(a) Signal trace (case P1)



(b) Signal trace (case P2)

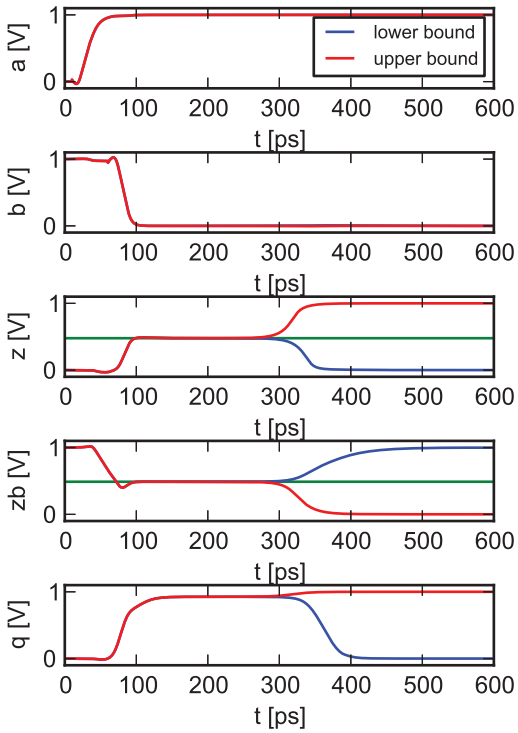


(c) Signal trace (case P3)

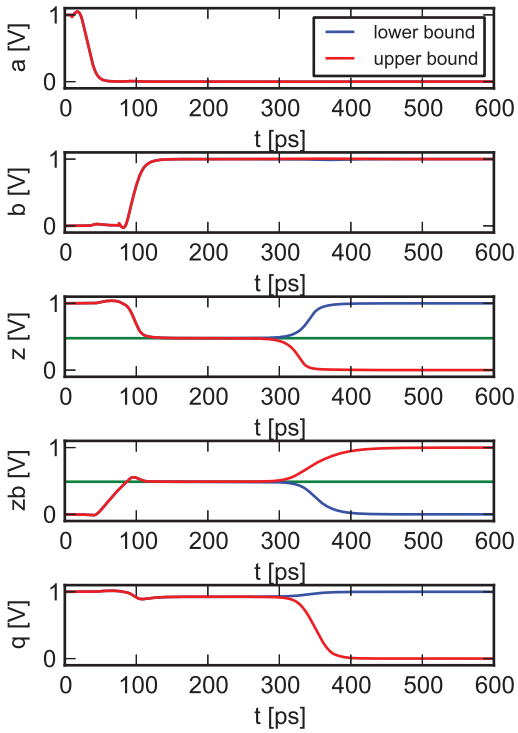


(d) Signal trace (case P4)

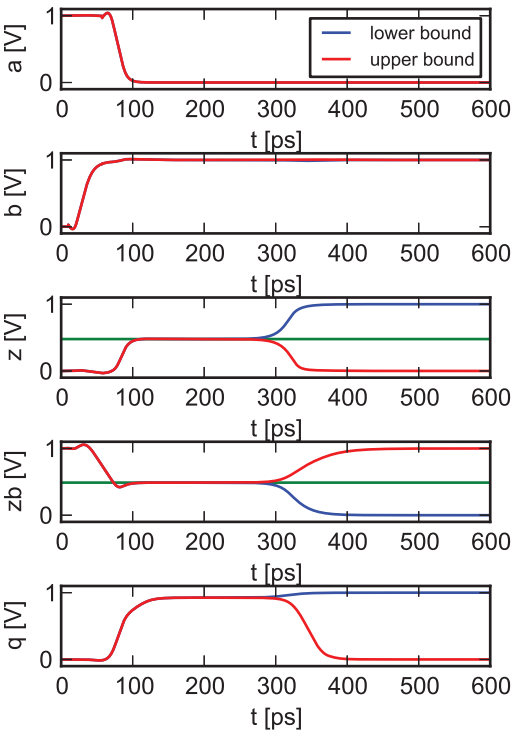
Figure A.20: Results of the conventional implementation with a low threshold output inverter (3)



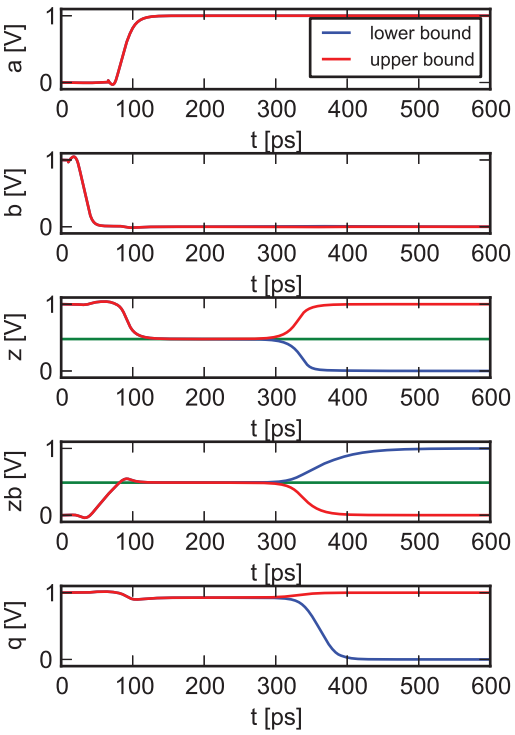
(a) Signal trace (case O1)



(b) Signal trace (case O2)

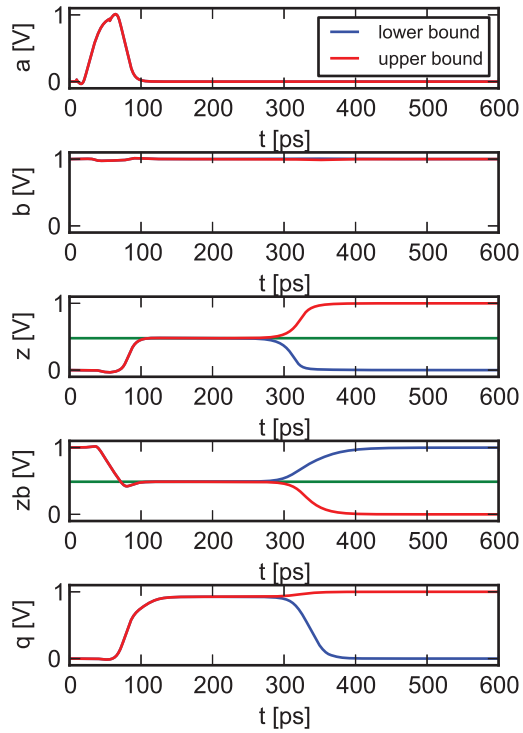


(c) Signal trace (case O3)

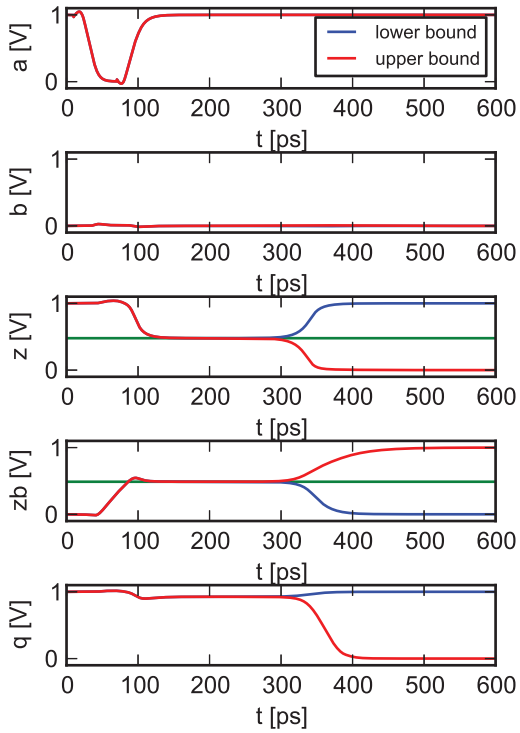


(d) Signal trace (case O4)

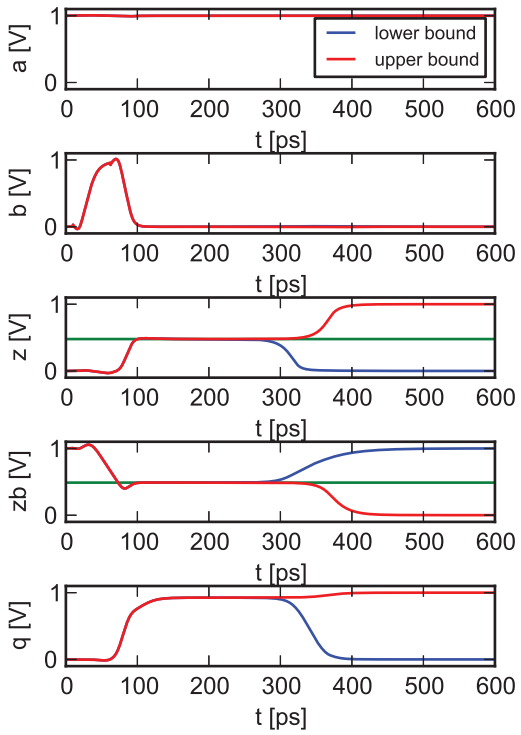
Figure A.21: Results of the conventional implementation with a high threshold output inverter (2)



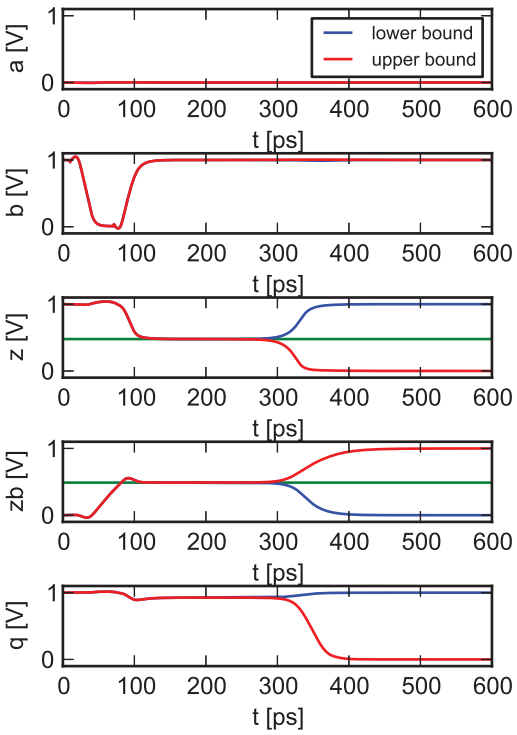
(a) Signal trace (case P1)



(b) Signal trace (case P2)

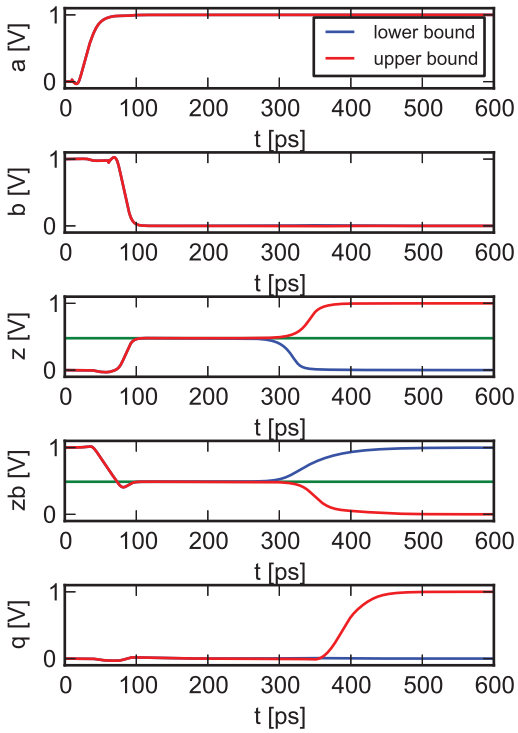


(c) Signal trace (case P3)

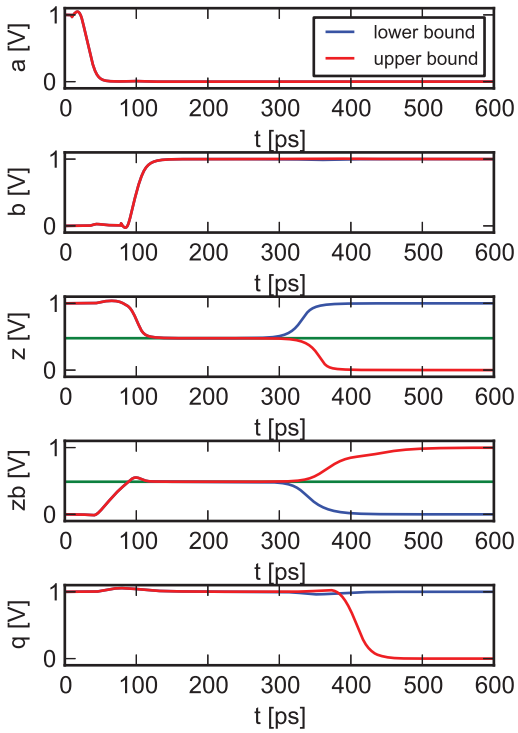


(d) Signal trace (case P4)

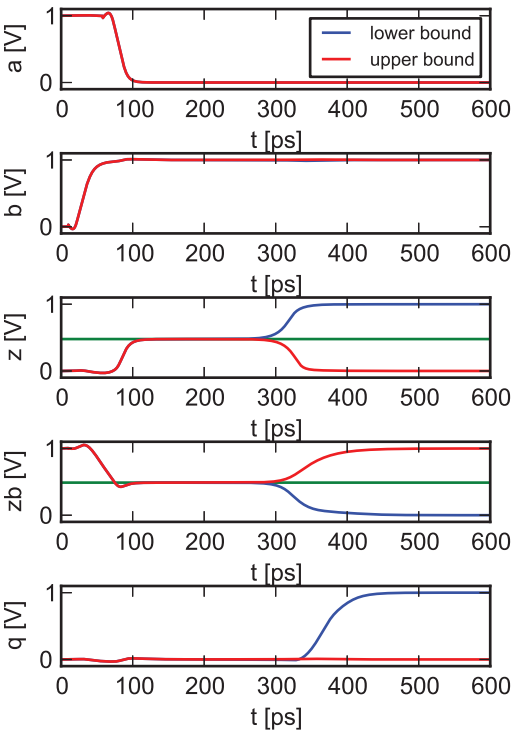
Figure A.22: Results of the conventional implementation with a high threshold output inverter (3)



