Inter-Module Interfacing Techniques for SoCs with Multiple Clock Domains to Address Challenges in Modern Deep Sub-Micron Technologies

Syed Rafay Hasan

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

December 2009

Your file Votre référence
ISBN: 978-0-494-67368-3
Our file Notre référence
ISBN: 978-0-494-67368-3

Abstract

Inter-Module Interfacing Techniques for SoCs with Multiple Clock Domains to Address Challenges in Modern Deep Sub-Micron Technologies

Syed Rafay Hasan, Ph. D.
Concordia University, 2009

Miniaturization of integrated circuits (ICs) due to the improvement in lithographic techniques in modern deep sub-micron (DSM) technologies allows several complex processing elements to coexist in one IC, which are called System-on-Chip. As a first contribution, this thesis quantitatively analyzes the severity of timing constraints associated with Clock Distribution Network (CDN) in modern DSM technologies and shows that different processing elements may work in different clock domains to alleviate these constraints. Such systems are known as Globally Asynchronous Locally Synchronous (GALS) systems.

It is imperative that different processing elements of a GALS system need to communicate with each other through some interfacing technique, and these interfaces can be asynchronous or synchronous. Conventionally, the asynchronous interfaces are described at the Register Transfer Logic (RTL) or system level. Such designs are susceptible to certain design constraints that cannot be addressed at higher abstraction levels; crosstalk glitch is one such constraint. This thesis initially identifies, using an analytical model, the possibility of asynchronous interface malfunction due to crosstalk glitch propagation. Next, we characterize crosstalk glitch propagation under normal

operating conditions for two different classes of asynchronous protocols, namely bundled data protocol based and delay insensitive asynchronous designs. Subsequently, we propose a logic abstraction level modeling technique, which provides a framework to the designer to verify the asynchronous protocols against crosstalk glitches. The utility of this modeling technique is demonstrated experimentally on a Xilinx Virtex-II Pro FPGA. Furthermore, a novel methodology is proposed to quench such crosstalk glitch propagation through gating the asynchronous interface from sending the signal during potential glitch vulnerable instances. This methodology is termed as crosstalk glitch gating. This technique is successfully applied to obtain crosstalk glitch quenching in the representative interfaces.

This thesis also addresses the clock skew challenges faced by high-performance synchronous interfacing methodologies in modern DSM technologies. The proposed methodology allows communicating modules to run at a frequency that is independent of the clock skew. Leveraging a novel clock-scheduling algorithm, our technique permits a faster module to communicate safely with a slower module without slowing down. Safe data communications for mesochronous schemes and for the cases when communicating modules have clock frequency ratios of integer or coprime numbers are theoretically explained and experimentally demonstrated. A clock-scheduling technique to dynamically accommodate phase variations is also proposed. These methods are implemented to the Xilinx Virtex II Pro technology. Experiments prove that the proposed interfacing scheme allows modules to communicate data safely, for mesochronous schemes, at 350 MHz, which is the limit of the technology used, under a clock skew of more than twice the time period (i.e. a clock skew of 12 ns).

To My Parents

Acknowledgements

This thesis could not have reached its conclusion without the support of many individuals. I take this opportunity to express my gratitude to all these people.

I am deeply indebted and grateful to Dr. Yvon Savaria and Dr. M. Omair Ahmad for their excellent supervision. Dr. Yvon Savaria supervised me with great conviction and patience. His deep and insightful knowledge of the digital design prevented me from straying in wrong directions. His guidance was really instrumental in polishing my technical skills. I am also very fortunate to have had Dr. M. Omair Ahmad as an affectionate mentor to turn to for guidance. His thoughtful suggestions assisted me tremendously in my research. And his scholarly advices were always a source of inspiration for me.

I am thankful to all the committee members for accepting to be on my doctoral advisory committee. It is an honour for me to have a scholar like Dr. Manoj Sachdev as my external examiner. I am highly indebted that he obliged our request to attend my thesis examination, in spite of his busy schedule. I would also like to thank Dr. Asim J. Al-Khalili, Dr. Ibrahim Galal Hassan and Dr. Otmane Ait Mohamed for their constructive feedback during the course of my studies.

I started my research under the supervision of Dr. Mohamed Nekili. I would like to thank him for being instrumental in the initial stages of my post graduation pursuits.

I would also like to thank my colleagues, mentors and friends, with special thanks to Dr. Normand Bélanger and Dr. Osman Hasan for their constructive inputs, Mr. Tadeusz Obuchowicz for spending endless hours with me in supporting EDA tools, my friend Bill

Pontikakis for his invaluable peer advice, and all the VLSI lab colleagues in Concordia University and École Polytechnique de Montréal.

I cannot thank my parents enough who selflessly supported all my decisions. My father, a scholar himself, took a lot of interest in my studies. Their constant encouragement was a great source of inspiration for me. All my siblings were supportive of me. In particular, my eldest brother Syed Ziaul Hasan was very kind in uplifting my spirits. I would like to mention the extended support from my in-laws. My parents-in-law showed enormous faith in my abilities, and their unconditional trust helped me a lot in achieving my goals.

Last, but not by any means least, I want to express my gratitude to my wife. I cannot imagine how this thesis could have reached its conclusion, had it not been for her understanding, endurance and support. And the new inclusion in our family, our daughter, has been a source of enormous zeal and energy for me.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Acronyms

AQX	Aggressor-to-Quiet-line Crosstalk
ASIP	Application Specific Instruction set Processors
CDN	Clock Distribution Network
Con_CLK	Conventional Clock
DDPS	Direct Digital Period Synthesis
DDS	Direct Digital Synthesis
DI	Delay Insensitive
DSM	Deep Sub-Micron
FIFO	First In First Out
F-to-S	Fast-to-Slow
GALS	Globally Asynchronous Locally Synchronous
GP	Glitch Propagation
GUM	Globally Updated Mesochronous
IP RX	Intellectual Property Receiver Side
IP TX	Intellectual Property Transmission Side
MCD	Multiple Clock Domains
MPSoC	Multi Processor SoCs
MTBF	Mean Time Between Failures
NRZ	Non-Return-to-Zero
NXE	No-Crosstalk glitch Effect
QBW	Quadruple Bus Width
RCU	Receiver Control Unit
RR(X)	Receiver Register(X)
RSMCLK	Receiver State Machine Clock
RTL	Register Transfer Logic
RTZ	Return-to-Zero
SCU	Sender Control Unit
SLID	Synchronous Latency Insensitive Design

SoC	System-on-Chip
SR(X)	Sender Register(X)
SSMCLK	Sender State Machine Clock
S-to-F	Slow-to-Fast
STSS	Self-Test Self-Synchronizing
WGE	Wire Glitch Element
WXE	With Crosstalk glitch Effect

Chapter 1: Introduction

1.1. Design-Related Technical Issues with SoCs in DSM Technologies

Advances in integrated circuit technology have revolutionized modern day electronic appliances. Laptops, PDAs, cell phones, GPS, and so many other electronic devices are now part of our daily lives. All these devices use higher and higher integration levels, along with a tremendous increase in functionalities and features. An electronic device that supports multiple features often requires a separate processing element for each of these features. Every processing element may be designed independently by a different vendor and may have some unique characteristics. Such independently designed processing elements are also called Intellectual Property Modules, or sometimes simply intellectual property (IP). With growing market competition and increasing demand of shorter time-to-market, IPs form a major constituent of modern-day integrated circuits that are often called Systems-on-Chip (SoCs).

Although designing SoCs is providing new and exciting dimensions to the electronic systems market, at the same time, SoC design experiences significant challenges. For the past few years, the International Technology Roadmap for Semiconductors (ITRS) has been stating that the RC delays through fixed-length wire increases as the base fabrication technology scales to smaller dimensions [1]. This is attributed to the use of Cu instead of Al, dominating fringe capacitances, skin effects etc. This phenomenon sets a limit to interconnect bandwidth, which makes the upper frequency at which complex SoCs operate no longer depends on the gate speed alone. In fact frequency of complex SoCs becomes a function of interconnect bandwidth as well.

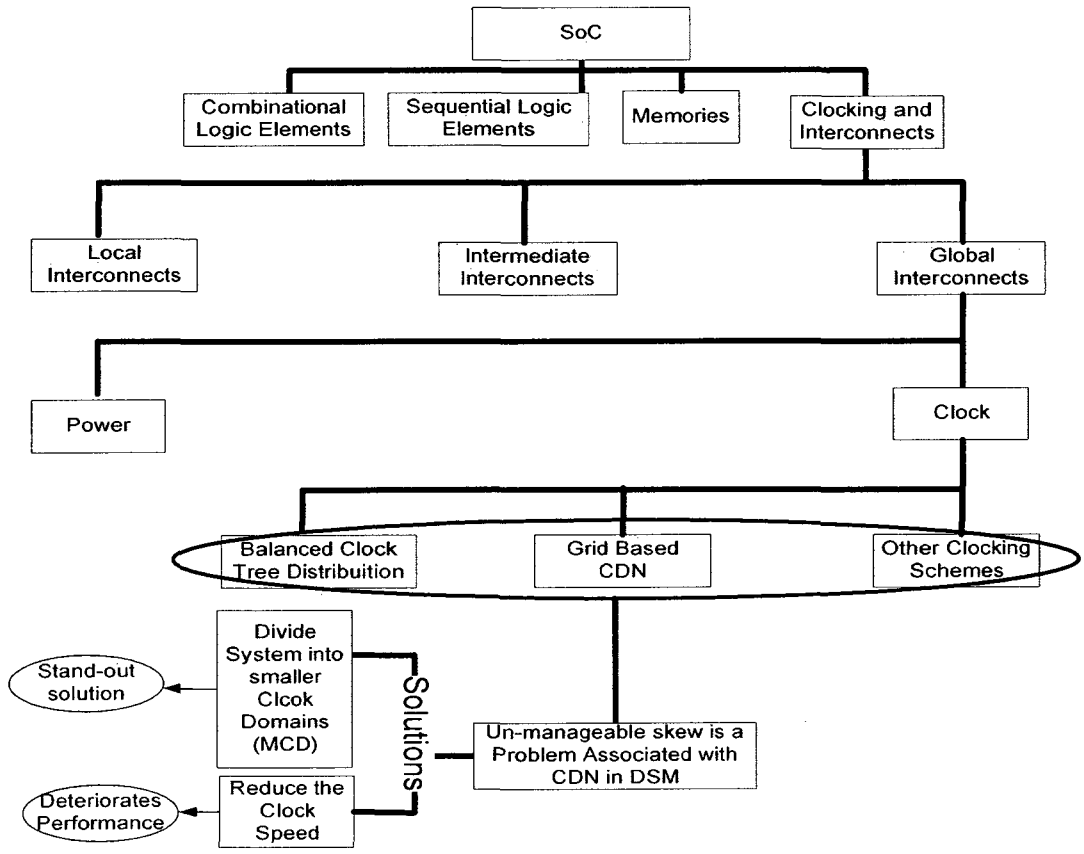


Figure 1.1. Design-related technical issues with clock distribution networks of SoC in DSM technologies

One of the major components of SoCs that requires long wires is the Clock Distribution Network (CDN) [2][3]. The trend of increment in clock speed follows the Moore's law [110], which says that frequency of the microprocessor doubles for each generation. This trend is started about 40 years ago and it is going to last at least another decade and a half [111]. Therefore, it has become an enormous challenge to design a reliable CDN while keeping pace with the demand of frequency increment. Figure 1.1 shows that the CDN requires global interconnects. In DSM technologies, interconnect bandwidth is a dominant factor to set the CDN frequency limit [4][5]. Repeaters are conventionally used to alleviate the problems associated with interconnect bandwidth [2], but repeaters introduce timing uncertainties due to process variations (which are becoming more and more significant as feature sizes are reduced), which leads to timing skew. Skew becomes unmanageable in CDN of modern DSM technologies [81] and this unmanageable timing skew may lead to intolerable phase discrepancies among communicating modules driven by the same CDN. Several de-skewing techniques are proposed in the literature [57], [58], [109]. These schemes introduce phase detection and correction schemes at different stages of clock distribution. De-skewing mechanisms have its limitation due to the additional buffers, which are needed to perform phase detection and correction. These buffers may introduce some delay that leads to extra jitter, which can overwhelm the improvements it may provide.

Large phase discrepancies due to unmanageable skew adversely affect the timing budget and, consequently, are detrimental to inter-module communication in SoCs. Figure 1.1 shows two obvious choices to overcome this problem of unmanageable skew. One solution is to reduce the clock frequency to accommodate the phase discrepancy within the

timing budget. Obviously, this solution deteriorates performance and hence it is not a desirable solution. Another and more prevailing solution, also shown in Figure 1.1, is the introduction of the concept of Multiple Clock Domains (MCDs), where different IPs may be grouped in separate clock domains. Hence, every clock domain has its own CDN, which does not need to be distributed throughout the entire chip. Although this solution alleviates the interconnect bandwidth issue for individual clock domains, in order to leverage the functionalities from different modules in the integrated circuit, IPs in different clock domains require an interfacing mechanism to interact with each other. Therefore, there is a growing need to investigate different mechanisms to interface modules in MCDs.

1.2. Design-Related Economic Issues with SoCs in DSM Technologies

Another aspect of designing an integrated circuit is its economic viability. It is shown in Figure 1.2 that time-to-market is one of the key economic design issues regarding SoCs in DSM technologies. In order to avoid long iterative design cycles (to reduce time-to-market), integrated circuit designers heavily rely on IPs and standard design practices.

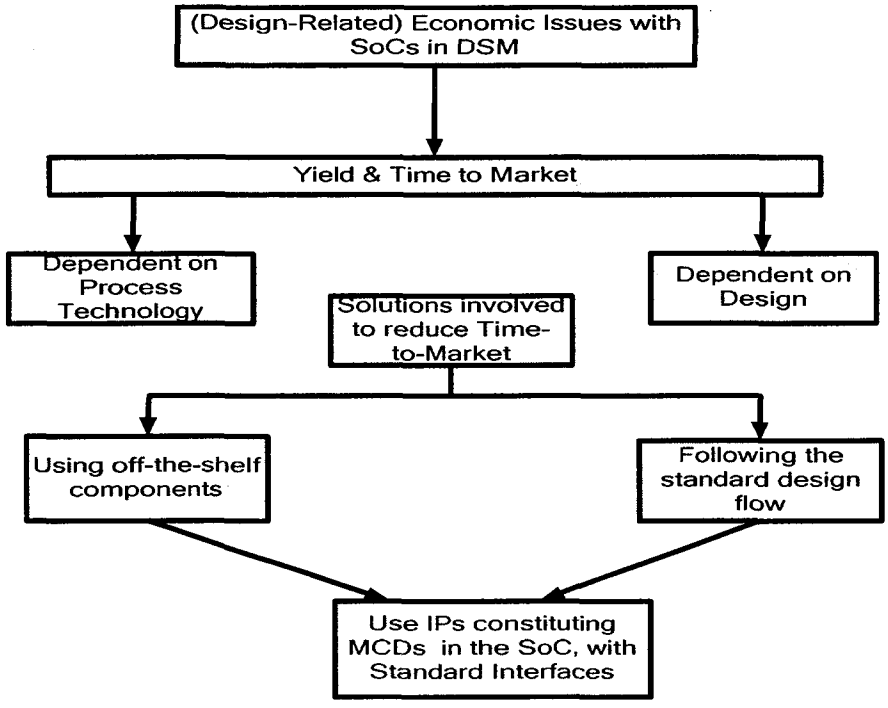


Figure 1.2. Design-based Economic issues with SoCs in DSM technologies

As discussed in the preceding section, these IPs may have different signalling and clock requirements, which usually leads to IPs running under different clock frequencies and results in MCDs in SoCs. This is pictorially represented in Figure 1.2. Summing up with the previous section, economic and technical design-related issues for SoCs in DSM technologies point toward the same solution of dividing the SoCs into MCDs.

1.3. Overview of Inter-Module Communication Techniques in MCDs

It was established in the previous section that SoCs with MCDs promise to solve most of the design-related technical and economical issues. This requirement results in the

growing need for designing interfaces that allow communications among modules in MCDs. In this section, an overview of possible inter-module communication techniques is provided. Figure 1.3 suggests that inter-module communications in SoCs with MCDs can be classified into four main classes: point-to-point communications [7], [8], [9], [10], [11], point-to-multipoint communications [12], Bus based, and Network-on-Chip (NOC) based [13], [14], [15], [16] design schemes.

In all the communication methods listed in Figure 1.3, there is always a need to physically connect a module with another module, irrespective of the nature of the inter-module communications. This requirement suggests that a point-to-point communication method is always required. Therefore, this research is directed toward proposing new and improved methods for point-to-point communications, as is indicated in Figure 1.3. Indeed, if a point-to-point communication method is broad enough to encompass all possible variations under different timing constraints, or can at least suggest a good solution under a given set of conditions, then it should lead to a better overall system architecture.

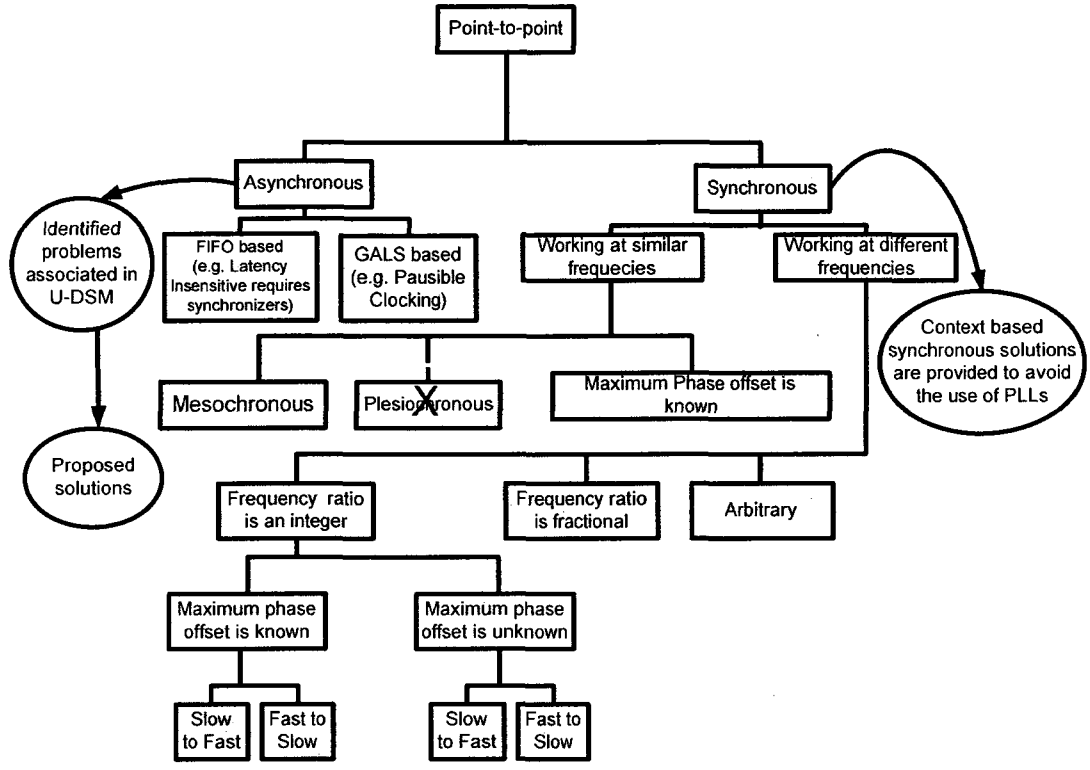


Figure 1.3. Overview of possible inter-module communication techniques in MCD

1.4. Context-Based Selection of Inter-Module Communication Method

Several different classes of point-to-point communication methods have been suggested in the literature. It can be seen in Figure 1.3 that some of the state-of-the-art solutions follow the synchronous design style for IP communication, while other designers seek solutions in asynchronous or partially synchronous and partially asynchronous (such as Globally Asynchronous Locally Synchronous (GALS) [17]) design methods. This section discusses the suitability of the available design methods under different design contexts.

Choosing among different point-to-point solutions, from synchronous and asynchronous design paradigms, is a difficult design decision. The suitability of a design scheme (synchronous, asynchronous, or GALS) depends on the context of the design and sometimes even on the natural inclination of the system designer(s). Moreover, different applications lead to different inter-module communication requirements and hence may lead to different solutions. For example, in multi-standard digital television applications, there is a requirement to produce clocks having frequencies, such as 354.6895 and 360 MHz that are related by a ratio of exactly 709379/720000. These clocks must have a very low timing jitter, typically below 200 ps [20]. Hence, such designs cannot rely on pausible clocking techniques, due to their use of jitter prone ring oscillators to generate clocks (see Table 1.1). Also, the required frequency precision and accuracy in this application imposes a tremendous effort toward designing ring oscillators with very fine resolution.

A similar example of contextual choice of communication interface is the asynchronous FIFO-based technique which falls in the asynchronous paradigm as shown in Figure 1.3. This is a typical solution to interface interacting modules in multiple clock domains in a significant number of the multi-million transistor chips [21], [22], [23]. These interfaces also require synchronizers for control signals. This solution is popular mainly because it requires less deviation from the conventional synchronous digital design flow and CAD tools. However, FIFO based techniques increase latency, because in order to transfer the control signals these interfaces requires synchronizers, which adds latency to the system. This latency issue with asynchronous FIFO-based solutions is not favourable for microprocessor interfaces that have stringent latency requirements. For

example, multi-core architectures are especially sensitive to inter-processor latency for remote cache lookups and local memory accesses [24]. Some low-latency FIFO-based designs have been suggested in the literature [25], but the use of synchronizers makes the interface non-deterministic. Therefore, for multi-gigahertz microprocessors, asynchronous FIFO based designs along with synchronizers reduce the performance. Consequently, alternative methods are required to cope with such problems.

The above examples of different design contexts show that choosing inter-module interfacing solutions in MCDs depends upon the context of the design and available expertise. A summary of state-of-the-art asynchronous/GALS solutions and synchronous solutions is shown in Table 1.1. This table shows that there can be many different combinations of criteria that may have to be looked upon before making a decision. Hence, it is virtually impossible to have one universal solution that fits all the contexts. However, it is entirely possible that a context may have more than one solution, which may even belong to different design paradigms. This can be seen in Table 1.1, e.g. in the first row for Mean Time Between Failures (MTBF); both pausible clocking (a asynchronous interfacing technique in which clock is paused for the duration of communication, explained in Chapter 2) in the asynchronous design paradigm and the solution when the maximum phase offset is known provide reliable solutions. Therefore, in such a case, it is the choice of the designer(s) to select the interface that is best for a particular system, which can be based on some other design metric.

Following the needs of context-based design solutions, this work proposes novel designs in both, synchronous and asynchronous, domains. The major objective of these novel solutions is to alleviate the design issues that restrict performance of SoCs in

modern DSM technologies. The following discussion illustrates the shortcomings of the state-of-the-art solutions to cope with the challenges of modern DSM technologies, and it provides a set of problems that are not resolved in the state-of-the-art designs.

Table 1.1. Overview of conventional Inter-Module Interfacing Solutions in MCDs

		State-of-the-Art Asynchronous/GALS Solutions		State-of-the-Art Synchronous Solutions			
				Working at same frequencies		Working at different frequencies	
		Pausible Clocking [7,8,9]	FIFO-Based [25, 26, 82]	Maximum phase offset is known [76, 77]	Mesochronous [28, 29]	Worst Case Known (Rational ratio) [34, 35, 78]	Worst Case Unknown [20, 31]
MTBF Prone		No	Yes	No	Yes	No	Yes
Worst Case Latency	Same Frequency	Undetermined (depends on the flow control signals)	FIFO Delay	Depends on phase adjustment of FIFO	Varies (usually several clock cycles)	NA	NA
	Arbitrary	Undetermined	FIFO Delay + integer raio	NA	NA	Depends upon the phase adjustment of FIFO	Several Clock cycles
Major Design Issues		Separate clocks, prone to jitter prone, non-deterministic latency, CAD issues, requires characterization at the physical level	Latency (not suitable for high performance designs)	Worst case a priori timing information is required	Training sequence required, Higher frequencies results in higher MTBF etc.	1)The maximum achievable frequency is the slowest of the two IPs (2) Same as column 4	MTBF, Frequency limitations, Low jitter tolerance
EDA Tools Availability		Limited	Limited	Matured	Matured with DFT issues	Matured	Matured but cumbersome

1.5. Problem Identification

As a precursor to our research it is observed that the recent literature addresses either the limitations of the CDN or the interfacing techniques for communication between modules in MCDs. Consequently, we realized that there is a need to establish a quantitative procedure to determine the requirement of MCD in advanced SoCs. Therefore, we address this problem first, before making a comprehensive study on the interfacing techniques.

As discussed earlier interfacing solutions for MCDs spread in both, asynchronous and synchronous, design domains. We carefully analyzed the current state-of-the-art interfacing solutions against the challenges posed by the scaling of modern DSM technologies. This careful analyses lead to identify problems in interface of both, synchronous and asynchronous, design paradigms. These problems are elaborated as follows:

Asynchronous Paradigm: In Figure 1.3 two sub-categories of asynchronous solutions are shown. Due to time constraint in this thesis, we restricted our attention to pausible-clocking based asynchronous handshake schemes, which are described in [7] – [11]. By convention, system designers tend to describe such handshake schemes at the RTL level and little emphasis is put on physical level issues. As the modern DSM technologies scales further, RTL description alone cannot encompass the inevitable physical-level non-ideality effects in the handshake schemes [4]. To deal with the challenges of modern DSM technologies, such asynchronous interfaces must be analyzed at the physical level. Among the different sources of non-idealities, we singled out crosstalk glitches, because

of the fact that, if crosstalk glitches are not treated properly, then they may lead to malfunctioning of the system. Also the current brute force repeater insertion solutions to cope with crosstalk glitches are overkill and power hungry. Understanding the gravity of this matter leads us to investigate this problem further. It is found that the current solutions are vulnerable to crosstalk glitch propagation. This is mainly because the protocol designers are unaware to such purely physical characteristics of electrical circuits. It is also noticed that state-of-the-art techniques fail to provide a framework that can identify the effects of crosstalk glitches in modern DSM technology at higher abstraction levels. A framework is necessary in order to establish awareness among the protocol designers about the physical characteristics. Furthermore, no proper methodology is defined to make such asynchronous interfaces tolerant against crosstalk glitches. These problems aggravate as the modern DSM technology scales further down, hence there is an urgent need to solve these issues. Therefore, in this thesis, we provide novel solutions to address these problems, which are concisely explained in Section 1.6, and detailed explanations are provided in Chapters 4 – 6.

Synchronous Paradigm: In Figure 1.3, synchronous design interfacing technique is further sub-divided into several categories. Here, due to time limitations, we restricted our study to interfacing technique where the maximum phase offset is known. Under this category our study addresses the communicating modules working at the same frequency or having a rational frequency ratio of integer or coprime numbers. State-of-art interfacing choices for such a case [76] – [79], show that current design solutions for MCD interfacing are not able to answer the following fundamental problems: as the current state-of-the-art design techniques only emphasise the correct functionality of the

system, hence the performance of the system are sometimes compromised. This in turn makes the maximum throughput of the communicating modules a function of clock skew. Hence, in modern DSM technologies, where worst-case clock skew between communicating modules can be very high, the current state-of-the-art techniques severely affects the performance of the system.

Furthermore, in current techniques, in order to communicate, two modules that work at different frequencies require rate multipliers [23], [35], [77] or lookup tables to skip clock edges [34]. These techniques limit the system throughput and the throughput becomes a function of the slowest module in the system. In high performance design scenarios, especially applications like SERDES or burst data communications, fast modules require to communicate with slower modules without slowing down.

Currently, the state-of-the-art synchronous interfacing methodologies are not part of the standard library of the EDA tools. This is because, to the best of our knowledge, the current techniques do not provide a comprehensive case-by-case timing constraints. For this, a mathematical formulation is required to obtain such timing relationships for all possible clock scenarios.

A relatively recent concern regarding such interfacing scenarios is in relation to the dynamic phase variations in modern state-of-the-art Multi Processor SoCs (MPSoCs), which is mainly due to varying hot spots due to the changes in thermal gradients [85] – [88]. Currently, the interfacing techniques do not address the dynamic time variations in the system hence there is a need to address this issue as well.

These problems are critical and need to be resolved before the scaling of modern DSM technology cease to extract any advantage out of the interfacing techniques. These

problems are addressed in this thesis. In the next section, a discussion about the contribution of this thesis is provided. Due to relatively diverse nature of asynchronous and synchronous design paradigms, these two paradigms are treated separately.

1.6. Contributions of this Thesis

In the preceding section w have seen the need for a quantitative study of the requirement of MCDs in modern DSM technologies. The first major contribution of this thesis is the quantitative analysis of the benefits of MCDs in terms of increasing the performance of SoCs. This study is elaborated in Chapter 3 of this thesis, where a balanced CDN is studied and a methodology is devised to alleviate the interconnect bandwidth reduction in modern DSM technologies. This work resulted in a publication "Split H-tree Design Method for High-Performance GALS Systems" in NEWCAS-2006.

Regarding interfacing requirements, this work advocates context-based design schemes. These design schemes recommend that designers should not restrict their efforts to a particular class of interfacing solutions; rather, they should examine all the possible design choices to obtain the optimum solution under a particular design context. Keeping the same concept of context-based design, this thesis provides a comprehensive study of different design styles for point-to-point communication under several design contexts. A summary of design contexts and their sate-of-the art solutions are provided in Figure 1.3 and Table 1.1, respectively. This thesis contributes in both, synchronous and asynchronous, design paradigms. Following, we summarise the contribution of this thesis

with respect to the solutions provided for interfacing methodologies in both, design paradigms:

Asynchronous Paradigm

This thesis addresses all the problems raised in the preceding section (in relation to asynchronous interfaces) and produced following pertinent solutions:

1) Effects of crosstalk glitches in modern DSM technology on the conventional asynchronous handshake schemes are identified. State-of-the-art techniques mostly address these problems from the synchronous design perspective [71], [72]. Our analysis is performed to understand the physical nature of these crosstalk glitches for event-driven asynchronous interfaces. It is emphasized that these effects are aggravating with the advancement in DSM technologies. This work led us to publish a paper entitled "Crosstalk Effects in Event-Driven Self-Timed Circuits Designed With 90nm CMOS Technology", in ISCAS 2007.
2) To this day, crosstalk glitches are treated using bus encoding techniques [68], [69], but these solutions fail to address the case when an aggressor line inflicts glitches on to a quiet neighbouring line. Such a phenomenon is called as Aggressor-to-Quiet-line Crosstalk (AQX) in this thesis. Our next contribution is to provide a framework, at the logical abstraction level, for modeling the possible behaviour of asynchronous handshake schemes, designed in modern DSM technologies, under the influence of crosstalk glitches due to AQX. These behaviours cannot otherwise be investigated without transistor-level circuit analysis. This modeling technique is intended to provide a basis for formal verification of asynchronous handshake schemes in the presence of the physical design challenges posed by modern DSM technologies. This

research led us to publish a paper "Crosstalk glitch Propagation Modeling for Asynchronous Interfaces in Globally Asynchronous Locally Synchronous Systems", to IEEE Transaction on Circuits And Systems I (TCAS -I). This paper is accepted for publication.

3) Conventional solutions to alleviate crosstalk glitches require the knowledge of detailed physical analysis [68], [69], and [74]. Utilizing our modeling approach, we proposed novel technique(s) to design robust asynchronous handshake schemes for modern DSM technologies that can cope with the crosstalk glitch propagation. This technique is named crosstalk glitch gating. Our solution can be implemented at an early stage in the design cycle. This solution may become part of the standard library, which can be used whenever potential crosstalk glitches can lead to malfunction of the system. This work is submitted to IEEE TCAS -I, in a paper entitled "Crosstalk Glitch Gating: A Solution for Designing Glitch Tolerant Asynchronous Handshake Schemes for GALS Systems". This paper in under revision.

Synchronous Paradigm

Our work also addresses the problems identified in the preceding section in relation to current synchronous interfaces. Our solutions can be applied to modules in MCDs where their clock frequency may be the same, or be integer multiple or rational ratio of coprime numbers. This solution leads to following significant contributions:

1. Through novel techniques adapted for clock scheduling an interfacing methodology is developed with two levels of interfacing registers, instead of one in the conventional source synchronous designs [76] – [79]. This technique made the communicating

modules clock frequency independent of the clock skew. In doing so, our design technique is free from clock data delay mismatches. This is made possible by utilizing worst-case *a priori* timing information to adjust clock phases. Our analysis provided solutions for both setup and hold time violations.

2. For providing solution to interface modules running at same frequency a comprehensive mathematical model is developed which makes our design readily available for EDA tools.

3. In order to provide solutions for interfacing techniques between modules having clock frequency ratio of integer number or coprime numbers, our work allows a slower module to communicate with a faster module without slowing down the faster module. These different inter-module bandwidth requirements (e.g. fast sender to slow receiver) are accommodated by managing the bus widths accordingly along with periodical clock scheduling. A complete methodology is developed and illustrated in Chapter 8^{th} to provide a solution, which includes a periodical clock scheduling for the special case of rational frequency ratio of coprime numbers.

Our work on the interfacing mechanism for modules working at the same frequency and for the modules having frequency ratio of an integer number, is submitted to Journal of VLSI integration (Elsevier) entitled "All Digital Skew Tolerant Synchronous Interfacing Methods for High-Performance Point-to-Point Communications in DSM SoCs". This work also appeared as a technical report # EPM-RT-2008-10 in the Départment de génie électrique, École Polytechnique de Montréal, Montréal, QC, Canada, December 2008 [84]. One of our proposed design, which allows communication between modules having frequency ratio of coprime numbers, was published as paper "All-digital

skew-tolerant interfacing method for systems with rational frequency ratios among Multiple Clock Domains: Leveraging *a priori* timing information", in MNRC-2008 [93]. Extension of this work to dynamically adjust the phase variations was published in ISCAS-2009 as a paper entitled "An All-Digital Skew-Adaptive Clock-scheduling Algorithm for Multiprocessor Systems on Chips (MPSoCs)" [98].

1.7. Thesis Outline

The next chapter of this thesis provides a concise review of the state-of-the-art of inter-module interfacing methodologies in SoCs. Chapter 3 discusses the problems associated with interconnect bandwidth in conventional CDNs. A specific and often used CDN, the H-tree, is studied. A technique is devised to introduce MCD by dividing the H-tree, and allowing modules in each clock domain to communicate with each other. A mathematical formulation and simulation results show that the resulting system may allow local synchronous modules to run at a faster frequency, compared to conventional globally clocked systems.

After establishing the need for MCD, an investigation is made into conventional pausible-clocking-based GALS designs in modern DSM technologies. In Chapter 4, it is recognized that the state-of-the-art pausible-clocking-based designs are vulnerable to glitch due to AQX at the asynchronous interfaces. Chapter 5 builds a framework that allows representing the possible behaviour of a logic structure in the presence of crosstalk glitch at the logic abstraction level, hence avoiding the need of detailed electrical simulation to analyze crosstalk glitch effects. In Chapter 6, crosstalk glitch gating is

proposed to quench the glitches due to AQX in such pausible-clocking-based GALS systems.

After investigating asynchronous domains for modern DSM technologies, we switch our attention to interfacing solutions in the synchronous design paradigm. Chapter 7 provides a skew tolerant solution for systems where communicating IPs have the same frequency, or frequencies related by an integer, but with phases that are not aligned, and for which the maximum phase offset is known. Chapter 8 deals with IPs having a frequency ratio of coprime numbers (i.e. rational clocking). A clock-scheduling algorithm is developed and ways to cope with dynamically varying phases are described. A complete design methodology is proposed to devise a clock-schedule that provides enough information to generate automatically a state machine for adjusting clock phases. Finally, Chapter 9 concludes this thesis, with suggested future work.

Chapter 2: State-of-the-Art Inter-module Interfacing Methodologies for SoCs

It was established, in the previous chapter, that MCDs are an inevitable design choice in SoCs designed using modern DSM technologies. Therefore, a mechanism is required to interface modules in different clock domains. In a GALS design, individual modules in a clock domain are synchronous but the two modules that communicate are mutually asynchronous. Note that the communication mechanism to interface communicating modules in a GALS design can be either synchronous or asynchronous. Initially the concept of GALS design was proposed by Chapiro in his thesis in 1984 [17]. GALS are getting popular in the design community for various reasons, such as its potential for low power design and promise to avoid safe data transaction in modern DSM technologies [49], [75], and therefore various design projects have started in this direction [18], [39]. This thesis has provided GALS-based interfacing solutions in both, synchronous and asynchronous, design domains, while addressing the challenges of modern DSM technologies, and providing guidelines of contextual suitability for each solution, as described in Table 1.1. To understand the challenges posed by DSM technologies to sustain the growth in performance and reliability in modern day SoCs, one has to critically

analyze the state-of-the-art of both design paradigms, asynchronous and synchronous. This chapter provides this overview and is divided into two separate sections, one for each paradigm. It should be noted that this thesis does not intend to prove the superiority of one design scheme over another; rather this thesis and this review help appreciate the benefits of each design paradigm and understand their limitations.

2.1 Asynchronous Interfaces

To interface two communicating modules in a GALS design both asynchronous or synchronous based design techniques may be used. This section discusses asynchronous interfaces. Broadly, asynchronous interfaces are divided into two classes: pausible clock-based design and asynchronous FIFO-based designs with synchronizers.

2.1.1 Pausible Clocking Based GALS Designs

Pausible clocking techniques stop the local clock of the synchronous module during asynchronous communications. The clock resumes only after it receives acknowledgement from the receiver, thus averts any reliability issue due to metastability that conventional synchronization schemes have (e.g. two flop synchronizers [79]). Generally in terms of handshaking mechanism, asynchronous designs are divided into bundled data protocols and delay insensitive protocols (DI), each of these mechanisms may adopt different signalling techniques, such as 4-phase, or 2-phase etc. [45]. Following is an overview of

GALS based designs in the literature and classified into bundled data designs and DI designs.

A) Bundled Data Protocol Based GALS Designs

Figure 2.1 represents the conventional bundled data protocol scheme. In such a protocol the data and control signals travels in different paths. Some of the earlier work in this domain is by Yun et al. [36]. Their work presents a bundled data scheme. It uses an asynchronous FIFO channel to communicate between each pair of locally synchronous blocks. The communication between a block and the FIFO is done using a request/acknowledge handshaking. Synchronization of handshaking signals to the local (block) clock is done with pausible clocking. The scheme requires at least two clock cycles to transfer data and at most one port per module can be active at a time because of the arbiter. Moreover, increasing fan-ins and fan-outs make their arbiter block large and impractical [8].

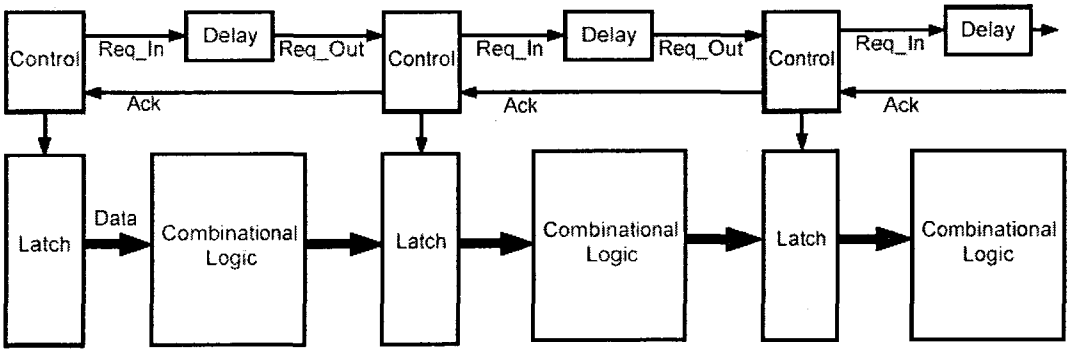


Figure 2.1. Bundled Data Protocol

Muttersbach *et al.* [8] used the similar pausible clock generation scheme to prevent metastability. The 4-phase bundled data handshake protocol has been used to signal

validity of the data. This technique resolves the disadvantage of the scheme shown in [36] where increasing fan-in and fan-out from a sub-block make its arbiter block large and impractical.

The four transitions required per handshake in [8] could result in significant performance penalty. De Clercq *et al.* [7] invented a high-speed GALS communication structure using bundled-data single-track (ST) handshaking pausible clocking. Due to the ST handshaking scheme, the latency in [7] is lower when compared to [8]. However, since these schemes rely on momentarily high impedance states on the ST lines, the implemented circuit will run correctly only if it is not exposed to heavy ambient noise.

Moore et al. [9] use a clock pausing methodology, and they devised two interfaces, one for synchronous producer to asynchronous consumer and the other for asynchronous producer to synchronous consumer. These two schemes can be joined together with asynchronous coupling (FIFO) such as GASP [37], to give a solution for communications between two synchronous domains. In [9], for larger frequency differences between consumer and producer, the effect on FIFO depth may hamper the performance severely. Moreover, the delay elements put a limit on the time span of the handshaking signals. This timing assumption is on top of clock pausing. Two more slight deviations of similar interfaces are called stretchable [40], [42] and data-driven [40], [43] techniques. These techniques are different in terms of the implementation of the clock generation mechanism. However, they fare better in terms of throughput and power consumption, respectively.

A design limitation of bundled-data protocol based designs is the interdependence in timing relations of the request signal and the data which lead to a similar kind of timing-

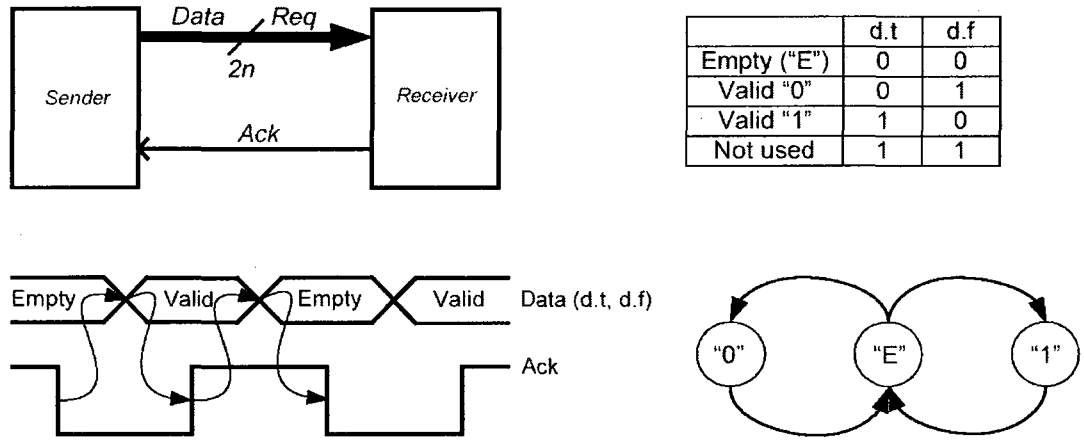
closure problem as synchronous designs face for long interconnects [44] . To cope with this issue, DI designs are introduced in the literature. An overview of such design methodologies is provided in the following paragraphs.

B) Delay Insensitive (DI) Protocol Based GALS Design

Figure 2.2 shows the block diagram of a dual rail DI design scheme. In DI designs, the request signal to send data and the data itself are encoded using a data encoding mechanism. The encoding mechanism may vary from simple dual rail protocol to complex m-to-n coding protocols. As data and control signals are encoded, therefore no delay matching is required to make sure that a request arrives after the data has reached the receiver (as is the case in designs based on bundled data protocol). As of writing of this thesis DI designs have not been seriously considered as an alternative asynchronous interface for pausible clocking based GALS designs and a handful of solutions are suggested that includes DI designs with pausible clocking [97]. In [44], a novel solution is introduced using the 1-of-4 data encoding DI interface technique introduced in [55]. Gasp [37] buffers are introduced to decouple the receiver and sender in order to establish an ST handshake mechanism. This solution utilizes the trigger signal for pausible clocking from the synchronous domain, hence making sure that reliability issues are completely eradicated. Other techniques were also introduced in the literature which do not rely on pausible clock and hence prone to reliability issues [46].

Pausible or stretchable clocking design suffers from the clock over-run problem [38]. With pausible clocking schemes, some data might get lost during the time period from when the clock is requested to be stopped to the time it is actually paused. In order to avoid this problem, different delay insertion mechanisms are introduced in [38], [10], [11].

Pausible clocking techniques, though very promising especially if the mutual frequency of the communicating modules is not known, pose some severe design challenges as well. In such methods, jitter vulnerability of the local clocks varies significantly from cycle to cycle as it restarts from a pause [41], [47]. Furthermore, design flow deviations due to the introduction of asynchronous components goes against a limitation on CAD tools, and hence are counter-productive aspects of pausible clocking interface mechanisms.



	d.t	d.f
Empty ("E")	0	0
Valid "0"	0	1
Valid "1"	1	0
Not used	1	1

Figure 2.2. Dual Rail Delay Insensitive Scheme: Block Diagram, Truth table, four phase waveform and state machine depiction

Conventional Asynchronous Circuits are Vulnerable to Malfunction in modern DSM Technologies due to Crosstalk Glitches: Conventional asynchronous circuits are designed to avoid data hazards and race conditions [65], [66], but design methods fail to address non-idealities introduced by crosstalk glitches in modern DSM technologies [4]. In [68], [69], [91], and [92] the crosstalk effects in circuits implemented in DSM technologies were mitigated with the introduction of encoding in the data bus. Though such encoding techniques are indeed very advantageous in alleviating the delays

introduced due to crosstalk, unfortunately in modern DSM technologies, the data bus is not the only crosstalk-affected region. Crosstalk glitches influence control signals, such as the handshake signals in asynchronous interfacing circuits, and may induce even more serious consequences if not treated properly. Here it is worth mentioning that the crosstalk glitches we are mentioning here is aggressor to quiet line crosstalk (AQX) glitches.

In [70], a behavioural level crosstalk detection methodology for sequential asynchronous circuits is developed. This methodology allows crosstalk detection in an early phase of the design cycle and it provides recommendations for transition reshuffling. This design method checks for intrinsic transition faults in a sandwiched wire, due to crosstalk glitches, only if its two adjacent signals are performing a transition toward the same polarity, i.e. 00 to 11 or 11 to 00. On the other hand, in modern DSM technologies, as stated earlier, considerable crosstalk glitches appear in quiet lines due to transitions in aggressor lines (AQX). These crosstalk glitches may propagate undesirably and lead to eventual system failure.

There are other types of glitches as well, which appear in the combinational logic due to mismatch in the delay paths. These glitches have settled logic values and research was done to learn how to avoid them for power reduction [71], [72]. Such glitches are studied in the synchronous design context. A robust asynchronous circuit is always designed to avoid such data hazards and race conditions [65], [66]. Thus, the context of crosstalk glitch propagation due to AQX is different from such settle-time glitches and henceforth drew our attention. Our study of crosstalk glitches due to AQX in asynchronous interfaces is reported in Chapter 4 to 6 of this thesis.

2.1.2 Asynchronous FIFO Based Methodologies

As shown in Figure 1.3, another asynchronous interfacing design possibility is the use of asynchronous FIFO based designs, without pausing the clock. Some of the earlier work in asynchronous FIFO based design is reported in [56] and [37] but these designs were proposed initially only for purely self-timed event-driven asynchronous systems.

Designs with Synchronizers: Another design style called latency-insensitive is proposed in the literature as an alternative asynchronous FIFO schemes [25], [26], [48], [82]. Chelcea et al. [26] propose a new design method for synchronous systems that makes the design functionality insensitive to long wire latency. They optimized the famous approach of pipeline synchronization, presented by Seizovic [50], by noting that synchronizations are only needed for the receiver when the buffer is almost empty and only needed for the sender when the buffer is almost full. In order for this system to work properly, it needs to be "patient", where a patient system is a synchronous system whose functionality only depends on the order of the events of each signals and not on their exact timing [51].

Another asynchronous FIFO approach in the literature is proposed by Chakraborty et al. [23]. This work proposes a novel solution using the STARI FIFO [83] technique to address arbitrary clock frequency domains. In this scheme, transmitter and receiver forward their clocks to each other. This helps in estimating the clock frequency. Whichever has the higher frequency uses a rate multiplier to create an approximation of other one. This solution also tolerates a phase discrepancy of up to two clock cycles, when the synchronous domains are working at the same frequency.

Asynchronous FIFO solutions are good solutions for inter-module communications especially when the mutual frequency between the modules is arbitrary and flow control information is limited. The latency of the synchronizer is a severe drawback in implementing such designs in high performance systems. A rule of thumb is to keep 40 gate delays to resolve the metastability, which for 130nm technology translates into 2.5 ns [41].

2.2 Synchronous Interfaces

Figure 1.3 shows that the synchronous design domain is further sub-divided into three sub-categories: mesochronous, plesiochronous, and the case when maximum phase offset is known. Plesiochronous schemes are usually found in telecommunication systems, where the two interacting modules typically have independent clocks generated from separate crystal oscillators. Such a scenario is not likely to be present in on-chip or even in on-board systems [27], and is not further considered in this thesis.

Conventionally, both the mesochronous synchronization and the category of known maximum phase offset are solved by the same set of solutions. Mesochronous synchronizing techniques such as the delay line synchronizer [52], the Self-Test Self-Synchronizing (STSS) method [28], or the Globally Updated Mesochronous (GUM) method [29], utilize synchronization schemes with a stringent timing constraint for resolving the metastability within half a clock cycle. This constraint restricts the maximum speed of an interface.

2.2.1 Known Maximum Phase Offset

Same Frequency: Recently in the third category, shown as "maximum phase offset is known" in Figure 1.3, a new solution was proposed which utilizes *a priori* timing information of the maximum phase offset. This work is described by Caputa et al. [76] and Edman et al. [77] [78]. A generalized cartoon depiction for such a case is shown in Figure 2.3. Here, it is depicted that clock is provided using the same source to two different modules and because of the potential difference in delay in the clock lines (shown as t_A and t_B in Figure 2.3), an interfacing technique is required to allow the two modules to perform safe data transaction. They devised a FIFO based methodology with *a priori* knowledge of the maximum phase offset. In their methodology, the clock signal from the sender, called strobe signal, is sent to the receiver along with the data. Data is latched, when it arrives, using the strobe signal. The receiver clock that latches the data is sourced from a particular clock and is delayed enough so that it can tolerate the maximum skew in the system.

Such a design requires stringent matching delay properties between the data and clock signals, as data has to arrive before the setup time of the strobe signal edge [80]. However, this design methodology tolerates a fairly large skew but the maximum attainable frequency for the terminating modules is a function of the clock skew [80]. Our solution, provided in Chapter 7 of this thesis, addresses these issues and our solution rectifies the mentioned shortcomings.

Different Frequencies: Another branch of synchronous methods, shown in Figure 1.3, consists of solutions for the case when the interacting modules are working at different

frequencies. This category is divided into sub-categories on the basis of the frequency ratio between the communicating modules, which may be an integer, fractional, or arbitrary rational number. Various techniques are proposed in the literature for interfacing modules working at rational frequencies [24], [34], [35]. In [34], Sarmenta et al. utilize the knowledge of the timing information to setup a lookup table in order to pass selected pulses to the communicating modules. These modules are supposed to have a rational frequency ratio. Due to the centralized nature of this design, modern DSM technologies pose a serious limitation to such designs as phase discrepancy grows with the wire length. In [35], Mekie et al. propose a solution that utilizes prior knowledge of the data flow and timing. By exploiting this knowledge, the design signals the receiver IP at instances that fall on the prohibitive window of the registers, and skips communication on those clock edges. One of the limitations of such designs is the size of the shift register required to store the desirable sequence of clocks, and hence this solution has its hardware limitations even for some commonly used rational frequencies.

To follow the conventional design flow, in [78], Edmand et al. extended their technique, for communicating between modules working at the same frequency, [76], [77], to modules working at rational frequencies. This work utilizes same rate multiplier as used in [23] to choose the right phase to send the data. This design technique can accommodate large clock skews, but the maximum achievable clock period of the IP modules is still dependent upon this clock skew. This technique also suffers from the control and data delay mismatch problem and also the classic problem of slowing down the faster module that limits the maximum possible throughput obtained to a function of the frequency of the slower module.

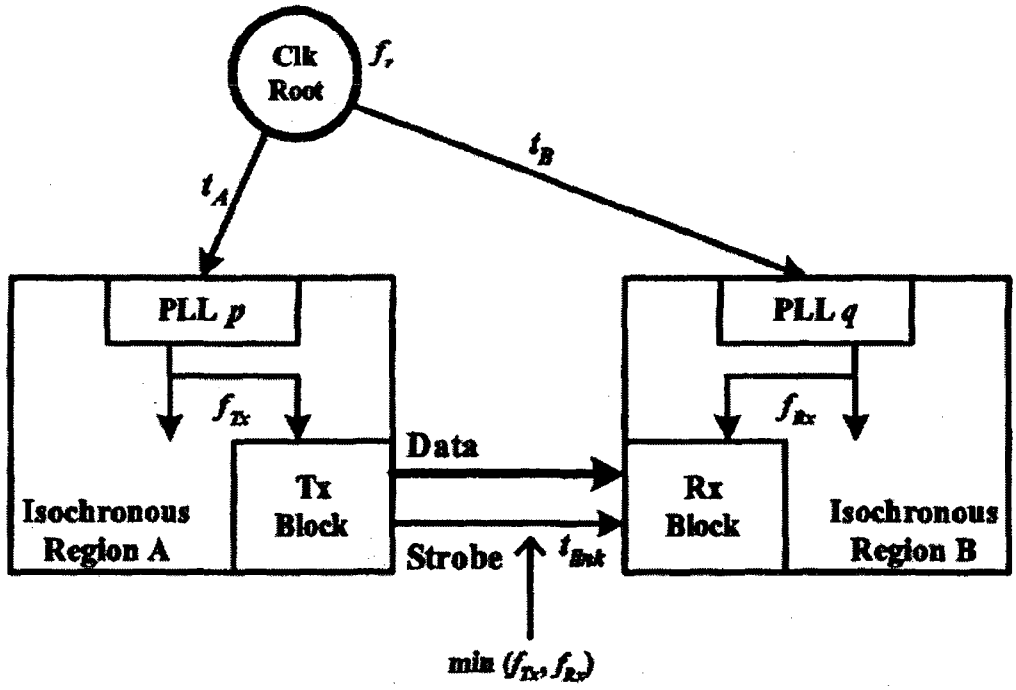


Figure 2.3. Two isochronous regions with communication links [78]

Along with the drawbacks mentioned, the above solutions (discussed in this section) have one common limitation: their frequency of data transfer cannot exceed the frequency of the slowest module in the interface. Solutions provided in Chapter 8 addresses the problems mentioned here.

Dynamic Phase Variations: As we go deeper into modern DSM technologies there is also a growing concern of dynamic phase variations due to varying thermal gradients. In particular, high-speed applications are more affected due to the non-uniform delays generated across chips by dynamic thermal variations. Thermal variations are observed at run time, since they are workload dependent [98]. It has been projected that, in high-performance ICs, the peak chip temperature could raise up to 160°C for the 90nm technology node, that surpasses the maximum acceptable junction temperature of 125°C

after which reliability issues arise [85]. In [86], it was shown that for every 20°C increase in temperature, the Elmore delay for the long global interconnects increases by approximately 5%-6%. Assuming a 25 °C nominal temperature, this translates into 30%-35% delay variation between the nominal and peak temperatures in integrated circuit implemented with 90-nm CMOS. In [87], it was shown that the inverter optimal size that minimizes delay per unit length in a repeater structure is dependent on the ratio of the transistor resistance (R_{on}) to the wire resistance. Thus, the temperature dependence of that ratio has a significant impact on the repeater insertion methodology. In [87], it was also shown that a 75°C increase in temperature from the nominal value results in 34% and 45% change of value of that resistance ratio for the 180nm and 65nm technologies, respectively. Such a large deviation in the resistance values, and therefore in delays, will translate into clock skew and could cause timing violations.

In order to mitigate the effects of on-chip temperature variations on delay, several solutions have been implemented. In a single core chip, a variability-aware micro-architecture partitions a processor into multiple independent voltage-frequency islands [88]. Another micro-architecture, may partition the chip into different clock domains with each clock being derived from a common source [83]. These frequencies are exact rational multiples of each other, and the clock frequency ratios are known *a priori*. Such a micro-architecture is common in System-on-Chip (SoC) designs having IP blocks running at their own frequencies. The IP blocks are usually optimized and often comprise various ASIPs (Application Specific Instruction set Processors) that are part of heterogeneous multiprocessor Systems-on-Chip (MPSoCs). This section focuses on this type of micro-architecture.

