Test Strategies for High-Speed Synchronization Interfaces

Octavian Petre

This research has been supported by the Technology Foundation STW, applied science division of NWO and the technology program of the Ministry of Economic Affairs in the Netherlands.

Composition of the Graduation Committee:
Friday 27^{th} of February 2004 at 16.45

Chairman:	prof.dr. W.H.M. Zijm	
Secretary:	prof.dr. W.H.M. Zijm	Univ. Twente, EWI
Promoter:	prof.dr.ir. T. Krol	Univ. Twente, EWI
Assistant Promoter:	dr.ir. H. G. Kerkhoff	Univ. Twente, EWI
Internal Members:	prof.dr. H. Wallinga	Univ. Twente, EWI
	prof.ir. A.J.M. van Tuijl	Univ. Twente, EWI
External Member:	prof.ir. M.T.M. Segers	Philips Semiconductors
Referee:	dr. R.J.W.T. Tangelder	Windesheim

Title: Test Strategies for High-Speed Synchronization Interfaces
Author: Octavian Petre
ISBN: 90-365-2022-3

Printed by Febodruk B.V., Enschede, The Netherlands, 2004
© Octavian Petre, Enschede, The Netherlands, 2004
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation.

Test Strategies for High-Speed Synchronization Interfaces

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof.dr. F.A. van Vught,
on account of the decision of the graduation committee,
to be publicly defended
on Friday 27^{th} of February 2004 at 16.45

by

Octavian Petre
born on March 6^{th}, 1975
in Bucharest, Romania

This dissertation is approved by promoter:
prof.dr.ir. T. Krol

and assistant promoter:
dr.ir. H. G. Kerkhoff

to my lovely wife Rita Christa and our angel Lucas David

Acknowledgments

While this thesis has only one author, the presented work would not have been possible without the help of many, many ... people. Therefore, I would like to thank ...

- my early daily supervisor, Ronald Tangelder, for writing the ASYNC project proposal; for his patience while dealing with my first published paper and for mediating my internship in Alcatel, Belgium.
- my assistant promoter, Hans Kerkhoff, for reading thoroughly all of my published papers and always giving painful, but very useful, comments; for his idea to organize every year a "TDT day-off" and every month a "TDT-lunch" which boosted the team spirit of our group.
- Hans Wallinga and Thijs Krol for being my promoters at different moments in time.
- my TDT-mates: Arun, Frank, Herman, Liquan, Marcel, Milan, Rob and Vladimir for spending only "quality time" together.
- Rob Tijink for his **invaluable** knowledge about "all" CAD tools on the market and for his professional approach to every problem.
- Benoit Jarry for his hard work implementing the delay-line demonstrator chip during his 5 months stay in The Netherlands.
- our skillful system administrators, Cor, Egbert and Frederik, for providing useful and professional IT support and for installing a *Linux* box which improved tremendously my work efficiency.
- Violeta and Vladimir for their "useful" coaching during my first weeks in The Netherlands.
- our secretaries Annemiek, Marie-Christine and Miranda for their prompt replies to my queries.
- our financial adviser, Joke Vollenbroek, for her promptitude when solving our group finances.
- Alexey, André, Cora, Daniel, Gigi, Henk, Jay, Lorenzo, Remco, Sander, Sheela, Simon, Svetlana, Vincent, Zhichun for the nice discussions we had during, but not only at, the coffee+ reunions.

- Javier and Johan for the good time spent together during the supervision of the ASIC students.
- Petrică & Corina, Bogdan & Graţiela, Szabolcs & Loredana, Adrian & Raluca, Orest & Maria, Victor & Andreea, Irinel & Irina, and Ciprian for all nice moments spent together and for their *healthy* humor.
- my two supervisors in Alcatel Antwerpen, Erik and Johan, for creating a stimulating working environment.
- all my friends in Alcatel: Alberto, Patrick, Mathieu, Silvio, with whom I spent many enjoyable periods, starting from football matches and ending with pub surveys.
- the 'great' Cezar, for sharing with my family some of his scarce spare time, and for accepting the role as a 'paranimf'.
- Marius for being the other 'paranimf' & Cami for being our long-term friend in good or bad times.
- my family back home, for providing the necessary support for my Dutch endeavor.
- my uncle Petrică, who played a vital role in my early education. I am **highly grateful** for his effort in training my mathematical skills.
- all my friend in The Netherlands and all over the world who did not have a direct influence in writing this thesis but helped me become a ... real me ;-)

And last, but definitely not least, I am more than grateful to my wife, Rita Christa, for being such an excellent spouse and for giving birth to our son Lucas David. There are no words available to describe my appreciation. Without you life would have been miserable ;-).

At the end of my assignment I just want to...

Thank you all !

Contents

Glossary of special definitions, symbols and notation

Abbreviation	Description
ADC	Analogue to Digital Converter
ASfc	Asynchronous to Synchronous with free running clock
ASsc	Asynchronous to Synchronous with stoppable clock
ATPG	Automatic Test Pattern Generation
BFS	Brute-Force Synchronizer
BER	Bit Error Rate
BIST	Built-In Self-Test
CAD	Computer Aided Design
CAT	Computer Aided Test
CLC	Combinational Logic Circuit
CMOS	Complementary Metal-Oxide Semiconductor
DAC	Digital to Analogue Converter
DfT	Design for Testability
DLS	Delay-Line Synchronizer
FF	Flip-Flop
FIFOS	First-In First-Out (FIFO) Synchronizer
FSM	Finite State Machine
IC	Integrated Circuit
IP	Intellectual Property
MCD	Multiple Clock Domains
MUTEX	MUTual EXclusion (element)
NRZ	Non-Return-to-Zero signaling

Abbreviation	Description
ΦC	phase Comparator
PCB	Printed Circuit Board
PS	Periodic Synchronizer
s-a-0	stuck-at zero
s-a-1	stuck-at one
SaB	(combined) Scan and BIST (test strategy)
SCD	Single Clock Domain
SoC	System-on-Chip
TRS	Two-Register Synchronizer
ULSI	Ultra Large Scale of Integration

Definitions	Description
aperture time of a flip-flop	the sum of setup and hold times of a flip-flop
average transmission latency	average delay from the time a new data is sent until the data is received
hold time	is the time following a clock event during which the data input to a latch or flip-flop must remain stable in order to guarantee that the latched data is correct.
minimum feature size	the dimension of the smallest feature actually constructed in the manufacturing process
scan-mode	special test mode for a digital system in which all flip-flops are connected in a shift-register like structure
setup time	is the time relative to a clock event during which the data input to a latch or flip-flop must remain stable in order to guarantee that the latched data is correct.

List of Tables

List of Figures

Chapter 1

Introduction

Every man-made item around us has been tested at some point in time. Test is an important step for all manufactured products. The complexity of products is increasing and the test strategies should definitely be prepared for this. The inability to detect crucial manufacturing faults can be the source of "expensive" failures: *"The Mars Polar Lander [10] spacecraft's engine was doomed to failure before it got off the ground due to inadequate and unrealistic testing"*, according to a space consultant.

Testing is becoming more and more an important part of the design process. As the design itself, testing is becoming more focused, more specialized. Ad-hoc methods of testing cannot survive with the ever increasing design complexity.

This chapter starts by giving a brief introduction to IC testing in section 1.1. Subsection 1.1.1 shortly analyzes, from the test point of view, the two main IC design styles in use nowadays being analogue and digital.

Section 1.2 presents the most common fault models used in the test community. Subsection 1.2.1 gives a brief introduction about the stuck-at fault model. Subsection 1.2.2 shows the bridging fault model and subsection 1.2.3 presents the delay fault model. These fault models represent the fundamental knowledge required in order to understand the thesis.

Section 1.3 shows briefly the problem description for the solutions presented in chapters 3, 4, 5 and 6. At the same time, the motivation behind our research is given.

The outline of this thesis is presented in the last section (1.4).

1.1 Introduction to IC Test

The design complexity of integrated circuits (IC) today is increasing at an accelerated rate, driven by the decrease of the minimum feature size. Increased complexity can only be managed by a well organized hierarchical structure. At the architectural level, test strategies should follow the trend of the design, meaning hierarchical test, while at the transistor level, test fault models should be adapted to the new technology-related issues. Previously ignored effects, such as wire delays, will dominate the design-space exploration for the chips of tomorrow. Test will have to adapt to both complexity and new technology-related issues.

The ICs of today can be divided into two major classes: analogue and digital. Analogue designs process continuous data while digital designs operate on numerical/binary data.

1.1.1 Testing of Digital Systems versus Analogue Systems

Testing digital ICs is by far easier than testing the analogue ones and this section will present the reasons behind this statement. Table 1.1 summarizes some interesting features of analogue and digital ICs.

Table 1.1: Some differences between digital and analogue designs

	digital	**analogue**
allowed signal values	'0' and '1'	infinite number of levels
basic component devices	transistor (CMOS)	transistor, capacitor resistor, inductance, etc.
primary building blocks	AND OR NOT state-holding elements	the ones above
main testing method	structural logic gate level	functional macro level

While in a digital design any data signal can have only two values '0' (usually ground) or '1' (usually power supply), in an analogue design any value is acceptable. The range of signal values in an analogue block is design specific. From the test point of view, this first difference between digital and analogue is extremely important for the automation of the test strategies as will be seen in section 2.3. If one assumes Boolean algebra for the digital circuits, there are four primary building blocks for a synchronous digital design, three of them are combinational (AND, OR, NOT) and one is a state-holding element being a latch. By combining these blocks, one can obtain the core part of any complex synchronous digital system. For asynchronous designs, the state-holding elements are more diversified which will hinder the automatic test generation as will be seen later in 2.2.2 and 2.3.2.

For any digital circuit, the testing strategy will be applied to the logic-gate level and not at the transistor level, the only exception being IDDQ testing [11, 13]. This is an important remark, since for the analogue circuits there is no equivalent of the digital logic-gate level. The liberty of combining the basic devices in any way possible is, at the same time, the strength and the weakness of analogue designs. The strength of analogue design is that some functions can be implemented which occupy far less silicon area and consumes far less power than a digital version. The weakness of analogue design is that it is difficult to automate the design-and-test strategy and therefore difficult to build complex systems.

If automated test-pattern generators (ATPG) would have been able to generate test vectors from the transistor netlist level, then the difference between digital and analogue testing would not have been so abrupt. However, this is not possible due to the steps required to test a transistor and the inability to control and observe its terminals. Also the fault models are a problem as will be illustrated in the next sections.

1.2 Faults and Fault-Models for Digital Systems

Every manufactured digital IC on a silicon wafer is slightly different from its neighbors. Most of the time these differences do not influence the behavior of the IC since they have been anticipated at the design stage. However, some ICs are affected by these differences and do not perform as intended.

Each technology has its own types of physical faults which influence the

behavior of an IC. A fault may affect the logic function or the response time of an IC. Logical faults can be seen as a higher abstraction level of the physical faults. *"Logical faults represent the effect of physical faults on the behavior of the modeled system"* [1, page 93]. A logical fault can correspond to several physical faults. *"An explicit fault model defines a fault universe in which every fault is individually identified and hence the faults to be analyzed can be explicitly enumerated"* [1, page 93]. Once the faults are enumerated, test vectors can be generated to target each fault separately. Some of the faults will be hard or impossible to test. The fault coverage number associated with a set of test vectors is the number of the detected faults divided by the number of the total faults in the considered fault universe. References [1, 3] contain a very good classification of the faults and fault models.

1.2.1 The Stuck-at Fault Model

The stuck-at fault model [1, pages 110-122] considers that logic gates are fault-free and only the interconnections between them can be faulty being stuck-at '1' (s-a-1) or stuck-at '0' (s-a-0). While this assumption is not fully in accordance with the real manufacturing faults, the model is widely accepted due to its simplicity and the good results achieved in practice in the past. There are single and multiple stuck-at fault models. The single stuck-at fault model considers that only one stuck-at fault is present in a system at a time. The multiple stuck-at fault model considers that multiple faults can be present at the same time. Generating test vectors for the multiple stuck-at fault model is not a trivial task, since the total number of faults considered in a system increases exponentially with the number of gates. In practice, the single stuck-at fault model is used for generating the test vectors. Many algorithms have been proposed for generating the test vectors. Among them, are the D-algorithm [12], the 9-V algorithm [4], PODEM [6], FAN [5]. The test vectors generated for all the single stuck-at faults can detect many, but not all [1, page 120] of the multiple stuck-at faults that might be present inside a digital IC.

1.2.2 The Bridging Fault Model

The bridging fault model [1, pages 289-304] uses the same assumption as the stuck-at fault model: the faults can only be present between the in-

terconnection lines to the logic gates and not inside the logic gates. The *bridging faults* are shorts between two nets that are not connected in the fault-free design. The stuck-at fault model can be seen as a particular case of the bridging fault model because the nets are only shorted to the power supply (s-a-1) or ground (s-a-0). If two lines, having different values, are shorted, then the resulting value is technology specific. In CMOS technology, the resulting value is undetermined; in other technologies the value can be described as an OR or AND operation between the values of the nets. Similar to the stuck-at-fault model, there is the single and the multiple bridging-fault model.

1.2.3 The Gate/Path-Delay Fault Model

The bridging and the stuck-at faults are affecting the logic function of a system. A delay-fault [8], however, affects the propagation delay of a signal which in turn affects the speed at which a synchronous digital system can run. If a delay fault is present, the system will fail to run properly at the designed speed but will function correctly at a lower speed, if the fault is not present in the clock-distribution network. A delay-fault can be detected as a stuck-at fault if the test vectors are run at-speed.

Two models are very popular, the gate-delay fault model and the path-delay fault model. The gate-delay fault model considers that only gates might be affected by delay faults. The path-delay fault model assumes that the delay-faults are present in paths rather than individual gates. The gate-delay fault model is usually more pessimistic than the path-delay fault model, since a faster logic gate might compensate for a slower gate in the path-delay model. The path-delay model is more realistic due to the distributed character of the delays.

1.3 Problem Description and Motivation

All the fault models described above represent the starting point for generating test patterns which are able to test the complex digital IC designs in use nowadays. The fault models are the basic foundation for testing such ICs. Throughout the thesis these fault models will be referred to many times.

The complexity and speed of system-on-chip (SoC) designs are increasing and chip designers are trying to keep up with this trend. Recently designed video processors contain more than 90 clock domains and more than 50 cores. New design paradigms, like core reuse of already designed synchronous modules and asynchronous designs are considered in order to cope with the ever increasing complexity. The future digital SoCs will contain multiple synchronous and asynchronous cores. As the software industry has demonstrated, the reuse of previously designed modules is an efficient way for building complex systems. However, the hardware engineer has not only to cope with the increased complexity of the systems but also with the deviation of the parameters of the devices resulting from a particular technology.

Synchronous architectures have been the main design stream until now. Increased complexity, low power consumption requirements and speed of the future ICs will hamper this design style. It will not be possible anymore to operate the entire design synchronously with a single clock.

The main goal of our research project is to increase the test fault coverage of digital SoC designs by inspecting the trends and the shortcomings of the existing test strategies. The trends of SoC design are:

- synchronous intellectual property (IP) core-based design. Test strategies for these cores are delivered by the IP core vendor and many of them are scan-compatible. The P1500 standard [7] provides a common strategy to use the IP test vectors at the SoC level.
- more asynchronous cores will be embedded. Test problems must be addressed for these emerging designs. In reference [16], asynchronous handshake circuits are tested using the synchronous scan technique.
- more analogue and mixed-signal blocks like PLL, DLL, delay-lines and data converters will find their way into the main design stream. Embedded test strategies are being developed for these cores.
- at the interface between IP cores, specific modules will be designed to efficiently (low latency and bit error rates (BER) [9]) send the data between them. These small modules called *synchronizers* are essential for the good operation of the entire SoC system.
- high-speed interfaces between SoCs. The on-chip speed increase will demand a higher data-communication speed between different ICs in order to optimize the entire printed-circuit board (PCB).

All these new cores, modules and blocks should be tested. While the synchronous cores are usually scan-test compatible, the asynchronous ones are slowly becoming scan-test compatible. The true embedded analogue blocks cannot be tested by scan methods. They are usually tested in a functional way using off-chip test equipment. However, because of the increased SoC complexity, they should be tested on-chip or the test time might be prohibitively large. Built-In Self-Test (BIST) techniques must be used to test the relevant parameters of the analogue designs. There are BIST techniques available for some important mixed-signal blocks like the PLL [14], ADC and DAC [2, 15]. Obtaining low latency, when sending data between cores, requires advanced synchronizer schematics as the one that will be presented in section 3.1. Finding test solutions for these synchronizers, also referred to as on-chip interfaces, is a must since the entire SoC functionality depends on them. The same also holds for off-chip interfaces. Higher off-chip data-speed rates are demanded by the on-chip speed increase. As a result even more advanced high-speed interfaces are being designed.

Nowadays, it is possible to test, using synchronous scan or derivatives of it, single-clock domain (SCD) designs, multiple-clock domain (MCD) designs and some asynchronous designs.

This thesis will present test strategies for the modules used in SoC designs for which test strategies have not been developed yet. These are the advanced synchronizers and the high-speed interfaces. While this work tries to address many possible synchronizers, it is by no means exhaustive; this is especially true for the asynchronous-synchronous interactions. As for the high-speed interfaces we have not addressed the entire interface, but only the most common and difficult-to-test part of it, the high-speed delay-line.

1.4 Thesis Outline

A brief description of the topics addressed in this thesis are presented below.

Chapter 2 will introduce the necessary terminology for core-based designs. Timing relations between different synchronous cores are introduced in section 2.2.1. The asynchronous design style is briefly presented in section 2.2.2. Test strategies for the synchronous design styles will be introduced in section 2.3. These test strategies will be extended for the synchronizers and will be presented in chapters 3 and 4.

Chapter 3 introduces the design and test strategies for synchronizers used in synchronous-to-synchronous (StS) interactions. Five different synchronizer schematics are analyzed and test strategies are proposed based on the stuck-at and delay-path fault models.

Chapter 4 presents the design and test strategies for synchronizers used in asynchronous-to-synchronous (with stoppable clock) interactions. Both data directions are analyzed. First, their functional descriptions are presented in 4.1.1 and 4.2.1, then the proposed scan-test strategies are shown in 4.1.2 and 4.2.2 respectively.

Chapter 5 presents how the stand-alone test strategies of the synchronizers can be integrated at the SoC top level. Some extensions, at the SoC level, of the synchronizer designs addressed in chapters 3 and 4, are shown.

Chapter 6 shifts our focus from on-chip synchronization to off-chip interactions. On-chip high-speed design demands also higher data-transfer rates between different SoCs. The core part of most high-speed synchronization interfaces is a common tapped delay-line. Accurate tap delays must be guaranteed for a fault-free operation of the synchronization mechanism. Chapter 6 will present a method for accurately measuring the tap-delays using the oscillation test technique. A slightly modified scheme of a common delay line is presented which is better suited for our proposed technique. Measurements on a real chip fabricated in the UMC $0.18\mu m$ technology are also presented. The achieved measurement for the tap delays accuracy is within $\pm 10ps$ as required from industrial conditions.

Chapter 7 provides the summary and accomplishments of this work. Recommendations for future research are also given.

Bibliography

[1] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design (revised printing)*. ISBN 0-7803-1062-4. IEEE Press, 1990. 1.2, 1.2.1, 1.2.2

[2] Karim Arabi, Bozena Kaminska, and Janusz Rzeszut. A new built-in self-test approach for digital-to-analog and analog-to-digital converters. *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*, pages 491–494, 1994. 1.3

[3] Michael L. Bushnell and Vishwani D. Agrawal. *Essentials of Electronic Testing - for Digital, Memory & Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, second printing with corrections edition, 2001. ISBN: 0-7923-7991-8. 1.2

[4] C. W. Cha, W. E. Donath, and F. Ozguner. 9-V Algorithm for Test Pattern Generation of Combinational Digital Circuits. *IEEE Transactions on Computers*, C-27(3):193–200, March 1978. 1.2.1

[5] H. Fujiwara and T. Shimono. On the acceleration of test generation algorithms. *IEEE Transactions on Computers*, C-32(12):1137–1144, December 1983. 1.2.1

[6] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Transactions on Computers*, C-30(3): 215–222, March 1981. 1.2.1

[7] IEEE P1500. Standard for Embedded Core Test (SECT). `http://grouper.ieee.org/groups/1500/`. 1.3

[8] C. J. Lin and S. M. Reddy. On Delay Fault Testing in Logic Circuits. *IEEE Transactions on CAD*, 6(5):694–703, September 1987. 1.2.3

[9] David G. Messerschmitt. Synchronization in digital system design. *IEEE Journal on Selected Areas in Communications*, 8(8):1404–1419, October 1990. 1.3

[10] MPL. Mars Polar Lander. `http://mars.jpl.nasa.gov/msp98/lander/`, 1999. 1

[11] C.F. Hawkins R.K. Gulati. *IDDQ Testing of VLSI Circuits*. Kluwer Academic Publishers, December 1992. 128 pp., Hardbound, ISBN 0-7923-9315-5. 1.1.1

[12] J. P. Roth. Diagnosis of automata failures: A calculus and a method. *IBM Journal of Research and Development*, 10(4):278–291, July 1966. 1.2.1

[13] J.M. Soden, C.F. Hawking, R.K. Gulati, and W. Mau. IDDQ Testing: A Review. *JETTA*, 3(4):291–303, December 1992. 1.1.1

[14] Stephen Sunter and Aubin Roy. BIST for phase-locked loops in digital applications. *International Test Conference (ITC)*, pages 532–540, September 1999. 1.3

[15] Stephen K. Sunter and Naveena Nagi. A Simplified Polynomial-Fitting Algorithm for DAC and ADC BIST. *International Test Conference (ITC)*, pages 389–395, November 1997. 1.3

[16] Frank te Beest. *Full scan testing of handshake circuits*. Ph.D. thesis, University of Twente, May 2003. 1.3

Chapter 2

Test Challenges in Synchronous and Asynchronous Core-Based Design

SoC design can be perceived as a synonym for core-based design. Integrating different synchronous and asynchronous cores on the same die will become the predominant trend when designing very complex systems. Data synchronization between cores is one of the key design issues. Knowing the timing relation between the cores will help finding the proper synchronization modules which in turn will improve the speed behavior of the entire design.

This chapter starts by explaining metastability (2.1), a phenomenon that is always present if synchronization is required. Next, section 2.2 will present many possible types of interactions, from the timing point of view, between digital cores that comprise a SoC design. By interactions is referred to data exchange between the asynchronous and/or synchronous cores. Section 2.3 will provide a very brief overview of the existing test strategies and methodologies for synchronous and asynchronous designs. Finally, the chapter concludes with a summary of the presented issues so far.

2.1 Metastability

A Boolean digital signal is allowed to have only two values, '0', which is usually ground, and '1', normally the power supply. Each technology defines the maximum and the minimum allowed physical values for the '0' and '1' levels. If a signal value is larger than the maximum value of '0' and smaller than the minimum value of '1', then a combinational logic circuit (CLC) which has this signal as one of its inputs, might give completely random results. A deterministic finite-state machine (FSM) will in this case be transformed into a random one. The power consumption in CMOS technology will also increase if an intermediate signal level is present in a CLC.

These 'middle' forbidden values can be generated by state-holding elements if they sample signals that are changing at the same time with the sampling operation. Most of the state-holding elements have positive feedback loops to settle the output signal to a unique value, being either '0' or '1'. However, there is always a point of equilibrium in the middle where the signal can theoretically remain indefinitely. This particular state of the flip-flop is known as a metastable state. In practice, due to the noise present in the circuit, the metastable state will settle to a '0' or '1' after an unknown time. The well-known analogy of a metastable state is that of a ball being on a top of a hill. If a flip-flop enters a metastable state, then the probability (P) that this state will exist after the time t_w is proportional with:

$$P \sim e^{-t_w/\tau_s} \tag{2.1}$$

where τ_s is the regeneration time constant of the sense amplifier [2, page 469].