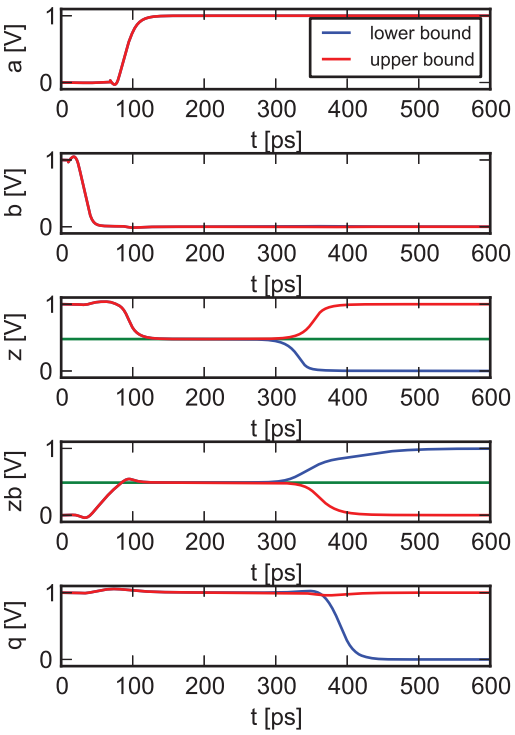
(a) Signal trace (case O1)



(b) Signal trace (case O2)

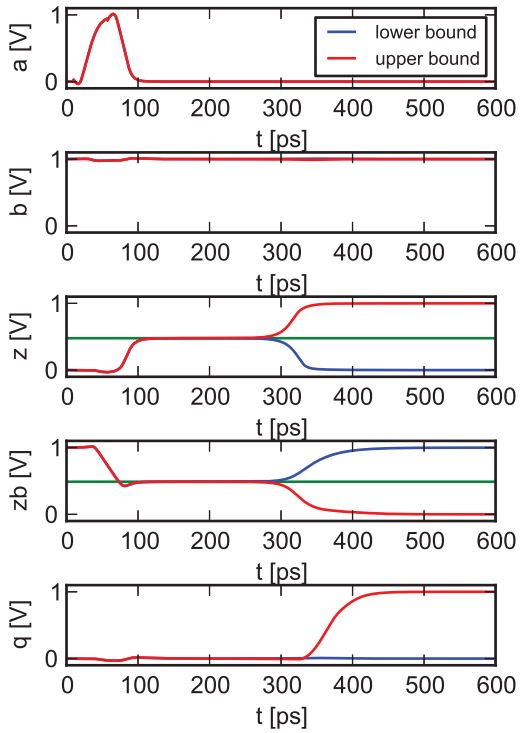


(c) Signal trace (case O3)

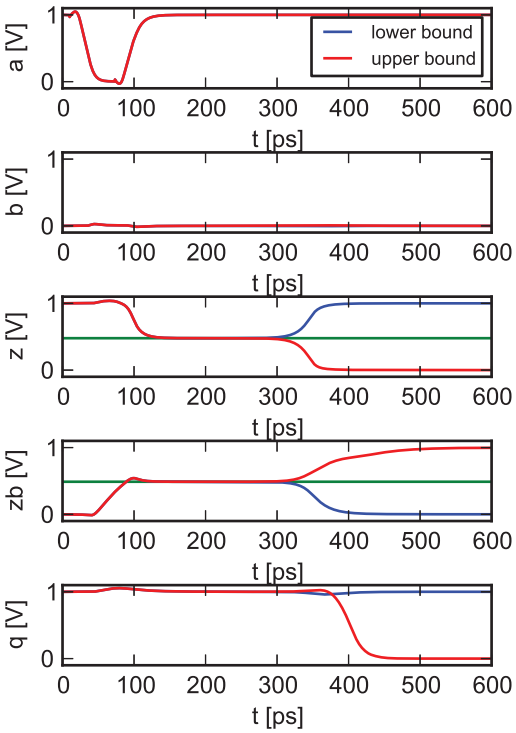


(d) Signal trace (case O4)

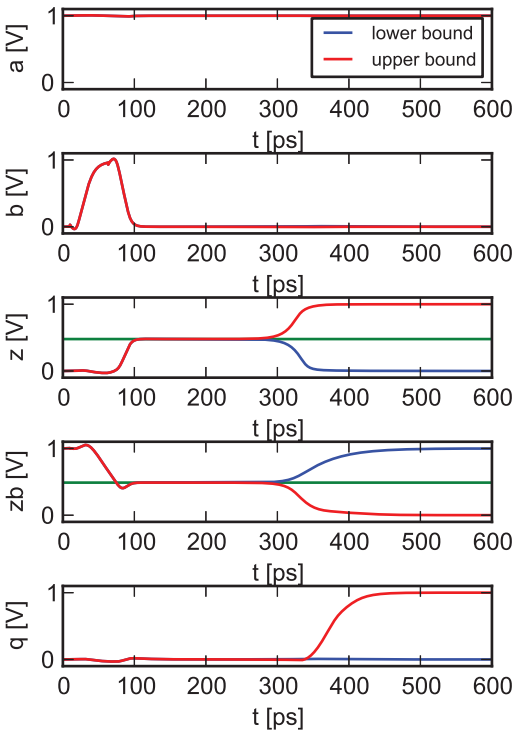
Figure A.23: Results of the conventional implementation with a Schmitt-trigger output (2)



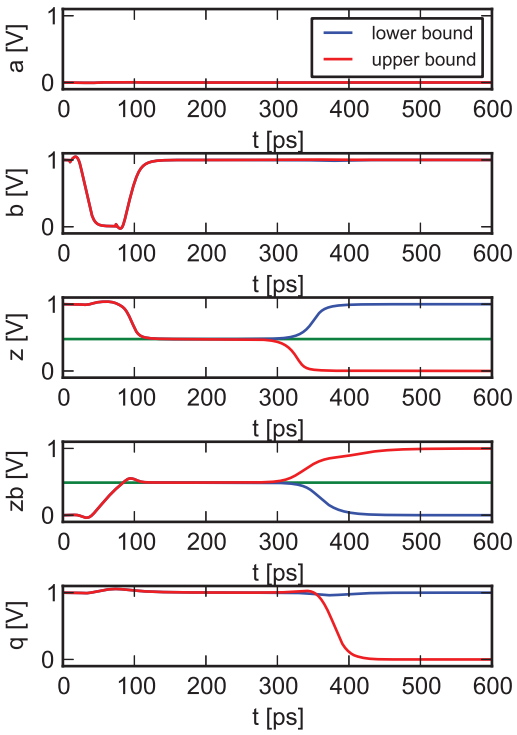
(a) Signal trace (case P1)



(b) Signal trace (case P2)



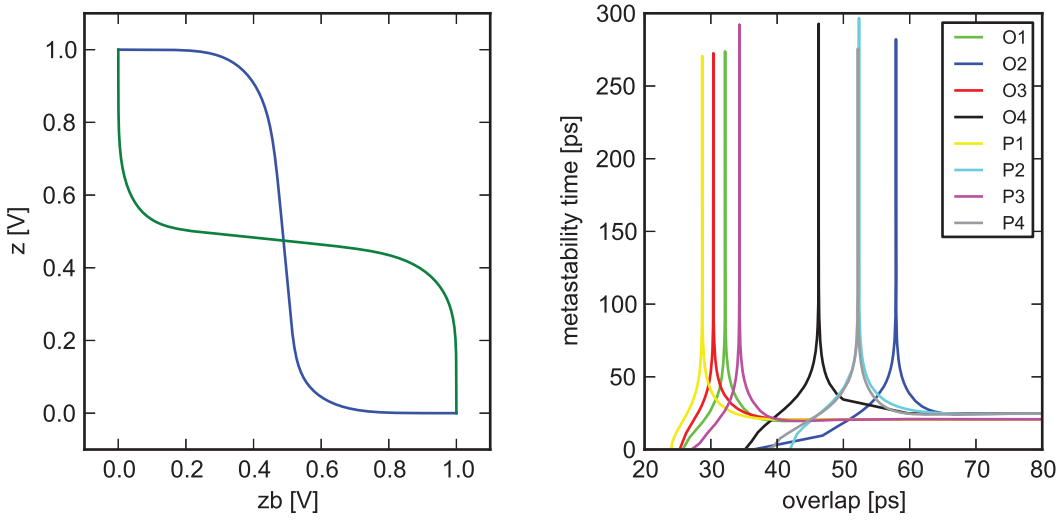
(c) Signal trace (case P3)



(d) Signal trace (case P4)

Figure A.24: Results of the conventional implementation with a Schmitt-trigger output (3)

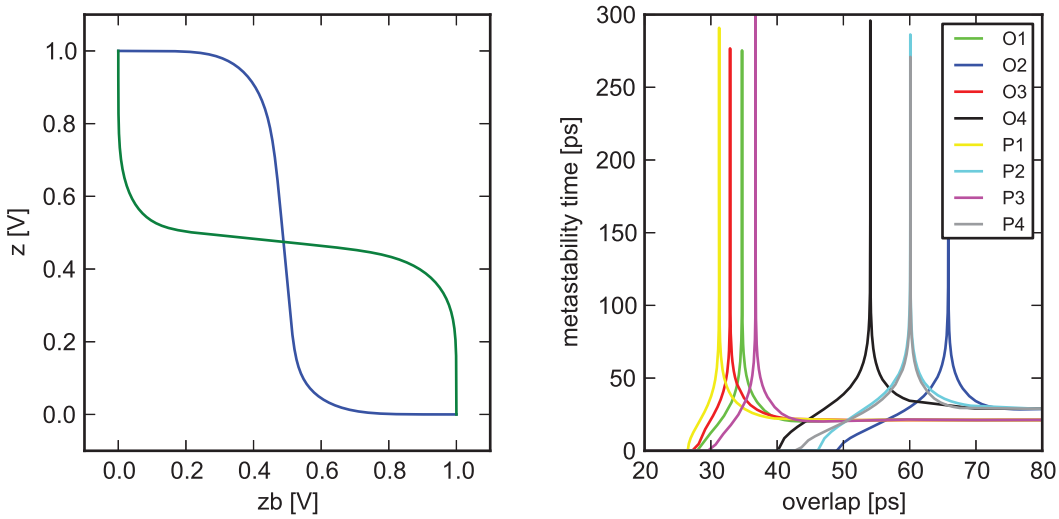
A.3 Weak Feedback Muller C-Element



(a) DC characteristic

(b) Response time vs. input overlap

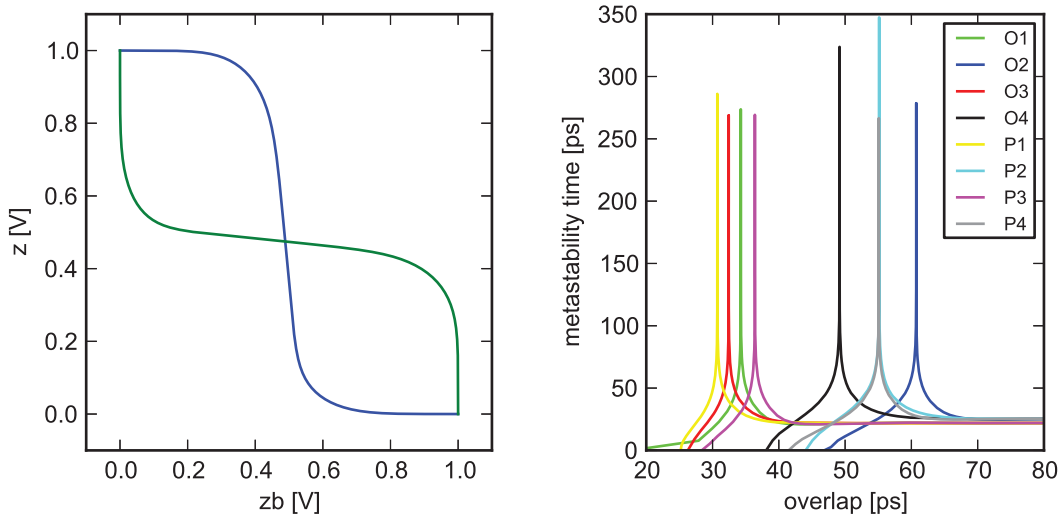
Figure A.25: Results of the weak feedback implementation with a matched output inverter (1)



(a) DC characteristic

(b) Response time vs. input overlap

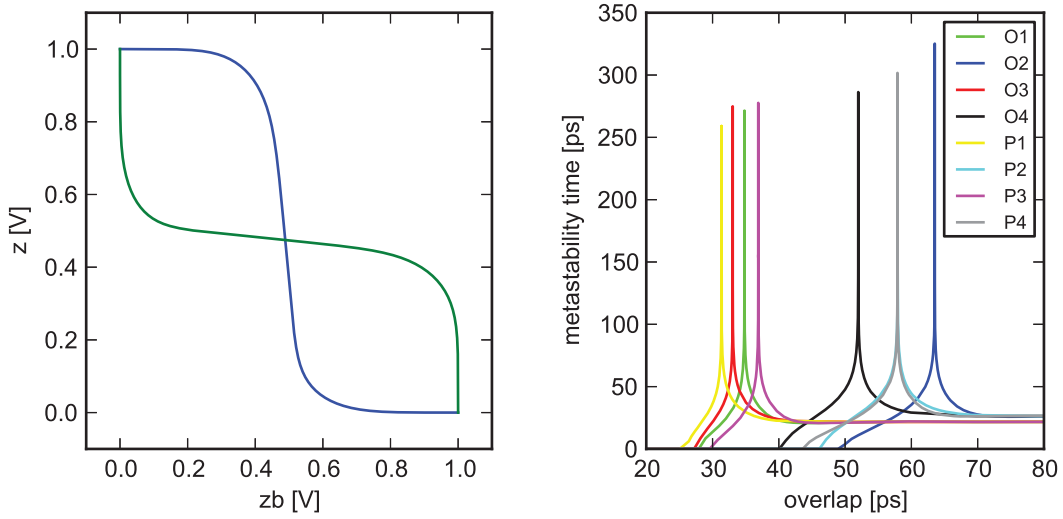
Figure A.26: Results of the weak feedback implementation with a low threshold output inverter (1)