In multiprocessor architectures, stop and go [96] policies are the most commonly used to reduce peak temperature. In such policies, in order to preserve the state of the core, it needs to be saved in memory before powering down. When the core is "awaken" again, significant current draw is required from the power supply, whose higher capacity output creates more heat. In addition, the load on the power supply is larger at start-up, with a consequence of larger temporal temperature variations. From the above, it is obvious that clock skew variations will be introduced by these policies. Hence, a technique to accommodate the run time delay variations due to thermal gradients is required. This technique is illustrated in the second half of Chapter 8

2.2.2 DDS based Solutions for Rational Frequencies

Another family of solution is dedicated for the rational frequencies of large co-prime numbers. Direct Digital Synthesis (DDS) [31] is a popular synchronizing method for communicating among the modules having such fractional frequency ratio. But, due to its dependency on analog components, DDS has its limitations with the maximum attainable frequency. Furthermore, DDS is unable to generate rational number frequencies if the denominator cannot be expressed by 2^n, where n is an integer [53].

Calbaza et al. proposed a solution, called DDPS (Direct Digital Period Synthesis) [20], for communicating at fractional frequency ratio by using only digital logic, hence that work promises to achieve a higher frequency. Boyer et al. used a similar concept to produce variable speed processors (VSP) [33].

Even though DDPS and VSP provide flexibility in clock multiplication, their control signals require a separate frequency phase detector and low pass filter [54].

2.3 Discussion

It has been indicated in the first chapter that this thesis is written based on the notion that there is no one universal solution for all the design contexts. Rather, based on a particular design context, the designer has to choose among the given alternatives keeping in view the design constraints of each design in hand. Shortcomings of each method are provided at the end of the discussion of each class of state-of-the-art solutions. Some of these shortcomings are addressed in the proposed solutions stated in this thesis. Chapters 4 to 6 provide solution to the challenges associated with the reliability of asynchronous interfaces designed for modern DSM technologies. Similarly, Chapters 7 and 8 provide answers to the challenges pointed out as shortcomings in this literature review for the synchronous interfacing method.

Chapter 3: Problems Associated with Interconnect Bandwidth in conventional CDN

As discussed in the preceding chapters, in modern DSM technologies, interconnect bandwidth is a dominant limiting factor. Repeaters are commonly used to alleviate the problems associated with interconnect bandwidth but this solution introduces timing non-idealities due to process variations (which get worse as technology is scaled down), which leads to timing skew. The skew associated with repeater-insertion methodologies limits the maximum frequency in Clock Distribution Networks (CDN).

In order to address problems linked to skew, various techniques have been suggested in the literature. Recent approaches use a phase detector or an additional mesh in the H-tree, to reduce skew [57], [58]. However, these approaches have limitations and they require additional routing and hardware, in addition to not addressing the problem of interconnect bandwidth specifically. They rather focus on having the least phase difference among the different clock sinks.

In this chapter, a method is presented that mitigates both interconnect bandwidth and clock skew problems. It follows a divide-and-conquer strategy. The proposed solution divides the H-tree CDN into smaller blocks and supports communications among different

IPs within the sub-blocks through self-timed or asynchronous circuits. This design methodology relaxes the timing constraints by a substantial factor, up to 3 times. A closed-form mathematical model for the length of interconnects, after successive splitting, is formulated. Simulation results support our analytical finding that split H-trees allow clocking chips, of a particular die size, at a higher frequency. Moreover, a comparison of different asynchronous communication mechanisms is achieved to suggest an optimum design based on the target system performance. It is observed that adapting some sort of interfacing technique for these split regions promises higher performance in modern DSM technologies.

3.1 Clock Frequency Limitations in Clock Distribution Network (CDN)

This section introduces some important concepts and states the basic equations that are essential to understand the rest of this chapter. In [5], Zarkesh-Ha quantitatively analyzed the factors responsible for limiting clock frequency. As a first order approximation, due to the difficulty of modeling and simulating lossy and non-uniform transmission lines, it is assumed that ground return path wiring has been implemented on the two metal levels above and below the clock wire to reduce inductance effects. Therefore, in this analysis, inductance effects are ignored.

Quantitatively speaking, maximum interconnect bandwidth can be defined by assuming that an interconnect is a low-pass RC filter such that its maximum 3dB frequency ($f_{3dB(MAX)}$) is represented by (3.1):

$$f_{3\,dB\,(MAX)} = \frac{1}{2\pi (r_{int}\, c_{int}) l^2} \qquad (3.1)$$

where r_{int} and c_{int} are interconnect resistance and capacitance per unit length respectively, and l is the length of interconnect from clock source to the middle of the sub-block, shown in Figure 3.1. Let us analyze quantitatively the effects of process variations in clock distribution networks (CDN). Considering a balanced H-tree with one source repeater and driving repeaters at each sub-block as shown in Figure 3.1, the delay from the H-tree source to an H-tree sub-block can be obtained as follows [5]:

$$T_{Delay} = 0.4\left(\frac{\rho \varepsilon_r}{H_{int} T_{ILD}}\right) l^2 + \frac{\sqrt{\varepsilon_r}}{c_o} l + 0.7 R_{tr} C_L \qquad (3.2)$$

where ρ, H_{int} and T_{ILD} are the line resistivity, thickness and inter-level dielectric (ILD) thickness between interconnects and ε_r is the relative dielectric constant of the ILD material, c_o is the speed of light in free space, R_{tr} is the driver resistance, and C_L is the total wiring and input capacitance of the destination register within the sub-block. Ignoring process variations within the sub-blocks, the clock skew, defined as the time difference between the maximum and minimum source-to-sink delay between CLK1 and CLK2 (T_{CSK}), see Figure 3.1, can be written as follows [4] (for further details, refer to [5]):

$$T_{CSK} = 0.4(r_{int} c_{int}) l^2 + \frac{\sqrt{\varepsilon_{r,ox}}}{c_o} l + \sum \left| \frac{\partial T_{Delay}}{\partial x_i} \right| \Delta x_i \qquad (3.3)$$

where $\{x_i\}$ is the set of parameters retained by Zarkesh Ha et al. in [5]. These parameters include variations in oxide thickness, threshold voltage, interlayer dielectric thickness, wire thickness, channel length, IR drop, non-uniform register distribution and temperature gradient. Figure 3.1 also shows two nearest neighbours that are subject to a maximum skew according to a summation model, since their common ancestor is farthest [59]. This is further explained in Section 3.2.

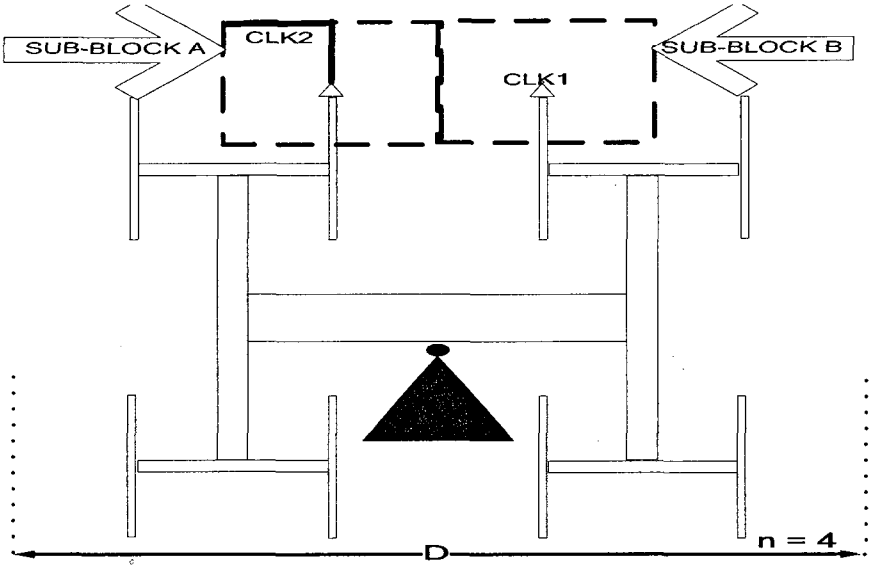


Figure 3.1. Balanced 4-Level H-Tree

Equation 3.3 has two components: the first two terms of the right side represent the wire delay in the sub-block, while the third term indicates the overall variation in delay from the clock source to the beginning of the sub-block identified as CLK1. Assuming a budget of 10% (as currently it is a normal practice to keep a budget of 10% for clock skew in digital systems [5], Chapter 12 of [62], clock frequency, f_k, is defined as follows:

$$f_{k(MAX)} = 0.1/T_{CSK} \quad (3.4)$$

Interconnect bandwidth and process variations affect the clock distribution network concurrently. Therefore, clock frequency is determined by the factor that turns out to be dominating, following the following criterion:

$$f_c = \min\left(f_{3db(MAX)}, f_{k(MAX)}\right) \quad (3.5)$$

3.2 H-tree Splitting

In order to take into account the observations made in the previous section, a technique is proposed to split the H-tree so that the benefits of shorter interconnect length can be exploited. H-tree splitting is the division of a conventional H-tree into two or more sections where each section has the same number of sinks as the others and has its own clock source repeater. In order to get the most out of H-tree splitting, identifying key splitting locations in the H-tree is required. As far as skew is concerned, H-tree splitting where communicating node pairs have the worst expected skew within a H-tree, is expected to lead to maximum skew reduction. To obtain the best splitting locations, a summation model, introduced by Fisher and Kung [59], is used. According to this fundamental architectural model (that is independent of process technology), the skew between two clock sinks of an H-tree is directly proportional to the sum of lengths leading to their common ancestor. This captures the effects of worst-case parametric variations. As far as interconnect bandwidth is concerned, the summation model implies that the skew between two nodes is worst when their common ancestor is the tree root. Indeed, the branches directly connected to the root are the longest in the tree. Therefore, it is recommended that designers split the H-tree at its root to avoid worst-case skew. The

same reasoning applies recursively. Thus, the second and subsequent considered splits break the sub-trees at their last common ancestor. In Section III of [60], Nekili *et al.* established the same fact. They showed that, in order for two clock sinks to produce the maximum skew their paths should diverge at the tree root.

The following analysis allows finding the effective length of H-tree paths during the splitting process. Assuming n is the number of H-tree levels and s_p is the numbers of splits, there are four possibilities for H-tree splitting depending on whether n and s_p are odd or even. Now, let us consider the case where the number of H-tree levels and numbers of splits are even (let's call this: case A). It is known from [3] that the length of the path from H-tree source to the sub-block input is as follows,

$$l = D\left[1 - \left(\frac{1}{2}\right)^{\frac{n}{2}}\right] \qquad (3.6)$$

where D is the dimension of the die as shown in Figure 3.1, and n is the number of H-tree levels before splitting. In case A, for every 2 splits, two branches of the same lengths are removed from the initial path of H-tree source to the sub-block input. The length of the resulting H-tree source to sub-block input is described mathematically as follows:

$$l = D\left[1-\left(\frac{1}{2}\right)^{\frac{n}{2}}\right] - D*2\left[\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^4}+\frac{1}{2^5}+.....+\frac{1}{2^2*2^{\frac{s_p}{2}-1}}\right]$$

where s_p is the number of splits. Further simplification of the above equation leads to (3.7):

$$l = D\left[1-\left(\frac{1}{2}\right)^{\frac{n}{2}}\right]-\frac{D}{2}\left[\frac{1}{2^0}+\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^3}+\ldots\ldots+\frac{1}{2^{\frac{sp}{2}-1}}\right]$$

$$l = D\left[1-\left(\frac{1}{2}\right)^{\frac{n}{2}}\right]-\frac{D}{2}\sum_{i=0}^{i=\frac{sp}{2}-1}\frac{1}{2^i} \qquad (3.7)$$

Similar analysis for other cases gives the following respective closed-form length equations: Equation (3.8) is the case when n even and s_p odd, while (3.9) corresponds to the case when both n and s_p are odd, and (3.10) models the situation when n is odd and s_p is even.

$$\begin{cases} l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\frac{D}{2^2} & s_p = 1 \\ l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\left[\sum_{i=0}^{\frac{s_p-1}{2}}\frac{D}{2^i}+\frac{D}{2^{\left[\frac{s_p-1}{2}+2\right]}}\right] & s_p > 1 \text{ and si odd} \end{cases} \qquad (3.8)$$

$$\begin{cases} l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\frac{D}{2^2} & s_p = 1 \\ l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\frac{D}{2^2}-\sum_{i=2}^{\frac{s_p+1}{2}}\frac{D}{2^i} & s_p > 2 \text{ and is odd} \end{cases} \qquad (3.9)$$

$$\begin{cases} l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\frac{D}{2^2}-\frac{D}{2^3} & s_p = 2 \\ l = D\left[1-\left(\frac{1}{2}\right)^{n/2}\right]-\frac{D}{2^2}-\sum_{i=2}^{\frac{s_p}{2}}\frac{D}{2^i}-\frac{D}{2^{\frac{s_p}{2}+2.}} & s_p > 2 \text{ and is even} \end{cases} \qquad (3.10)$$

3.3 Simulation setup

For analyzing the effect of splitting the H-tree on the clock frequency, and validating the concept of H-tree splitting, simulations are performed with Matlab. Design parameters

for the 0.18 micrometer (TSMC) CMOS technology were used and equations (3.1), (3.3) and (3.4) were implemented in order to calculate frequency. Furthermore, we substituted equations (3.7) to (3.10) into equation (3.3). In order to perform the sensitivity analysis, the parameters proposed in [5] were used to allow comparing results (for details on sensitivity equations please refer to [5]). We assumed that die size ranges from 1 cm to 4 cm (in steps of 1 cm), with C_L= 6.25pF, $r_{int}c_{int}$=115ps/cm^2, V_{DD}=1.8V, V_T=0.32V, R_{tr} = 12Ω, L_{eff} = 0.18 μm. Moreover, we assumed that, before splitting, the number of levels in the H-tree, n, was 8.

Based on the model formulated above, the maximum frequencies according to process variations and to the standard 3dB constraints are plotted in Figure 3.2. These results show that interconnect bandwidth dominates the clock frequency when the number of splits considered is small. Process variations start dominating the clock frequency when the number of splits increases (3 or more). An observation that confirms the validity of the proposed splitting locations is the increase in 3dB frequency with every considered split (having inverse square relationship with reduction in length as given in equation 3.1). The splitting process keeps producing large improvements in maximum 3dB frequency at each considered split. For illustration, consider the second dotted curve from the top (data points are shown in asterisks in Figure 3.2). This curve indicates the variation in frequency with respect to the number of splits for a die size of 2 cm due to process variations. For the same die size, the interconnect bandwidth curve is shown as the second solid curve from the top. The above-mentioned two curves and equation (3.5) imply that, for the first two splits, the frequency is dominated by interconnect bandwidth, and further splitting will make it dominated by process variations.

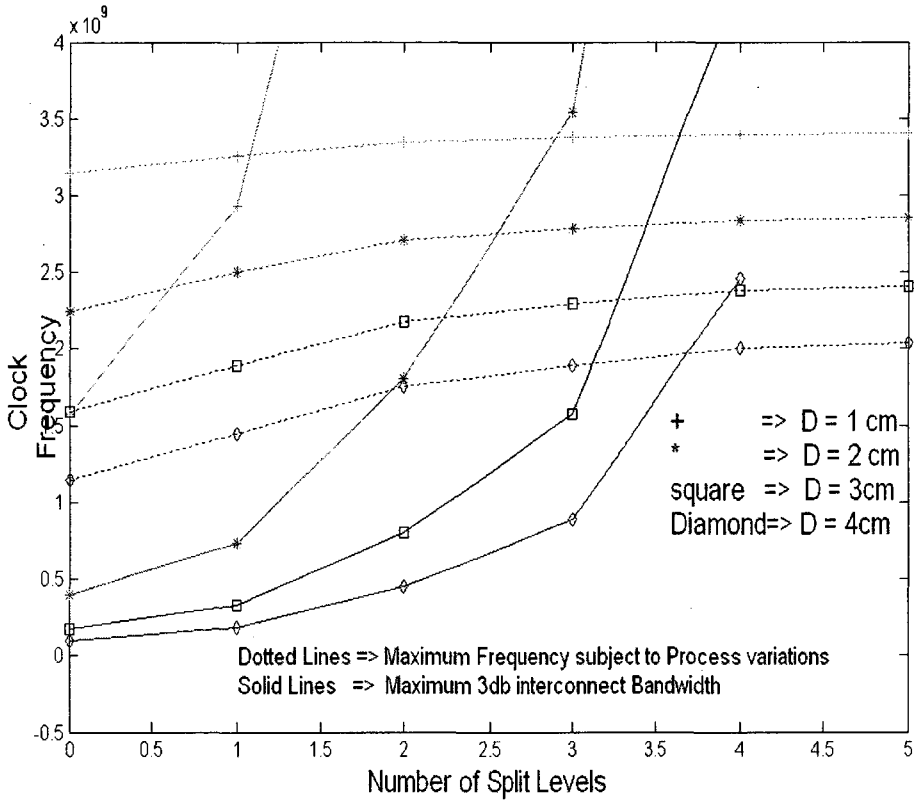


Figure 3.2. Maximum Frequency subject to process variation and Maximum 3 db Frequency vs. Number of Splits

As an example, in Figure 3.2, for a non-split H-tree, s_p=0 and die size of 2 cm, the maximum achievable frequency is 393.6 MHz. The first split results in an increase in clock frequency that is a factor of almost 1.85 (732 MHz). Similarly, the second split gives a 2.45 times improvement in frequency as compared to s_p = 1. Although the length of the first and second H-tree levels are the same for s_p=0, in this case, their widths are different to minimize reflections. This explains why the second and first splits do not lead to the same frequency increase. From the third split, the dominant factor switches to process variations and it leads to a 1.54 times increase in frequency compared to s_p=2 but, from the fourth split and onwards, the frequency increment reaches a quasi-saturation, where

the improvements are small due to smaller interconnection length reductions. This analysis takes us to the conclusion that if an H-tree clock distribution network is split, it can then work at a higher frequency. However, in order to make this solution practical, it requires additional circuit mechanisms that can allow communication among the different sub-H-trees created by this splitting; this is addressed in the next section.

A possible concern with the proposed H-tree splitting scheme comes from the fact that clock generation circuits are often driven by PLLs and that splitting could mean inserting a PLL at the root of each branch. This is illustrated in figure 3.3 where branches of a split H-tree are driven be separate PLLs. This is costly in area, power and design complexity. In this thesis, what we have in mind for a practical way to implement the split H-tree as shown in Figure 3.4, which shows that the root wire of the H-tree can be replaced by minimum size inverters to feed the two split halves of the H-tree with a single PLL at the root. The proposed overall clocking scheme is elaborated in Figure 3.4b. This method may introduce extra clock skew, but this can be accommodated in the interfacing methodology proposed in the subsequent chapters.

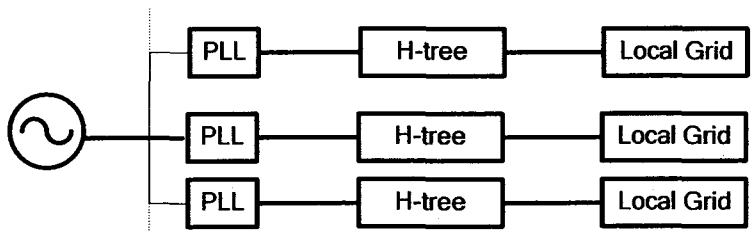
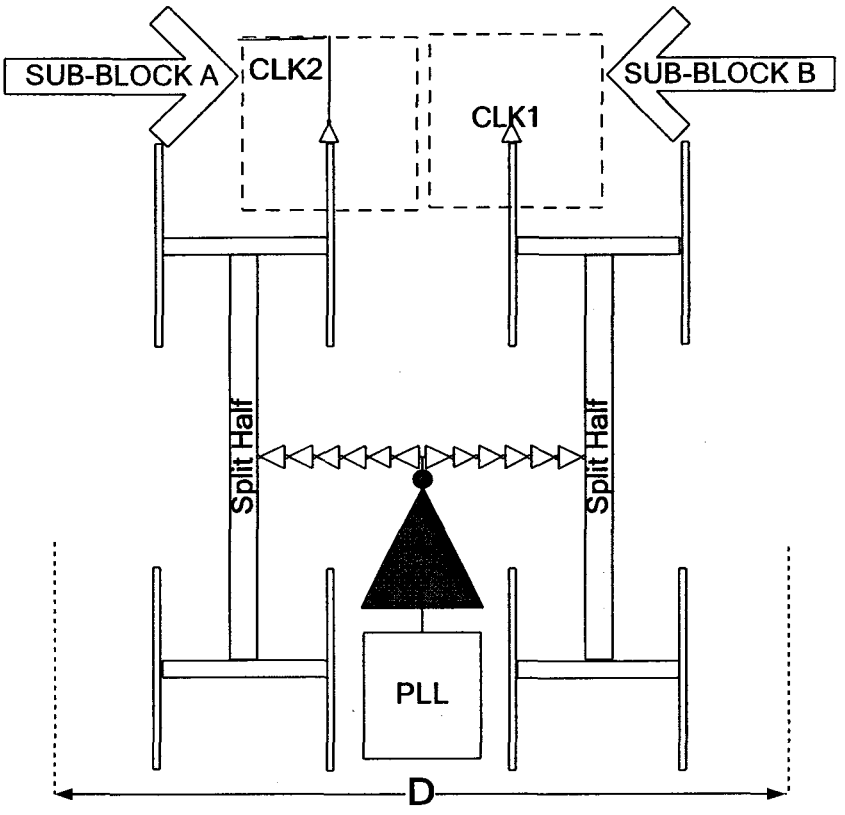
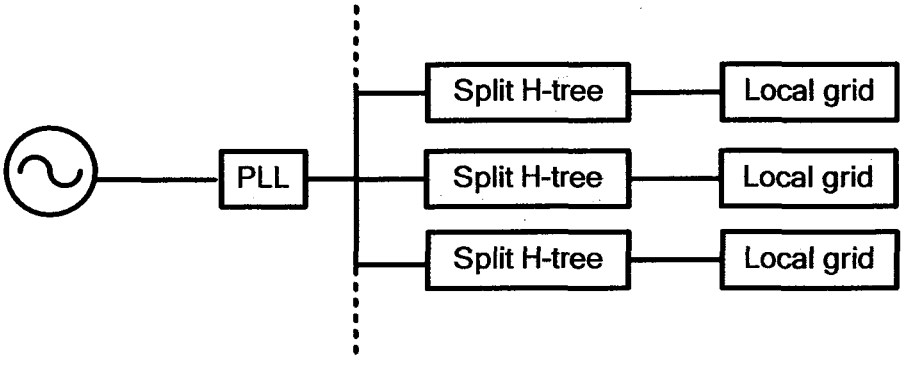


Figure 3.3. Clocking scheme for SoCs with split H-tree and potential requirement of PLLs at every split node



(a)



(b)

Figure 3.4. a) Modified H-tree after one split (b) Proposed clocking scheme with split H-tree for SoCs with the requirement of only one PLL

3.4 Communication Mechanism

The proposed solution (that involves H-tree splitting) comes with a price. H-tree splitting (a virtual process) leads to two H-tree halves with $n^2/2$ nodes in each half. Suppose the H-tree was working at a clock frequency of F_P before the splitting, and then works at a frequency of F_P+A after the splitting. Only a communication mechanism that can safely transfer data within $1/(F_P+A)$ (in time units) will make splitting a beneficial solution. If $f_{w(MAX)}$ is the reciprocal of the maximum delay of the asynchronous wrapper, then equation (3.5) can be rewritten as,

$$f_c = \min\left(f_{3db(MAX)}, f_{k(MAX)}, f_{w(MAX)}\right) \qquad (3.11)$$

Several different approaches have been proposed in the literature to support communication between two mutually asynchronous domains. Generally speaking, these sorts of asynchronous interfaces are divided into two classes, the first class uses an interface with pausible clocking and the second class uses some self-timed FIFO-based arrangement, with reliability issues (as discussed in Chapter 2). In order to choose a suitable interface type, first, the designer has to identify the clock source for each sub-tree.

There are generally two different ways of supplying clock to different synchronous blocks in GALS systems. Clock is either provided by a single (slow) global clock that supplies different (multiple) clock domains and each domain contains a clock multiplier to make the clock run at its desired frequency. Otherwise, each clock domain has its own clock generator. In the former case, a self-timed interface is an obvious choice, as these systems are analogous to mesochronous designs where self-timed design was proven successful [23]. For the latter case, a self-timed wrapper will need an extra level of effort

to make it work safely while abiding all the timing constraints. Therefore, in this case a wise choice is pausible clocking.

If a self-timed interfacing wrapper is introduced (the corresponding modification in the H-tree is shown in Figure 3.3), then the timing constraints of that wrapper will define the limitations of the entire system. For example, let us consider the recently developed self-timed wrappers by Chakraborty et al. [23]. With their wrapper, the timing limitations of the system will be $P > 2\ \eta$, where η is the time from triggering the latch controller to subsequently returning the self-timed circuit to its initial condition, and P is the clock period [23]. H-tree splitting puts an additional limit to this wrapper, and, in order for the design to work properly, it has to respect the condition $\eta < 1\ /2(F_p+A)$. Circuit simulation of the wrapper proposed in [23], using the 0.18 micrometer TSMC technology, shows that this interface can run up to a point-to-point delay of approximately 340 pico-seconds (see Table 3.1). Thus, it could run at approximately 1.25 GHz, compared to 393 MHz for the initial non-split H-tree that did not require asynchronous communication mechanisms (a 3-fold improvement). It should be noted that the maximum frequency is limited by process variations rather than the actual latency of the wrapper as illustrated in equation 3.11. The proposed splitting method thus trades extra hardware for high performance.

Table 3.1. Comparison of Designs

Die Size= 2cm and n = 8			
Asyn. Int	***# of advantageous splits***	***Interfaces (number)***	***Freq Hz.***
Self-timed	*3*	*8*	*1.25 G*
Pausible	*2*	*4*	*1.1 G*
w/o split	-	-	*393.6M*

Now, let us analyze the case when pausible clocking is used. For illustrative purposes, we adopted the design by De Clercq et al., which is one of the efficient pausible clocking wrappers reported [7] (the corresponding modification in the H-tree is shown in Figure 3.4). This wrapper is known as "single-track adaptor". This design reportedly works with a latency of 909 ps using the 180 -nm TSMC CMOS technology.

The above two interfaces allow designers to split the H-tree up to two levels or more. Table 3.1 summarizes our results. One can see the significant advantage of self-timed wrappers. The choice of wrappers is context based, and depends on several factors. A guide line can be obtained by using Table 1.1.

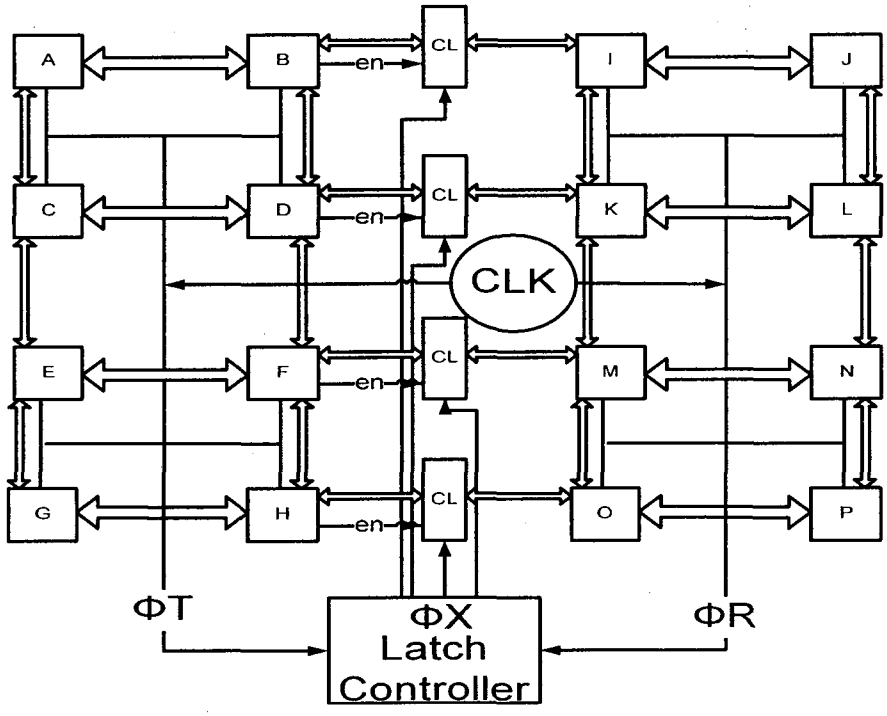


Figure 3.5. H-Tree with Self-timed circuit

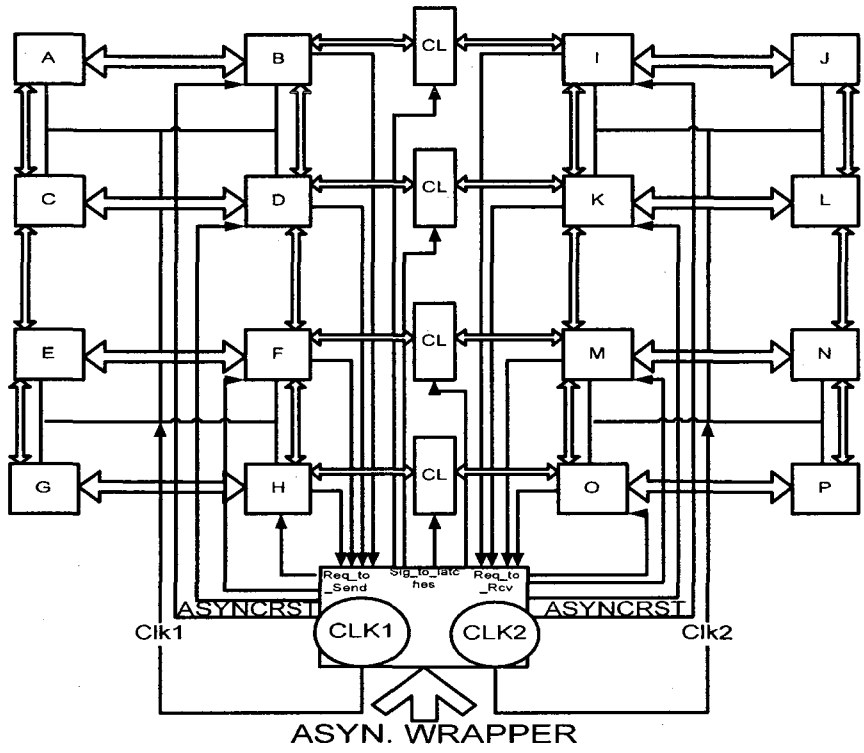


Figure 3.6. H-Tree with Pausible Clocking interface.

3.5 Summary and Discussions

This chapter introduced a high performance H-tree design methodology combining H-tree splitting and the use of asynchronous wrappers. Closed-form equations were derived for interconnect length as a function of the number of splits. Different design choices were investigated to cope with the challenges of communication between mutually asynchronous designs. Based on this study, design recommendations are made regarding the number of H-tree splits for an optimal performance given a certain die size. This design method shows a substantial improvement in clock frequency, up to 3 times, for a die size of 2 cm by 2 cm in 0.18 micrometer TSMC CMOS technology.

This work draws the attention of engineers and scientists that a class of CDN, which promises to be skew tolerant, poses severe problems of clock skew that it promises to solve due to process variations in modern DSM technologies. Hence, this work encourages system designers to rely on small and independent IP modules that may communicate through some interfacing mechanism. These independent IP modules in SoCs are likely to be associated with MCDs. Therefore, subsequent chapters of this thesis focus on the different interfacing mechanisms to communicate such IPs in MCDs. Both the design paradigms, asynchronous interfaces and synchronous interfaces, are investigated for a complete understanding of the subject.

Chapter 4: Crosstalk Effect in Event-Driven Asynchronous Handshake Schemes

It is described in the preceding chapter that CDN experiences a tremendous problem in providing synchronous clocks to the entire chip in the modern Deep Sub-micron (DSM) process technologies. Due to this limitation to CDN, SoCs designed in modern DSM technologies often comprise smaller modules in Multiple Clock Domains (MCD), so that the CDN can meet the required skew budget for each module of the system. Based on the knowledge of the preceding chapter, a designer may be able to identify the regions that are most affected by skew, where an interface must be inserted. Asynchronous handshake schemes are among the popular communication methods for interfacing these modules or Intellectual Properties (IPs) in MCD. Along with the need for methods to interface MCDs, there are also other challenges in advanced VLSI process technologies. Leakage power due to sub-threshold currents, domination of interconnect parasitic delays over gate delays [1], and increased crosstalk due to higher coupling interconnect parasitics are some of the phenomena that are affecting SoCs more significantly in modern DSM technologies than in earlier process technologies. Therefore, it is more challenging to design robust asynchronous interfacing mechanisms as the technology evolves.

Small spacing between adjacent interconnects and increased thickness of the metal layers lead to more coupling capacitance in modern DSM technologies. These coupling capacitances have dual deteriorating effects on the performance of the system. They add delay and introduce unwanted glitches (called as crosstalk glitches) that may lead to malfunction of the digital logic. Moreover, glitches on a signal in an event-driven design (a set which encompasses most of the asynchronous interfaces) can affect more adversely than glitches on a signal in synchronous designs. This is because, in event-driven systems, an event can cause a process to start, terminate, or change state at an inappropriate time. To the contrary, in a synchronous system, a process is sequenced with respect to clock edges in synchronous designs.

In this chapter, the increased importance of crosstalk glitches between different metal lines with the advancement of process technologies is investigated. To simplify the analysis, it is assumed that a good current return path surrounds each wire that is susceptible to an inductance effect. Therefore, this study is limited to the analysis of coupling capacitances only. This is in line with most of the crosstalk glitch analyses in advanced DSM technologies found in the recent literature [61]. It is found that, in 90nm and beyond technologies, crosstalk glitches can lead to system failure even for fairly short wires. The next section discusses the effects of inter-wire capacitance on some widely known asynchronous interfacing methods at the logic level. This logic level analysis is validated by electrical simulations for the representative interface designs for two well known asynchronous handshake schemes. Toward the end of this chapter, results of these circuit level simulations are discussed and finally a discussion summarized the chapter.

4.1 Inter-wire Capacitance

As the technology improves, multiple interconnect layers are introduced in order to provide more functionalities. The metal layers are usually divided in three categories namely local, intermediate, and global interconnects. For example, in the 180-nm TSMC process, a six metal layer technology, layer 1 to layer 3 are used for local interconnect, layer 4 and 5 for intermediate, and the top layer is reserved for global interconnect [62].

Capacitance in DSM technologies is not limited to parallel plate structures; rather, fringing and coupling capacitances are becoming a substantial portion of the total capacitance if not the dominant portion [3] [61]. Figure 4.1 shows the cross section of a metal layer sandwiched between an upper and a lower metal layer. Showing the upper and lower layers as carrying the ground signal in Figure 4.1 may be justified by the fact that each metal layer usually runs orthogonally to its immediate vertical neighbours, thus the combined effect due to all signalling is nullified. Therefore, the coupling effect of metal layers immediately above or below a metal layer reduces to that of a grounded wire. In this analysis, C_c refers to the coupling capacitance and C_g refers to the total ground capacitance.

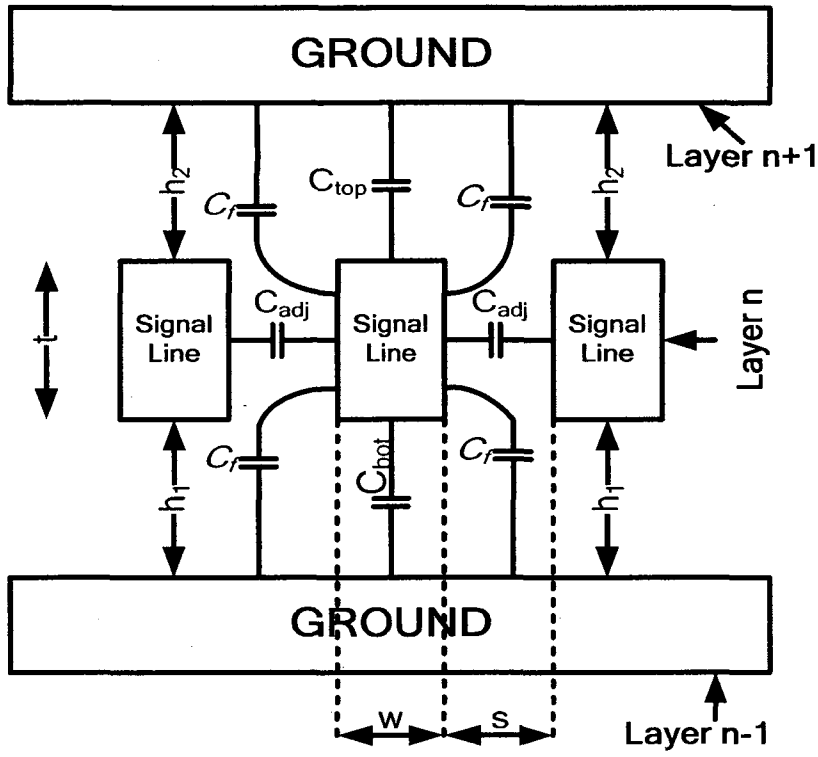


Figure 4.1. Local Layer interconnects

In Figure 4.1, $C_{gnd} = C_{bot} + C_{top}$ and total capacitance $C_{total} = C_{gnd} + 2C_c + C_{fringe}$. In the case of the top metal layer, Figure 4.1 does not have any Layer n+1 coupling. In this context, the analytical models of C_c and C_g are given by the equations in Chapter 8 of reference [61]. For ease of reading, these equations are reproduced here:

$$\frac{C_g}{\varepsilon} \cong \frac{w}{h} + 2.22\left(\frac{s}{s+.70h}\right)^{3.193} + 1.171\left(\frac{s}{s+1.51h}\right)^{.7642}\left(\frac{t}{t+4.53h}\right)^{0.1204} \qquad (4.1)$$

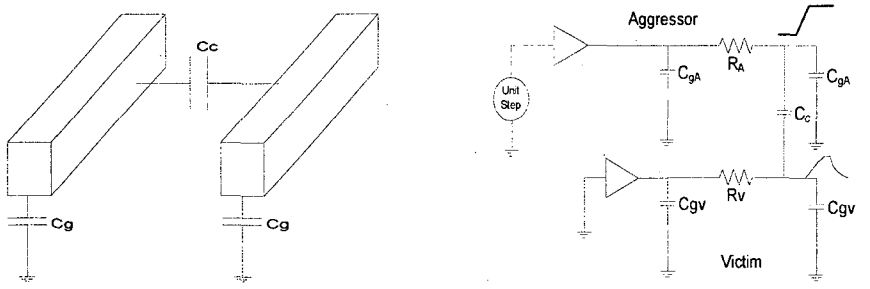
$$\frac{C_c}{\varepsilon} \cong 1.14\frac{t}{s}\left(\frac{h}{h+2.06s}\right)^{.0944} + .73\left(\frac{w}{w+1.6s}\right)^{1.4144} + 1.16\left(\frac{w}{w+1.87s}\right)^{.1612}\left(\frac{h}{h+.98}\right)^{1.179} \qquad (4.2)$$

where w, and t are the width and thickness of the wire, whereas s is the pitch (distance between the neighbouring lines of the same metal layer) and h is the distance from the

adjacent metal patch in the different metal layer. These parameters are summarized in Figure 4.1. As the process technologies grow deeper into sub-micron, spacing and thickness between layers are reduced [62], [63], [64]. Reduction in these dimensions leads to denser metal layout, and, consequently, to higher capacitive effects due to neighbouring wires, i.e. higher coupling capacitance. This is discussed further in the next section.

4.2 Crosstalk Comparison among Different Technologies

In order to understand the effects of the aforementioned parasitics, electrical simulations are required. Figure 4.2a, presents a three dimensional view of two parallel metal lines. Figure 4.2b shows the electrical parameters of the lines, each of the lines is driven by an amplifier. In order to provide practical value to the simulation we added 5 to 6 inverters to the ideal voltage source before supplying this voltage to the aggressor line. The aggressor line is considered to be the one that inflicts glitches, due to a transition of its own value, to the adjacent metal line. The line that exhibits a glitch in response to the transition in the aggressor line is called the victim line.



(a) (b)

Figure 4.2. (a)Three dimentional view of the metal lines (b) Electrical Equivalent Circuit for Simulations

As mentioned earlier, there are two major deteriorating effects due to crosstalk: increased delay and crosstalk glitches. Though delay hampers the performance of the system, crosstalk glitches may potentially result in system failure.

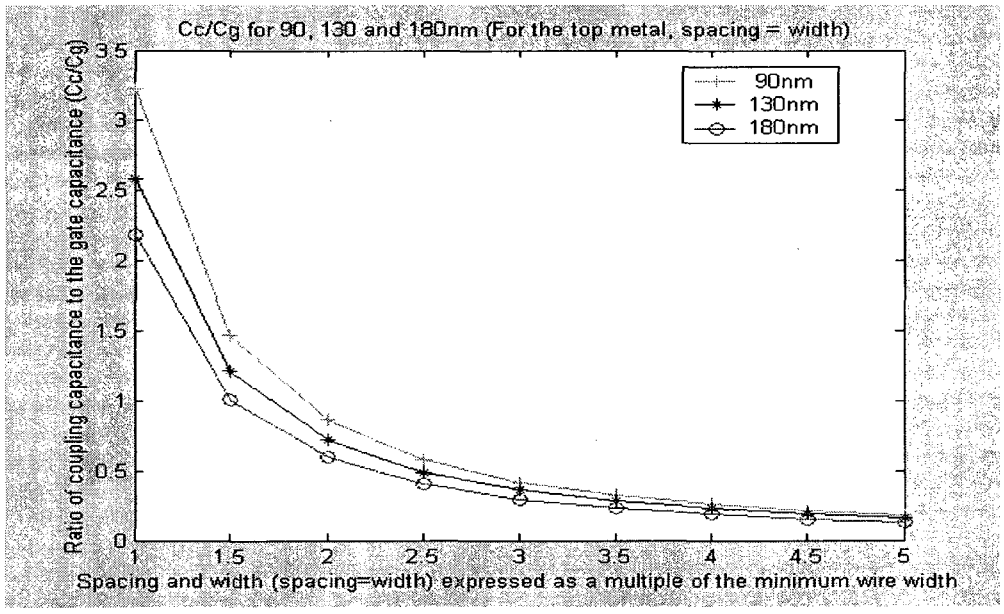


Figure 4.3. Cc/Cg for 180nm, 130nm and 90nm for top metal layers

In this study, emphasis is put on the investigation of the effects of glitches due to crosstalk. These crosstalk glitches can cause the system to fail when two conditions are met. First, the peak value of the glitch must be high enough to cross the threshold of the receiving device. Second, the crosstalk glitch pulse width must be wide enough to drive the loading capacitance to a potential that can be interpreted as a different logic value. In [61], glitch magnitude is computed as follows,

$$\frac{V_{peak}}{V_{DD}} = \frac{C_c}{C_c + C_{gv}} = \frac{C_c / C_{gv}}{C_c / C_{gv} + 1} \qquad (4.3)$$

where V_{peak} is the peak glitch voltage, V_{DD} is the supply voltage, C_{gv} is the victim ground capacitance and C_c is the coupling capacitance. It can be seen from (4.3) that the crosstalk glitch voltage gets to its peak theoretical value (V_{DD}) as C_c/C_{gv} ratio increases. With the C_c/C_{gv} ratio getting as high as 4 in 45nm technology [4], [61], crosstalk glitches for the circuits designed in U-DSM technologies are more susceptible to get close to its theoretical maximum value i.e. V_{DD}. Here, it should be noted that the effective coupling capacitance (which constitutes the equivalent load-to-ground capacitance on the driving gate) in a crosstalk prone environment is also dependent on the transitioning signal status on the neighbouring wires [6], [68], [69], [89], [90], which is represented as a switching factor (SF) in the literature [61]. Here on the other hand, in crosstalk glitches affecting quiet neighbouring lines due to transitions in aggressor lines, the SF is almost always unity. The SF is unity because quiet neighbouring line in ideal condition does not change its electrical potential, for details please refer to [61, Chapter 8]. Moreover, an increase in the coupling capacitance, as the one caused by reduced feature size, increases the settling time constant. This phenomenon contributes to longer crosstalk glitches, along with higher crosstalk glitch magnitude in U-DSM technologies.

Figure 4.3 shows that the value of C_c/C_g increases as the process technologies shrink. Utilizing the capacitance values from Figure 4.3, an electrical simulation is performed on the model of Figure 4.2b. In Figure 4.4, the result of the electrical simulation is shown for two different process technologies under similar driving conditions (minimum technology size drivers) and same wire length. Figure 4.4 and Table 4.1 provide a comparison of peak glitch voltage, under minimum size amplifiers, for two different process technologies. It can be seen from Figure 4.4 and the corresponding values in Table 4.1 that smaller

technologies are more susceptible to crosstalk glitches. It is shown in Table 4.1 that, in the case of the 180nm technology, the crosstalk glitch voltages do not cross the $V_{DD}/2$ level. Whereas, in 90nm, these crosstalk glitch magnitudes cross over $V_{DD}/2$ (which is the threshold voltage for a matched CMOS gate) for a substantial period of time. Appreciating the severity of crosstalk glitches, the ITRS-2007 introduced a new crosstalk glitch parameter which shows that, by 2015, in the 17nm technologies, even the first metal layer will have glitches of 25% of the switching voltage for a 46μm wire length; in comparison to 105μm for the 50nm technology [1].

TABLE 4. 1. SIMULATION RESULTS FOR CROSSTALK GLITCHES IN 90NM AND 180NM TECHNOLOGIES

90nm, 1mm wire, min. tech. size driver for victim, top metal layer, Cc= $113.6*10^{-18}$ * 1000 = $113.6*10^{-15}$, Cg= $35.6*10^{-18}$ * 1000 = $35.6*10^{-15}$, R= 0.022*1000/.42 = 52.38 Ohms, V_{DD} = 1.2V							
Period (ns)	Rise time (ps)	Vic/Agg (V/V)		Time Width (ps)		T for glitch > $V_{DD}/2$ (ps)	
		+ve	-ve	+ve	-ve	+ve	-ve
8	500	0.722/1.2	0.689	782	580	335	180
4	100	0.711/1.19	0.638	845	573	308	139
2	100	0.645/1.17	0.450	752	535	190	-
180nm, 1mm wire, min. tech. size driver for victim, top metal layer, Cc=$101.8*10^{-18}$*1000 = $101.8*10^{-15}$,Cg= $46.5*10^{-18}$ * 1000 = $46.5*10^{-15}$, R= .02*1000/.860 = 23.25 Ohms, V_{DD}= 1.8V							
Period (ns)	Rise time (ps)	Vic/Agg (V/V)		Time Width (ps)			
		+ve	-ve	+ve	-ve		
8	500	0.847/1.8	-0.873	623	465		
4	100	0.862/1.8	-0.855	669	460		
2	100	0.803/1.75	-0.671	633	431		

Hence, from the above discussion it is concluded that, as the gate length in DSM technologies is shrinking, the crosstalk glitch amplitude is increasing for similar driving strengths. These high crosstalk glitch magnitudes can sometimes be interpreted as a valid signal at the receiving end, for a particular range of load capacitance. Hence, it gives way

to crosstalk glitch propagation, which may lead to some serious consequences on system behaviour. Because, this phenomenon introduces a crosstalk glitch in the quiet neighbouring (victim) line, induced by the switching (aggressor) line, therefore it is termed as Aggressor-to-Quiet line Crosstalk (AQX).

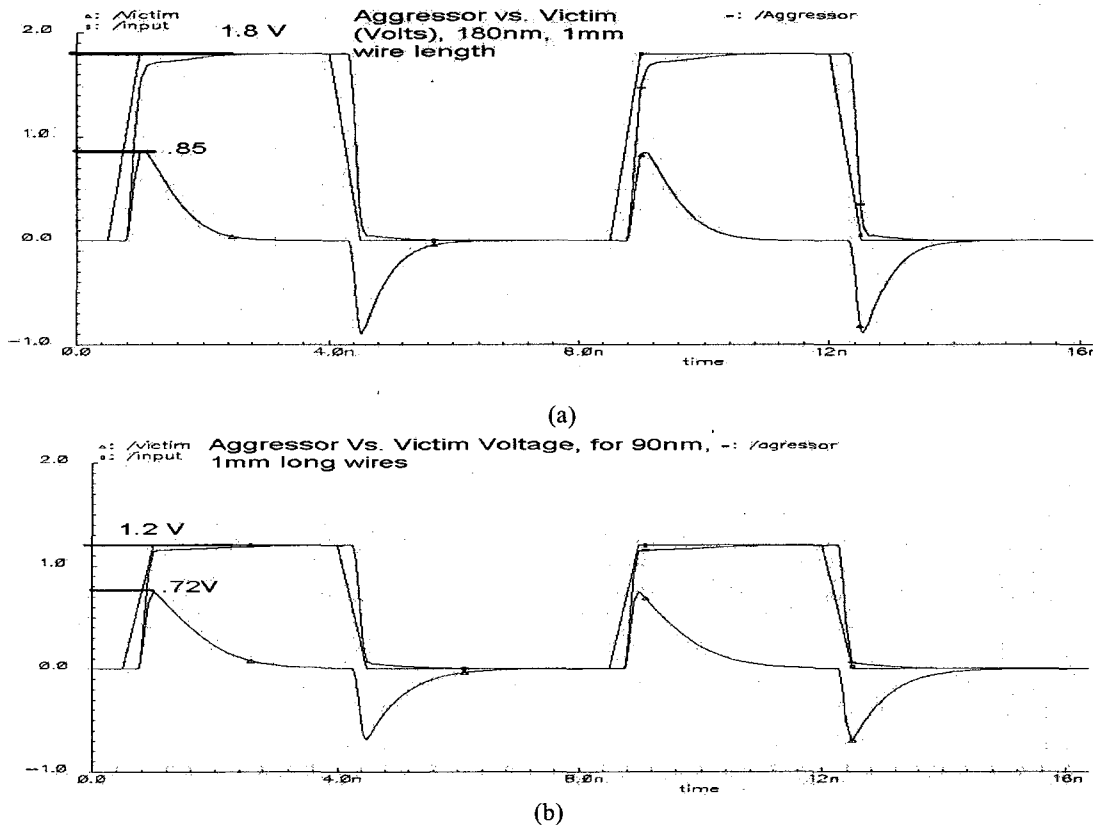


Figure 4.4. a) Aggressor and Victim Voltage for 90nm, 1 mm long wires (above) b) Aggressor and Victim Voltage for 180nm, 1 mm long wires (below)

For a first order comparison, the peak positive glitch (transition from 0 to positive potential represented as +ve glitch) observed for a 8 ns period, reported in the +ve column of the vic/agg (victim/aggressor) group in Table 4.1, is compared for the two technologies and it is found that the observed glitch, normalized for respective V_{DD}, is up to 1.3 times

larger in 90nm than in 180nm technology (e.g. 0.722/1.2 is 1.28 times more than 0.847/1.8). The time width column of Table 4.1 indicates the time the +ve glitch, or –ve glitch (transition from 0 to negative potential) for the respective –ve column, remains at more than half of the peak glitch value (i.e. more than 0.847/2 in the case of +ve glitch column of 180nm technology for 8 ns period). When the time width of glitches for the two technologies, given in Table 4.1, are compared, an up to 1.26 times wider width is observed for smaller (i.e. 90nm) technology, e.g. when results in the 8 ns row for both technologies are compared, 782 ps is found, which is 1.26 times more than 623 ps. The 90nm technology has one extra column for pulse width during which the crosstalk glitch magnitude is greater than $V_{DD}/2$. It can be seen in Table 4.1 that, for the 180nm technology, the crosstalk glitch magnitude does not reach $V_{DD}/2$. Here it is worth mentioning that the duration for 90nm technology glitches above $V_{DD}/2$ is more than three times the FO4 gate delay for the conventional 90nm CMOS technology [105].

4.3 Effects of Inter-Wire Capacitance on Well Known Asynchronous Interfacing Methods: Logic Level Analysis

The effects of glitches on average signals are not of much concern in synchronous designs because they behave with reference to a sequencing signal (clock). Synchronous designers choose clock periods such that signals become stable before the next assertion of the clock signal. Unfortunately, asynchronous designs do not have the luxury of clock signals. They are event-driven self-timed designs. So, if an event is generated due to an error in the system, it might lead to a series of erroneous signal transitions, which

consequentially trigger unwanted transactions with the possible result of system failure. Previously, asynchronous designers aimed only at making asynchronous systems hazard immune and race free. On the other hand, in advanced DSM technologies, a considerable attention must be given to crosstalk glitches. It is shown, in preceding sections that, in 90nm and smaller technologies, wide glitches in adjacent wires may reach $V_{DD}/2$. Crosstalk Glitches of this magnitude have significantly higher probability of reaching the threshold level of the driven gates, thus leading to erroneous event generation. The rest of this section elaborates on the crosstalk glitch effect in some of the well known self-timed asynchronous designs that are widely used as an interfacing technique in GALS systems.

4.3.1. Bundled-Data Protocol Based Design

In a bundled data protocol, data is bundled with request and acknowledge signals as shown in Figure 4.5(a). In Figure 4.5(b) waveform for a possible case of system failure due to crosstalk glitch is illustrated. Here, its is shown that the Ack signal glitches, labelled as False ACK in Figure 4.5(b), due to a 0-to-1 transition on the Req_in signal, shown as Req #1 in the Figure 4.5(b).

In the circumstances mentioned above, the sending module falsely assumes that the sent signal has been received by the receiver. Consequently, the sender may generate another request signal, shown as Req#2, while the receiver is still in the process of generating the Ack signal for the earlier request, Req#1, signal. Continuing the usual operation, Req#1 goes through a delay and produces Req_Out, which in turn generates the Ack signal. Hence, the sender may receive this Ack signal at a sensitive time window, i.e.

concurrent or close to the assertion of Req#2 signal as shown in Figure 4.5(b). Therefore, this phenomenon results in either false latching of the data or the loss of the data. This example is a logic level depiction of possible spurious signal generation under the influence of crosstalk glitch which in turn leads to system malfunctioning.

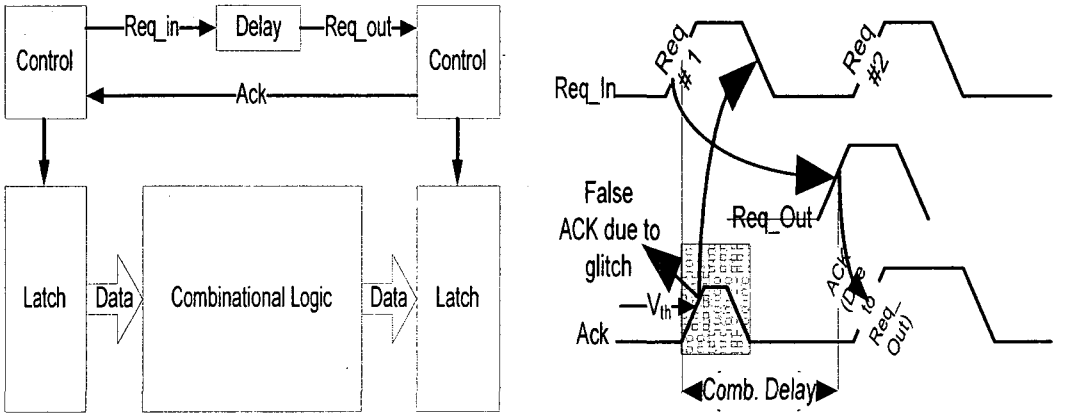


Figure 4.5. (a) Conceptual hardware implementation of the bundled data protocol (left) (b) Conceptual waveform to illustrate failure due to crosstalk glitch in the hardware implementation of the bundled data protocol (right)

4.3.2. 1-of-N Data Encoded Delay Insensitive (DI) Designs

Delay insensitive (DI) data encoding is used for communication between different modules that are distant and when *a priori* prediction of their timing constraints is difficult [45]. 1-of-N data encoding scheme is chosen in this thesis because, presumably, this scheme is least affected by crosstalk glitches. This is because of the fact that only one line get activated during signal transmission.