Metastable states are avoided in synchronous designs by allowing sufficient time for the CLC outputs to settle, before they are sampled by the state-holding elements such as the flip-flops. When sampling asynchronous signals, the easiest way to avoid a metastable state propagating in a system is by sampling the signal twice using two flip-flops connected in a pipeline-like structure as will be presented in section 3.1.1. In this case, the value t_w is equal to the receiving clock period. Every decrease of the receiving clock period will dramatically increase the probability of the metastable state to exist. Let us assume $t_w/\tau_s = 50$. By doubling the receiving clock frequency the probability is increasing by a factor of $e^{50/2} = 7.2 \cdot 10^{10}$. This is equivalent to say that instead of having a synchronization failure

every 136993 years there will be one every minute. More of such interesting calculations can be found in [9]. If the probability of a synchronization failure is increasing to an unacceptable rate, then one should either design a faster flip-flop or increase the time the metastable state is allowed to decay. The first option is not easy to implement, since the flip-flops of today are very well optimized for each technology. The second option is not appealing either since the gain of the high-speed design is lost waiting for the metastable state to decay. The only remaining option is to 'divert' the synchronization away from the data path in order to allow more time for the metastable state to settle and not affect the data-transmission latency [12] This is accomplished by designing advanced synchronization schemes, as will be presented in chapter 3, which are exploiting the predictability of the clock signals in order to reduce the latency of transmitting the data.

Metastability is addressed in many books and articles. For more information, the reader is encouraged to read the references [10, pages 120-131], [2, pages 468-470], [3] and [9].

2.2 On-Chip Interactions between Digital Cores

This section will investigate possible combinations of interactions between different digital cores in a SoC design. This classification is important since it gives an insight in the synchronization strategies that will be presented in chapters 3 and 4. First, the classical synchronous to synchronous interactions are analyzed in section 2.2.1. Then, the interactions between asynchronous cores are presented in section 2.2.2. The last two subsections, 2.2.3 and 2.2.4, present combinations of interactions between synchronous and asynchronous cores.

2.2.1 Synchronous - Synchronous Interactions

For this type of interactions all cores in the SoC are considered to be synchronous. Throughout this thesis, synchronous design is regarded as the same as single-clock domain (SCD) design. The phrase *"synchronous design"* will be used more if referring to a design style. The *"single-clock domain"* phrase will be used in the case of referring to testing of the synchronous designs. A template of a SCD design will be presented in figure 2.4.

All possible types of interactions between synchronous cores are explained in the next subsections.

Classification of Interaction between Synchronous Clock Domains

There is a high degree of predictability in synchronous systems that use a periodic clock. If two synchronous cores have to interact with each other then their clock periodicity can help optimizing the time required for the synchronization process. Therefore, it is important to investigate the possible synchronous clock-domains interactions [2]. Knowing the relation between the clock domains will help the design of the synchronizers presented in chapter 3.

Synchronous

If two or more synchronous cores are clocked by signals having the same frequency and phase, then they are said to be **synchronous** with each other. The data transfer between the synchronous cores can be performed without any additional synchronization mechanism. The present widely-used synchronous design style is part of this category.

Mesochronous

Clock distribution networks are used to spread the clock to all flip-flops in a design. For large designs is not easy anymore to distribute a synchronous clock to all the flip-flops, due to variations in technological parameters between different portions of the design. The "leaves" of the clock-distribution network will still have the same frequency as the "root" point but different phases.

If two synchronous cores are synchronized by the same clock frequency but the phase between their clocks is different and constant, then the relation between the clock domains is said to be mesochronous. This type of interaction will probably be the most common one in future synchronous digital designs.

Sending data from one core to another causes no problems as long as the receiving flip-flops do not sample the data during their aperture time. The aperture time of a flip-flop is the sum of setup and hold times of a flip-flop. Synchronizers that are able to efficiently send data between these types of clock domains are presented in sections 3.1.2 and 3.1.3.

Plesiochronous

Two or more synchronous cores are said to have a plesiochronous interaction with each other if their clock frequencies are almost the same. In time, the phase shift between the clocks of the cores is slowly changing. This situation can be encountered if two cores are separately clocked by two stand-alone oscillators that have been designed to have the same nominal frequency.

Data can be safely sampled for large time intervals using methods for mesochronous interactions. Additional circuitry can be added to adjust for duplicated or dropped data. Since the transmitter and receiver clock periods are mismatched, data can be lost, if the transmitter is fast, or duplicated, if the transmitter is slow, if the phase drifts over a clock period [2, page 480]. If the largest mismatch between the frequencies is known, the sender can insert null symbols from time to time in order to compensate for the effect described above.

These types of synchronizers can be seen in sections 3.1.2, 3.1.3 and 3.1.4.

Periodic

Two or more synchronous cores are said to have a periodic interaction with each other if their clock frequencies are completely different. It represents the most general case of synchronous to synchronous interaction. The periodic nature of the events can still be exploited to improve the average data-transmission latency of the synchronization. Because of the generality, the synchronizers used to efficiently send data between the cores are more complex. An example of such synchronizer is presented in 3.1.5.

All of the above synchronous to synchronous interactions cases can be summarized as shown in table 2.1. Parameters $\Delta\phi$ and Δf represent the phase and frequency difference between the transmitter and the receiver synchronous domains. Parameter ε is a small frequency error.

Table 2.1: Classification of signal-clock synchronization [2]

Classification	**Periodicity**	$\Delta\phi$	Δf
synchronous	yes	0	0
mesochronous	yes	unknown constant	0
plesiochronous	yes	slowly variable	$< \varepsilon$
periodic	yes	n/a	$> \varepsilon$

2.2.2 Asynchronous - Asynchronous Interactions

The next subsection is briefly explaining some of the multitude of asynchronous design styles in order to better understand the possible interactions between asynchronous cores.

Asynchronous Design Styles

A good overview of several possible asynchronous design methodologies is presented in [6]. It is obvious that asynchronous design is by far more diversified than its synchronous counterpart, due to the elimination of the time constraint that exists in synchronous designs. The classification in [6] is based on the timing models that are assumed.

The bounded-delay model

This is similar to the synchronous design style. It presumes that all delays of the logic gates and wires are known and therefore all signals in a design respond within a maximum bounded time. Instead of having state-holding elements in the feedback loops, there are delay elements to eliminate the hazards [2, page 405]. A logic gate output may perform several transitions before reaching the steady state due to the order in which the gate inputs are changing. The extra transitions are called a *hazard*.

The delay-insensitive model

This is the opposite of the bounded-delay model. It assumes that the delays of the logic gates and wires can have any value. The functionality of such a circuit will not be affected by delay faults.

The bounded-delay model and the delay-insensitive model represent the extreme cases for asynchronous design. In practice, a combination of the two models are used to build asynchronous designs. They are driven by power, area, speed, decomposition, etc. requirements.

Asynchronous Interactions

Asynchronous cores interact with each other by using three types of what is called Asynchronous Signaling Protocols [2, pages 489-492]. These types of interactions are briefly explained below:

Four-Phase Asynchronous Signaling

An interface, obeying this type of signaling has to perform the following four events in order:

1. all inputs are asserted
2. acknowledge is asserted
3. all inputs are deasserted
4. acknowledge is deasserted

For a bundle data transfer between asynchronous cores, the four-phase signaling protocol is shown in figure 2.1

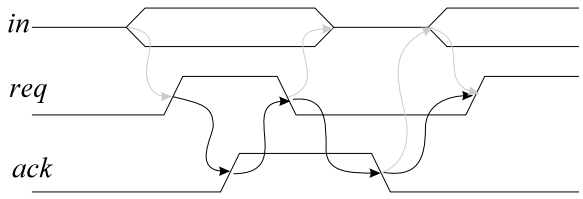


Figure 2.1: Bundled four-phase asynchronous signaling convention

The signal "*in*" in figure 2.1 is the data to be transmitted and should appear before the request signal "*req*" is set to logic one. When the receiver accepts the data, it assigns a logic one to signal "*ack*". The transmitter can now set the request signal "*req*" to logic zero and at the same time remove the data signal "*in*". The receiver responds by lowering the signal "*ack*". A new cycle can start after signal "*ack*" is set to a logic zero.

The black arrows represent the order of events at the interface between the transmitter and the receiver while the light-shaded arrows represent the order of events within the transmitter.

Two-Phase Asynchronous Signaling

The two-phase asynchronous signaling uses the same idea as in the case of the four phase but eliminates the last two steps (deassertion).

For a bundle data transfer between the asynchronous cores, the two-phase signaling protocol is shown in figure 2.2.

The signal "*in*" in figure 2.2 is the data to be transmitted and should appear before the request signal "*req*" is set to logic one. When the receiver accepts

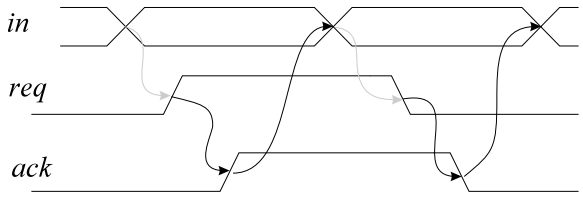


Figure 2.2: Bundled two-phase asynchronous signaling convention

the data, it assigns a logic one to signal "*ack*". The transmitter can now provide new data on the signal "*in*". Request signal "*req*" is set to logic zero to indicate that new data is available and a new data-transmission cycle is started.

Weak Conditions

In synchronous designs, the inputs and outputs of a system are changing at the same time due to the discrete timing imposed by the state-holding elements. In asynchronous designs the inputs and outputs can change at any time. Therefore, it is necessary to constrain the order in which these inputs and outputs are changing. A particular asynchronous module can have the following events of the inputs and outputs:

1. some input is asserted (sia)
2. all inputs are asserted (aia)
3. some output is asserted (soa)
4. all outputs are asserted (aoa).
5. some input is deasserted (sid)
6. all inputs are deasserted (aid)
7. some output is deasserted (sod)
8. all outputs are deasserted (aod)

If a non-return to zero (NRZ) protocol is used [2, page 412], then steps 5 to 8 should be ignored.

Figure 2.3 presents two constrains on event ordering being the *strong conditions* and *weak conditions*. A circle indicates an event and an arrow denotes a precedence relationship. If a NRZ protocol is used then deassertion will

not occur and the transitions of the gray lines should be followed. The steps 5 to 8 should be ignored.

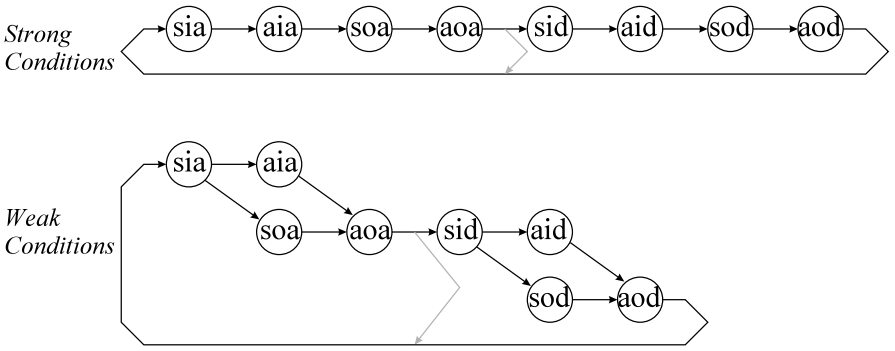


Figure 2.3: Event ordering for the strong and week conditions

Any circuit composed from modules that obey the weak condition without forming any cycles, will also obey the weak conditions [2]. Cycles should be understood as loops in the circuits.

2.2.3 Asynchronous - Synchronous (with free-running clock) Interactions

If asynchronous and synchronous cores have to interact, they usually use the two/four-phase protocols. The synchronous part will emulate this type of interaction by providing the "*request*" or "*acknowledge*" signals synchronously with its operational clock. Free-running clocks mean that the synchronous cores are synchronized with periodic clocks that cannot be stopped or changed in terms of their frequency during the normal operation. As opposed to free-running clocks, there are also stoppable clocks, where the clock of a synchronous IC can be stopped for reasons of power constraints, synchronization, or other purposes.

The only possible way to synchronize data between an asynchronous and a synchronous core, with free-running clock, is by using a brute-force synchronizer as will be presented in 3.1.1. Since there is no way of predicting when the asynchronous signal arrives, the average transmission latency is high compared with some fast methods for synchronizing two synchronous domains (see sections 3.1.2, 3.1.3 and 3.1.5).

2.2.4 Asynchronous - Synchronous (with stoppable clock) Interactions

A synchronous core using a static implementation of the logic gates can still operate correctly if the clock is stopped, or gated, from time to time. It should be noted that dynamic implementations of the logic gates, such as domino logic [7], can be affected by this mechanism.

As has been previously stated, brute-force synchronization is the only way of synchronizing data between asynchronous and synchronous cores with free-running clocks. A more efficient solution of sending data between a synchronous and an asynchronous core is by stopping the clock or by postponing the request/acknowledge signal coming from the asynchronous core, whenever they arrive nearly at the same time. Arbitration takes place in order to find out which of the two signals have arrived first. Since this process can last indefinitely, the clock can be stopped for an unknown time. This strategy is called stoppable clock in [2, 14]. The clock for the synchronous cores must be generated on-chip by, for example, a stoppable ring oscillator; it is unique for each synchronous core. Moreover, the generated clock signal is not periodic anymore. Therefore, the synchronizers presented in 3.1 cannot be used if the stoppable-clock strategy is utilized. A possible implementation of an interface that synchronizes a two-phase asynchronous core with a synchronous one using a stoppable clock is presented in reference [13] and in sections 4.1 and 4.2.

2.3 CAT Tools Availability

The "simple" Boolean mathematics is at the basis of any digital design, providing the computational power of any processor on the market. Boolean mathematics is the calculus key engine. The combinational logic maps any Boolean function to a technology-specific implementation. In order to build the present powerful processors, there must be some kind of control. This is accomplished with the help of state-holding elements like flip-flops, latches for synchronous design and C-elements, arbiters, etc. for asynchronous design.

Asynchronous and synchronous designs are the two main directions for digital design. In synchronous design, all the signals are synchronized with a single signal usually referred to as *clock*. The synchronization is performed

by the state-holding elements that are controlled by the clock signal. In asynchronous design, the state-holding elements are controlled by data signals or by signals coming from other state-holding elements. In synchronous systems the control is centralized while in asynchronous systems it is local.

In theory, combinational logic circuits (CLC), having no redundancy and considering the stuck-at fault model only, can be fully tested if all inputs are controlled and all outputs are observed. For synchronous digital design, the inputs (outputs) of a CLC can be controlled (observed) either directly from the IC pins or via modified state-holding elements called scan flip-flops (section 2.3.1). For asynchronous design, controlling and observing the CLC inputs and outputs is not as simple as for the synchronous design (section 2.3.2).

There are many companies that provide computer-aided test (CAT) tools for synchronous digital designs. Among them are, Mentor Graphics [5] (fastscan), Synopsys [17] (TetraMAX), SynTest [18] (TurboScan), LogicVision [11] (Logic BIST), etc. The main disadvantage of the commercial test tools is that they can handle only single-clock domain (SCD) designs well. However, some of the tools can parse **simple** multiple-clock domains (MCD) designs. These limitations will be further investigated in section 2.3.3.

There are no commercial CAT tools for asynchronous designs. Usually, CAT tools for synchronous designs are used for generating test vectors after the asynchronous design has been modified to be parsed by the synchronous CAT tool.

2.3.1 Single-Clock Domain (SCD) Testing

A SCD design can be represented as shown in figure 2.4. We have considered the design style where flip-flops are utilized as the state-holding elements. Some designs use simple latches instead of flip-flops [4]. Their testing strategy can be derived from the one presented below for the flip-flop design style.

In order to test the CLC, one should control the input pins "*pin[0:m]*" and the internal wires "*ff_in[0:n]*" and observe the output pins "*pout[0:k]*" and the internal wires "*ff_out[0:n]*". While for the input and output pins "*pin[0:m]*" and "*pout[0:k]*" it is possible to control and observe them, this is not the case for the internal wires. There are many ways to get access to

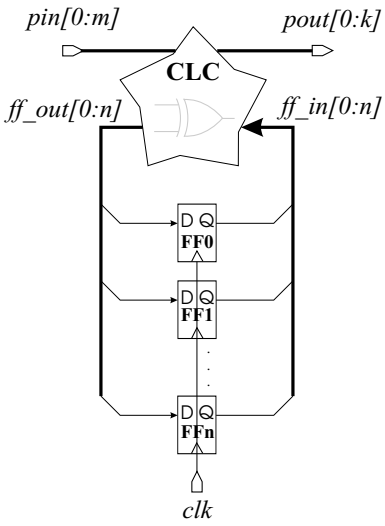


Figure 2.4: Single-Clock Domain (SCD) design

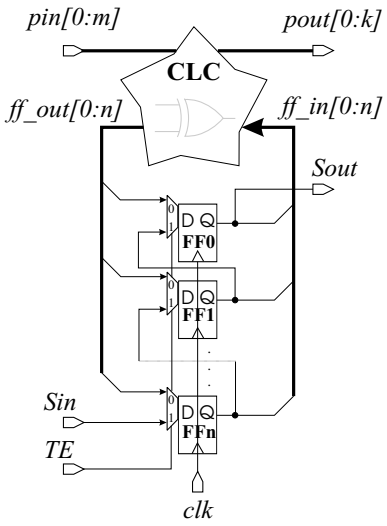


Figure 2.5: The scan chain present in the SCD design

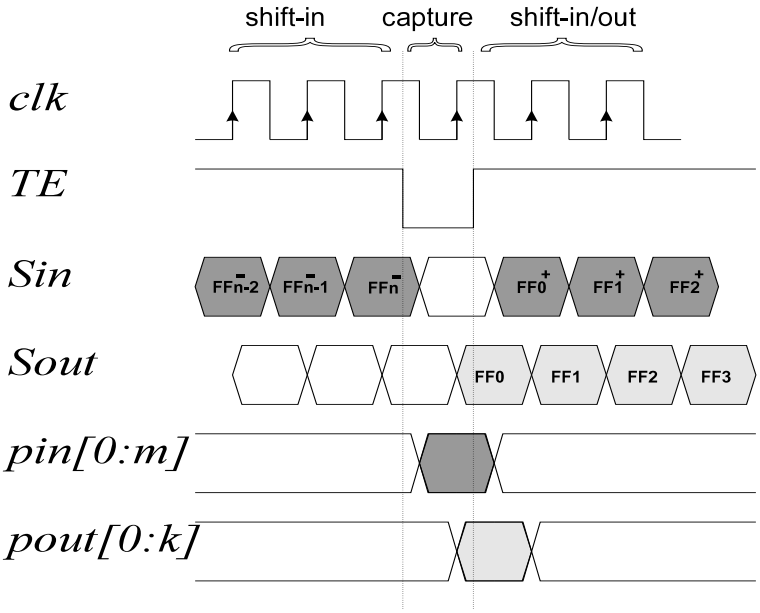


Figure 2.6: Waveforms associated with the scan-test strategy

the internal wires ([1], page 347). The most utilized technique is connecting all the flip-flops in a scan chain, as shown in figure 2.5. A pattern is shifted into the flip-flops which are part of the *scan-chain*, when the "*TE*" signal is high. If all the values of the flip-flops are well defined, the "*TE*" signal will become low, putting the design in functional mode. Then the primary inputs "*pin[0:m]*" are set to the required values. At this moment all input values to the CLC are controlled. While keeping the "*TE*" low, the design is clocked once. Next, the output pins "*pout[0:k]*" can be verified against the expected values. Then the signal "*TE*" will become high and the "*ff_out[0:n]*" wire values will be "scanned-out" via "*Sout*". The relevant signal waveforms involved in the scan-test strategy are depicted in figure 2.6. The flip-flops in the scan chain are first tested by scanning in and out some known values. This first test is also known as the *scan-continuity test*.

2.3.2 Asynchronous Design Testing

As seen in the previous section, any synchronous digital design can be described as a CLC part and an array of flip-flops interrupting all the feedback loops. This simplicity helped the development of test-generation procedures tremendously.

Asynchronous design is not as "organized" as its synchronous counterpart. There are many different types of implementations for the same problem. Because of its diversity, general test strategies are difficult to find. Poor controllability and observability are the main problems while testing asynchronous circuits. Moreover, many asynchronous designs contain deliberately introduced redundancy in their CLC in order to eliminate hazards.

Throughout the years, test strategies have been proposed for asynchronous designs [8]. They resemble ad-hoc test techniques, because each type of asynchronous design has it own particularities and hence test protocol. It is not possible to formulate a general test strategy as for the synchronous design. Recently, 4-phase handshake circuits have been tested by the scan technique used in synchronous circuits [16]. The main idea was to modify all the state-holding elements to be scan compatible.

Asynchronous designs are difficult to test, but they are a natural design trend given the high-speed technologies of today, where power consumption and clock distribution in synchronous designs are even more difficult to control. Test strategies for the most promising asynchronous designs should

be developed and their efficiency should be higher or at least equal with those available for the synchronous design.

2.3.3 Multiple-Clock Domains (MCD) Testing

The synchronous or asynchronous design strategies represent two extreme cases from the test point of view. The fully synchronous design is relatively easy to test, while the asynchronous design is the opposite. In-between lies the testing problem of multiple-clock domains systems. The problem has been addressed in many papers out of which only two [15, 19] are mentioned here. As the reader will notice in chapters 4 and 3, these test strategies will be enhanced for more complex designs that utilize advanced synchronization schematics.

An MCD system is comprised of more than one synchronous area. The relation between the different clock signals of the synchronous areas has already been explained in section 2.2.1. In order to minimize the complexity of test generation and application, there is usually only one clock utilized while testing a MCD system. However, if no care is taken when signals are crossing the clock domains border, the test strategy is compelled to fail. Even if there is only one clock in the test mode, the unknown clock skew between the synchronous areas will make the direct applicability of the synchronous scan-test strategy impossible. Caution is required if scan-data or functional data is crossing the boundaries between different clock domains. There are different methods for adapting the SCD scan-test strategy to the MCD design. One should either develop a more advanced test-application procedure, or add additional hardware between clock domains in order to simplify the test-application procedure. Our chosen method makes use of both of them and is going to be explained in the next two subsections.

Data Against Clock Strategy

This strategy is explained in reference [19]. It uses some of the information available about the clock skew between two clock domains. If there is knowledge about the skew between the clock signals, then the clocking can be performed against the data propagation. Figure 2.7 shows how a scan chain can be inserted if delayed versions of the clock are controlling some flip-flops. The delays should be less than half the test-clock period. It can be considered that each flip-flop has its own clock domain.

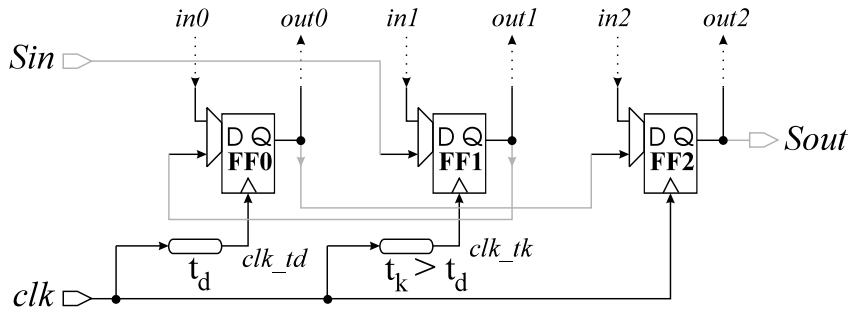


Figure 2.7: Scan-chain insertion for a delayed version of the same clock

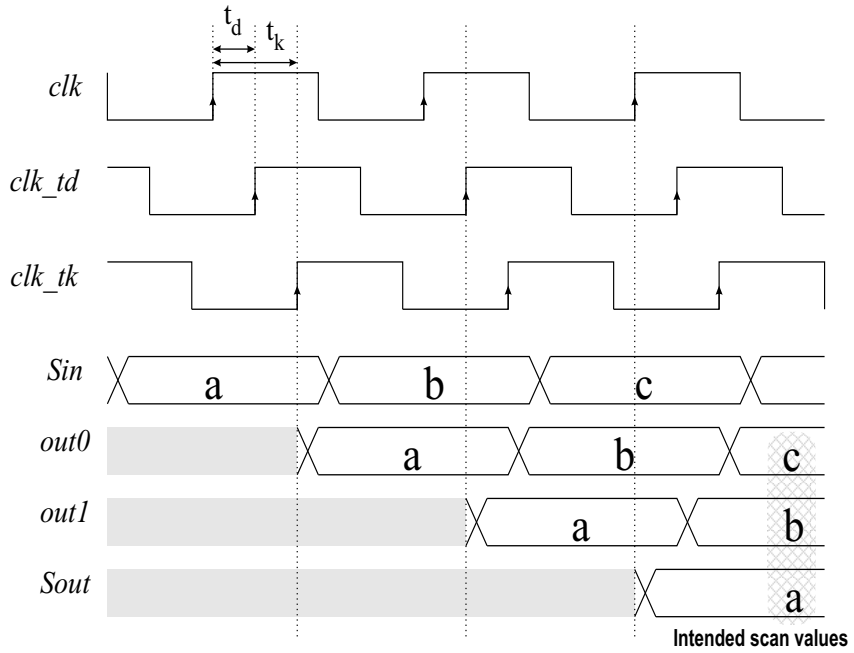


Figure 2.8: Waveforms for the scan operation of the scheme shown in figure 2.7

The waveforms associated with the scan mode of figure 2.7 are shown in figure 2.8. It can be seen that at the end of the scan-mode, the flip-flops in figure 2.7 will contain the intended values.