(a) DC characteristic

(b) Response time vs. input overlap

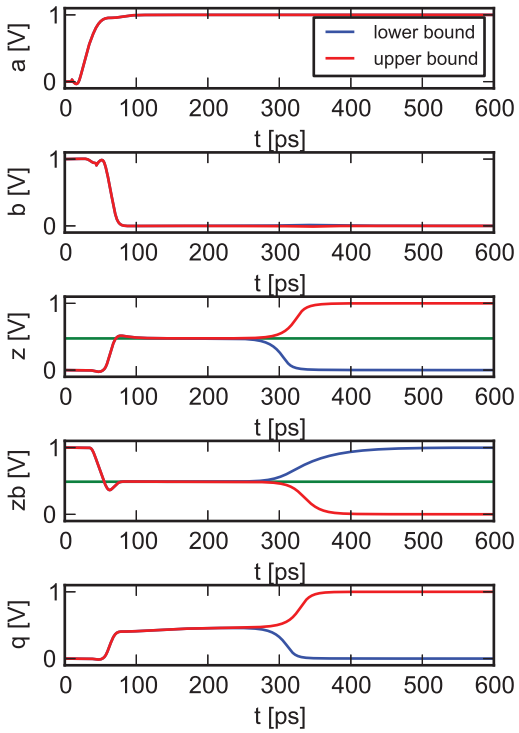
Figure A.27: Results of the weak feedback implementation with a high threshold output inverter (1)



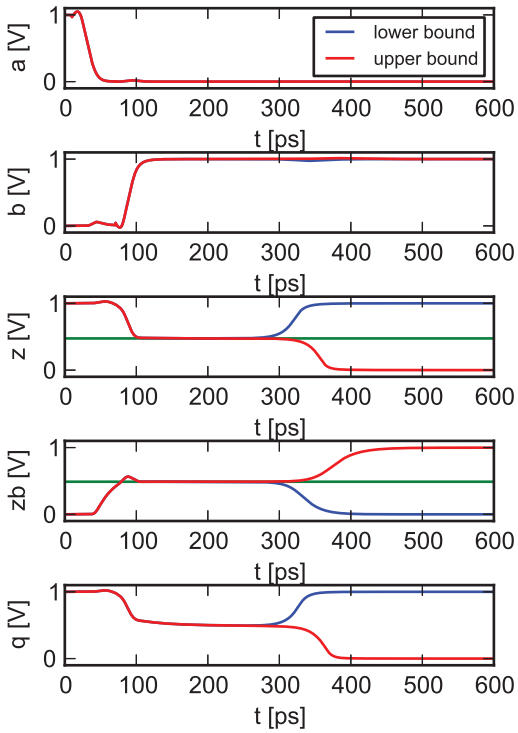
(a) DC characteristic

(b) Response time vs. input overlap

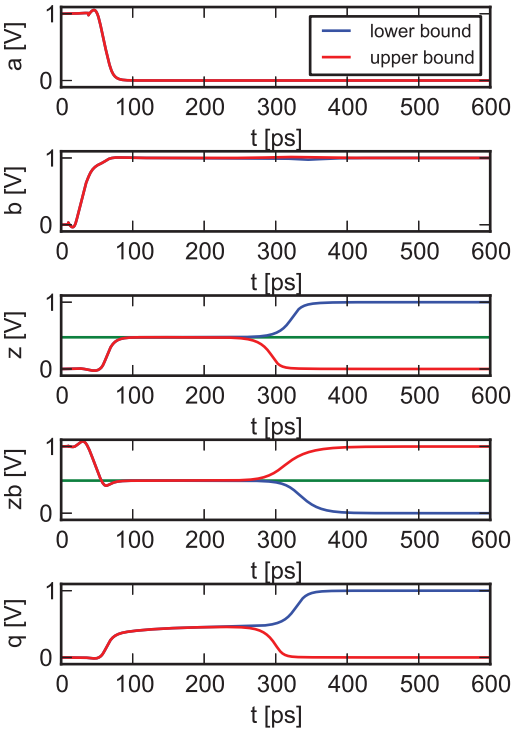
Figure A.28: Results of the weak feedback implementation with a Schmitt-trigger output (1)



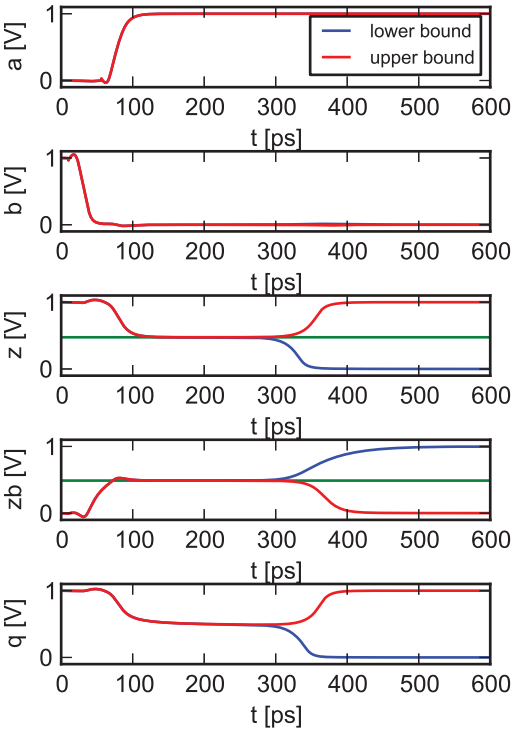
(a) Signal trace (case O1)



(b) Signal trace (case O2)

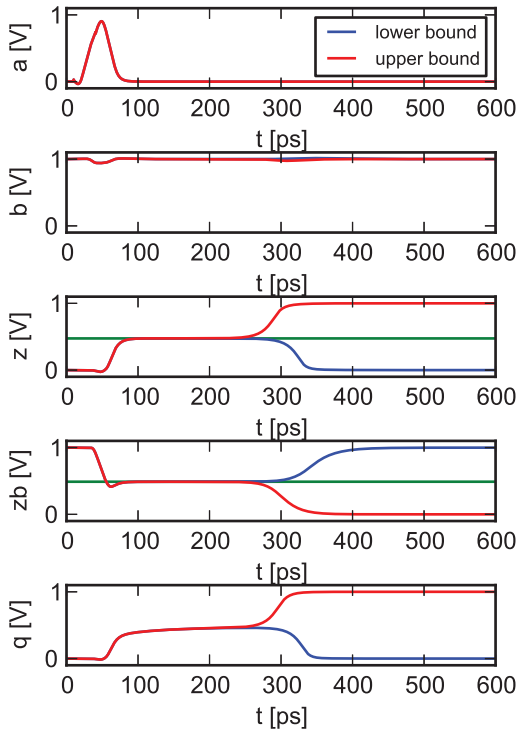


(c) Signal trace (case O3)

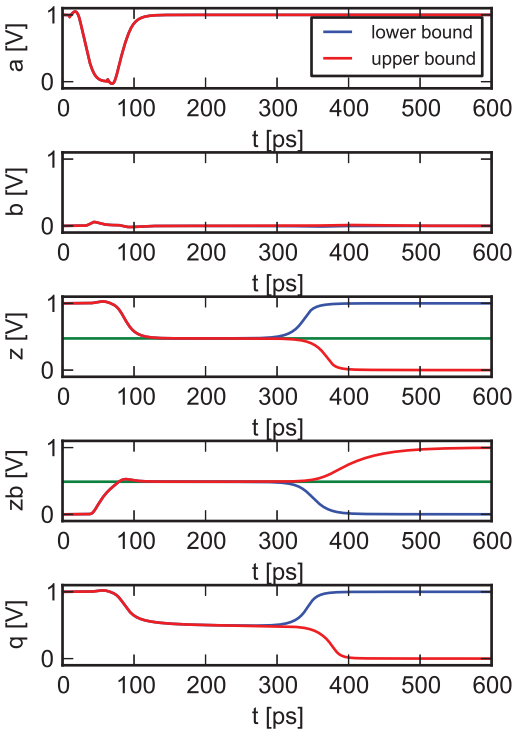


(d) Signal trace (case O4)

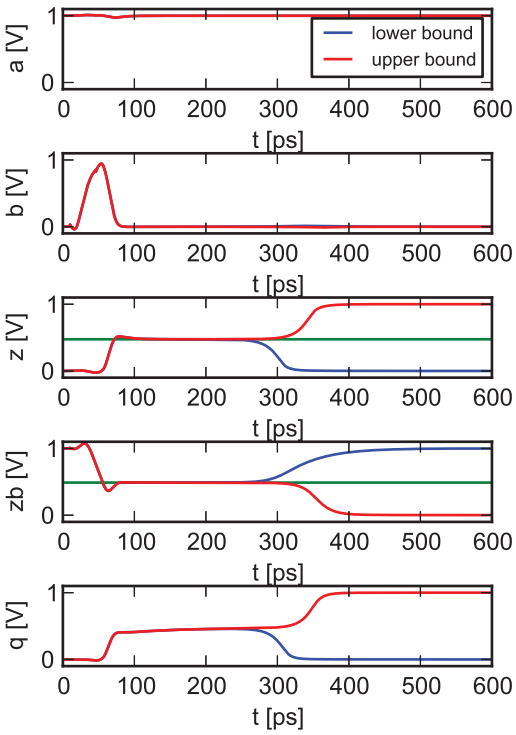
Figure A.29: Results of the weak feedback implementation with a matched output inverter (2)



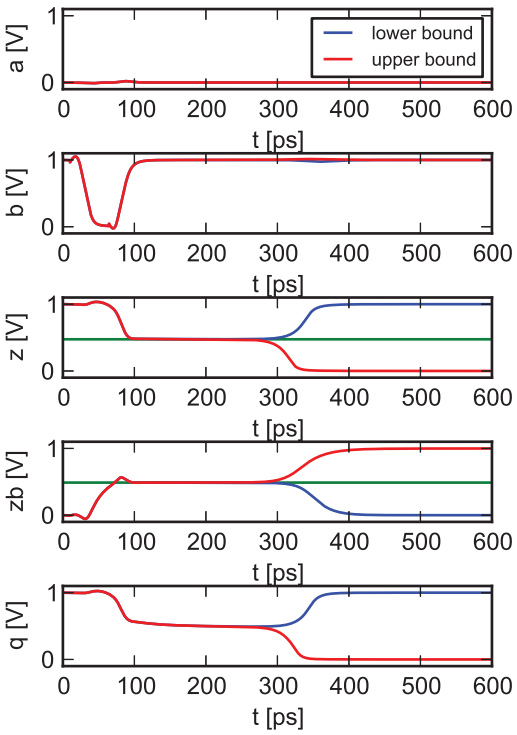
(a) Signal trace (case P1)



(b) Signal trace (case P2)

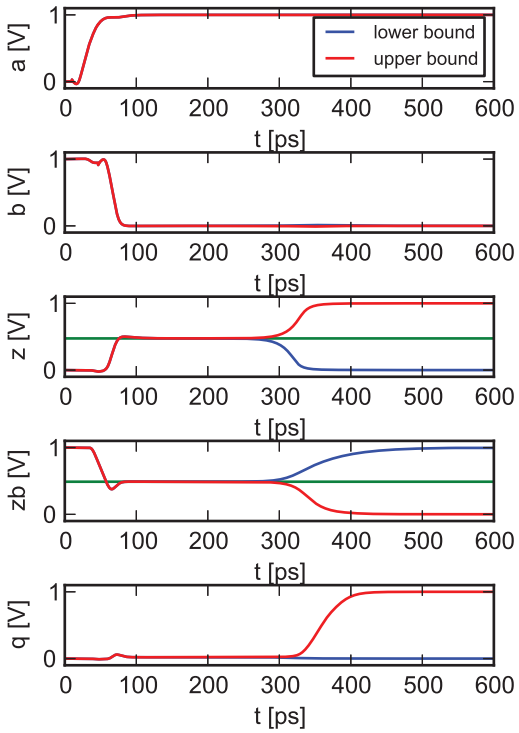


(c) Signal trace (case P3)

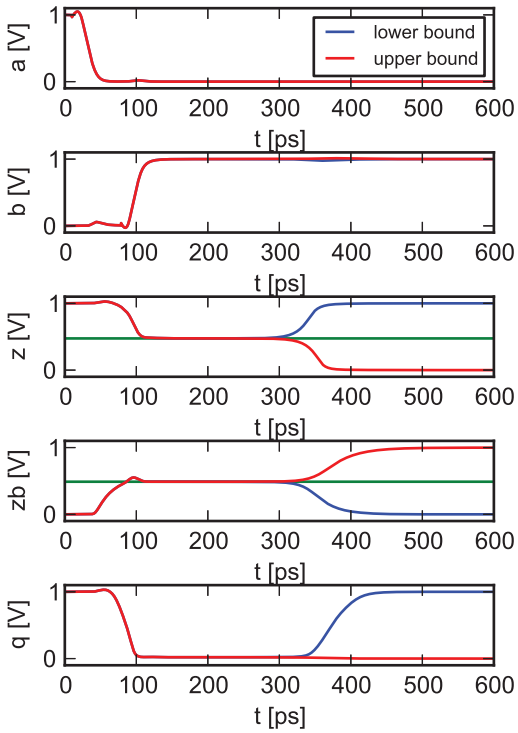


(d) Signal trace (case P4)

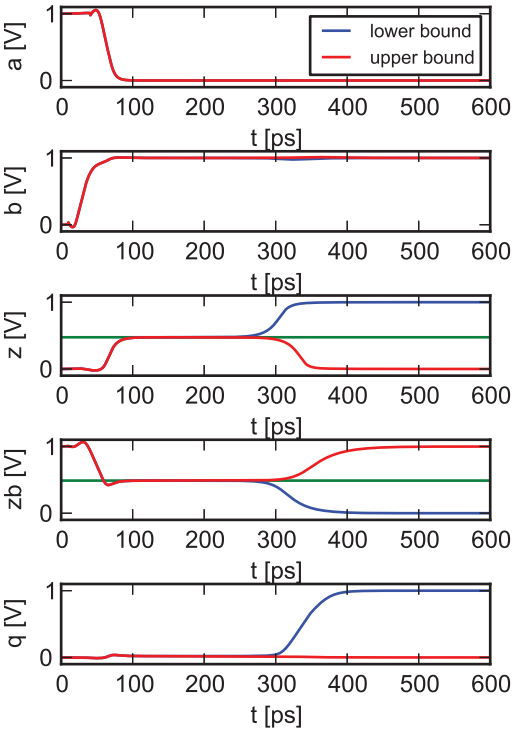
Figure A.30: Results of the weak feedback implementation with a matched output inverter (3)