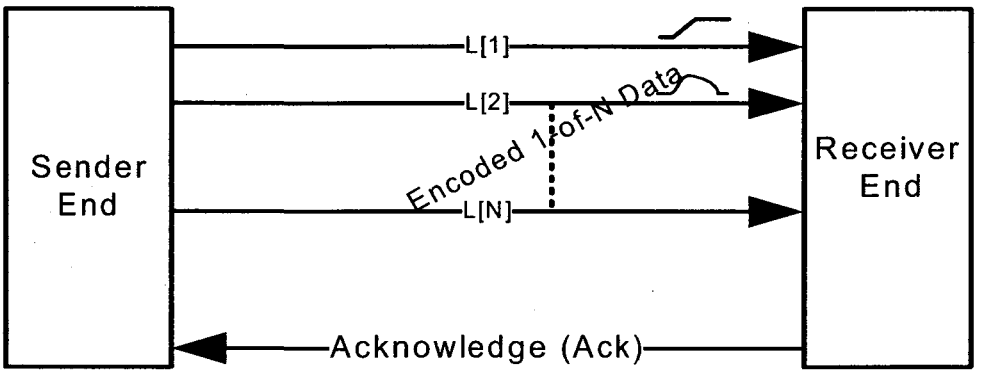


Figure 4.6. Generalized Hardware Implementation of 1-of-N Data Encoded DI Schemes

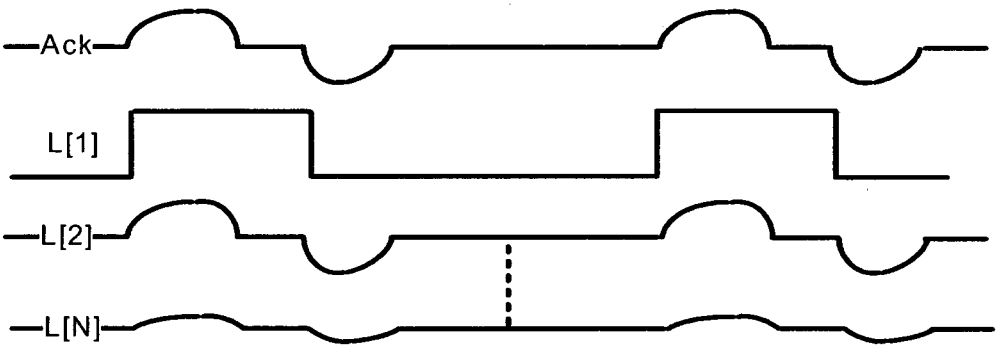


Figure 4.7. Expected waveform depiction under the influence of crosstalk glitches

Figure 4.6 shows the generalized hardware implementation of such data encoding schemes. These designs assert only one signal at a time, and the data and request are encoded within that signal [45]. The receiver is designed to decode these encoded data items. However, due to crosstalk glitches, it is possible that the receiver falsely understands that the line adjacent to the true asserting signal line is also asserted. Such a scenario is illustrated in Figure 4.7 where L[1] is the true asserting signal but inflicts crosstalk glitches on L[2]. L[1] may also induce a crosstalk glitch on the acknowledge

signal, 'Ack', if this acknowledge signal is neighbouring L[1]. It is also depicted in Figure 4.7 that the effects of crosstalk glitches reduce in farther neighbouring signals. As the receivers are designed to entertain only one signal at a time, therefore, concurrent assertion of two signal lines is improperly handled by the receiver and, in turn, this leads to system malfunction.

4.4 Validation of Crosstalk Glitch Effects in Asynchronous Circuits Using Electrical Simulations

This section characterizes the effects of AQX, which were described at the logic level in the preceding section, on asynchronous interfaces. Before going into details of asynchronous circuit mechanisms, some general assumptions and choice of asynchronous protocols are stated to facilitate understanding.

Assumption: In this work, only crosstalk glitches induced by an intrinsic signal transition within the same asynchronous domain, due to AQX, are analyzed. This assumption is justified, for instance, in GALS systems, for which the top metal layer is used to implement asynchronous interfacing signals. In this case, these signals are surrounded by signals that are typically least affected by AQX, i.e. V_{DD} and GND. Hence, such an implementation reduces the possibility of any signal external to the asynchronous domain to cause AQX. This makes the contribution from external signals to AQX small compared to the contribution from intrinsic signals.

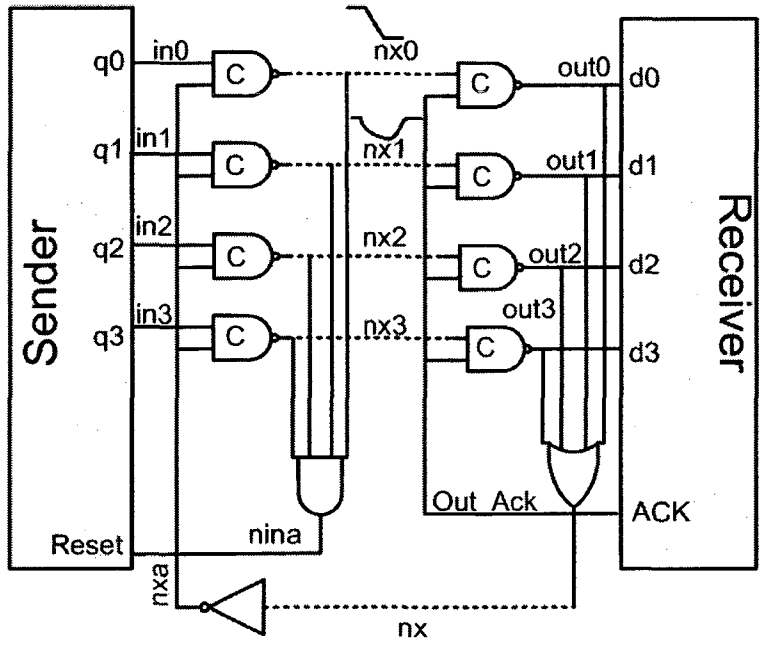
To simplify the analysis, it is also assumed that a good current-return path surrounds each wire, which alleviates inductive effects. Therefore, this study is limited to the

analysis of coupling capacitances only. This is in line with most crosstalk analyses in U-DSM technologies found in recent literature [55].

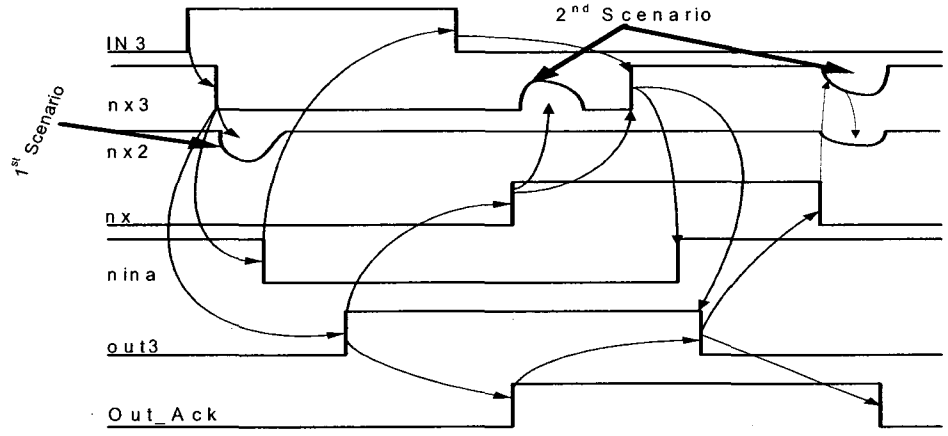
Choice of Asynchronous Interface: Several asynchronous interfaces are proposed to implement GALS systems. Most of these interfaces follow one of the two protocols bundled data protocol [8], [106], [107]or Delay Insensitive (DI) data encoded protocol [55], [67], [108]. In order to understand the effects of crosstalk glitches in these asynchronous handshake protocols, a comprehensive investigation is performed in the following sections for representative circuits of the above mentioned asynchronous interfacing protocols. The representative asynchronous interfaces chosen for this study are numerously cited and used as benchmark circuits in many recent papers as well [106], [107], [108].

4.4.1. Quantitative Crosstalk Glitch Analysis of the Conventional 1-of-4 DI Asynchronous Interface

This section analyzes the reasons for the vulnerability to AQX of a conventional 1-of-4 data encoded delay insensitive (DI) design [12], [55], as shown in Figure 4.8a. This analysis is based on the waveforms of Figure 4.8b that illustrates possible AQX scenarios. Subsequently, these scenarios are examined in relation to possible hardware implementations of Figure 4.8a, to identify the actual gates that are susceptible to propagate crosstalk glitches.



a)



b)

Figure 4.8. a) Hardware implementation of the data encoded DI Scheme b) Expected Waveform of the Design with glitch scenarios

Logic Level Sequence of Signals Causing AQX

In order to appreciate the effect of crosstalk glitches in such an interface, understanding the mechanics of the design is essential. The circuit of Figure 4.8a shows the following AQX-occurrence scenarios during normal circuit operation. The nx0 to nx3 group of

signals are set to V_{DD} as an initial condition. This group of signals is represented as nx[0-3] in the rest of this chapter. Similarly, in[0-3] represents the in0 to in3 group of signals and out[0-3] represents the out0 to out3 group of signals. According to the 1-of-N DI data encoded protocol, of which 1-of-4 is the special case shown in Figure 4.8a, only one of the input lines of in[0-3] can go high at a given time. When any of these input signals becomes high (in order to transfer some data), the corresponding line in the group nx[0-3] is pulled down to logic '0'.

Due to coupling capacitances, each switching line becomes an aggressor that can affect the neighbouring lines. Such perturbation lasts until the driver of the neighbouring line restores the charge on the quiet neighbouring line. This phenomenon introduces a crosstalk glitch in the quiet neighbouring (victim) line, induced by the switching (aggressor) line, and hence causes AQX. This is shown as the first scenario in Figure 4.8b, where nx3 falls in response to a rise in in3 that introduces a glitch on nx2.

The second glitch scenario arises when the transition in the nx signal, shown at the bottom of Figure 4.8a, causes a glitch in a neighbouring line (nx3 in this example). The second scenario appears twice in Figure 4,8b, one for each logic level transition on the nx line.

Validation using Transistor Level Simulation: To validate the above analysis, two separate sets of circuit level simulations were performed using the 90nm STMicroelectronics CMOS technology [64]. Initially, a 1-of-4 data encoded DI design is simulated with a model that includes the impedance of the lines but without the coupling effect. This set of simulations is designated as the no-crosstalk glitch effect (NXE) simulations. Even though the coupling effect is not taken into consideration in the first

simulations, for fair comparison with the simulations that include coupling capacitance, an additional ground capacitance of the same magnitude, as the coupling capacitance is included in NXE simulations. When performing NXE simulations, the 1-of-4 data encoded DI design is optimized for a particular worst-case delay. This worst-case delay is measured from the rise of any of the in[0-3] signals to the latching of the data at the corresponding output. The value of this particular delay was optimized and is close to 450 ps for a wire length of 1.5 mm. In simulation results reported later, the interconnect length was set to 1.5 mm and the same 90- nm STMicroelectronics CMOS technology was used. In all these simulations, the top level metal layer is used, and interconnects are modeled using a 5Π model, for which the relative delay errors associated with the use of lumped models are less than 3% [3]. Top metal layer is used because inter-module communication requires longer wire lengths and top metal layer provides lowest resistance.

The second set of simulations includes coupling capacitances and is dubbed "with crosstalk glitch effect" (WXE). This set of simulations is performed on the same optimized interface design (i.e. with no change in transistor sizing), while the line impedances include coupling effects.

Simulations on the circuit of Figure 4.8a for the four process corner cases (SS, SF, FF, FS)[1] and for a typical case (TT)[1] were performed. These transistor-level simulations consider the case when in3 and in2, alternately, rise to logic '1' to measure the undesirable crosstalk glitch magnitude at the output. The WXE transient circuit level simulation results are shown in Figure 4.9. These results show that such glitches may be latched as a valid output signal. For example, Figure 4.9 shows that the out3 glitch magnitude is

[1] SS means both PMOS & NMOS are Slow; SF Slow PMOS, Fast NMOS, FS Fast PMOS, Slow NMOS, and FF means both are Fast, TT represents both MOS have typical values

approximately VDD/2. This crosstalk glitch propagation can lead to a malfunction of the system, as the simulation shows that two of the out[0-3] signals are asserted concurrently, which is forbidden in the protocol.

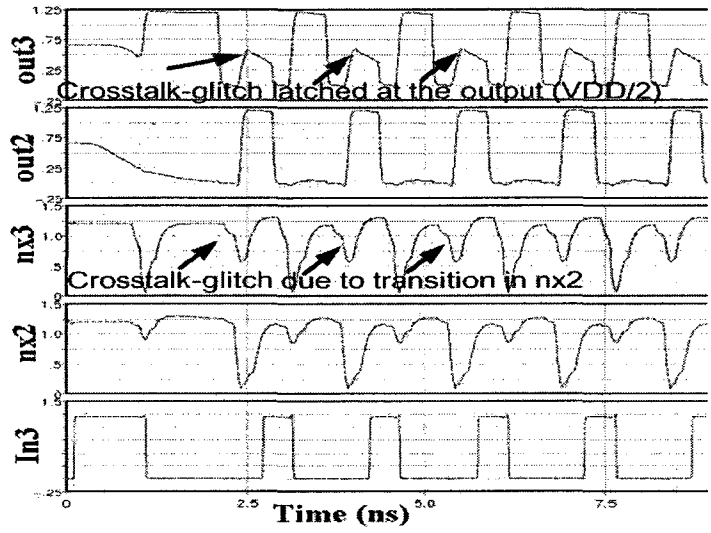


Figure 4.9. Circuit level simulation results for optimized 1-of-4 Data Encoded DI Design scheme, shown in Figure 4.8-a, for an interconnect length of 1.5mm (WXE Simulation).

The magnitude of the crosstalk glitch peak voltage at out3 is shown in Figure 4.10 as a function of the interconnect length. This illustrates that the crosstalk glitch magnitude on signal out3 increases with the length of the wire. It is also observed that the crosstalk glitch magnitude may reach VDD/2 (0.6V), for 1.5 mm length interconnect, which is high enough to turn on the subsequent gates and allow glitches to propagate. Note that the variations in crosstalk glitch magnitude among different corner cases in Figure 4.10 is due to the non-balanced sizing of PMOS and NMOS transistors, which is a requirement to optimize delays.

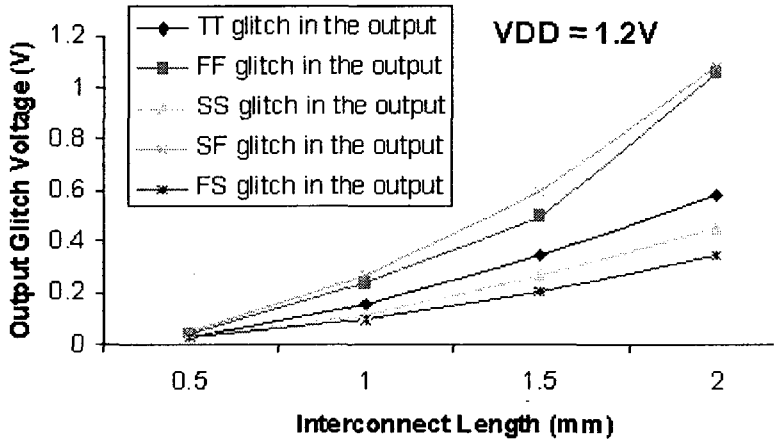


Figure 4.10. Crosstalk glitch peak voltage in 1-of-4 Data Encoded DI Design (having beein optimized for latency in NXE simulations)

Through the above simulation results, it is demonstrated that crosstalk glitches can lead to malfunction in the 1-of-4 data encoded delay insensitive asynchronous interfaces. The next section demonstrates the same behaviour occurs in the conventional bundled data asynchronous interface.

4.4.2. Crosstalk Glitch Analysis of Bundled Data Handshake Schemes

Another class of asynchronous interface often used in GALS systems for inter-module communication is the bundled data protocol [8], [9]. This section examines the functionality of those asynchronous designs and the possibilities of AQX during their normal operation. Figure 4.11 shows the generalized block level structure for asynchronous interfaces utilizing this protocol [45]. In [8], this concept is further elaborated using state machines along with synthesis results obtained through the 3D tools [65], [66]. The synthesized Boolean equations of the State Transition Graph (STG),

elaborated in [8], are given in Table 4.2. These equations are obtained by synthesizing the STG through the 3D tools.

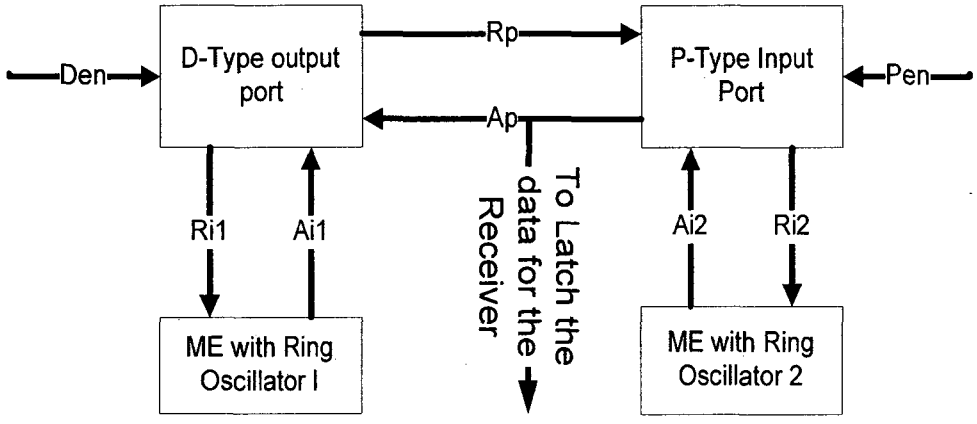


Figure 4.11. Conventional bundled data protocol: the block diagram Simulation Results

Signal Transition Sequences Causing AQX in the Bundled Data Protocol

TABLE 4. 2. SYNTHESIZED BOOLEAN EQUATIONS FOR THE STG OF THE PROTOCOL ELABORATED IN [8]

	Synthesized Boolean Equations
Doutput Port	Ri = Ap + Den' Z1 + Den Z1' + Den' Ri Z0' Rp = Den' Ai Ap' Z0' + Ai Ap' Z0' A1' Z0 = Den'Ap + Ai Z0 Z1 = Den Ap + Den Z1 + Ai'Z1
Pinput port	Ri = RpRi + Pen' Rp Ti Ap = Ai Ti = PenAi + Ai'Ti+Ri'Ti

From Figure 4.11, it is obvious that the most crosstalk-prone signals in this interface are the longest parallel-running handshake signals R_p and A_p. On closer inspection of this protocol [8] and the corresponding synthesized Boolean equations in Table 4.2, it is observed that signal R_p undergoes a transition prior to A_p. Therefore, there is a possibility of an AQX glitch occurrence in A_p. Large crosstalk glitch in A_P may lead to premature

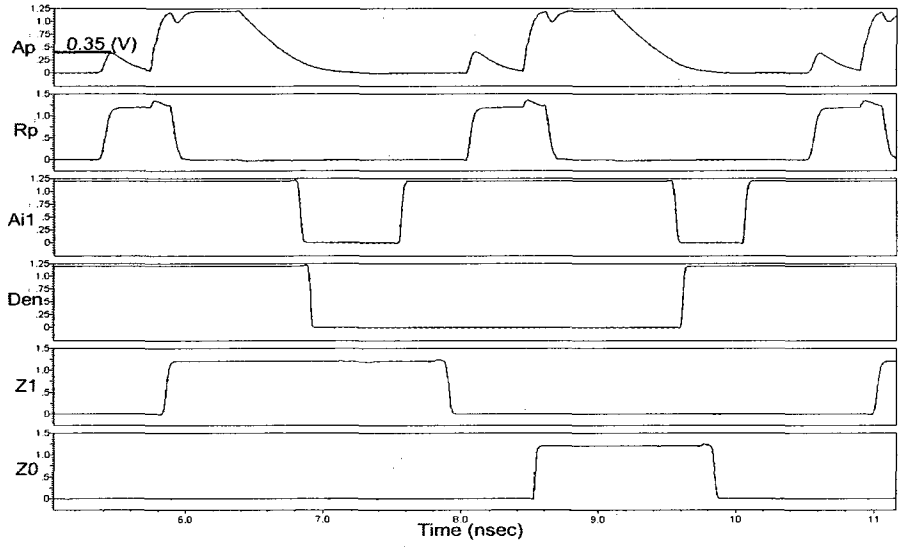
termination of the handshake. It is also noticed that this glitch in A_p forces false latching of the data in A_p, which is an added adverse effect due to crosstalk glitches.

Validation Using Transistor Level Simulation: To validate these analytical results and to quantitatively characterize the crosstalk glitch effect, two sets of simulations were performed. Similar to the case of DI designs, these sets are designated as NXE and WXE simulations. As part of the first set of simulations with no crosstalk (i.e. NXE), the bundled data interface is optimized for a delay of, approximately, 3 ns when the interconnect length between the two communicating module is 2 mm. The delay is measured from the assertion of the Den signal to the negation of A_{i1}, which indicates that the request (R_p) has been sent and acknowledged (A_p), for complete protocol specification please see [8]. As is usually done with GALS systems using asynchronous interfaces, the sender clock is paused throughout the handshaking [7], [8], [9].

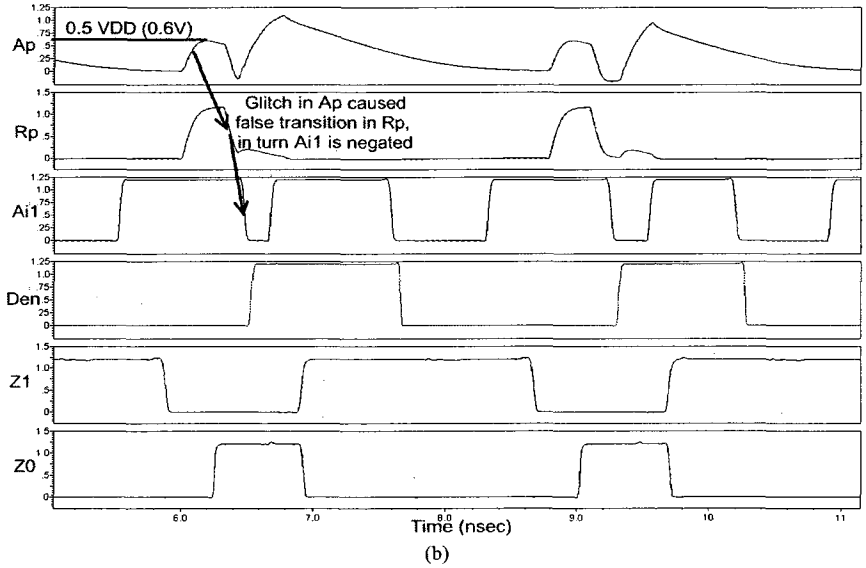
In order to do a fair comparison between NXE and WXE, all interconnect parasitic capacitances are connected in both cases. However, in the NXE set of simulations, both nodes of the C_c capacitors are connected with an equal magnitude capacitor to the ground. By contrast, in the WXE simulation set, one C_c capacitor is connected between each pair of handshake signals. The WXE simulation set uses the same transistor sizing as the optimized NXE design.

Figure 4.12 shows the WXE transient circuit level simulation results for this protocol. The simulated interconnect length between the two modules was varied from 0.5 mm to 3 mm. In Figure 4.12a, simulations performed for an interconnect length of 0.5 mm, it can be seen that A_p glitches with the rise of R_p, but this glitch magnitude is not large enough to cause a glitch to propagate in the circuit. However, in Figure 4.12b, when the interconnect length is increased to 2 mm, the magnitude of the glitch is large enough to falsely negate R_p and consequently terminating the handshake scheme prematurely (indicated by the negation of A_{i1}). Arrows in Figure 4.12b indicate one such occurrence. Figure 4.13 shows

the rise in crosstalk glitch magnitude with the increase in interconnect length. Glitch magnitude of $V_{DD}/2$ (which is the threshold voltage in balanced static CMOS gates that allows glitches to propagate) is reached for an interconnect length of 2 mm.



(a)



(b)

Figure 4.12. Electrical simulation results for a circuit level implementation (TT) of the bundled data protocol (a) interconnect length of 0.5 mm (b) interconnect length of 2 mm

Hence, the above examples demonstrate that conventional asynchronous circuits are susceptible to glitches due to AQX under normal operating conditions. This quantitative analysis also shows that, the identification of AQX effects requires detailed electrical simulations, and these effects are otherwise hidden from logic designers.

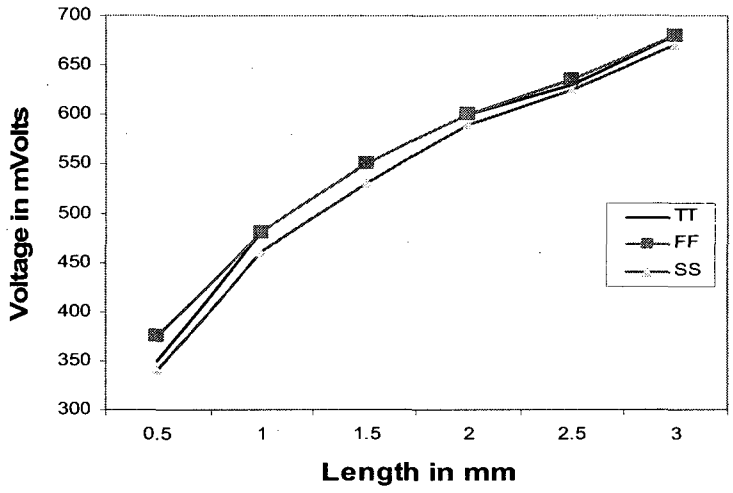


Figure 4.13. Crosstalk glitch peak voltages in Bundled Data protocol

4.5 Summary and Discussions

In this chapter, it was shown that digital designs are getting vulnerable to crosstalk glitches as technology feature size is shrinking. Quantitatively it is shown that, for a 1 mm wire length, substantial increase, about 1.3 times, in crosstalk noise-impulse width and magnitude is observed for the 90nm ST Microelectronics technology when compared to the 180nm TSMC technology. The adverse contributions of these crosstalk glitches on

event-driven asynchronous handshake schemes, that are notably proposed as a solution to the clock distribution and timing problems found in advanced SoCs, are studied in details. It is demonstrated that the asynchronous handshake schemes may allow crosstalk glitches to propagate into the circuits when influenced by AQX under normal operating conditions in modern DSM technologies. Representative circuits of two widely used classes of asynchronous handshake circuits are simulated under the 90nm STMicroelectronicds CMOS technologies. It is shown that a crosstalk glitch can propagate in these interfaces for reasonably short interconnect lengths: 1.5 mm for the 1-of-4 DI protocol and 2 mm for the bundled data protocol can cause glitches of magnitude $V_{DD}/2$ or higher, which is significant enough to trigger the subsequent gates to which this signal acts as an input. Hence these simulation results prove that conventional asynchronous interfaces are, in principle, not immune to intrinsic crosstalk glitches due to AQX.

It can be observed that, in order to understand the AQX glitch propagation in asynchronous interfaces, detailed circuit level analysis is required. Asynchronous protocols are designed at logic level (or further higher in the abstraction levels), and hence often due emphases on the physical aspects are ignored. This is mainly because there is no work that can translate these physical level behaviours into logic level. Therefore, there is a requirement to develop a methodology that can predict the vulnerable nodes/conditions/primary inputs that must be avoided at the logic abstraction level. On the other hand, when they cannot be treated at logic abstraction level then this information is relayed to the physical level, where any of the glitch quenching mechanism can be utilized. In this regards, the next chapter proposes a novel modeling

approach to identify the possibilities of AQX glitches in asynchronous interfaces at logic abstraction level.

Chapter 5: Crosstalk Glitch Propagation: Generalized Notion, Modeling, and Validation

Asynchronous interfaces are typically defined at the gate or RTL logic levels [7], [8], [44], [67]. Hence, some of the physical level characteristics are ignored. Crosstalk glitch propagation due to AQX, as demonstrated by electrical simulations in the previous chapter, is one such physical-level phenomenon. Crosstalk glitch propagation due to AQX may lead to system malfunctioning but, currently, there is no mechanism or analysis method available (other than circuit level simulations) that may check such catastrophic outcome of AQX at the logic abstraction level and warn the design engineer before implementation of the asynchronous protocols. Therefore, there is a need to bridge this gap between these higher abstraction level definitions of asynchronous interfaces and the problems associated with their physical level implementation. As an attempt to bridge this gap, in this chapter, a novel technique is proposed to model crosstalk glitch propagation due to AQX at the logic abstraction level. This technique facilitates asserting asynchronous interface robustness to crosstalk glitches. This model can accurately identify the possibility of intrinsic crosstalk glitch propagation, at the logic level, in asynchronous circuits that utilize conventional logic gates and Muller 'C' elements. As most of the

asynchronous circuits consist of conventional elements [7] – [9], [19], therefore we may say that our technique is applicable to most of the asynchronous interface schemes. Hence, this analysis builds a framework that allows representing the possible behaviour of a logic structure in the presence of a crosstalk glitch without resorting to circuit level simulations. The proposed technique includes a novel technology-independent wire glitch element (WGE) to represent glitches at the logic level. Utilizing AQX identification using the WGE, glitch propagation (GP) sets are proposed for each logic element to describe the conditions under which a crosstalk glitch may propagate. Leveraging these GP sets, a complete set of conditions is modeled for crosstalk glitch propagation. These conditions are applicable to the most widely applied asynchronous handshake schemes used for interfacing MCD modules. This modeling can make the logic designer aware, at an earlier design-flow stage, of the possible system conditions that lead to crosstalk glitch propagation.

The next section elaborates the uniqueness of AQX glitches as compared to other glitches addressed in the literature. The following section describes the proposed modeling technique, while providing definitions of the new concepts along the way. Subsequent section leverages these GP sets to formalize a comprehensive set of conditions for modeling crosstalk glitch propagation in asynchronous handshake schemes. A design perspective is discussed before an experimental validation of the model is demonstrated. Back-annotated simulation results using Xinlinx Virtex II-Pro (XC2VP30-7F896) FPGA is reported and, to finish off this chapter, a summary is provided.

5.1 Uniqueness of AQX glitches: A Motivation for Designing a Logic Level Modeling Technique for Crosstalk Glitches

Conventional asynchronous circuits are designed to avoid data hazards and race conditions [65], [66], but these design methods fail to address non-idealities introduced by crosstalk glitches. Usually the crosstalk effects in circuits implemented in DSM technologies were mitigated with the introduction of encoding in the data bus [68], [69], [91], [92]. But such techniques falls short of addressing crosstalk glitch influence on control signals, such as the handshake signals in asynchronous interfacing circuits, which may induce even more serious consequences if not treated properly. Furthermore, in modern DSM technologies, as stated earlier, considerable crosstalk glitches appear in quiet lines due to transitions in aggressor lines (AQX). These crosstalk glitches may propagate undesirably, and lead to eventual system failure, which cannot be alleviated by using behavioural design method that checks for intrinsic transition faults in a sandwiched wire [70]. Also, crosstalk glitches due to AQX are different, in nature, than settle-time glitches, which appear in the combinational logic of the synchronous designs due to mismatch in the delay paths. Thus, the context of crosstalk glitch propagation due to AQX is different from all the other type of glitches, which are commonly known to the design community. Therefore, a need arises to investigate the affect of crosstalk glitch propagation due to AQX. An investigation is performed, in the following sections, which leads to the conditions that model crosstalk glitch propagation due to AQX in asynchronous handshake schemes. The benefit of this modeling technique is its applicability at logic abstraction level, which is earlier in the design cycle than physical

abstraction level and hence helps the designer to come up with a robust interface that avoids crosstalk glitch propagation within the interface.

5.2 Generalized Notion to the Glitch Propagation Phenomenon

This section provides a generalized notion to the glitch propagation phenomenon in asynchronous interfacing circuits. Using this notion, a designer can identify the possibility of crosstalk glitches, which may propagate into the asynchronous circuits and make them unreliable.

5.2.1. Preliminary Notations

Inspired by the D-algebra, proposed by the test community [73], [95], a new concept is introduced in this section for modeling the effects of crosstalk glitches in asynchronous handshake schemes. It helps in establishing criteria for crosstalk glitch propagation in logic gates and 'C' elements in the context of asynchronous handshaking schemes. As part of the proposed framework, an essential notation is first defined as follows.

As it is shown in the preceding chapter, in the context of asynchronous handshake schemes, the aggressor line (AL) and the victim line (VL) of interest are handshake signals that are communication lines between two mutually asynchronous communicating modules, which may be classified as sender and receiver modules. Hence, such lines (AL

and VL) are either input or output to these modules. The analysis performed in this section only considers the modules where VL is an inbound signal. This is because the possibility of glitch propagation is only associated with such modules (this is further explained in definition 5). Thus, the input and output notation is used with respect to the modules to which VL is an input. Note that, because asynchronous circuits, in general, consist of logic elements or Muller 'C' elements, AL and VL can always be studied as inputs or outputs to various logic elements. The '_in' in AL_in (VL_in) and the '_out' in AL_out (VL_out), respectively denote whether the AL (VL) signal is an inbound signal or an outbound signal to the relevant asynchronous module.

Definition 1: T (T') is a signal transition from logic level 0 to 1 (1 to 0).

Lemma: T (T') is said to be deterministic if it is associated with AL_out. This is because the conditions for asserting the signal are known to the module under study. Consequently, T (T') is called nondeterministic if it is associated to AL_in.

Definition 2: G (G') represents glitches in the VL due to transition T (T') in the AL. Glitch on a particular VL, say A, due to T (T') in AL is represented as G_A (G_A'). VL returns to its stable state after a bounded delay, Δt_G.

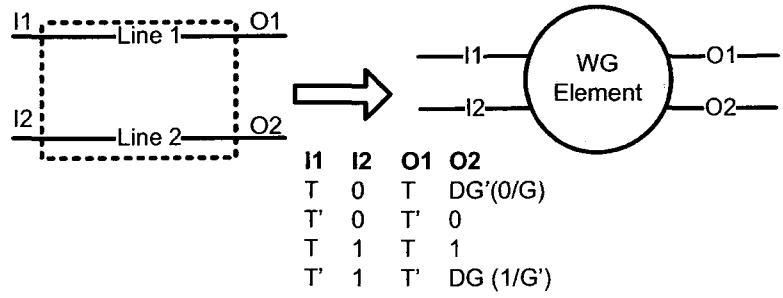
Note that, in the context of AQX, VL may glitch, G (G'), only if VL and AL are at the same logic level before the transition T (T') in AL. This is explained further in definition 4.

Definition 3: DG (DG') represents composite logic values of the form v/vg, where v and vg are values of the same signal in glitch-free and non glitch-free circuit, respectively. The composite logic values that represent errors, 1/G' and 0/G, are denoted by the symbols DG and DG', respectively. This notation was chosen by analogy with D-algebra;

where 1/0 and 0/1 are, respectively, represented as D and D'.

5.2.2. Wire Glitch Element

Definition 4: Wire Glitch Element (WGE) is a logic level representation of the AQX phenomenon, as shown in Figure 5.1. The WGE translates a physical-level AQX-affected signal into a representation that can be understood at the logic abstraction level. In Figure 5.1 the values on I1 and I2 are the values imposed on the AL and VL, respectively. Provided there is no AQX phenomenon, then O1 and O2 keep the logic values that are imposed on I1 and I2, respectively. On the other hand, if the wires are subject to AQX then O1 and O2 represent the resulting propagated values on the wire due to the AQX effect. The logic table in Figure 5.1 shows that, whenever I1 and I2 are at same initial logic levels, T (T') in I1 causes DG' (DG) in O2, while O1 keeps the same value T (T'). It is imperative for this modeling approach to define the criteria for WGE insertion.



I1	I2	O1	O2
T	0	T	DG'(0/G)
T'	0	T'	0
T	1	T	1
T'	1	T'	DG (1/G')

Figure 5.1. Logic Level Representation of AQX, called Wire Glitch Element (WGE)

Criteria to insert a WGE: Any pair of intrinsic handshake signals that may utilize global or intermediate wires and that, at the same time, experiences the following condition will require WGE insertion:

According to the asynchronous protocol, at some stage during the handshake, one of the two signals, which can potentially run parallel, does a transition T (T') while the other signal remain stationary at logic level 0 (1.

As glitch magnitude is proportional to wire length, therefore consideration should be given to the length of wires as well. From the simulation results for the technology that is been used in the preceding chapter, the 90nm STMicroelectronics CMOS technology, it can be safely concluded that, in an asynchronous interface, only the longest wires require WGE insertion. As we move deeper into the DSM technologies, WGE insertions will be required even for local length interconnects as explained earlier. Hence, the requirement of evaluating more factors will arise; following is a list these factors:

1) Knowledge of the maximum length of wire for which a FO4 gate can keep the glitches within $V_{DD}/4$. For example, in ITRS-2007[11], a new metric is introduced to cope with the catastrophic crosstalk effects predicted in future technologies.
2) Identify the potentially weak drivers for non-transitioning wires.
3) Special consideration should be given to feedback signals because they are usually driven by large drivers. Hence, glitches inflicted due to a feedback signals are strong candidates for WGE insertion.

Leveraging WGE to identify AL and VL

In order to model AQX glitches at the logic abstraction level, it is necessary to identify the AL and the VL. The following steps are proposed for identifying AL and VL in an asynchronous interface. These steps leverage the WGE:

I) Identify the signals where WGE insertion is required (using the criteria for WGE insertion)

II) Leverage the knowledge of the asynchronous protocol to find the sequence of transitions in the handshaking schemes.

III) For each transition, check the WGE truth table shown in Figure 5.1 for possible glitch occurrence; hence, identify AL and VL.

IV) If the asynchronous handshake scheme involves a RTZ signaling scheme, then such a glitch may occur twice (positive and negative transition); therefore, reapply this method for the second transition.

5.2.3. Glitch Propagation (GP) Sets

Definition 5: Glitch Propagation (GP) set is a set of conditions that allow a glitch, in different logic gates and Muller 'C' elements, to propagate. This work focuses on four specific gate types: inverter, two-input AND and OR gates, and Muller 'C' elements. It is possible to express all circuits of interest using these gates. Also, the analysis method used to derive the glitch propagation sets can easily be generalized to cover all other gates of interest. As an example, the GP set for the AND gate that produces DG at the output is {(1, DG), (DG, 1), (DG, DG)}. Figure 5.2 summarizes these conditions for the four considered logic elements. The GP sets for the Muller 'C' element have a timing notation as well (t- and t+). The second GP set of the 'C' element, {(0,DG), (DG, 0), (DG, DG)} → t-=1 and t+=DG, indicates that, if the output of the 'C' element is logic '1' before the occurrence of composite logic value DG in any or both the inputs of the 'C' element, then, due to the glitch, the output of the 'C' element will have a composite logic value of DG at the output after the gate propagation delay at time t+.

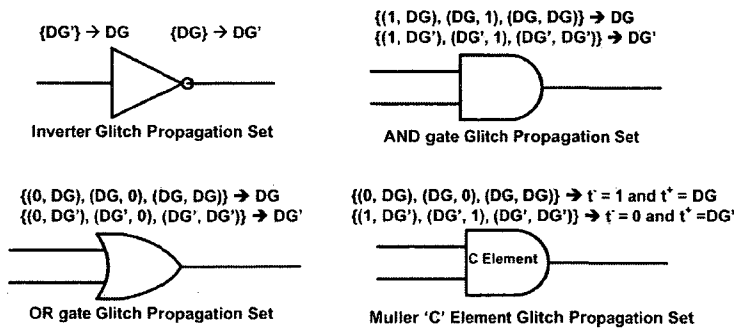


Figure 5.2. GP sets for the inverter, the AND and OR gates, and the Muller 'C' element

Note that each element of these GP sets always contains a VL_in (in the form of DG or DG'), as shown in Figure 5.2. In the example of the GP set of the AND gate discussed above, each element of the set has at least one input as DG, which corresponds to the VL_in, while the other input is either DG or the non-controlling value of the gate, which is '1' for the AND gate. Therefore, the glitch-propagation analysis is considered complete only if it contains all the modules to which VL_in is one of the inputs. VL may also be an output to the module, but it cannot contribute to glitch propagation in that module. However, it may act as an input to the opposite module, so it must be analyzed when the opposite module is investigated. A DG or DG' on the right hand side of GP sets in Figure 5.2 shows that the output of the gate represents propagation of the glitch.

5.3 Crosstalk Glitch Propagation Modeling

This section utilizes the notions developed in the preceding section to model crosstalk glitch propagation. It has been seen above that G (G') in VL is a consequence of T (T') in

AL, which may be AL_in or AL_out to the module. Therefore, it can be concluded that "for all GP sets, there exist some AL, such that AL is either AL_in or AL_out causing G (G') in VL_in". The statement in quotes can be written mathematically as the following corollary:

Corollary: $\forall GP \exists AL \,|\, ((AL\_in \wedge VL\_in) \vee (AL\_out \wedge VL\_in))$

Explanation: Before elaborating on the above corollary, some background information is provided here: Each asynchronous module may consist of many logic elements. AL_out and VL_in can be associated with one or more logical elements inside the module. Note that the only necessary input signal for the glitch to propagate in a logic element is VL_in, which is represented as DG or DG' in GP sets. AL_in and AL_out signals can cause VL_in to G (G'), but may not necessarily be part of the same logic element that propagates the glitch. The above corollary shows that glitch propagation is possible in any of the following three cases:

1) AL_out $\wedge$ VL_in, i.e. the module has a VL_in that glitches, G or G', due to T or T' in AL_out. Based on the possibilities of inputs to the logic elements through which the glitch may propagate, this case can be further sub-divided. It is evident from Definition 5 that one of the two inputs of the glitch propagating logic element should be VL_in. The other input should have a sensitizing value in order to fulfill the condition of a GP set. Physically such a signal can be: (a) an independent sensitizing signal, shown in Figure 5.3a (b) another handshaking signal which is affected by crosstalk, illustrated in Figure 5.3b or (c) a feedback signal which is derived from AL, shown in Figure 5.3c.

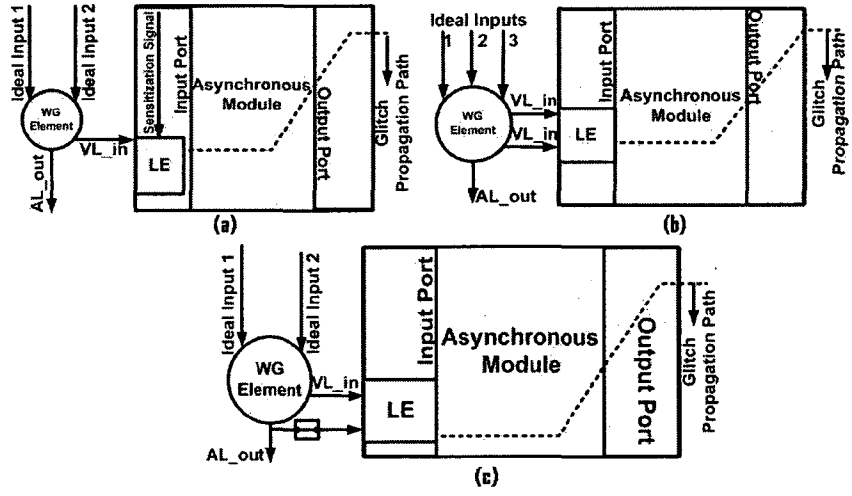


Figure 5.3. Pictorial illustration of possible inputs to the glitch propagating Logic Element (LE) for the case AL_out ∧ VL_in.

2) AL_in ∧ VL_in, i.e. the module has a VL_in that glitches (G or G') due to transition (T or T') in AL_in. Again based on the possibilities of inputs to the logic element through which glitches may propagate, this case can also be further sub-divided into the following two cases (following the same guidelines as adopted for case 1): a) GP set inputs are VL_in and sensitizing signal, or b) the GP set consists of VL_in and AL_in (or two VL_in signals). Figure 5.4 elaborates these cases pictorially.

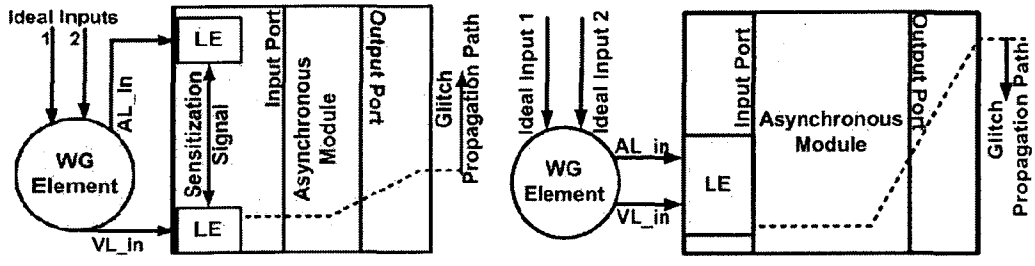


Figure 5.4. Pictorial illustration of the possible inputs to the glitch propagating Logic Element(s) (LE) for the case *AL_in* ∧ *VL_in*

3) ($AL\_in \wedge VL\_in$) $\wedge$ ($AL\_out \wedge VL\_in$). There are several possibilities associated with this case. To simplify the analysis, it is treated as two different non-concurrent events. This case is divided into two sub-cases as follows: a) if VL_in is the same physical line for both events (this is a possibility when the same VL_in is subject to the glitch due to two physically different ALs), and b) if two physically different VL_in(s) are involved (i.e. the asynchronous module receives two or more VL at two different time instances due to physically different AL(s)). The GP sets for these sub-cases can be obtained by utilizing the description of the preceding two cases (i.e. cases 1 and 2).

Modeling crosstalk glitch propagation at the logic abstraction level: In essence, a designer begins the analysis by utilizing the proposed WGE to identify the AL and VL and their directions (input or output to the module). Next, the designer has to identify the logic element(s) to which the VL is an input (VL_in). Benefiting from the different cases of the corollary, which use the proposed GP sets, a designer can predict whether or not the glitch may propagate in the circuit under study. Hence, this modeling technique enables the designers to identify crosstalk glitch propagation at the logic abstraction level. This modeling approach is broad, as it analyzes the constituents of asynchronous circuits (i.e. the logic elements) and therefore can be applied to a broad range of asynchronous handshake interfaces. These steps are summarized in the flowchart shown in Figure 5.5. . In this figure, it is assumed that S is the total number of WGE required for the interface. This number is obtained by applying the WGE insertion criteria to the given interface. The algorithm shown in Figure 5.8 illustrates the conditions under which the AQX glitches propagate to Primary Outputs (PO). The final outputs of this flowchart are the Primary Inputs (PI) and/or initial conditions along with the identification of the WGE node where

the glitch enters the asynchronous interface. The following section utilizes this modeling approach to obtain the conditions under which glitches may propagate in the example asynchronous interface.

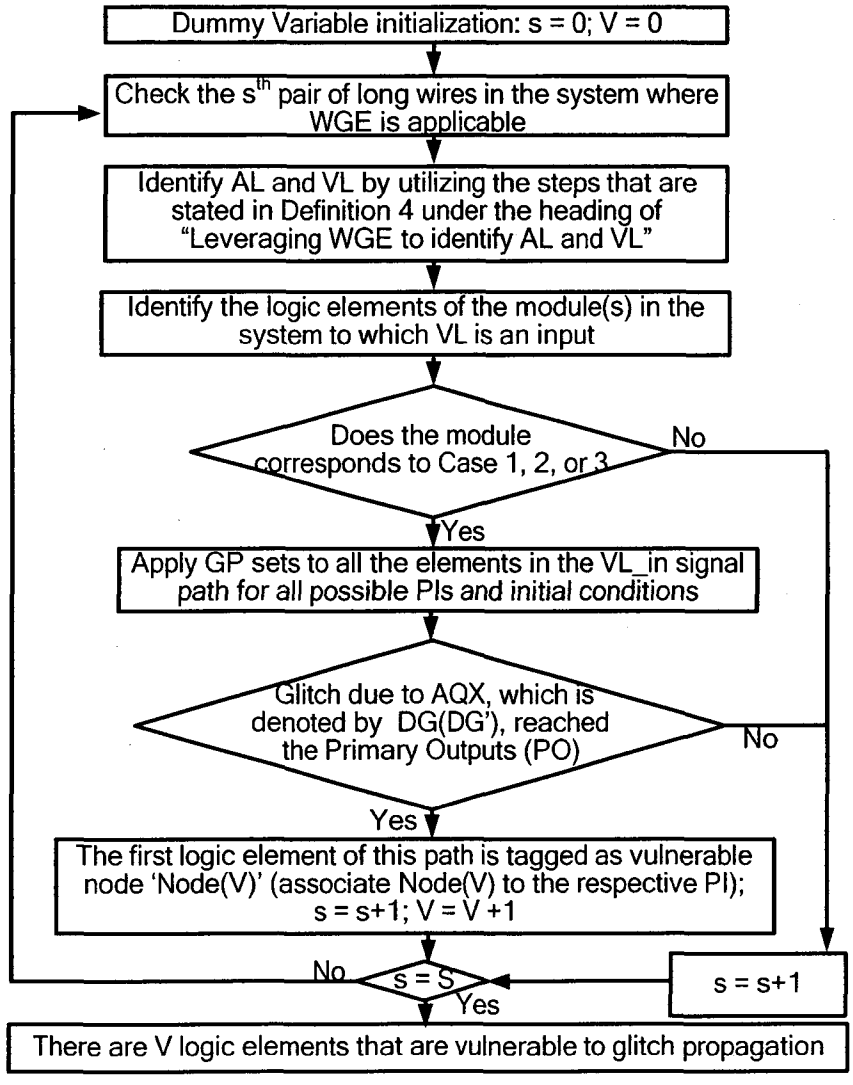


Figure 5.5. Flowchart of the Proposed Modeling Approach

5.4 Application of the Proposed Model

The proposed modeling scheme can be applied to any asynchronous interfacing scheme to identify the possibility of glitch propagation at the logic abstraction level. In this section, the usefulness of the proposed technique is illustrated by applying it on the same bundled data protocol based interface circuits as described in Chapter 4. It is imperative for establishing the validity of the proposed model that this analysis, using the proposed modeling approach, gives the same conclusions as obtained using the circuit level simulations which is elaborated in Section 4.4.2. This is demonstrated in the following analysis.

Application to the Bundled Data Protocol: Table 4.2 provides the Boolean equations describing the two ports of an interface implemented according to the bundled data protocol. For convenience, these Boolean equations for the two outputs of the Doutput port are reproduced here: $R_{il} = A_p + Den'Z1 + DenZ1' + Den'R_{il}Z0'$ and $R_p = Den' A_{il}A_p' Z0' + A_{il} A_p'Z0'Z1'$. It is shown in Section 4.4.2 that A_p is the only VL_in for the Doutput port, which is the sender module output. In this section, the proposed modeling technique is applied to this interface. Considering the first WGE insertion criterion, the WGE is applicable on one set of parallel wires, which consists of signals R_p and A_p. Now, the application of algorithm (Figure 5.5) identifies that R_p is the AL and A_p is the VL signal (as explained in the preceding chapter). A_p is an input (i.e. VL_in) to the logic gates that generate both Primary Output (PO) signals. Therefore, a separate analysis is conducted on both (R_{il} and R_p), the PO signals. This analysis, as the algorithm suggests, is used to predict the circuit behaviour due to the crosstalk glitches (with A_p as VL_in) at the

logic abstraction level for all the PIs and initial conditions. This is the same circuit that was described in Section 4.4.2 through circuit level simulations.

The R_{il} signal: The hardware implementation of the R_{il}-generating circuit, along with the WGE for the glitch-affected signal, A_p, is shown in Figure 5.6. From the truth table of the WGE, the condition for A_p (VL_in) to glitch due to the transition in R_p (AL_out) can be obtained. AL_out and VL_in make this circuit an example of case 1 of the corollary. The GP set of the OR gate suggests that the glitch may propagate through the OR gate of Figure 5.6 when the derived inputs (i.e. other than VL_in signal) act as sensitizing signals while VL_in is subject to DG'. Hence, this example can be further sub-classified as case 1-A, where the GP set inputs are VL_in and a sensitizing signal. For the next step in the algorithm (application of the GP sets for all possible PIs), the GP sets for the OR gate (the logic element to which VL is an input) is utilized. It is shown in Figure 5.2 that the set {(0, DG'), (DG', 0), (DG', DG')} will cause a glitch (composite logic DG') that propagates to the output of the OR gate. As the second part of this step, all the PI combinations are investigated for which the two derived inputs (i.e. the inputs other than the VL_in) of the OR gate can become "00" (the non-controlling value for OR gate). These PI vectors are found to be {Den, Z1, Z0} = {001, 110, 111}. These three vectors are obtained from classic digital design technique. Hence, it is shown that this modeling approach provides digital designers with the conditions that can lead to crosstalk glitch propagation.

Further inspection of the protocol under study [8] shows that, according to the Boolean equations of the state machine (Table 4.2), only vector '110' is allowed to occur in this protocol. The other elements are filtered out by the protocol itself. Hence, the proposed modeling technique allows identifying at least one possible scenario in which a crosstalk

glitch may propagate to R_{i1}. An experimental validation is performed for this case in Section 5.5.

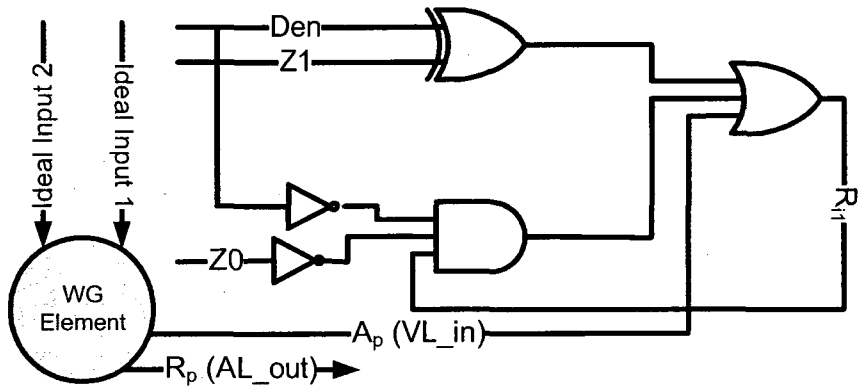


Figure 5.6. Hardware implementation for R_{i1} signal generation of the Doutput port, with conceptual realization of WGE

The R_p Signal: Figure 5.7 shows the hardware implementation of the R_p signal (the Boolean equation is shown in Table 4.2) along with the WGE. A similar analysis as exercised for the R_{i1} signal led us to identify A_p as VL_in and R_p as AL_out. It is seen from this figure that A_p (VL_in) is an input to both AND gates (after passing through an inverter). As the next step, it is observed from Figure 5.7 that this signal-generating circuit is an example of case 1-A of the corollary (where one of the inputs to the logic element is VL_in and the other is a sensitizing signal). The following step requires application of the GP sets for all the PIs. The GP sets for the AND gates (Figure 5.2) shows us that, when Den, Ai1, Z0, Z1= {1100, 0100}, then they act as sensitizing signals for propagating the composite value of DG'. Here, it is worth mentioning that the glitch occurrence that is shown in Figure 4.12b is for input vector (0100), which is identical to one of the vectors

predicted by the proposed modeling. Hence, it is demonstrated through the above analysis that the proposed model has correctly predicted the outcome of the AQX affected bundled data protocol at the logic abstraction level. An experimental validation is provided in the next section. In order to see the practicality of this result it is observed that, in the protocol provided in [8], both these conditions are possible.