In scan mode, the system will behave as a SCD design. In the capture mode, following the scan-mode, the system will have a behavior similar to a SCD design as long as the functional inputs of the flip-flops are coming from the same domains as their clock or more delayed ones. Signal "*in0*" should be generated from a clock domain that is delayed by t_d or by $t_k \geq t_d$. Otherwise, the data-against-clock scan-test strategy will not function.

If the implementation of the scan chain does not follow the one in figure 2.7, then some of the flip-flops will become transparent in test mode, which will lead to a test failure.

Sometimes the clock skew between the synchronous domains is not known in advance. In this case, the data-against-clock strategy cannot be used. In order to cope with an unknown clock skew, some additional hardware will have to be inserted for a correct operation of the test strategy. This will be seen in the following subsection.

MCD Scan-Test Strategy

If the clock skew between different synchronous domains is unknown, the scan chain cannot be build using a data-against-clock strategy. The method proposed in [15] has been used and enhanced for our test strategies presented in chapter 3 and will be briefly explained in this subsection.

Let us consider figure 2.9, where two separate clock domains are shown. Between them there is a synchronizer module which will be described in detail in chapter 3. The off-chip communication is accomplished via the high-speed synchronization interface (HSI). For the time being, the synchronizer and the HSI modules will be ignored.

Any stand-alone synchronous domain can be tested relatively simply by using a scan-test strategy. Nevertheless, whenever there is an interaction between separate clock domains, the fault coverage drops if one attempts to test one clock domain at a time. Therefore, one has to simultaneously, scan-in, evaluate and scan-out for the two clock domains. A scan chain will have to be build consisting of flip-flops from the two clock domains. The solution presented in [15] is to modify the traditional scan cell in such a

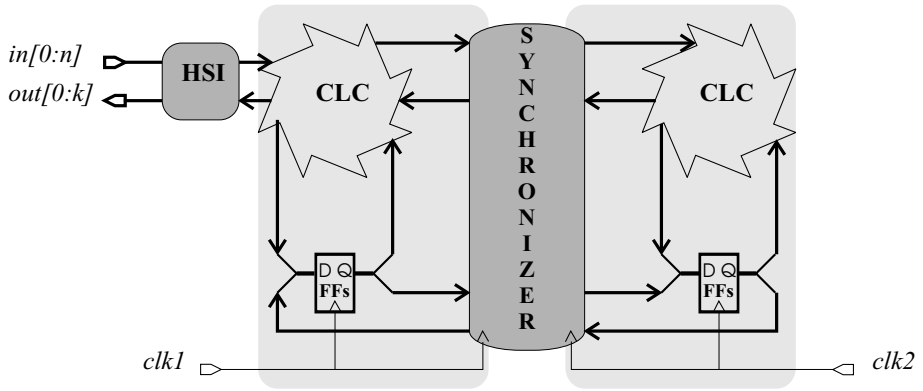


Figure 2.9: Generic architecture of a MCD design

way that it tolerates the skew between clock domains. We have chosen only one type of the MCD scan-cell out of the four that the authors present, and this is shown in figure 2.10.

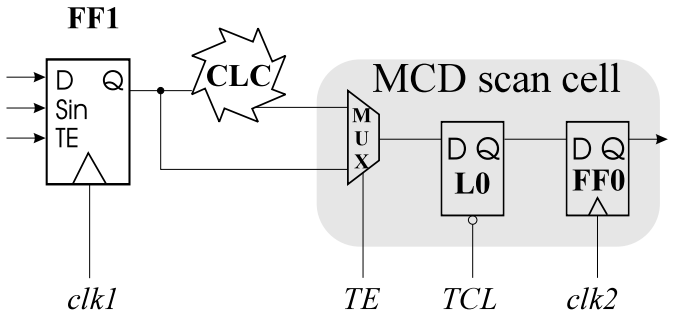


Figure 2.10: The MCD scan cell (shaded part)

The MCD scan cell in figure 2.10 has an extra latch **L0**, controlled by a new signal "*TCL*", inserted between the MUX and the **FF0** flip-flop. Without this latch, the MCD scan cell becomes a normal scan-cell. MCD scan cells should be introduced only at the border between the clock domains. The timing of the relevant signals in this new configuration is presented in figure 2.11.

By introducing the extra latch in the scan cell, the sender **FF1** and the receiver **FF0** flip-flops are decoupled. The signal "*TCL*" is accessible from an external pin and its timing is dependent on the estimated skew between the clock domains "*clk1*" and "*clk2*". Whenever two scan chains of dif-

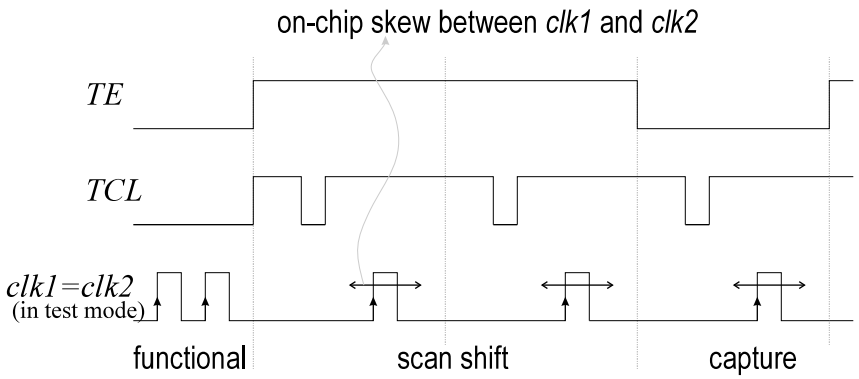


Figure 2.11: Timing for the MCD scan-test mode

ferent clock domains are concatenated, a MCD scan cell should be used. This small hardware modification, together with the addition of the "*TCL*" signal, allows the synchronous scan-test strategy to be also used for MCD systems which do not use advanced synchronization strategies as presented in chapter 3.

Most of the automatic test-pattern generation (ATPG) tools cannot deal with MCD designs. They only generate test vectors for fully synchronous SCD circuits. The MCD scan cell transforms the MCD test strategy into a SCD test strategy. However, the circuit netlist read into the ATPG tool should be fully synchronous. Therefore, a remodeling of the MCD system for ATPG must be carried out. For an MCD cell this is equivalent with the **L0** latch removal. Also the separate clock signals "*clk1*" and "*clk2*" are connected together. The resulting design can then be parsed by any commercial ATPG tool and test vectors can be generated. The "*TCL*" signal can be automatically generated from the clock signal as shown in figure 2.11. In scan-test mode, before every rising edge of the clock signal "*clk1*", signal "*TCL*" should be low.

2.4 Summary

This chapter has presented a brief overview of digital design approaches in use nowadays and their suggested test methodologies. Asynchronous and synchronous designs have been identified as the main two digital design styles. Test strategies for the synchronous design style have been briefly

explained. The scan-test methodology has been shown for a SCD design. For an MCD design, an extension of the scan-test has been presented. For the asynchronous design, the trend has been mentioned to make them testable using synchronous test strategies like the scan technique.

Particular attention has been given with regards to the interactions between the cores. For each type of synchronous interaction, specific interface schemes will be presented in chapter 3 together with their test strategies. As for the asynchronous-synchronous interfaces, the two-phase types have been addressed in the thesis and their test strategies will be given in chapter 4.

Bibliography

[1] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design (revised printing)*. ISBN 0-7803-1062-4. IEEE Press, 1990. 2.3.1

[2] William J. Dally and John W. Poulton. *Digital Systems Engineering*. ISBN 0-521-59292-5 (hb). Cambridge University Press, 1998. (document), 2.1, 2.2.1, 2.2.1, 2.1, 2.2.2, 2.2.2, 2.2.2, 2.2.2, 2.2.4

[3] Charles Dike and Edward (Ted) Burton. Miller and Noise Effects in a Synchronizing Flip-Flop. *IEEE Journal of Solid-State Circuits*, 34(6): 849–855, June 1999. 2.1

[4] E. B. Eichelberger and T. W. Williams. A Logic Design Structure for LSI Testing. *Proc. 14th Design Automation Conference*, pages 462–468, June 1977. 2.3.1

[5] FASTSCAN. *Mentor Graphics Corporation*. 8005 SW Boeckman Road, Wilsonville, OR 97070. `http://www.mentor.com/fastscan/`. 2.3

[6] Scott Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE, Vol 83, No.1*, pages 69–93, January 1995. 2.2.2

[7] David Hodges, Horace Jackson, and Resve Saleh. *Analysis and Design of Digital Integrated Circuits*. McGraw-Hill, July 2003. ISBN: 0072283653. 2.2.4

[8] Henrik Hulgaard, Steven M. Burns, and Gaetano Borriello. Testing

asynchronous circuits: A survey. *INTEGRATION, the VLSI journal*, 19(3):111–131, 1995. 2.3.2

[9] Jerry Jex and Charles Dike. A Fast Resolving BiNMOS Synchronizer for Parallel Processor Interconnect. *IEEE Journal of Solid-State Circuits*, 30(2):133–139, February 1995. 2.1

[10] Howard W. Johnson and Martin Graham. *High-Speed Digital Design (A Handbook of Black Magic)*. Prentice Hall PTR, April 1993. ISBN: 0133957241. 2.1

[11] LogicVision. *Embedded Test*. LogicVision, Inc., 101 Metro Drive, Third Floor, San Jose, CA 95110. `http://www.logicvision.com/`. 2.3

[12] David G. Messerschmitt. Synchronization in digital system design. *IEEE Journal on Selected Areas in Communications*, 8(8):1404–1419, October 1990. 2.1

[13] Simon Moore, George Taylor, Robert Mullins, and Peter Robinson. Point to point GALS interconnect. *Eighth International Symposium on Asynchronous Circuits and Systems*, pages 69–75, April 2002. 2.2.4

[14] S.W. Moore, G.S. Taylor, P.A. Cunningham, et al. Using stoppable clocks to safely interface asynchronous and synchronous subsystems. *AINT'2000 Asynchronous Interfaces: Tools, Techniques, and Implementations*, pages 129–132, July 2000. 2.2.4

[15] Josef Schmid and Joachim Knblein. Advanced Synchronous Scan Test Methodology for Multi Clock Domain ASICs. *IEEE VLSI Test Symposium (VTS)*, pages 106–113, April 1999. 2.3.3, 2.3.3, 2.3.3

[16] Frank te Beest. *Full scan testing of handshake circuits*. Ph.D. thesis, University of Twente, May 2003. 2.3.2

[17] TetraMAX ATPG. *High-Performance Automatic Test Pattern Generator Methodology Backgrounder*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043, May 1999. `http://www.synopsys.com/products/test/tetramax_wp.html`. 2.3

[18] TurboScan. *SynTest Technologies, Inc.* 505 South Pastoria Avenue, Suite 101, Sunnyvale, CA 94086, USA. `http://www.syntest.com/`. 2.3

[19] Bart Vermeulen, Maurice Lousberg, and Paul Merkus. Scan test tech-

niques for multi synchronous digital circuits. *Intenational Test Synthesis Workshop (ITSW)*, March 1998. 2.3.3, 2.3.3

Chapter 3

Test Strategies for Synchronizers Used in Multiple-Clock Domains

This chapter will analyze several examples of synchronous-to-synchronous synchronizers from the test point of view. These particular synchronizers are utilized to transmit/receive data from clock domains that are interacting with each other as explained in chapter 2, section 2.2.1.

In section 3.1 their design, as well as their functional description are presented. General test strategies for these types of synchronizers are introduced in section 3.2, as a starting point for the development of subsequent test strategies. Customized test strategies for most of the mesochronous, plesiochronous or periodic synchronizers are described in section 3.3 and 3.4 respectively. Finally, conclusions with respect to our implemented test strategies, are given in section 3.5.

3.1 Environment-Specific Synchronizer Schematics

As has been presented in chapter 2, there are several different ways in which the clock-domains are interacting with each other. Different types of synchronizers can be designed to take advantage of the periodicity of the clock-domains, in order to improve the average data-transmission latency.

The following subsections will present, for each type of possible clock-domain interactions, several synchronizer designs able to exploit the predictability of the clocks for a specific interaction.

The synchronizers to be analyzed are:

- The brute-force synchronizer (BFS). It is described in section 3.1.1 and is the simplest design possible. It can be used for all types of clock-domain interactions.
- The delay-line synchronizer (DLS), the two-register synchronizer (TRS) and the FIFO synchronizer (FIFOS) are presented in the sections 3.1.2, 3.1.3 and 3.1.4 respectively. These are mainly utilized to synchronize data between mesochronous (2.2.1) domains. However, they can be easily modified for plesiochronous (2.2.1) domain interactions.
- The periodic synchronizer (PS) is presented in section 3.1.5. It is able to safely pass data between synchronous domains with arbitrary frequencies. These domains are referred to as `periodic` in chapter 2 section 2.2.1.

3.1.1 The Brute-Force Synchronizer (BFS)

The brute-force synchronization is the simplest mechanism for synchronizing a signal that passes the border between synchronous domains. It can also be used to synchronize data coming from asynchronous domains. The simple scheme of a brute-force synchronizer (BFS) is presented in figure 3.1. This type of structure will be present in all synchronizer designs.

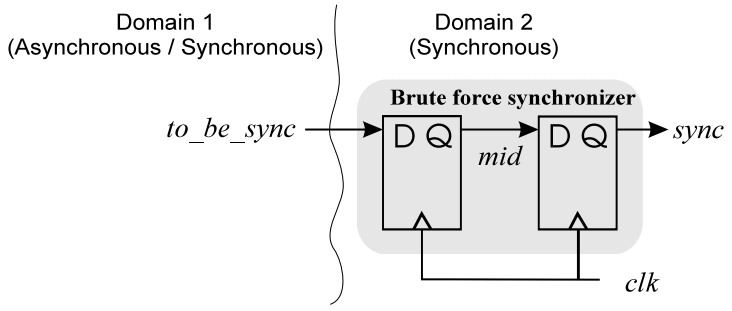


Figure 3.1: The brute-force synchronizer (BFS)

The two flip-flops are connected in a pipeline-like structure in order to

decrease the probability of a metastable state (2.1, [1, page 469]) being able to propagate to the receiving clock domain.

Figure 3.2 shows the functional behavior of the BFS when the signal to be synchronized ("*to_be_sync*") is changing simultaneously with the receiving clock "*clk*". The signal "*mid*" is in a metastable state which is filtered out

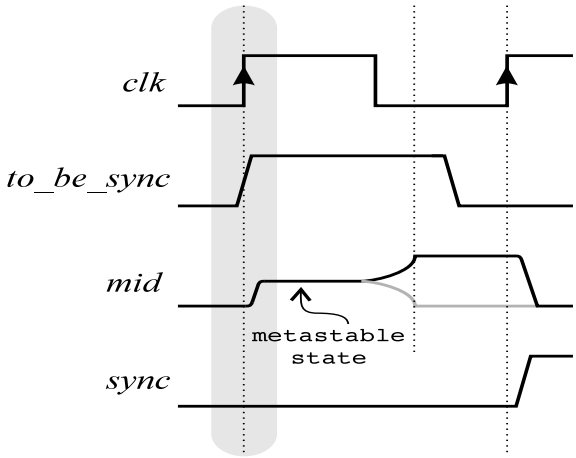


Figure 3.2: Waveforms associated with the brute-force synchronizer (BFS)

by allowing sufficient time to settle to either a logic one or zero, before the second flip-flop is sampling the data again.

As a result of its simplicity and versatility, the BFS features the highest average transmission latency (average time necessary to send data between the transmitting clock domain and the receiving one) among the synchronizers. This is approximately $3 \cdot T_{cy}/2$, where T_{cy} is the period of the clock signal "*clk*".

The BFS can be used in any circuit where the transmission latency is not a crucial design parameter. The BFS design is small and easy to test using the MCD scan strategy as presented in section 2.3.3 and therefore it will not be analyzed for testing in the next sections. It has been briefly described here for the completeness of the classification.

3.1.2 The Delay-Line Synchronizer (DLS)

The DLS, presented in figure 3.3, is mainly utilized to synchronize data

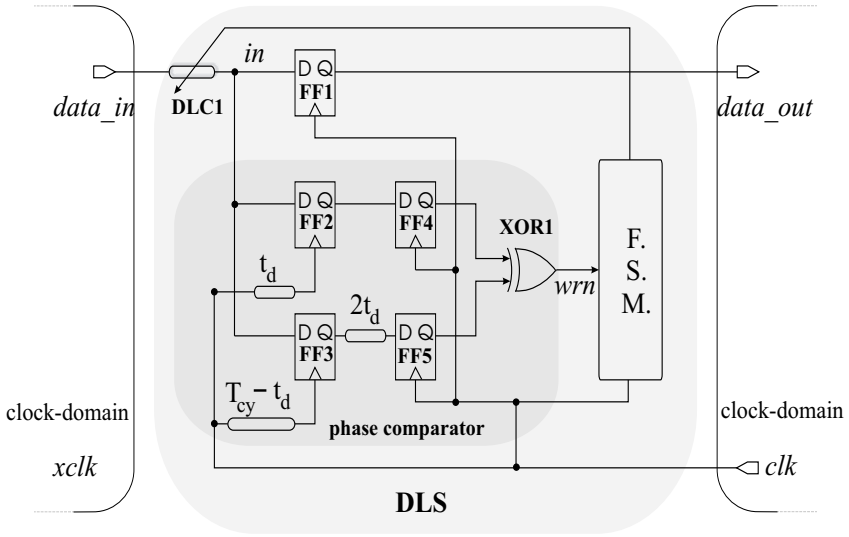


Figure 3.3: The delay-line synchronizer (DLS)

between synchronous clocks that are mesochronous (2.2.1) with each other. However, with small modifications, it can also be used in plesiochronous (2.2.1) clock-domains environments.

The DLS comprises of three important parts:

1. a flip-flop **FF1** and a controllable delay-line **DLC1** which are part of the data path.

2. a phase comparator consisting of the flip-flops **FF2**-**FF5**, three delay-lines having associated delays of t_d, $2t_d$, and $(T_{cy}-t_d)$ and an exclusive OR gate **XOR1**. The parameter T_{cy} is the period of the clock signal "*clk*".

3. a finite state machine (**FSM**) controlling the delay-line **DLC1**

Figure 3.4 shows the signals of the DLS in the functional mode. The "*data_in*" signal, synchronized with the "*xclk*" clock-domain is sufficiently delayed by the controllable delay-line **DLC1**, in order to be safely sampled by the **FF1** flip-flop which is part of the "*clk*" clock domain. The signal "*data_in*" should not change when the receiving clock signal "*clk*" is changing. In order to detect this situation, a `keep-out` region has been defines as seen in figure 3.4.

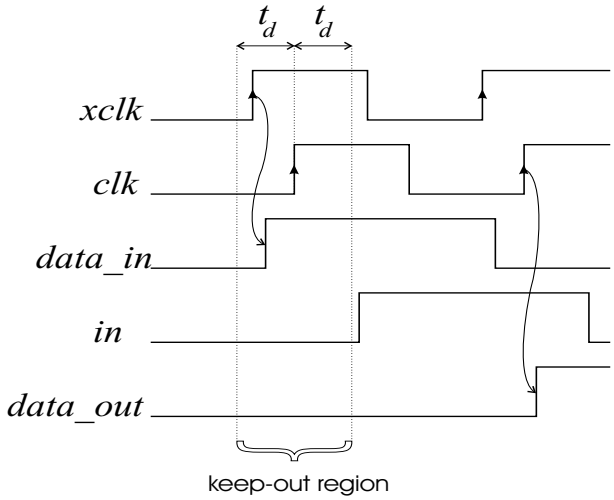


Figure 3.4: DLS signals in functional mode

The phase comparator detects if the "*data_in*" signal is changing within the $(-t_d, t_d)$ `keep-out` region of the rising edge of the clock signal "*clk*". The **FSM** is using this information to adjust the controllable delay-line **DLC1**, by selecting the smallest delay possible as seen in figure 3.4.

Recommendations of how to select the t_d delay are given for the two-register TRS on page 41. The presented analysis can also be applied for the DLS by considering $t_k \approx t_d$ since the sampling point is very close to the $(-t_d, t_d)$ keep-out region.

The delay block $(T_{cy} - t_d)$ in the phase comparator is particularly hard to design; it is dependent on the "*clk*" clock-signal period which in turn makes the whole phase comparator technology dependent. For a faster technology (higher clock speeds), the delay block $(T_{cy} - t_d)$ must be redesigned. Furthermore, the silicon-area occupied by this delay-line can be larger that the synchronizer itself.

The new phase comparator presented in figure 3.5 eliminates these problems by delaying the data with respect to the clock. Because of this modification, the $2t_d$ delay-line is no longer necessary.

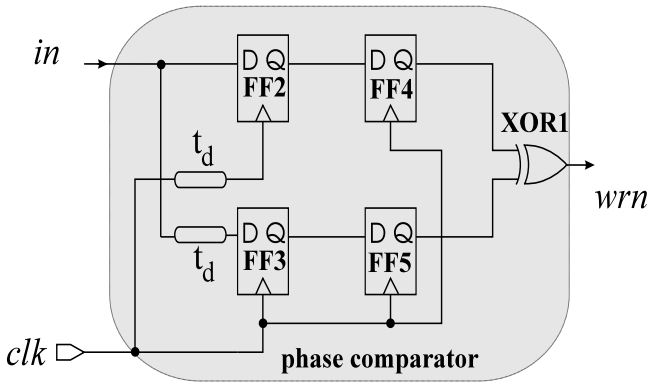


Figure 3.5: Technology-independent basic phase comparator (ΦC)

3.1.3 The Two-Register Synchronizer (TRS)

A two-register synchronizer (TRS) can be utilized in either mesochronous or plesiochronous environments (2.2.1). The basic scheme of a TRS is presented in figure 3.6. The "*data_in*" signal, which is synchronized with the "*xclk*" clock domain, is safely sampled by at least one of the two flip-flops **FF0** or **FF1**.

The phase comparator of the TRS detects if the rising edge of the "*xclk*" clock signal occurs within the $(-t_d, t_d)$ range of the rising edge of the "*clk*" clock signal. If this is the case, then the signal "*wrn*" is '1' and the **FF1** flip-flop will be selected to safely sample the "*data_in*" signal. In contrast to the DLS, the ΦC phase comparator analyzes two clock signals instead of a clock and a data signal.

Figure 3.7 shows the relevant waveforms for the TRS. In this figure the rising edges of the clock signals "*xclk*" and "*clk*" are close to each other. The phase comparator detects this situation and asserts the signal "*wrn*". Therefore, the flip-flop clocked by the signal "*clk_tk*", will be selected to drive the signal "*data_out*".

Two delay values are utilized in this scheme; the first one, t_k, represents the delay until the second sampling point of **FF1**, while the delay t_d is half of the `keep-out` region, which is centered around the rising edge of the clock signal "*clk*". These three delays are indicated in figure 3.8. Area "II" represents the `keep-out` region.