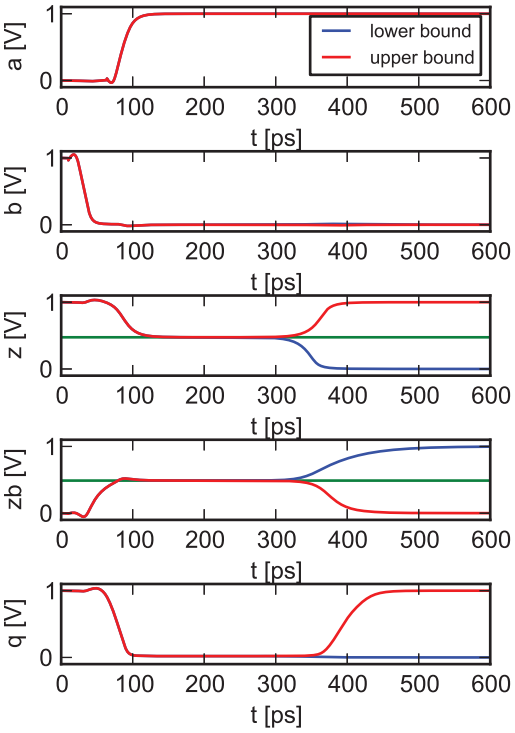
(a) Signal trace (case O1)



(b) Signal trace (case O2)

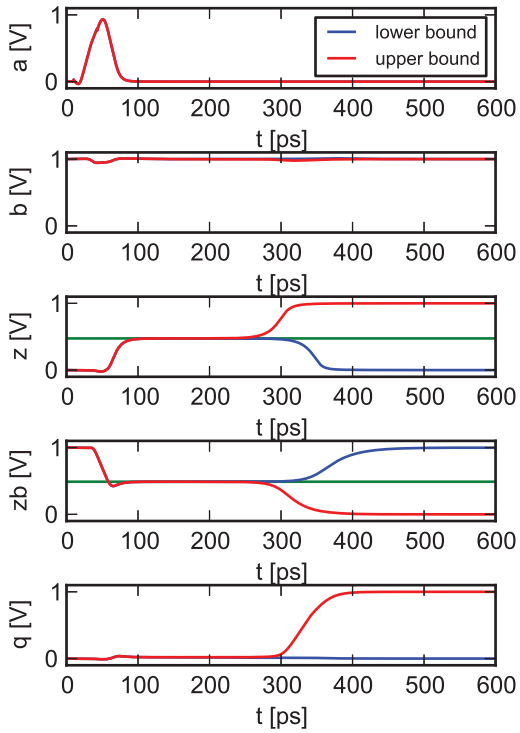


(c) Signal trace (case O3)

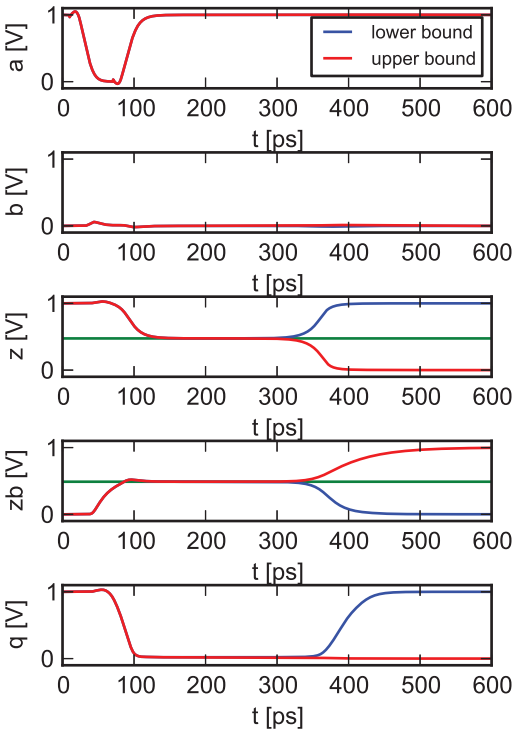


(d) Signal trace (case O4)

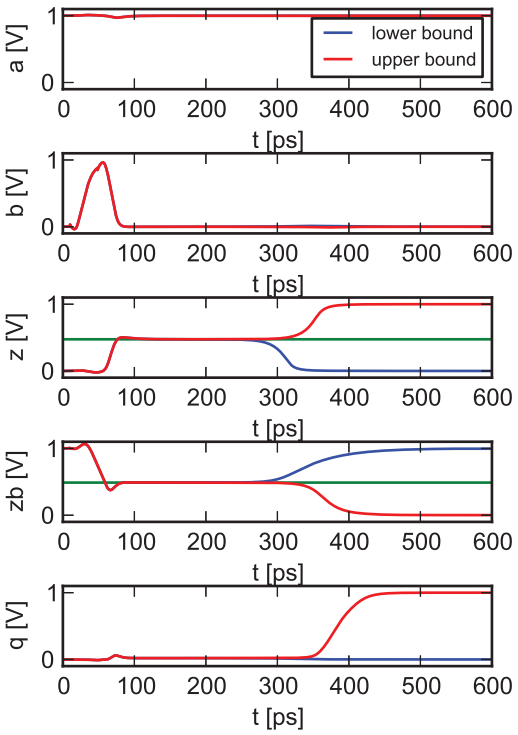
Figure A.31: Results of the weak feedback implementation with a low threshold output inverter (2)



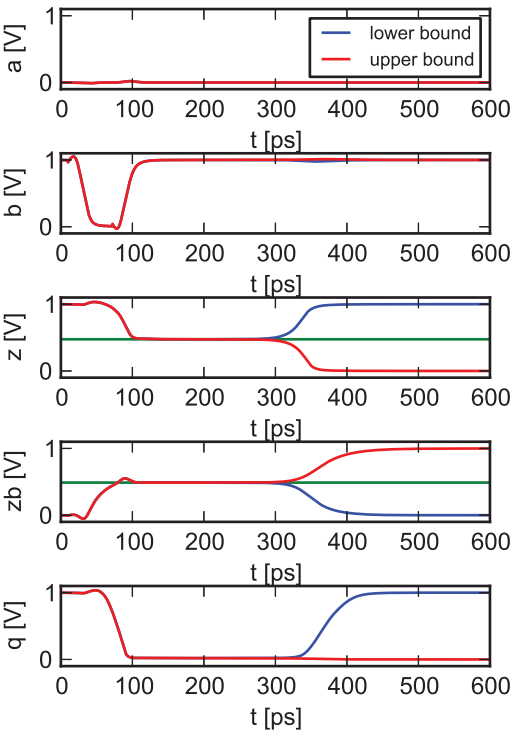
(a) Signal trace (case P1)



(b) Signal trace (case P2)

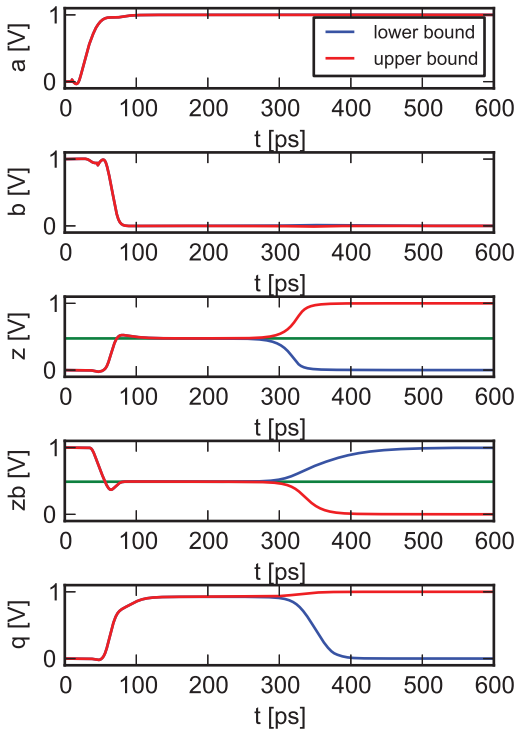


(c) Signal trace (case P3)

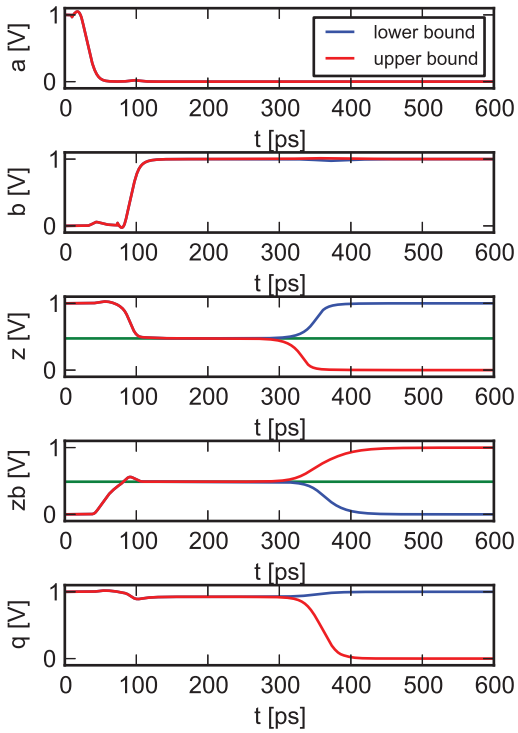


(d) Signal trace (case P4)

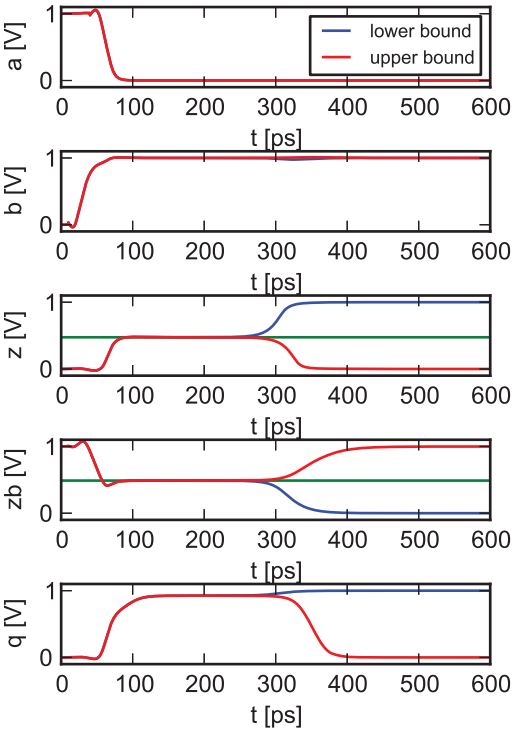
Figure A.32: Results of the weak feedback implementation with a low threshold output inverter (3)



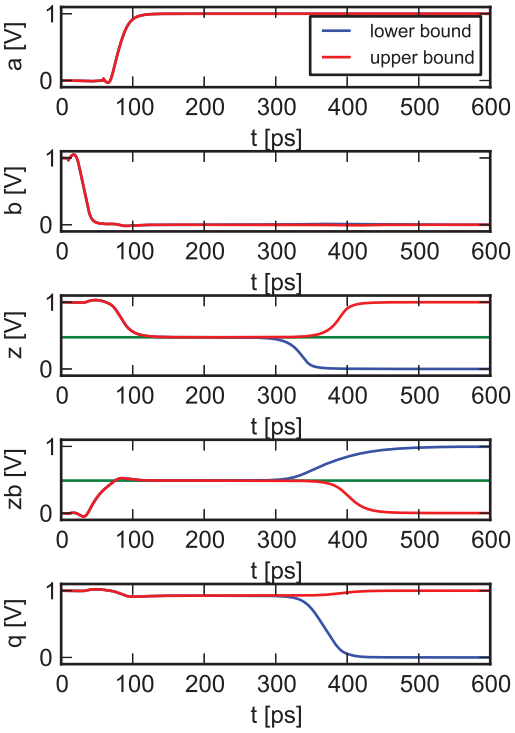
(a) Signal trace (case O1)



(b) Signal trace (case O2)

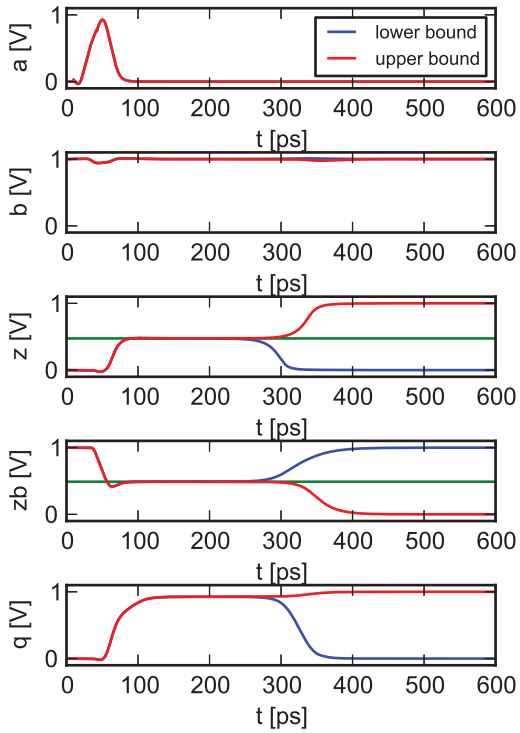


(c) Signal trace (case O3)

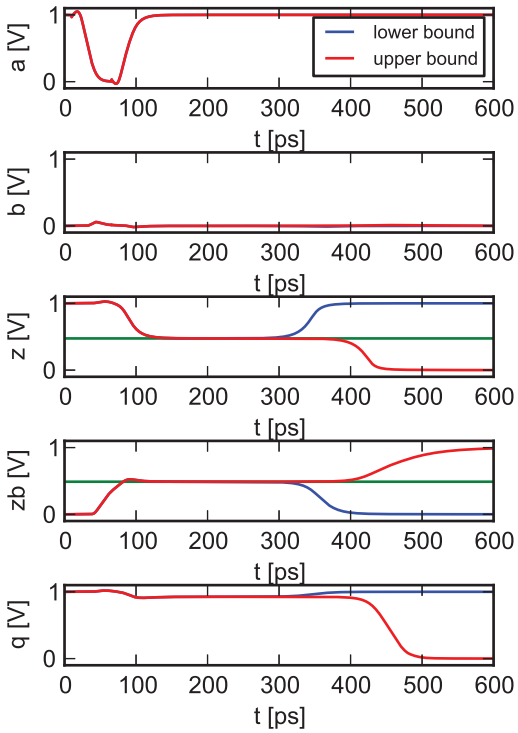


(d) Signal trace (case O4)