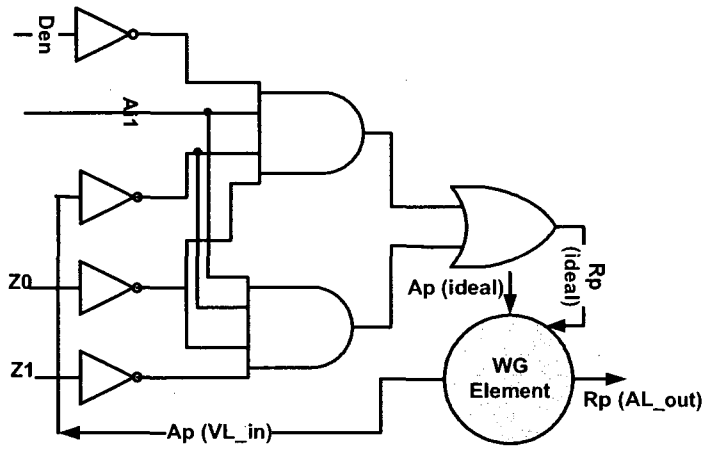


Figure 5.7. Implementation for Rp signal generation with WGE

Application to the DI Data Encoded Protocol: Knowledge of the WGE insertion criteria leads the designer to quantify the number of WGE elements. To provide an example with the DI data encoded protocol, we use a simplified version of the interface shown in Figure 4.8a. This simplified interface is shown in Figure 5.8. As it can be seen, there are two WGEs in this figure, one for each scenario depicted in Figure 4.8b. As a first step toward implementing our modeling approach, we need to know the AL and VL of the design. Here, the knowledge of the protocol plays a major role in understanding the nature of the aggressor and victim lines. Through the steps illustrated for "leveraging WGE to identify AL and VL", which is defined in the preceding section under Definition 4, it is observed

that both nx2 and nx3 may act as both the AL and VL at different stages of the protocol. Furthermore, the nx line can also act as an AL, which may inflict a glitch on the VL (nx3). In this protocol, VL(s), under all the conditions, are inputs to the C elements at the receiving end. Hence, this interface can be treated as an example of case 3, where the WGE1 depicts the case when both AL and VL are inputs to the receiver module. Whereas, WGE2 shows that the AL is an output to the receiver module and the VL is an input to the receiver module. Therefore, WGE1 can be considered, by utilizing the information of case 2-A and glitch propagation at WGE2, a case 1-A. Hence, both cases may contribute to the crosstalk glitch propagation and should be handled accordingly. Following the steps of the algorithm for each WGE, the GP sets are applied to the glitching output of both WGEs separately.

WGE1: According the protocol, when idle, out[0-3] are at logic '0' and nx[0-3] are at logic '1'. Therefore, the GP set for the inverted Muller 'C' element that applies to this case is {(0, DG), (DG, 0), (DG, DG)} → $t^- = 0$ and $t^+ = DG'$. Hence, if Ack = 0, this glitch may propagate.

WGE2: Again according to the protocol, it is known that nx rises only when one of the out[0-3] signal rises. The rise of out[0-3] is possible only when either nx2 or nx3 is at logic '0'. Hence, we can say that the glitch propagation is possible when one of the following two combinations of initial conditions according to the GP sets happen: (Out3=0 and ack=1) or (out3 =1 and ack=0).

The above examples show that this modeling approach is flexible and can predict the conditions of glitch propagation on different asynchronous interface circuits. As our modeling approach is dependent upon the GP sets that identify glitch propagation through

logic elements that are the constituent elements of the asynchronous interfaces, therefore, it is broadly applicable to various asynchronous designs.

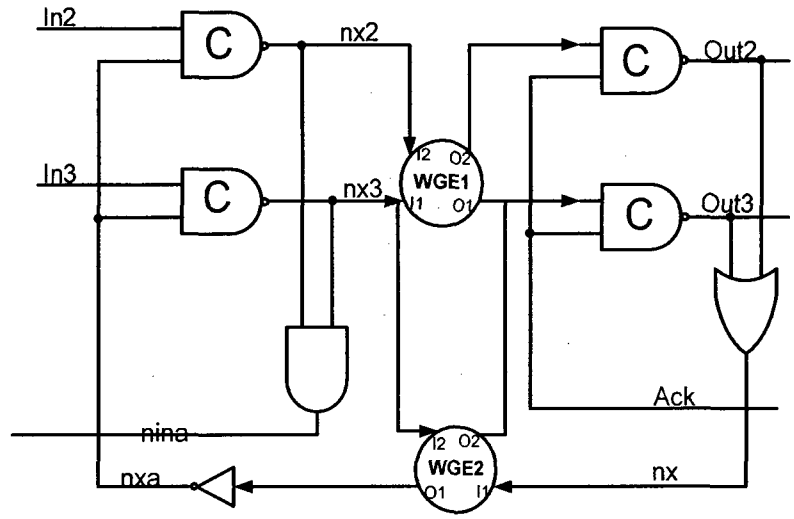


Figure 5.8. Hardware Implementation of the DI Data Encoded Interface with Conceptual WGEs

5.5 Experimental Validation

This section discusses the result of the experiments, which were performed to validate the proposed modeling method on the bundled data asynchronous interface. Hardware implementations of the R_{il} and R_p signal generating circuit, along with WGE for the glitch-affected signal, A_p, are shown in Figure 5.6 and Figure 5.7, respectively.

Glitches coming out of conceptual WGE in Figure 5.6 and Figure 5.7 are introduced utilizing the circuit shown in Figure 5.9 and synthesized with the Xilinx Synthesis Tool (XST) using the 'keep' attribute. The simulation results given in Figure 5.10 and Figure 5.11 show the back-annotated simulation results for an FPGA implementation, using a

Xilinx Virtex II-Pro (XC2VP30-7FF896), of the hardware design shown in Figure 5.6 and Figure 5.7, respectively.

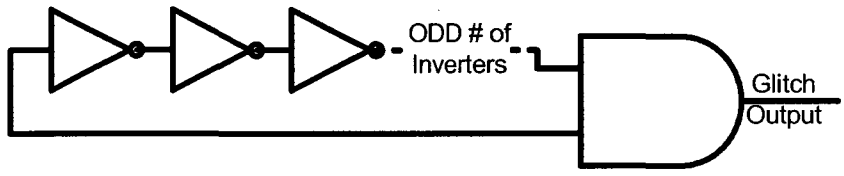


Figure 5.9. Circuit used for Simulating WGE behaviour at the logic level

FPGA implementation of Ri1 signal: Figure 5.10 shows back-annotated simulation results for the R_{i1} signal generating hardware shown in Figure 5.6. This simulation is performed for both conditions, with and without the glitches for the same input vector, in order to show the usefulness of the proposed model. The left circle in Figure 5.10 shows that, if there is no glitch, then R_{i1} (represented as ri in Figure 5.10) is not asserted for the input vector Den, Z1, Z0 = "110". The right circle shows the converse case, that is, when the circuit is subject to glitch (A_p is VL_in) for the same input vector, then this glitch is propagated to the primary output R_{i1}. The Count value shows the number of transitions in R_{i1}. This result is in agreement with the analysis provided in the preceding section.

These implementations are carried out at the hardware as well, utilizing an FPGA board provided by the Xilinx University Program. For the input vector {"110"}, count_val (measuring the number of assertions in R_{i1}) is observed for both, with and without glitch, scenarios. The results were found to be in compliance with the simulation results.

FPGA implementation of Rp signal: Figure 5.11 shows the back-annotated simulation results for the FPGA implementation of the circuit shown in Figure 5.7. The right hand side of Figure 5.11 shows that, when {Den, A_{i1}, Z0, Z1}= {(0100)}, a glitch in the A_p

signal (VL_in) (labeled as glitch_output) is propagated to R_p thus forcing it to do a false transition from logic 1 to 0. This figure shows that the results that were obtained through the transistor level simulation (and shown in Figure 4.12b) are reproduced here at the logic abstraction level, using the proposed modeling approach. Similarly, the transitions on the left side of Figure 5.11 illustrates that R_p signal does a false transition for the case {Den, A_{i1}, Z0, Z1} = {(1100)} as well.

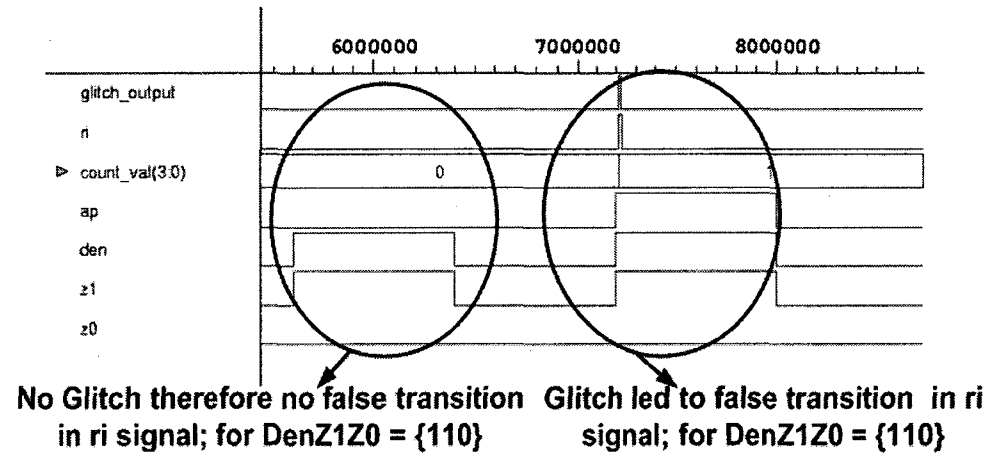


Figure 5.10. Virtex II-Pro back-annotated simulation results of the design shown in Figure 5.6

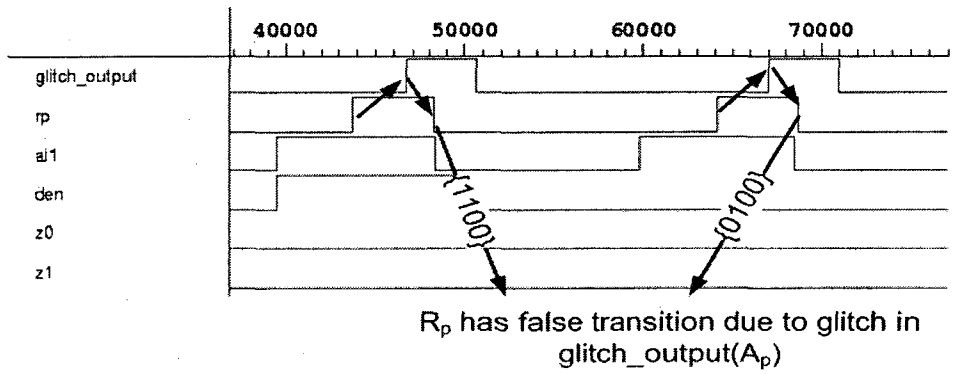


Figure 5.11. Virtex II-Pro back-annotated simulation results of the design shown in Figure 5.7

5.6 Summary and Discussions

In order to understand AQX effects at a higher abstraction level, a glitch propagation modeling technique is proposed in this chapter. This model identifies the possibility of intrinsic crosstalk glitch propagation in asynchronous handshake interfaces at the logic abstraction level. This modeling technique is applied to one of the asynchronous protocols under study to predict the glitch propagation possibility. This approach successfully predicted the conditions under which this interface may propagate glitches. The results obtained through the proposed modeling technique were experimentally validated with an FPGA implementation on Xilinx Virtex II-Pro XC2VP30-7FF896. It is seen that the same glitch behaviour is obtained at the logic abstraction level, through the proposed modeling technique, as was obtained at the transistor level simulations. This is a first work on a framework that allows representing possible behaviour of a logic structure, for asynchronous handshake scheme, in the presence of crosstalk glitch without the help of circuit level simulations. This framework can be used as a step toward formalizing the asynchronous circuit behaviour under crosstalk glitches. Furthermore, it is utilized in proposing solutions to stop such glitches from propagating to the primary output, which is the subject of next chapter.

Chapter 6: Crosstalk Glitch Quenching Solution

In the previous chapter, a crosstalk glitch modeling technique is proposed that provides a framework, which allows representing the possible behaviour of a logic structure for asynchronous handshake schemes in the presence of crosstalk glitches without the help of circuit level simulations. This chapter leverages the knowledge of this crosstalk glitch modeling in asynchronous handshake schemes, and proposes a novel solution called crosstalk glitch gating, to design crosstalk glitch tolerant asynchronous design schemes. The crosstalk glitch gating proposed in this chapter provides a pool of design modifications, which can be applied, as the need arises, for different classes of asynchronous circuits as characterized in the previous chapter that are, subject to crosstalk glitches.

Conventional solutions for crosstalk glitch elimination are usually implemented at the detailed layout stage of the design flow [74], [68], [69]. In this chapter, a complete design methodology is proposed to implement crosstalk glitch gating in asynchronous handshake schemes. This methodology can be introduced as early as at the logic- synthesis stage of the design flow. This design methodology may also require some iterations at the back-

annotated STA (Static Timing Analysis) stage, for globally routed designs, which is still at an earlier stage compared to the detailed routing stage.

The proposed methodology is implemented on representative circuits of two broad classes of asynchronous handshake schemes: the bundled data protocol based schemes [8], [19], [45] and the data encoded delay insensitive schemes [45], [55]. It is shown that, for the same interconnect length, the proposed method leads to designs that efficiently block the crosstalk glitch propagation.

In the next section, a pool of crosstalk glitch gating solutions is proposed for each case of asynchronous handshake schemes as characterized in the previous chapter. The subsequent section describes the method to implement the proposed crosstalk glitch gating solutions. Section 6.3 demonstrates the implementation of this methodology on two representative circuits of widely used asynchronous handshake schemes. Simulation results validate the design modifications.

6.1 Crosstalk glitch Gating: Crosstalk glitch Quenching Solution

Crosstalk glitch gating is a technique in which glitch propagation is quenched by introducing control signals. These control signals are generated using some determinism or pseudo-determinism (described later on in this section) on AL signals. This section introduces the crosstalk glitch gating technique proposed to prevent crosstalk glitch propagation in all the cases of the corollaries in the preceding chapter.

6.1.1 Case 1-A & 1-B

It is assumed in the previous chapter that all the logic elements are reduced to two input elements. Here, we are keeping the same assumptions. Here it is assumed that, Y represents an input other than VL_in to the logic elements, 'Cnt_val' defines the control value of the gate [73], [95], and ⊗ is a generalized symbol for any logic operator. From the GP sets shown in Figure 5.2 it can be seen that if the value of Y does not have a 'Cnt_val' then, according to the GP sets, the glitches to the input propagate to the output: Provided glitches reach the threshold limit, which is ascertain by the criteria of WGE insertion. This glitch propagation condition can be defined as follows: if Y ≠ Cnt_val then, from the GP sets, it can be concluded that Y is a sensitizing signal. Thus, it allows the glitches on VL_in to propagate. Consequently, Y ⊗ DG' (DG) produces either DG (DG') or Y ⊗ DG' (DG) produces DG' (DG) depending on the logic gate behaviour. Following paragraphs elaborate on a crosstalk glitch gating technique to block glitches for this case.

To Block G (G') for only one type of glitch event, DG or DG'

Because, in this case, AL_out is deterministic (Lemma in Chapter 4), therefore, Δt_G can be estimated using timing libraries. In order to block glitch propagation, it is required to introduce control signals for the Δt_G duration. Different gate implementations are required for each GP set. Below, an example GP set for the AND gate is analyzed. A similar analysis can be performed for the other GP sets shown in Figure 5.2.

Let us consider the GP set of the AND gate: [{(1, DG), (DG, 1), (DG, DG)} → DG]. In such a case, G' is propagated instead of logic '1' for the Δt_G duration. This problem is resolved by an additional logic element that can keep the logic '1' at the output of the

AND gate for Δt_G duration through a crosstalk glitch gate control signal. The implementation of this crosstalk glitch gating solution is shown in Figure 6.1b. Figure 6 summarizes the crosstalk glitch gating solutions for the rest of the two input GP sets shown in Figure 5.2.

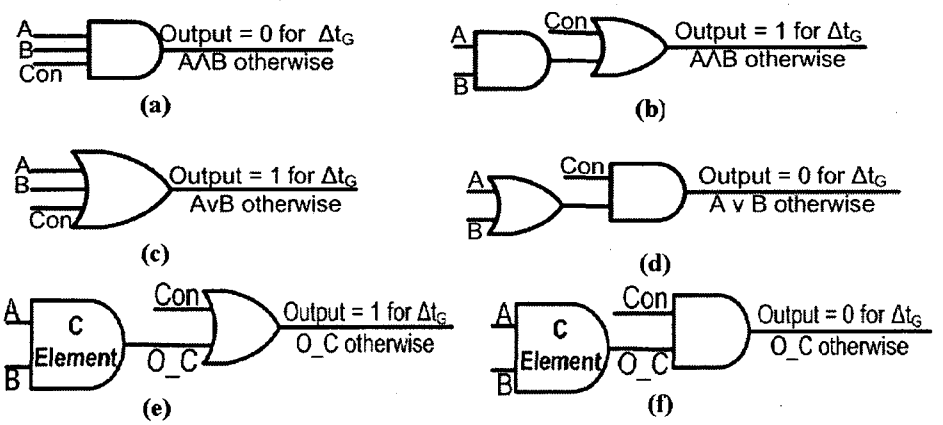


Figure 6.1. Glitch gating solution for: (a) and (b) are the solutions for GP sets ofthe AND gate with outputs DG' and DG, respectively. (c) and (d) are the solutions for the GP sets of the OR gate with outputs DG and DG', respectively. (e) and (f) are the solutions for GP sets of the Muller 'C' element with outputs DG and DG' at t^{+2}, respectively.

To Block G (G') for both types of glitch event

Figure 6.2a, and b describe the crosstalk glitch gating solutions for the case when a Boolean logic equation within an asynchronous module is subject to both types of glitch events, DG and DG' at different time instances. The order of occurrence is irrelevant in this case, as long as the crosstalk glitch gate controlling signals, 'Con', are raised properly for their respective logic elements. The two glitch durations are represented as Δt_{G1} and Δt_{G2} in the figure. Here, Δt_{G1} is the glitch duration when VL_in experiences G'. Similarly

---

[2] The t^+ notation is associated to the asynchronous logic, see [6 , chapter 10] for details.

Δt_{G2} is the glitch duration when VL_in experiences G. Figure 6.2a illustrates that the combination of Figure 6.1a and b results in a circuit that can block both types of glitch propagation through AND gate. A similar solution is introduced for an OR gate by combining Figure 6.1c and d (not shown in Figure 6.2).

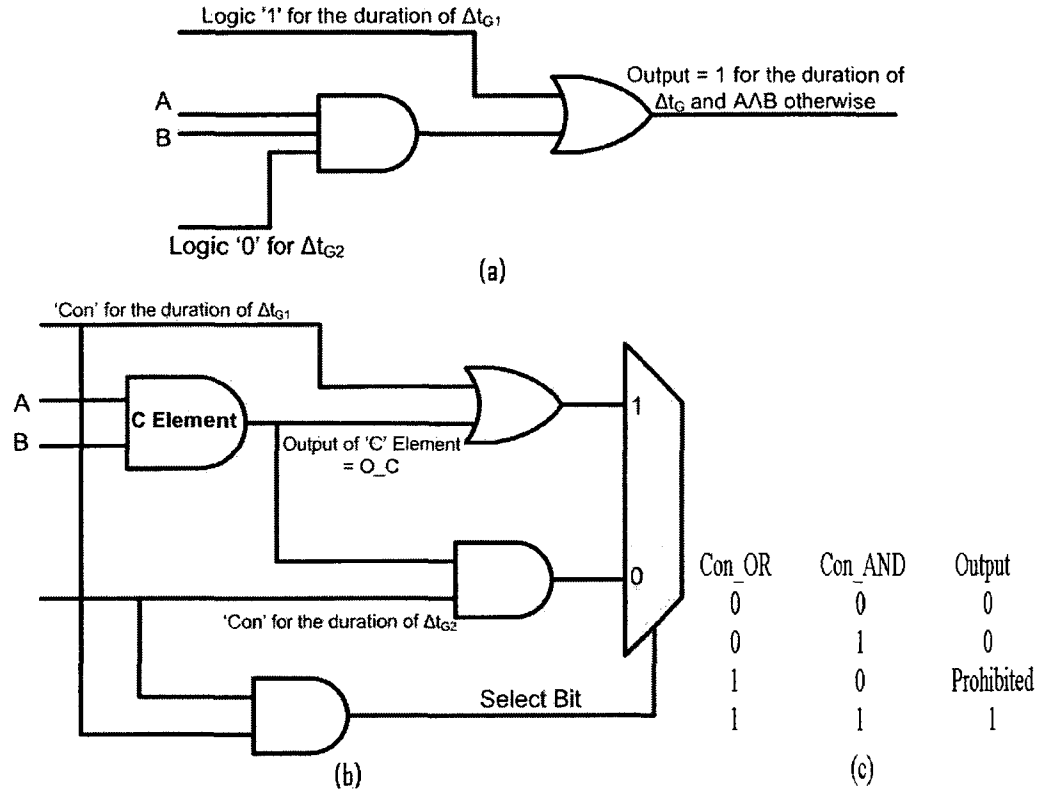


Figure 6.2. The solutions if both the GP sets, with outputs DG and DG', exist for the same logic element (a) shows the solution for AND gate, (b) shows the solution of Muller 'C' element (c) shows the truth table for the asserting the select bit for the Multiplexer.

Figure 6.2b shows a circuit that rectifies such a glitch condition in a Muller 'C' element. A multiplexer is introduced in this circuit, in addition to combining the solutions described in Figure 6.1e and f, to filter the glitch value. The select bit is dependent on the 'Con' values for the OR and AND gates. When both the Con_OR and Con_AND are idle, (i.e. they are respectively at logic level '0' and '1'), then the multiplexer may choose any

input. For such a situation, the implementation shown in Figure 6.2b and c chooses bit '0'. Note that it is prohibited to have both the 'Con' bits asserted concurrently.

Case 1-C:

As explained in Chapter 5, this case contains a feed back signal derived from AL. The generalized hardware implementation is shown in Figure 5.3c.

To Block such Glitches: If the 'GP sets' sensitization, which may be for any type of logic element, depends upon the AL feedback signal, then, the return path of the AL signal should be delayed by a Δt_G duration in order to curb the glitch propagation. E.g., it is illustrated in Figure 6.3 that one of the inputs of the OR gate is VL_in signal and another input is derived from AL_out, which is feedback to the same asynchronous module. This derived signal acts as a sensitization signal to the OR gate, as shown in Fig 6.3. In this case, by delaying the sensitization signal, which is dependent on AL_out, for Δt_G duration, the glitch propagation can be blocked. An example of such a case is discussed in length later on in this chapter in Section 6.3.2.

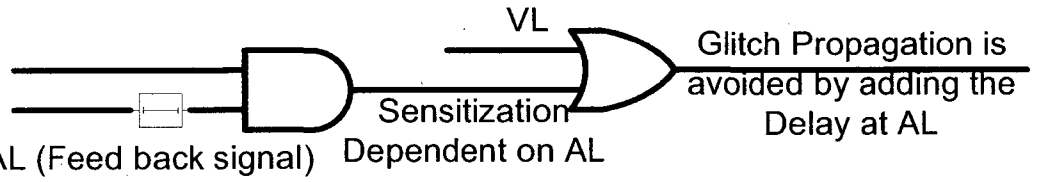


Figure 6.3. The solution for the case when AL_out is fed back to a logic element in the module. This example has OR gate as logic element

6.1.2 Insertion of Crosstalk glitch Gate Control Signal6, 'Con', during Δt_G (Δt_{G1} and Δt_{G2})

This section discusses the generation and insertion of the crosstalk glitch gate-controlling signal during Δt_{G1} and Δt_{G2}. This phenomenon can be divided into two sub-cases one for the NRZ (Non-Return-to-Zero) and another for RTZ (Return-to-Zero) signalling schemes.

NRZ Signalling Scheme: This scheme produces fewer transitions than RTZ in a data transaction; hence, it is more robust in a crosstalk glitch sensitive system. As shown in Figure 6.4, each data transaction starts with a transition in the 'En' signal, denoted by '1st (2nd) Data ready' in Figure 6.4, which in turn asserts the handshake signals. Hence, AL_out is always dependent on the 'En' signal, which makes it deterministic. Here, it is appropriate to mention that 'En' cannot be dependent on a VL transition because G (G') in VL is a reactionary process to T (T') in AL. Leveraging the fact that crosstalk glitches have bounded delay, the crosstalk glitch gate controlling signal, 'Con', can be generated with the transition in the 'En' signal and should remain asserted for the duration shown as 'Cnt. Duration' in Figure 6.4 to stop the propagating glitch in the VL_in signal.

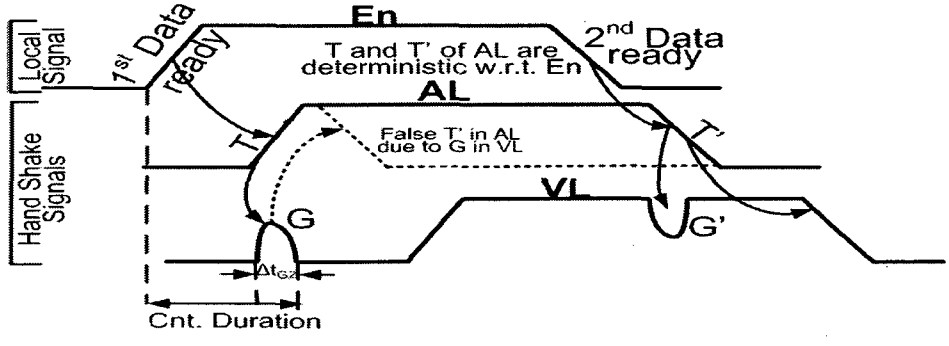


Figure 6.4 NRZ Signaling Scheme

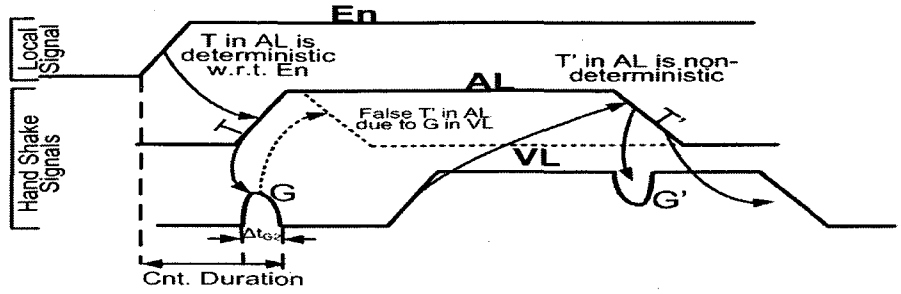


Figure 6.5 RTZ Signaling Scheme

RTZ Signalling Scheme: This scheme is more vulnerable to crosstalk glitch effects because both AL and VL perform two transitions in a data transaction. Figure 6.5 depicts this scenario; it is worth mentioning that Figure 6.5 represents one data transaction and Figure 6.4 shows two data transactions. This is due to difference in signalling scheme. There are at least two scenarios of glitch propagation for each data transaction in RTZ signalling scheme shown in Figure 6.5. One scenario in such designs (the one at the beginning of the protocol) is identical to what is discussed in NRZ scheme, and is dealt with in the same way. In the second scenario (when AL_out performs T') it is highly likely that this transition (T' in Figure 6.5) in the AL_out signal is directly dependent on the VL_in signal. In such cases, it is required to introduce some dummy signal, which can reduce the dependence of the AL_out signal on the VL_in signal and helps in generating the corresponding 'Con' signal. The insertion of additional signals also requires reformulation of the logic equations of the state machine. Section 6.3.1 elaborates one such case with a detailed analysis.

6.1.3 Case 2-A

Let a system follow case 2-A (as defined in Chapter 5), and comply with the RTZ signalling scheme. The RTZ scheme is chosen because, in the previous section, it is observed that this scheme is more cumbersome to deal with, in terms of crosstalk issues, compared to single transition NRZ scheme. Due to the un-deterministic nature of AL_in (refer to Lemma), the crosstalk glitch gating technique cannot directly be implemented. To apply this crosstalk glitch gating technique, some sort form of determinism is required for the AL_in signal. To obtain this determinism, signals may be modified so that AL_in is produced in response to an output signal from the asynchronous module. Such modified AL_in signals are called as pseudo-deterministic signals. The following theorem is proposed to make the AL_in signal pseudo-deterministic.

Theorem

In the case when the asynchronous module is subject to **AL_in** $\wedge$ **VL_in**, whereas the inputs to the glitch propagating logic gate are: VL_in and a sensitization signal (case 2-A), H, then the theorem states: *If both AL and VL are function of the same sensitized signal (say H) then the glitch propagation can be temporally shifted by changing the initial logic level of one of the signal lines.*

Assumptions: Let 'H' be the common sensitizing signal. The initial stable state of the system assumes that (1) AL_in and the potential VL_in are at the same logic level, and (2) H is not a controlling value 'Cnt_val'.

Proof: According to the Lemma in Chapter 5, AL_in is non-deterministic. Since in this case AL and VL are inputs to the module, it is also possible that the same line may act as both at different time instances. Therefore, the VL notation is treated differently in this

analysis and called Potential VL or PVL. The previously stated assumptions can be defined as follows:

(1) AL_I = PVL_I (i.e. AL and PVL are initially at the same logic level, '1' or '0')

(2) H ≠ Cnt_val

Due to the RTZ scheme, AL_in and PVL_in are subject to two transitions in a single data transaction. Let AL_I(T) represent the transition, T, and similarly AL_I(T') represent the transition, T'. Similar definition leads to PVL_I(T) and PVL_I(T'). If one of the input signals to the two different (two-input) logic elements in a given asynchronous module is PVL_in or AL_in and the other one is the common sensitizing signal 'H', then, to avoid crosstalk glitches due to coupling capacitance, the following conditions are specified in asynchronous protocols:

AL_I(T) $\wedge$ PVL_I(T) = False; i.e. these transitions are mutually exclusive (I) and

either

(AL_I(T) $\otimes$ H) produces T or T' at the output of the logic element (because, as stated earlier, H ≠ Cnt_val)

or (PVL_I(T) $\otimes$ H) produces T or T' at the output of the other logic element (here it is noticed that, for this case, the glitch may propagate only if PVL glitches toward G) (II).

But, it is also possible that AL_I(T) leads to glitches, which can be expressed as:

$G_{PVL\_I} \rightarrow$ AL_I(T) (i.e. G_{PVL_I} is true only when AL_I(T) is true, the arrows in this analysis follows the discrete mathematics' definition)

this implies that: DG' $\rightarrow G_{PVL\_I}$

which allows the glitch to propagate, as shown below;

(AL_I(T) $\otimes$ H) produces T or T' **and**, concurrently,

DG' $\otimes$ H produces DG or DG' (following the relevant GP set)

A similar analysis proves that glitches may propagate during the second transition, T', as well, thus, we conclude that, if AL_I and PVL_I are having same initial logic level, then crosstalk glitch may propagate, under the above explained scenario.

Now, as stated in the theorem, if the first assumption is changed as follows:

AL_I(AL_I') = PVL_I' (PVL_I) (i.e. initially AL_in is at logic '1' ('0') and PVL_in is at logic '0' ('1')),

then, the two conditions of the protocol change as follows:

AL_I(T) $\wedge$ PVL_I(T') = False (i.e. the mutually exclusive transitions are opposite in polarity) a

And

either (AL_I(T) $\otimes$ H) produces T or T' at the output of the logic element

or (PVL_I(T') $\otimes$ H) produces T or T' at the output of the other logic element (here it is noticed that, for this case, a glitch may propagate only if PVL glitches toward G').

A similar analysis that has been performed for the former case, leads to the following: $G_{PVL\_I'}$ does not occur when AL_I(T); this is because PVL_in is initially at the same logic level where AL is transiting toward. But $G'_{PVL\_I'}$ may occur when AL is making T' transition, as here the two signals are initially at the same logic value, this is mathematically written as follows,

$G'_{PVL\_I} \rightarrow$ AL_I(T'), which implies that DG $\rightarrow G_{PVL\_I'}$

This allows the glitch to propagate, as shown below;

DG $\otimes$ H produces DG or DG'

As the glitch is propagating at AL_I(T') instead of AL_I(T), therefore, it can be said that the glitch has been temporally shifted with the change in the logic levels because it happens toward the terminating end of the handshake scheme.

Crosstalk glitch Gating Solution to Case 2-A

Though this theorem cannot directly solve the problem of glitch propagation, it helps in resolving it by making AL pseudo-deterministic. This theorem shows that the first transition of AL (i.e., AL_I(T)) becomes glitch free but this change makes the glitch temporally shifted to the later transition of the AL, (i.e. AL_I(T')), where AL_I(T') is generated in response to an output signal (usually an acknowledge signal). Thence, it makes the AL_in signal pseudo-deterministic, and allows a gate control signal to be generated to avoid any potential glitch propagation through the sensitizing signals.

Let us now describe the usefulness of this theorem in the presence of sensitizing signal 'H'. Considering the assumption of the theorem, crosstalk glitch gating is applied as such that it negates signal H as soon as the AL_I(T) transition is over. This negation should start before sending the acknowledge signal, if this acknowledge is part of the protocol. H should remain negated, through crosstalk glitch gating, at least up until Δt_G duration after AL_I(T'). Therefore, it is made sure that the signal H does not have a sensitizing value during the transition on AL_I(T'), which causes the potential VL (PVL) to G', denoted as G'_{PVL_I}. Because, H is not at a sensitizing value during Δt_G with the application of this theorem, therefore DG⊗H' blocks the glitch propagation. An example of such a case is discussed in section 6.3.1.

6.1.4 Case 2-B

In the context of asynchronous handshaking schemes, only data encoded DI systems may have AL_in and VL_in (input) to a particular module. Therefore, it is known to the designer whether the data encoding technique permits the specific AL and VL to make concurrent transitions. By exploiting this knowledge, the designer may seek the solution to such a case using data error detection techniques [67]. Such correction techniques are not explained in this text as it requires a separate analysis that is beyond the scope of this study.

6.1.5 Case 3: (AL_in ∧ VL_in) ∧ (AL_out ∧VL_in)

Case 3 is basically a union of case 1 and case 2. Therefore, it is resolved by treating each case separately. The corresponding analysis, as explained in preceding subsections, is applied to each sub-case. The order in which these analyses should be used depends on the order of occurrence of these problems. In section 6.3.2, an example of such a case is explained.

6.2 A Method to Introduce Crosstalk Glitch Gating in Asynchronous Handshake Schemes

This section describes the proposed method to introduce crosstalk glitch gating solutions for making 'asynchronous handshake schemes' crosstalk glitch tolerant. This

method utilizes the theory developed in the preceding sections to curb the triggering of false events caused by glitch due to AQX. This solution can be introduced early in the design cycle, i.e. at back-annotated gate-level synthesis stage as shown in Figure 6.6, thus, providing faster time-to-market. The steps of the proposed method are defined as follows:

Step 1. Identification of wires that run the longest distance in parallel: these are the signals that are most sensitive to crosstalk glitches. This identification should be made through understanding the asynchronous protocol rather than waiting for detailed layout of the system. Usually, these are handshaking signals that connect the two interfacing modules.

Step 2. Utilize knowledge of the protocol and the definitions of WGE and DG(DG'), introduced in the preceding chapter, to distinguish between AL and VL. Conclude about the direction of AL, i.e. whether it is an inbound signal or an outbound signal (AL_in or AL_out), which is a requirement in the Lemma (Chapter 5) to find out the determinism of AL signal. This step is implementable at an earlier stage of design flow, because it requires only the understanding of the behavioural description of the asynchronous protocol.

Step 3. Leverage the GP sets to find out the possibility of crosstalk glitch propagation in most glitch affected signals, which are obtained in step 1. This step requires the knowledge of gate-level logic synthesis of asynchronous state machines.

Step 4. Depending on the direction of AL, found in step 2, and its susceptibility to propagate crosstalk glitches, observed in step 3, conclude which of the discussed three cases corresponds to the asynchronous system under study. Implement the pertinent crosstalk glitch gating solutions to the corresponding cases as explained in the preceding section.

Step 5. Perform the circuit adjustments required to accommodate the changes needed for implementing the solution(s) suggested in step 4. This step may require iterations as it initially introduce the AQX blocking circuit based on library cell and wire delay model to ensure that the gating covers the required time interval (Δt_{G1} or Δt_{G2}). The need to readjust the proposed circuit implementation based on back-annotated wire and gate delay values (obtained after global routing) may arise.

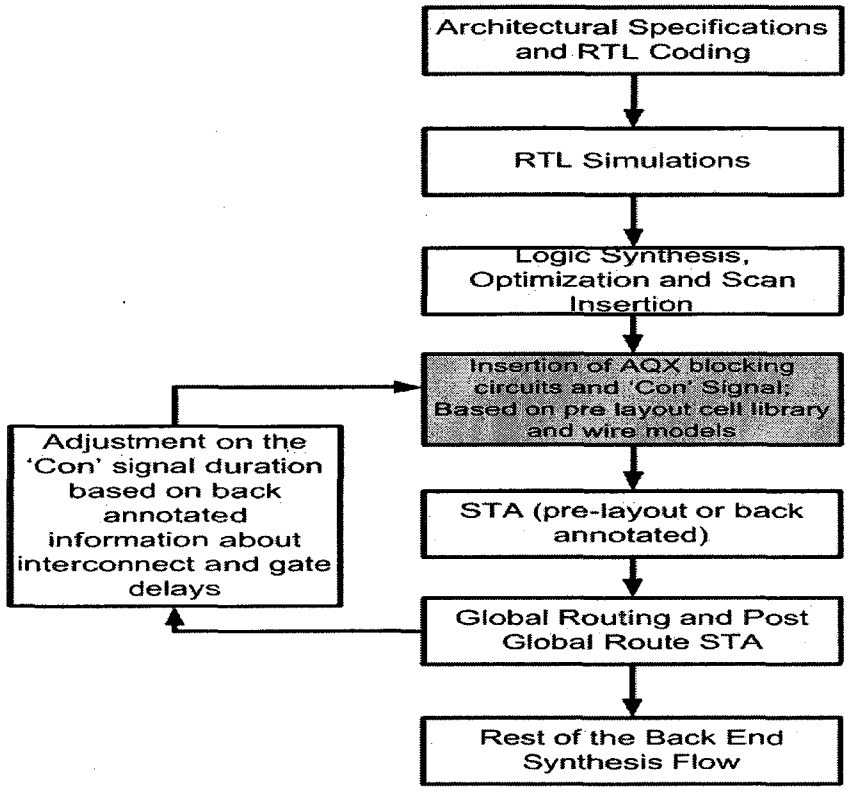


Figure 6.6 **Graphical depiction of when to insert the AQX blocking circuit in the conventional design flow**

6.3 Application of the Proposed Method on Representative Asynchronous Handshake Schemes of Two Different Classes

This section applies the proposed method to the representative circuits of two popular classes of asynchronous handshake scheme: bundled data and delay insensitive.

6.3.1 The Bundled Data Asynchronous Interface

This section applies the proposed method to the conventional bundled data protocol based design [8], which is introduced in Chapter 4 and further elaborated in Figure 6.7, to obtain crosstalk glitch tolerant asynchronous interface. The steps of the method yield the following results:

Step 1: In Figure 4.11, A_p and R_p are identified as the most crosstalk glitch sensitive signals. This is because these two handshaking signals run in parallel between the two communicating blocks, which is the longest distance a signal traverses in such an interface. **Step 2:** It is found, by inspecting the expected waveform of the protocol, shown in Figure 6.7, that T (T') in R_p precedes T (T') in A_p. Hence, it is concluded that Rp is an AL and A_p is a VL. The state transition diagram shown in Figure 6.7 indicates that A_p is an input to the sender module, i.e. VL_in, and R_p is an output of the sender module, i.e. AL_out. Only the sender module is analyzed for AQX glitch propagation, as the receiver module does not have an inbound VL signal or VL_in.

TABLE 6. 1. Equations for Asynchronous State Machine for Conventional and proposed Bundled Data Protocol

Asynchronous State Machine in Conventional Design	D output Port : *Ri* = Ap + Den' Z1 + Den Z1' + Den' Ri Z0' *Rp* = Den' Ai Ap' Z0' + Ai Ap' Z0' A1' *Z0* = Den'Ap + Ai Z0 *Z1* = Den Ap + Den Z1 + Ai'Z1 P input port: *Ri* = RpRi + Pen' Rp Ti *Ap* = Ai *Ti* = PenAi + Ai'Ti+Ri'Ti (Ti is not affected by AQX, therefore not discussed further)
Asynchronous State Machine in Proposed Design	**Modified D output Port:** R_i = A_p + Den'Rct + DenRct' R_p = Den'AiRc2t + DenAiRct' *Dmy* = DenAp + Ap'Dmy + Rp'Dmy **Delay State Machine:** *Rct* = Dmy Rct + Den Rct + Ai' Rct + Dmy Den Ai Z0' *Z0* = Ai' Rct + Ai Z0 + Rct Z0

Step 3: Signal A_p, which is the VL in this system, is an input to all the four Boolean equations of the state machine shown in the first row of Table 6.1. Utilizing the GP sets, it is seen that all these equations may potentially allow glitch propagation. To elaborate, the conditions for propagation of G in A_p in all the different signals are illustrated. Z0 propagate G in A_p, when Den = logic '0', this may happen during state transition 6 to 7 (in the state diagram of Figure 6.7). Z1 propagates crosstalk glitches due to G in A_p when Den = logic '1', this may happen during the state transition from 1 to 2 in the state transition diagram (Figure 6.7). R_p may propagate crosstalk glitches due to G in A_p when Den, A_{i1}, Z0 and Z1 = {0,1,0,0}. Likewise, R_{i1} may propagate crosstalk glitches due to G in A_p when Den, Z1, Z0 = {000, 001, 110, 111}. Luckily, this protocol filters all the possible glitch-propagating vectors for R_{i1} except for 110 (where Z1 may become a subject to glitch propagation due to G in A_p, which is further explained in the next paragraph). A

similar analysis can be done to find out vectors for propagating crosstalk glitches due to G' in A_p.

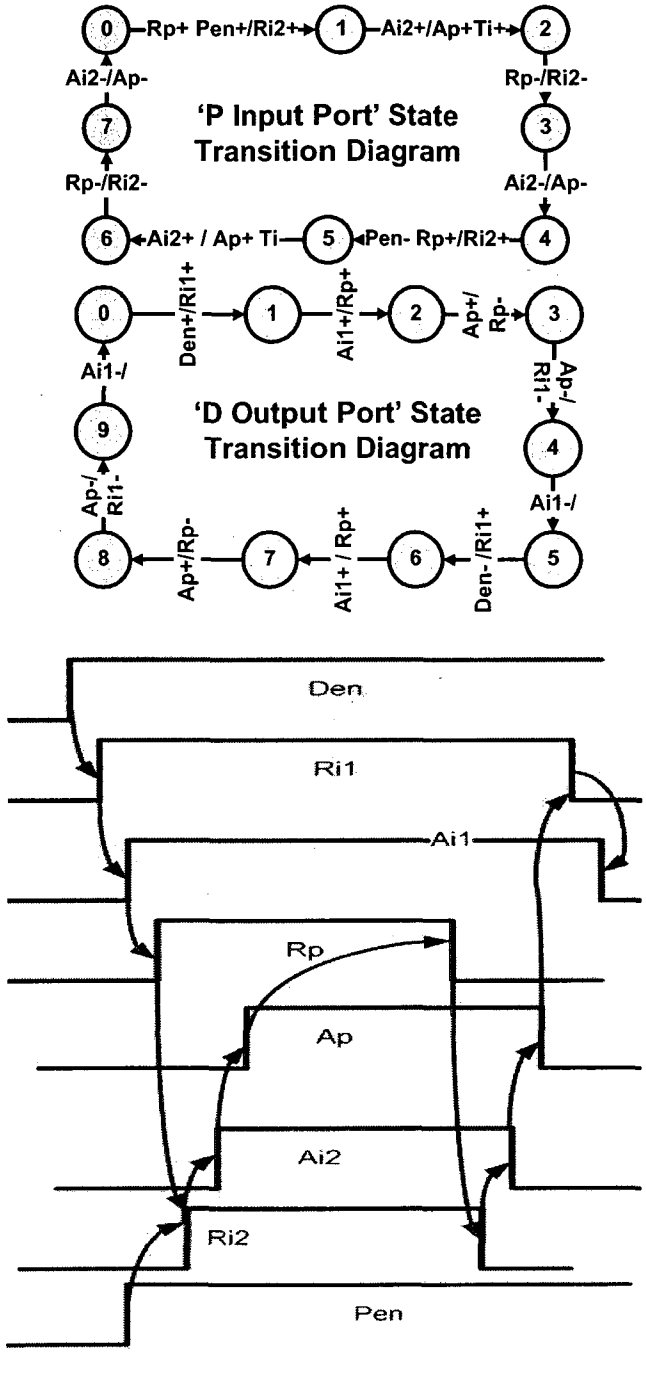


Figure 6.7 **State Trasition Graph (aboveleft) and Expected waveform (right(below) for the Conventional Bundled Data Protocol**

To find out that, whether, these crosstalk glitch propagations lead to the malfunctioning of the protocol or not, an extrapolation of one of the scenarios is performed here. Suppose that, during state transition 1 to 2 in Figure 6.7, G occurs in A_p and, hence, it may cause, R_p to make a false T' (following equation of R_p in the first1st row of Table 6.1). Also, this glitch in A_p causes Z1 to rise. The equation of Z1 reveals that it has a property to hold its value. When A_p settles back to logic '0', the equation for R_{i1} (given in Table 6.1) reveals that this signal will go to logic '0' (because Den is logic '1' and Z1 is falsely equal to logic '1'). As T' on R_{i1} leads to the termination of the protocol, therefore, it can be said that such glitches lead to premature termination of the handshaking protocol. A glitch in A_p also leads to false data latching, due to the fact that the A_p signal is used as a data-latching signal (shown in Figure 4.11).

Step 4: This step introduces crosstalk glitch gating to stop the system malfunctioning elaborated in the implementation of step 3. The expected waveform, provided in Figure 6.7, shows that there are two state transitions, 1 to 2 and 6 to 7, when crosstalk glitch is inflicted upon A_p (i.e. VL_in). Transition 1 to 2 indicates that T in R_p introduces G in A_p during Δt_{G2}. Similarly, the duration of the glitch that occurs during state transition 6 to 7 is indicated by Δt_{G1}. As the AL_out, which is R_p for this interface, is not a feedback signal, therefore, this asynchronous interface belongs to case 1-A of the corollary. Application of the proposed solution for case 1-A requires identification of the signal transitions that may be utilized to generate the crosstalk glitch gate-controlling signal, 'Con', to avoid glitch propagation. Due to the use of the RTZ signalling scheme, both transitions, T and T' of AL have to be analyzed. It is observed from Figure 6.7 that, during transition from state 1 to 2, T in R_p is caused by a local signal (local to D output port state machine), A_{i1}, which

is analogous to En in Figure 6.5. Therefore, for Δt_{G2}, an auxiliary signal can be derived from the transition, T, in A_{i1}. The state diagram in Figure 6.7 also shows that, during the other state transition (6 to 7), T' in R_p is the sole signal transition to cause G' in the VL_in signal, A_p. Hence, here dummy signals are required to generate the crosstalk glitch gate-controlling signal to block the glitches during Δt_{G1}. These dummy signals introduce additional states in the D output port state machine, which is shown in Figure 6.8. These additional states are inserted between states 2 to 4, and 8 to 10, as shown in Figure 6.8a. The new signals are called Dmy and Rct signals, which are controlled by a new state machine called "delay state machine", shown in Figure 6.8b. The P input port state machine remains the same. The corresponding state machines were synthesized using the 3-D synthesis tool [65], [66].

Step 5: ***Crosstalk glitch Gating Control Signal Generation***– The modified D output port state machine equations, based on step 4, are shown in row 2 of Table 6.1. These modified equations show that the VL (A_p) is the input to the two Boolean equations. These equations generate the Dmy and Ri1 signals.

For the Dmy Signal: Figure. 6.8a illustrates that a transition on the Ai1 signal, while changing state from state 1 to 2, leads to a transition, T, on AL (R_p). This produces G in the VL_in (A_p). According to the Boolean equation for Dmy, in Table 6.1, glitch (G) in A_p forces a G in Dmy for the duration of Δt_{G2}. Therefore, the first control signal, Con_AND in Figure 6.8c, is generated at the rising edge of A_{i1} for a Δt_{G2} duration to avoid glitch propagation through Dmy signal. Furthermore, Dmy signal experiences a glitch, G', during the state transition from 7 to 8. Therefore, the auxiliary signal Con_OR is introduced, shown in Figure 6.8c. The required circuit is similar to Figure 6.2a, except for

the fact that, here, the crosstalk glitch signal is associated with an OR gate rather than with an AND gate.

Glitch Gating for R_{i1}: With the introduction of the delay state machine, the transition, T', in AL (Rp) is made dependent on any transition on the Rct signal through a delay state machine. The Boolean equation of R_{i1}, given in the second row of Table 6.1, shows that the R_{i1} signal is sensitive to G' in the VL_in signal (A_p). R_{i1} may propagate G' in A_p when the state machine of Figure 6.8a is in transitions from state 3 to 4 and 9 to 10. Therefore, the gate control signal, shown as 'Con' in Figure 6.8d is generated through Rct for duration Δt_{G1}. This signal ensures that no glitch is propagated through the R_{i1} generating circuit. G in VL_in does not affect R_{i1} during Δt_{G2}, as it is not solely dependent on VL_in signal (A_p) for this interval.

Simulation Results: Figure. 6.9 shows the transistor level simulation results for the conventional and proposed bundled data asynchronous handshake scheme for same interconnect length, 7.5 mm, under the 90nm STMicroelectronics CMOS technology. Although conventional designs malfunctioning starts at 2 mm, as seen in previous chapter, but here more than 3 times extended length is chosen to test the sustainability of the proposed solution under severe operating conditions. Interconnect parasitics are obtained using the data sheet provided from STMicroelectronics [64]. All these simulations utilize the 5Π model, due to its effectiveness in the reduction of lumped model inaccuracies to 3% [3]. In Figure 6.9a, the false negation of the R_p signal is caused by the high glitch magnitude on A_p, which is acting as VL_in for this module.

Figure 6.9b shows that, by the implementation of crosstalk glitch gating, the modified design functions properly under the same design constraints, as there is no occurrence of

false transition. Hence, the proposed crosstalk glitch gating technique provides a solution to design flawless asynchronous bundled data protocol based designs under the conditions of interconnect at which the conventional designs failed to operate correctly.

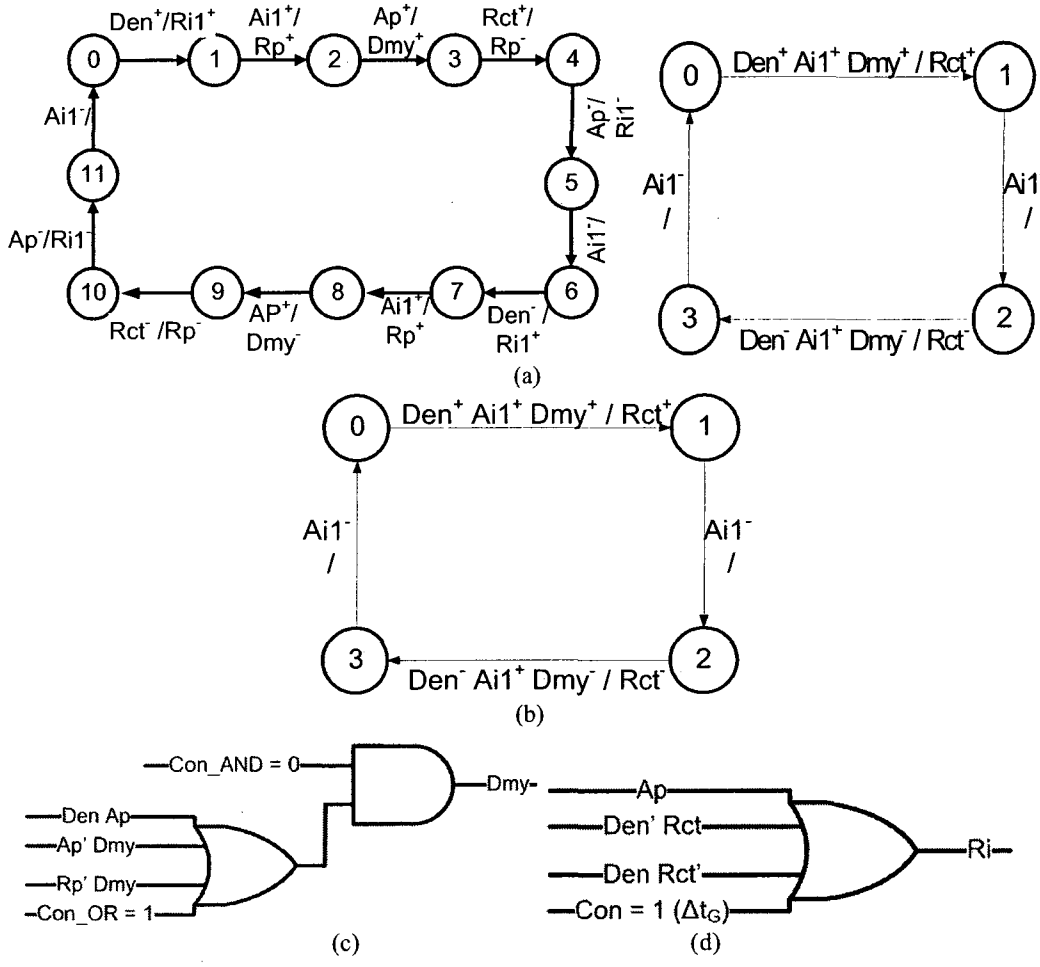
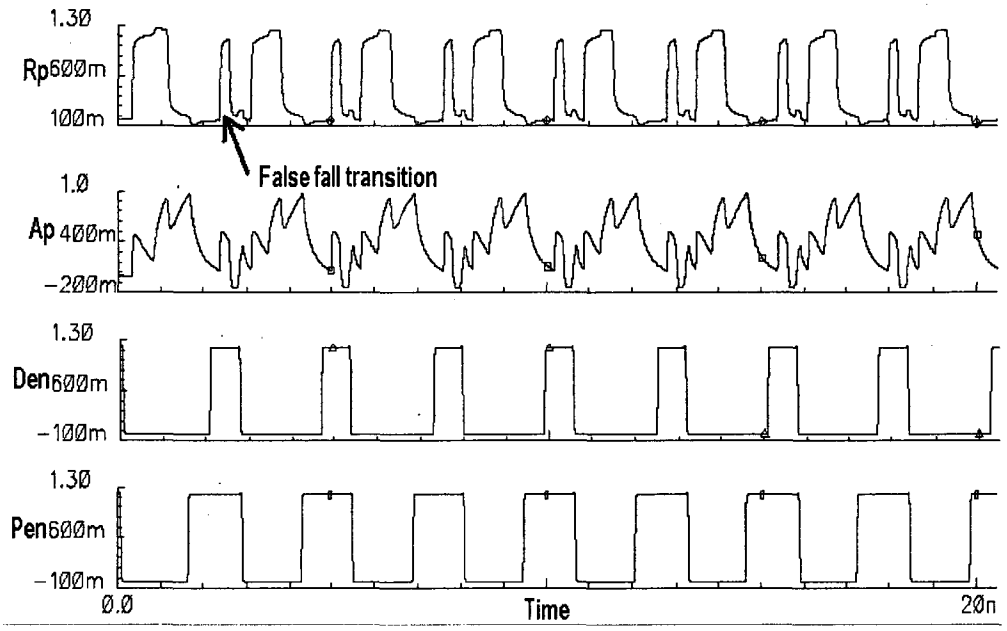
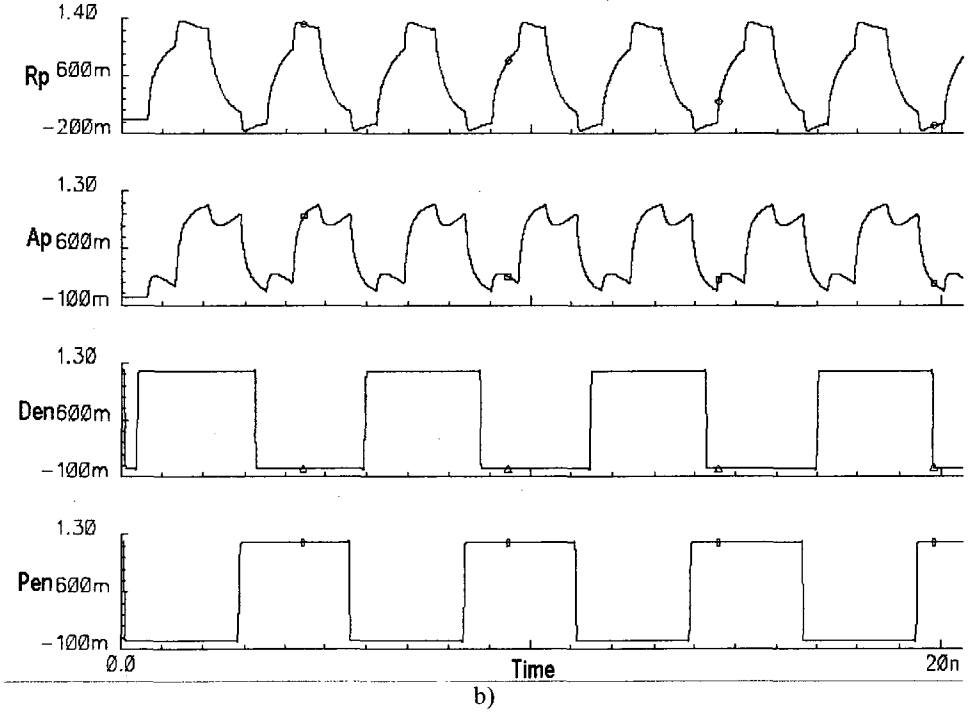


Figure 6.8 STG for the Proposed D output Port of the Bundled Data Protocol (b) STG of the Delay state machine (c) Crosstalk glitch gating for Dmy signal. (d) Crosstalk glitch gating for Ri signal.



a)



b)

Figure 6.9 Transistor level simulation results (a) for conventional design (b) for modified design (crosstalk glitch gating implemented)

6.3.2 Proposed 1-of-4 Data Encoded DI Asynchronous Interface

This subsection shows the implementation of the proposed crosstalk glitch gating method on the 1-of-4 data encoded DI asynchronous interface.