For the TRS, the t_d and t_k delay values are fixed and are to be decided

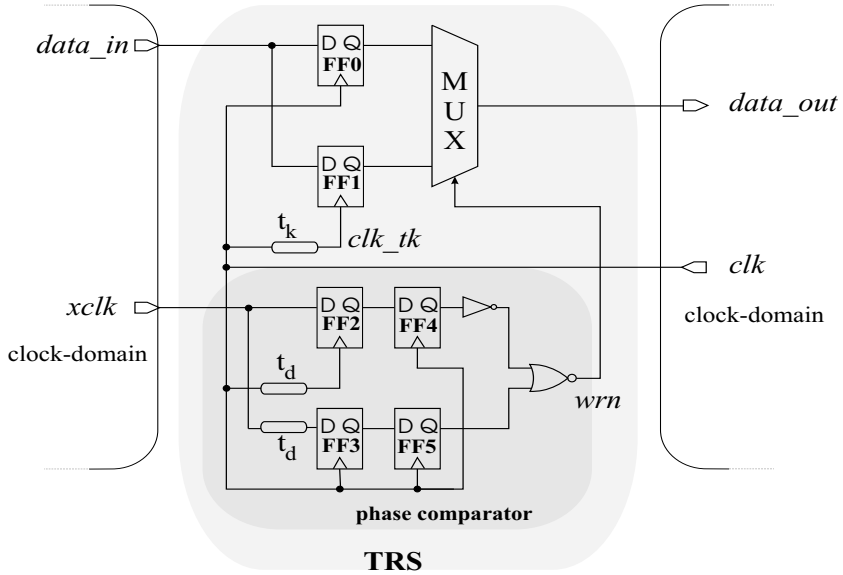


Figure 3.6: The two-register synchronizer (TRS)

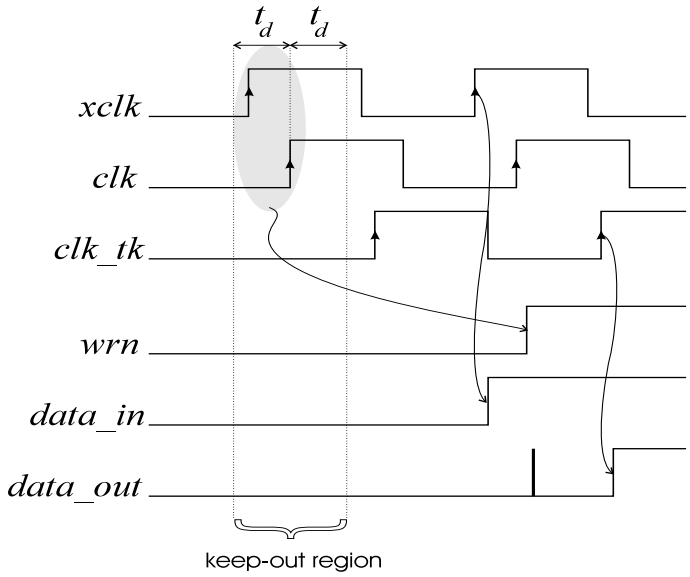


Figure 3.7: Typical waveforms for a two-register synchronizer (TRS)

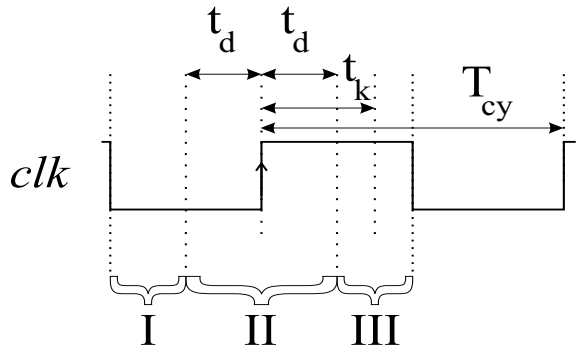


Figure 3.8: Relevant delays and regions for the TRS

in the design phase. The next paragraph is investigating how these delays should be chosen in order to minimize the average synchronization latency.

The average synchronization latency, denoted by $\overline{st}(t_d, t_k)$, is dependent on the t_k and t_d values and is given in relation 3.1. It is assumed that the phase shift between the two clock-domains has a random uniform distribution.

$$\begin{aligned}\overline{st}(t_d,t_k) &= \frac{2t_d}{T_{cy}}\cdot\frac{(t_k+t_d)+(t_k-t_d)}{2}+\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}\cdot\frac{\frac{T_{cy}}{2}+t_d}{2}+\\ &+\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}\cdot\frac{T_{cy}-t_d+\frac{T_{cy}}{2}}{2}\end{aligned} \tag{3.1}$$

where:

- $\frac{2t_d}{T_{cy}}\cdot\frac{(t_k+t_d)+(t_k-t_d)}{2}$ is the probability ($\frac{2t_d}{T_{cy}}$) of the "*data_in*" input signal to be in region II, as illustrated in figure 3.8, multiplied by the average synchronization latency ($\frac{(t_k+t_d)+(t_k-t_d)}{2}$) of the same region. Note that in this region II, the "*data_in*" input signal is sampled by the **FF1** flip-flop (see figure 3.6).
- $\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}\cdot\frac{\frac{T_{cy}}{2}+t_d}{2}$ is the probability ($\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}$) of the "*data_in*" input signal to be in region I, multiplied by the average synchronization latency ($\frac{\frac{T_{cy}}{2}+t_d}{2}$) of the same region. Note that in this region I, the "*data_in*" input signal is sampled by the **FF0** flip-flop (see figure 3.6).
- $\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}\cdot\frac{T_{cy}-t_d+\frac{T_{cy}}{2}}{2}$ is the probability ($\frac{\frac{T_{cy}}{2}-t_d}{T_{cy}}$) of the input signal to

be in region III, multiplied by the average synchronization latency ($\frac{T_{cy}-t_d+\frac{T_{cy}}{2}}{2}$) of the same region. Note that in this region III, the "*data_in*" input signal is again sampled by the **FF0** flip-flop.

Relation 3.1 may be rearranged as:

$$\overline{st}(t_d, t_k) = \frac{T_{cy}}{2} - t_d + 2\frac{t_d \cdot t_k}{T_{cy}} \tag{3.2}$$

The above function represents the average synchronization latency as a function of the t_k and t_d delay values. The flip-flops and multiplexer propagation delays have been ignored for a better understanding of the reasoning. The delay values t_d and t_k should be chosen to minimize $\overline{st}(t_d, t_k)$, while $t_k \geq t_d$ since the sampling point should always be outside the keep-out region.

The minimum value of the function $\overline{st}(t_d, t_k)$ is achieved if $t_k = t_d = T_{cy}/4$ and is $\overline{st}(t_d, t_k) = 3 \cdot T_{cy}/8$. In practice, due to uncertainties, the value of t_k should be chosen higher than the value of t_d. For design simplicity, a value of $t_d = t_k/2$ has been chosen.

The reasoning with regards to the delay values has been mainly performed in order to dismiss the belief that one should keep the values of t_d and t_k as small as possible and therefore very close to the aperture time of the flip-flop. This is not the case and this conclusion gives the designer much freedom for choosing values for t_d and t_k.

3.1.4 The FIFO Synchronizer (FIFOS)

The First-In First-Out synchronizer (FIFOS), shown in figure 3.9, can be used to synchronize data between mesochronous or plesiochronous domains (2.2.1). When one flip-flop samples the signal "*data_in*", the other flip-flop is driving the synchronized signal "*data_out*".

No delay-lines of any sort or phase comparators are included in the design. From this point of view, the FIFOS resembles the BFS. However, the arrangement of the sampling flip-flops differs, since these are not connected in a pipeline-like manner but rather in parallel. This placement does not improve the transmission latency as compared to the BFS, but allows more time for the synchronized signal to settle. This reduces the probability of

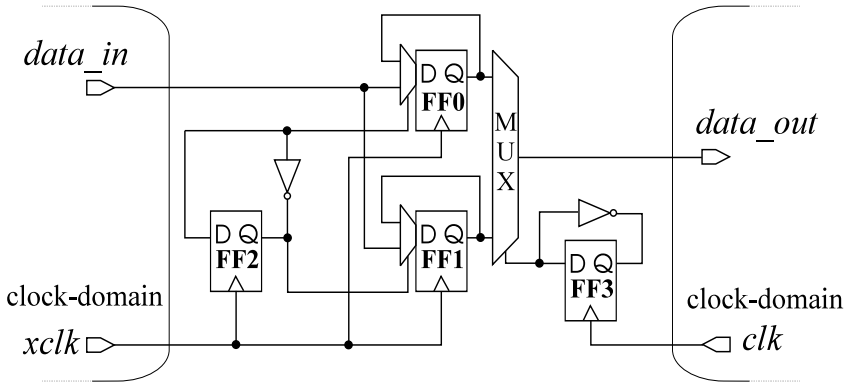


Figure 3.9: The first-in first-out synchronizer (FIFOS)

the metastability state to propagate to the receiving clock domain. The FIFOS allows the "*data_in*" signal to settle between two "*clk*" clock periods and one "*clk*" clock period, depending on the phase shift between the receiving and transmitting clocks.

Testing the FIFOS can be performed by directly applying the MCD scan-test strategy presented in 2.3.3, since it only comprises of flip-flops and usual digital logic gates like inverters and multiplexers.

3.1.5 The Periodic Synchronizer (PS)

This type of synchronizer is utilized for reliable transmission of data between clock-domains that are clocked by completely different frequencies. The relation between such clocks has been defined as *periodic* in section 2.2.1. The PS design is in fact the TRS design augmented with an analog module called **predictor**. This module is used to predict the value of the transmitting signal clock "*xclk*", one receiving "*clk*" clock period in advance. The PS, including the **predictor** module, is shown in figure 3.10. Figure 3.11 shows the typical waveforms for a PS. The "*pxclk*" signal is the "*xclk*" clock signal predicted one period in advance of the "*clk*" clock. The "*pxclk*" signal is compared by the phase comparator (ΦC) with the "*clk*" clock signal to ensure that there will be no set-up or hold time violations when the next rising edge of the clock signal "*clk*" arrives. In figure 3.11, the "*pxclk(td)*" and "*clk(td)*" signals represent the "*pxclk*" and "*clk*" signals delayed by t_d. One may notice the "*keep-out*" signal change from zero

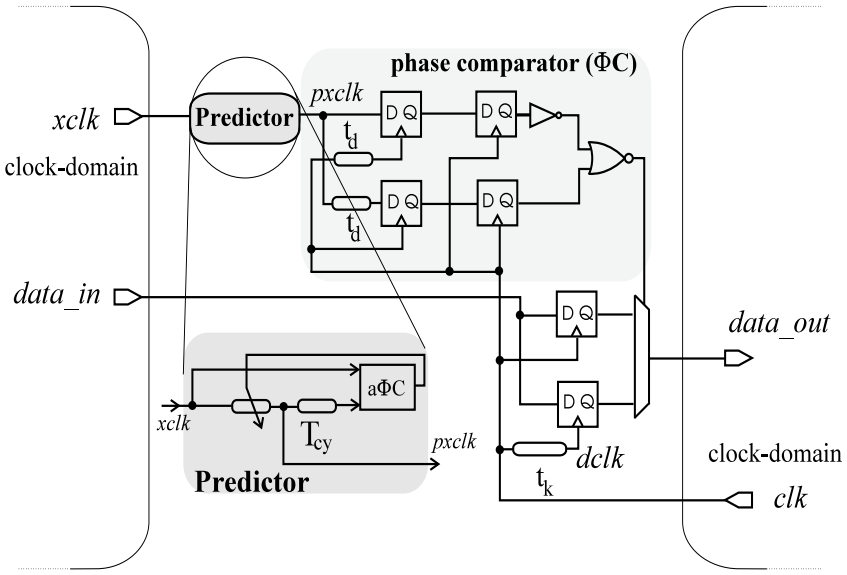


Figure 3.10: The periodic synchronizer (PS)

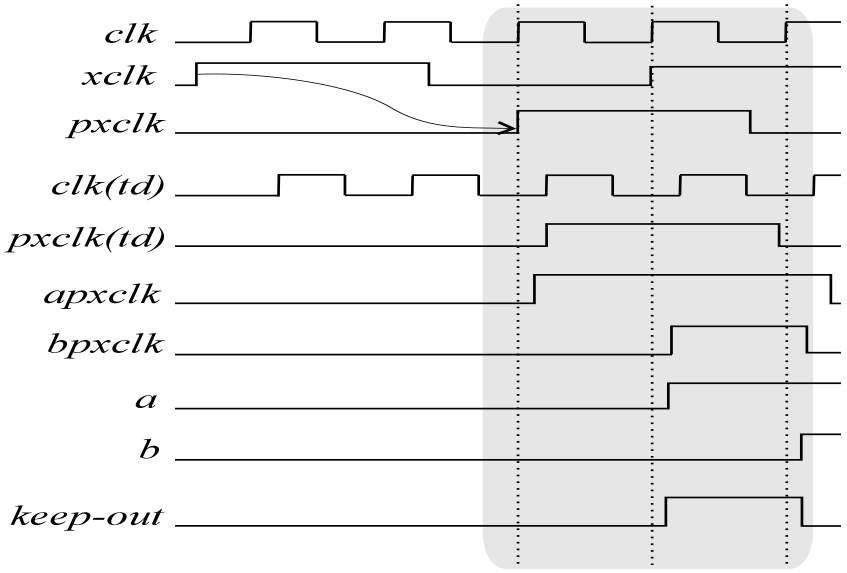


Figure 3.11: Typical waveforms for a PS

to one when the "*xclk*" clock signal is simultaneously changing with the "*clk*" clock signal.

While the idea of such a design is very interesting, the implementation of the **predictor** module is very rigid and cannot be adapted to a new technology without a major redesign of the **predictor** part. The delay value T_{cy} is equal with the "*clk*" clock period. If a new "*clk*" clock period is considered, this delay should be redesigned. In other words, the clock frequency of "*clk*" should be precisely known at the design stage. This is a major drawback for the present technologies which have a short development cycle. For some designs it can be even unacceptable. The analogue phase comparator ($a\Phi C$) and the controllable delay-line inside the **predictor** have a predefined operation range. Each time the sender frequency is changed, a redesign of the two previously mentioned parts must be performed. The silicon area of the **predictor** module is also quite high compared to the rest of the PS.

Because of its major design challenges, the PS should only be used whenever a low latency of sending data between clock-domains is required. If the latency parameter is not so important, then a BFS can be used instead which occupies a less area, and can be easily tested.

3.2 General Test Strategies for Synchronizers

The following sections will present the developed test strategies for most of the synchronizers discussed in the previous section. Not all of them are analyzed, since some are trivial cases from the test point of view and they can be tested using the already developed MCD scan-test strategy as presented in reference [3] or section 2.3.3.

All the synchronizers presented in this chapter are utilized in functional mode to safely pass data between synchronous domains. During the test phase, all the surrounding synchronous domains are put in scan-test mode. Therefore, in order to improve the fault coverage, it seems natural that the synchronizers should also be tested using scan-test methods. If this is not entirely possible, the synchronizers should at least be scan-compatible when the synchronous cores surrounding them are tested. This is necessary in order not to hinder the system-level test strategy. Two main goals for the DfT hardware and the test strategy have been set:

- the synchronizers including their associated DfT hardware, should have a stuck-at fault coverage of 100%
- the delay-lines inside the synchronizers should be tested for delay-faults.

Table 3.1 analyzes all synchronizers described in the previous sections (3.1) from the testing point of view.

Table 3.1: Synchronous-to-synchronous synchronizers

Clock domain relations	Types of synchronizers	ΦC	Simple delay-line in		Controllable delay-line in		Clock signals in data path
			Clock lines	Data path	Clock lines	Data path	
Any type	Brute force	NO	NO	NO	NO	NO	NO
Mesochronous or Plesiochronous	Delay-line	YES	YES	NO	NO	YES	NO
	Two-register	YES	YES	NO	NO	NO	YES
	FIFO	NO	NO	NO	NO	NO	NO
Periodic	Periodic	YES	YES	YES	YES	YES	YES

As can be observed in table 3.1, some of the synchronizers comprise of delay-lines and phase comparators. Since the scan-test methodology has been mainly developed for testing stuck-at faults in digital designs, it cannot be successfully utilized for testing the correctness of implementation of the delay-lines and phase comparators. In order to test them, timing resolution of less than one clock cycle has to be included in the test mode. To accomplish this, a built-in self-test (BIST) strategy has been considered. At the same time, the synchronizers should be scan-compatible. This implies that any design has to consist of only combinational logic circuits (CLC) and scan flip-flops. Moreover, the clock has to be distributed synchronously to all flip-flops in the scan chain. The BFS presented in subsection 3.1.1 is the easiest to test using only MCD scan methods because it does not include any hard-to-test blocks such as phase comparators or delay-lines. A BIST structure is not required for such a synchronizer. In contrast, the PS is the most difficult one to test since it contains all the difficult-to-test blocks.

Our generic test strategy for synchronizers is presented in figure 3.12. The surrounding synchronous blocks clocked by "*xclk*" or "*clk*" clocks are partially shown. The signal "*data_in*" is synchronized with the clock signal

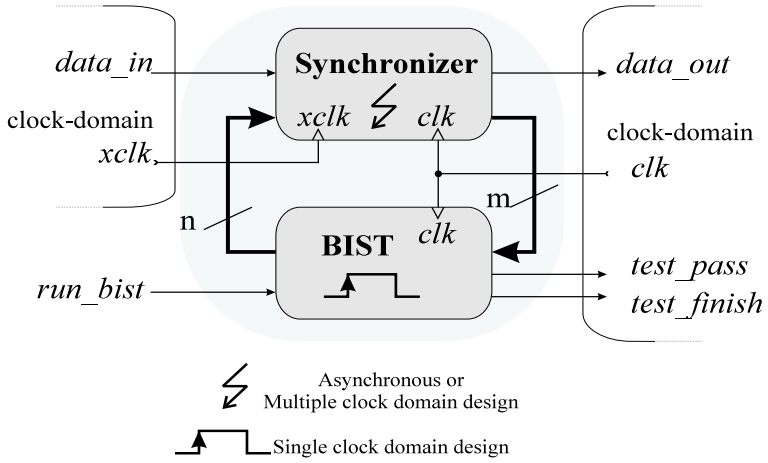


Figure 3.12: General test strategy for synchronous synchronizers

"*xclk*", while the "*data_out*" signal is synchronized with the clock signal "*clk*". One can notice that the **BIST** module is fully synchronous with the receiving clock. All of the flip-flops in the synchronizers are clocked by the receiving clock ("*clk*") or delayed versions of it. This is why the **BIST** clock has been chosen to be the receiving clock of the synchronizer ("*clk*"). Since the **BIST** is fully synchronous with the receiving clock, the scan-test methodology can be employed to verify its correct implementation.

The structure in figure 3.12 will be tested in two steps.

1. Scan Test for MCD. During this step, the synchronizer will be replaced with a special derived model for the sole purpose of ATPG. This will be done in order to be able to generate test vectors for the synchronizer, BIST and the surrounding synchronous blocks, using the MCD method described in [3] and briefly presented in 2.3.3. The remodeled (ATPG) version of the synchronizer will include as many parts as possible in the scan-test, but inevitably will leave out some of them, like for example delay-lines.

As a result, the synchronizer itself is not fully tested. However, since the BIST is fully tested by scan, it can be used during the next step described below, to test the remaining untested faults in the synchronizer.

2. Built-In Self-Test (BIST). If the scan-test has been successful, it can be concluded that the **BIST** is not faulty and can be used to test the remaining stuck-at faults and delay-faults in the synchronizer. The **BIST** structure is activated by the "*run_bist*" signal. When this occurs, the **BIST** will start applying specific test patterns in order to test the remaining stuck-at faults which have not been covered by scan, and the delay-faults that might be present in the critical components of the synchronizer, such as the delay-lines. If the test is completed successfully, the "*test_pass*" and "*test_finish*" signals will have a logic one assigned to them, otherwise the "*test_pass*" will have a logic zero assigned to it.

This combined test strategy of MCD Scan and BIST will be further referred to as **SaB**.

At the top level, the synchronizer with its associated BIST has only 7 in/out signals, being "*data_in*", "*data_out*", "*clk*", "*xclk*", "*run_bist*", "*test_pass*" and "*test_finish*". Four out of these seven signals belong to the synchronizer. As it will be seen later on, the delay-lines and phase comparators will be implemented as technology-independent as possible. Therefore, the synchronizer can be part of a synchronization library where each synchronizer will have the following information associated to it:

- the types of clock-domains the synchronizer can be used for (functional-information)
- the remodeled scheme, for the purpose of ATPG, used to generate test vectors (scan-information)
- the names of signals for activating the BIST (BIST-information)
- the names of signals for observing the results of the BIST (BIST-information)

During the synthesis process, a CAD tool could select the proper synchronizer design if there is a need for synchronizing data between clock-domains. With the above information available, it is possible to generate scan-test vectors and integrate the BIST strategy of the synchronizer at the chip level. This process could even become semi-transparent to the digital designer, in the same way synthesis is nowadays.

There is one tool on the market that incorporates this kind of automation when testing memories, PLLs, MCD logic blocks, etc. The company that provides the tools is LogicVision [2].

The following sections will introduce the test strategies for the DLS, TRS and PS synchronizers. For the rest of the presented synchronizers in the previous sections, an MCD scan-test strategy (2.3.3) will be sufficient. While developing the BIST structures, the single stuck-at fault and path delay-fault models will be assumed.

3.3 Test Strategies for Mesochronous and Plesiochronous Synchronizers

Within the category of mesochronous and plesiochronous synchronizers belong the DLS (3.1.2), the TRS (3.1.3) and the FIFOS (3.1.4). The next subsections will only present test strategies for the DLS and TRS. The FIFOS will not be analyzed since its design is very simple and a normal MCD scan-test strategy (2.3.3) is sufficient for testing it.

3.3.1 Test Strategy for the DLS

This type of synchronizer has been presented in section 3.1.2. The controllable delay-line can be designed using only digital logic gates. However, this does not imply that it can be easily tested using the scan-test methodology.

Let us consider a functionally simplified delay-line as the one shown in figure 3.13. None of the stuck-at faults affecting one of the "*sel[0:2]*" control

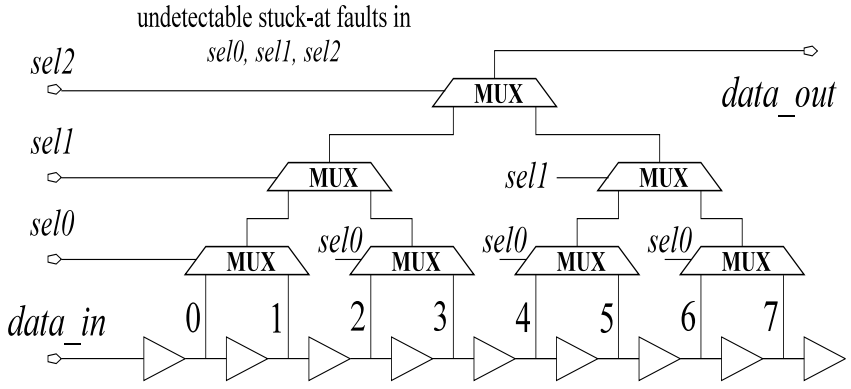


Figure 3.13: Inability to test stuck-at faults at the "*sel[0:2]*" control line signals

signals can be tested due to the inability to control the MUX inputs separately. For example, if a stuck-at fault is present in the "*sel0*" signal, then at least two adjacent inputs of the MUXs should have different values in order to detect the fault. This is not possible since the "*data_in*" signal controls all the inputs of the MUXs.

To overcome this situation, several "OR test points" should be inserted in the delay-line in order to increase the controllability. Test points should be inserted at the 1, 3, 5 and 7 tap inputs. The insertion of these test points will increase the delay values at these taps, but will leave the synchronization mechanism unaffected, due to the close-loop design.

The design of the delay-line including the OR test point is presented in figure 3.14. In functional mode, the **SF1** scannable flip-flop will be loaded

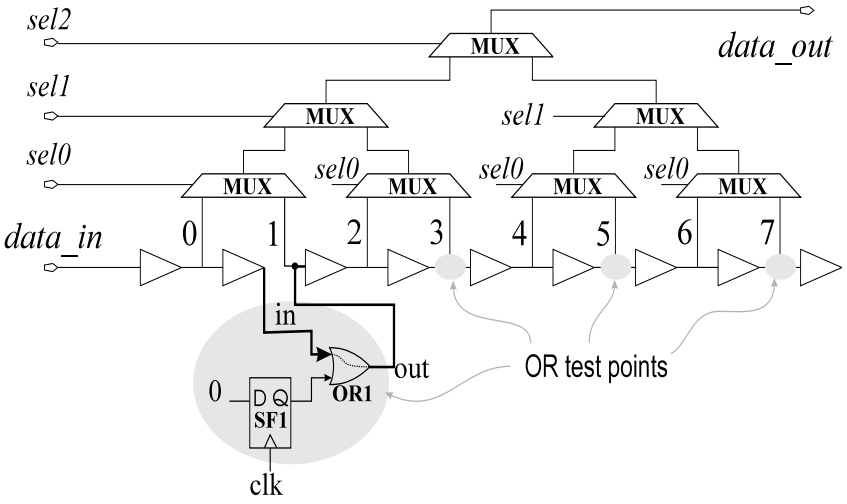


Figure 3.14: Insertion of OR test points necessary to test all the stuck-at faults of a controllable delay-line

with a logic zero. As a result, the signal "*in*" will propagate unchanged to the signal "*out*". In scan-test mode, the insertion of these test points will result in a stuck-at fault coverage of 100% for the controllable delay-line, including the newly inserted OR test points.