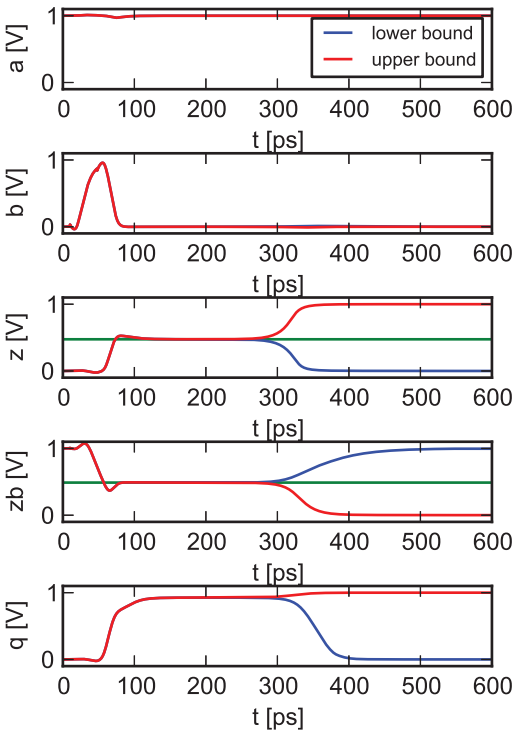
Figure A.33: Results of the weak feedback implementation with a high threshold output inverter (2)



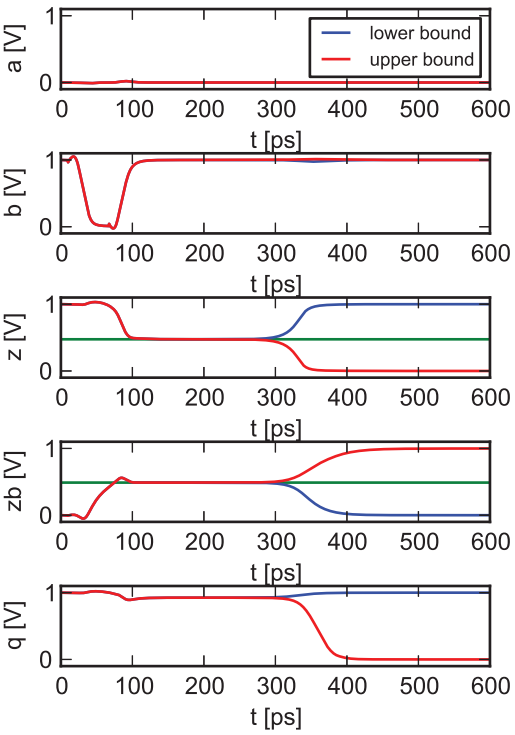
(a) Signal trace (case P1)



(b) Signal trace (case P2)

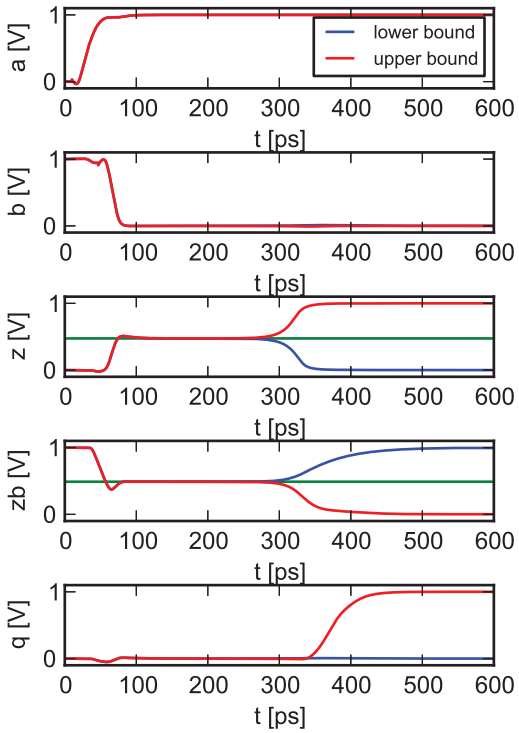


(c) Signal trace (case P3)

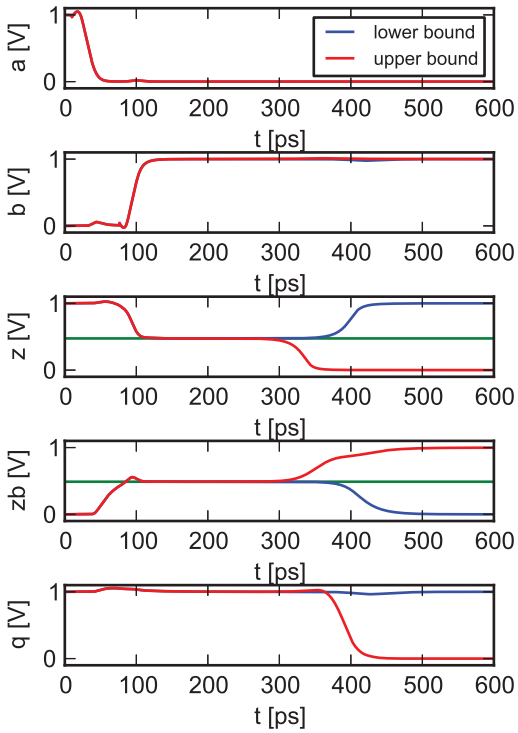


(d) Signal trace (case P4)

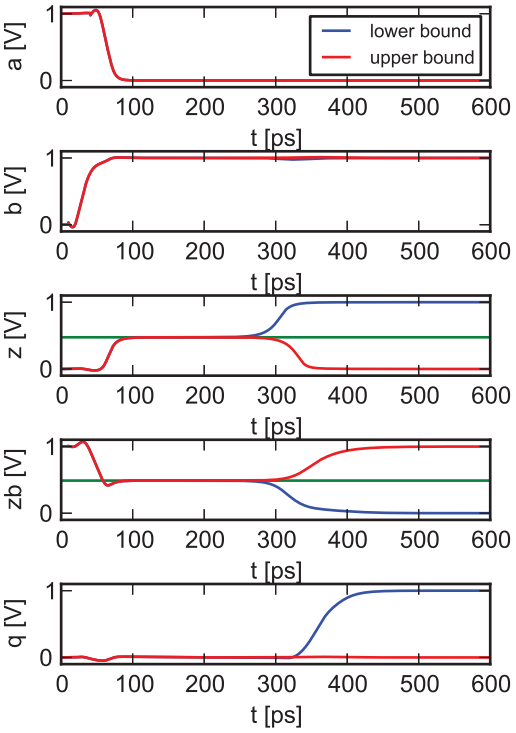
Figure A.34: Results of the weak feedback implementation with a high threshold output inverter (3)



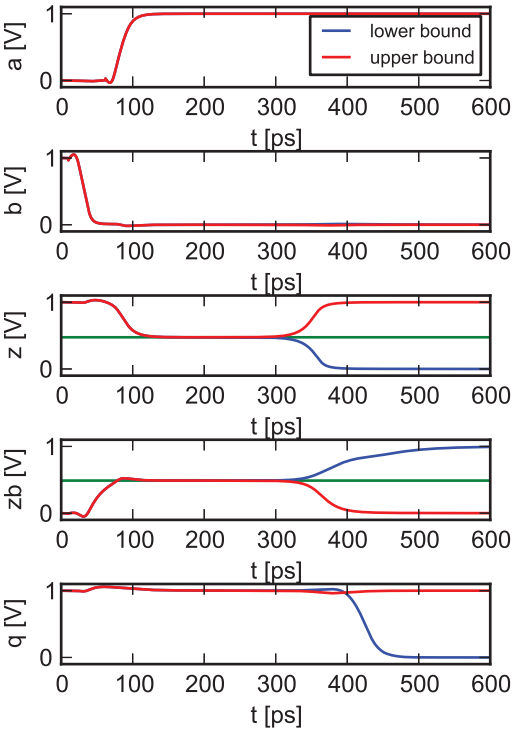
(a) Signal trace (case O1)



(b) Signal trace (case O2)

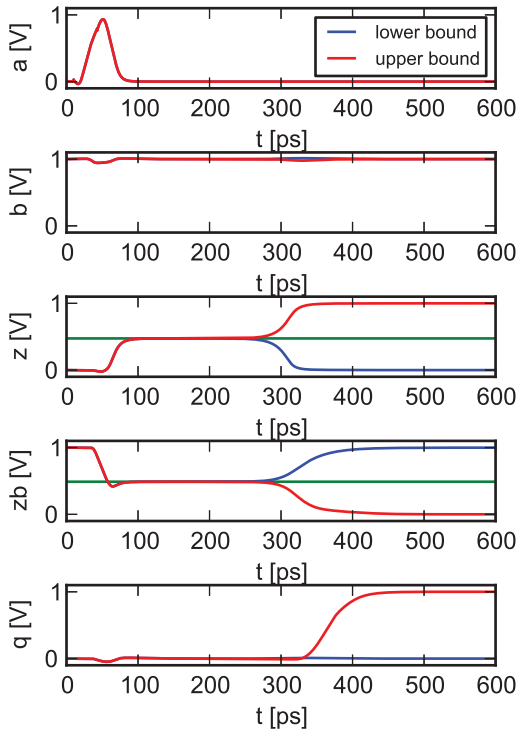


(c) Signal trace (case O3)

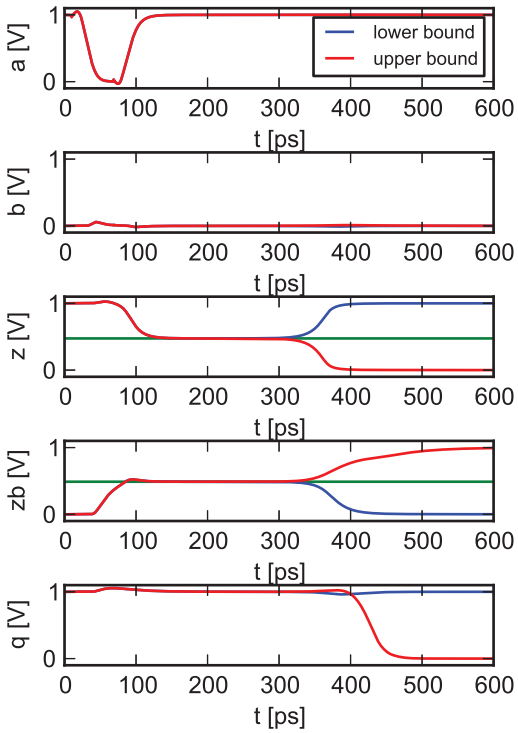


(d) Signal trace (case O4)

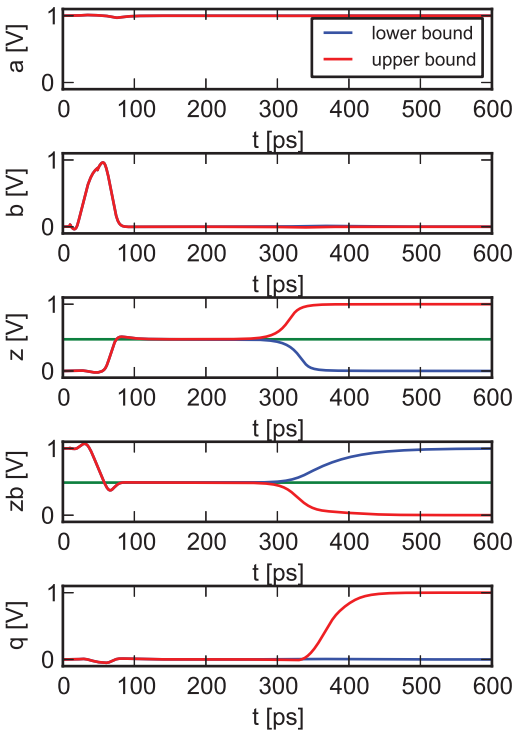
Figure A.35: Results of the weak feedback implementation with a Schmitt-trigger output (2)



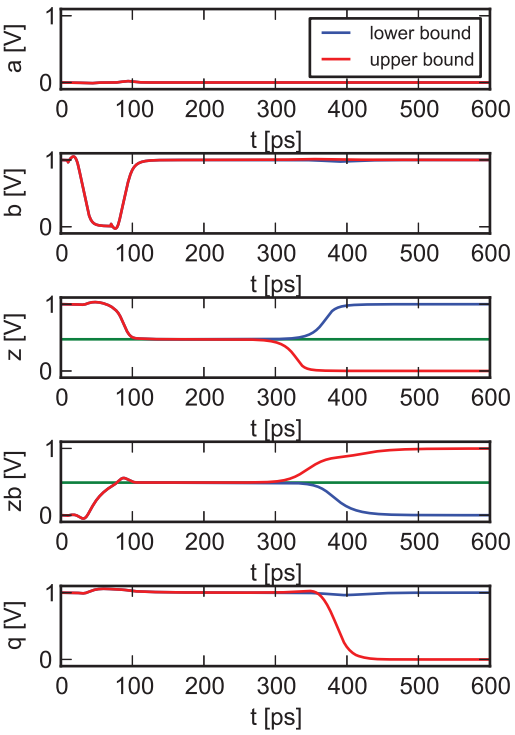
(a) Signal trace (case P1)



(b) Signal trace (case P2)



(c) Signal trace (case P3)



(d) Signal trace (case P4)

Figure A.36: Results of the weak feedback implementation with a Schmitt-trigger output (3)

B

Digital Simulation Model - Slave Latch Fitting

In this chapter we present the results of fitting the slave latch to our model. The same procedure as was used for the master latch in Section 8.4.5 is applied.