Step 1: The wires that run the longest length in this interface are nx[0-3] and nx (because they link the two sides), as shown in Figure 4.8a. Hence, these are designated as the most crosstalk glitch prone signals.

Step 2: To identify the direction of AL and VL, it is necessary to understand the mechanics of this interface. The following description, of Figure 4.8a, shows the scenarios of AQX occurrence during the normal circuit operation. The nx0 to nx3 group of signals are initially at logic '1'. This group of signals is represented as nx[0-3]. Similarly, in[0-3] represents the in0 to in3 group of signals and out[0-3] represents the out0 to out3 group of signals. According to the 1-of-N DI data encoded protocol, of which 1-of-4 is a specific case, only one of the input lines of in[0-3] can go high at a given time. There are two scenarios of crosstalk glitch occurrence in this design with respect to the receiver module and they are, shown in the expected waveform of the conventional design in Figure 4.8b.

First Scenario: If any of the lines in the nx[0-3] group of signals, which rises initially at logic '1', performs a transition T' then its neighbouring line within the same group experiences glitch G'; e.g. if nx3 does a transition T' then its neighbouring line nx2 experiences glitch G', as shown in Figure 4.8b. Hence, nx3 is acting as an AL and nx2 as a VL in this case. Such a situation relates to the case 2-A as both AL and VL are inputs to the receiver module.

Second Scenario: The interface shown in Figure 4.8a may also experience glitch, G, in

nx3 due to T in nx. This occurs during the handshaking termination phase, when any of the out[0-3] group of signals makes a transition toward logic '1'. In this situation, nx is the AL which is an output from the receiver module end and nx3 is the VL which is an input to the receiver module. Hence, this scenario is an example of case 1-A. The second scenario appears twice in Figure 4.8b, one for each logic level transition in nx line.

These two glitch scenarios are illustrated with arrows in the expected waveform, shown in Figure 4.8b. The existence of these scenarios makes this design an example of the third case of the corollary. where (AL_in $\wedge$ VL_in) $\wedge$ (AL_out $\wedge$VL_in) = True. Figure 4.8b suggests that the first scenario belongs to case 1, whereas the second scenario belongs to case 2. As stated in the preceding chapter, these two cases are treated separately in order to quench the glitches.

Step 3: This step shows that whether the crosstalk glitches propagate into the interface or not. It can be noticed here that all the Muller 'C' elements have an inverted output in Figure 4.8a. Also, initially, all receiver-end Muller 'C' elements are sensitized via the Out_Ack signal. Therefore, if any of the nx[0-3] signals produces G' due to transition T' in an adjacent line then, the GP sets of Muller 'C' element suggests that the glitch propagates, provided that the output of the C element at t^- in Figure 6.2, is at the initial condition of logic '0'. Similarly, the second scenario of glitch propagation may take place through the respective receiver end Muller 'C' element if Out_Ack signal is at logic '1' when nx[0-3] glitches to G, due to T in nx.

Step 4: While identifying the direction of AL and VL in step 2, it is concluded that this design requires a two-stage procedure to resolve the crosstalk glitch issue: one stage each for case 2-A and case 1-B. Step 3 has shown that these glitches haves the potential to

propagate into the interface, therefore, in this step, the proposed solutions are applied to resolve each of the two cases.

Application of the proposed solution for Case 2-A: Application of the theorem results in alternating the polarities of nx[0-3], as shown in Figure 6. 10. These changes are labelled as 1st in Figure 6.10; e.g. initial condition of nx[0-3] has changed from '1111' to '1010'. The following explanation illustrates the effects of such changes.

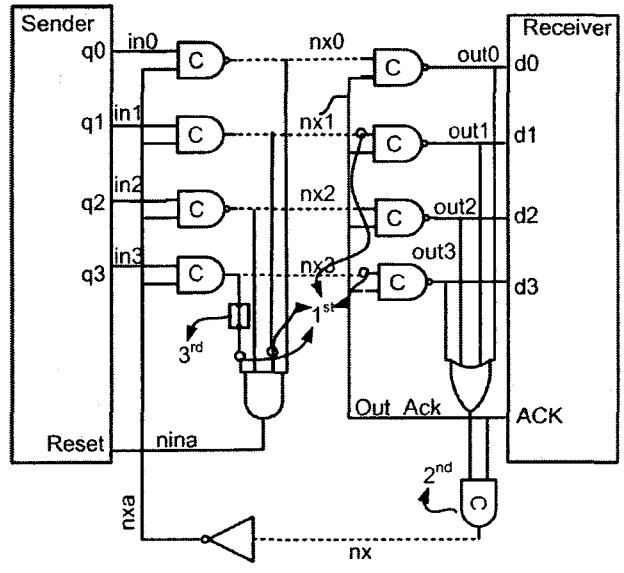


Figure 6.10 Proposed Hardware implementation of 1-of-4 Data Encoded DI Asynchronous Interface

In contrast to the conventional design, now, when nx3 does a transition, T, it does not inflict a glitch in nx2. Rather, during the termination phase, i.e. pseudo-deterministic phase, of the handshake, due to RTZ signalling, if nx3 does a transition T' then nx2 experiences a glitch G'. The modified expected waveform, shown in Figure 6.11, demonstrates the temporal shift of the crosstalk -glitch toward the terminating end of the protocol. Out_Ack is the sensitizing signal that causes the glitch to propagate in this

interface via C elements. It is seen from the protocol that, once Out_Ack is utilized by nx3 to generate out3, it is not necessary to keep its value to the same logic level during the termination phase of that particular data transaction. Therefore, if Out_Ack signal is reversed after the assertion of out3 and before the inversion of nx3 then the glitch propagation is blocked for the first scenario. Also, another implicit advantage of this solution is that the glitch occurrence due to the second scenario in Figure 4.8 is also eliminated. On the other hand, as a consequence, another glitch of the same type (case 1-B) has appeared at a different place in the circuit. This is caused by the introduction of the polarity reversal on nx3, which causes a glitch on nx, as shown in Figure 6.11. Solution to this new occurrence of the glitch is provided in the following section.

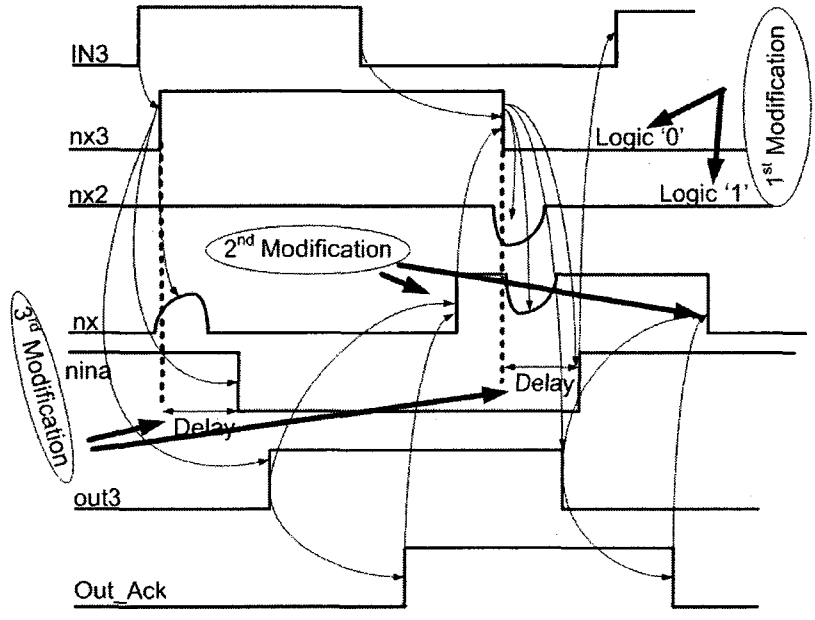
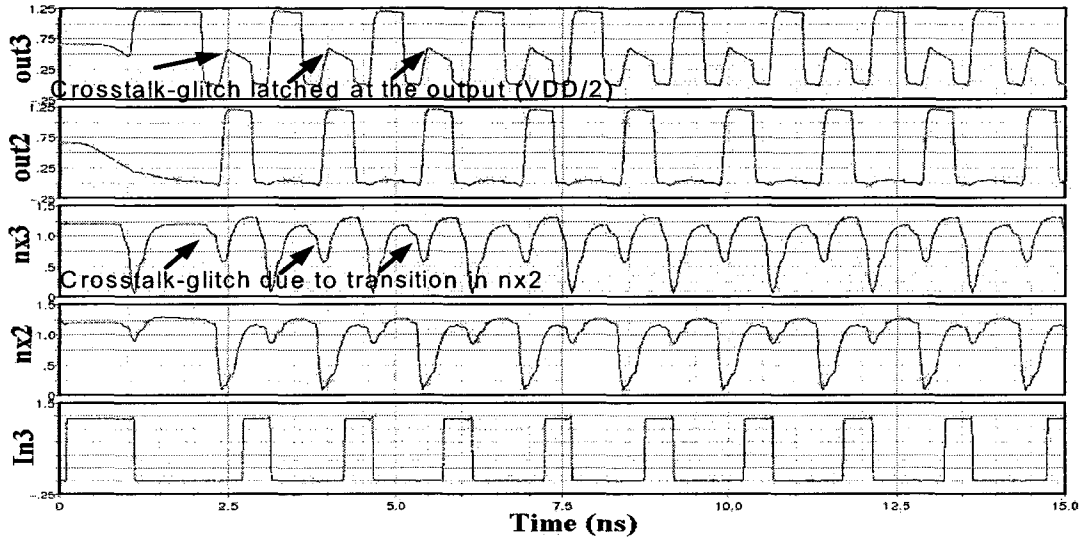


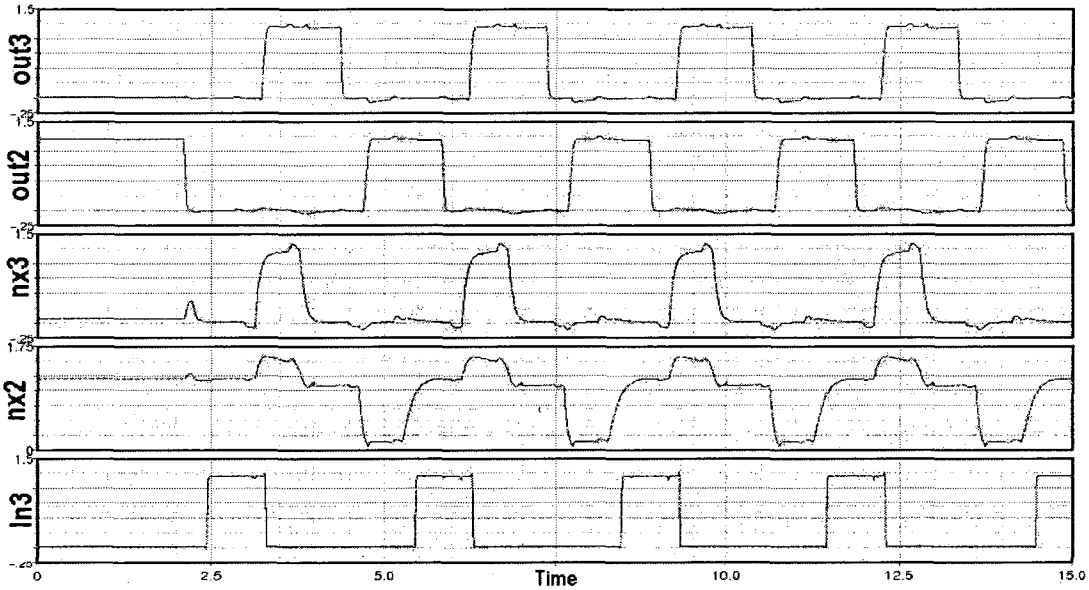
Figure 6.11 Expected Waveform of the Proposed Hardware implementation of 1-of-4 Data Encoded DI Asynchronous Interface

Application of the proposed solution for Case 1-B: As explained above, such a design choice introduces the potential for glitch propagation at the sender end. This can be seen by glitches in nx in Figure 6.11. The glitch in nx due to transitions in nx3 is an example of case 1-B of the corollary, where nx acts as a VL_in signal to the sender module. Here, glitch propagation is dependent on the feedback signal from the AL, which generates the 'nina' signal in Figure 6.10. Application of the solution to such a case, as shown in Figure 6.3, suggests the introduction of the delay element at one of the inputs of the AND gate generating the nina signal.

Step 5: Finally, the hardware implementation of the circuit, based on the modification of the design, is explained here. The modified design is shown in Figure 6.10. There are three modifications done in this design and are labelled as 1^{st}, 2^{nd}, and 3^{rd} in Figure 6.10: 1) The nx[0-3] signals now have alternating polarity because due to half of the output bubbles in the Muller 'C' element at the sender end that have been shifted to the inputs of the Muller 'C' element at the receiver end and to the inputs of the AND gate (that generates the 'nina' signal), in order to keep the same Boolean relations. 2) According to the theorem, the first modification introduces a temporal shift in the glitch and this shifted glitch is barred from propagation through an additional Muller C element (shown in the bottom of Figure 6.10). (This modification ensures that the sensitizing signal is inverted before the termination of the RTZ signalling scheme begins). 3) The third modification is to the nx signal, which becomes a VL when its adjacent nx[0-3] line is asserted. Therefore, a delay is introduced to the nina signal generation, as explained in step 4. These adjustments in the hardware implementation affect the expected waveform and are indicated by arrows, which represent respective modifications, in Figure 6.11.



(a)



(b)

Figure 6.12 Transistor- level simulation results for conventional design (above) and for modified design where crosstalk glitch gating is implemented (below)

Simulation Results: Figure 6.12 shows the transistor level simulation results of the conventional and modified (with the implementation of crosstalk glitch gating) 1-of-4 Data Encoded DI asynchronous handshake schemes. Both designs are simulated for the same interconnect length, 1.5 mm, in the 90nm STMicroelectronics processing

technologies [64]. It is seen in the conventional design that crosstalk glitch may be latched at the output as can be seen in the topmost signal of the simulated waveform of Figure 6.12a. The proposed design obtains a cleaner waveform and avoids crosstalk glitch propagation. Figure 6.12b shows that the proposed design avoids latching any glitches for the same interconnect length. Hence, it is experimentally shown that the proposed crosstalk glitch gating technique makes this asynchronous handshake schemes tolerant to crosstalk glitches.

6.4 Summary and Discussions

A comprehensive set of possible cases in asynchronous handshake signalling schemes that may lead to crosstalk glitch propagation was formulated in the previous chapter. Leveraging the knowledge from the previous chapter, a novel set of design solutions, called crosstalk glitch gating, is proposed to resolve crosstalk glitch propagation on a case-by-case basis. A detailed methodology is developed to analyze the AQX effect on asynchronous handshake schemes and to implement the proposed crosstalk glitch gating technique. Implementation of the proposed methodology on two representative asynchronous handshake schemes, of different classes, is performed. It is demonstrated through transistor- level electrical simulations that the proposed method generates crosstalk glitch gate controlling signals, which effectively stops glitch propagation. This work is a step towards dealing with glitch sensitivity as part of logic design and synthesis as opposed to dealing with the issue as part of the physical design steps.

While comparing with conventional crosstalk glitch solutions it is observed that conventional solutions usually deal with a wire sandwiched between the two transitioning wires. For example, bus encoding techniques [6], [23] or behavioral-level crosstalk detection methodology [24] for sequential asynchronous circuits check for intrinsic transition faults in sandwiched wires only. On the other hand, the proposed technique solves the problem of AQX glitches in asynchronous handshake schemes. Therefore, an objective apple-to-apple comparison is not possible. Still, there are some measures that can distinguish our technique. For instance, conventionally brute force buffer insertion is used for signal integrity but this is done much later in the design flow, usually post-place–and-route. Our method is advantageous because it is introduced earlier in the design cycle. The automation level that is possible with our solution is also comparable to that of the buffer insertion technique. With only basic knowledge of the asynchronous protocol, designers may obtain the desired glitch tolerant circuit. As it is implemented only at the places where there is a potential for glitches to propagate, therefore it reduces overall area and power overhead in comparison to brute force buffer insertion or Triple Modular Redundancy (TMR) with phase shifting. An objective comparison with conventional designs obtained twice the performance of the conventional Bundled Data designs and more than 20% performance improvement is observed in Delay Insensitive designs.

Chapter 7: Skew Tolerant Synchronous Interface for High-Performance Point-to-Point Communication

It is mentioned, in the first chapter, that this thesis addresses issues related to modern DSM technologies in both synchronous and asynchronous design domains. In the preceding chapters, significant reliability issues in modern DSM technologies to implement asynchronous interfaces have been discussed. This and the following chapter discuss the challenges of communicating modules in MCD through synchronous interfaces when they are designed in modern DSM technologies.

It was established in Chapter 3 of this thesis that high-performance clocking of intellectual properties (IP) modules, within a skew budget, is becoming difficult in modern DSM technologies. It has been seen in Chapters 4 to 6 that, to communicate between two modules in MCD, asynchronous interfaces may be utilized. At the same time, the design community has investigated different methods in synchronous design paradigms. Synchronous interfacing methods often require PLL based synchronization, which requires phase correction that consumes useful bandwidth and mixed-signal components. Another synchronous design methodology from Edman et al., in [78], is the Synchronous Latency Insensitive Design (SLID) method. This design technique can accommodate large clock skews, but the maximum achievable clock period of the IP modules is a function of

the clock skew, which consequently limits the throughput of the system. This technique also suffers from the control and data signal delay mismatch problem [80], which is a severe constraint in modern DSM technologies, especially for long interconnects.

This chapter contains the description of a new class of solutions that can be utilized for communications between two modules running at the same frequency or at different frequencies (Chapter 8) that have non-aligned clock phases. This novel and all digital synchronous design method for point-to-point communications uses two stages of n interfacing registers and a locally delayed clock with phase adjustments. This design is free from synchronizers and clock-data delay mismatch problems. Moreover, the communicating modules run at frequencies which are virtually independent of the clock skew. A comprehensive case-wise mathematical analysis is also provided to facilitate design automation for synthesizing such designs as standard cells. It is assumed in this study that worst-case timing information is available *a priori* for the interfacing system that helps in quantifying the worst-case phase offset among the communicating blocks. This assumption is in line with the stat-of-the-art techniques presented in recent literatures [76] – [78].

It is assumed, in this work, that there are at least two IP modules, a sending and a receiving IP module, in each system. Sending and receiving modules are referred to as terminating modules, unless otherwise stated. Terminating modules are supplied with the clock from the same clock source, but with a certain phase difference between each other. This phase difference is due to skew in the CDN. It is further assumed that the maximum skew bound is known. Links in the system are assumed to be subject either to positive or

negative skew and a comprehensive mathematical treatment is performed on all the possible sub-cases of timing constraints.

Contribution of the Work described in this chapter: The proposed design leverages larger bus width (as wires are usually abundant in on-chip systems [16]) to alleviate the problems in inter-module communications. By careful adjustment of the interfacing clock (slow) phases and through knowledge of maximum skew bound, higher skew tolerance is achieved.

The work described in this chapter provides the following contributions to the current literature:

1) Introduction of the new design concept of higher-bandwidth multi-stage interfacing registers to overcome clock skew of far-apart (with respect to relative difference in clock signal delay from the clock source) modules: The shortcoming of the state-of-the-art solutions in fulfilling the requirements for far-apart terminating modules in MCDs, to run at a speed independent of the clock skew and of highly unpredictable clock-data delay mismatches, instigated this work.

2) A comprehensive mathematical analysis in an attempt toward making the proposed design scheme synthesizable as standard cells: this analysis is beneficial to design automation.

This chapter is organized as follows. Section 7.1 presents an overview of the fundamental higher-bandwidth scheme to accommodate large skews. Sections 7.2 and 7.3 utilize the concepts introduced in Section 7.1 for designing interfaces for two special cases: identical or integer multiple frequencies between the communicating modules. A complete mathematical analysis of the proposed interfacing method is performed in this

section along with a summary of the advantages achieved by the proposed designs. The next section provides a general discussion about the advantages of the proposed design schemes. Section 7.5 discusses the simulation setup and results as well as a comparison with conventional synchronous designs. Section 7.6 explains the prototype implementation of the proposed design using a Virtex-II Pro FPGA from Xilinx. Section 7.7 summarises this chapter.

7.1 Concept of Wider Bus Width

The requirement of interfacing registers to better utilize a CDN is well established in Chapter 3 of this thesis. It is apparent from the discussions in that chapter, and from the elaborations on equation 3.11, that, usually, the initial splits in H-tree work at a higher frequency, and the fastest speed at which modules may communicate in MCD is usually lower than the speed with which each terminating module can function individually. This is due to the delay variations in CDN designed in modern DSM technologies, which give rise to clock skew. The design proposed in this chapter leverages larger bus width (as wires are usually abundant in on-chip systems [16]) to alleviate the problems in inter-module communications. Using the concept of larger bus width and by careful adjustment of the interfacing clock phases, and benefiting from the knowledge of maximum skew bound, higher skew tolerance is achieved.

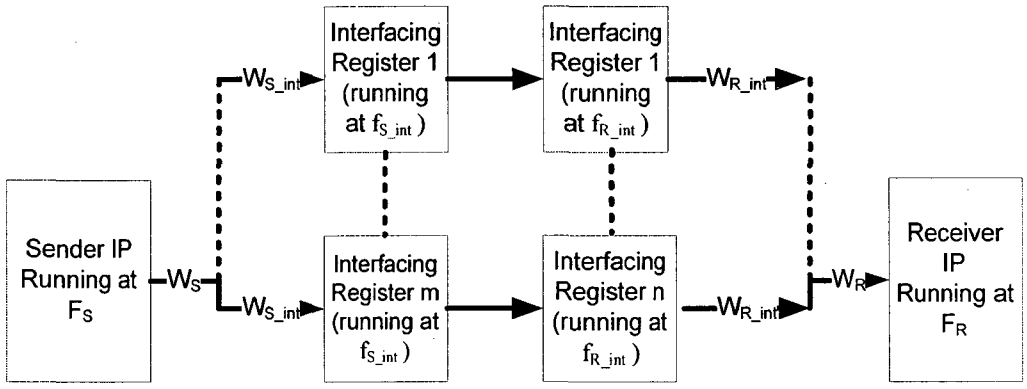


Figure 7.1. Elaboration of two IP modules with interfacing registers

What is Larger bus width: This design scheme allows any two terminating modules that have known maximum phase discrepancy, to communicate through slow but wide (or parallel) stages of interfacing registers, and that is why it is given the term larger bus width in this thesis. In Figure 7.1, the concept of larger bus width is introduced for two mutually asynchronous IPs. These IPs are running at frequencies F_S (sender frequency) and F_R (receiver frequency), with their bus width being W_S and W_R, respectively. Consequently, the time periods of the sender and receiver modules are denoted by $T_1=1/F_S$ and $T_2 = 1/F_R$, respectively. It is assumed that the clock skew limits the sender and receiver interfacing registers to run at frequencies of f_{S_int} and f_{R_int}, respectively, to accommodate the setup timing constraint. In this scenario, the only requirement of such designs to function properly is to follow equation (7.1), which expresses the fact that the sustained bandwidth of all stages of the interface, in Figure 7.1, must be the same.

$$F_S W_S = W_{S\_int} f_{S\_int} m = W_{R\_int} f_{R\_int} n = F_R W_R \qquad (7.1)$$

where m and n, denote the number of interfacing registers of widths W_{S_int} and W_{R_int} for the sending and receiving ends, respectively. This general formulation can represent cases where m and n are related by a fractional ratio. A simpler situation would be the

case where that ratio or its inverse is an integer. For example if $m=2n$, the sender would operate at twice the speed of the receiver. However, in this chapter, we will first focus on the situation where $m=n$ and then generalize to integer ratios. Means to support fractional ratios is investigated in the next chapter.

In contrast to the design proposed by Edman et al. [78], this design does not use a strobe signal, rather, based on *a priori* timing information, phases of the interfacing registers are adjusted to accommodate phase discrepancies. Avoiding the use of a strobe signal solves the control signal and data delay-mismatch problem. The proposed technique introduces two stages of interfacing registers (Figure 7.1). These registers act as FIFO stages and absorb the clock skew, hence they allow the maximum clock period of the IPs to be independent (virtually) of the clock skew (mathematically explained in the next section). These registers also use multiple phases of the clocks to manage bandwidth while alleviating clock-skew problems in a manner that is similar to previous work done by our research group [94]. The following section proves the usefulness of this concept of higher bandwidth for interfacing the modules in MCD.

7.2 Proposed Interface for $m=n$: Leveraging Higher Bandwidth

This section describes how to use higher bandwidth to provide reliable communications among modules that share the same clock frequency but varying phases, called mesochronous schemes [79].

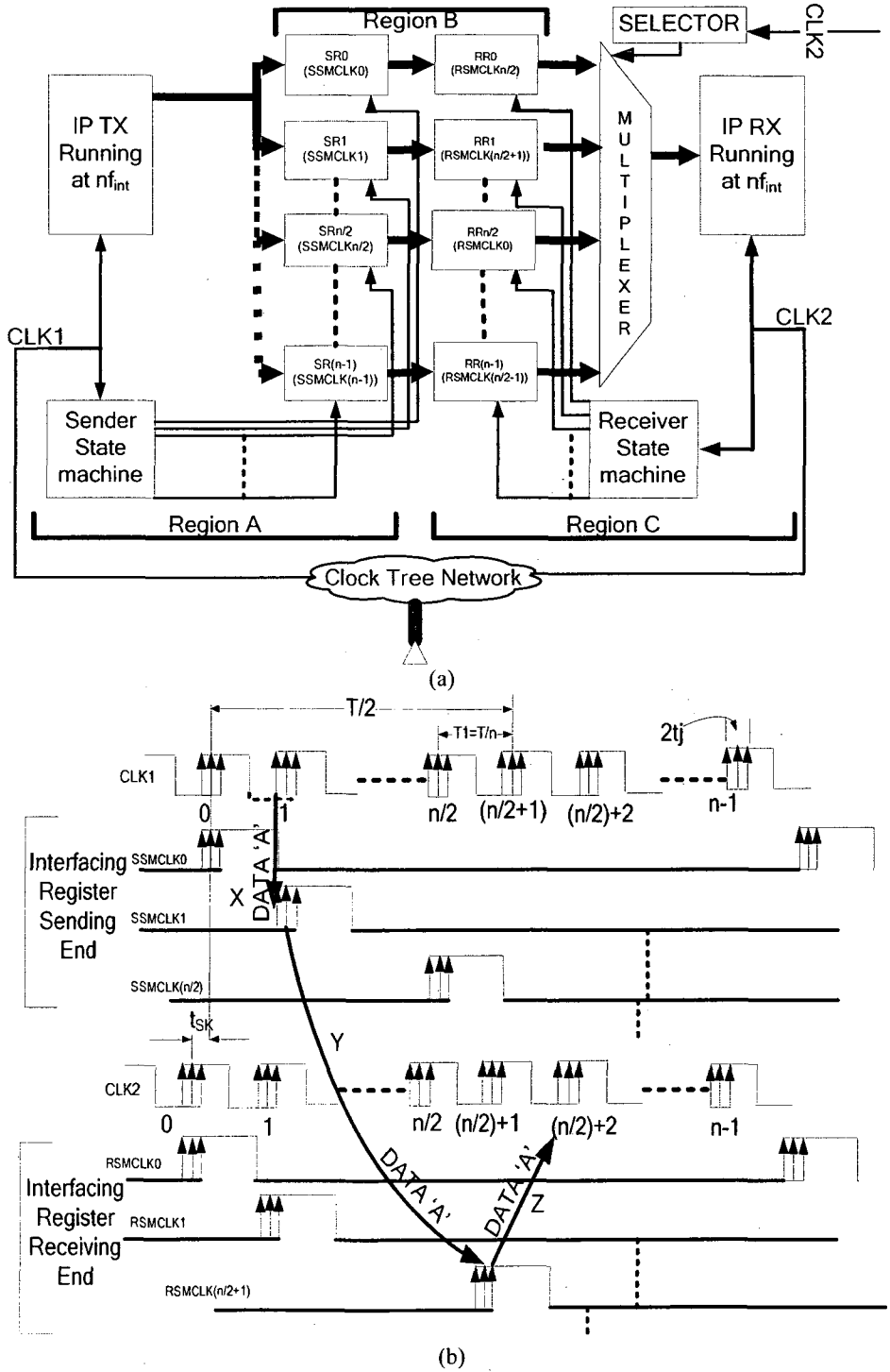


Figure 7.2. (a) Hardware implementation of interfacing registers for the n modules with same frequency and bus-width (n assumed even here for simplicity) (b) Waveform representation of interfacing registers for the n modules version with same frequency and bus-width (Arrows X, Y and Z, are showing one complete data path)

An analysis (that is provided below) shows that this design allows terminating modules to run at a frequency that is independent of the clock skew, which is in contrast to what the state-of-the-art design offers [78] (elaborated further through Equation 7.6 later in this section). The analysis also shows that the two stages of interfacing registers in Figure 7.2 decouple the two clock domains and, in the process, help achieve clock skew independency for the terminating modules. Furthermore, in this study, a comprehensive investigation is performed for unidirectional and bi-directional communications with known and unknown clock skew orientations for following clock skew magnitude: $t_{sk} <= n/2T_1$ and $n/2T_1 < t_{sk} < nT_1$, where t_{sk} is clock skew, n is the number of interfacing registers in each stage, and T_1 is the frequency of the terminating (sending and receiving) modules. This study provides timing information required by design automation tools to use such class of interface designs.

In this section, it is assumed that all the registers have the same width, terminating modules have the same frequency and correspondingly $m = n$ and $f_{S\_int} = f_{R\_int} = f_{int} = F_S/m = mT_1$ ($F_S/n = nT_2$). Figure 7.2a shows a practical implementation of such a design. In this analysis, it is also assumed that the setup time (t_{SU}), hold time (t_{hold}), and clock-to-Q delay (t_{CQ}) of all the registers in the system are equal between registers. The maximum clock skew between the phases of the clocks of the two modules involved in a point-to-point link is represented by t_{sk}, and the maximum possible jitter on each side of the nominal edge of the clock is denoted t_j. Therefore, the maximum possible deviation between the two clocks, due to jitter, is $2t_j$. The clock skew is assumed to be random but fixed (or changing very slowly) while the jitter component may change rapidly. As shown in the Figure 7.2a, the system is divided into three regions: A, B, and C. Region A and C

have deterministic timing relations for the data paths within each region, as all the modules are clocked by a single phase edge of the same clock source. Following the concept of higher bandwidth, Region B has two stages of interfacing registers and the clock signals travel different lengths for each stage. Hence, the timing relation between the left and right parts of this region is non-deterministic.

Figure 7.2b shows the expected waveform of the proposed design, shown in Figure 7.2a, where the top waveform is the sending IP (TX) clock, denoted CLK1. The next three waveforms correspond to Sender State Machine clocks, denoted SSMCLK(x), where x is an integer number from 0 to n–1 (in this figure only $n/2$ phases are depicted) and which represents the clock phase. Similarly, the next (fifth) waveform represents the receiving IP RX clock, denoted CLK2. The last three waveforms belong to the Receiver State Machine, denoted RSMCLK(x). The numbers below the CLK1 and CLK2 waveforms, in Figure 7.2b, indicate the clock sequence number. The time period of the two state machine clocks, SSMCLK(x) and RSMCLK(x) is T, which is $n\mathrm{T}_1$, with the interfacing frequency $\mathrm{f}_{int} = 1/(n\mathrm{T}_1)$. This, in turn, implies that the terminating modules are running at frequency $n\mathrm{f}_{int}$, as shown in Figure 7.2a. It can further be noticed that any two successive interfacing registers in region A and in region C respectively, RR0 and RR1 for example (or SR0 and SR1), receive clock signals separated by a T_1 time interval (or T_2 as $\mathrm{T}_1 = \mathrm{T}_2$). Moreover, in Figure 7.2b, t_{SK} represents a negative skew between CLK1 and CLK2. Generally, the skew is defined as the time difference between the nominal edges of the sending and receiving clocks, further discussion on the skew orientation is deferred to subsequent sections. The three arrows linking the rising edges represent the expected drift (jitter) in the clock. Arcs X, Y, and Z relate to one complete data-path from the sender

module to the receiver module. Sender and receiver module are shown as IP TX and IP RX in Figure 7.2a, respectively.

The following section provides the timing description about each region in Figure 7.2a.

7.2.1 Region A and C (skew independent regions)

A closer inspection of the Region A in the design shown in Figure 7.2a yields the following set of relations. Equalities (7.2) and (7.3) represent the case of one-cycle latency, while (7.2a) and (7.3a) show the case of zero-cycle latency:

$$t_{CQ} + t_{SU} < T_1 + t_{SM} \quad (7.2) \qquad t_{CQ} > t_{hold} + t_{SM} \quad (7.3)$$

or

$$t_{CQ} + t_{SU} < t_{SM} \quad (7.2a) \qquad T_1 > t_{SM} + t_{hold} - t_{CQ} \quad (7.3a)$$

where all the timing parameters correspond to those of the registers used when data traverses from IP TX to SR(x) in Region A of Figure 7.2a, while t_{SM} is the clock-to-output delay of the Sender State Machine (SSM). Similarly, by inspecting the data path from RR(x) to IP RX in Region C of Figure 7.2a, inequalities (7.4) and (7.5) are obtained:

$$max(t_{SM} + t_{MUX\_D} + t_{CQ} + t_{SU}, t_{SEL} + t_{MUX\_S} + t_{SU}) < T_1 \quad (7.4)$$

$$max(t_{SM} + t_{CQ} + t_{MUX\_D}, t_{SEL} + t_{MUX\_S}) > t_{hold} \quad (7.5)$$

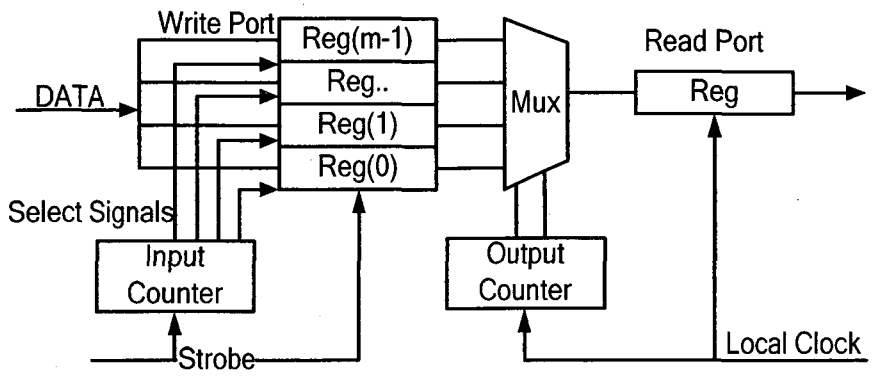


Figure 7.3. Conventional source synchronous interfacing scheme [78]

where t_{MUX_D} and t_{MUX_S} are the multiplexer delays from the data input to output, and from the selector signal to the output, respectively. t_{SEL} and t_{SM} represent the clock-to-out delay of the selector and of the Receiver State Machine (RSM) respectively. Inequalities (7.3) to (7.5) show that T_1, the terminating-IP time period, is independent of the clock skew. This is a significant improvement in comparison with the state-of-the-art designs, where the frequency of the terminating modules is always a function of the skew. For example, the frequency of the terminating modules requires fulfilling condition (7.6) in a source synchronous design:

$$t_{cnt} + t_{cq} + t_{sk} + 2t_j < T - t_{SU} \quad (7.6)$$

where t_{cnt} is the counter delay. One such design is described in [78] and is shown in Figure 7.3. In this relation, the time period of the terminating modules, denoted by T, is dependent on clock skew t_{sk}. If $|t_{sk}|$ is so large that it requires an increase of the time period T, then relation (7.6) shows that designs of this class lead to slower terminating modules. Our proposed solution solves this problem by decoupling the sender and receiver clocks with the introduction of two stages of interfacing registers, whereas clock skew is absorbed in region B that is described in the following analysis.

7.2.2 Region B

As explained earlier, the analysis for region B is the most critical as it involves two different derived clocks (from the same clock source), which feature a phase difference. In order to better illustrate the timing constraint in this region, first, an analysis that assumes a conventional design is performed devised in this analysis is either implicit or not explained in other references. For example, the introduction of delay insertion, as denoted by Δ in the following analysis, and its upper and lower limits, are usually not discussed in the literature. Also, the notations vary in different references. Hence, it is hard to avoid ambiguity while analyzing the complete set of equations of the proposed designs (developed in the following section) with the equations provided in the different references. Moreover, the following analysis allows establishing the notation in a complete self-contained derivation, and it is used as a basis to obtain timing constraint relations of the proposed schemes and serves as a benchmark for comparison. Foundations of this analysis can be found in the literature [6], [16].

Timing constraint in a conventional design, similar to region B

Hardware implementation of a conventional design is shown in Figure 7.4a. CON_CLK1 and CON_CLK2, are derived from the same clock tree network. CON_CLK1 has the time period of T_1, and similarly CON_CLK2 has the time period T_2.

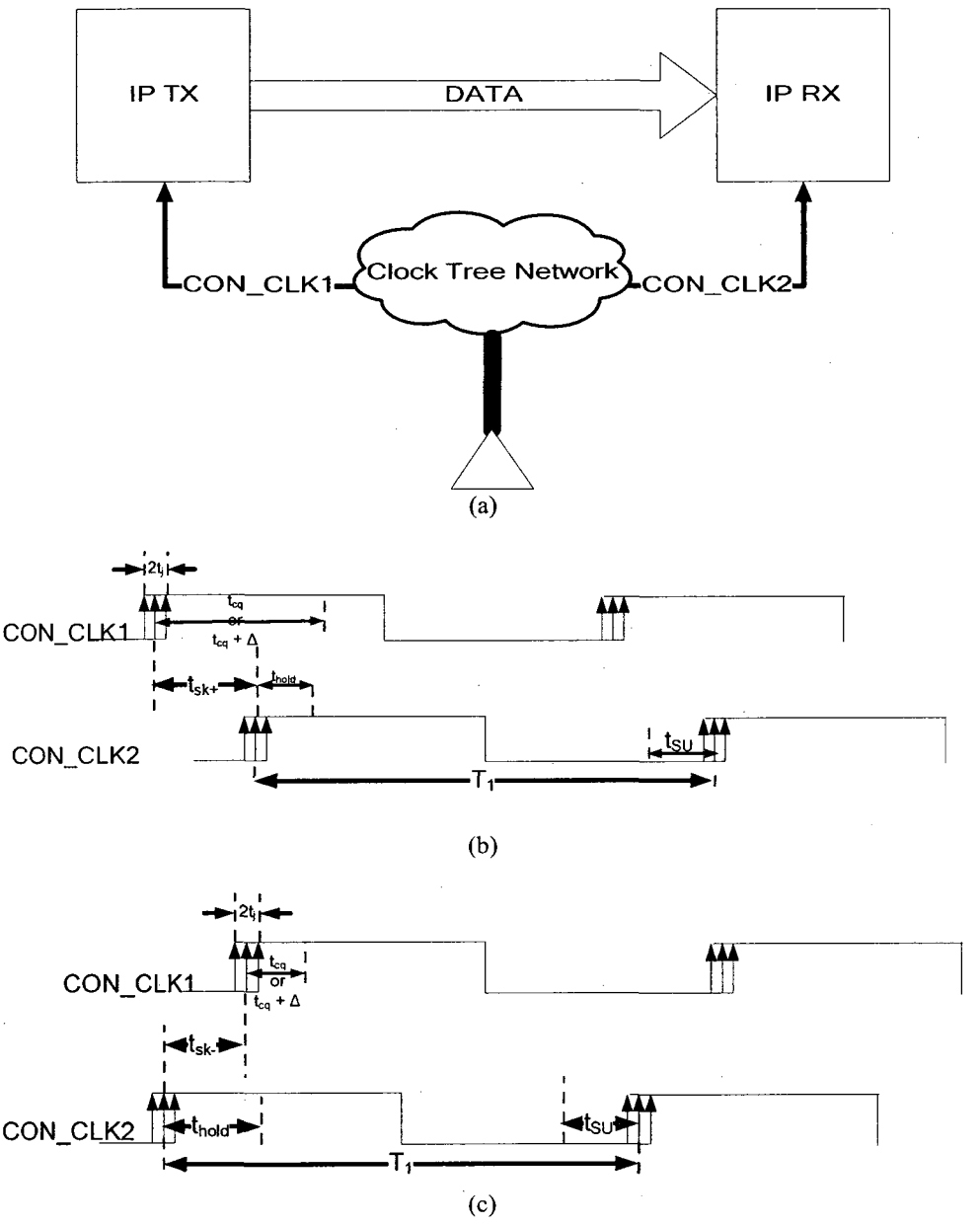


Figure 7.4. (a) Conventional two-register communication. Both registers are working with the same frequency but different phase relationship, due to skew caused by non-idealities of the clock tree network (b) Waveform representation of a conventional design subject to positive skew only (c) Waveform representation of a conventional design subject to negative skew only

Due to the non-idealities in the clock tree network, as stated earlier, there may be some phase discrepancies between the CON_CLK1 and CON_CLK2. There are two possible cases either CON_CLK2 clocks IP RX, in Figure 7.4a, before CON_CLK1 clocks IP TX

or after. Depending on the direction of data flow, these phase discrepancies are treated differently.

The two terminating modules in Figure 7.4a, IP TX and IP RX, may act as a sender or receiver at a time. If IP TX is the sending module and IP RX is the receiver module and CON_CLK2 clocks IP RX after CON_CLK1 clocks IP RX, then such a phase discrepancy is called positive skew, t_{sk+}. Similarly, if CON_CLK2 clocks IP RX before CON_CLK1 clocks IP RX, then the phase discrepancy is called negative skew, t_{sk-}. The expected waveforms for t_{sk+} and t_{sk-} are shown in Figure 7.4b and Figure 7.4c, respectively. In these figures, it is assumed that the two modules of this conventional design are communicating with a latency of one clock cycle.

To generalize the analysis, the phase discrepancy between the two clock modules in Figure 7.4a can always be represented as t_{sk}, which is defined as follows:

$$t_{sk} = t_{sk+} \text{ when } t_{sk} > 0$$

$$\text{and } t_{sk} = -t_{sk-} \text{ when } t_{sk} < 0$$

In a system subject to phase discrepancy, setup time and hold time constraints of the registers require a more careful analysis. These timing constraints are treated one by one in the following analysis. In all of the subsequent analyses, the latency is assumed to be of one clock cycle, unless otherwise stated. The following inequality shows how the setup time affects the time period of CON_CLK1 of IP TX as shown in Figure 7.4a. Provided that the data is traveling from IP TX to IP RX, irrespective of the sign of the skew, the setup time constraint can be expressed as follows:

$$T_1 + t_{sk} > t_{cq} + 2t_j + t_{SU} \qquad (7.7)$$

where, T_1 is the time period of CON_CLK1 and CON_CLK2, t_{cq} is the clock to Q delay

for the IP TX and t_{SU} is the setup time for IP RX. The term $2t_j$, in inequality (7.7), shows the worst-case, i.e. CON_CLK2 rises duration of t_j before the nominal edge, and CON_CLK1 rises t_j duration after the nominal edge. The above inequality shows that if $t_{sk} > 0$, i.e. there is a positive skew (t_{sk+}), then it is advantageous to the system as a shorter T_1 can be acceptable, this is mathematically shown in inequality (7.8). Likewise, when $t_{sk} < 0$ in inequality (7.7), i.e. there is a negative skew (t_{sk-}), then T_1 becomes longer and in turn, it makes the system slower, this is illustrated by inequality (7.9).

$$T_1 > t_{cq} + 2t_j + t_{SU} - t_{sk+} \qquad (t_{sk} > 0, \text{ waveform shown in Figure 7.4b}) \qquad (7.8)$$

$$T_1 > t_{cq} + 2t_j + t_{SU} + t_{sk-} \qquad (t_{sk} < 0, \text{ waveform shown in Figure 7.4c}) \qquad (7.9)$$

The following analysis considers the timing constraint with respect to the hold time. Conventionally, the hold time violation for the design shown in Figure 7.4a can be avoided if inequality (7.10) holds, with the same assumption that data is traveling from IP TX to IP RX:

$$t_{cq} > t_{sk} + 2t_j + t_{hold} \qquad (7.10)$$

where, t_{cq} is the clock to Q delay for IP TX, and t_{hold} is the hold time for IP RX. Again, $2t_j$ is associated with the worst-case assumption that CON_CLK1 rises for duration of t_j before the nominal edge and CON_CLK2 raises t_j after the nominal edge. Mathematically, utilizing inequality (7.10), the hold time constraints for the two possible cases of skew can be written as follows:

$$t_{cq} > t_{sk+} + 2t_j + t_{hold} \quad (t_{sk} > 0, \text{ waveform shown in Figure 7.4b}) \quad (7.11)$$

$$t_{cq} > -t_{sk-} + 2t_j + t_{hold} \quad (t_{sk} < 0, \text{ waveform shown in Figure 7.4c}) \quad (7.12)$$

If the clock-to-Q delay of IP TX does not meet the requirements of the hold time constraints in the above inequalities, then the insertion of an additional delay, Δ, is

required to cope with the phase discrepancy, t_{sk}. Hence, the above inequalities are modified as follows:

$$t_{cq} + \Delta > t_{sk+} + 2t_j + t_{hold} \ (t_{sk} > 0, \text{ waveform shown in Figure 7.4b}) \quad (7.13)$$

$$t_{cq} + \Delta > -t_{sk-} + 2t_j + t_{hold} \ (t_{sk} < 0, \text{ waveform shown in Figure 7.4c}) \quad (7.14)$$

The insertion of delay, Δ, is not used only to cope with the phase discrepancy alone; in DSM processing technologies, due to long interconnects between interfacing modules, additional buffers are inserted in the interconnects, which in turn increases the phase discrepancy between the two modules, Δ is used to manage such problems as well. However, insertion of delays has its limitations. For example, for a system with a maximum latency of one clock cycle, the delay should be inserted in such a manner that it does not affect the subsequent clock cycles. Such a limitation can be avoided by observing the following constraint:

$$t_{cq} + \Delta < T_1 - 2t_j - t_{hold} + t_{sk}$$

$$\Delta < T_1 + t_{sk} - 2t_j - t_{SU} - t_{cq} \quad (7.15)$$

where, t_{cq} is the clock to Q delay of IP TX, t_{SU} is the setup time of the IP RX, and $2t_j$ shows the worst-case timing jitter.

Relations (7.7) to (7.15) are general guidelines for the timing constraints of conventional designs. Depending upon the context of the system, a different set of relations is valid. Following, different contexts are discussed; it is assumed that all the systems use different forward and backward channels to pass the data.

System is only subjected to unidirectional communication with positive skew, i.e. $t_{sk} > 0$ or $t_{sk} = t_{sk+}$: In such systems, inequality (7.8), holds for setup time constraint and inequalities (7.11) and (7.13) holds for hold time constraint. Inequality (7.13) and (7.15)

lead to the following relationship for the delay insertion,

$$t_{sk+} + 2t_j + t_{hold} - t_{cq} < \Delta < T_1 + t_{sk+} - 2t_j - t_{SU} - t_{cq} \quad (7.16)$$

System is only subject to unidirectional communication with negative skew, i.e. $t_{sk} < 0$ or $t_{sk} = -t_{sk-}$:

Here, inequality (7.9) defines the setup time limit and inequality (7.12) is valid for the hold time constraint. As the only possible orientation of skew is negative, therefore, there is no requirement of any delay insertion.

System is subject to bi-directional skew, the maximum value of t_{sk+} and t_{sk-} is known: When the system communicates in either direction, then skew orientation has converse relationship for each pair of communications. So, under this context, the designer has to choose the worst-case design consideration. This leads to the choice of following worst-case inequalities. Inequality (7.9) is valid for the worst-case setup-time constraint, hence, it indicates the most restricted limit on time period, T_1. Similarly, inequality (7.11) is valid for the worst-case hold-time constraint when no delay is inserted. Inequality (7.16) leads to the worst-case relation for the magnitude of delay insertion.

System is subject to bi-directional unknown skew orientation, the maximum value of t_{sk+} and t_{sk-} is known: This is a context in which the digital system designer is provided only with the maximum phase discrepancy and no knowledge of the skew orientation. In such a case, the choice of the magnitude of the inserted delay, Δ, is of real importance. Indeed, increasing the magnitude of Δ is advantageous for one skew orientation (t_{sk+}) while it is disadvantageous for the other skew orientation (t_{sk-}). The value of Δ is obtained using the following set of rules. Firstly, the minimum value of Δ is obtained using inequality (7.13). Secondly, this minimum value of Δ is checked for the maximum

bound using inequality (7.15) for the negative skew. These two steps can be summarized mathematically as follows, where t_{sk} on either side of the relation has the same magnitude but differs in their orientation for a given context of communication,

$$t_{sk+} + 2t_j + t_{hold} - t_{cq} < \Delta < T_1 - t_{sk-} - 2t_j - t_{SU} - t_{cq} \quad (7.17)$$

Finally, the setup time constraint is checked for this minimum value of Δ using inequality (7.9a), which, in turn, advises the designer about any requirement for increasing the time period T_1 to accommodate the given t_{sk}. Inequality (7.9a) is a modified form of inequality (7.9) as shown below,

$$T_1 > t_{cq} + \Delta + 2t_j + t_{SU} + t_{sk-} \quad (7.9a)$$

7.2.3 Timing Analysis in region B of the Proposed Design

In inequalities (7.8) and (7.9), and later from inequalities (7.15) to (7.17), it is shown that, in conventional designs, the skew tolerance is proportional to the time period of the clock, which is denoted by T_1 in the above relations. In the proposed design, this limit on skew tolerance is relaxed by introducing a suitable number of interfacing registers activated by a version of the clock with an appropriate phase difference. The resulting skew tolerance becomes proportional to an integer multiple of T_1. This integer number may range from, depending upon the skew conditions, the total number of interfacing registers, n, to half of this number ($n/2$, assuming n is even; this assumption can be relaxed at the expense of more elaborate analysis).

In the rest of this section, a timing analysis of the proposed design is performed that encompasses several possible design variations. Figure 7.2b shows the expected waveform of the proposed design. To simplify the following derivations, without loss of generality,

we are performing the analysis for even number of interfacing registers in each stage. Analysis for odd number of interfacing registers is a trivial extension to this analysis. All the interfacing registers, SR(x) or RR(x) in Figure 7.2a, are clocked once, in a successive manner by the SSMCLK(x) or RSMCLK(x) signal, within duration of nT_1. At time nT_1, the state machine sends out the reference signal again, and the cycle is repeated.

To exploit the benefits of interfacing registers, the clocking order of sending and receiving registers are kept different, as shown in Figure 7.3. For example, the SR1 interfacing register at the sender end (or first stage) is clocked by SSMCLK1. SR1 is connected to RR1 at the receiving end (or second stage) of interfacing registers and it is clocked by RSMCLK((n/2)+1), where n is an even number, as stated earlier. Therefore, a phase difference of $n/2T_1$ is created between the SR(x) and RR(x). This in turn makes the digital system more tolerant to the phase discrepancy, while trading off a latency of $n/2$ times the shorter clock period, T_1.

As stated earlier arcs X, Y, and Z in Figure 7.2b illustrate the propagation of Data 'A'. Initially, the transfer of Data 'A' is initiated by the sender clock CLK1 on the sender register, IP TX. Arc X shows that Data 'A' is clocked by SSMCLK1 at the sender-end interfacing register, SR1. Arc Y shows the data path between the interfacing registers of stage I (sending end), SR1 in this case, and stage II (receiving end), RR1 in this case. This arc also shows that the RR1 interfacing register is clocked by RSMCLK(n/2+1). Arc Z shows the data path between the RR1 interfacing register and the receiver register, IP RX. The receiver-end register, IP RX, safely receives Data 'A' at the $n/2+2$ clock edge, as shown in Figure 7.2b.

In order to understand the effect of setup and hold time constraints on the clock period

and delay insertion requirements in the proposed design, an analysis is performed in line with the conventional design analysis, carried out in the preceding section. Figure 7.2b shows the expected waveform of the proposed design when the system has zero skew, i.e. $t_{sk} = 0$. For illustration purposes, the signals associated with arc X and arc Y are drawn separately in Figure 7.5. Arcs X and Y show that the data is initiated from IP TX and destined to IP RX. If the rising edge of the signal RSMCLK(n/2+1) shifts toward the left from the position shown in Figure 7.5, then the skew between CLK1 and CLK2 is negative, and the converse shift indicates a positive skew between CLK1 and CLK2. As described earlier, $t_{sk} < 0$ is a negative skew and $t_{sk} > 0$ is a positive skew.

To avoid any data loss, the maximum phase discrepancy (or maximum clock skew magnitude, $|t_{sk}|$), has to be within one large clock period, nT_1 (otherwise, an initial offset scheme or some sort of pipelining within the interconnect has to be introduced). As the RSMCLK(x) is clocked $(n/2)T_1$ apart from SSMCLK(x) at $t_{sk} = 0$, therefore, the timing constraints of such designs can be split into two cases. The following discussion first neglects the setup and hold times, which will be considered later. Let us first consider the case where $|t_{sk}| < (n/2)T_1$, with reference to Figure 7.5; in this case, the data sent on the 'First' rising edge of SSMCLK1 signal has to be latched by the 'First' rising edge of RSMCLK(n/2+1). For this case, the latency in terms of the slow (nT_1) clock period is zero. The other cases considered are when $(n/2)T_1 < |t_{sk}| < nT_1$. Of course, here, timing relations of the two interfacing registers depends upon the skew orientation. For a positive skew, the first rising edge of RSMCLK(n/2+1) moves toward the right and shifts beyond the boundary point C shown in Figure 7.5. So, the data from the 'First' rising edge of SSMCLK1 is latched beyond boundary point C by the 'First' following rising

edge of RSMCLK(n/2+1). For negative skew, the 'First' rising edge of RSMCLK(n/2+1) moves to the left with respect to the position shown in Figure 7.5 and shifts beyond boundary point A. Therefore, the data sent on the 'First' rising edge of SSMCLK1 is latched at the following rising edge of RSMCLK(n/2+1), which is one cycle later, with the system exhibiting a one nT_1 cycle latency. Of course, this 'Second' rising edge of RSMCLK(n/2+1) is a nT_1 duration away from the 'First' nominal rising edge of RSMCLK(n/2+1).