Built-In Self Test - BIST

The delay-lines inside the phase comparator of the DLS are essential for a correct operation of the entire DLS. However, the scan-test strategy only verifies that no stuck-at faults are present. It cannot verify the correct delay values in the delay-lines.

The proper functionality of the delay-line synchronizer is based on the correct values of its delays. Therefore, a BIST module has been developed for testing the delay-lines that can be present in the phase comparator. The proposed BIST strategy is presented in figure 3.15.

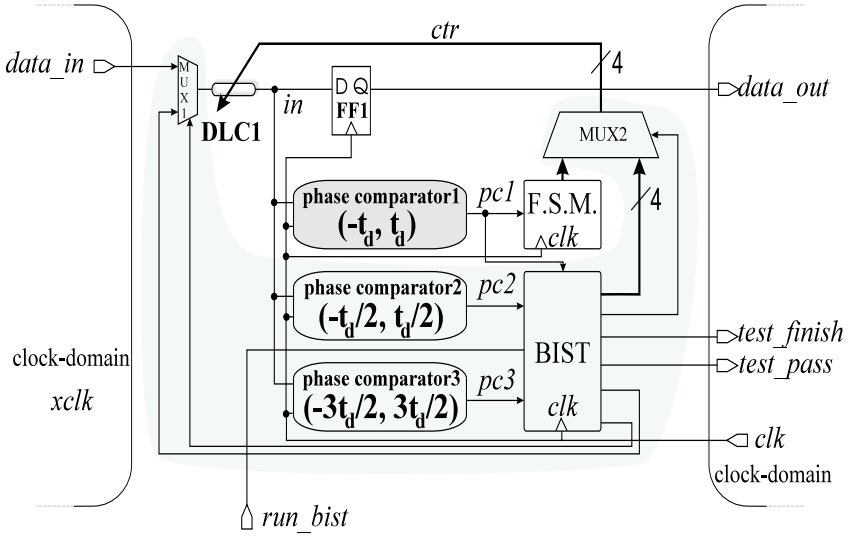


Figure 3.15: BIST strategy for DLS

A BIST module, two MUXs and two new digital phase comparators have been added. Phase comparator2 has a detection range between $-t_d/2$ and $t_d/2$, and phase comparator3 between $-3t_d/2$ and $3t_d/2$. In this way the detection range of the initial phase comparator has been encompassed by the two new phase comparators. The scheme of a generic phase comparator has been presented in figure 3.5.

The **BIST** module takes care of the test procedure and of selecting the proper signals in BIST mode. If the **BIST** is activated, it will provide the "*in*" and "*ctr*" signals via the access multiplexers. The signal "*in*" will toggle every clock cycle, in order to sensitize the phase comparators, while

the 4-bits signal "*ctr*" will increase from time to time in order to modify the delay of the controllable delay-line **DLC1**. By observing the order in which the signals "*pc1*", "*pc2*" and "*pc3*" are changing, the BIST is detecting any delay-fault that might be present in one of the 3 phase comparators. Without any delay-fault present, the "*pc3*" signal is changing (either to 1 or 0) before the "*pc1*" signal, which in turn is changing before the "*pc2*" signal. If this order is not respected, then one of the delay-lines is faulty and the **BIST** will detect it.

The **BIST** is designed to be synchronous with the receiving clock signal "*clk*". Therefore, it is possible to insert a scan chain.

Scan Step

The delay-line synchronizer presented in section 3.1.2 comprises of logic gates, flip-flops and delay-lines. The delay-lines in the data paths can be scan-tested as explained earlier by inserting additional OR test points. In contrast, the delay-lines in the clock paths are hindering the scan-test procedure.

All the flip-flops in the DLS will be replaced with scan flip-flops. However, since some of the flip-flops are synchronized with delayed versions of the "*clk*" clock signal, a data-against-clock test strategy will be employed as explained in reference [5] or section 2.3.3. Moreover, at the interface between the two clock-domains, MCD latches, controlled by the "*TCL*" signal as explained in 2.3.3, should be inserted. For example, just before the "*in*" signal is distributed to the **FF1**, **phase comparator1**, **phase comparator2** and **phase comparator3** (figure 3.15), a latch should be inserted in order to decouple the two clock-domains in scan-test mode.

Remodeling of the DLS has to be employed in order to be able to generate scan-test vectors using any commercial ATPG tool. This remodeling is equivalent with removing the delay-lines in the clock paths and the MCD latches. This removal does not influence the final fault coverage since the faults in the removed parts are 100% functionally equivalent with the ones for which the ATPG has generated scan-test vectors.

As will be shown in later sections, this last assumption will not hold for the TRS and the PS, since their original versions will have faults that are not functionally equivalent with the re-modeled ATPG versions. As a result,

a full verification step has to be carried out to prove that all the stuck-at faults are tested either by BIST or MCD scan.

Fault simulations have been carried out to confirm that the DLS stuck-at fault coverage for the presented BIST and scan-test strategy is 100%. Delay-faults that might be present inside the **phase comparator1** are also detected by the BIST. The penalty for developing a comprehensive SaB test is the large area overhead. The calculated silicon area overhead for the SaB is 193%. In comparison, an MCD scan strategy adds 27% area. Our high area overhead for the SaB test is due to the two additional phase comparators and the BIST structure. However, DLS structures are only used to send few bits across the clock-domains borders. As a result, the area occupied by these modules is very small. If one assumes this area to be 5% of the total chip area, then the previous 193% will be equivalent to 9.65%. It is the same reasoning as for the scan flip-flop area overhead which is more than 30% for one flip-flop but around 7% for a complete chip. Therefore, even if the SaB DfT area overhead might seem high, the real impact on the total design will be small.

3.3.2 Test Strategy for the TRS

The functional description and the schematic of the TRS have been presented in section 3.1.3. There are some similarities between the TRS and the DLS. However, the test strategy for the DLS cannot be directly applied to the TRS because the phase comparator inside the TRS compares two clocks instead of comparing a data and a clock signal as for the DLS. In other words, the TRS cannot be made fully scannable. This in turn decreases the fault coverage obtained by the scan-test, subsequently leaving the burden of detecting the remaining faults to the BIST structure.

In order to better appreciate our proposed SaB test strategy, a plain full-scan test strategy has been directly applied to the TRS. In the case of a plain full scan-test strategy, all the flip-flops are replaced with their scan counterparts and all the clocks are directly controlled during scan mode by inserting MUXs. On top of that, the clock signal "*xclk*" is masked out in the phase comparator. Without this masking, no test vectors can be generated by the ATPG tool because the clock signal "*xclk*" will propagate in the data path. After all these modifications, the TRS has been analyzed by the ATPG tool TetraMAX [4]. The fault coverage results are shown in table 3.2.

Table 3.2: Fault coverage results of the TRS if a plain full-scan test strategy is applied

Uncollapsed Stuck Fault Summary Report		
fault class	code	#faults
Detected	DT	190
Possibly detected	PT	0
Undetectable	UD	2
ATPG untestable	AU	88
Not detected	ND	0
total faults		280
test coverage		68.35%

As can be noticed, the test coverage, is extremely low. Critical signals lines like clock "*xclk*" and modules such as the delay-lines are not tested at all.

The SaB strategy for the TRS will dramatically improve the stuck-at fault coverage and will also be able to test delay-faults within the delay-lines.

The top view of the TRS together with its BIST is shown in figure 3.16. The TRS together with its DfT hardware is presented in figure 3.17. The additional DfT hardware inside the phase comparator is indicated in figure 3.18.

It should be noted that the DfT hardware inside the TRS, as presented in figure 3.17, is divided in two parts. One part is transforming the TRS in a scan-test compatible module (MCD scan FF in figure 3.17), while the other part is helping the BIST to apply the on-chip test vectors (Added DfT in figure 3.17). More about the TRS and its associated DfT hardware can be found in the next paragraph.

TRS and its associated DfT hardware

The main idea for developing the DfT hardware was to test the correctness of all delay-lines in the TRS. To perform this check, the ΦC input signals "*PC_xclk*" and "*PC_clk*" are controlled in the BIST test mode. The **I3** delay-line, in figure 3.17, generates delayed versions of the "*clk*" clock signal which are further routed by the **SN1** switching network to the ΦC. The input signals of the **SN1** have the delay values, with respect to the "*clk*" signal, equal to 0, $t_k/2^-$ and $t_k/2^+$. The notation $t_k/2^-$ denotes that the

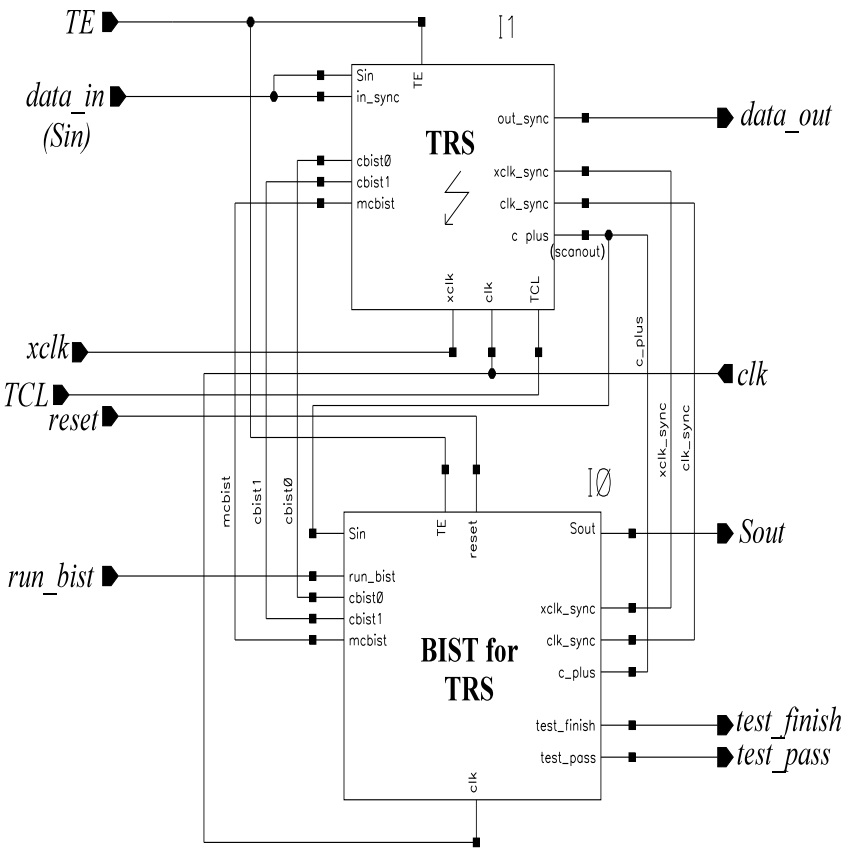


Figure 3.16: TRS together with its associated BIST structure

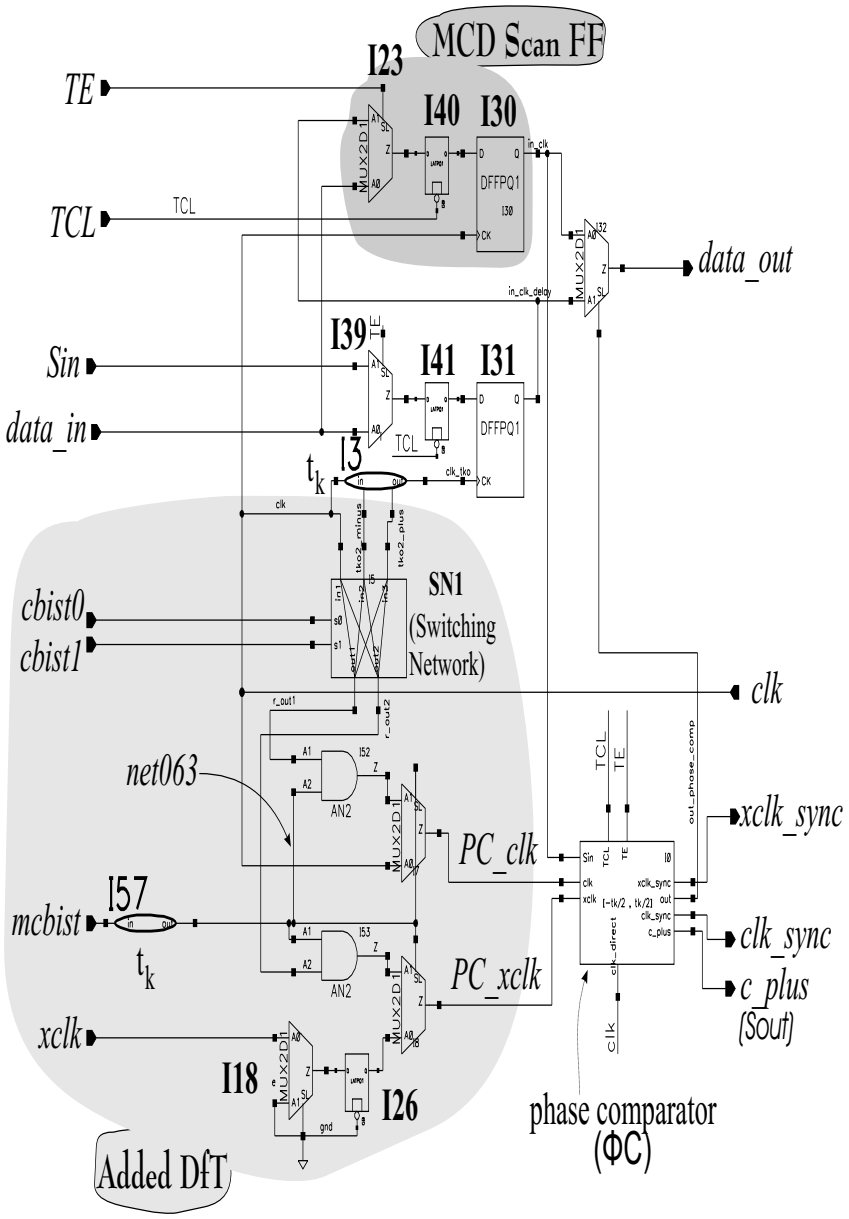


Figure 3.17: TRS and its associated DfT hardware (the lighter shaded part)

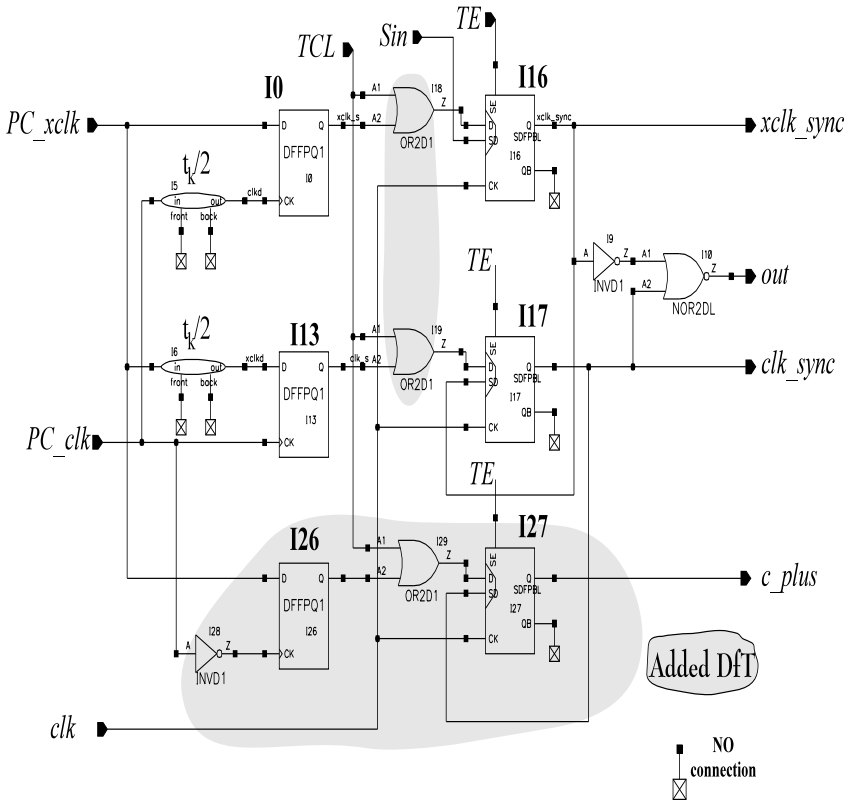


Figure 3.18: ΦC and its associated DfT hardware (the shaded part)

delay is just slightly smaller than $t_k/2$, while $t_k/2^+$ means that the delay is just slightly bigger than $t_k/2$. The four possible combinations, with respect to the delays of the signals "*PC_xclk*" and "*PC_clk*", are applied to the ΦC together with its output signal values are shown in figure 3.19. The signals "*cbist0*" and "*cbist1*" are selecting different delayed versions of the clock signal "*clk*" that will become the signals "*PC_xclk*" and "*PC_clk*". The signal "*mcbist*" is used to switch between the BIST mode and the scan-test.

Any delay-fault that might affect all the delay-lines inside the phase comparator (ΦC) or the delay-line **I3** are detected by the ΦC output values. Furthermore, all the stuck-at faults on the selected paths are also detected.

Another idea used for designing the DfT hardware was to make the TRS together with the BIST completely transparent in the MCD scan mode. The MUX **I23** together with the latch **I40** and the original flip-flop **I30** represent an MCD scan cell as presented in [3] or 2.3.3. The scan chain has been inserted in the TRS and the ΦC using a data-against-clock strategy, as presented in [5] or 2.3.3. As already mentioned, not everything will be tested by the MCD scan-test, but the TRS will not hinder this test strategy.

BIST for TRS

The **BIST** has been implemented as a fully synchronous finite-state machine and its goal is to send control signals to the TRS and observe the outputs. By observing the outputs it can capture all the remaining stuck-at faults not detected by the MCD scan strategy and the delay-faults present in the delay-lines in the TRS design.

The BIST itself is tested by applying the MCD scan-test strategy. The scan chain inside the BIST has been concatenated with the scan chain defined inside the TRS. Therefore, the combination TRS+BIST has only one scan chain. In figure 3.16, the scan-in signal is "*Sin*", the scan-out is "*Sout*" and the scan-enable signal is "*TE*".

DfT hardware for the phase comparator

While comparing the ΦC of figure 3.6 with the one in figure 3.18, one will notice two additional flip-flops (**I26** and **I27**) added in the last figure. They have been added in order to distinguish, in BIST test mode, between the top

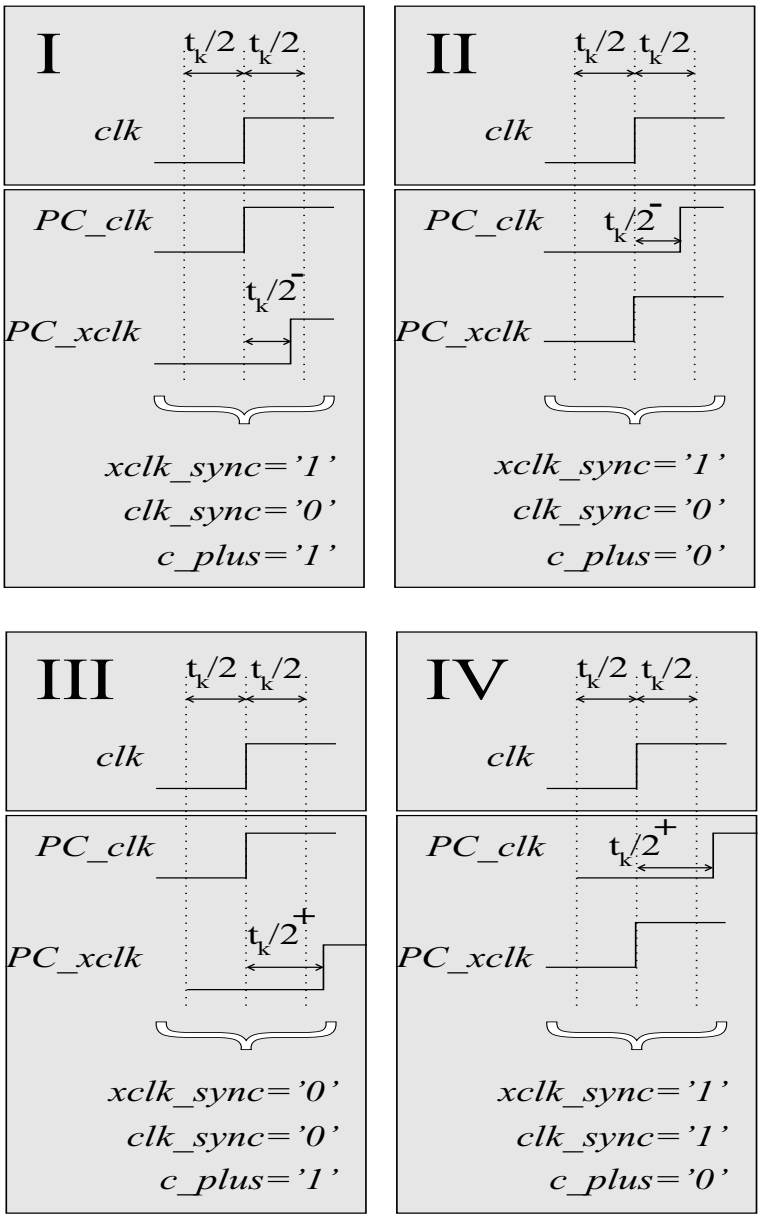


Figure 3.19: "*PC_xclk*" and "*PC_clk*" signals in the BIST test mode together with the expected outputs "*xclk_sync*", "*clk_sync*" and "*c_plus*" of the phase comparator (ΦC)

two cases presented in figure 3.19. Without these two flip-flops, generating the signal "*c_plus*", there would have been no way of differentiating between case I and case II.

The flip-flops **I16**, **I17** and **I27**, inside the ΦC are scan flip-flops. In order to avoid unknown values propagating during scan-test from the **I0**, **I13** and **I26** flip-flops, the "*D*" inputs of the **I16**, **I17** and **I27** flip-flops have been OR-ed with the "*TCL*" signal. This operation will emulate a '1' signal in scan-mode operation. During functional or BIST modes of operation, the input of the **I16**, **I17** and **I27** flip-flops will be unaltered since in these modes the "*TCL*" signal will always be zero.

Remodeled TRS for ATPG

The TRS version presented in figure 3.17 cannot be submitted to an ATPG tool like TetraMAX [4], because:

- The "*clk*" and "*xclk*" clocks are passing through several logic gates and the controlling values for these gates cannot be set to certain values, as an ATPG tool would require, without the loss of fault coverage.
- The flip-flop **I31** in figure 3.17 and the **I0** and **I26** flip-flops in figure 3.18 are clocked by a delayed version of the clock signal "*clk*". This is not allowed for a full-scan ATPG tool.
- The flip-flops **I0** and **I26** in figure 3.18 are capturing the "*PC_xclk*" signal which is derived from the clock signal "*xclk*". ATPG tools cannot parse such designs.
- The signal "*TCL*", together with the latches **I40** and **I41** in figure 3.17 that it controls, cannot be parsed by an ATPG tool.

In order to be able to carry out an ATPG run on the TRS, one has to remodel it. The `remodel-for-ATPG` version of the TRS in figure 3.17 is shown in figure 3.20.

All the flip-flops that can be included in a synchronous MCD scan-chain have been considered. Out of a total of 8 flip-flops in the TRS, including the ones in the ΦC, 5 have been included in the scan chain. All the signals in figure 3.20 are synchronous with the clock "*clk*".

The scheme in figure 3.20 can now be used as input for an ATPG tool.