Model	Parameters
Model from [Vee80]	$\tau = 4.568339894953574ps$, $\theta = 1.129944562502486V/ps$, $t_0 = 65.0413098977594ps$, $\Delta t_{in_0} = 18.7137801312151ps$
Derived model	$\tau = 4.568339894953574ps$, $\Delta t_{in_0} = 18.7137801312151ps$, $c = 1V/ps$, $K = 0.01543141439328096$, $a = 0.07616479208867728$, $b = 0.577202024066728/ps$, $t_0 = 58.69674781345312ps$

Table B.1: Delay model parameters for slave latch (rising edges)

Model	Parameters
Model from [Vee80]	$\tau = 28.62777990417225ps$, $\theta = -1.008641821202886V/ps$, $t_0 = 34.91508033111081ps$, $\Delta t_{in_0} = 19.33412593485409ps$
Derived model	$a = 15.05143240903296$, $b = 0.00831871911626106/ps$, $\Delta t_{in_{02}} = 18.80017836855531ps$

Table B.2: Pulse length model parameters for slave latch

Model	Parameters
Model from [Vee80]	$\tau = 28.19253053828319ps$, $\theta = 0.2883072409501136V/ps$, $t_0 = 23.14815256515822ps$, $\Delta t_{in_0} = -0.997550017908745ps$
Derived model	$\tau = 28.19253053828319ps$, $\Delta t_{in_0} = -0.997550017908745ps$, $c = 1V/ps$, $K = 0.02300851829983843$, $a = 14.6544200669967$, $b = 10.91877254639688/ps$, $t_0 = 10.95212649345824ps$

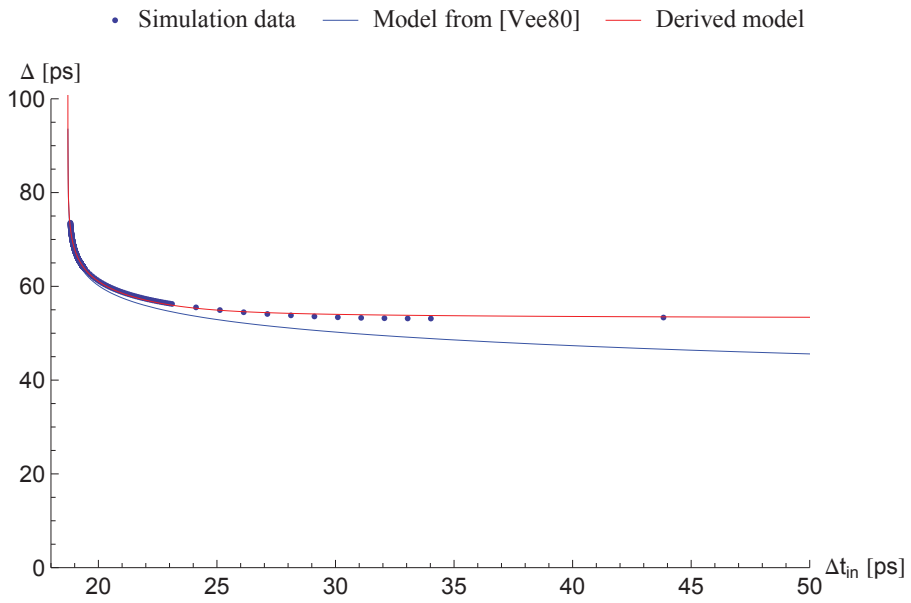
Table B.3: Delay model parameters for slave latch (falling edges)

Model	Parameters
Rising edges	$a = 0.007614644289329924$, $b = 0.0670426234672787/ps$, $c = 0.1083440582266819$, $d = -14.86016832429647ps$, $\Delta t_{in_0} = -88.8916592019543ps$, $f = 17.9568253669792ps$
Falling edges	$a = 0.02875526710089669$, $b = 0.07037724430255166/ps$, $c = 0.1804095783764784$, $d = -14.41121822640258ps$, $\Delta t_{in_0} = -65.68883712890381ps$, $f = 28.6395891142136ps$

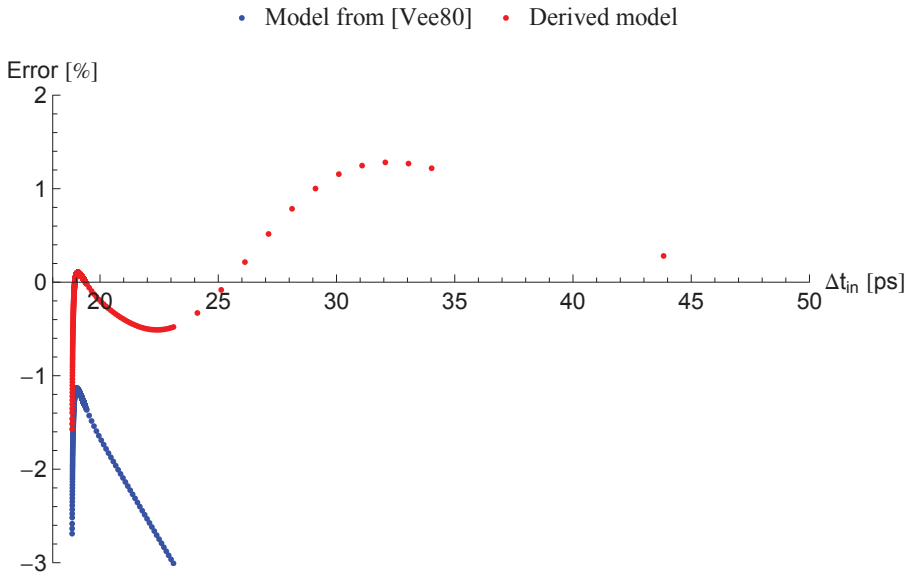
Table B.4: Enable model parameter for slave latch

Model	Parameters
High polarity	$a = -0.2710396061971605/ps$, $b = 1.013521134851046$, $d = 29.96779812313923ps$
Low polarity	$a = -0.3299553593436541/ps$, $b = 0.992047318403824$, $d = 31.36160218059379ps$

Table B.5: Pulse propagation model parameters for the transparent slave latch

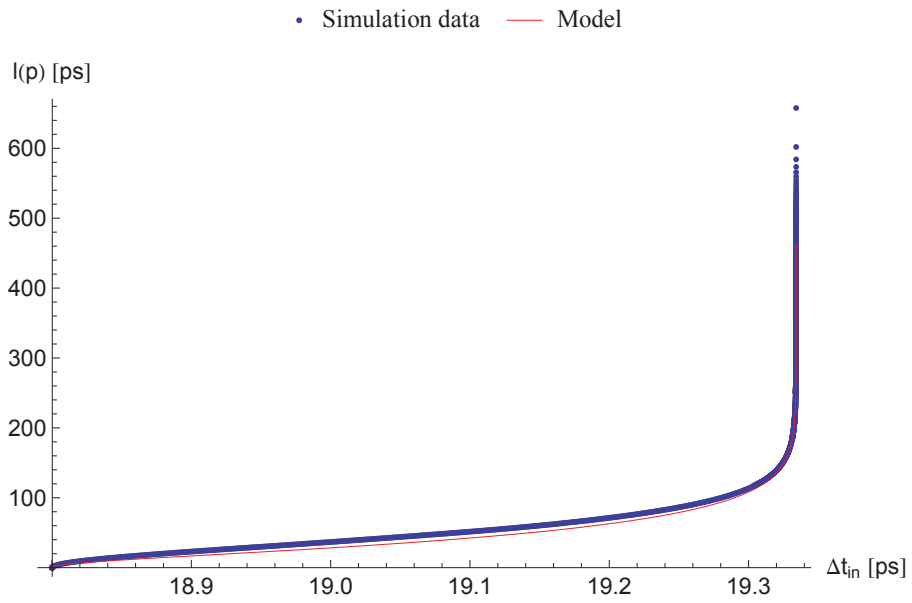


(a) Simulation result and fitted model

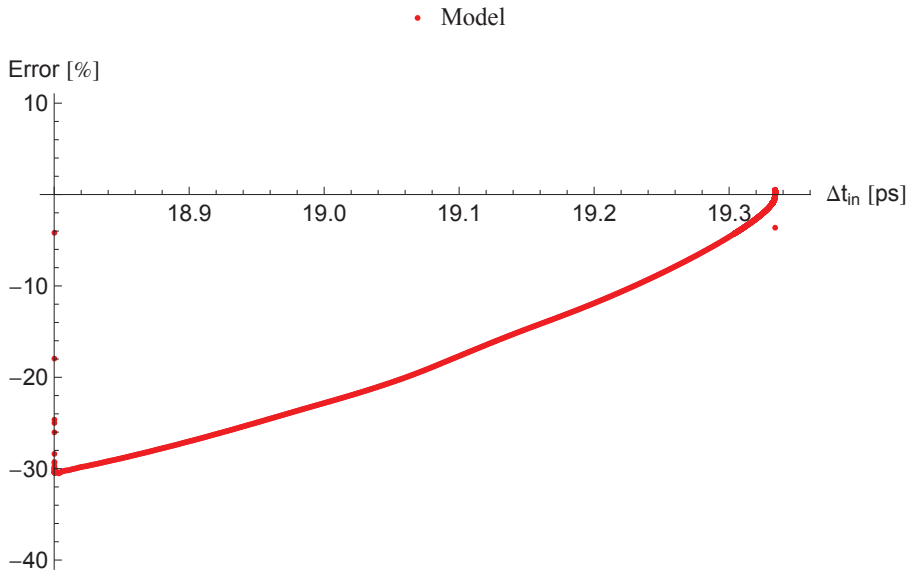


(b) Relative error for fitted model

Figure B.1: Model-fitting slave latch, rising edges

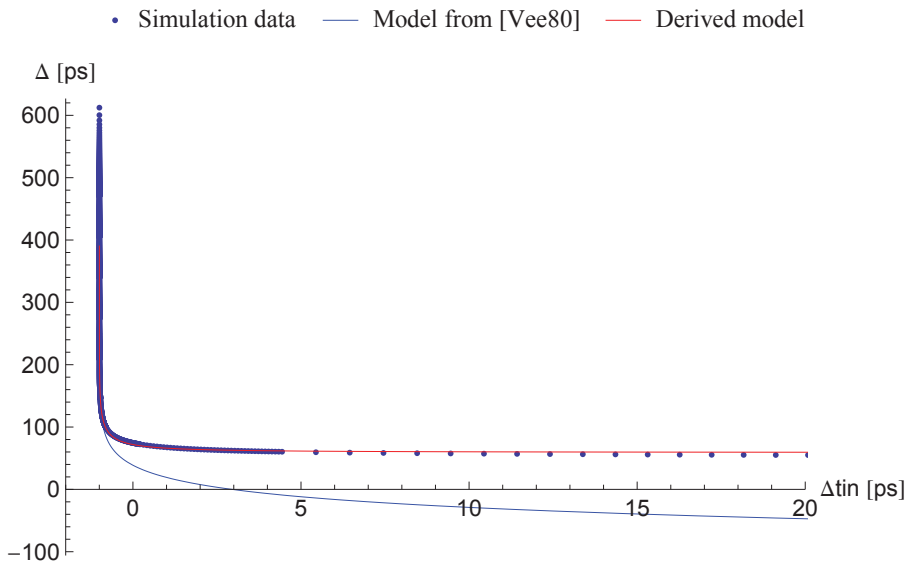


(a) Simulation result and fitted model

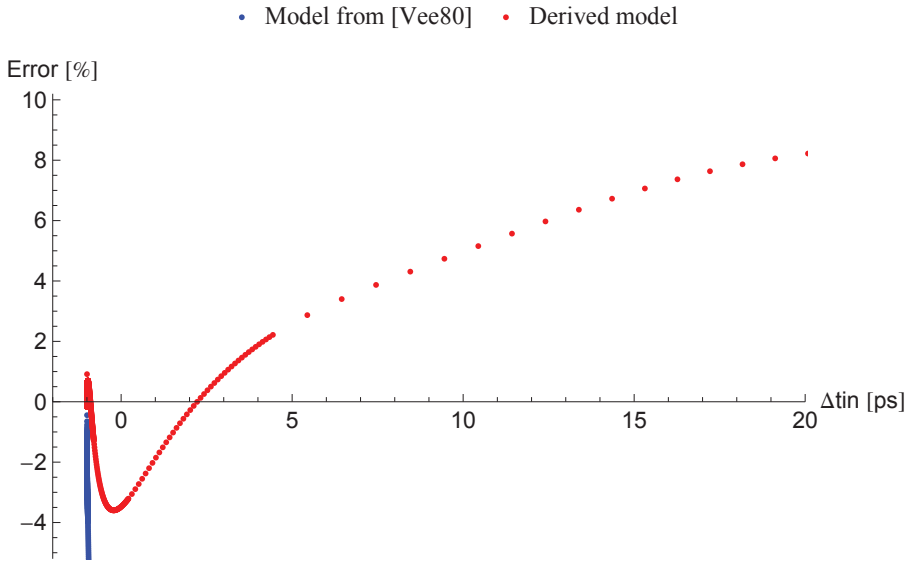


(b) Relative error for fitted model

Figure B.2: Model-fitting slave latch, rising edges (pulse model)