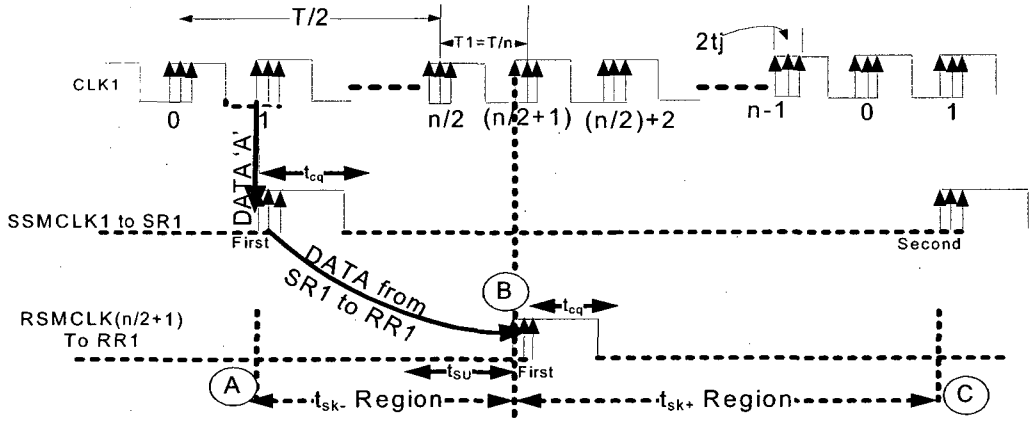


Figure 7.5. Elaboration of the proposed design for modules working at same frequency

For the first case, i.e. when |tsk| < (n/2)T_1, the setup timing constraints of the proposed design are modeled by the following inequality, (see inequality(7.7) where T_1 has been replaced by $n/2T_1$),

$$(n/2)\,T1 + tsk > t_{cq} + t_{SU} + 2tj \qquad (7.18)$$

here t_{cq} is the clock-to-Q delay of the SR1 register in Figure 7.2a, and t_{SU} is the set up time of the RR1 register. $2t_j$ is the worst-case jitter amplitude. This indicates that the clock edge defining the data path that arrives first rises t_j before its nominal time, while

the clock edge that arrives second, due to phase discrepancy, rises t_j after its nominal time. Inequality (7.18) can be written as follows for the two different skew orientations, similar to inequality (7.8) and (7.9),

$$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j - t_{sk+} \qquad (for\ t_{sk} > 0) \qquad (7.19)$$

$$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j + t_{sk-} \qquad (for\ t_{sk} < 0) \qquad (7.20)$$

For the second case, when $(n/2)T_1 < |t_{sk}| < nT_1$, the setup time constraint leads to the following generalized inequality, which is independent of skew orientation, with the assumption that latency is one clock cycle in terms of the slow clock period and t_{sk} may have either orientation:

$$nT_1 + t_{sk} > t_{cq} + t_{SU} + 2t_j \qquad (7.21)$$

where, the definition of t_{cq}, t_{SU} and $2t_j$ are the same as in inequality (7.18). In this case, t_{sk} may be formulated as:

$$|t_{sk}| = (n/2)T_1 + \delta \qquad (7.22)$$

where, δ is the portion of t_{sk} that is above $(n/2)T_1$. Substituting (7.22) into (7.21) yields:

$$nT_1 + [\delta + (n/2)T_1] > t_{cq} + t_{SU} + 2t_j \quad (7.23)$$

This inequality can be written as follows, when $t_{sk} > 0$:

$$(3/2)nT_1 > t_{cq} + t_{SU} + 2t_j - \delta \quad (7.24) \quad (for\ t_{sk} > 0)$$

In inequality (7.24), t_{cq} represents the clock-to-Q delay of the first rising edge of SSMCLK1, shown in Figure 7.5, while t_{SU} is the setup time of 'First' rising edge of RSMCLK(n/2 +1), shown in Figure 7.5. When $|tsk| > n/2T_1$, RSMCLK(n/2+1) shifts to the right beyond the boundary point C, shown in Figure 7.5, and this may cause an ambiguity condition with respect to latching the data with the first or the second rising edge of SSMCLK1. This possible system malfunction is always avoided in worst-case

designs because the worst-case setup timing-constraint condition takes care of this ambiguity condition. This condition is derived by rewriting inequality (7.23) for $t_{sk} < 0$ as follows:

$$nT_1 > t_{cq} + t_{SU} + 2t_j + \delta \qquad (for\ t_{sk} < 0) \qquad (7.25)$$

In inequality (7.25), the definitions of t_{cq} and $2t_j$ are same as above. But t_{SU} is the setup time of the second rising edge of RSMCLK(n/2+1). This edge may be visualized in Figure 7.5 when RSMCLK(n/2+1) is considered to have moved toward the left farther than $n/2T_1$. At this instant, the next rising edge of RSMCLK(n/2+1), which is not shown in Figure 7.4d, appears before boundary point C.

In the same way as the inequalities for setup time constraints are obtained, the hold time constraint is also divided into two cases. When $|t_{sk}| < (n/2)T_1$, the latency in terms of the slow clock period is zero clock cycle, hence there is no possibility of hold time violation. For the second case, when $(n/2)T_1 < |t_{sk}| < nT_1$, the following inequality (which is similar to inequality (7.10)) holds, with the addition of Δ and $n/2T_1$ (Δ is 0 when no delay insertion is required) to generalize the inequality:

$$n/2\ T_1 + t_{cq} + \Delta > t_{sk} + t_{hold} + 2t_j \qquad (7.26)$$

In inequality (7.26), t_{cq} is the clock-to-Q delay of SR1, while t_{hold} is the hold time of RR1. For a positive skew $t_{sk} > 0$, the inequality (7.26) can be written as follows:

$$n/2\ T_1 + t_{cq} + \Delta > t_{sk+} + t_{hold} + 2t_j \ (for\ tsk > 0)\ (7.27)$$

Substituting (7.22) into (7.27) leads to (7.28) which is same as inequality (7.13) in which t_{sk+} is replaced by δ:

$$t_{cq} + \Delta > \delta + t_{hold} + 2t_j\ (7.28)$$

Here, t_{cq} is measured with respect to the 'Second' rising edge of SSMCLK1 in Figure 7.5. t_{hold} is the hold time of the RR1, when the 'First' rising edge of RSMCLK(n/2+1) has moved further right to the 'Second' rising edge of SSMCLK1, i.e. beyond the boundary point C in Figure 7.5. The definition of $2t_j$ remains the same. As the rising edges are explicitly stated therefore any ambiguity with respect to clock phases has also been resolved.

Likewise, for a negative skew, i.e. $t_{sk} < 0$, inequality (27) becomes:

$$n/2\ T_1 + t_{cq} + \Delta > -t_{sk-} + t_{hold} + 2t_j \quad (for\ t_{sk} < 0)\ (7.29)$$

substituting (7.23) into (7.29), and knowing that the RSMCLK(n/2+1) signal repeats itself after an nT_1 duration, the above inequality can be written as inequality (7.30), which is similar to inequality (7.14) where $-t_{sk-}$ is replaced by $-\delta$:

$$t_{cq} + \Delta > -\delta + t_{hold} + 2t_j\ (7.30)$$

here, t_{cq} is the clock-to-Q delay of the Second rising edge of SSMCLK1 in Figure 7.5 and t_{hold} is also measured with respect to the Second rising edge of RSMCLK(n/2+1). This edge may be imagined as appearing from the shown hypothetical point C in Figure 7.5 and moving toward the left with the increment of δ. $2t_j$ still has the same definition.

In order to maintain the maximum latency of one clock cycle, there is an upper limit on delay insertion. The following inequality shows the maximum value of this delay insertion:

$$t_{cq} + \Delta < n/2\ T_1 + t_{hold} - 2t_j$$

$$\Delta < n/2\ T_1 + t_{hold} - 2t_j - t_{cq} \quad (7.31)\ \ (for\ |tsk| > n/2T_1)$$

where t_{cq} is the clock-to-Q delay of SR1 with respect to the Second rising edge of SSMCLK1 in Figure 7.5, and similarly t_{hold} is the hold time for the RR1 register. The $2t_j$

term reflects the case where the rising edge of SSMCLK1 is t_j after its nominal time while the clock edge of RSMCLK is t_j before its nominal time.

In modern DSM processing technologies, due to long interconnects between interfacing modules, additional buffers are inserted in the interconnects. This, in turn, increases the phase discrepancy between the two modules to a value that is higher than the maximum $|t_{sk}|$ specified in the analysis by the relation $(n/2)T_1 < |t_{sk}| < nT_1$. The proposed design may be extended to support such interconnect delays by leveraging the wave pipelining concept to achieve higher performance, but it is not studied here.

Comparison of the Proposed Design with the state-of-the-art design Techniques:

In line with the conventional design analysis performed in the preceding section, the same contexts are investigated here. So that a performance comparison can be made between the conventional design and the proposed design. These investigated contexts are as follows:

The system is only subject to unidirectional communication with positive skew, i.e. $t_{sk} > 0$ or $t_{sk} = t_{sk+}$. In such systems, inequality (20) holds for setup time constraint when $t_{sk} < (n/2)\ T_1$. When $t_{sk} > (n/2)T_1$, inequality (25) follows the setup time constraint. Similarly, inequalities (7.27) and (7.28) set a limit for the hold-time constraint (i.e. the hold-time constraint is only valid for $t_{sk} > (n/2)T_1$, as explained earlier). The delay insertion limit is obtained from inequalities (7.28) and (7.31):

$$\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\ T_1 + t_{hold} - 2t_j - t_{cq} \quad (7.32)$$

The system is only subject to unidirectional communication with negative skew, i.e. $t_{sk} < 0$ or $t_{sk} = -t_{sk-}$. Here, inequality (21) defines the setup-time constraint when $t_{sk} > -(n/2)T_1$ (or $|t_{sk}| < (n/2)T_1$). On the other hand, for $t_{sk} < -(n/2)T_1$ (or $|t_{sk}| > (n/2)T_1$),

inequality (7.25) follows the setup-time constraint. Inequalities (7.29) and (7.30) are valid for the hold time constraint, whereas the delay insertion limit is obtained from inequalities (7.30) and (7.31),

$$-\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\, T_1 + t_{hold} - 2t_j - t_{cq} \quad (7.33)$$

The system is subject to bi-directional communication and the orientation of the skew is known, (the maximum values of phase discrepancy is t_{sk}). This analysis must be divided into two cases: when $|t_{sk}| < (n/2)T_1$, there is no possibility of hold-time violation and, hence, only the worst-case inequality of setup time constraint, inequality (7.20), is sufficient for the whole system.

On the other hand, when this bi-directional system is subjected to $(n/2)T_1 < |t_{sk}| < nT_1$, both the hold-time constraint and the setup-time constraint have to be considered. Inequality (7.25) shows the worst-case setup-time constraint, whereas inequality (7.27) and (7.28) define the worst-case hold-time constraint. The worst-case delay-insertion value must be within the bounds shown in inequality (7.32).

The system implements bi-directional communications and the skew orientation is not known, (the maximum values are t_{sk+} and t_{sk-}): As the maximum values of phase discrepancy is known, therefore, it is known *a priori* whether $|t_{sk}|$ is higher than $(n/2)T_1$ or not. If $|t_{sk}|<(n/2)T_1$, then, as analyzed in the case of bi-directional communications, inequality (7.20) describes the period of the system and there is no possibility of hold-time violation. When the system is subject to $|t_{sk}| > (n/2)T_1$, then, following an analysis similar to that performed for the conventional design under identical conditions, a set of rules are devised, as follows, to find the maximum value of Δ that it can tolerate. Firstly, inequality (7.28) imposes the minimum value of Δ. Secondly, this minimum value of Δ is

checked with inequality (7.31) to find out whether, for a given set of conditions and a given T_1, Δ is within the bounds (specified in 7.31) or not. Finally, the setup time constraint is checked for this minimum value of Δ using inequality (7.25a), which in turn advises the designer about any requirement for incrementing the time period T_1 to accommodate the maximum specified t_{sk}. Inequality (7.25a) is a modified form of inequality (7.25) provided below,

$$nT_1 > t_{cq} + \Delta + t_{SU} + 2t_j + \delta \quad (7.25a)$$

The above inequalities of the proposed design can be compared with their corresponding inequalities in the conventional design analysis to estimate the benefits of the proposed design. This comparison is tabulated in Table 7.1 and 7.2 for uni-directional and bi-directional communication respectively. From the second column of Table 7.1, it is observed that inequality (7.20) and (7.25) of the proposed design and inequality (7.9) of the conventional design indicate worst-case setup-time constraints. Comparing these inequalities, it is found that the skew tolerance is increased, in the proposed design, in proportion to the number of interfacing registers, i.e. n, up to $n/2$ times for zero-cycle latency and up to n times for one-cycle latency.

Similarly, from the first column of Table 7.1, it is observed that relation (7.32) for the proposed design and relation (7.16) for the conventional design show that the limit of delay insertion, to cope with the hold-time constraint, is increased in the proposed design. This increase is proportional to $n/2$.

Column1 of Table 7.2 leads to exactly the same result for the setup and hold-time violations as obtained by analyzing Table 7.1. On the other hand column 2 of Table 7.2, which shows the results of bi-directional communications for unknown skew orientation,

leads to similar results but via different inequalities.

TABLE 7. 1. Timing constraints for unidirectional communication between the terminating modules of conventional and proposed designs, running at same frequency

	$t_{sk} > 0$	$t_{sk} < 0$
Conventional Design	$T_1 > t_{cq} + 2t_j + t_{SU} - t_{sk+}$ (7.8) $t_{cq} > t_{sk+} + 2t_j + t_{hold}$ (7.11) $t_{sk+} + 2t_j + t_{hold} - t_{cq} < \Delta < T_1 + t_{sk+} - 2t_j - t_{SU} - t_{cq}$ (7.16)	$T_1 > t_{cq} + 2t_j + t_{SU} + t_{sk-}$ (7.9) $t_{cq} > -t_{sk-} + 2t_j + t_{hold}$ (7.12) No requirement of any Δ insertion in this case
Proposed Design ($\lvert t_{sk}\rvert < (n/2)T_1$)	$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j - t_{sk+}$ (7.19) No hold time violations	$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j + t_{sk-}$ (7.20) No hold time violations
Proposed Design ($(n/2)T_1 < \lvert t_{sk}\rvert < nT_1$)	$(3/2)nT_1 > t_{cq} + t_{SU} + 2t_j - \delta$ (7.24) $t_{cq} + \Delta > \delta + t_{hold} + 2t_j$ (7.28) $\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\,T_1 + t_{hold} - 2t_j - t_{cq}$ (7.32)	$nT_1 > t_{cq} + t_{SU} + 2t_j + \delta$ (7.25) $t_{cq} + \Delta > -\delta + t_{hold} + 2t_j$ (7.30) $-\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\,T_1 + t_{hold} - 2t_j - t_{cq}$ (7.33)

TABLE 7. 2. Timing constraints for bi-directional communication between the terminating modules of conventional and proposed designs, running at same frequency (for known and unknown skew orientations)

	Bi-directional Communication with known skew orientation	**Bi-directional Communication with unknown skew orientation**
Conventional Design	$T_1 > t_{cq} + 2t_j + t_{SU} + t_{sk-}$ (7.9) $t_{sk+} + 2t_j + t_{hold} - t_{cq} < \Delta < T_1 + t_{sk+} - 2t_j - t_{SU} - t_{cq}$ (7.16)	$T_1 > t_{cq} + \Delta + 2t_j + t_{SU} + t_{sk-}$ (7.9a) $t_{sk+} + 2t_j + t_{hold} - t_{cq} < \Delta < T_1 - t_{sk-} - 2t_j - t_{SU} - t_{cq}$ (7.17)
Proposed Design ($\lvert t_{sk}\rvert < (n/2)T_1$)	$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j + t_{sk-}$ (7.20) no hold time violation	$(n/2)\,T_1 > t_{cq} + t_{SU} + 2t_j + t_{sk-}$ (7.20) no hold time violation
Proposed Design ($(n/2)T_1 < \lvert t_{sk}\rvert < nT_1$)	$nT_1 > t_{cq} + t_{SU} + 2t_j + \delta$ (7.25) $\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\,T_1 + t_{hold} - 2t_j - t_{cq}$ (7.32)	$nT_1 > t_{cq} + \Delta + t_{SU} + 2t_j + \delta$ (7.25a) $\delta + t_{hold} + 2t_j - t_{cq} < \Delta < n/2\,T_1 + t_{hold} - 2t_j - t_{cq}$ (7.32)

Another aspect for which the proposed design fares better is the requirement of delay insertion. Inequality (7.11) indicates that, in the conventional design, delay insertion is required when:

$$t_{sk+} > t_{cq} - 2t_j - t_{hold} \quad (7.34)$$

whereas, in the proposed design, delay insertion is required only when $t_{sk} > (n/2)T_1$. It is

known that, in most digital systems, to meet the setup-time constraint, $t_{cq} << T_1$ as can be seen in inequality (7.9) for example. Therefore, it is concluded that there is a substantial relaxation in timing budget to enable delay insertion in the proposed system. This relaxation is also proportional to *n/2*. The clock period of the proposed design can be estimated by using all the relevant relations, if $|t_{sk}| < (n/2)T_1$,

$$T_1 = max(((t_{cq}+\Delta+t_{SU}+2t_j+t_{sk})/n),$$

$$max(t_{SM}+t_{MUX\_D}+t_{cq}+t_{SU},\ t_{SEL}+t_{MUX\_S}+t_{SU}),(t_{cq}+t_{SU}-t_{SM}),(t_{SM}+t_{hold}-t_{cq})) \qquad (7.35)$$

and if $nT_1 < |t_{sk}| < (n/2)T_1$

$$T_1 = max(((t_{cq}+\Delta+t_{SU}+2t_j+t_{sk})/n),\ 2/n(\Delta+t_{hold}+2t_j+t_{cq}),$$

$$max(t_{SM}+t_{MUX\_D}+t_{cq}+t_{SU},\ t_{SEL}+t_{MUX\_S}+t_{SU}),(t_{cq}+t_{SU}-t_{SM}),(t_{SM}+t_{hold}-t_{cq})) \qquad (7.36)$$

where, the first term is the generalized form of inequalites (7.25), (7.25a), and (7.20), and the second, third, fourth, and fifth term in (7.36) are directly taken from inequalities (7.31), (7.5), (7.3), and (7.4a), respectively. This equation can be compared to the similar expression for conventional design which is obtained from inequality (7.9a) and (7.15) and is as follows:

$$T_1 = max\ (t_{cq}+\Delta+2t_j+t_{SU}+t_{sk},\ \Delta+2t_j+t_{hold}+t_{cq}) \qquad (7.37)$$

The first term of equations (7.35), (7.36), and (7.37) contains t_{sk} and it is likely in DSM designs that this term would dominate. Hence, comparison of these first terms shows that the proposed design allows terminating modules to run at faster frequencies. Quantitatively, T_1 can be t_{sk}/n when $t_{sk} >> t_{cq} + \Delta + 2t_j + t_{SU}$, which is a realistic condition in designs utilizing modern DSM technologies.

In comparison with the state-of-the-art design, which was explained when describing relation (7.6), it is observed that only the Region-'B' components of the proposed design

require clock-skew information in this analysis. Therefore, it is concluded that *the clock-skew (up to the stated limits) is absorbed by the interfacing registers. Also, as the maximum achievable frequency of terminating modules is obtained through the analysis of Region A and C, it is concluded that their clock period is virtually independent of clock skew.* Consequently, our proposed design allows the terminating modules to run at higher clock frequencies, which leads to achieving higher overall performance of the digital system.

In comparison with conventional pipeline implementation we compared our design approach with the state-of-the-art design shown in Fig. 7.3. The details of this comparison are provided in the preceding section while explaining Region A and C. Here, it is worth mentioning that in comparison to conventional pipelining approach our proposed design allows the terminating modules to run faster, which allows more throughput.

7.3 Utilization of the higher bandwidth concept, when $m < n$

Having studied the utilization of the higher bandwidth concept for $m=n$, in this section, we develop a design scheme that allows communication when the modules are working at different frequencies with their frequency ratio being an integer. In this study, only communications between Fast-to-Slow modules are discussed, denoted as F-to-S systems. Analyzing the constraints for the case of communications between Slow-to-Fast (S-to-F) modules is simply a trivial extension of this analysis. S-to-F and F-to-S variants of the proposed design can be applied to design parallel-to-serial and serial-to-parallel

data converters, respectively. These data converters are fundamental blocks for the design of Serializers-Deserializers commonly known as SERDES. The hardware implementation and expected waveform of the conventional design of this category is shown in Figure 7.6, while the proposed design is depicted in Figure 7.7.

Timing Constraint in conventional designs: In conventional design schemes, if the sender module is n times faster than the receiver module, a F-to-S system, then the receiving module has to be designed such that it latches n data items at each receiving end (slower) clock cycle edge. Mathematically, the frequency relationship between the terminating modules can be represented as: $F_S = nF_R$ and $f_{int} = F_R$, and it implies that $F_S=1/T_1$, $F_R=1/T_2$, and $n=T_2/T_1$. The hardware implementation and the expected waveform of conventionally implemented F-to-S systems are shown in Figure 7.6a and Figure 7.6b, respectively. A detailed analysis similar to the one presented for $m = n$ in Section 7.2 was applied to conventional F-to-S systems. The timing constraints were obtained by assuming the data is traversing between IP TX to IP RX(X) via the Demultiplexer in Figure 7.6a. The inequalities obtained for this design are summarized in Table 7.3. For simplicity, it is assumed, in inequalities 7.38 to 7.44 in Table 7.3, that the Demultiplexer has the same delay in both selector to data output and data input to data output paths. Table 7.3 also shows the corresponding inequality obtained in Section 7.2 to illustrate the similarity of these relations.

Timing constraints in the proposed F-to-S system. This section performs the mathematical analysis in the same way as was done for the proposed design, in the preceding section, when $m=n$. Here, the frequency relationship between the terminating modules can be represented as: $F_S = nF_R$ and $f_{int} = F_R$, and it implies that $F_S=1/T_1$,

$F_R=1/T_2$, and $n=T_2/T_1$. In the proposed F-to-S system, it is considered that the slow receiver module is able to latch the data only once in the entire slow cycle nT_1, where as the sender may send data at its every clock tick. Figure 7.7a and 7.7b respectively, show the proposed interfacing mechanism and expected waveforms for the proposed F-to-S system.

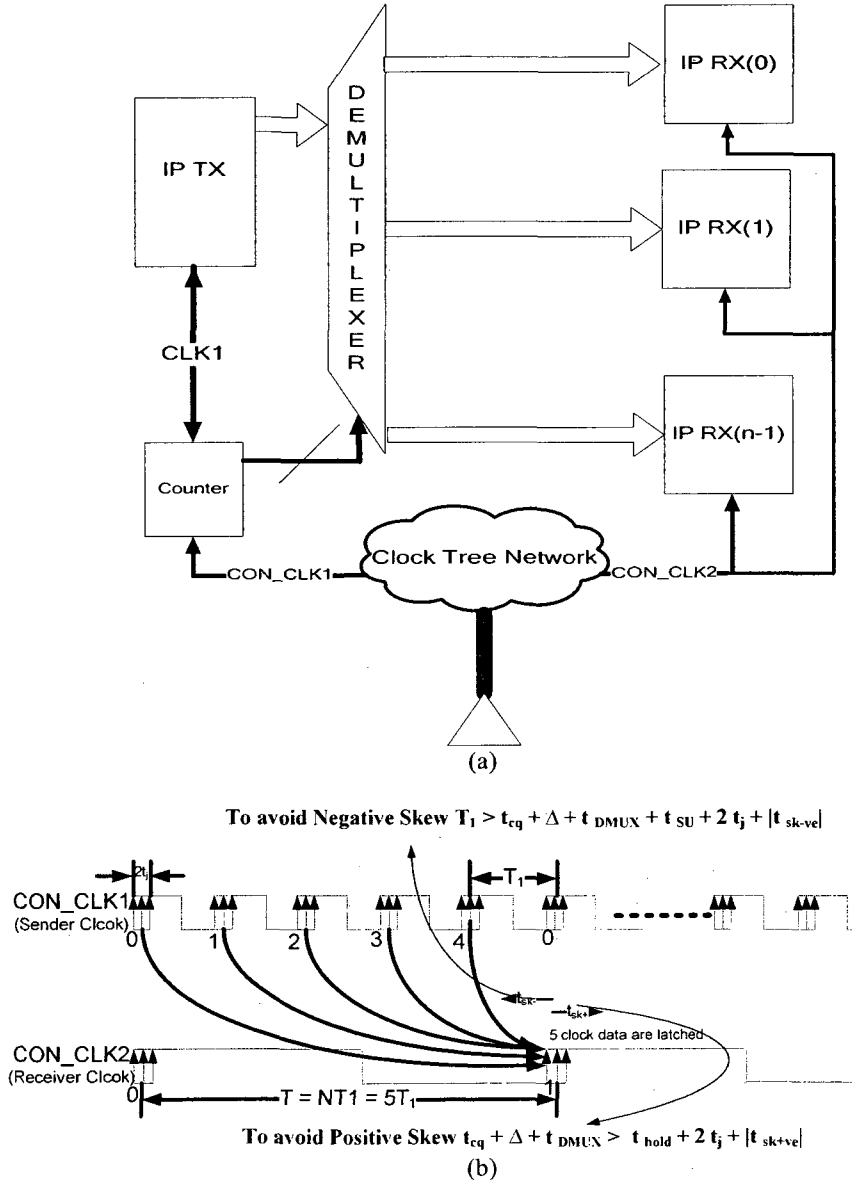


Figure 7.6. (a) Hardware implementation of F-to-S Conventional Design. (b) Waveform Representation of conventional design, with faster sender module and slower receiver module (F-to-S systems)

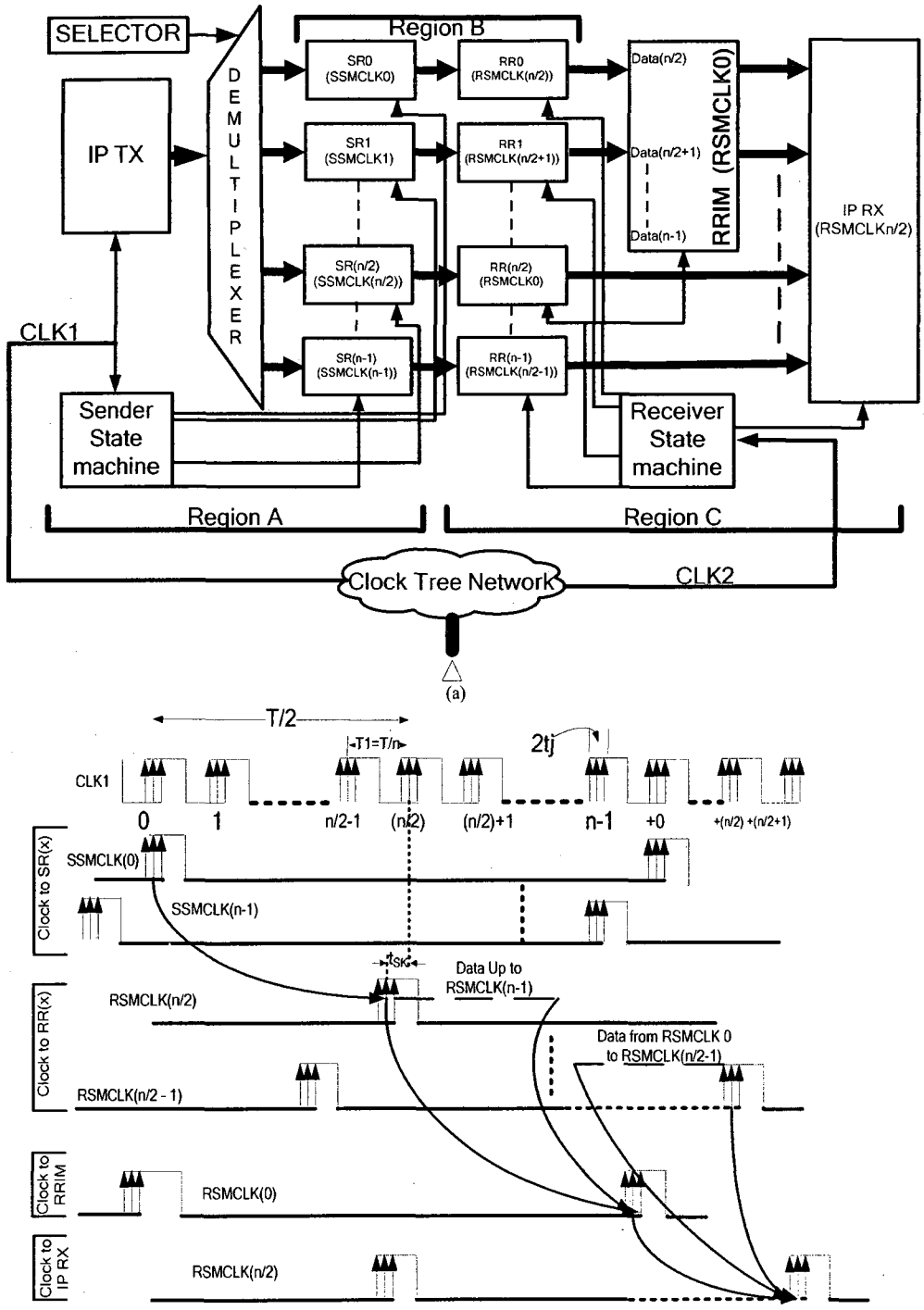


Figure 7.7. (a) Hardware implementation of Interfacing registers for the n modules with integer multiple frequencies. Faster sending module and slower receiving module (F-to-S system–II) (b) Waveform representation of Interfacing registers for the n modules with integer multiple frequencies. Faster sending module and slower receiving module (F-to-S system-II)

TABLE 7. 3. A Summary of timing constraints for conventional F-to-S system, along with the corresponding inequalities of conventional design of Figure 7.4a, i.e. when the terminating modules have same frequency

	F-to-S conventional System shown in Figure 7.6a	**Conventional Design shown in Figure 7.4a**
Setup time constraint	$T_1 + t_{sk} > t_{cq} + t_{DMUX} + t_{SU} + 2\,t_j$ (7.38) $T_1 > t_{cq} + t_{DMUX} + 2t_j + t_{SU} - t_{sk+}$ $(t_{sk} > 0)$ (7.39) $T_1 > t_{cq} + t_{DMUX} + 2t_j + t_{SU} + t_{sk-}$ $(t_{sk} < 0)$ (7.40)	$T_1 + t_{sk} > t_{cq} + 2t_j + t_{SU}$ (7.7) $T_1 > t_{cq} + 2t_j + t_{SU} - t_{sk+}$ (7.8) $T_1 > t_{cq} + 2t_j + t_{SU} + t_{sk-}$ (7.9)
Hold time constraint	$t_{cq} + \Delta + t_{DMUX} > t_{hold} + 2\,t_j + t_{sk}$ (7.41) $t_{cq} + \Delta + t_{DMUX} > t_{sk+} + 2t_j + t_{hold}$ $(t_{sk} > 0)$ (7.42) $t_{cq} + \Delta + t_{DMUX} > -t_{sk-} + 2t_j + t_{hold}$ $(t_{sk} < 0)$ (7.43)	$t_{cq} > t_{sk} + 2t_j + t_{hold}$ (7.10) $t_{cq} + \Delta > t_{sk+} + 2t_j + t_{hold}$ (7.13) $t_{cq} + \Delta > -t_{sk-} + 2t_j + t_{hold}$ (7.14)
Delay insertion limit	$\Delta < T_1 + t_{sk} - 2t_j - t_{hold} - t_{cq} - t_{DMUX}$ (7.44)	$\Delta < T_1 + t_{sk} - 2t_j - t_{SU} - t_{cq}$ (7.15)

Region A and B: Inspection of Figure 7.7a demonstrates that Region A and B of this design are identical to Region A and B of Figure 7.2a. Hence, their timing constraints are also identical. Table 7.3 shows that the conventional design shown in Figure 7.6 has timing constraints of the same order (but a little higher due to the demultiplexer) as for the conventional design shown in Figure 7.4. Due to the fact that Region B is identical for the proposed design of both the cases, $m = n$ and $m < n$, the skew absorption and tolerance advantages (illustrated previously for $m=n$) are similar for both these proposed designs.

Region C: Region C of Figure 7.7a is also comparable to Region C of Figure 7.2a, where the multiplexer of Figure 7.2a is replaced by a wider register RRIM and IP RX of Figure 7.2a is replaced by a wider IP RX in Figure 7.7a. Due to the different nature of IP

RX in this context (i.e. IP RX is slower than IP TX), a wider register is required to latch the entire set of *n*W_int* data items concurrently at the receiving end (i.e. Region C). This, in turn, introduces a worst-case timing constraint that the data transaction between RR(X) and IP RX has to be completed within one (fast) clock cycle, T_1. To alleviate this local timing constraint an intermediate register, RRIM is introduced. This register has data port of width *n/2 * W_int*, which allows concurrent latching of half of the data items from the RR(x) interfacing registers. Therefore, IP RX receives half the data (sent by RR(X)) from RRIM and the other half directly from RR(x). Hence, RRIM retimed the design and acts as a buffer stage to improve timing constraint between the RR(X) to IP RX up to $(n/2)T_1$.

This phenomenon is further elaborated in Figure 7.7a. This figure shows that RR(X) receives the data sent by IP TX (via SR(X)) and latches it at RSMCLK(n/2) to RSMCLK(n/2-1) clock edges. RRIM latches half of the data items, from RR(0) to RR(n/2-1) at the next clock edge, RSMCLK(0). IP RX is clocked with RSMCLK(n/2) and it latches data safely from all the RR(X) registers, with half of the data coming via RRIM. The benefit of using RRIM is due to the fact that, for the duration when it receives the data from RR(X) (that are latched at RSMCLK(n/2) to RSMCLK(n-1)), IP RX does not receive any data. Therefore, if the internal delay of Region C exceeds duration T_1, then RRIM acts as a buffer stage and allows IP RX to utilize clock edges for which the IP RX does not receive any data. Hence, IP RX may latch the data as late as RSMCLK(n-1) enabling the design to tolerate a local delay, to Region C, of $n/2T_1$, as stressed by the arrows from RR(X) to IP RX in Figure 7.7b. This timing constraint is deterministic in Region C as all these modules are clocked by CLK2 or its derivatives. The worst-case

timing constraint is described by the following inequality for the data path from RR(X) to IP RX(X) via RRIM in Figure 7.7a:

$$(n/2)T_1 > \max(t_{cqRR(X)}, t_{cqRRIM}) + t_{SU} + t_{SM} \quad (7.45)$$

where $t_{cqRR(X)}$ and t_{cqRRIM} are the clock-to-Q delay of RR(X) and RRIM, respectively. Inequality (7.45), along with equalities (7.2) and (7.3) obtained for region A in the preceding section, indicates that *the clock frequency of the terminating modules, IP TX and IP RX, are still independent of the clock skew.* IP RX may latch the data at any clock phase edge from RSMCLK(n/2) to RSMCLK(n-1). *This design latches the data without slowing down the fast sender module.*

A similar design may be utilized for S-to-F interface. We may draw parallels from F-to-S design for S-to-F designs, such as Region B should be identical, and Region A and Region C should be swapped. Thus, the analysis performed for F-to-S system may be extended for S-to-F system. Due to its similar nature, this is not performed separately in this work. However, it is safe to say that the major advantage of skew tolerance is almost identical to what is shown for F-to-S systems. .

7.4 Summary of the Advantages of the Proposed Design:

This section summarizes the advantages of the proposed design over state-of-the-art design schemes, such as shown in Figure 7.3, and conventional designs, such as shown in Figure 7.4 and Figure 7.6. This summary shows that the proposed design achieves skew tolerance and delay insertion relaxation while averting the drawbacks associated with state-of-the-art designs.

Skew Tolerance is increased – Inequalities (7.20) and (7.25) indicate that the skew tolerance is increased in the proposed design in proportion to the number of interfacing registers.

Delay Insertion limit is increased – Relation (7.32) indicates that the limit of delay insertion, to cope with the hold time constraint, is proportional to *n/2*. This is a substantial improvement compared to conventional design requirements where clock skew directly limits the delay insertion capability.

The above mentioned advantages are achieved by some of the state-of-the-art designs as well. However, our design not only provides these benefits but it also avoids the following shortcomings of the state-of-the-art designs:

The terminating-module timing margins are virtually independent of the clock skew – As seen in section 7.2 and 7.3, inequalities (7.3) to (7.5), and (7.46) show that T_I, the terminating-IP time period, is independent of the clock skew. A closer inspection identifies that the relaxation in T_I is directly proportional to the number of registers in each stage.

No possibility of clock-data delay mismatch – No clock signal needs to travel the entire length of the data path. Therefore, there is no possibility of clock-data delay mismatch, which is a major performance bottleneck in state-of-the-art source synchronous designs [80].

A faster module can communicate with a slower module without compromising the frequency of faster module. On the other hand, the source synchronous state-of-the-art designs, [76], [23], use rate multipliers that limit the frequency of the interface to be equal to or slower than the slowest communicating module.

Our work also provides a complete mathematical model for all the different possible scenarios in a digital system while crossing the clock domains. This is beneficial in implementing such designs using CAD synthesis tools based on standard cells.

7.5 Simulation Setup and Results

In order to verify the validity of the proposed solution, detailed design and simulations were performed for the case explained in the section 7.2, where two terminating modules are communicating at the same frequency. Gate level synthesis, simulations, and static timing analysis are performed for 0.18 micron TSMC CMOS process technology, using Synopsys' Design Compiler (DC) [99], and Prime Time for static timing analysis [100]. This section describes the assumptions and simulation setup, along with the results and their comparison with the conventional synchronous designs.

Two different sub-cases are simulated. The first sub-case simulates the hardware implementation of the structure shown in Figure 7.4a, which is a conventional design and does not contain any interfacing registers. The other sub-case simulates an example of the proposed hardware implementation shown in Figure 7.2a. This simulated design example has four interfacing registers at each terminating end.

It can be seen in Figure 7.8 that, for a given skew value, under unidirectional communication, and when the system is subject to positive skew only, the proposed design with four interfacing registers at each stage can communicate at a higher frequency than the conventional design. The proposed solution with four interfacing registers at each terminating end is called the Quadruple Bus Width (QBW) solution.

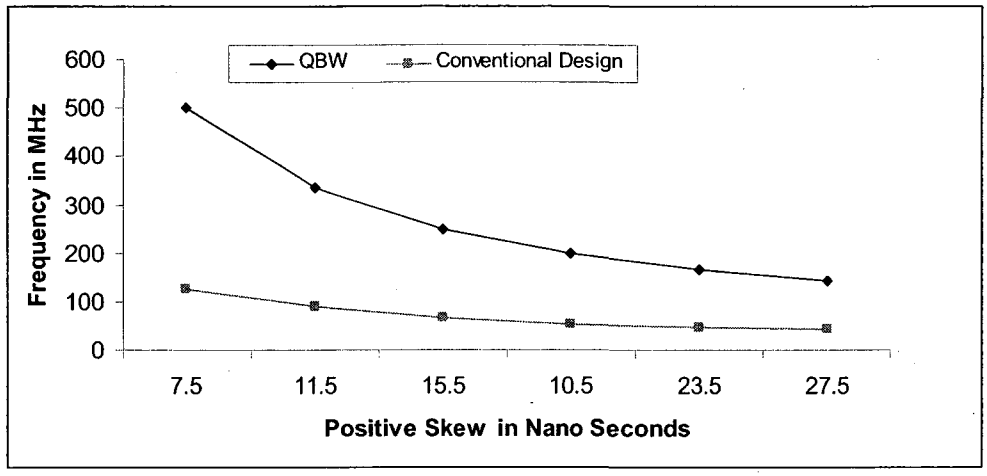


Figure 7.8. Simulation results showing effect on frequency with the increase in positive skew for unidirectional communication (QBW and Conventional Designs)

Initially, to keep the speed of terminating modules within practical limits, it is observed, from gate level simulations, that a very simple circuit, the TFF (Toggle Flip Flop), can operate at a maximum frequency of approximately 500 MHz with the 0.18 micron TSMC CMOS process technology. It is possible to obtain higher frequency using custom design for the same circuit using the same technology but, as this would digress from the goal of our research, we considered the maximum frequency for this technology synthesized using standard design library for Design Compiler. In Figure 7.8, it is shown that the conventional design can run at up to 125 MHz for a 7.5 nsec skew, whereas the proposed design allows the terminating modules to work at a frequency as high as 500 MHz for the same skew. This improvement in positive skew tolerance is in accordance with the analytical study performed in section 7.2. Here the skew is bounded by, $n/2T_1 < |tsk| < nT_1$ and the simulation results are within the limit provided by inequality (7.24).

The second simulation result, shown in Figure 7.9, addresses the timing constraints associated with unidirectional systems that are subject to a negative skew only. The

simulation results show that, with a 3.3 nsec negative skew, a conventional design can work at a maximum frequency of 250 MHz, while the proposed design for the same skew can work at double the frequency, i.e. 500 MHz. This means QBW works twice as fast (n/2 = 4/2) as the conventional design for the same skew budget. This is in agreement with the analytical results obtained for the proposed design when $|t_{sk}| < (n/2)T_I$ and expressed by inequality (7.20).

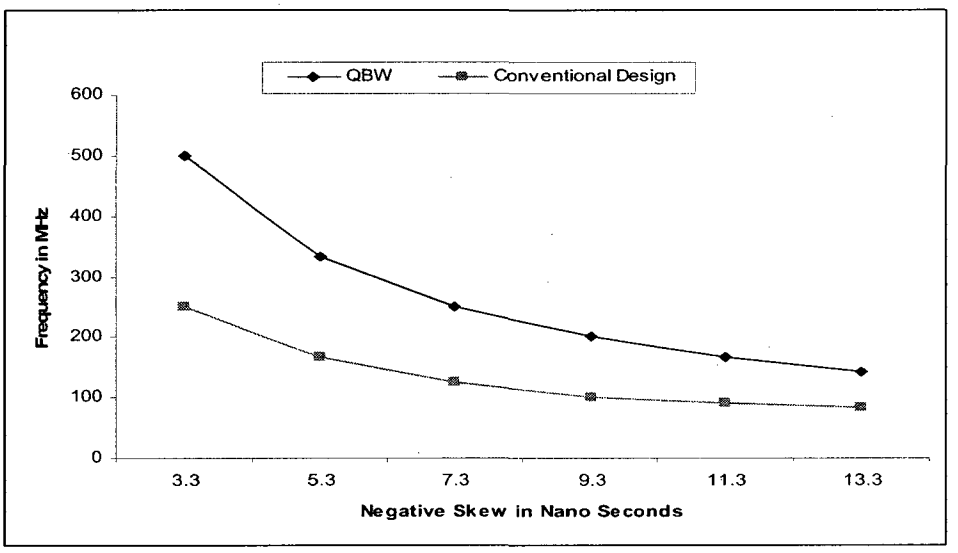


Figure 7.9. Simulation results showing effect on frequency with the increase in positive skew for unidirectional communication. (QBW and Conventional Designs)

Figure 7.10 shows a third case where bi-directional communications are simulated under the assumption that skew orientation is not known. Data points for conventional design are shown with the squares. Data points for QBW design are represented by diamonds. The simulation results in Figure 7.10 for the zero clock cycle latency case show, for a skew of 3.3 nsec, that the terminating modules of the proposed design communicate at 500 MHz, as compared to 142 MHz for conventional design under the

same context. It can be noticed that 3.3 nsec is less than $(n/2)T_1$; hence, the obtained frequency values are in compliance with inequality (7.9a) for the conventional design and with inequality (7.20) for the proposed design. Similarly, the proposed design tolerates about 13.3 nsec of skew at the frequency of 142 MHz. This result is still within the limit of $(n/2)T_1$ duration, in compliance with the theory, and also shows a skew tolerance of approximately 4 times than its conventional counterpart running at the same frequency.

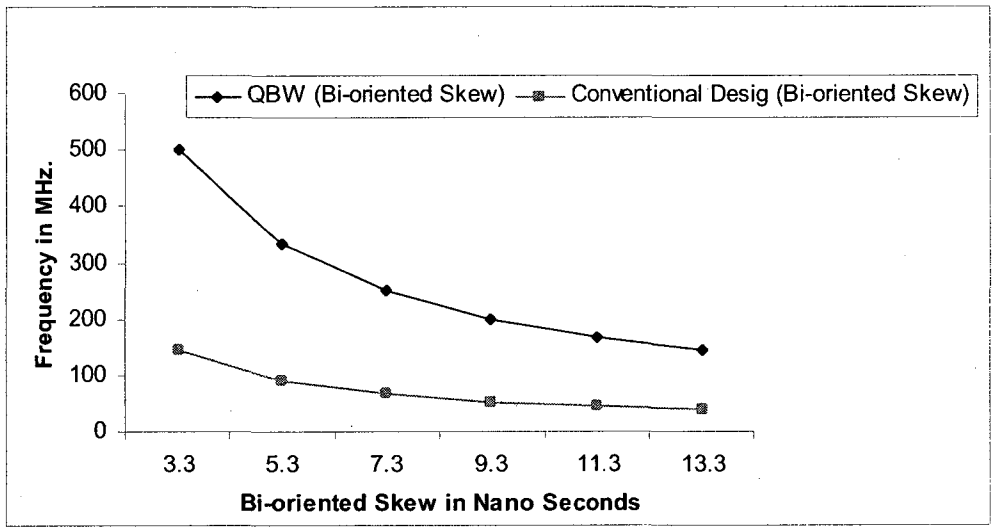


Figure 7.10. Bi-Oriented Skew vs. Frequency for Case A (QBW and Conventional Designs)

7.6 Prototype Implementation and Back-Annotated Simulation Results

Prototypes of the proposed designs were implemented using a Virtex-II Pro-based FPGA board from Xilinx. A gate-level HDL description was written for the proposed design shown in Figure 7.2a. In this prototype implementation, a 3-bit binary counter is

the (data generating) sender module, along with two stages of four interfacing registers and the receiver unit comprises a 4-to-1 Multiplexer and a receiver register. Figure 7.11 (and Figure 7.12) shows the back-annotated simulation results of the design where the terminating modules are working at 250 MHz and a negative (and positive, respectively) clock skew of 12 ns is applied (which is more than $(n/2)T_l$). Both design parameters are limited by the FPGA technology used. The Virtex-II Pro FPGA XC2VP30-7FF896 can run at a maximum clock frequency of 250 MHz under the slow model and at 320 MHz under the fast model, when an internal clock of 100 MHz is used to synthesize frequencies through Digital Clock Managers (DCM) [101]. The 12–ns skew used is close to the maximum delay that can be introduced by this FPGA technology [101]. Outputs 1 to 4 in Figure 7.11 and Figure 7.12 correspond to data output from SR(X) and similarly outputs 5 to 8 represent data output from RR(X). Output_counter and System_output represent data of IP TX and IP RX modules, respectively. Also, CLK FX and CLKFX2 represent CLK1 and CLK2 respectively.

Similarly, Figure 7.13 and Figure 7.14 show the back-annotated simulation waveforms of the proposed F-to-S design described in Figure 7.7a under the assumption that the skew can be negative or positive, respectively. The signal names in Figures 7.13 and 7.14 are almost identical to those in Figures 7.11 and 7.12, with few additions and exceptions: IMoutput1 and IMoutput2 represent data coming out of RRIM while the wider bus is used for IP RX, therefore the width of system_output is four times that of the sender, which is denoted as System_output(X) in Figures 7.13 and 7.14. It can be seen, from the waveforms, that the sender module is working at 250 MHz and that the skew, in either orientation, is 12 ns. Following the data path from the counter (IP TX in Figure 7.7a) to

the System_output (IP RX in Figure 7.7a), it is concluded that the sender data is completely and safely latched at the receiver.

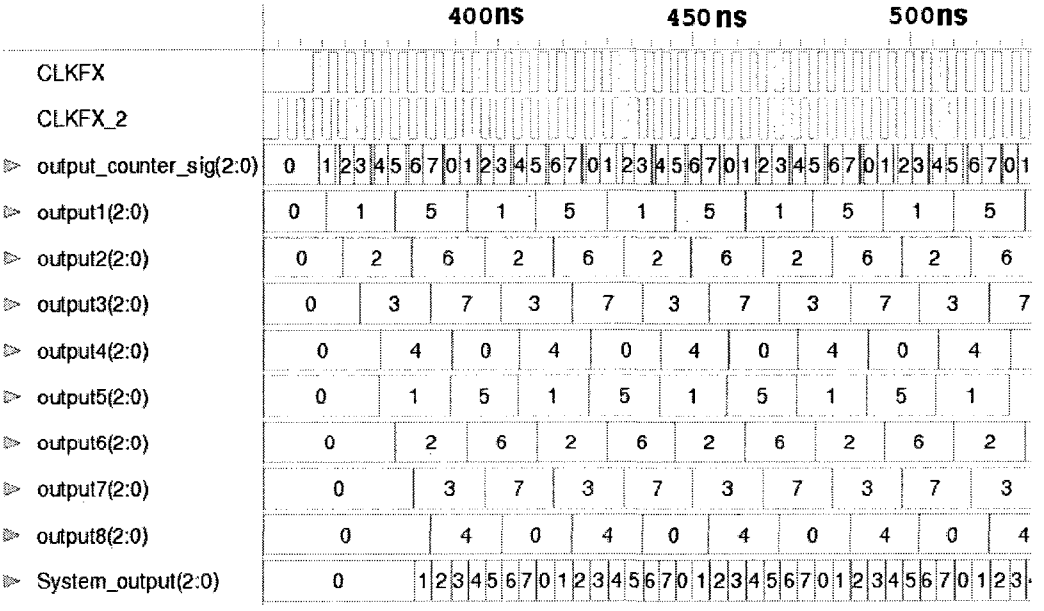


Figure 7.11. Back-Annotated Simulation Results using Xilinx Virtex II-Pro, for the proposed design shown in Figure7.3a, terminating module working at 250 MHz. and negative skew of 10 ns (> (n/2)T1)

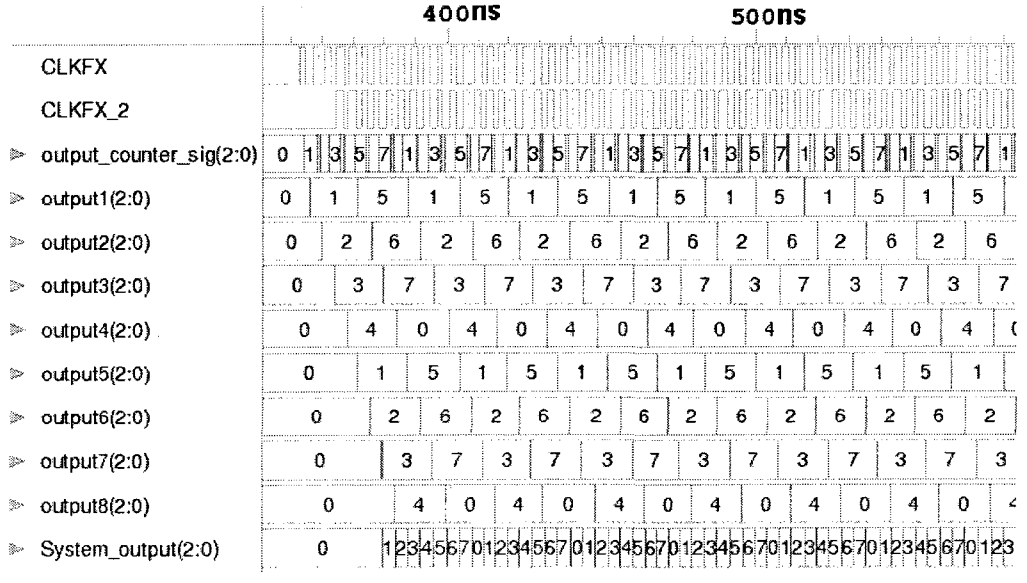


Figure 7.12. Back-Annotated Simulation Results using Xilinx Virtex II-Pro, for the proposed design shown in Figure7.3a, with terminating module working at 250 MHz. and a positive skew of 10 ns (> $(n/2)T_1$)

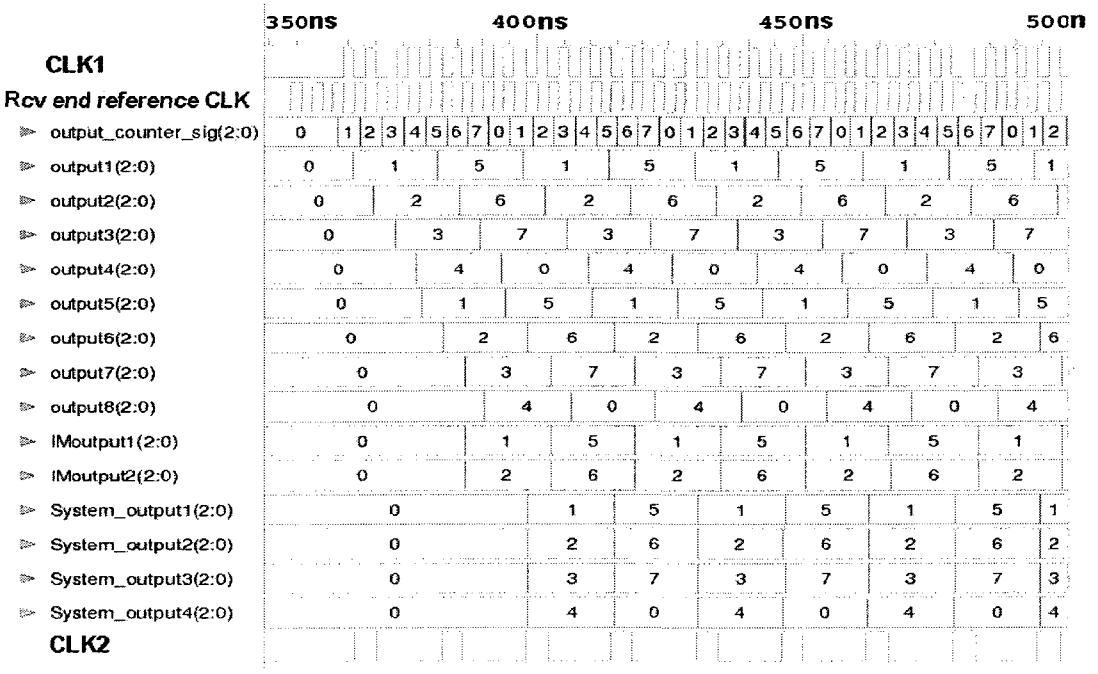


Figure 7.13. Back-Annotated Simulation Results using Xilinx Virtex II-Pro, for the proposed design shown in Figure7.7a, with terminating module working at 250 MHz. and a negative skew of 10 ns (> (n/2)T1)

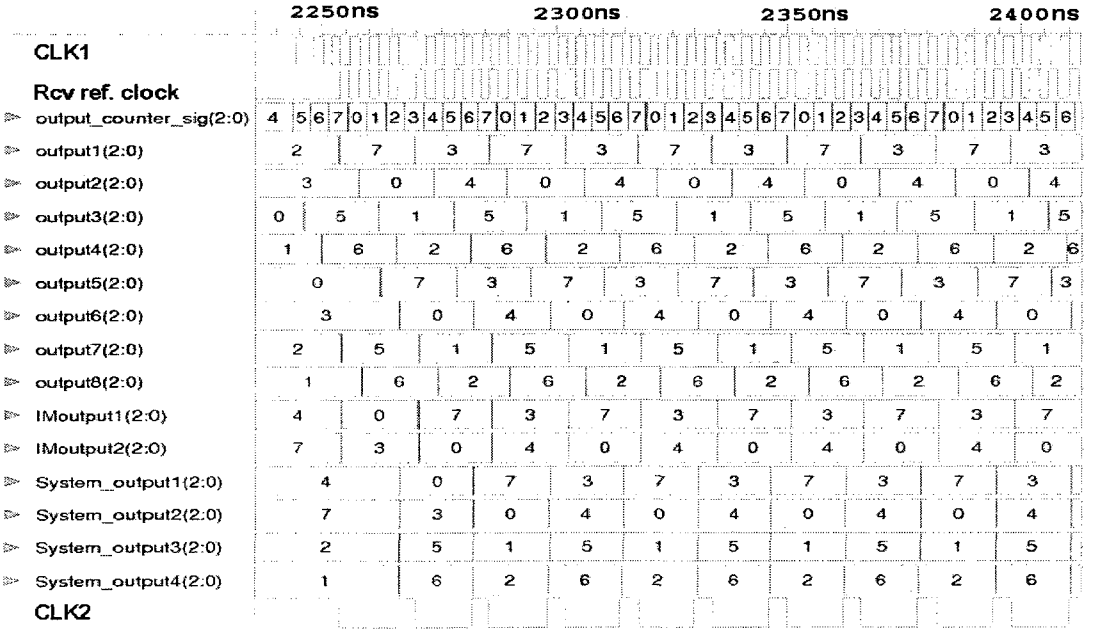


Figure 7.14. Back-Annotated Simulation Results using Xilinx Virtex II-Pro, for the proposed design shown in Figure7.7a, with terminating module working at 250 MHz. and a positive skew of 10 ns (> (n/2)T1)

Due to the limited number of channels that can be shown on the oscilloscope that was used for this experiment, all the data signals cannot be shown concurrently. The design chosen for demonstration purposes is shown in Figure 7.7a where the output module is slowed down to one fourth of the sender frequency. Figure 7.15 shows a representative waveform of the prototype implementation. This figure demonstrates the case when the sender module is working at 350 MHz, which corresponds to a time period of 2.67 ns. Note that, although the frequency limit according the manual is 320 MHz [101], in practice it is seen that a frequency of 350 MHz is achieved. This may be attributed to the fact that the system designed in this work does not exactly match the loading criterion and the process parameters for which the value is reported in the design manual. As it is of lesser significance for our design goal, we did not investigate further this issue.