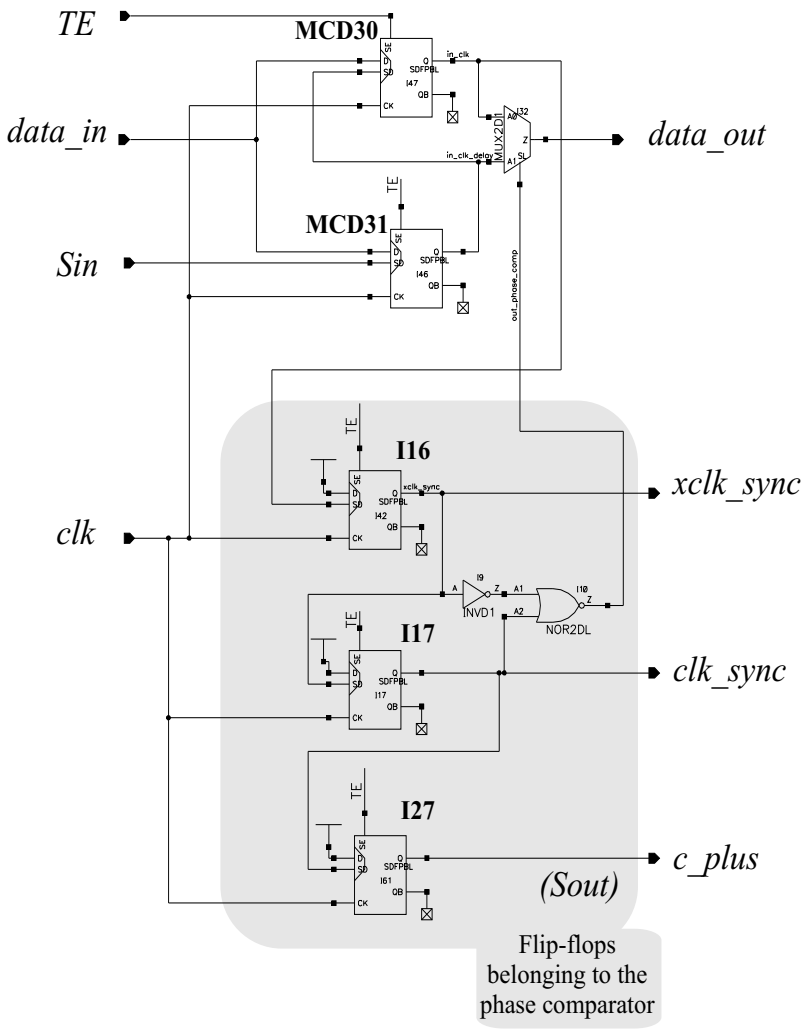


Figure 3.20: The TRS **remodeled-for-ATPG**, including the phase comparator

Fault simulations results for the TRS and BIST

A scheme comprised of the `remodeled-for-ATPG` TRS together with its BIST structure has been submitted to the TetraMAX [4] ATPG tool. The remodeled scheme submitted to the ATPG tool was fully synchronous with the clock signal "*clk*". The results of the ATPG run are given in table 3.3.

Table 3.3: Fault-coverage results for the TRS (remodeled for ATPG) and BIST

Uncollapsed Stuck Fault Summary Report		
fault class	code	#faults
Detected	DT	696
Possibly detected	PT	0
Undetectable	UD	8
ATPG untestable	AU	0
Not detected	ND	0
total faults		704
test coverage		100.00%

The tool reported 704 faults out of which 8 were undetectable. Six of the undetectable faults are located within the phase comparator because the flip-flop inputs are tied to a logic one. Two undetectable faults are reported because there are two scan flip-flops in a chain which prevent the detection of any stuck-at fault on the "*TE*" signal. Therefore, the fault coverage of 100% reported by TetraMAX is correct, since the 8 stuck-at faults do not influence the behavior of the remodeled circuit.

The test vectors generated by TetraMAX have been used to verify that they can be applied to the original design presented in figure 3.16. One should remember that TetraMAX generates test vectors for a fully synchronous remodeled design, while the real design of figure 3.16 has two clock signals ("*xclk*" and "*clk*") and many delay-lines that are creating more small separate clock domains.

After generation of the "*TCL*" signal as presented in 2.3.3, the test vectors generated for the SCD remodeled design ran flawlessly on the MCD design of figure 3.16.

After this step, all the stuck-at faults that might be present inside the TRS have been simulated, either by TetraMAX test vectors or by the BIST

structure. The results are shown in table 3.4. The stuck-at faults in the BIST structure have not been included in table 3.4, since they are reported as tested by TetraMAX and there has been no remodeling for the BIST structure.

Table 3.4: Stuck-at fault simulations for the original TRS shown in figure 3.16

Number of stuck-at faults	considered inside the TRS	108
	Detected by TetraMAX test vectors only	43
	Detected by BIST only	44
	Detected by BIST and TetraMAX test vectors	9
Total number of detected faults		**96**
Fault Coverage (only for the TRS)		**98.14%**

One may notice the presence of 12 (108-96) undetectable stuck-at faults but the total fault coverage is 98.14%. This is the case because on one hand the custom written script for inserting the faults has been too pessimistic by inserting four extra stuck-at faults in two unconnected signals, and on the other hand the remaining eight faults do not influence the correct behavior or they are potentially detected. Among the eight faults we find three "grounded" signals being s-a-0, one "*TCL*" signal being s-a-0, one "*TE*" being signal s-a-0 and s-a-1, and two "*net063*" inputs to the AND gates which are s-a-1 (see figure 3.17).

The three "grounded" signals can be found at the **I18** MUX and **I26** latch at the bottom of figure 3.17. The signals should be connected to '0' and therefore the s-a-0 faults should not have been considered. The "*TCL*" signal s-a-0 is potentially detected. Since in reality the skew between the "*xclk*" and "*clk*" clocks is unknown, the scan-test might work with the latches (**I40 I41**) always being transparent, which is equivalent with signal "*TCL*" being s-a-0. It should also be noted that this fault does not influence the functional behavior since the signal "*TCL*" is logic zero in this mode. The "*TE*" signal for s-a-0 and s-a-1 has also been reported by TetraMAX. As explained before, there are two scan flip-flops in a chain which prevent the detection of any stuck-at fault on the signal "*TE*" controlling the scan operation for the receiving scan flip-flop. The last two faults in the "*net063*" signal cannot be detected due to redundancy in the circuit. This is why the fault coverage is not 100%. However, they do not influence the behavior of the circuit in functional mode.

The fault coverage result presented in table 3.4 looks very appealing. Our

SaB test strategy covers most of the stuck-at faults. The developed BIST module detects also the path delay-faults of the designed delay-lines. This has been verified by VHDL simulations.

An extension of the TRS for two bits is presented in figure 3.21. The additional hardware required to transform the 1-bit TRS into a 2-bits TRS has been indicated on shaded area in figure 3.21. The BIST hardware has also been modified in order to accommodate more than 1 bit. The number of undetected faults remained the same since no fundamental change have been performed. However, since more logic gates have been added, the fault coverage number has been increased.

The test-area overhead data is presented in table 3.5. The first column

Table 3.5: Test hardware area overhead for a TRS

Number of bits	**AREA OVERHEAD**			
	SCD scan	MCD scan	SaB test	Total chip area(SaB)
1	23.32%	34.72%	429.82%	21.49%
8	24.32%	61.53%	239.34%	11.97%
16	24.59%	68.99%	186.14%	9.31%
32	24.78%	73.91%	149.57%	7.48%
64	24.88%	76.79%	131.70%	6.58%

represents the number of bits of the considered TRS. The second column shows the area overhead values if the TRS would be tested by a single clock domain (SCD) scan technique only. In some cases this test method cannot be applied to MCD designs. It has been presented here for comparison reasons only. The third column shows the test-area overhead for MCD scan. The fourth column lists the area overhead of our combined test strategy, SaB. The last column shows the area overhead if the TRSs modules would occupy 5% of a total design and it would be tested with our SaB technique.

The area overhead for the SCD scan-only strategy might seem very large compared to the general approximate value of 7%. The increased area overhead is resulting from the large number of flip-flops present in the TRS. The same explanation holds for the MCD scan. Our presented SaB strategy might have a higher area overhead but it definitely has a higher fault coverage of stuck-at faults and it can detect the delay-faults for all the delay-lines in the design.

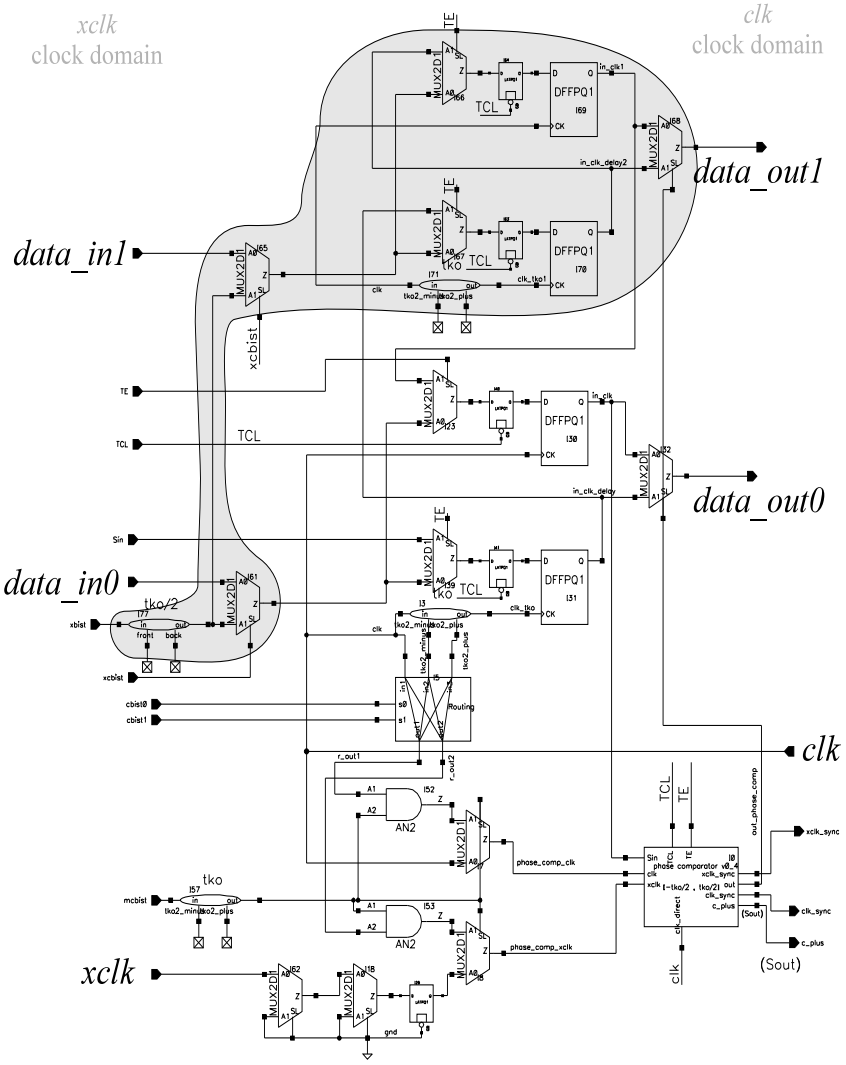


Figure 3.21: 2-bits extension of the TRS. The additional hardware has been shaded

The last column of table 3.5 is very interesting since it shows that for 64-bits two-register synchronizers, occupying 5% of a chip design, the additional area overhead for the SaB test strategy at the chip level is less than 7%.

These two-register synchronizers could be utilized for large deep sub-micron chips where one clock cannot be safely distributed over the entire surface; in addition, the number of bits sent from one clock domain to another can be even bigger than 64.

3.4 Test Strategies for Periodic Synchronizers

3.4.1 Test Architecture of the PS

The functional description of the PS has been presented in section 3.1.5. As briefly mentioned, the PS is a TRS augmented with a **predictor** module. Test strategies for the PS will consist, as for the other synchronizers, of an MCD scan and a BIST. Since the PS is quite similar to the TRS, the BIST strategy of the TRS will be adapted to the PS. It is however not possible to use the same BIST hardware because the phase comparator ΦC inside the TRS compares two identical frequencies, while the phase comparator ΦC inside the PS compares two totally different frequencies. Moreover, the **predictor** module which is part of the PS should also be tested.

The PS is even harder to be tested by the scan-test strategy, due to the **predictor** module. The fault-coverage results of a scan-test will be far worse than the numbers presented in table 3.2 for the TRS.

As previously mentioned, a SaB test strategy will be applied again. The scan-test strategy will smoothen the integration of the surrounding cores in a general chip scan-test. It will also test some stuck-at faults inside the PS, and completely test all the stuck-at faults for the BIST part.

All hierarchical levels of the PS are presented in figures 3.22, 3.23, 3.24 and 3.25.

The PS contains several key components that must be definitely checked for fabrication faults. The first of them is the predictor module, shown in figure 3.25. It is comprised of an analogue phase comparator ($a\Phi C$), a controllable delay-line (**DLC1**) and a fixed delay-line (**DL2**) having the value of the "clk" clock period being T_{cy}. If the predictor module functions

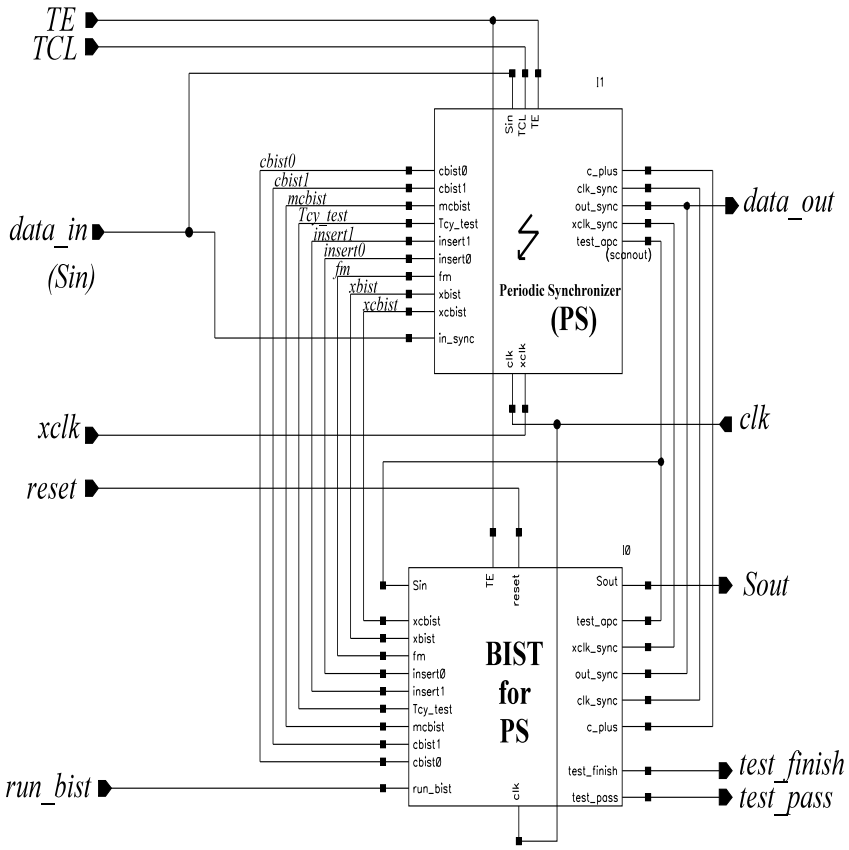


Figure 3.22: PS together with its associated BIST structure

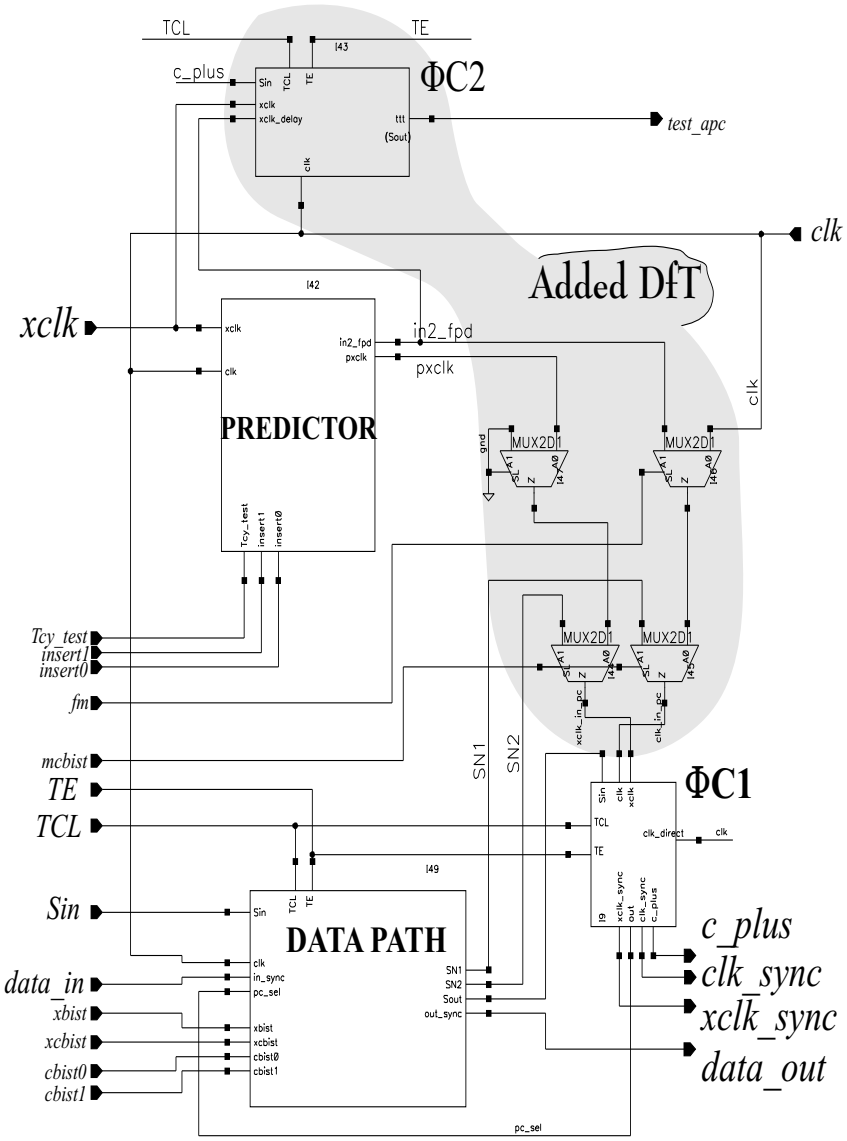


Figure 3.23: PS and its associated DfT hardware

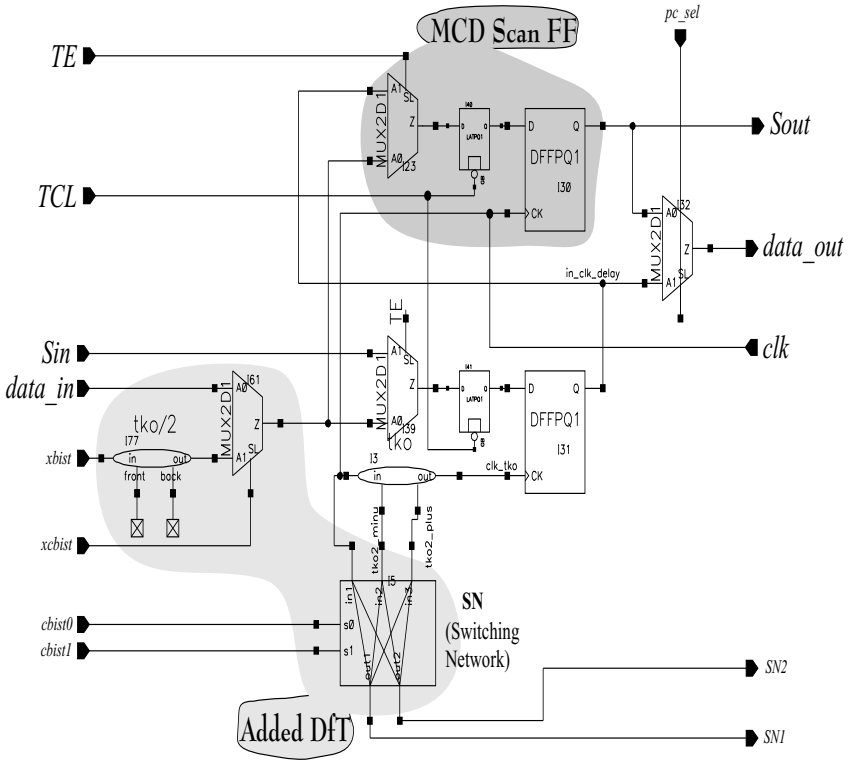


Figure 3.24: DATA PATH of the PS and its associated DfT hardware (the lighter shaded part)

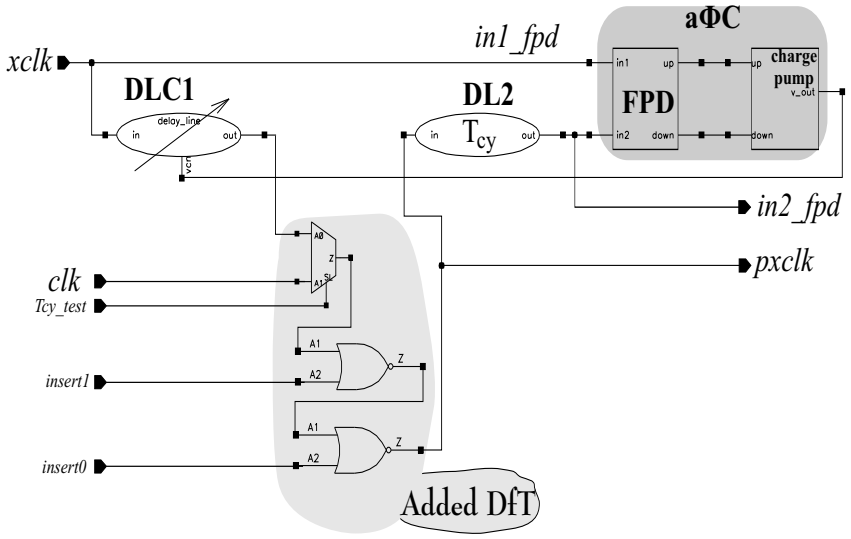


Figure 3.25: Predictor module and its DfT hardware (the lighter shaded part)

properly, the "*in1_fpd*" and "*in2_fpd*" signals are in phase and the **DL2** delays the signal "*pxclk*" by T_{cy}.

To check whether the "*in1_fpd*" and "*in2_fpd*" signals are in phase, an auxiliary digital phase comparator $\Phi C2$ has been added, which is similar to $\Phi C1$ (see figure 3.23). If the two inputs to the analogue phase comparator ($a\Phi C$) are not in phase then the newly introduced phase comparator $\Phi C2$ will detect this. To verify that delay-line **DL2** is delaying by T_{cy} (figure 3.25), a multiplexer has been inserted in front of it, which can apply the "*clk*" clock signal to the **DL2** delay-line. If there is no delay-fault present, the "*pxclk*" and "*in2_fpd*" signals of figure 3.25, will be in phase. The functional phase comparator $\Phi C1$ in figure 3.23 is utilized to perform this check by properly selecting the MUXs in front of its two inputs "*clk*" and "*xclk*".

Other important modules to be checked are, similar to the TRS, the delay-lines present in the data path and in the phase comparator $\Phi C1$. The strategy employed is the same as for the TRS and it will not be presented again here. The $\Phi C1$ phase comparator is checked by the same test sequence as presented for the TRS in subsection "TRS and its associated DfT hardware" which is part of section 3.3.2.

All the necessary steps to be carried out in order to test the above men-

tioned parts are performed by the BIST structure which is essentially a FSM, controlling the test sequence and observing some outputs of the PS. The BIST is designed to be as much independent of the PS as possible. However, since the PS design parameters might change for a new utilization, like the lock time, the clock-domains frequencies, etc., some parts of the BIST structure should also be adjusted.

The BIST is synchronous with the receiving clock "*clk*". Hence, it is 100% scan-test compatible.

3.4.2 Fault-simulation results for the PS and BIST

Most of the flip-flops in the PS have been replaced by their scan counterparts. Out of 14 flip-flops inside the PS, nine were replaced by scan versions. The scan chain has been inserted in the PS using again a data against clock strategy, as presented in 2.3.3. Following the scan insertion, the PS has been remodeled for the ATPG tool in order to be able to generate test vectors. The BIST structure, which is also equipped with scan, has been attached to the PS module and the two scan chains have been concatenated as can be seen in figure 3.22. The resulting scheme has been submitted to the ATPG tool TetraMAX [4]. The results of the ATPG are presented in table 3.6.