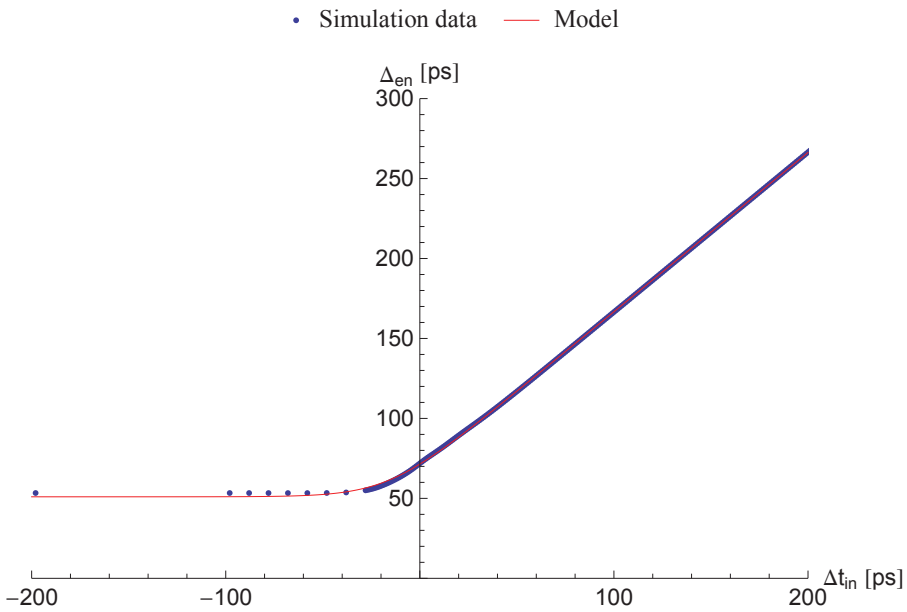


(a) Simulation result and fitted model

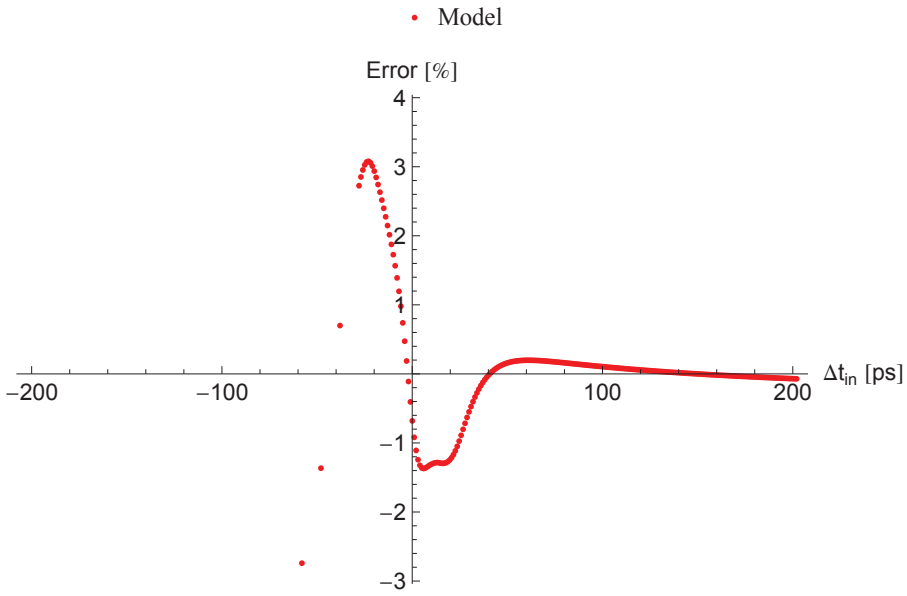


(b) Relative error for fitted model

Figure B.3: Model-fitting slave latch, falling edges

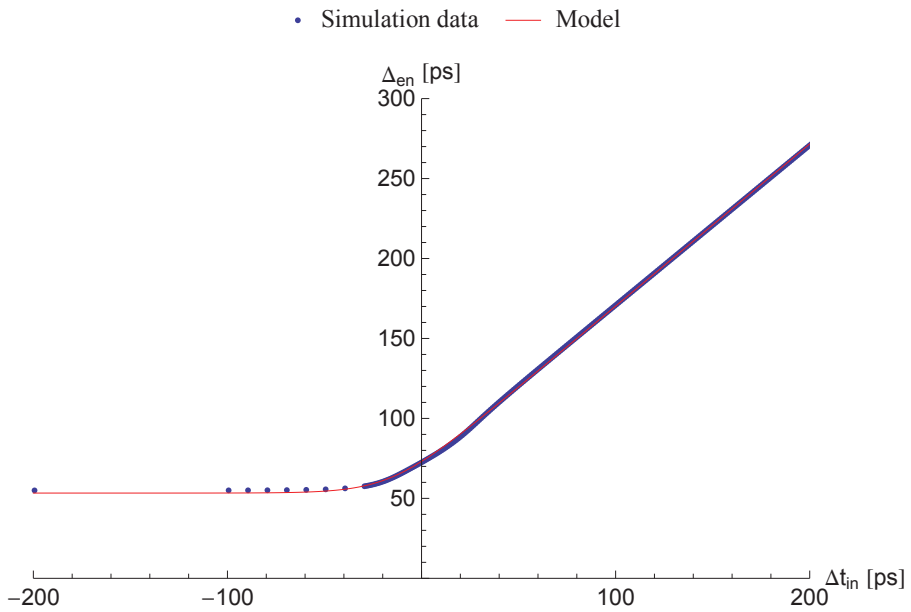


(a) Simulation result and fitted model

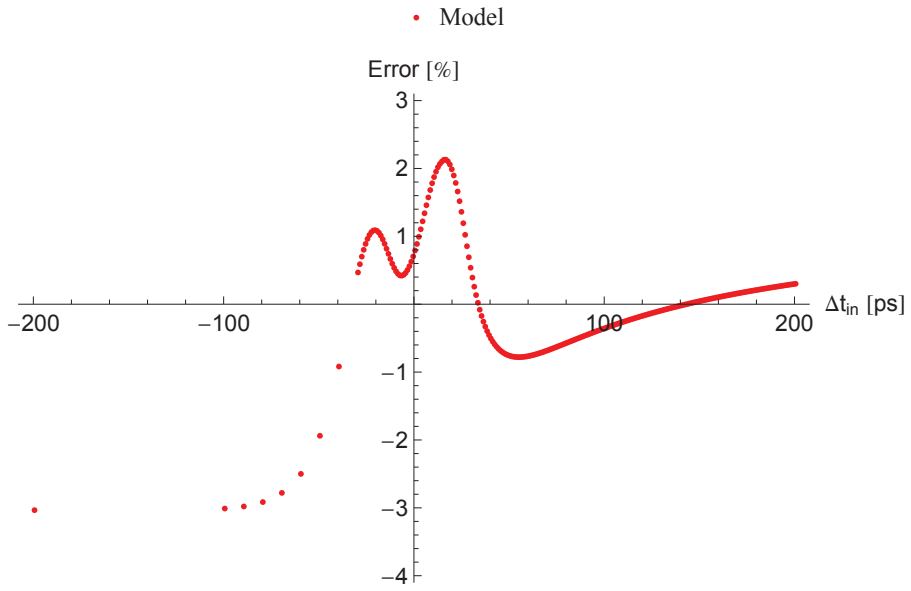


(b) Relative error for fitted model

Figure B.4: Model-fitting slave latch, rising edges (enable delay)



(a) Simulation result and fitted model



(b) Relative error for fitted model

Figure B.5: Model-fitting slave latch, falling edges (enable delay)

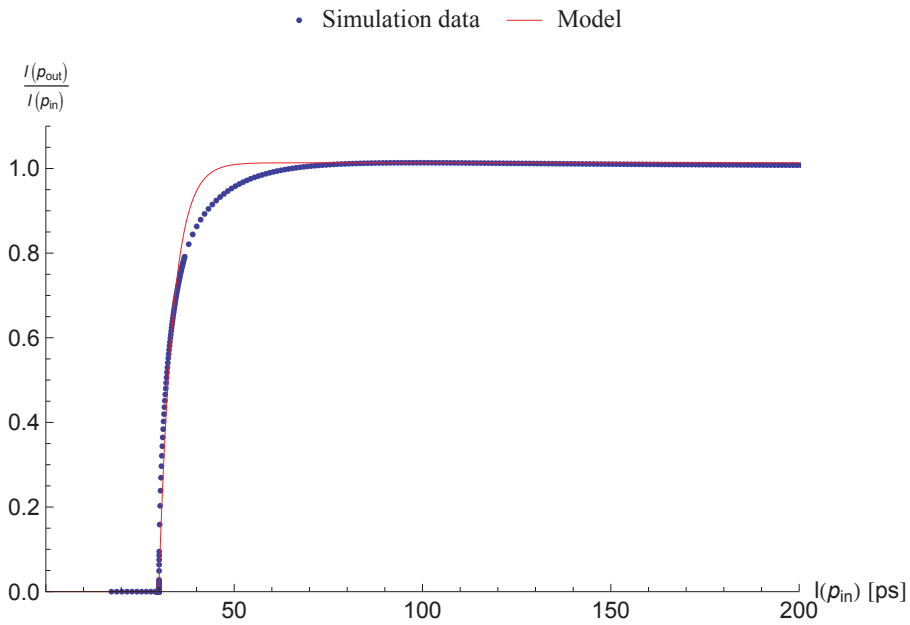


Figure B.6: Model-fitting slave latch, rising edges (pulse propagation)

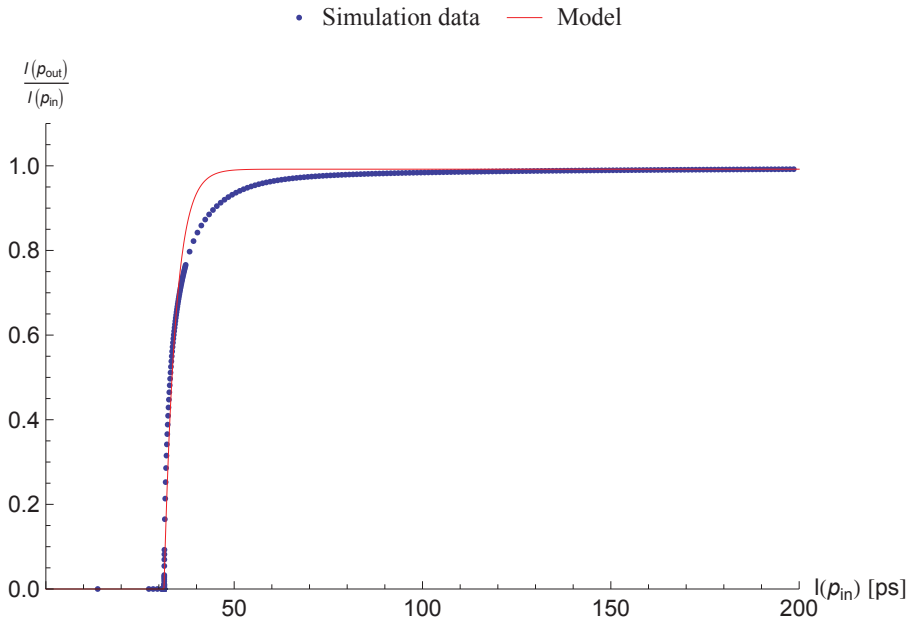


Figure B.7: Model-fitting slave latch, falling edges (pulse propagation)

Acronyms

ack	Acknowledge
ASIC	Application specific integrated circuit
BSIM	Berkeley Short-channel IGFET Model
CMOS	Complementary metal oxide semiconductor
DUT	Device under test
DLL	Delay locked loop
FPGA	Field programmable gate array
GND	Ground
IGFET	Insulated gate field effect transistor
LTD	Late transition detector
LTD	Late transition detection
LUT	Lookup table
NCL	Null convention logic
NMOS	n-type metal oxide semiconductor
PLL	Phase locked loop
PMOS	p-type metal oxide semiconductor
PVT	Process, voltage, technology
req	Request
SER	Soft error rate
SET	Single event transient
SEU	Single event upset
VDD	Power supply
VLSI	Very large scale integration

Bibliography

[AAW+11] N.M. Atkinson, J.R. Ahlbin, A.F. Witulski, N.J. Gaspard, W.T. Holman, B.L. Bhuva, E.X. Zhang, Li Chen, and L.W. Massengill. Effect of Transistor Density and Charge Sharing on Single-Event Transients in 90-nm Bulk CMOS. *IEEE Transactions on Nuclear Science*, 58(6):2578–2584, Dec. 2011.

[AKY09] M. Alshaikh, D. Kinniment, and A. Yakovlev. On the trade-off between resolution time and delay times in bistable circuits. In *16th IEEE International Conference on Electronics, Circuits, and Systems (ICECS2009)*, pages 355–358, 2009.

[Alf05] Peter Alfke. Metastable Recovery in Virtex-II Pro FPGAs, 2005.

[Alf08] Peter Alfke. Metastable Delay in Virtex FPGAs, 2008.

[Bau05] R.C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, Sept. 2005.

[BB97] M.P. Baze and S.P. Buchner. Attenuation of single event induced pulses in CMOS combinational logic. *IEEE Transactions on Nuclear Science*, 44(6):2217–2223, Dec 1997.

[BC09] J. Bhasker and Rakesh Chadha. *Static Timing Analysis for Nanometer Designs – A Practical Approach*. Springer, 2009.

[BCCZ13] Salomon Beer, Jerome Cox, Tom Chaney, and David M. Zar. MTBF Bounds for Multistage Synchronizers. In *19th International Symposium on Asynchronous Circuits and Systems (ASYNC2013)*, pages 158–165, 2013.

[BDM02] K.A. Bowman, S.G. Duvall, and J.D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *Solid-State Circuits, IEEE Journal of*, 37(2):183–190, 2002.

[BG13] Salomon Beer and Ran Ginosar. An Extended Metastability Simulation Method for Synchronizer Characterization. In Josef Ayala, Delong Shang, and Alex Yakovlev,

editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 7606 of *Lecture Notes in Computer Science*, pages 42–51. Springer Berlin Heidelberg, 2013.

[BGC+13] Salomon Beer, Ran Ginosar, Jerome Cox, Tom Chaney, and David M. Zar. Metastability challenges for 65nm and beyond; simulation and measurements. In *Conference Exhibition Design, Automation Test in Europe (DATE2013)*, pages 1297–1302, 2013.

[BGDW13] Salomon Beer, Ran Ginosar, Rostislav Dobkin, and Yoav Weizman. MTBF Estimation in Coherent Clock Domains. In *19th International Symposium on Asynchronous Circuits and Systems (ASYNC2013)*, pages 166–173, 2013.

[BGP+10] S Beer, R Ginosar, M Priel, R Dobkin, and A Kolodny. The Devolution of Synchronizers. In *Symposium on Asynchronous Circuits and Systems*, pages 94–103, 2010.

[BKD08] Manjul Bhushan, Mark B Ketchen, and Koushik K Das. CMOS Latch Metastability Characterization at the 65-nm-Technology Node. In *Conf. on Microel. Test Struct.*, pages 147–151, 2008.

[Cat66] Ivor Catt. Time Loss Through Gating of Asynchronous Logic Signal Pulses. *IEEE Transactions on Electronic Computers*, EC-15(1):108–111, 1966.

[Cha84] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford Univ., 1984.

[CM73] Thomas J Chaney and Charles E Molnar. Anomalous Behavior of Synchronizer and Arbiter Circuits. *Transactions on Computers*, C 22(4):421–422, 1973.

[Con03] Cristian Constantinescu. Trends and Challenges in VLSI Circuit Reliability. In *IEEE Micro*, volume 23, 2003.

[CRL+03] K.A. Clark, A.A. Ross, H.H. Loomis, T.R. Weatherford, D.J. Fouts, S.P. Buchner, and D. McMorrow. Modeling single-event effects in a complex digital device. *IEEE Transactions on Nuclear Science*, 50(6):2069–2080, Dec. 2003.

[CSC+10] Doris Chen, Deshanand Singh, Jeffrey Chromczak, David Lewis, Ryan Fung, David Neto, and Vaughn Betz. A Comprehensive Approach to Modeling, Characterizing and Optimizing for Metastability in FPGAs. In *Symposium on Field Programmable Gate Arrays*, pages 167–176. ACM, 2010.

[CVK+12] Y.S. Chauhan, S. Venugopalan, M.A. Karim, S. Khandelwal, N. Paydavosi, P. Thakur, A.M. Niknejad, and C.C. Hu. BSIM 2014; Industry standard compact MOSFET models. In *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, pages 30–33, 2012.

[DT10] William J. Dally and Stephen G. Tell. The Even/Odd Synchronizer: A Fast, All-Digital, Periodic Synchronizer. In *IEEE Symposium on Asynchronous Circuits and Systems (ASYNC2010)*, pages 75–84, 2010.

[DW11] A. Dixit and A. Wood. The impact of new technology on soft error rates. In *IEEE International Reliability Physics Symposium (IRPS2011)*, pages 5B.4.1 –5B.4.7, april 2011.

[DY07] S. Dasgupta and A. Yakovlev. Comparative analysis of GALS clocking schemes. *Computers Digital Techniques, IET*, 1(2):59–69, 2007.

[FB96] K.M. Fant and S.A. Brandt. NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. In *ASAP 96. Proceedings of International Conference on Application Specific Systems, Architectures and Processors, 1996.*, pages 261–273, Aug 1996.