The skew applied to this design is 12 ns in either orientation. This skew was chosen to observe the effect of skew of a magnitude higher than $(n/2)T_l$ on the proposed skew tolerant design. Two separate tests are performed for positive and negative skews, applying the same skew.

The three-bit binary counter counts up to 7 and then resets to 0. Each set of three bits is called a count value below. Therefore, the 3-bit counter has a total of 8 possible count values, from 000 to 111. The time to complete the counting from 000 to 111 and back to 000 is called a count cycle. The top waveform of Figure 7.15 represents the MSB (most significant bit) of the count values from one of the four sending-end interfacing registers. The second waveform from the top shows the MSB of the corresponding IP RX register at the receiving end. As the three-bit counter is working four times faster than the corresponding receiver, and since, as explained above, each count cycle is performed in

eight fast clock cycles, therefore, each receiving-end register can hold only two of the eight possible count values in one count cycle. Each count value in the receiving-end register lasts for four sending-end (faster) clock cycles. Therefore, the fastest transition that can be seen at the receiver end is one fourth of the faster clock cycle and is visible at the MSB (and this is why only that bit is shown). Note, however, that it has been observed through the FPGA implementation, that the receiver register receives all the count values.

Figures 7.15 to 7.18 show the output of the first-stage interfacing registers and compare them with the values latched in corresponding IP RX register. This experiment not only demonstrates that the correct functionality is obtained at the maximum frequency of the FPGA used, but it also shows that the 12-ns clock-skew is tolerated. It is not possible to distinguish, in these waveforms, whether the skew is positive or negative (only the phase difference is shown) but the experiment was performed for both types of skew. The decreasing phase difference of IP RX with each of the interfacing register validates the functionality. The frequency of the analog waveform is measured to be 43.6 MHz, which is very close to (350/8) MHz. The delay between the two MSB waveforms ranges from 10 to 20 ns, depending on the four different phases used to clock the interfacing registers. Hence, overall, this implementation shows that the proposed design scheme successfully retrieves all the data elements sent by the four-times faster sender module, along with a phase shift of more than $(n/2)T_1$.

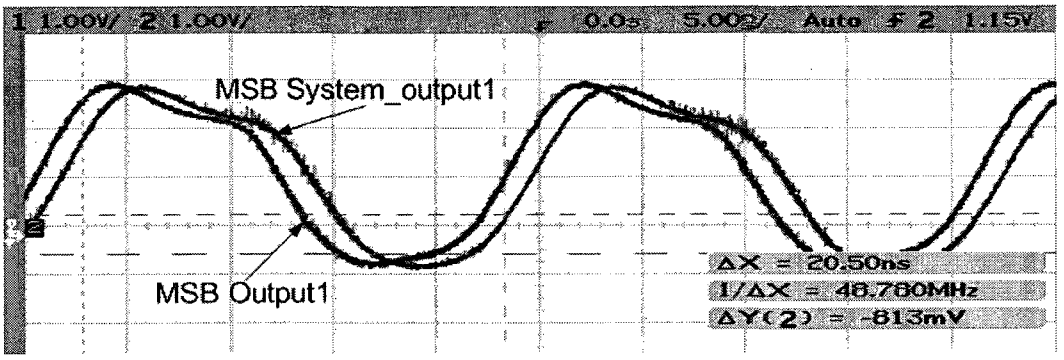


Figure 7.15. Waveform of a prototype FPGA implementation of the proposed F-to-S system: Signal at MSB of Output 1 received by MSB of System_output1

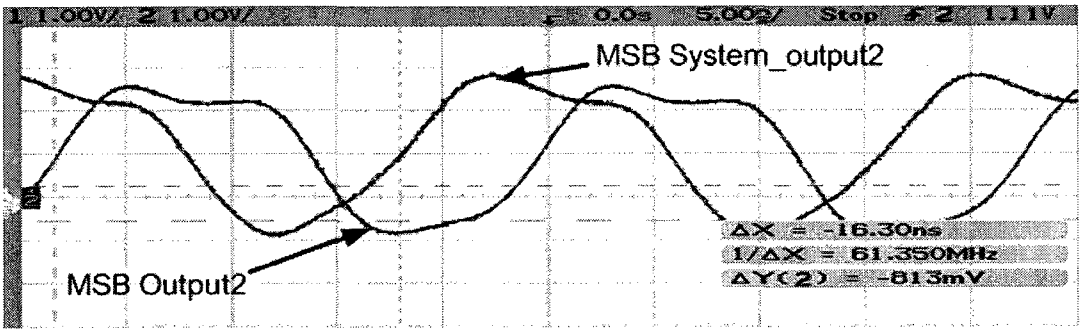


Figure 7.16. Waveform of a prototype FPGA implementation of the proposed F-to-S system: Signal at MSB of Output 2 received by MSB of System_output2

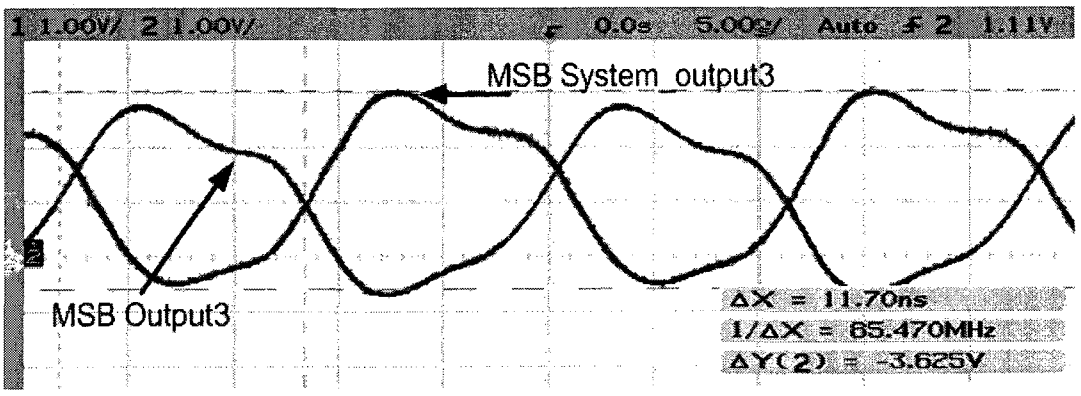


Figure 7.17. Waveform of a prototype FPGA implementation of the proposed F-to-S system: Signal at MSB of Output 3 received by MSB of System_output3

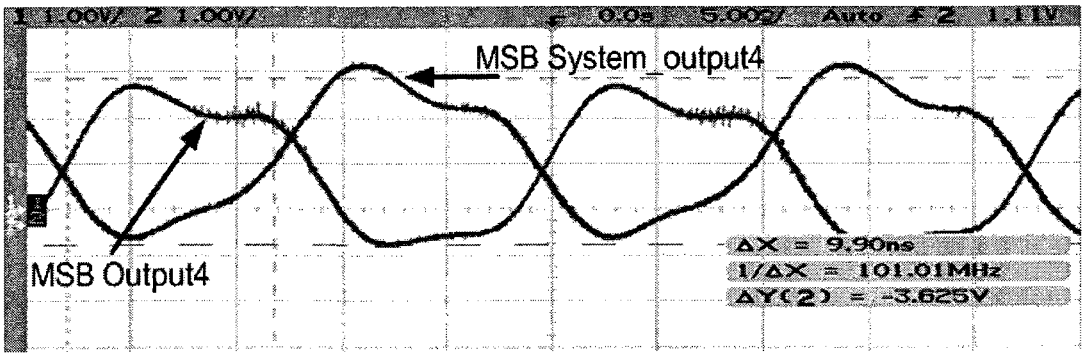


Figure 7.18. Waveform of a prototype implementation on FPGA of the proposed F-to-S system: Signal at MSB of Output 4 received by MSB of System_output4

7.7 Summary And Discussions

In this chapter, two different cases of point-to-point communications for MCDs are addressed: first, when the terminating modules communicate at the same frequency and, second, for fast-to-slow systems. The timing constraints for all the possible cases of unidirectional and bi-directional communications in the proposed solutions have been mathematically established. It is observed that skew tolerance increases linearly with the number of interfacing registers, which is a classical result for such interfacing methods.

The proposed technique is insensitive to clock-data delay mismatches. Also, it is proven mathematically (and supported with simulations and through a prototype implementation) that the clock skew is absorbed in the interfacing registers. It is also validated through simulations that the terminating modules clock frequency limitation is independent of the clock skew (at a hardware cost that grows with the length of the tolerated skew).

Gate level simulations were performed for different clock frequencies using the TSMC 180nm technology library. These results are in full compliance with the analytical results.

Comparison with the conventional design shows that, a tolerance in skew of up to $n/2$ or n clock cycles is achieved when the clock skew magnitude $|t_{sk}|$ is <= $(n/2)T_1$ and when it is between $(n/2)T_1$ to nT_1 respectively. Prototype implementations of the proposed systems were also done using a Virtex-II Pro FPGA from Xilinx. Back-annotated simulation results confirm the validity of the proposed design with the terminating module working at 250 MHz and subject to skew of 12 ns (> $(n/2)T_1$) in either orientation. Hardware implementation of a fast-to-slow system verifies the proper functionality of the design under a timing (skew) constraint of 12 ns (> $(n/2)T_1$), where the sender module works at 350 MHz and the frequency at the receiver module is 87.5 MHz. A natural extension of this design is to provide an interface for complex frequency ratios between the communicating modules. In the next chapter details of the solution for the communicating modules that have a frequency ratio of coprime number is described.

Chapter 8: Skew Tolerant Synchronous Interface for Modules Having Rational Frequency Ratio of Coprime Numbers

In the preceding chapter, an interfacing methodology was described that allows skew tolerant communications between different modules in multiple clock domains (MCD) running at the same frequency (or integer multiple frequencies) and non-aligned phases. However, it is often the case that the two communicating modules have frequencies such that their ratio is a rational number and that they are coprime. Another issue, which has recently gained attention, is the non-uniform delays across chips in modern DSM era, which is mainly caused by the thermal variations observed at run time due to the dynamic changes in the workload in chips that consist of multiple processors within a SoC called multiprocessor Systems on Chips (MPSoCs) [93]. In this chapter, the skew tolerant design, presented in the preceding chapter, is extended for such systems that have rational ratio frequencies (of coprime numbers) between the module clocks. The methods proposed in this chapter utilize a similar concept of phase adjustment, as was used in the preceding chapter, to tolerate clock skew. However, for the case of rational ratios of coprime numbers, the clock-scheduling mechanism is more involved and requires a clock-scheduling algorithm. The first section of this chapter discusses this new static clock-scheduling and its implementation. Furthermore, appropriate measures are also required to

manage the run time variations in clock phases. The second section of this chapter elaborates the dynamic clock-scheduling algorithm and its implementation to handle run time variations in clock phases.

The general assumptions are similar to the ones considered in the preceding chapter, i.e. the clocks for the communicating modules are derived from a common source and the frequency ratio between the communicating modules is a rational ratio of two coprime numbers.

8.1 Static Clock-scheduling Methodology

This section discusses the static clock-scheduling algorithm and a complete methodology which describes communication between the modules working at rational frequencies (ratio of coprime numbers) while the maximum phase difference is known in advance.

8.1.1 What are the Limitations of Existing Designs for Rational Clocking?

In this section we emphasized the challenges faced by the designers to make robust interfaces for modules having rational frequency ratio of coprime numbers. Conventional design schemes, as discussed in section 2.2, have limitations in accommodating dynamic changes in the timing behaviour of the system. Also most of the high performance conventional solutions suffer from data signal delay mismatch issues due to the

introduction of strobe signal. On the other hand, other solutions that resort on FIFO and synchronization mechanism for control signals introduce long latency and fail to follow conventional design flow. Furthermore in high performance techniques the maximum achievable clock period of the IP modules is dependent upon the clock skew. Moreover, along with these drawbacks, all the above solutions also have one common limitation: their frequency of data transfer cannot exceed the frequency of the slowest module in the interface. Hence these drawbacks instigated need of a high performance design which can be hardware efficient while keeping the interfacing scheme skew tolerant and allow faster module to communicate slower module at the pace of faster module.

As was the case in preceding chapter, the proposed design does not involve a strobe signal; rather, based on *a priori* timing information, phases of the interfacing registers are adjusted. The delay mismatch problem is resolved by avoiding the use of a strobe signal. Using a cyclic clock-scheduling scheme and a wider data bus, this technique allows the faster module (among the interacting modules) to send or receive data at its own data rate. The proposed technique introduces interfacing registers that absorb the clock skew, hence it allows the maximum clock period of the IPs to be independent (virtually) of the clock skew at the expense of some hardware. It also uses multiple phases of the clocks to manage bandwidth in a manner that is similar to one of previous work [94].

8.1.2 Skew Tolerant Design for Rational Clocking

This section provides an overview of the proposed skew tolerant design system and the algorithm for clock-scheduling, subsequent sections deals with the details of different aspects of the proposed design. Figure 8.1 shows an implementation of the proposed

design. This design exploits knowledge of the *a priori* timing information regarding the clock frequency ratio of the communicating modules. The system is composed of a sender module, a receiver module, two sets of interfacing registers, and control units for those registers. The interfacing registers are in two groups: one that is controlled using the sender clock (or clocks related to the sender clock) and one controlled by the receiver clock (or clocks related to the receiver clock). Using two groups allows absorbing the phase difference between the two clocks, thus providing clock skew tolerance.

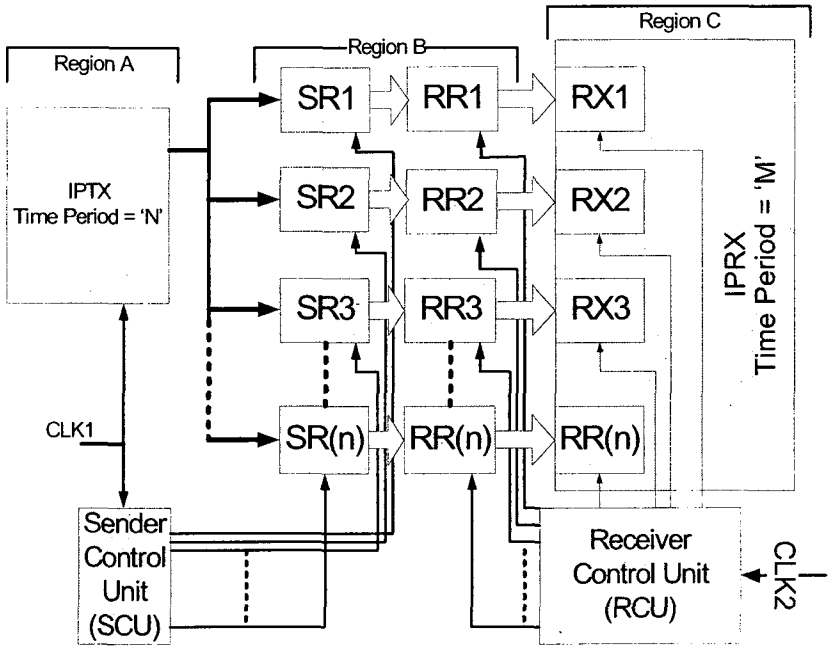


Figure 8.1. Block Level Description of the Proposed Hardware Design

As was the case in the previous chapter, for ease of analysis, Figure 8.1 is divided into three regions. These regions are defined based on their clock source. Region A only involves components that are clocked by the sender clock or CLK1. Similarly, region C consists of components that are clocked by the receiver clock, CLK2, or the clock that is

derived from CLK2. Region B comprises interfacing registers that are driven by clocks from each clock source, CLK1 and CLK2.

8.1.3 Algorithm to Generate Cyclic Phase Mapping

Since the frequencies of the two modules are related by the ratio of two co-prime numbers, their exact phase mapping does not follow a simple regular pattern and thus, it is difficult (if not impossible) to come up with a closed form mathematical relation for phases in the two clock domains. To resolve such an issue, usually an algorithmic design approach is utilized. Hence, in this section, a complete set of rules is provided, in an algorithmic form, to generate a cyclic phase mapping between the sender and receiver clock phases that allows data transfers to occur at the right pace for both the sender and the receiver.

Without loss of generality, it is assumed that the sender and receiver clocks, CLK1 and CLK2, have frequencies that are co-prime of each other. Otherwise, this frequency ratio is simplified until the quotient is the ratio of two co-prime numbers and the simplification factor can be used to scale the solution. The time period of CLK1 is N and that of CLK2 is M. For brevity, the case where $M > N$ is studied, therefore data is transferred from a fast module to a slow module. Analysis for the converse case is a trivial extension of this work. In addition, worst-case flow control is assumed, i.e. the sender sends data at every clock cycle. Any deviation from this flow control assumption will lead to a more relaxed hardware implementation. It is also assumed that the clock can be divided into n pulses where $n = ceil(M/N)$. Finally, it is assumed that a common reset signal initiates the

communication between the modules. Steps of the proposed phase mapping algorithm are elaborated as follows:

Step 1: Let, $n = ceil(M/N)$.

Step 2: The receiver control circuit provides n different phases of the slow clock. The delay between two successive phases is almost M/n.

Step 3: Because M and N are co-prime numbers, two clock edges coincide again after MN time units. In order to allow the (fast) sender to send data at every clock edge, a clock-scheduling technique is required. This technique makes sure that the clock-scheduling is periodic after every MN time units. The following PHA and PHB sets denote the clock edge instances for CLK1 and n phases of CLK2, respectively,

$PHA = \{0, N, 2N, \ldots (M-1)\, N\}$; and similarly, the receiving clock edges are,

$PHB = \{PHB^1, PHB^2, \ldots PHB^n\}$, where elements of PHB are:

$PHB^1 = \{[0], [M], [2M], \ldots [(N-1)M]\}$

$PHB^2 = \{[M/n], [M+M/n], \ldots [(N-1)M + M/n]\}$

and so on until PHB^n, which is defined as follows,

$PHB^n = \{[(n-1)M/n], [M+(n-1)M/n], \ldots [(N-1)M+(n-1)M/n]\}$

Step 4: Map the clock edges of set PHA onto edges of PHB so that data associated with each sending-end clock edge is safely latched by the mapped edge of one of the n receiver-end clock phases, exploiting the *a priori* timing information to meet the minimum required delay.

A total of M CLK1 pulses must be mapped onto Nn slow phases of the clock. This proposed clock-scheduling algorithm accommodates skew and various other timing non-

idealities, associated with DSM processing technologies, by leveraging the worst-case *a priori* timing information of the system. In order to keep this design scheme periodic, the scheduling scheme is repeated after *MN* clock cycles. The scheduling technique is further elaborated with an example in the next section.

8.1.4 Practical Example of Clock-scheduling Utilizing the Proposed Algorithm

Suppose the sender module is fast and is working with a clock period equal to five time units ($N = 5$), and the receiver module is working with a clock period equal to eleven time units ($M = 11$). Following **Step 1** of the algorithm, $n = ceil(M/N) = 3$. **Step 2** suggests that three phases of CLK2 are required, with approximately $M/n = 3.66$ *time units*. **Step 3** defines the two sets of clock edges. Following, the proposed technique used to generate set PHA and PHB in the preceding section, the two sets of clock edges for the given example are:

PHA = {0, 5, 10, 15, 20. 25, 30, 35, 40, 45, 50, 55} and

PHB= {PHB^1, PHB^2, PHB^3} as $n = 3$;

PHB^1 = {0, 11, 22, 33, 44}

PHB^2 = {3.66, 14.66, 25.66, 36.66, 47.66}

PHB^3= {7.32, 18.32, 29.32, 40.32, 51.32}

Following **step 4**, a mapping of the sender and receiver clock edges is performed. The proposed method utilizes the *a priori* timing information to decide upon mapping clock edges. The following criteria are required for safe data latching:

1) The receiver mapped clock edge must happen later than the corresponding sender mapped clock edge.

2) Due to timing discrepancies, phase mismatches and various non-idealities in DSM processing technologies, it is necessary to keep a minimum distance between the mapped values. This minimum distance is dependent on the minimum tolerable delay between the sender and receiver clock edges, further elaborated in the explanations of equation (8.1).

Figure 8.2 shows the mapping of clock edges for two different delay tolerances for this particular example. PHB^1, PHB^2 and PHB^3 represent three different phases of the receiver-end clock (represented by B^1, B^2, and B^3 in Figure 8.2). The initial delay, required to latch the 0^{th} pulse of sender clock, is termed as the offset in Figure 8.2. The minimum offset value increases with the required delay tolerance. As shown in Figure 8.2, those timing offsets are, respectively, 3.66 time units and 7.32 time units for minimum delay tolerances of 2 time units and 4 time units. Note that for all related phases identified by arrows, the time difference between the respective receiver and sender is larger than the specified delay tolerance. Because M sending-end clock edges can be mapped to Nn receiving-end clock phase edges, there are some redundant clock edges. The number of redundant clock edges is $Nn - M$. On the timing axis in Figure 8.2, there are $(15 - 11) = 4$ such redundant clock phase edges at the receiving clock, indicated by ovals.

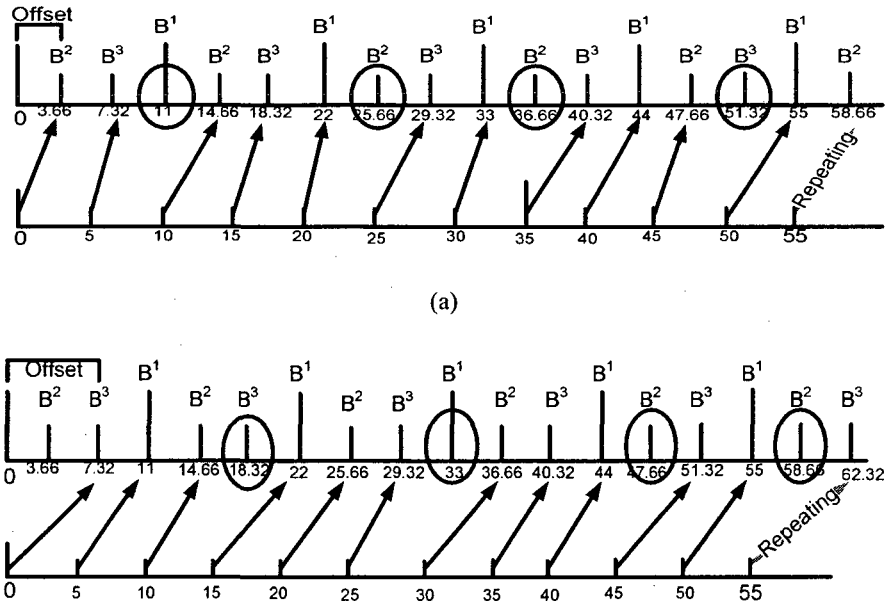


Figure 8.2. (a) Clock-scheduling with Minimum Delay tolerance of 2 time units (b) and of 4 time units

8.1.5 Hardware Implementation

As previously explained, Figure 8.1 shows the block level representation of a hardware implementation of such designs. This section further elaborates on each individual block.

Region A

Region A of the hardware consists of the transmitting-end IP module, which has a clock period of *N* time units, Sending-end Registers, which are denoted SR(X), and a Sender Control Unit (SCU).

Sender Control Unit (SCU): Figure 8.3 shows an expanded view of the SCU. It contains three units. A one-hot state machine generates *n* clock pulses. These clock pulses pass

through a switch to SR(X). Based on the static scheduling described in the preceding section, this switch connects the clock pulses, from the one-hot state machine to the appropriate SR(X). The control bits for the switch are generated from a second state machine. This state machine exploits the scheduling knowledge of the system and changes the switch outputs as desired.

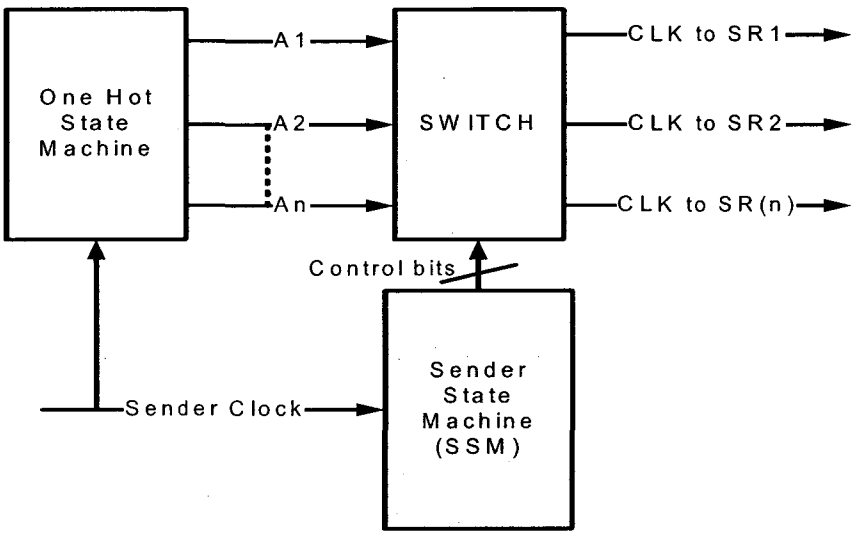


Figure 8.3. Sender Control Unit (SCU): Hardware Realization

SR(X): Region A consists of *n* sending-end interfacing registers. These registers are clocked by the SCU based on the static scheduling technique described in the preceding section.

Region C

Region C of the hardware implementation shown in Figure 8.1 consists of the receiving-end IP module (IP RX), the Receiver Control Unit (RCU), the Receiving-end interfacing Registers RR(X), and Receiver-end registers, RX(X).

Receiver Control Unit (RCU): This section illustrates the hardware realization of the Receiver Control Unit (RCU), which is shown in Figure 8.4. The clock phase generator produces *n* different phases of the receiver clock, CLK2, which has a clock period *M* with *M>N*. The switch unit selects one of the *n* phases of CLK2 for each of the RR(X). The phase selection depends on the control signals. These control signals are generated from a Receiver State Machine (RSM) based on the clock-scheduling.

RR(X): Similar to region A, region C consists of *n* registers RR(X). They are clocked by the RCU as shown in Figure 8.1.

RX(X): There are *n* Receiving-end registers and they are denoted RX(X). These registers can be clocked by one of the phases from the set of clock phases 'PHB'. During each *MN* macro cycle, a total of *N* occurrences of a particular phase are possible. Each occurrence should latch only the valid data available since the last occurrence of the phase.

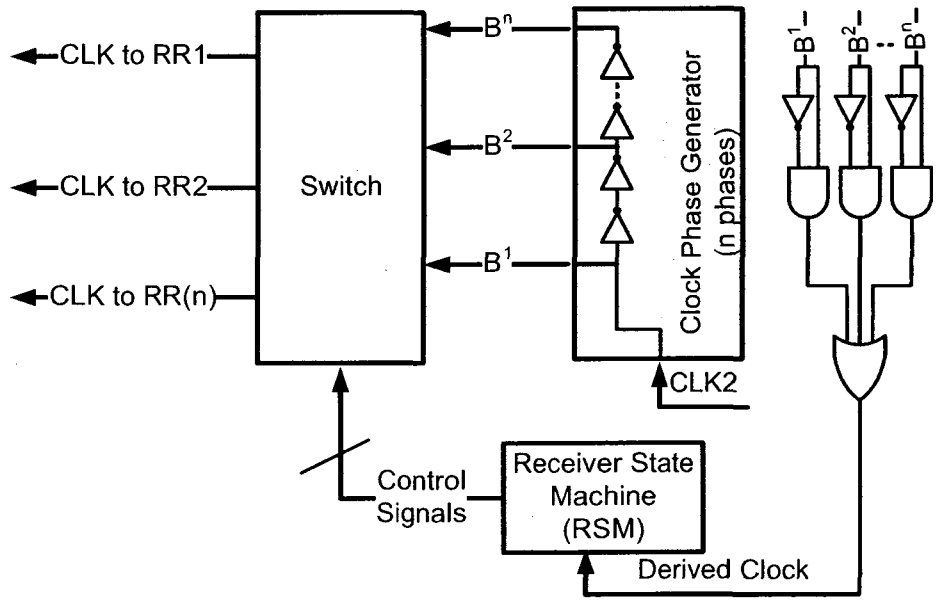


Figure 8.4. Hardware Realization of the Receiver Control Unit (RCU)

Region B

This region contains two stages of interfacing registers, SR(X) and RR(X). These interfacing registers are clocked by different phases of CLK1 and CLK2, following the clock-scheduling shown in Figure 8.2. This region absorbs the clock skew and hides the skew constraints from timing relations of Regions C and A. The worst-case timing constraint for clock sequence scheduling should respect the following conditions:

$$Min\_Delay > t_{cq} + t_{SU} + t_{SK} + 2t_J \qquad (8.1)$$

$$M/n > t_{RCU} \qquad (8.2)$$

$$Offset = (M/n)\ (ceil\ [Min\_Delay/M/n]) \qquad (8.3)$$

where Min_Delay is the minimum tolerable delay between the sender and subsequent receiver clock edges based on *a priori* timing information. t_{cq} and t_{SU} are the clock-to-Q delay of SR(X) and setup time of RR(X), respectively. t_{SK} is the clock skew and t_J is the jitter associated with the clock on either side of the nominal clock edge. t_{RCU} is the delay associated with the RCU. Note that (8.1) involves only timing values related to the interfacing registers. Relation (8.2) shows that the period of CLK2 is not a function of the clock skew, hence, the clock skew can completely be absorbed in the interfacing registers. Once Min_Delay and *M/n* are obtained, the offset that gives minimum latency can be calculated using relation (8.3), where the definition of offset is same as defined earlier for Figure 8.2 in section 8.1.4.

8.1.6 Simulation Results of the Design Example

Clock-scheduling for a *M/N* ratio of 11/5, shown in Figure 8.2, was simulated to validate the scheduling algorithm. The state diagrams of the sender and receiver state machines are shown in Figure 8.5. The sender state machine has three states, each state changes after 11 CLK1 pulses. Each state sends a particular control signal to the switch that determines the output pattern of CLK1 phases. Similarly, the receiver state machine has five states. The number of states is chosen based upon the utilized and skipped clock edges for the clock-scheduling shown in Figure 8.2a. Each state sends a different control signal to the switches and changes the output pattern of CLK2 phases accordingly.

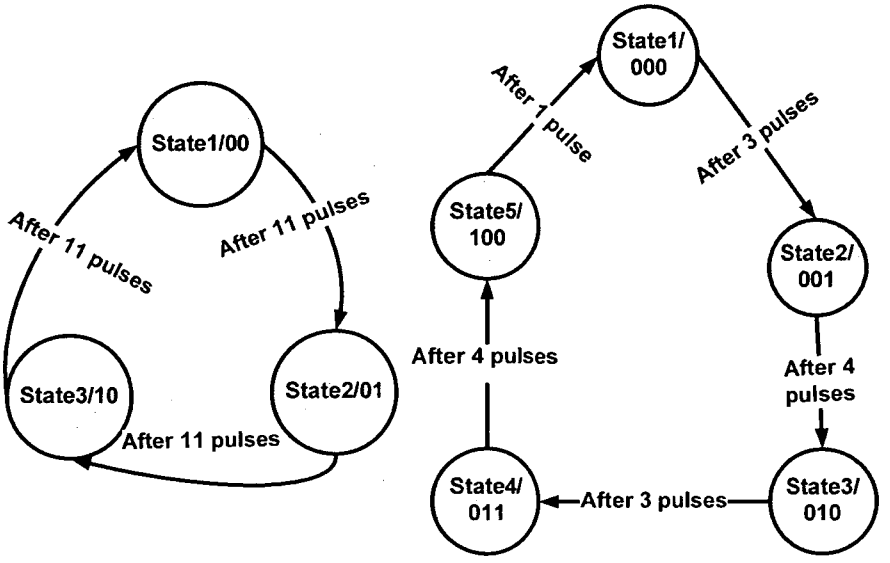


Figure 8.5. Sender State Machine (Left) and Receiver State Machine (Right)

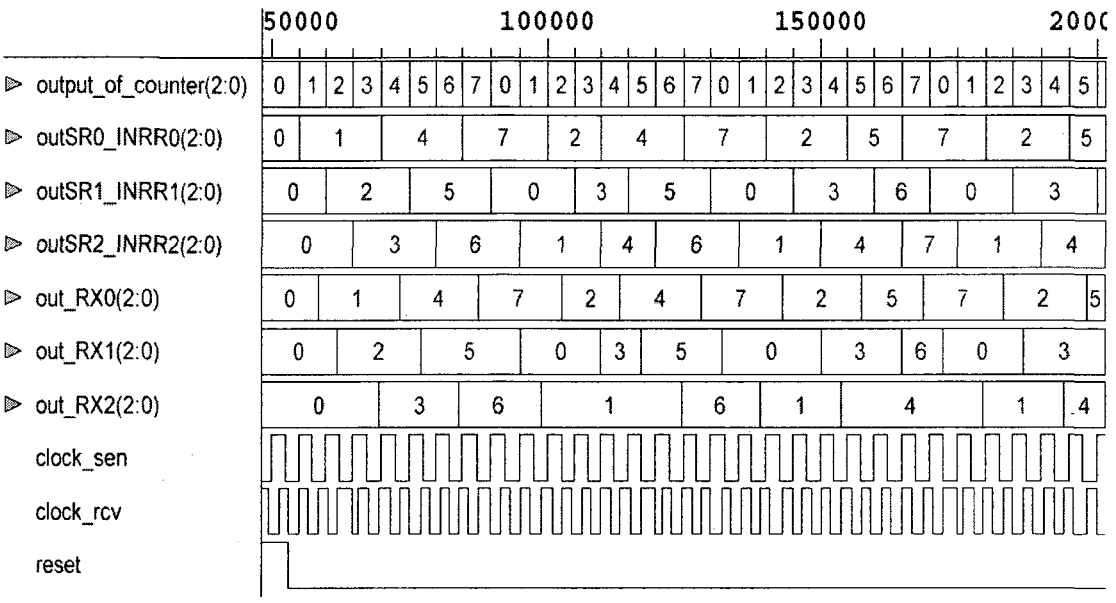


Figure 8.6. Simulation Results for Functional Verification of the design propsed in Figure 8.1

Simulation results, shown in Figure 8.6, validate the functionality of the design. The control signals, generated by the state machines (not shown in Figure 8.6), change the output pattern for each cycle. It is demonstrated in Figure 8.6 that, after every 11 sender-clock (clock_sen) cycles, the clock-scheduling scheme changes. This change appears on the output pattern when outSR0_INRR0 latches a count value of '4' for the second time. This phase adjustment is propagated to the output signals. In Figure 8.6, it can be seen that out_RX2 signal latches the count value '1' (the fourth count value on this data line from the left) for a longer duration. This is to accommodate state switching in RSM, leading to a periodic clock-scheduling in the proposed phase adjustment technique. Overall, it is seen that the data generated at a faster rate (with a 5-time-unit period) is completely and safely transferred to the slower clock domain (11-time–unit period), without slowing down the faster module.

8.2 Dynamic Clock-scheduling Methodology

This section extends the static clock-scheduling algorithm to accommodate dynamic phase variations due to run time variation in delays. Here, a dynamic clock-scheduling algorithm is proposed and a complete methodology is described for communication between modules working at rational frequencies (ratio of coprime numbers) with a varying phase difference between the communicating modules. The design was successfully synthesized using Xilinx's Virtex-II Pro FPGA technology.

8.2.1 Motivation for Proposing a Dynamic Clock-scheduling Methodology

As we move deeper into the DSM era, power and heat management are becoming key issues across most application segments [1]. In particular, high-speed applications are more affected due to the non-uniform delays generated across chips by dynamic thermal variations, hence some modifications in static clock-scheduling algorithms are required to accommodate non-uniform delays. For example in [86], it was shown that for every 20°C increase in temperature, the Elmore delay for the long global interconnects increases by approximately 5%-6%. Assuming a 25 °C nominal temperature, this translates into 30%-35% delay variation between the nominal and peak temperatures in integrated circuit implemented with 90-nm CMOS. In order to mitigate the effects of on-chip temperature variations on delay, several solutions have been proposed in the literature (discussed in detail in Chapter 2). For example, in multiprocessor architectures, stop and go [96]

policies are the most commonly used to reduce peak temperature. In such policies, the state of the core needs to be saved in memory before powering down. Also, a significant current is drawn when the core is "awaken" again. Also it is common to observe non uniform load in these MPSoC due to irregular "stop" and "go" for different processors in different points in time. This leads to larger temperature variations which then translate into varying delay during run time hence causing varying clock skew.

From the above discussion it is obvious that an extension to the clock-scheduling scheme for modules having rational frequency ratio of coprime numbers is required to handle this run time delay variations due to thermal gradients. Therefore, in the subsequent sections a dynamic clock-scheduling algorithm is described which allows dynamic clock phase changes, depending on runtime clock skew variations that arise from thermal gradients. The method described in Section 8.1 follows a relatively pessimistic approach of worst-case design that results in performance penalty. This is particularly true if the worst-case scenario is rarely encountered. In addition, designs using *a priori* information are not adaptable to real-time changes in clock skew that can occur due to run-time temperature variations and may lead to perturbations especially for the systems with continuous data flow.

8.2.2 Dynamic Clock-scheduling Algorithm

Keeping the same assumption as described in Section 8.1.3, the proposed dynamic clock skew scheduling method embeds an algorithm to map clock phases from the sender to the receiver domain. It operates by examining several possible phase variations. This section describes the proposed algorithm that maps different clock phases of the two clock

domains, to allow a reliable communication between them. The following algorithm serves two purposes: first, it maps the sender and receiver clock phase edges and secondly, it checks whether the dynamic clock-scheduling will provide any performance improvement compared to the worst-case design techniques. Figure 8.7 describes this algorithm, where *Max_Difference* is the maximum time difference between two mapped clock edges for a particular minimum delay tolerance constraint (*Min_Delay_Tolerance*). The variables i_1 and i are dummy variables. *Average_latency* is the average time duration for the receiver to receive data after being transmitted by the sender for a particular value of *Min_Delay_Tolerance.*

In order for the reader to better understand the proposed algorithm, the following example is given, based on the algorithm given in Figure 8.7. Let $M = 11$ and $N = 5$, which leads to $n = 3$. Let *Min_Delay_Tolerance* = 4, then during the first iteration of the algorithm, *Max_Difference* = 4, $i_1 = 1$, $i = 1$. Following this, the first IF condition is true since $(i-i_1)N$ is equal to 0, which is less than $(M/n)i$ that results in a value of 3.66. Then, the second IF condition evaluates to false as 3.66 is less than 4. Before reiterating the inner FOR loop, i_1 is incremented to 2. A complete run of this algorithm produces the following values for the variable *Min_Delay_Tolerance* (shown by arrows in Figure 8.8) with duration of 4 and 5 time units, with 5 being the worst-case scenario. The results are included within closed brackets denoted by $(x_1,\ldots x_i)_4$ and $(x_1,\ldots x_i)_5$, respectively. It should be noted that the *Average_latency* shows that the worst-case design will lead to a poor latency.

Sender_Clock = {(0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50)$_4$, (0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50)$_5$}

Receiver_Clock = $\{(7.32, 11, 14.66, 22, 25.66, 29.32, 36.66, 40.32, 44, 51.32, 55)_4, (7.32, 11, 18.32, 22, 25.66, 33, 36.66, 40.32, 47.66, 51.32, 55)_5\}$

Average_Latency = $\{(5.67)_4, (6.67)_5\}$

```
for Min_Delay_Tolerance = 1:N (discrete integer values)
Max_Difference = Min_Delay_Tolerance
i1 = 1
    for i = 1 : (Nn)
        if {(i – i1) N < (M/n)i} then
            if {(M/n)i – (i – i1) N } ≥ Min_Delay_Tolerance then
                " The sender data of (i – i1)N can be latched at
                (M/N)i: Hence an arrow can be drawn from Sender to
                        corresponding receiver clock edge (as shown in
                        Figures 8.2 and 8.8)
                if {Max_Difference < ((M/n)i – (i – i1) N )} then
                    Max_Difference = {(M/n)i – (i – i1) N }
                else
                end
            else
                i1 = i1+1
            end
        else
            i1 = i1+1
        end
    end
    Average_latency[Min_Delay_Tolerance] = (Min_Delay_Tolerance +
                                            Max_Difference)/2
end
```

Figure 8.7. Algorithm for Dynamic Clock Phase Mapping

8.2.3 Implementation of Clock-scheduling For Variable Skew Tolerance

In this section, the means of transforming the worst-case design scheme of Section 8.1 to the dynamically skew tolerant design scheme are presented. It is assumed that the skew is slowly varying and the switch to a new scheduling scheme occurs after the current scheduling cycle finishes thus, this approach does not allow the scheduling to switch to a

different delay tolerance value (shown as *Min_Delay_Tolerance* in Figure 8.7) in the middle of a scheduling cycle. Figure 8.8 depicts the clock phase mapping of the sender and receiver clocks for the values given by the example at the end of Section 8.2.2 (delay tolerance of 4 and 5 time units). Figure 8.8 is divided into three different sections. The top and bottom sections show the static clock-scheduling following the algorithm developed, depending on different worst-case minimum-delay requirements of the systems. The middle section of Figure 8.8 depicts how the system may adapt dynamically to changes brought to meet minimum delay requirements. The hardware implementation to accommodate such a technique requires a Phase Margin (PM) signal that indicates phase mismatches between the two clock phases of the communicating IPs, and hence, determines the change in minimum delay tolerance. This signal is an additional input signal to RCU of Figure 8.1 and is not shown in that figure. At the block level, this is the only difference between the two implementations. In the example depicted in Figure 8.2, the PM signal indicates a warning that the minimum delay tolerance has to change from 4 ns to 5 ns. The PM signal is checked at the beginning of each clock-scheduling cycle, which is after every $M \cdot N$ time units (55 time units in this example). Therefore, the PM signal indicates the overall changes in phase within a scheduling cycle. If any system experiences faster phase shifts, then this information has to be checked more frequently, this case will be the subject of future work.

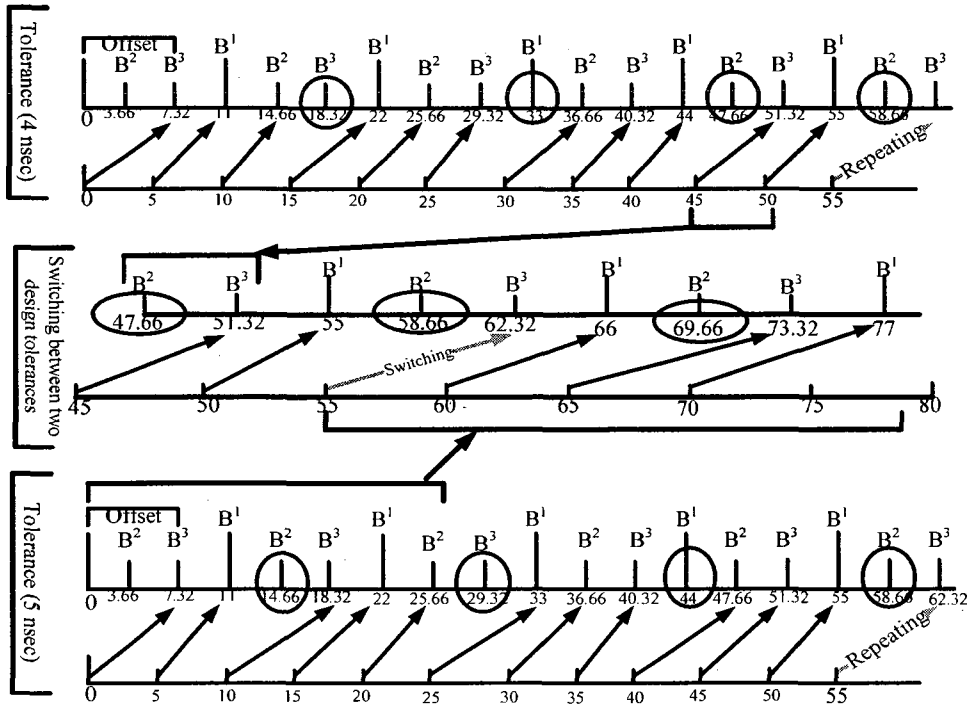


Figure 8.8. Clock Phase Mapping (Magnifying the switching process)

8.2.4 Hardware Implementation of Dynamic Clock-scheduling Methodology

As stated earlier, at the block level, this solution is very similar to that of Figure 8.1 except for an additional signal in RCU, which is named as PM signal. The PM signal is checked at the end of the current scheduling cycle to decide whether switching to a new phase scheme is needed, depending, for example, on a slowly varying skew as mentioned in Section 8.2.3. As the emphasis of this thesis is on the interfacing techniques, therefore the generation of the PM signal is not treated in this chapter (a work on the development of a design to obtain a warning on phase variations was developed as a precursor of this thesis and provided in Appendix, this work was also published [30]). The main

architecture of the Sender Control Unit (SCU) and Receiver Control Unit (RCU) remains the same as shown in Figure 8.3 and Figure 8.4, respectively. The main difference is the implementation of state machine of the RCU. The following steps are followed, in order to obtain the state diagram, shown in Figure 8.9, of the receiver state machine that accommodates phase variations dynamically.

***Step 1*:** All possible clock edge vectors under all possible minimum delay tolerance values are obtained using the algorithm proposed in Figure 8.7.

***Step 2*:** Utilizing the vectors obtained in step 1, generate a list of selected and discarded clock edges (e.g. the circled edges in Figure 8.8 are the discarded ones).

***Step 3*:** Assign the used clock edges to the RR(X) in sequence.

***Step4*:** Check the discarded edges and allocate them to registers where these edges will not lead to any fault (utilizing the knowledge of clock-scheduling).

***Step5*:** Check the PM signal at the start of every clock-scheduling cycle. Decide which of the minimum delay tolerance values is best suited for the upcoming scheduling cycle (denoted as Initial State in Figure 8.9); this is done within the state machine.

***Step 6*:** Repeat Step 5 at the beginning of every clock-scheduling cycle.

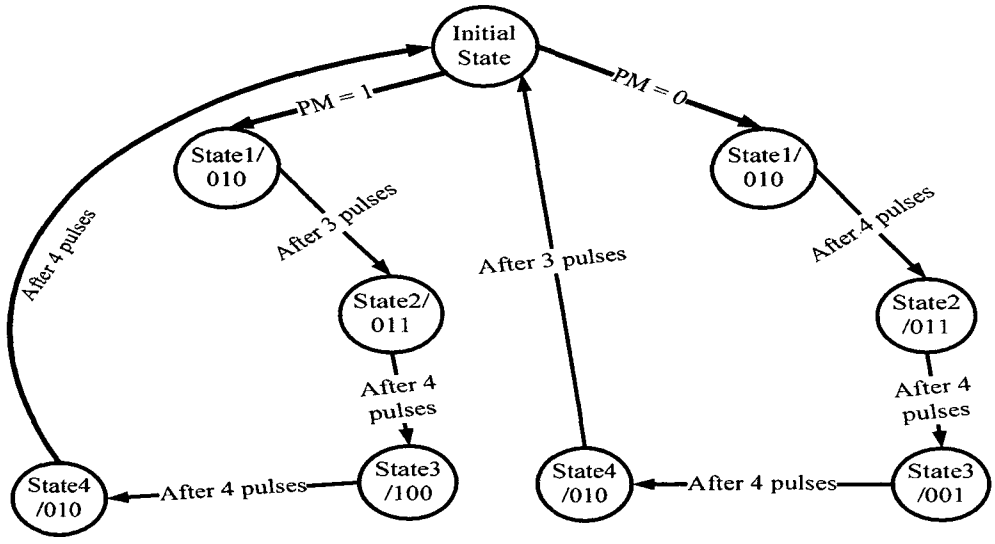


Figure 8.9. State Diagram for RCU

8.2.5 Simulation Results

The dynamic clock-scheduling algorithm provided in Figure 8.7 was first implemented in MATLAB, and the clock-scheduling data graph was obtained as shown in Figure 8.8. Let us consider a case where the minimum and maximum delay tolerances are set to 2 ns and 5 ns, respectively. According to Table 8.1the best case resulting latency is 2.67 ns and the worst-case is 6.67 ns. If the system switches between the two extreme scenarios with some probability, then the resulting latency is reported in Table 8.1. As can been seen from that table, up to 60% improvement in latency is provided with the proposed solution, compared to the worst-case scenario.

The proposed design was implemented using the clock-scheduling shown in Figure 8.8 and the state diagram of the receiver state machine (within RCU) shown in Figure 8.9. The design was synthesized using the Xilinx ISE suite for Virtex-II-Pro technology.

Back-annotated simulations were performed with ModelSim using an SDF (Standard Delay File) file generated by the Xilinx Implementation tools.

TABLE 8. 1. Latency improvement comparison

Best to Worst-case Ratio	***Best Case***	***10:90***	***30:70***	***50:50***	***70:30***	***90:10***	***Worst-case***
Average_Latency (ns)	2.67	3.07	3.87	4.67	5.47	6.27	6.67
% Improvement	60	54	42	30	18	6	0

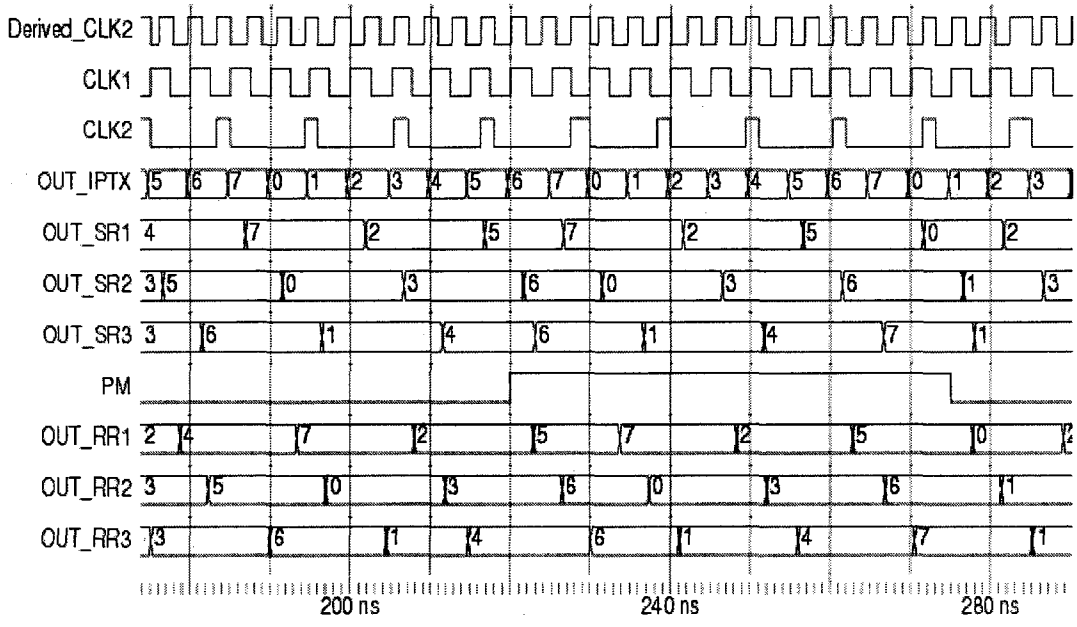


Figure 8.10. Back-annotated Simulation Results of the synthesis of our adaptive interfacing scheme (Xilinx Virtex-II-P FPGA is used for synthesis)

Figure 8.10 shows the back-annotated simulation results that verified the functionality of the design. In this simulation, the faster module (IP TX) is running at 200 MHz, while the slower module (IP RX) receives data with a 90.9 MHz clock. In Figure 8.10, it is shown that depending on the PM signal status, the state machine switches between different optimai clock phase mappings. The PM signal changes value depending on the minimum delay tolerance value. For the simulations, the delay tolerance values are set to

4 ns and 5 ns, for PM equal to 1 and 0, respectively. It is seen that after each change of state in the PM signal, the successive clock-scheduling cycle started working with the new scheduling scheme, and transparently start latching the data. The different sequence of delays, between consecutive selected and unselected phases are evident through the width variation of each count value. This can be observed in Figure 8.10, for example with signal *OUT_RR3*, where between 200 and 240 ns of simulation time, the register holds the value '1' for a shorter duration than the value '4'. As the data is completely reproduced at the *output_RX(X)*, this simulation confirms that the scheduling scheme is seamless and that the functional verification was successful.

8.3 Summary and Discussions

In this chapter, a novel design technique is proposed for interfacing multiple clock domains having clock frequencies expressed by ratio of coprime numbers. The introduction of a wider bus allows the slower module to handle safely the fast data without reducing the speed of the fast module. The previous chapter provided detailed illustration of clock skew absorption due to interfacing registers. Similar mathematical relations are obtained in this chapter for these interfaces (see Eq. 8.1 to 8.3), which demonstrates that the minimum time period of the terminating modules are not dependent on the clock skew.

In the second section of this chapter, it is established that, in modern DSM technologies, the new SoC architectures (e.g. MPSoC) are more vulnerable to dynamic thermal variations. These variations affect the timing relations among the IP modules in

real time. The modified clock-scheduling technique adjusts dynamically to slow phase variations in real-time.

Complete design methodologies, with clock-scheduling algorithms, are elaborated. A functional verification of these methodologies is also provided. The dynamic clock-scheduling scheme is functionally verified through back-annotated simulation results, using Xilinx Virtex II Pro technology. The results prove the validity of the design at the most error prone locations in the timing diagram. A further advantage of this design scheme is an overall improvement in performance of up to 60%, for the illustrated example, as compared to the worst-case static scheduling scheme.

Conceptually this methodology is similar to the dual-port memory technique [62]. However, the major difference between the two methodologies is that the proposed technique provides a fine control mechanism for safely reading and writing the data. This technique requires only registers hence it does not require the memory space and avoids long memory access times. In comparison to FIFO based schemes [25], [26], [78], our scheme does not impose separate synchronization requirement and FIFO controlling protocols. Furthermore, our design technique allows the faster module to communicate with the slower module without slowing down the faster modules, an attribute not easily attainable using FIFO based or dual port memory techniques.

Chapter 9: Conclusions and Future Work

9.1 Conclusions

In this thesis, we advocated that the optimal choice of inter-module communication method in MCD is context dependent. A designer can choose, based on the timing constraints and design requirements, what interfacing techniques best suit a particular design context. This work provided a new dimension to the design challenges associated to both synchronous and asynchronous interfacing methodologies with respect to process variations and other non-idealities that lead to timing uncertainties in modern DSM technologies.

In the context of the asynchronous paradigm, we characterized the crosstalk glitch propagation effect and proved that these glitches may lead to system malfunction under normal operating conditions. We proposed a pioneering work that provides a framework to the digital designer to analyze the crosstalk glitch propagation possibilities at the logic abstraction level. As our approach studies the effects on the constituent elements of the asynchronous interface, which are logic elements, our crosstalk glitch propagation

modeling technique is broadly applicable to a wide set of asynchronous design techniques. One of the benefits of this modeling approach is to identify the vulnerable primary input vectors and initial conditions, which can propagate the crosstalk glitches and lead to potential system malfunction. Once we know the vulnerable primary input vectors and/or initial conditions then, using the knowledge of the protocol used, we explain how to examine whether such a scenario is filtered out by the protocol or not. If it is not filtered out, then such a vector should be monitored. We applied this technique to representative circuits of two widely used asynchronous circuit design schemes, and we obtained the vectors for which the asynchronous interfaces are vulnerable to propagate glitches.

Motivated to introduce a mechanism to block glitch propagation if such conditions are bound to occur, we investigated further for a solution. Leveraging our crosstalk glitch propagation modeling approach, we suggested the introduction of auxiliary signals to the specific logic elements to prevent glitches from propagating. We proposed a crosstalk glitch gating solution that quenches crosstalk glitch propagation by blocking the glitches for a particular duration of time. A systematic methodology is presented to implement this technique. It is seen that this technique results in a complete removal of crosstalk glitch propagation. Two case studies, on representative interfaces of most widely used asynchronous protocols, are discussed and it is shown that our approach is broadly applicable to many asynchronous interface methodologies.