Table 3.6: Fault-coverage results for the PS (remodeled for ATPG) and BIST

Uncollapsed Stuck Fault Summary Report		
fault class	code	#faults
Detected	DT	1068
Possibly detected	PT	0
Undetectable	UD	12
ATPG untestable	AU	0
Not detected	ND	0
total faults		1080
test coverage		100.00%

All the "Undetectable" stuck-at faults are of no concern. They either have been deliberately introduced in order not to contaminate some flip-flops with unknown values or are false reports due to two, or more, scan flip-flops connected in a chain. All relevant faults will be detected in BIST

mode.

The generated test vectors by TetraMAX have been applied to the original design presented in figure 3.22, after the proper generation of the signal "*TCL*" as presented in 2.3.3. Three hundred and four additional stuck-at faults have been considered in the PS module that have not been taken into account during the ATPG vector generation. The MCD scan-test strategy has been applied, as described in 2.3.3, and it could detect only 77+106 of them. When the MCD scan-test has finished, the BIST has been activated and it detected 113 additional faults. The fault coverage results of these two testing steps are presented in table 3.7.

Table 3.7: Fault-simulation results for the original PS presented in figure 3.22

Number of stuck-at faults	considered inside the PS	304
	Detected by TetraMAX test vectors only	77
	Detected by BIST only	113
	Detected by BIST and TetraMAX test vectors	106
Total number of detected faults		**296**
Fault Coverage (only for the PS)		**100%**

One may notice that there are 8 undetectable stuck-at faults, but the total fault coverage is still 100%. These faults do not influence the behavior of the PS and should be ignored. They are either ground ports stuck-at zero, "*TCL*" stuck-at zero or as a result of two scan flip-flops in a chain. Since the BIST structure is 100% tested by the ATPG test vectors, the stuck-at fault coverage for the complete design consisting of the PS and BIST is 100%. Our DfT structure covers all the stuck-at faults and any delay-fault present inside the delay-lines or the predictor module.

Besides fault coverage, silicon area overhead is another important parameter when assessing a DfT strategy. The test area-overhead numbers are presented in table 3.8.

The first column represents the number of bits of the considered PS. The second column shows the area overhead values if the PS would be tested by SCD scan only. It should be mentioned again that this method of testing cannot be applied to some MCD designs. It has been included here just for comparison reasons. The third column provides the test-area overhead for MCD scan, and the fourth column represents the area overhead of our combined SaB test strategy. The last column shows the area overhead if

Table 3.8: Test-area overhead for a PS

Number of bits	AREA OVERHEAD			
	SCD scan	MCD scan	SaB test	Total chip area(SaB)
1	23.32%	34.72%	623.28%	31.16%
8	24.32%	61.53%	322.94%	16.15%
16	24.59%	68.99%	232.55%	11.62%
32	24.78%	73.91%	177.06%	8.85%
64	24.88%	76.79%	144.18%	7.21%

the original PSs would occupy 5% of a total design.

The area overhead might seem quite high. However, this is acceptable, since the fault coverage has been increased from less than 70% to 100%. The delay-lines, which are critical components for the proper operation of the PS, have been tested for delay-faults by the BIST part.

The analogue **predictor** module area has not been considered while calculating the area overhead. Therefore, the numbers presented above will be smaller in reality. The predictor module can be more than 5 times bigger that the BIST and PS parts together. Hence, it only makes sense to use the PS for wide data busses.

3.5 Conclusions

The key part of any advanced synchronizer is its phase comparator module. To be as effective as possible, any test strategy should be to a large extend technology independent. By modifying one delay value and changing its position from the clock line to the data line, the present phase comparator in most of the synchronizers, has become technology independent. From the functional point of view, this modification did not seem important but from the test point of view it was a great step forward. It allowed us to develop BIST modules independent of the technology or of the running clock of the receiving domain. Only the periodic synchronizer (PS) module remained technology dependent due to its predictor module which is an analogue design. For the PS, the BIST must be adapted whenever a new technology is targeted.

Advanced synchronizers architectures, as the one presented at the beginning of this chapter, will slowly find their way in the future Ultra Large Scale Integration (ULSI) designs. The trend is mainly driven by the inability to distribute a single clock over the entire system. Testing these new modules is a must for a smooth operation of the entire system. The traditional scan-test strategy will no longer be able to provide even a satisfactory stuck-at fault coverage.

The proposed Scan-and-BIST (SaB) test strategy is composed of two testing steps. During the scan step, the synchronizers are transformed into scan-transparent devices easing the way towards a complete scan-test of the entire system. However, many stuck-at faults and delay-faults will remain untested during the scan-test step. The second step, based on Built-In Self-Test (BIST), will test the relevant delay-faults and the remaining stuck-at faults from the previous scan step. The VHDL designs of the BIST modules are not unique and they can still be improved with respect to the area overhead. The BIST itself is tested during the first scan step. Using this double test strategy, the synchronizers are fully tested. Not only the stuck-at fault coverage numbers are increasing but also the delay-faults that might be present in the designed delay-lines are tested. The penalty for this SaB test strategy is the area overhead. This might seem quite high for the stand-alone synchronizer (between 100% and 200%). However, because synchronizers only occupy a small area of the entire system, the overall area overhead does not exceed 10%. This can be seen as a small price to pay considering the gain of fault coverage and the vital importance of these small modules.

The following results have been obtained:

- the designs of the synchronizers were modified to be as much as possible technology independent
- the synchronizers were remodeled for ATPG in order to be able to generate scan-test vectors
- BIST structures were designed for the synchronizer modules which are able to increase the stuck-at fault coverage and to detect relevant delay-faults in the delay-lines.

Bibliography

[1] William J. Dally and John W. Poulton. *Digital Systems Engineering*. ISBN 0-521-59292-5 (hb). Cambridge University Press, 1998. 3.1.1

[2] LogicVision. *Embedded Test*. LogicVision, Inc., 101 Metro Drive, Third Floor, San Jose, CA 95110. `http://www.logicvision.com/`. 3.2

[3] Josef Schmid and Joachim Knblein. Advanced Synchronous Scan Test Methodology for Multi Clock Domain ASICs. *IEEE VLSI Test Symposium (VTS)*, pages 106–113, April 1999. 3.2, 3.2, 3.3.2

[4] TetraMAX ATPG. *High-Performance Automatic Test Pattern Generator Methodology Backgrounder*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043, May 1999. `http://www.synopsys.com/products/test/tetramax_wp.html`. 3.3.2, 3.3.2, 3.3.2, 3.4.2

[5] Bart Vermeulen, Maurice Lousberg, and Paul Merkus. Scan test techniques for multi synchronous digital circuits. *Intenational Test Synthesis Workshop (ITSW)*, March 1998. 3.3.1, 3.3.2

Chapter 4

Scan Testing of Asynchronous - Synchronous Interfaces

An increasing number of asynchronous cores will be used in future SoC designs due to their low-power, low electromagnetic emissions and low design effort when changing the technology. Synchronous designs will still be used because of the huge number of IP cores in use nowadays. As a result, asynchronous-synchronous interfaces are likely to be used extensively in the future. However, no new design style will be accepted for mass production unless, among other things, there is a good testing strategy associated with it. Many challenging designs are put on hold as a result of the inability to test them.

Nowadays, the most important test strategy for digital ICs is the scan technique. The technique is simple, easy to implement and gives very good results in practice, even if the model used does not fully match the real faults. In the meantime, some of the asynchronous designs have become scan-testable [8]. Synchronous design is, by default, scan-friendly. A natural conclusion would be to also use the scan-test technique to test the asynchronous-synchronous interfaces.

This chapter will present the scan-test strategies for the asynchronous-synchronous interfaces (ASI) mentioned in section 2.2.4. These type of interfaces will be further referred to as asynchronous-synchronous synchro-

nizers, or in short, just synchronizers when there is no possibility of confusion.

There are different types of synchronizers based on the data-transfer direction between the cores. Sending data from a synchronous core to an asynchronous one will imply a different synchronizer design and DfT hardware than for the reverse communication of data between an asynchronous core and a synchronous one.

The first section (4.1) of this chapter will describe the *Asynchronous Producer - Synchronous Consumer* [5] synchronizer. The functionality of this synchronizer will be discussed in subsection 4.1.1. Following this functional description, a scan-test strategy is presented in subsection 4.1.2. The results of the fault simulations are presented in subsection 4.1.3.

The second section (4.2) of the chapter will address the *Synchronous Producer - Asynchronous Consumer* [5] synchronizer. Following the same path as before, the functionality of this synchronizer will be presented first in subsection 4.2.1. Then, the scan-test strategy for this type of synchronizer is presented in subsection 4.2.2, followed by its fault simulations in section 4.2.3.

In the third section (4.3), the fault-coverages of the ASIs are increased even more by adding extra DfT hardware which is fully tested by scan-test.

Finally, the last section (4.4) will present the conclusions with respect to the presented test strategies.

4.1 Asynchronous Producer Synchronous Consumer

In figure 4.1, a possible interface between an asynchronous core and a synchronous one is presented. In reference [5], this type of interface is called *"asynchronous producer and synchronous consumer"*.

Different from the interface, which will be later referred to as synchronizer, in reference [5], the clock-distribution network has been included in the delay path, indicated by the shaded area in figure 4.1, similar to what has been presented in [6].

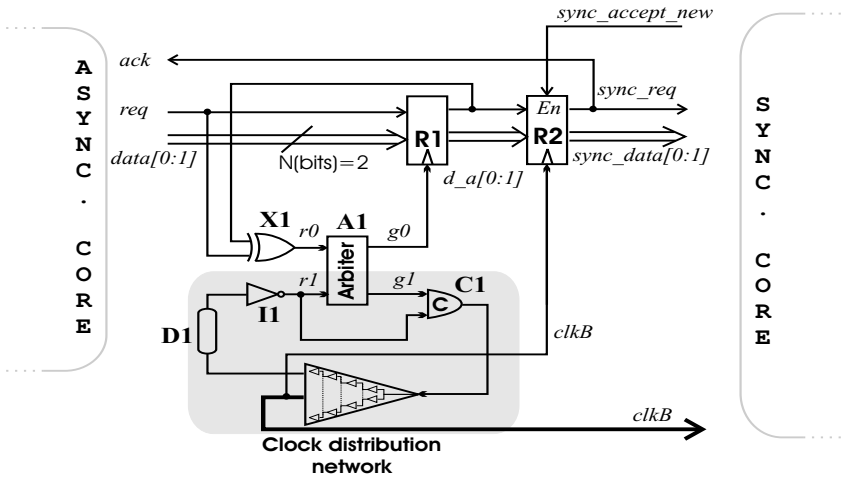


Figure 4.1: Asynchronous producer - synchronous consumer interface [5]

4.1.1 Description of the Functionality

An exhaustive functional description of this circuit is given in [5]. However for the sake of completeness, its functionality will be briefly explained in this section.

The sending of data from the asynchronous core to the synchronous one is signaled via the "*req*" signal using a two-phase handshake protocol [2, pages 489-490]. The data to be sent to the synchronous domain is bundled [2, page 413] with the "*req*" signal.

The "*clkB*" clock signal shown in figure 4.1 is generated on-chip by a stoppable ring oscillator comprised of the clock-distribution network, the delay line **D1**, the inverter **I1**, the arbiter **A1** and the C-element **C1**.

An arbiter [4], also known as MUTEX (mutual exclusion element) is an asynchronous component designed to distinguish the order in which two or more events appear. Each request input of an arbiter has an associated grant signal. If the request inputs are asserted nearly at the same time, only one output grant signal corresponding to the first asserted request input is asserted. A typical two inputs arbiter implemented in CMOS technology is presented in figure 4.2.

The signals "*r0*" and "*r1*" are the request ones while signals "*g0*" and "*g1*" represent their associated grant signals. The four MOS transistors are used

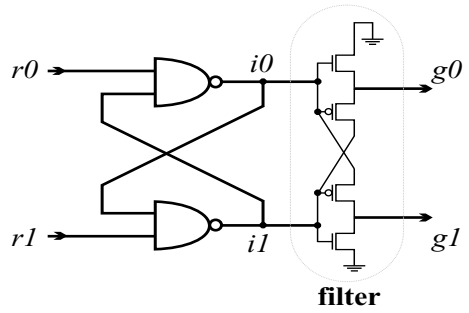
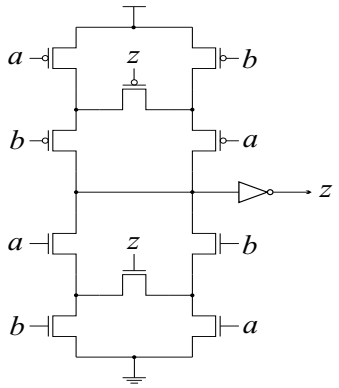


Figure 4.2: CMOS implementation of a two-inputs arbiter

to filter out the metastable (section 2.1) state that might be present in the signals "*i0*" or "*i1*".

A C-element, also known as C-Muller element, can be regarded as a combination of an AND and an OR gates with memory. The output of a C-element will change only if all inputs have the same value; otherwise, the C-element will keep the previous value, denoted with z^- in figure 4.3. A CMOS implementation of a C-element, together with its functional table are presented in figure 4.3.



a	b	z^+
0	0	0
0	1	z^-
1	0	z^-
1	1	1

Figure 4.3: CMOS implementation and the associated functional table of a C-element

The arbiter and the C-element can be seen as non-conventional state holding elements having specific functionality. Therefore, the classical scan-test methodology (2.3.1) is hampered, since the arbiter and the C-element are neither combinational logic circuits nor conventional state holdings elements such as flip-flops or latches.

The **A1** arbiter is the key element of the synchronization strategy. If the "*r0*" signal, which is related to the "*req*" signal, arrives at the same time with the "*r1*" signal, which is derived from the clock "*clkB*" signal, the arbiter will decide which signal arrived first and asserts "*g0*" or "*g1*". Presuming that:

- the "*g0*" signal is asserted during the arbitration operation, the clock will be stopped for a short period of time, allowing the safe latching of the asynchronous data (data[0:1]) in the **R1** register. Together with the data, the "*req*" signal is also latched in the same register. This will in turn change the value of the "*r0*" signal to zero. Following this operation, the clock will be restarted again by letting the logic one value of the "*r1*" signal propagate through the arbiter, the C-element, the clock distribution network and so on.

- the "*g1*" signal is asserted, the clock will continue to run. The asynchronous part will have to wait until the "*r1*" signal will be set to zero. When this happens, the "*r0*" signal will be honored by the arbiter and "*g0*" will become a logic one. This will safely latch the data and the "*req*" signal in the **R1** register, which in turn will force the "*r0*" signal back to zero.

Functional simulations of the circuit shown in figure 4.1 have been carried out and are presented in figure 4.4 to illustrate its correct behavior.

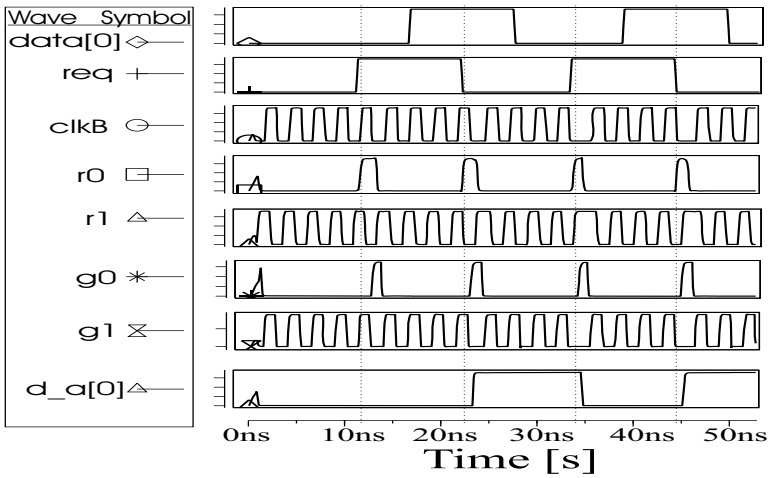


Figure 4.4: Simulated waveforms for an asynchronous producer - synchronous consumer interface

One can notice a delayed clock ("*clkB*") around the 34ns and 44ns time steps. This is a result of the "*req*" signal coming at the same time with the self-generated clock. The arbiter resolved the conflict by granting the request to the "*r0*" signal and holding back the "*r1*" signal. The opposite case where the request signal is put on hold with respect to the clock signal can be observed in the same figure around the 11ns and 22ns points on the time scale.

4.1.2 Scan-Test Strategies

The main idea behind the scan-test strategy for the ASIs is to observe that they are basically behaving as buffers.

However, testing the interface module presented in figure 4.1 is not a straight-forward task if a full-scan technique is desired for compatibility with the synchronous digital design style. One can recognize hard-to-test asynchronous modules like the arbiter and the C-element. Moreover, the "*req*" signal is propagating both in the control and data path. An immediate solution would be to split the "*req*" signal in test mode. However, this initial solution would result in a decreased fault-coverage. Our solution is to add 100% testable DfT hardware, followed by a remodeling for the ATPG tool.

The main assumption for our presented test strategy is that the asynchronous part can be made scannable. If the asynchronous part is not scannable, then the solution can still be applied but it will not be so efficient in terms of fault-coverage. Recently, handshake asynchronous circuits have been tested using the synchronous scan-test technique [7, 8]. Therefore, our main assumption can be fulfilled.

Figure 4.5 presents the test-scannable asynchronous-synchronous interface. In this figure one can recognize:

- the initial synchronizer schematic as presented in figure 4.1
- the relevant asynchronous part made scannable represented by the **CLC** block and the **R3** register
- an inverter in front of an extra **MUX** controlled by the "*TE*" signal
- another inverter and **MUX** controlled by the "*TM*" signal

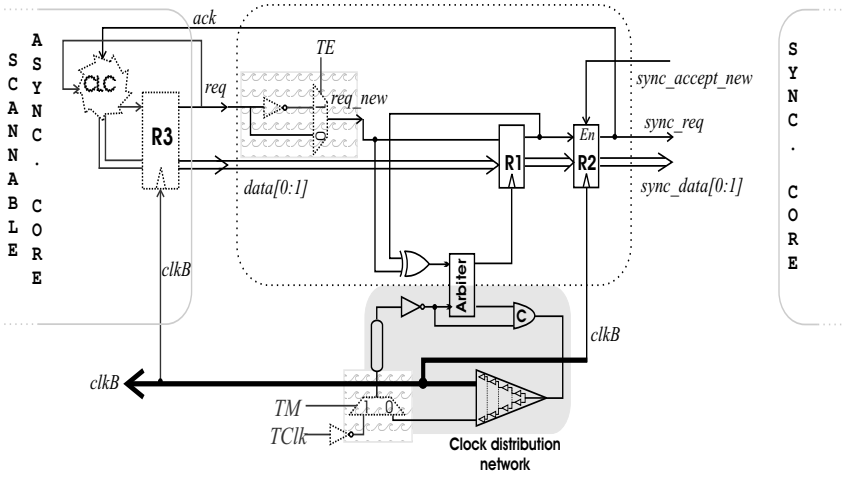


Figure 4.5: The test structures included in the asynchronous producer - synchronous consumer interface

The "*TE*" signal represents the test-enable signal and is logic one during the scan operation. During the capture operation, the "*TE*" signal value is logic zero. The "*TM*" signal is also known as the *scan-test-mode* signal and is always high for the whole duration of the scan-test.

Not all the registers in figure 4.5 will be replaced by their scan counterparts. For example, the **R1** register cannot be replaced by a scan register since its clock is generated by the **arbiter** module and therefore it cannot be directly controlled by the ATPG tool without loss of fault-coverage. Therefore, only the **R3** and **R2** registers will be replaced by scan registers. With this observation in mind it can be inferred that during the scan-test operation, the "*req*" signal is continuously changing depending on the data that is scanned in the **R3** register. Therefore, the value of the "*req*" signal stored in the **R1** register, at the end of the scan operation, is the negation of the last value of the "*req*" signal scanned in the **R3** register.

After the last bit has been scanned-in, the "*TE*" signal will go low which will force a transition in the "*req_new*" signal. This transition will latch the "*req*" signal together with the data signals ("*data[0:1]*") in the **R1** register. The transition is compulsory since the "*data*" bus signals must be latched in the **R1** register. Without it, the "*data*" signals could have different values than the intended scan data.

With the above explanation in mind, one can infer that from the ATPG

point of view, all the circuitry between the **R3** and the **R2** registers can be replaced by an array of buffers, or, even better, just wires. Hence, the ATPG scheme to be used in order to generate test vectors for this circuit is presented in figure 4.6.

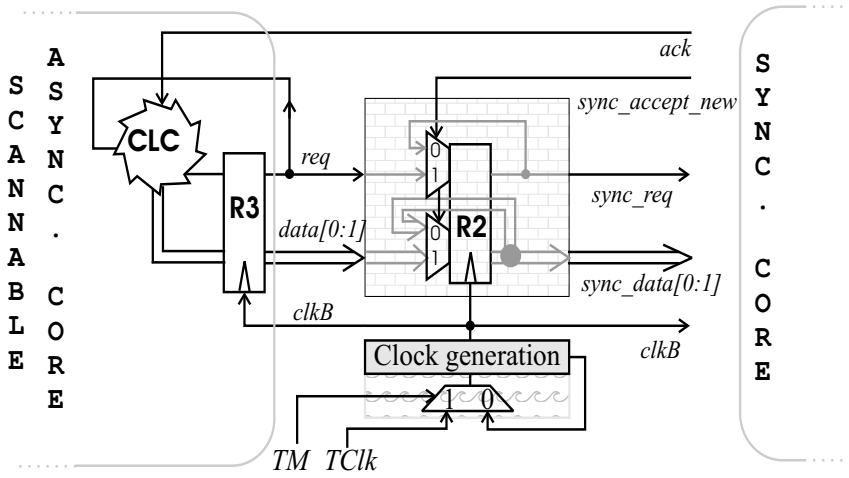


Figure 4.6: "Modeled-for-ATPG" scheme of the asynchronous producer - synchronous consumer interface

The bricks shading style shows the remodeling for ATPG of the initial ASI interface presented in figure 4.1. The waves shading style shows the additional hardware used to control the test clock.

The scheme in figure 4.6 behaves, in test mode, exactly the same as the one in figure 4.5 if the following conditions are met:

- The period of the test-clock signal, "*TClk*", should be sufficiently long in order to accommodate the propagation of the "*req*" signal through the complete logic path - inverter, MUX, XOR, ARBITER - until the **R1** register.
- The "*TE*" signal change from '1' to '0' should be delayed sufficiently long so that the arbitrary transition on the "*req*" signal, due to the last bit scanned-in, is able to propagate through the MUX, XOR, ARBITER and the **R1** register. If this is not the case, the captured values in the **R1** register may have a random value.

The clock "*TClk*" has usually a low frequency as compared to the functional one, while the delay of the "*TE*" signal can usually be programmed in the

test equipment. Therefore, the above restrictions will neither affect the test time nor hamper the test strategy.

4.1.3 Fault-Simulation Results

This section will show that the test vectors, generated by a conventional ATPG tool for the circuit in figure 4.6, will cover all the stuck-at faults that might occur in the complete scheme presented in figure 4.5.

An arbitrary scheme was considered, which consists of a synchronous core, an asynchronous core and an asynchronous-to-synchronous synchronizer. The asynchronous core was replaced with the scan-synchronous counterpart. Following this change, the synchronizer has been replaced with the "modeled-for-ATPG" version. The next step was to generate test vectors for the newly created schematic. The ATPG tool used in our work is *TetraMAX* [9]. The generated test vectors have been used to perform fault simulations of the additional faults not included in the "modeled-for-ATPG" scheme. Since the synchronizer contains an arbiter and a C-element, it is not possible to fault-simulate the entire system using conventional fault simulators. As a result, *HSPICE* [3] has been chosen to perform the fault simulations for the entire circuit. The SPICE model and the technology used were BSIM3v3 and UMC $0.18\mu m$ respectively. Other tools, like a VHDL/verilog simulator could also have been used. *HSPICE* has been preferred since a small circuit has been simulated, the simulations results are closer to reality and the scheme was generated automatically at the transistor level.