[FFS09] Gottfried Fuchs, Matthias Fuegger, and Andreas Steininger. On the Threat of Metastability in an Asynchronous Fault-Tolerant Clock Generation Scheme. Research Report 11/2009, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2009.

[FH90] Paul Forshaw and Reinhard Hahn. Synchronous Design: The Right Technique for Digital ASIC's. In *ASIC Seminar and Exhibit*, pages P6/1.1 – P6/1.5, 1990.

[Fol96] C. Foley. Characterizing Metastability. In *Second International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC1996)*, pages 175–184, 1996.

[Fri01] Eby G. Friedman. Clock Distribution Networks in Synchronous Digital Integrated Circuits. In *Proceedings of the IEEE*, volume 89, pages 665–692, 2001.

[GAN+11] M.J. Gadlage, J.R. Ahlbin, B. Narasimham, B.L. Bhuva, L.W. Massengill, and R.D. Schrimpf. Single-Event Transient Measurements in nMOS and pMOS Transistors in a 65-nm Bulk CMOS Technology at Elevated Temperatures. *IEEE Transactions on Device and Materials Reliability*, 11(1):179–186, March 2011.

[Gin03] R. Ginosar. Fourteen Ways to Fool Your Synchronizer. In *Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC2003)*, pages 89–96, 2003.

[Gin11] R. Ginosar. Metastability and Synchronizers: A Tutorial. *IEEE Design Test of Computers*, 28(5):23–35, 2011.

[GYB07] K T Gardiner, Alex Yakovlev, and A Bystrov. A C-element Latch Scheme with Increased Transient Fault Tolerance for Asynchronous Circuits. In *13th IEEE International On-Line Testing Symposium, 2007 (IOLTS2007)*, pages 223–230, 2007.

[Hau95] S. Hauck. Asynchronous Design Methodologies: An Overview. *Proceedings of the IEEE*, 83(1):69–93, Jan 1995.

[Huf57] David A. Huffman. The Design and Use of Hazard-Free Switching Networks. *Journal of the ACM*, 4(1):47–62, jan 1957.

[IEE93] IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std-logic1164). *IEEE Std 1164-1993*, 1993.

[JYG09] Ian W Jones, Suwen Yang, and Mark R Greenstreet. Synchronizer Behavior and Analysis. In *Symposium on Asynchronous Circuits and Systems*, pages 117–126, 2009.

[Kae08] Hubert Kaeslin. *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*. Cambridge University Press, 2008.

[KBY02] D.J. Kinniment, A. Bystrov, and A.V. Yakovlev. Synchronization Circuit Performance. *IEEE Journal of Solid-State Circuits*, 37(2):202–209, 2002.

[KC87a] Lindsay Kleeman and Antonio Cantoni. Metastable Behavior in Digital Systems. *IEEE Design and Test of Computers*, 4(6):4–19, 1987.

[KC87b] Lindsay Kleeman and Antonio Cantoni. On the Unavoidability of Metastable Behavior in Digital Systems. *IEEE Transactions on Computers*, C-36(1):109–112, 1987.

[KD90] Lee-Sup Kim and R.W. Dutton. Metastability of CMOS Latch/Flip-Flop. *IEEE Journal of Solid-State Circuits*, 25(4):942–951, 1990.

[KGD07] M. Kayam, R. Ginosar, and C.E. Dike. Symmetric Boost Synchronizer for Robust Low Voltage, Low Temperature Operation. Technical report, Technion, 2007.

[KHP04] T. Karnik, P. Hazucha, and J. Patel. Characterization of soft errors caused by single event upsets in CMOS processes. *IEEE Transactions on Dependable and Secure Computing*, 1(2):128–143, April-June 2004.

[KHR06] David J Kinniment, Keith Heron, and Gordon Russell. Measuring Deep Metastability. In *Symposium on Asynchronous Circuits and Systems*, pages 2–11, 2006.

[Kin07] David J Kinniment. *Synchronization and Arbitration in Digital Systems*. Wiley, 2007.

[KJ04] Jozef Kalisz and Zbigniew Jachna. Dual-Edge Late-Transition Detector for Testing the Metastability Effect in Flip-Flops. In *Conference on Microelectronics*, pages 780–782, 6-8 2004.

[Kle09] A. J. Kleinosowski. Method for soft error modeling with double current pulse. US Patent US 7627840 B2, International Business Machine Corporation, 2009.

[KW76] David J Kinniment and J V Woods. Synchronisation and arbitration circuits in digital systems. In *Proceedings of the Institution of Electrical Engineers*, pages 961–966, 1976.

[KZYD10] Weidong Kuang, Peiyi Zhao, J S Yuan, and R F DeMara. Design of Asynchronous Circuits for High Soft Error Tolerance in Deep Submicrometer CMOS Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(3):410–422, 2010.

[Mar81] Leonard R. Marino. General Theory of Metastable Operation. *IEEE Transactions on Computers*, C-30(2):107–115, 1981.

[MB59] D. E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proceedings of an International Symposium on the Theory of Switching*. Harvard University Press, 1959.

[ME07] D.G. Mavis and P.H. Eaton. SEU and SET modeling and mitigation in deep submicron technologies. In *Proceedings 45th Annual IEEE International Reliability physics symposium*, pages 293–305, April 2007.

[Moo75] Gordon E. Moore. Progress in digital integrated electronics. In *International Electron Devices Meeting*, volume 21, pages 11–13, 1975.

[MRL05] Y Monnet, M Renaudin, and R Leveugle. Asynchronous Circuits Transient Faults Sensitivity Evaluation. In *42nd Design Automation Conference, 2005 (DAC2005)*, pages 863–868, 2005.

[Mur07] Saburo Muroga. Sequential Networks. In Wai-Kai Chen, editor, *The VLSI Handbook*, chapter 31. CRC Press, 2007.

[NP03] M. Nicolaidis and R. Perez. Measuring the width of transient pulses induced by ionising radiation. *41st Annual IEEE International Reliability Physics Symposium*, pages 56–59, 2003.

[PCV12] J Pontes, N Calazans, and P Vivet. Adding Temporal Redundancy to Delay Insensitive Codes to Mitigate Single Event Effects. In *18th IEEE International Symposium on Asynchronous Circuits and Systems, 2012 (ASYNC2012)*, pages 142–149, 2012.

[Pec76] Miroslav Pechoucek. Anomalous Response Times of Input Synchronizers. *IEEE Transactions on Computers*, C-25(2):133–139, 1976.

[PG07] Ivan Miro Panades and Alain Greiner. Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures. In *First International Symposium on Networks-on-Chip, 2007. NOCS 2007.*, pages 83–94, May 2007.

[PM05] Song Peng and R Manohar. Efficient Failure Detection in Pipelined Asynchronous Circuits. In *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005 (DFT2005)*, pages 484–493, 2005.

[Pol09] Thomas Polzer. Fault-Tolerant Hardware Implementation of a Consensus Algorithm. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2009.

[PS13a] Thomas Polzer and Andreas Steininger. An Approach for Efficient Metastability Characterization of FPGAs through the Designer. In *19th IEEE International Symposium on Asynchronous Circuits and Systems, 2013 (ASYNC2013)*, pages 174–182, 2013.

[PS13b] Thomas Polzer and Andreas Steininger. Digital Late-Transition Metastability Simulation Model. In *16th EUROMICRO Conference on Digital System Design*, 2013.

[PS13c] Thomas Polzer and Andreas Steininger. Metastability Characterization for Muller C-Elements. In *23rd International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2013.

[PS13d] Thomas Polzer and Andreas Steininger. SET Propagation in Micropipelines. In *23rd International Workshop on Power and Timing Modeling, Optimization and Simulation*, 2013.

[PSL13] Thomas Polzer, Andreas Steininger, and Jakob Lechner. Muller C-Element Metastability Containment. In JosL. Ayala, Delong Shang, and Alex Yakovlev, editors, *22nd International Workshop on Power and Timing Modeling, Optimization and Simulation*, volume 7606 of *Lecture Notes in Computer Science*, pages 103–112. Springer Berlin Heidelberg, 2013.

[RC82] Fred Rosenberger and Thomas J Chaney. Flip-Flop Resolving Time Test Circuit. *Journal of Solid-State Circuits*, 17(4):731–738, 1982.

[RCI94] G.L. Ries, G.S. Choi, and R.K. Iyer. Device-level transient fault modeling. In *Twenty-Fourth International Symposium on Fault-Tolerant Computing (FTCS-24)*, pages 86–94, Jun 1994.

[RD93] A Rothermal and F Dell'ova. Analog Phase Measuring Circuit for Digital CMOS ICs. *Journal of Solid-State Circuits*, 28(7):853–856, 1993.

[RDK+08] R. Ramanarayanan, V. Degalahal, R. Krishnan, J. Kim, V. Narayanan, Y. Xie, M. Irwin, and K. Unlu. Modeling Soft Errors at Device and Logic Level for combinational circuits. *IEEE Transactions on Dependable and Secure Computing*, 2008. (to appear).

[RSS+10] Branka M Rogina, Peter Skoda, Karolj Skala, Ivan Michieli, Maja Vlah, and Sinisa Marijan. Metastability Testing at FPGA Circuit Design using Propagation Time Characterization. In *East-West Design and Test Symposium*, pages 80–85, 17-20 2010.

[SEE96] Maitham Shams, Jo C. Ebergen, and Mohamed I. Elmasry. A Comparison of CMOS Implementations of an Asynchronous Circuits Primitive: the C-Element. In *International Symposium on Low Power Electronics and Design*, pages 93–96, 1996.

[Sei79] Charles L. Seitz. System Timing. In Carver Mead and Lynn Conway, editors, *Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1979.

[SF01] Jens Sparso and Steve Furber. *Principles of Asynchronous Circuit Design - A Systems Perspective*. Kluwer Academic Publishers, 2001.

[SFGP09] Yebin Shi, S. B. Furber, J. Garside, and L. A. Plana. Fault Tolerant Delay Insensitive Inter-chip Communication. In *15th IEEE Symposium on Asynchronous Circuits and Systems*, pages 77–84, May 2009.

[SG03] Yaron Semiat and Ran Ginosar. Timing Measurements of Synchronization Circuits. In *Symposium on Asynchronous Circuits and Systems*, pages 68–77. IEEE Computer Society, 2003.

[Sut89] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[TBC+97] Yuan Taur, D.A. Buchanan, Wei Chen, D.J. Frank, K.E. Ismail, Shih-Hsien Lo, G.A. Sai-Halasz, R.G. Viswanathan, H.-J.C. Wann, S.J. Wind, and Hon-Sum Wong. Cmos scaling into the nanometer regime. *Proceedings of the IEEE*, 85(4):486–504, 1997.

[TGL07] P. Teehan, M. Greenstreet, and G. Lemieux. A Survey and Taxonomy of GALS Design Styles. *Design and Test of Computers, IEEE*, 2007.

[TY10] Ghaith Tarawneh and Alex Yakovlev. Modified Bisection Search for Faster Metastability Characterization. Technical Report NCL-EECE-MSD-TR-2010-154, Microelectronic System Design Group, School of EECE, Newcastle University, 2010.

[Vee80] Harry J M Veendrick. The Behavior of Flip-Flops Used as Synchronizers and Prediction of Their Failure Rate. *Journal of Solid-State Circuits*, 15(2):169–176, 1980.

[VPSS12] V.S. Veeravalli, T. Polzer, A. Steininger, and U. Schmid. Architecture and Design Analysis of a Digital Single-Event Transient/Upset Measurement Chip. In *15th Euromicro Conference on Digital System Design (DSD2012)*, pages 8–17, 2012.

[WMZ+09] Jie Wu, Yichao Ma, Jie Zhang, Yang Kong, Hongzhi Song, and Xiaoquan Han. Research on Metastability Based on FPGA. In *Conference on Electronic Measurement & Instruments*, page 4, 2009.

[YG07] Suwen Yang and M.R. Greenstreet. Simulating Improbable Events. In *44th ACM/IEEE Design Automation Conference (DAC2007)*, pages 154–157, 2007.

[YHHW99] Nian Yang, W.K. Henson, John R. Hauser, and Jimmie J. Wortman. Modeling Study of Ultrathin Gate Oxides Using Direct Tunneling Current and Capacitance-Voltage Measurements in MOS Devices. *IEEE Transactions on Electron Devices*, 46(7):1464–1471, 1999.

[YJG11] Suwen Yang, Ian W. Jones, and Mark R. Greenstreet. Synchronizer Performance in Deep Sub-Micron Technology. In *17th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC2011)*, pages 33–42, 2011.

[ZKD+08] Jun Zhou, David J Kinniment, Charles E Dike, Gordon Russell, and Alex Yakovlev. On-Chip Measurement of Deep Metastability in Synchronizers. *Journal of Solid-State Circuits*, 43(2):550–557, 2008.

[ZKRY06] Jun Zhou, D. Kinniment, G. Russell, and A. Yakovlev. A robust synchronizer. In *IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, volume 00, pages 2 pp.–, 2006.

[ZKRY08] Jun Zhou, David J Kinniment, Gordon Russell, and Alex Yakovlev. Adapting Synchronizers to the Effects of on Chip Variability. In *Symposium on Asynchronous Circuits and Systems*, pages 39–47, 2008.

Curriculum Vitae

Personal data

First name	Thomas
Last name	Polzer
Year of birth	1980
Citizenship	Austria

Education

2003-2006	Bachelor program in Computer Engineering Vienna University of Technology Graduation with distinction
2007-2009	Master program in Computer Engineering Vienna University of Technology Graduation with distinction
2009-2013	PhD program in Computer Engineering Vienna University of Technology

Career

2000-2003	Software developer COMMARO mobile trading systems Klagenfurt, Austria
2005	Tutor in the course Project Management Vienna University of Technology
2005-2006	Tutor in the course Softwareengineering Vienna University of Technology
2009-ongoing	Research assistant Vienna University of Technology

Awards

Feb. 2007	University Award 2007, third prize Embedded World, Nürnberg Bachelor thesis: Graphical Microcontroller Programming Environment (GMCP)
Dec. 2009	Distinguished Young Alumnus Award Faculty of Informatics, Vienna University of Technology Diploma thesis: Fault-Tolerant Hardware Implementation of a Consensus Algorithm
Dec. 2009	Frequentis Förderpreis Diploma thesis: Fault-Tolerant Hardware Implementation of a Consensus Algorithm
Sep. 2012	Best paper award, DSD conference 2012 Varadan Savulimedu Veeravalli, Thomas Polzer, Ulrich Schmid and Andreas Steininger – Architecture and Design Analysis of a Digital Single-Event Transient/Upset Measurement Chip