After making a significant contribution to the asynchronous design paradigm, we shifted our attention to the synchronous design paradigm. Here, we addressed the issue of timing non-idealities due to process variations in synchronous design paradigm. An all-digital skew-tolerant design scheme for high performance communication was proposed.

This design technique addresses several different cases including mesochronous design schemes, where the clock frequencies of two clock domains are the same but where the phase varies. Also, our design scheme is applicable if the two modules have an integer frequency ratio. We provided a comprehensive mathematical analysis of all the possible timing constraint scenarios. This mathematical analysis makes our design readily available to be embedded in any EDA tool library. This design scheme was implemented in a Xilinx Virtex II Pro FPGA using the Xilinx ISE design suite 9.2. Results for the hardware implementations are in compliance with the theory.

We enhanced the above mentioned methodology to accommodate communications between the two modules whose frequency ratio is a coprime number. Another extension of this methodology is proposed to accommodate dynamic phase variations that are expected to occur in modern DSM Multi-Processor SoCs due to thermal variations in the wake of sudden activation of a few (or even, many) processors from a dormant to an active state and vice versa. We provided a detailed algorithm to establish the clock-scheduling mechanism and proposed design guide lines which can be useful to design such a class of interface for many coprime frequency ratios. It was proven through mathematical relations that our proposed design scheme is more tolerant to clock skew than conventional design schemes. Furthermore, it was also mathematically formulated that this technique absorbs the clock skew in the interfacing registers, which in turn let the terminating modules work at relaxed constraints independent of the clock skew term. Another especial feature of such interfaces is that they allow a faster module to communicate with a receiver module without slowing down the faster module. Such a design approach is especially beneficial for SERDES (Serializer and Deserializer), an

intended application for our design group, and for burst data transfers. We proved the functionality of the design using back-annotated simulations using the Xilinx Virtex II Pro technology. Through one of the experiments (to implement the proposed technique to interface four times faster module to slower module at the physical hardware level), it is shown that this design technique allows the faster terminating module to communicate, without slowing down, at 350 MHz., under severe clock skew constraint. The frequency achieved is the maximum operating frequency of the given FPGA (Xilinx's Virtex-II Pro XC2VP30-7FF896), when it is made subject to severe timing constraint of more than 12 ns (this constraint is the maximum possible delay that can be applied in the aforementioned FPGA technology). This proved the fact that our design in fact weathers the clock skew quite efficiently without losing on performance. In another experiment, to verify our design technique for modules with rational frequency ratio of coprime numbers, it is shown that a faster module, running at 200 MHz, safely communicates, without slowing down, with a slower module which has a frequency of 83.33 MHz. Rational frequency ratio between the two frequencies was 5 to 11.

It can be concluded from this thesis that there is no fixed inter-module interfacing solution that fits to all the design scenarios in SoCs with MCDs. Instead designers must choose an interfacing methodology that suits the best for their particular design context, from a pool of solutions. Our thesis provided major contributions in identifying the upcoming challenges of modern DSM technologies that can potentially deter the performance of these interfaces. We provided novel design techniques in both design domains to address these issues. With high performance systems in growing demand, due to unprecedented advancements in the electronic industry in recent years, our design

solutions can play a major role in coping with the forthcoming issues of high performance and reliable system development.

9.2 Future Work

Asynchronous Paradigm: We intend to lead the proposed crosstalk glitch propagation modeling to the next level to make it a part of a design automation toolset. Another intended area of research is to extend this modeling approach for certain unpredictable scenarios. Such unpredictable scenarios will make this technique tolerant to extrinsic crosstalk effects, which may be due to soft errors, transition in external circuits, power surge etc.

Synchronous Paradigm: A novel implementation of a phase detection scheme was recently proposed by our group, which stems out of this research. Preliminary results are promising, further research is underway for its jitter and performance analysis. This phase detection will become part of a large framework that is intended as a solution to detect phase variations at run time and to dynamically adjust the clock phases of our proposed interfacing mechanism. This framework will allow communication among multi-processor SoCs in MCD with dynamically varying phases.

Another future direction of research is to extend the proposed interfacing methodologies to accommodate point-to-multipoint communications and GALS based Network-on-Chip (NoC) architectures.

REFERENCES

[1] Semiconductor Industry Association, "International technology roadmap for semiconductors," Executive Summary 2007, available at http://www.itrs.net/Links/2007ITRS/ExecSum2007.pdf

[2] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," in Proceedings of the IEEE, Vol. 89, No. 5, May 2001, pp. 665-92.

[3] H. Bakoglu, "Circuits, Interconnections and Packaging for VLSI", Reading MA: Addison-Wesley, 1990.

[4] S. R. Hasan, Y. Savaria, "Crosstalk Effects in Event-Driven Self-Timed Circuits Designed With 90nm CMOS Technology," in IEEE International Symposium on Circuits and Systems, May 2007, ISCAS 2007, pp. 629-632.

[5] P. Zarkesh-Ha, "Global Interconnect Modeling for a Gigascale System-on-a-Chip," Ph.D. thesis, Georgia Institute of Technology, February 2001. available at http://www.ece.gatech.edu/research/labs/gsigroup/publications/Dissertation_Payman.pdf.

[6] J. M. Rabaey, A. Chandrakasan, B. Nikolic, "Digitial Integrated Circuits; A Design Perspective: Second Edition," Prentice Hall Electronics and VLSI Series, 2003.

[7] Mark De Clercq and Radu Negulescu, "1.1-GDI/s Transmission between Pausible Clock Domains," in Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS 2002, Vol. 2, pp. II-768-II-771, 2002.

[8] J. Muttersbach, T. Villiger, W. Fichtner, "Practical Design of Globally-Asynchronous Locally-Synchronous systems," in Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems, April 2000, pp. 52-59.

[9] S. Moore, G. Taylor, R. Mullins, P. Robinson, "Point to point GALS interconnect," in Eighth IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2002, pp. 69-75.

[10] R. Dobkin, R. Ginosar, C.P. Sotiriou, "Data synchronization issues in GALS SoCs," in tenth IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2004, pp. 170-179.

[11] R. Mullins, S. Moore, "Demystifying Data Driven and Pausible Clocking Scheme," in thirteenth IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2007, March 2007, pp. 175-185.

[12] J. Bainbridge, S. Furber, "Chain: a delay-insensitive chip area interconnect," Micro IEEE, Vol. 22, Issue 5, September October 2002, pp. 16-23.

[13] ARM, Technical Specification: AMBA Specification, Doc No: ARMIHI-0011A, Issued: May 2001.

[14] IBM, Technical Specification: 32-bit Processor Local Bus – Architecture Specification, Doc NO: SA-14-2531-01, Issued May 2001.

[15] HyperTransport Technology Consortium, Technical Specification: Hypertransport I/O Link Specification, Doc NO: HTC2001021-0009-0022, Issued: Aug. 2003.

[16] W. J. Dally, and B. Towles, "Route Packets not Wires: On-Chip Interconnection Networks," in Proceeding of Design Automation Conference, 2001, DAC 2001.

[17] D. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems," Ph.D. thesis, Stanford University, Oct. 1984.

[18] The Advanced Processor Technolgies Group, University of Manchester, http://intranet.cs.man.ac.uk/apt/

[19] W. J. Bainbridge, "Asynchronous System-on-Chip Interconnect," Ph.D. thesis, Department of Computer Science, University of Manchester, UK, March 2000, ftp://ftp.cs.man.ac.uk/pub/amulet/theses/bainbridge_phd.pdf

[20] D. E. Calbaza, Y. Savaria, "A direct digital period sysnthesis circuit," IEEE Journal of Solid-State Circuits, IEEE Journal of Solid-State Circuits, Vol. 37, Issue 8, August 2002, pp. 1039-1045.

[21] R. Clauberg, P. Buchmann, A. Herkersdorf, D. J. Webb, "Design methodology for a large communication chip," in IEEE Design & Test of Computers, Vol. 17, Issue 3, July-September 2000, pp. 86-94.

[22] T. Singh, A. Taubin, "A Highly Scalable GALS Crossbar Using Token Ring Arbitration," in IEEE Design & Test of Computers, Vol. 24, Issue 5, September-October 2007, pp. 464-472.

[23] A. Chakraborty, and M. R. Greenstreet, "Efficient Self-Timed Interfaces for Crossing Clock Domains", in Ninth IEEE International Symposium on Asynchronous Circuits and Systems, ASYNC 2003, pp. 78-88.

[24] S. Balasubramanian, N. Natarajan, O. Franza, C. Gianos, "Deterministic low-latency data transfer across non-integral ratio clock domains", in 19th International Conference on VLSI Design, VLSID2006). Held jointly

with 5th International Conference on Embedded Systems and Design, January, 2006, 5 pp.

[25] M. Singh, M. Theobald, "Generalized latency-insensitive systems for single-clock and multi-clock architectures", in the Proceedings of Design, Automation and Test in Europe Conference and Exhibition, DATE 2004, February 2004, Vol. 2, pp. 1008-1013.

[26] T. Chelcea., S. M. Nowick, "Robust-Interfaces for Mixed-timing Systems ", IEEE Transaction on Very Large Scale Integration (VLSI), Vol. 12, Issue 8, August, 2004, pp. 857-873.

[27] D. J. William, and J. W. Poulton, "Digital Systems Engineering." Cambridge University Press, 1998

[28] F. Mu, C. Svensson, "Self-tested self-synchronization circuit for mesochronous clocking," IEEE Transaction on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 48, Issue 2, Feb. 2001 pp. 129-140.

[29] I. Soderquist, "Globally updated mesochronous design style," IEEE Journal of Solid State Circuits, Vol. 38, Issue 7, July 2003 pp. 1242-1249.

[30] S. R. Hasan, Y. Savaria, "Metastability Tolerant Mesochronous Synchronization," in IEEE MidWest Symposium on Circuits and Systems, Aug. 2007, MWSCAS 2007, pp 13-16.

[31] M . Kihara, "Digital clocks for synchronization and communications," Boston : Artech House, Chapter 11, c2003.

[32] D. E. Calbaza, Y. Savaria, "A direct digital period sysnthesis circuit," IEEE Journal of Solid-State Circuits, IEEE Journal of Solid-State Circuits, Vol. 37, Issue 8, Aug. 2002, pp. 1039-1045.

[33] F. R. Boyer, H. G. Epassa, Y. Savaria, "Embedded power-aware cycle by cycle variable speed processor," IEE Proceedings on Computers and Digital Techniques, Vol. 153, Issue 4, 3 July 2006, pp. 283-290.

[34] L. F. G. Sarmenta, G. A. Pratt, S. A. Ward, "Rational clocking [digital systems design]," in the Proceedings on IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1995. ICCD-1995. Oct. 1995, pp. 271-278

[35] J. Mekie, S. Chakraborty, G. Venkataramani, P. S. Thiagarajan, D. K. Sharma, "Interface design for rationally clocked GALS systems," 12th IEEE International Symposium on Asynchronous Circuits and Systems, March 2006, ASYNC 2006, 12 pp.

[36] K. Y. Yun, and R. P. Donohue, "Pausible clocking: A First step toward heterogeneous systems," in Proc. of International Conference on Computer Design, Oct. 1996, ICCD 1996, pp. 118-123.

[37] I. Sutherland, and S. Fairbanks, "GasP: a minimal FIFO control," in 7th International Symposium on Asynchronous Circuits and Systems, March 2001, ASYNC 2001, pp. 46-53.

[38] J. Mekie, S. Chakraborty, and D. K. Sharma, "Evaluation of pausible clocking for interfacing high speed IP cores in GALS framework," in the Proceedings of 17th International Conference on VLSI Design, 2004. pp. 559-564.

[39] GALS @ ETH Zurich, http://www.iis.ee.ethz.ch/async/

[40] S. Dasgupta, A. Yakovlev, "Comparative analysis of GALS clocking schemes," in IET Journal of Computers & Digital Techniques, Vol. 1, Issue 2, March 2007, pp. 59-69.

[41] P. Teehan, M. Greenstreet, G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," in IEEE Design & Test of Computers Vol. 24, Issue 5, September-October 2007, pp. 418-428.

[42] J. Kessels, A. Peeters, P. Wielage, and S. Kim, "Clock Synchronization Through Handshake Signalling," in Proceedings of Internatironal Symposium of Asynchronous Circuits and Systems, April 2002, ASYNC 2002, pp. 59-68.

[43] D. S. Bormann, and P. Y. K. Cheung, "Asynchronous Wrapper for Heterogeneous Systems," in Proceedings of ICCD-1997, pp. 307-314.

[44] A. Upadhyay, S. R. Hasan, M. Nekili, "A Novel Asynchronous Wrapper Using 1-of-4 Data Encoding And Single Track Handshaking," in 2nd IEEE North-East Workshop on Circuit and Systems , June 2004, NEWCAS 2004, pp. 205-208.

[45] Sparsö, and S. Furber, "Principles of Asynchronous Circuit Design.", Kluwer academic publishers, Boston, 2001.

[46] A. J. Martin, and M. Nystrom, "Asynchronous techniques for system-on-chip design," in Proc. of the IEEE , Vol. 94, Issue 6, June 2006, pp. 1089-1120.

[47] F. K. Gurkaynak, S. Oetiker, H. Kaeslin, N. Felber, and W. Fichtner, "GALS at ETH Zurich: Success or Failure ?," in the Proceedings of the Twelfth IEEE International Symposium on Asynchronous Circuits and Systems, Grenoble France, March 2006, ASYNC 2006, pp. 150 – 159.

[48] L. P. Carloni, and K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, "A methodology for correct-by-construction latency insensitive design," in IEEE/ACM International Conference on Computer-Aided Design, Nov. 1999, pp. 309-315.

[49] A. Iyer, and D. Marculescu, "Power and performance evaluation of globally asynchronous locally synchronous processors," in the Proceedings of 29th Annual International Symposium on Computer Architecture, May 2002, pp. 158-168.

[50] J.N. Seizovic, "Pipeline Synchronization," in the Proceedings of International Symposium of Advanced Research in Asynchronous Circuits and Systems, (ASYNC-1994), IEEE CS Press, 1994, pp. 87-96.

[51] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli, "A methodology for correct-by-construction latency insensitive design," in the IEEE/ACM International Conference on Computer-Aided Design, November 1999, pp. 309-315.

[52] R. Ginosar and R. Kol, "Adaptive Synchronizartion," IEEE International Conference on Computer Design, Oct. 1998, ICCD 1998, pp. 188-189.

[53] D. E. Calbaza, and Y. Savaria, "Direct digital frequency synthesis of low-jitter clocks," IEEE Journal of Solid-State Circuits, Vol. 36, Issue 3, March 2001, pp. 570 – 572.

[54] D. E. Calbaza, I. Cordos, N. Seth-Smith, and Y. Savaria, "An ADPLL circuit using a DDPS for genlock applications", Proceding of the 2004 International Symposium on Circuits and Systems, 2004, , Vol. 4, 23-26 May 2004, ISCAS 2004, pp. IV- 569-72.

[55] W. J. Bainbridge, and S. B. Furber, "Delay insensitive system-on-chip interconnect using 1-of-4 data encoding", in the Seventh International Symposium on Asynchronous Circuits and Systems, 2001, , March 2001, ASYNC 2001, pp. 118-126.

[56] I.E. Sutherland, "Micropipelines," Communications of the ACM, Vol. 32, No.6, June 1989, pp. 720-738.

[57] A. Kapoor and N. Jayakumar., "Novel clock distribution and dynamic de-skewing methodology," November 2004, ICCAD-2004, pp. 626-631.

[58] M. Mori, H. Chen, B. Yao, and C. Cheng, "A multiple level network approach for clock skew minimization with process variations," in Design Automation Conference, January 2004, DATE 2004, pp. 263-268.

[59] A. L. Fisher, and H. T. Kung, "Synchronizing Large VLSI Processor Arrays," IEEE Transactions on Computers, Vol. C-34, No. 8, August 1985, pp. 734-740.

[60] M. Nekili, G. Bois, and Y. Savaria, "Pipelined H-trees for high-speed clocking of large integrated systems in presence of process variations," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 5, Issue 2, June 1997 pp. 161-174.

[61] B. P. Wong, A. Mittal, Y. Cao, and G. Starr, "Nano-CMOS Circuit and Physical Design." Wiley-Interscience: A John Wiley and Sons, Inc., Publication, 2005.

[62] N. H. E. Weste, and D. Harris, "CMOS VLSI Design: A circuit and systems perspective." third edition, Pearson: Addison Wesley, 2005.

[63] Department BEW, Mixed Signal Technology Development, IBM Microelectronics Division "CMOS8RF (CMRF8SF) Design Manual", Reviewed 2006. ES # 57P9006.

[64] STMicroelectronics Design Manual for CMOS90nm technology, 2005.

[65] K. Y. Yun, and D. L. Dill, "Automatic synthesis of extended burst-mode circuits: part I (specification and hazard-free implementations)," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No. 2, Feb. 1999, pp 101-117.

[66] K. Y. Yun and D. L. Dill, "Automatic synthesis of extended burst-mode circuits: part II (automatic synthesis)," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No. 2, Feb. 1999, pp. 118-132.

[67] W. J. Bainbridge, W.B. Toms, D. A. Edwards, S. B. Furber, "Delay-insensitive, point-to-point interconnect using m-of-n codes," Proceeding of Ninth International Symposium of Asynchronous Circuits and Systems, May 2003, ASYNC 2003, pp. 132-140.

[68] E. G. Jung, B. S. Choi, Y. G. Won, and D. I. Lee, "Handshake protocol using return-to-zero data encoding for high performance asynchronous bus," in IEE Proceedings of Computers and Digital Techniques, Vol. 150, Issue 4, July 2003 pp. 245-251.

[69] E. G. Jung, J. G. Lee, K. S. Jhang, and D. S. Har, "Differential value encoding for delay insensitive handshake protocol," IEICE Transactions on Information and Systems, Vol. E88, No. 7, July, 2005, pp. 1437-1444.

[70] M. R. Becer (Adv. Tools Group, Austin, TX, USA), D. Blaauw, V. Zolotov, R. Panda, and I. N. Hajj, "Analysis of noise avoidance techniques in DSM interconnects using a complete crosstalk noise model," Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 2002, DATE 2002, pp. 456-63.

[71] K. S. Chung, T. Kim, and C. L. Liu, "G-vector: a new model for glitch analysis in logic circuits," Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, Vol. 27, No. 3, March, 2001, pp. 235-251.

[72] L. Hyungwoo, S. Hakgun, and K. Juho, "Glitch elimination by gate freezing, gate sizing and buffer insertion for low power optimization circuit," Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE Vol. 3, November, 2004, pp. 2126 – 2131.

[73] M. Abramovici, M. A. Breuer, A. D. Friedman, "Digital Systems Teststing and Testable Designs." IEEE Press, 1990.

[74] Khosrow Golshan (Conexant Systems, Inc.), "PHYSICAL DESIGN ESSENTIALS An ASIC Design Implementation Perspective", Springer, 2006.

[75] R. Mullins, S. Moore, "Demystifying Data-Driven and Pausible Clocking Schemes," in IEEE International Symposium on Asynchronous Circuits and Systems , March 2007, ASYNC 2007, pp. 175-185.

[76] A. Edman, C. Svensson, "Timing closure through a globally synchronous, timing partitioned design methodology," in Proceedings of the 41st Design Automation Conference, 2004, DAC 2004, pp. 71-74.

[77] P. Caputa, C. Svensson, "An on-chip delay- and skew-insensitive multicycle communication scheme," in IEEE International Conference of Technical Papers on Solid-State Circuits, February, 2006, ISSC 2006, pp. 1765-1774.

[78] A. Edman, C. Svensson, "Synchronous latency-insensitive design for multiple clock domain," in Proceedings of IEEE international SOC Conference, September, 2005, pp. 83-86.

[79] W. J. Dally and J. W. Poulton, "Digital Systems Engineering." Cambridge University Press, 1998.

[80] H. T. P. Nguyen, "Conception d'un module de synchronisation pour l'intégration à l'échelle de la tranche de routeurs de communication." Masters' Thesis, Départment de génie électique, École Polytechnique de Montréal, Montreal, QC, Canada, December 2005.

[81] S. R. Hasan, Y. Savaria and Mohamed Nekili, "Split H-tree Design Method for High-Performance GALS Systems," in 4th IEEE NorthEast Workshop on Circuit and Systems, June 2006, NEWCAS 2006, pp. 161-164, 2006.

[82] T. Chelcea, S.M. Nowick, "Robust interfaces for mixed-timing systems with application to latency-insensitive protocols," in IEEE Proceedings of Design Automation Conference, 2001, DAC2001, pp. 21-26, 2001.

[83] M. R. Greenstreet, "Implementing a STARI chip," in IEEE International Conference on Computer Design: VLSI in Computers and Processors, Oct. 1995, ICCD 95, pp. 38 – 43.

[84] S. R. Hasan, N. Bélanger, Y. Savaria, "All Digital Skew Tolerant Synchronous Interfacing Methods for High-Performance Point-to-Point Communication in

DSM SoCs." Technical Report, EPM-RT-2008-10, Département de génie électrique, École Polytechnique de Montréal, Montréal, QC, Canada, December 2008.

[85] S. Im and K. Banerjee, "Full chip thermal analysis of planar (2-D) and vertically integrated (3-D) high-performance ICs," in Proc. Int. Electron Device Meeting, 2000, pp. 727-730.

[86] A. M. Ajami, K. Banerjee, and M. Pedram, "Modeling and Analysis of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects," in IEEE transcations on CAD of Integrated Circuits and Systems, Vol. 24, No.6, June 2005, pp. 849-861.

[87] J. C. Ku and Yehea Ismail, "On the Scaling of Temperature-Dependence," in IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 26, No. 10, Oct. 2007, pp. 1882-1888.

[88] U. Y. Ogras, R. Marculescu, P. Choudhary, D. Marculescu, "Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip," in Design Automation Conference, DAC-2007, June 2007, pp. 110-115.

[89] Pinhong Chen; D.A. Kirkpatrick, K. Keutzer, "Miller factor for gate-level coupling delay calculation," IEEE/ACM conference on Computer Aided Design, Nov. 2000, ICCAD 2000, pp. 68-74.

[90] A.B. Kahng, S. Muddu, E. Sarto, "On switch factor based analysis of coupled RC interconnects," Proceedings of 37th Design Automation Conference, June 2000, DAC 2000, pp. 79-84.

[91] A. Katoch, (Philips Res. Labs., Eindhoven, Netherlands), M. Meijer, S.K. Jain, "Active noise cancellation using aggressor-aware clamping circuit for robust on-chip communication," Proceedings of 18th International Conference on VLSI Design, 2005, pp. 325-329.

[92] Jiun-Sheng Huang, Shang-Wei Tu, Jing-Yang Jou, "On-chip bus encoding for LC cross-talk reduction," in International Symposium on VLSI Design, Automation and Test, April 2005, VLSI-TSA-DAT 2005, pp. 233 – 236.

[93] S. R. Hasan, N. Bélanger, Y. Savaria, "All-digital skew-tolerant interfacing method for systems with rational frequency ratios among Multiple Clock Domains: Leveraging a priori timing information," 1st Microsystems and Nanoelectronics Research Conference, Oct. 2008, MNRC-2008, pp. 129 – 132.

[94] M. Bubois, Y. Savaria, D. Haccoun, N. Belanger, "Low-power configurable and generic shift register hardware realisations for convolutional encoders and decoders," in IEE Proceedings on Circuits Devices and Systems, Vol. 153, No. 3, June 2006, pp. 207-213.

[95] J.P. Roth, "Diagonosis of Automata Failures: A Calculus and a Method," IBM Journal of Research and Development, Vol. 10, No. 4, July 1966, pp. 278-291.

[96] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in Proceedings of Seventh International Symposium on High Performance Computer Architecture, 2001, pp. 171-182.

[97] A. Upadhayay, S. R. Hasan, M. Nekili, "A Novel Asynchronous Wrapper using 1-0f-4 data encoding and single-track handshaking," in 2nd North East Workshop on Circuits and Systems, June 2004, NEWCAS 2004, pp. 205-208.

[98] S. R. Hasan, B. Pontikakis, Y. Savaria, "An All-Digital Skew-Adaptive Clock-scheduling Algorithm for Multiprocessor Systems on Chips (MPSoCs)," in IEEE International Symposium on Circuits and Systems, May 2009, ISCAS 2009, Taipei, Taiwan.

[99] Synopsys, Design Compiler User guide, Version V-2004.06, June 2004.

[100] Synopsys, PrimeTime Tutorial, Version V-2004.06, June 2004.

[101] Xilinx's Virtex-II Pro, Platform FPGA Handbook, UG012 (v2.0), October 14, 2002.

[102] S. R. Hasan, N. Bélanger, Y. Savaria, M. O. Ahmad "All-digital skew-tolerant interfacing method for systems with rational frequency ratios among Multiple Clock Domains: Leveraging a priori timing information," Submitted in Elsevier Journal of VLSI Integration.

[103] S. R. Hasan, N. Bélanger, Y. Savaria, M. O. Ahmad, "Crosstalk glitch propagation modeling for Asynchronous interfaces in Globally Asynchronous Locally Synchronous Systems," accepted for publication in IEEE Transaction of Circuits and Systems Part I (TCAS-I).

[104] S. R. Hasan, N. Bélanger, Y. Savaria, M. O. Ahmad, "Crosstalk glitch gating: A solution for designing glitch tolerant asynchronous handshake scheme for GALS systems", under revision in IEEE Transaction of Circuits and Systems Part I (TCAS-I).

[105] R. Ho, K. W. Mai, M. A. Horowitz, "The future of wires," in the Proceeding of the IEEE, Vol. 89, No. 4, April 2001, pp. 490-504.

[106] G. Birtwistle, K. S. Stevens, "The Family of 4-phase Latch Protocols," in 14th IEEE International Symposium on Asynchronous Circuits and Systems, 2008. ASYNC 2008. April 2008, pp. 71-82.

[107] E. Amini, M. Najibi, Z. Jeddi, H. Pedram, "FPGA Implementation of Gated Clock based Globally Asynchronous Locally Synchronous Wrapper Circuits," in International Symposium on Signals, Circuits and Systems July 2007 ISSCS 2007. pp. 1-4.

[108] W. J. Bainbridge, S. J. Salisbury, "Glitch Sensitivity and Defense of Quasi Delay-Insensitive Network-on-Chip Links" in 15th IEEE Symposium on Asynchronous Circuits and Systems, May 2009. ASYNC 2009, pp. 35-44.

[109] P. Mahoney, E. Fetzer, B. Doyle, S. Naffziger, " Clock Distribution on a Dual-Core Multi-Threaded Itanium®-Family Processor," in IEEE International Solid-State Circuits Conference, 2005. Digest of Technical Papers, Feburary 2005, Vol. 1, ISSCC 2005, pp. 292 – 599.

[110] K. J. Kuhn, " Moore's Law Past 32nm: Future Challenges in Device Scaling," in 13th Internaational Workshop on Computational Electronics, May, 2009, IWCE'09, pp.1 – 6.

[111] J. R. Powell, " The Quantum Limit to Moore's Law," in IEEE Proceedings, Vol. 96, No. 8, August, 2008, pp. 1247 – 1248.

Appendix: Metastability Tolerant Mesochronous Synchronization

This appendix presents a methodology to obtain mesochronous synchronization and, in the process, establishes a methodology to estimate phase variations. This is especially helpful for SoCs where the clocks of the communicating modules change their phases in real time, as the case elaborated in Section 8.2 of the thesis. This work also presents a new synchronization scheme for mesochronous communication. This design has better metastability tolerance compared to state-of-the-art synchronizers. It has low latency and is only composed of standard digital components. This solution avoids the prevalent assumption, in many contemporary synchronizing techniques, of solving the metastability in half a clock cycle. The new design achieves latency as low as one clock cycle for a 500 MHz system clock, under the 180nm TSMC technology. A proof of concept simulation is also performed to validate the proposed design methodology.

A.1 Limitations of Available Mesochronous Synchronizers

Clock distribution is becoming increasingly difficult in DSM technologies. Various timing constraints, including skew budget due to process variations, put a limit on the

fastest speed a chip can work at [81]. A split H-tree design methodology that alleviates this problem is discussed in Chapter 3 of this thesis, which shows that each of the split halves of the H-tree can individually work at a faster frequency. On the other hand, these halves may constitute a mesochronous system. In such a system, the clock frequency is the same, but the phases of the domains may differ by a random quantity subject to parametric variations. Thus, interfaces are required to communicate between mesochronous regions.

Several mesochronous interfacing designs are available in the literature, such as the delay-line synchronizer [52], and the clock edge synchronizer [28]. The reported solution with the highest performance is a self-testable self synchronization (STSS) design [28]. In some of these circuits, it is assumed that metastability is resolved in half a clock cycle. This constraint can restrict the maximum speed of an interface. This sets a need for architectures that provide more freedom in the timing budget of mesochronous systems.

The proposed design utilizes *a priori* timing information of the synchronous system. *A priori* knowledge of frequencies helps developing the sampling scheme, such that at most, one sample may become metastable. These samples are analyzed using an algorithm to detect the phase difference between the remote and local clocks. Selection of the proper phase of the local clock, which is most in-synch with the remote clock, is performed by the Decision Maker (DM), the hardware implementation of this algorithm. This synchronizer is easy to implement, yet requires lower latency than existing solutions, while it alleviates the design from the severe constraint of resolving the metastability in half a clock cycle.

The rest of this appendix is organized as follows. Section A.2 explains the architecture of the low latency synchronizer. Section A.3 presents two different sampling techniques

used with the method. Section A.4 describes a proposed decision making algorithm and its hardware implementation. Section A.5 presents simulation results of two key modules: The Sampler and the DM. Finally, Section 6 provides a summary of this appendix.

A.2 Low Latency Synchronizers

In this work, a new mesochronous synchronizing scheme is proposed. Figure A.1 shows its basic block diagram, and explains the general principle of the design. The system has two modes, synchronization mode and normal mode, under the control of a state machine. A sender module sends the clock and data signals to a receiver module. Once the system enters the synchronization mode, a control signal from the state machine sends the signal to the synchronizer to synchronize the remote clock with the reference local clock. In the mean time, the sender data is stored in the buffer module. The synchronization mode starts periodically at a predefined time interval. This interval is chosen by the designer based on the *a priori* timing information about the statistical process variations, environmental variations and noise, jitter, setup and hold time etc. In the normal mode, data is received by the receiver module using the in-synch local clock.

Figure A.2, shows the various components of the synchronizer block. A remote clock signal is sampled, at least five times (five is a possible solution chosen as a starting point) by the bank of DFFs under the control of a set of equally delayed local clocks. The sampling time interval is calculated based on the knowledge of the timing parameters. These parameters include a forbidden zone (t_{fz}), the time interval during which, if a DFF is clocked, it cannot guarantee a metastable free output. Other parameters are the time

period, the frequency of the clock, and the number of samples. Two different sampling techniques are discussed in Section A.3.

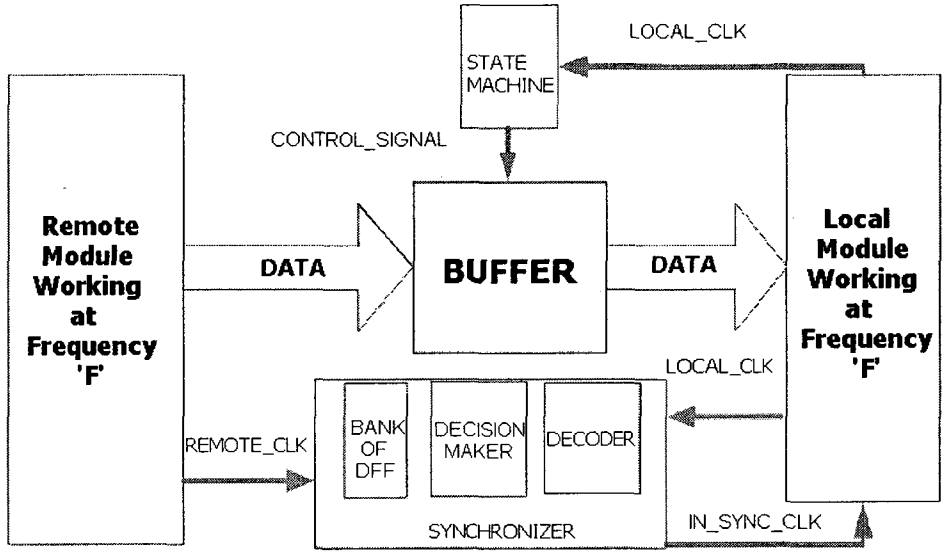


Figure A.1. Low latency Synchronizer System

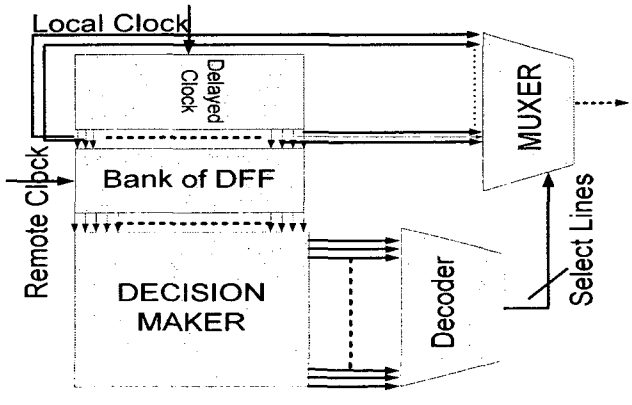


Figure A.2. Synchronizer Block Diagram

These sampled signals are provided to a decision making combinational block that determines the most in-synch delayed version of the local clock with the remote clock. The DM module feeds the signal to the $\lceil 2^{(\log N/\log 2)} \rceil$ to $\lceil \log N / \log 2 \rceil$ line decoder. The output of the decoder serves as a select signal for the multiplexer. This multiplexer,

through the select signal, passes the delayed clock that samples the remote clock signal with the best phase margin.

A.3 Sampling Technique

In this section, two sampling techniques are discussed. These techniques can be implemented in hardware using the two blocks labelled Delayed Clock and Bank of DFF, as shown in Figure A.2. The key requirement in such designs is to avoid having more than one metastable output value in the sampled vector. It can be seen in the example of Figure A.3 that, if the separation between consecutive samples of the remote clock signal is more than the forbidden time duration, then there will never be more than one metastable output for any given remote clock cycle period. Recall that the sampling time interval corresponds to the time interval between two consecutive rising clock edges, of the locally delayed clock. The resulting sampling intervals are shown in Figure A.3, where for illustration, Delta1 leads to a metastable output, while Delta2, Delta3, Delta4, and Delta5 will give valid outputs.

Such sampling technique requires further careful design optimization in order to make sure that no two edges of the remote clock fall on the t_{fz} in the same sampling cycle. In Figure A.3, in order to prevent Delta4 from falling on the t_{fz}, two different techniques can be adopted. One possibility is to make sure that Delta4 and Delta3 are late and early enough, respectively, to avoid the falling edge t_{fz}. This can be done by proper delay adjustment in the transient generator, as discussed in the 1st technique for delayed clock generator. An alternative method is to avoid this timing condition by making sure that the

next edge of the remote clock falls in the safe region. This requires an edge detection mechanism and proper duty cycle adjustment, this is discussed further in the 2nd technique of the next section.

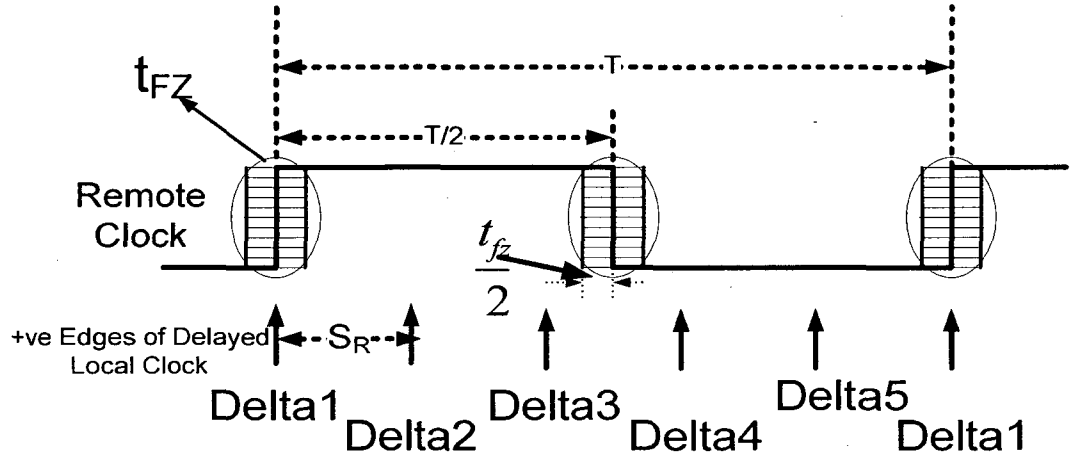


Figure A.3 Sampling Technique

Delayed Clock Generator

This block of the design generates, as the name suggests, delayed local clocks. These clock signals sample the incoming remote clock signal. The sampling period is decided upon the *a priori* knowledge of the time period, and the t_{fz} of the DFF. Two techniques can be used for this sampling as described next.

First Technique: Figure A.3 shows the sampling technique used in this work. The sampling time interval, which is denoted by S_R in Figure A.3, is calculated beforehand based on a worst-case timing analysis. This timing analysis requires previous knowledge of t_{fz}. The bounds on S_R are calculated as follows. For the case where the number of samples is odd, the $((N-1)/2)^{th}$ sample should respect the following worst-case inequality:

$$-\frac{t_{fz}}{2}+\frac{T}{2}-\frac{t_{fz}}{2}>\frac{N-1}{2}*S_R \rightarrow S_R<\frac{T-2t_{fz}}{N-1} \quad (1)$$

where T is the clock cycle of the local (or remote) clock, and N is the number of samples per clock cycle. Similarly, the $((N+1)/2)^{th}$ sample should respect the following condition

$$\frac{t_{fz}}{2}+\frac{T}{2}+\frac{t_{fz}}{2}<\frac{N+1}{2}*S_R \rightarrow S_R>\left(\frac{T+2t_{fz}}{N+1}\right) \quad (2)$$

Rearranging inequalities (1) and (2) leads to the following

$$\frac{T+2t_{fz}}{N+1}<S_R<\frac{T-2t_{fz}}{N-1} \quad (3)$$

The above analysis is valid only if $t_{fz} < S_R$. Such a sampling technique guarantees that, in this mesochronous design, there will be at most one metastable output.

Second Technique: Another design technique to generate delayed clock is based on an edge detection mechanism. This technique generates a pulse, using a rising edge detector, which controls the pulse width. This pulse width, according to the DM requirement, should possess a duty cycle such that a minimum of two locally-delayed clocked DFFs sample each of the two logic levels, 1 and 0. Furthermore, the duty cycle should guarantee that the two consecutive edges of the remote clock will not both fall in the t_{fz} of the local delayed clocked DFFs. Inequalities (4), (5), and (6) express the constraints on the minimum and maximum edge-detect pulse-width duration or ED_{PW}.

$$ED_{PW} > 2\,S_R + t_{fz} \quad (4)$$

$$ED_{PW} < 3\,S_R - t_{fz} \quad (5)$$

Combining inequalities (4) and (5) leads to the following inequalities,

$$2S_R + t_{fz} < ED_{PW} < 3S_R - t_{fz} \quad (6)$$

A similar analysis is required when there is an even number of sampling registers. The optimum choice of even or odd number of samples, along with the total number of

samples, requires further detailed analysis and is not dealt in this work. The rest of this work has been designed based on the assumption that the number of samples is odd.

Bank of DFFs

Figure A.4 shows a bank of DFFs. The input signal to all of the flip-flops is the same, i.e. the remote clock. The transition generator generates delayed clocks based on inequalities (1) to (3). The output of DFFs labelled A1 to AN can have at most one metastable output, and it is the responsibility of the DM to filter metastability in the system.

A.4 Decision Maker (DM)

The decision maker (DM) is the most critical design element of this solution. The DM, through its combinational logic, selects a delayed clock, which provides metastability free output. DM implements a combinational logic that selects a single phase from the set of delayed clocks (delta1 to delta5 in Figure A.3). Although, through the sampling technique specified in Section A.3, it is guaranteed that metastability can occur at most once in the entire duration of each sampling cycle, metastability may be due to a rising or a falling edge of the clock. The DM is designed to check both the rising and falling edge concurrently. It is mandatory for a robust implementation that the selection criterion leading to the use of each delayed version of the local clock, based on a set of samples, be unique. The following analysis describes the functionality of DM.

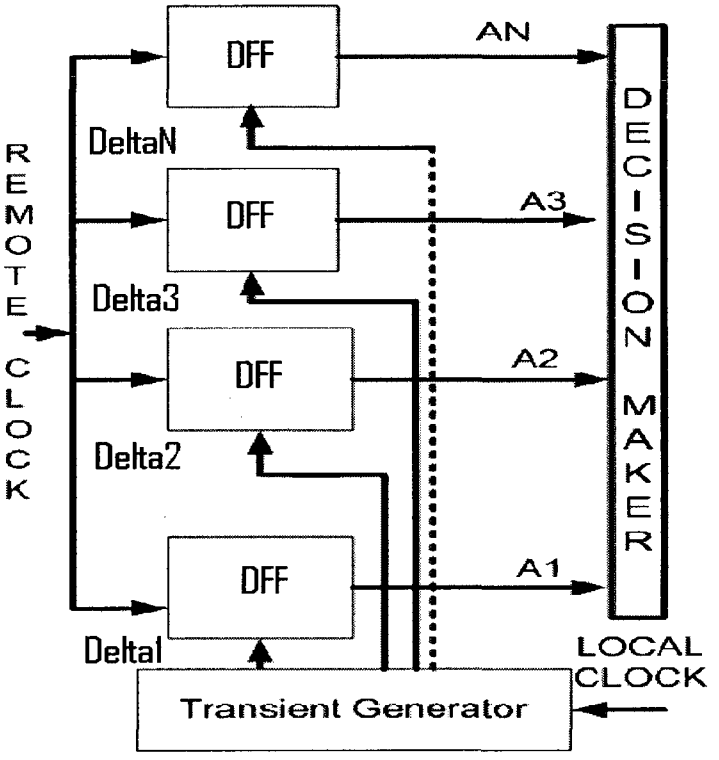


Figure A.4. Bank of DFFs

The following discussion assumes that the number of samples per period is odd and is equal to 5. Let A_1 to A_5 be the input to the DM as shown in Figure A.4. It is assumed that there is no more than one metastable value, and as was explained earlier, there are two possibilities for the metastable input, when one is present. Either metastability occurs due to rising edge (0 → 1) coinciding with the t_{fz}, or the falling edge (1→0) occurring at t_{fz}. Tables A.1 and A.2 show the truth tables converting sampled vectors to decision, for the rising and falling edge detection cases respectively, for this specific context of five samples. It can be noticed that Tables A.1 and A.2 encompass all the possible sets of events in this context: five samples and a maximum of one metastable sample. For illustration purposes, a specific case from the Table A.1 is explained. As shown in

Table A.1, case 1 contains the logic value X1100. This leads to the Boolean form, $A_2A_3\bar{A}_4\bar{A}_5$. If this function is true, then it selects A_2 as the sampling pulse. Note that, in Table A.1, each case has a unique Boolean function, and all are mutually exclusive.

Table A.1. Decision Maker's Truth Table (Rising Edge)						
Case	A1	A2	A3	A4	A5	Decision
1	X	1	1	0	0	A2
2	0	X	1	1	0	A3
3	0	0	X	1	1	A4
4	1	0	0	X	1	A5
5	1	1	0	0	X	A1

Table A.2. Decision Maker's Truth Table (Falling Edge)							
Case	A1	A2	A3	A4	A5	Decision Rule X=1	Decision Rule X=0
6	X	0	0	1	1	A5	A4
7	1	X	0	0	1	A1	A5
8	1	1	X	0	0	A2	A1
9	0	1	1	X	0	A3	A2
10	0	0	1	1	X	A4	A3

In order for the Boolean functions, derived from Table A.1, to cover all the possible events, they should provide the correct result for the cases of Table A.2. To illustrate the effectiveness of the Boolean function acquired through Table A.1, an example is demonstrated. It is again assumed that case 1 is true. If this case is true due to the rising edge of the remote clock, it selects A_2 as the most in-synch sampling edge. At the same time, it is also possible that Boolean function $A_2A_3\bar{A}_4\bar{A}_5$ (X1100) is activated due to the aliasing in Table A.2. Cases 8 and 9 are the aliases for X1100, when X=1 and X=0, respectively. The robustness of the design is still protected as it can be seen that, in both cases, A_2 is still a valid output. Thus, the Boolean function $A_2A_3\bar{A}_4\bar{A}_5$ covers these aliases as well. Due to the cyclic property of the clock samples, such an analysis will lead to a unique solution for all the five sampling signals, which are input to the DM. Note that one

may be tempted to conclude that since X1100, which is case 1, and 11X00, which is case 8, are both leading to select A_2, then it is possible to implement X1X00 as a Boolean function for A2, and similarly, for the rest of the sampling signals. However, a closer inspection shows that such Boolean functions lead to some irresolvable aliasing. For example, it can be shown that X1X00 would lead to irresolvable aliasing with 0X1X0 and 1X00X.

A.5 Simulation Results

The validation of the design is accomplished by performing HSpice simulations of the bank of DFFs and DM, two of the five blocks shown in Figure A.2. The pre requisite of this simulation is to generate delayed local clocks following the condition mentioned in inequality (3). The parameters of this inequality consist of T, t_{fz}, and N. Parameter t_{fz} is obtained through parametric analysis, using HSpice for a TSPC based DFF [6], under TSMC's 180nm technology. The value of t_{fz} was measured to be approximately 90 psec for a load of 250 fF. To consider the effects of process variations and other noise sources, t_{fz} is assumed to be 180 psec. Inserting the value of t_{fz} in inequality(3), and solving through Matlab, it is found that, for a T of 2 ns and N=5, S_R may vary between 393 ps and 410 ps. In the following simulations, S_R is kept at 400 ps.

Proof-of-concept simulation results are shown in Figures A.5 to A.8. Sampling signals A_1 to A_5 correspond to the five sampling signals out of the bank of DFFs in Figure A.5, under the condition that falling edge is falling on the t_{fz}. It is shown that output A_3 is in a metastable condition, which corresponds to logic value set 11X00 or case 8 in Table A.2.

The corresponding select signal from the DM is shown in Figure A.7. It can be seen in this figure that initially, when A_3=0, the DM has selected A_1, since the condition related to case 5 is valid. Later, when the remote clock is delayed further, A_3 =1, and, in such cases, the first row in Table A.1 becomes valid and, as a result, A_2 is selected. Similarly, Figure A.6 demonstrates the output of the bank of DFFs under the condition that rising edge happens on t_{fz} and A_1 is experiencing metastability. Figure A.8 exhibits the simulation results of DM, showing that A_2 is selected. This selection signal corresponds to case 1 in Table A.1, i.e., the logic value set is X1100, as can be seen in Figure A.6.

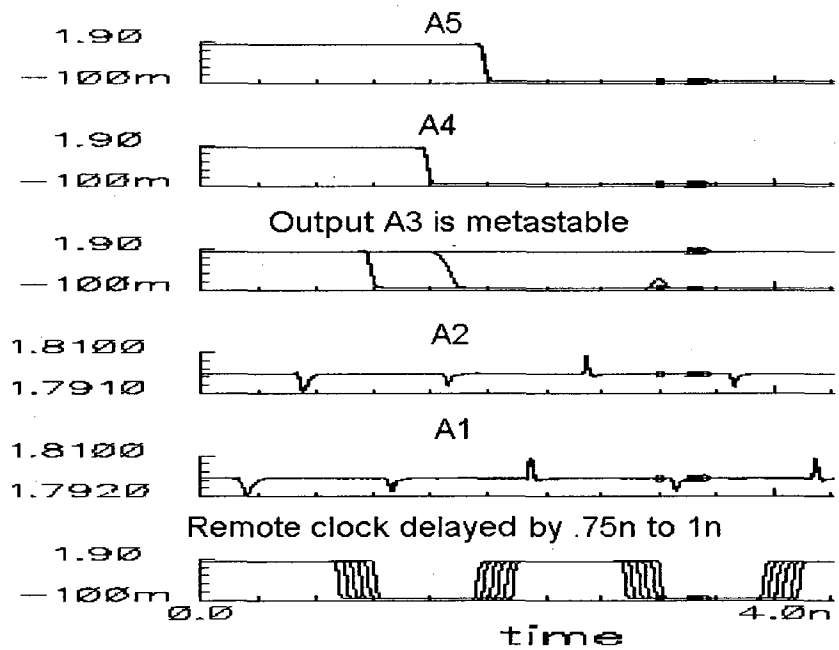


Figure A.5. Outputs from the Bank of DFFs (Falling edge)

The proposed design does not assume that metastability can be resolved in half a clock cycle. Figures A.5 and A.7 demonstrate metastability in A_3 and A_1, respectively. It is also shown in these figures that the metastability took more than half a clock cycle to get resolved, for a 500 MHz clock frequency. The latency of the proposed design is dependent

on the combinational logic, and therefore, with proper optimization of the circuits, it can result in less than one clock cycle latency. It can be seen from Figure A.7 and A.8 that even for the worst-case, the latency of the DM is less than half a clock cycle of a 500 MHz system clock. Considering typical technology delays of decoders and multiplexers, this design can work at an overall latency smaller than one clock cycle at 500 MHz with TSMC's 180nm CMOS technology.

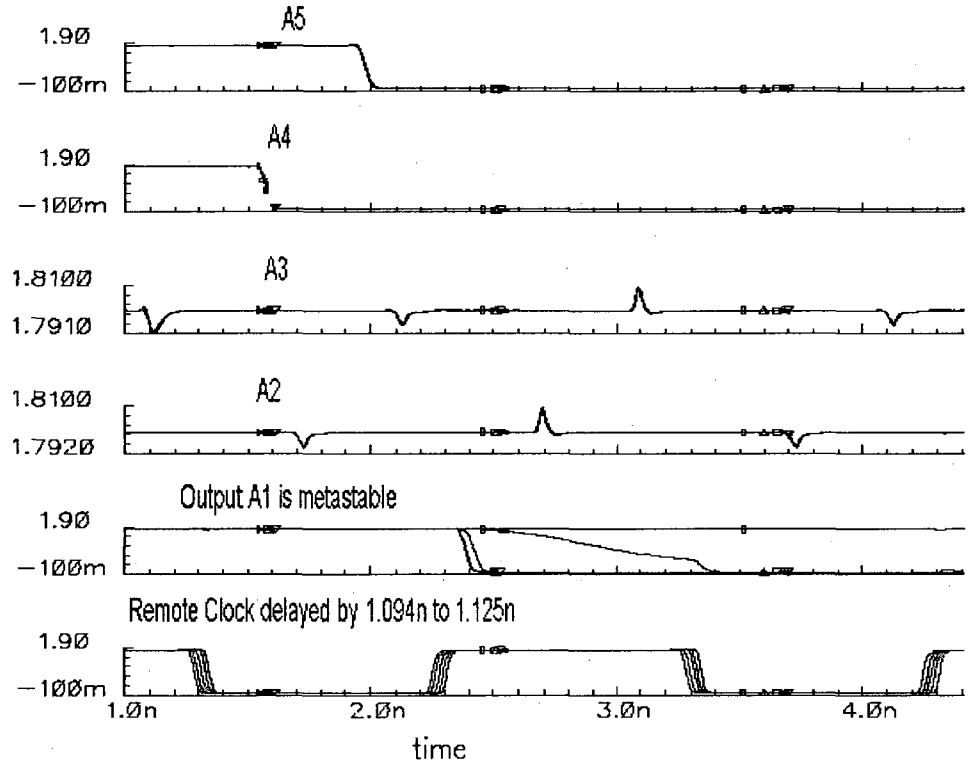


Figure A.6. Outputs from the Bank of DFFs (Rising edge)

A.6 Summary

This work proposes a new synchronizer scheme for mesochronous communication in integrated systems. Overall, the proposed mesochronous synchronizer is more tolerant to metastability than state-of-the-art designs. It has a comparable hardware cost and typically produces close to one clock cycle latency for 500 MHz. system clock, when implemented with the 180nm TSMC CMOS technology. This solution avoids the prevalent assumption, made in many contemporary synchronizing techniques, of solving the metastability in half a clock cycle. Two different sampling techniques are proposed, and a new DM algorithm is suggested. A proof of concept simulation is performed that shows the functionality of this proposed synchronizer, and its tolerance to metastability

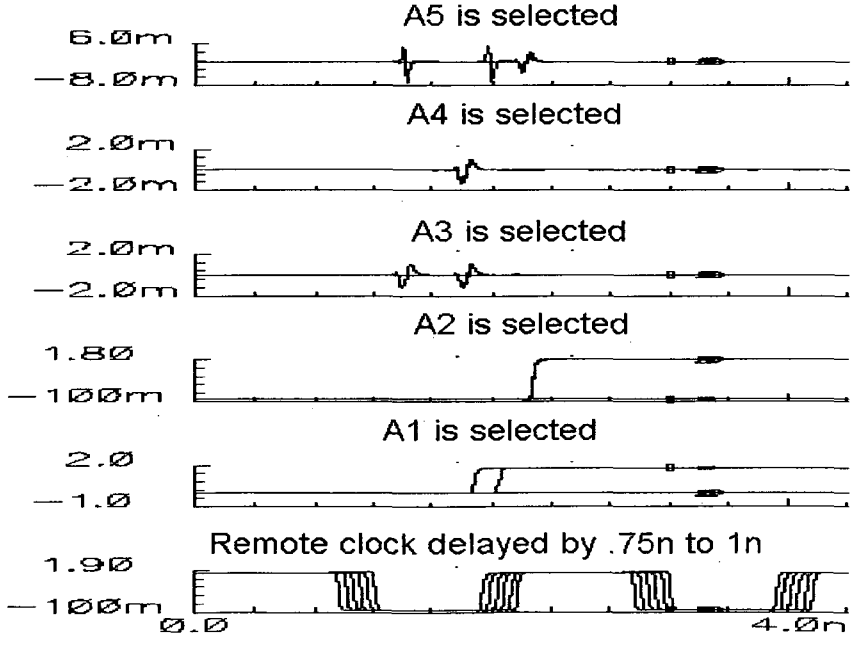


Figure A.7. Selecting singnal from DM (Falling edge on t_{fz})

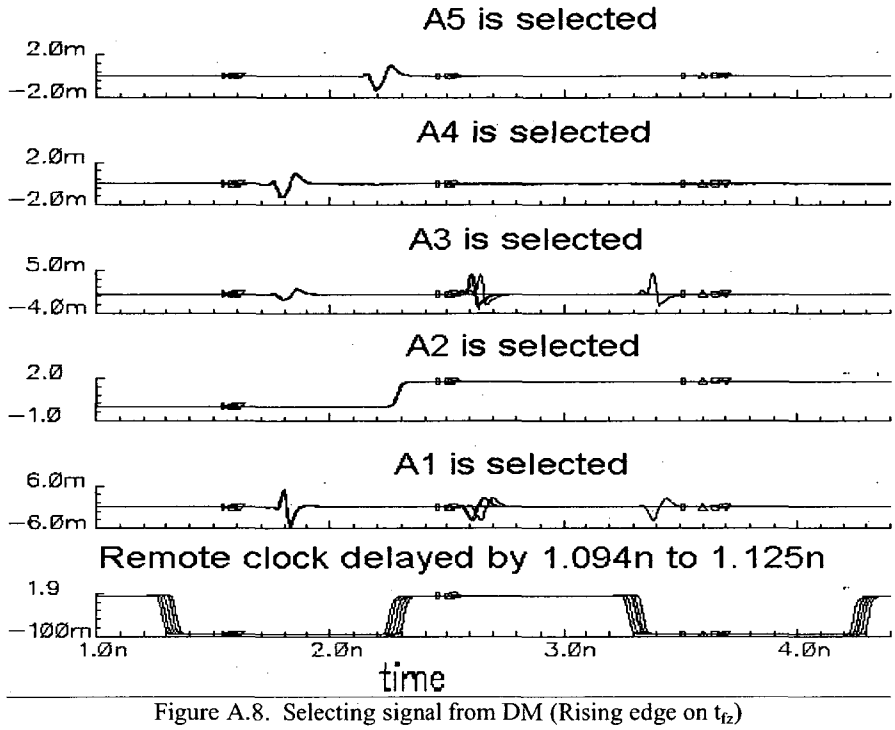


Figure A.8. Selecting signal from DM (Rising edge on t_{fz})