By using the ATPG test vectors as input stimuli, it was possible to prove that all the additional stuck-at faults were detected by the test vectors generated by *TetraMAX*. Table 4.1 shows the results for the considered circuit. The left columns present the data reported by the ATPG tool. The right columns show the data of the entire structure presented in figure 4.5. One may notice the additional faults that were not considered by the ATPG tool. However, these additional faults are still 100% tested by the ATPG vectors. This is the case because the synchronizer can be seen as a buffer, and the 55 additional faults are functionally-equivalent [1, pages 106-107] with the faults recognized by the ATPG tool. As a result, the total fault coverage is improving.

Table 4.1 also shows the number (3) of test vectors generated by the ATPG

Table 4.1: Fault coverage for the considered scheme containing an asynchronous-to-synchronous interface

"Modeled-for-ATPG" scheme (*TetraMAX*)		The complete scheme (*HSPICE*)	
Number of faults reported by the ATPG tool	280	Additional number of faults	55
Number of undetected faults	2	Additional undetected faults	0
Number of ATPG vectors	13	Detected faults by ATPG vectors	55
Reported ATPG fault-coverage	99.29%	Additional fault-coverage	100%
Total number of faults: **335**			
Total number of detected faults: **333**			
Total fault-coverage: **99,40%**			

tool. This is a small number of test vectors which are generated for the entire circuit. Therefore, for a real-life SoC, the burden of generating and applying additional test vectors, in order to test all ASIs, is negligible.

The 2 undetected stuck-at faults are related to the multiplexing operation of the clock signal, shown in figure 4.5.

Section 4.3 will tackle this apparent loss of fault-coverage in more detail.

4.2 Synchronous Producer Asynchronous Consumer

Figure 4.7 illustrates the reverse data communication, as compared to the previous section, the one between a synchronous producer and an asynchronous consumer. The asynchronous core is again using the two-phase handshake protocol [2, pages 489-490] for data exchange. In reference [5], this type of interface is called *"synchronous producer and asynchronous consumer"*. The clock distribution network has been included in the delay-path as before.

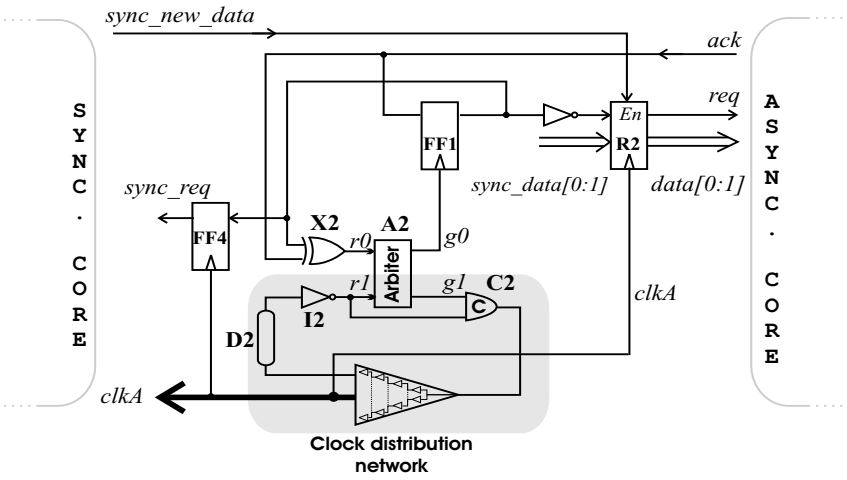


Figure 4.7: Synchronous producer - asynchronous consumer interface [5]

4.2.1 Description of the Functionality

For this type of ASI, the data to be passed from the synchronous core to the asynchronous one is synchronized with the "*clkA*" clock signal.

Whenever the synchronous domain is able to send data to the asynchronous one, the "*sync_new_data*" signal is asserted. The next clock cycle, the data ("*data[0:1]*") together with the request signal "*req*" are sent to the asynchronous domain. The asynchronous domain sends an acknowledge signal "*ack*" as soon as it finishes accepting the data.

Further more, this "*ack*" signal has to be synchronized with the synchronous clock "*clkA*". This synchronization is carried out by the **A2** arbiter. If the signals "*r0*" and "*r1*" are asserted simultaneously at the **A2** arbiter, then one of them will be put on hold. Presuming that:

- the signal "*r0*" is put on hold, the signal "*r1*" derived from the clock signal "*clkA*" will propagate further on through the arbiter, the C-element and so on. As a result, the "*r0*" signal, derived from "*ack*" signal, will be sensed by the **A2** arbiter before the next clock cycle.
- the signal "*r1*" is put on hold, the clock will be stopped for a short time until the "*r0*" signal propagates through the **A2** arbiter, the **FF1** flip-flop and the **X2** XOR gate.

The functional waveforms associated with the interface in figure 4.7 are pre-

sented in figure 4.8. It should be noted that in this case the request ("*req*")

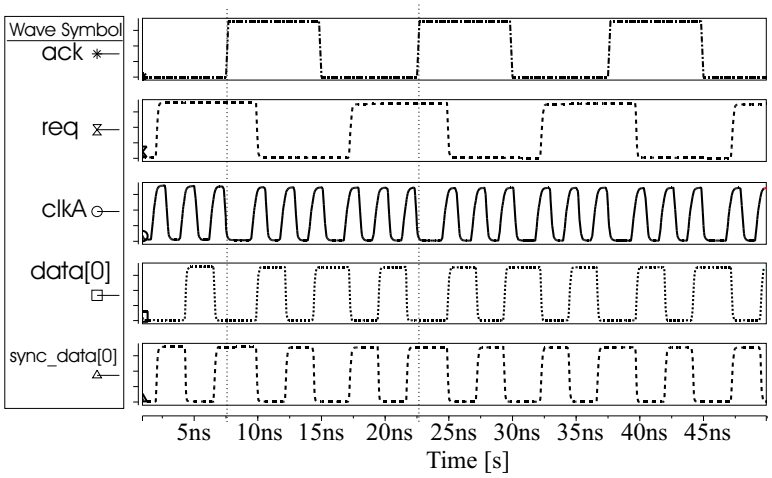


Figure 4.8: Simulated waveforms for a synchronous producer and an asynchronous consumer

and the acknowledge ("*ack*") signals are generated by the synchronous domain and the asynchronous domain respectively. In the previous case, presented in subsection 4.1, it was the other way around. As a result of this configuration, the arbiter will resolve the conflicts between derivatives of the "*ack*" signal and the "*clkA*" signal.

It can be seen in figure 4.8 that the "*clkA*" signal is delayed with respect to the "*ack*" signal around 7ns and 23ns.

4.2.2 Scan-Test Strategies

Figure 4.9 presents the test-scannable ASI version for a synchronous producer - asynchronous consumer. Apart from the synchronous-asynchronous interface presented in figure 4.7, one may notice the relevant scannable asynchronous part represented by the combinational logic circuit (**CLC**) block and the **FF3** flip-flop.

It can be seen that very little additional hardware has been added to the asynchronous-synchronous interface itself. This DfT hardware is represented by the wavy filling in figure 4.9, being basically a multiplexer and an inverter.

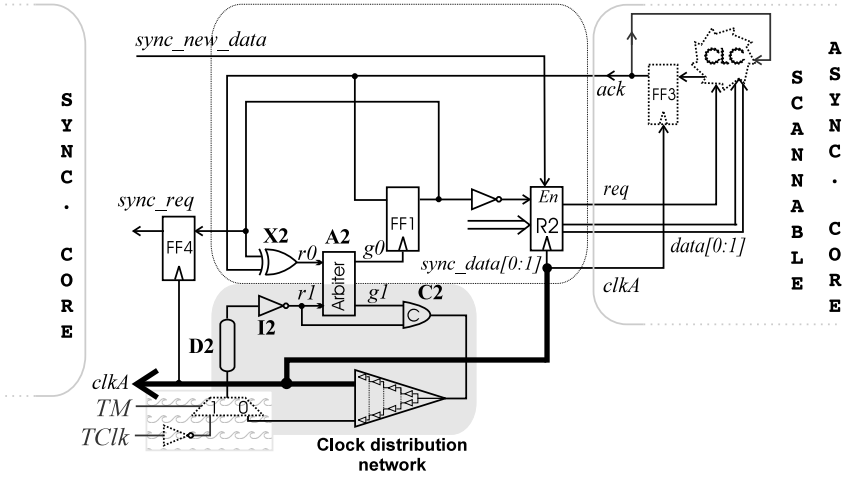


Figure 4.9: Test structures included in the synchronous producer - asynchronous consumer interface

However, if this scheme would be fed into an ATPG tool, it could not be recognized as a scannable design. Therefore, it is necessary to remodel the scheme for the ATPG tool required for generating the test vectors. The resulting scheme is shown in figure 4.10.

This scheme behaves, in test mode, exactly the same as the one in figure 4.9 if all the flip-flops are reset before the first test vector is scanned-in. In practice this is always the case; before starting the scan procedure, all the flip-flops are reset.

All the flip-flops/registers in figure 4.9 have been replaced by their scan counterparts except **FF1**. During the scan mode, the "*ack*" signal is changing all the time, depending on the data that is scanned in. Therefore, the value stored in the **FF1** flip-flop is the last value scanned in the **FF3** flip-flop. Hence, from the ATPG point of view, all circuitry between the "*ack*" signal and the **R2** register can be replaced by an inverter.

The test vectors can now be generated by a commercial ATPG tool for the circuit in figure 4.10. These vectors will be able to detect all stuck-at faults that may exist in the complete scheme shown in figure 4.9. The next section (4.2.3) will present the fault-coverage results obtained from the fault simulations.

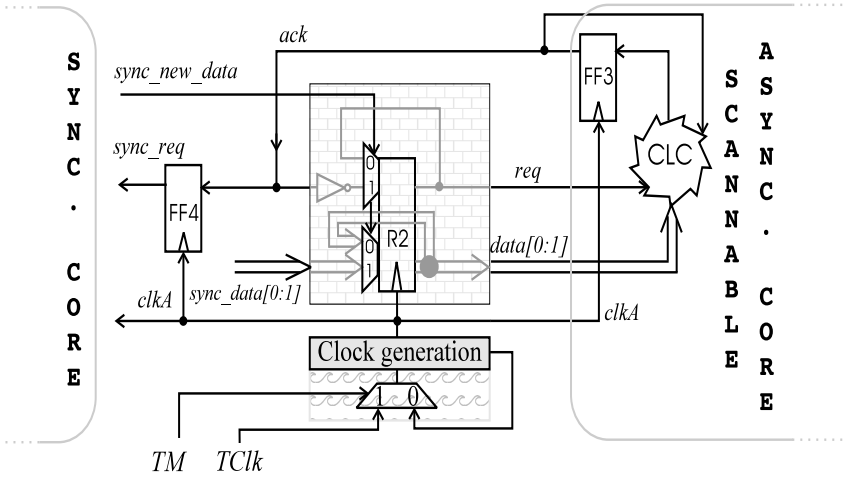


Figure 4.10: "Modeled-for-ATPG" scheme of the synchronous producer - asynchronous consumer interface

4.2.3 Fault-Simulation Results

An arbitrary scheme, comprising of a synchronous core, a synchronous-to-asynchronous synchronizer and an asynchronous core has been considered.

The asynchronous core has been replaced with its scan-test equivalent scheme. Following this change, the synchronizer has been replaced with its "modeled-for-ATPG" scheme. The obtained design has been used as input for the ATPG tool *TetraMAX* [9]. Then, the test vectors have been converted to *HSPICE* [3] format.

These vectors were used to fault simulate the complete synchronizer shown in figure 4.9. Table 4.2 summarizes these results. The same technology and models were used as for the previous case.

The main left column of table 4.2 shows the information reported by the ATPG tool if applied to the "modeled-for-ATPG" scheme. The total number of faults, for the entire "modeled-for-ATPG" scheme (the ASI plus the surrounding blocks) was 292. Out of these 292 faults only 2 were untested, and these belong to the MUX introduced in the clock path.

The main right column shows the additional faults for which the ATPG tool has not generated vectors. However, because these faults are functionally-equivalent with the already detected faults, all of them are detected. As

Table 4.2: Fault coverage for the considered scheme containing a synchronous to asynchronous interface

Modeled-for-ATPG scheme (*TetraMAX*)		The complete scheme (*HSPICE*)	
Number of faults reported by the ATPG tool	292	Additional number of faults	27
Number of undetected faults	2	Additional undetected faults	0
Number of ATPG vectors	11	Detected faults by ATPG vectors	27
Reported ATPG fault-coverage	99.31%	Additional fault-coverage	100%
Total number of faults: **319**			
Total number of detected faults: **317**			
Total fault-coverage: **99,37%**			

a result, the additional fault coverage, calculated as the number of the additional detected faults over the number of the additional faults, is 100%. The total fault-coverage has been increased from 99.31% to 99.37%.

As for the previous synchronizer, one may notice the small number of test vectors needed to test the complete scheme. In this case only 11 vectors are required.

4.3 Additional Fault Coverage Increase for ASIs

As was mentioned in the previous sections, there are two stuck-at faults belonging to the MUX inserted in the clock path, that are not tested in scan mode, since the "*TM*" signal will always be high.

Hence, scan cannot be used to test those two faults. A combined test strategy comprised of scan and BIST can be employed. Scan will test the majority of the faults, while BIST will test the remaining two.

The BIST structure will be very simple. A flip-flop as the one in figure 4.11 could be connected to the clock lines of the synchronous domains. This inserted flip-flop is made part of a scan chain, so it will be tested during scan-test operations. During a BIST-mode operation, the signal

"*TM*" will be zero and the internal free running clock will be generated. If a stuck-at fault is present in the untested part of the MUX then there will be no internal clock generated. As a result, the output of the inserted flip-flop will remain logic zero. If there is no stuck-at-fault present, then a "1" (Pass) will be latched in the flip-flop.

By using this extra flip-flop, both of the ASIs presented in section 4.1 and 4.2 will have 100% fault-coverage.

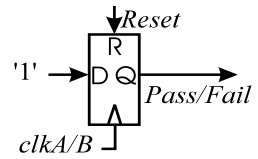


Figure 4.11: Introduction of an extra flip-flop for increasing the fault coverage in ASIs

4.4 Conclusions

In the coming years, the asynchronous-synchronous interfaces will become a common component in complex SoC designs. Therefore, it is essential to develop integrated test strategies with the surrounding cores. Albeit the area of the ASIs is small, their functionality is crucial for the whole SoC system. Testing is difficult since the ASIs contain asynchronous circuits inside.

An approach to test specific two-phase asynchronous - synchronous interfaces has been presented. By analyzing their functionality from the test point of view, it was possible to accommodate the scan technique as the test strategy.

These interfaces were previously tested in a functional way or not tested at all. The presented solution is able to test them in a structural way by using commercially available ATPG tools.

The DfT area overhead, in order to accommodate the scan chain, is extremely low since only one or two MUXs are added in order to make the circuits testable. The remodeling for the ATPG tool is the key step for the test process.

The burden of generating the few test vectors by the ATPG tool is negligible as explained earlier.

Fault simulations have been carried out to prove that the structural test vectors generated by the ATPG tool are detecting all the stuck-at faults that may appear in the asynchronous-synchronous interfaces. Using the previously presented DfT hardware and test strategies, it has been proven that it is possible to test any combination of asynchronous and synchronous cores as long as the asynchronous parts can be made scannable. If the asynchronous part is not scannable, then the solution can still be applied but it will not be so efficient in terms of fault coverage.

Section 5.3 in the next chapter presents how these scan-test techniques can be used to test a complete SoC design comprised of asynchronous and synchronous domains.

Bibliography

[1] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design (revised printing)*. ISBN 0-7803-1062-4. IEEE Press, 1990. 4.1.3

[2] William J. Dally and John W. Poulton. *Digital Systems Engineering*. ISBN 0-521-59292-5 (hb). Cambridge University Press, 1998. 4.1.1, 4.2

[3] HSPICE. *Golden Accuracy Circuit Simulator*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043. `http://www.synopsys.com/products/mixedsignal/hspice/hspice.html`. 4.1.3, 4.2.3

[4] Alain J. Martin. Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits. Technical Report CaltechCSTR:1989.cs-tr-89-01, California Institute of Technology, 1989. `http://resolver.caltech.edu/CaltechCSTR:1989.cs-tr-89-01`. 4.1.1

[5] Simon Moore, George Taylor, Robert Mullins, and Peter Robinson. Point to point GALS interconnect. *Eighth International Symposium on Asynchronous Circuits and Systems*, pages 69–75, April 2002. (document), 4, 4.1, 4.1.1, 4.2, 4.7

[6] Allen E. Sjogren and Chris J. Myers. Interfacing synchronous and asyn-

chronous modules within a high-speed pipeline. *Proceedings of the 17th Conference on Advanced Research in VLSI (ARVLSI '97)*, pages 47–61, September 1997. 4.1

[7] Frank te Beest. *Full scan testing of handshake circuits*. Ph.D. thesis, University of Twente, May 2003. 4.1.2

[8] Frank te Beest, Ad Peeters, Marc Verra, Kees van Berkel, and Hans Kerkhoff. Automatic scan insertion and test generation for asynchronous circuits. *International Test Conference (ITC)*, pages 804–813, October 2002. 4, 4.1.2

[9] TetraMAX ATPG. *High-Performance Automatic Test Pattern Generator Methodology Backgrounder*. Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043, May 1999. `http://www.synopsys.com/products/test/tetramax_wp.html`. 4.1.3, 4.2.3

Chapter 5

Unified Test Strategies for Core-Based Design Using Synchronizers

5.1 Introduction

This chapter will present test strategies at the system-level by integrating the test solutions presented in chapters 3 and 4 with established scan-test strategies in the test community.

Our test solutions have targeted hard-to-test synchronous-to-synchronous and asynchronous-to-synchronous synchronizers. Their function is to transfer data from one domain, being asynchronous or synchronous, to another. However, the design of these advanced synchronizers is not trivial and, as a result, the test strategies are not straight-forward.

With the previously presented synchronizer designs and their associated test strategies, a digital designer can use them "out-of-the-box" without worrying about their complex functionality or associated testing problems.

Our presented synchronizers have three modes of operation:

- *Functional mode.* In this mode, the synchronizer is performing its intended task
- *Scan-test mode.* The synchronizers are scan-test compatible. Scan-

test vectors can be generated by conventional commercial ATPG tools

- *BIST mode.* In this mode, the internal BIST circuit is activated. Additional test vectors are applied which will detect remaining stuck-at faults and the relevant path-delay faults

Figure 5.1 illustrates these three modes of operation for a generic synchronizer.

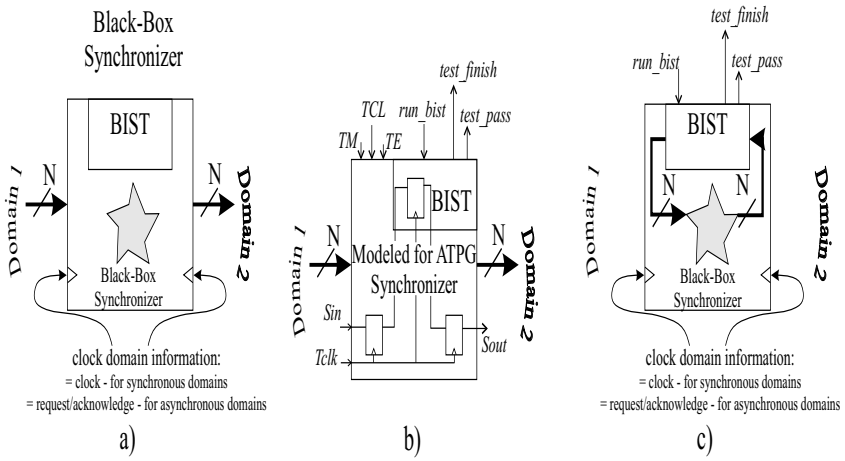


Figure 5.1: Three different representations for a generic synchronizer: a) functional mode; b) scan-test mode; c) BIST mode

In the *functional-mode* view, the minimal necessary design information are the number of bits to be synchronized and the domains information represented by the clock signals for the synchronous cores, and/or the acknowledge/request signals for the asynchronous cores.

For the *scan-test mode*, one recognizes the necessary signals to perform the scan operation:

- "*TM*": scan *T*est *M*ode signal. If the signal "*TM*" is logic one, the scan-test operation is activated. As opposed to a BIST operation where the signal "*run_bist*" is logic one and "*TM*" is logic zero.
- "*TE*": the *T*est *E*nable signal is used to switch between scan and capture operations
- "*Tclk*": The clock signal used in scan-test mode

- "*Sin*": *S*can *in* signal
- "*Sout*": *S*can *out* signal
- "*TCL*": Special signal used for multiple-clock domains (MCD) testing. The signal is described in [12, 18] and briefly in section 2.3.3 of this thesis, and it is used, together with additional latches which it controls, to stop the flip-flops in different clock domains to become transparent under certain skew conditions.

In the case of the *BIST mode*, the setup consists of three signals, being "*run_bist*", "*test_pass*" and "*test_finish*". This mode is resembling the functional mode because the synchronous clocks are running at-speed and the synchronizer is performing its functional task. However, the BIST structure will generate the inputs for the synchronizers and will monitor its outputs in this case. At the end of the BIST mode, a "*test_pass*" signal will indicate whether or not a defect has been detected by BIST. It should be mentioned that the stuck-at faults of the BIST structures are tested during the scan-mode test.

The combined scan-test and BIST strategies used for the synchronizers will now be extended to the system level.

A hypothetical SoC design is presented in figure 5.2; for the sake of simplicity only 2 cores are being considered.

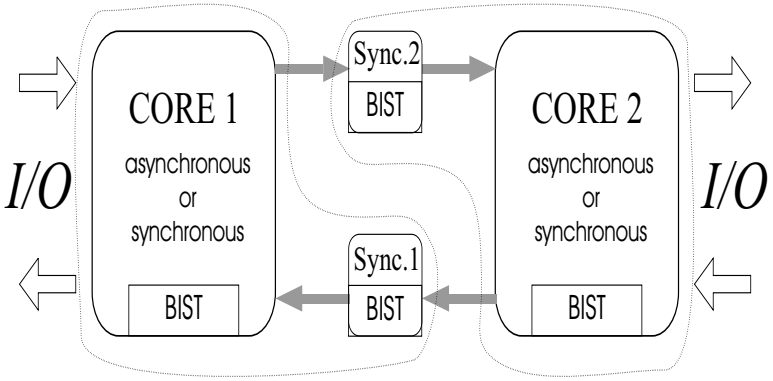


Figure 5.2: Hypothetical SoC design comprised of 2 cores

These two cores, **CORE1** and **CORE2**, are scan-test compatible. The two synchronization modules, **Sync.1** and **Sync.2**, are also scan-compatible as presented in chapters 3 and 4. All of them might incorporate an additional

BIST structure in order to increase the stuck-at/path-delay fault coverage and/or to perform additional tests.

The overall test strategy at the *top level* of the SoC design consists of the following two steps:

- ***Scan-test***. During this step, the traditional scan-test methodology is employed. If the test clock, which is the functional clock for synchronous cores and a special test clock signal for the asynchronous ones, can be distributed over the whole chip without any skew problems then the complete SoC can be tested using the classical scan-test method for one clock domain. If, however, the skew creates problems in test mode, more complicated, but available [12, 18], test strategies can be employed, and the SoC will be considered as a multiple-clock domains system.

  The predefined scan-chains of the synchronizers are compatible with both scan-test strategies, being single or multiple clock domain(s). During the scan-test all the BIST structures of the synchronizers are also tested.

- ***BIST test***. Additional at-speed test vectors are generated on-chip by the BIST structures for a higher fault coverage. At the end of the BIST test, "*test_pass*" and "*test_finish*" signals should be checked.

The exact methodology of applying these two test modes will not be described in this chapter. The test engineer can integrate them using a JTAG 1149.1 [3] standard or, better, a future P1500 [4] standard.

The following two sections will present the necessary steps needed in order to modify the SoC design for obtaining compatibility with the scan and BIST test strategy given above.

Section 5.2 will investigate the test strategy for a multiple-clock domains SoC design incorporating the synchronizers presented in chapter 3.

Section 5.3 will address the same problem as in section 5.2, but here the synchronizers presented in chapter 4 will be used.

In section 5.4 a summary and conclusions will be given.

5.2 Unified Test Strategies for Synchronous Core - Based Design

The usage of synchronous core-based design is a common trend nowadays. Together with this design style, test strategies at the system level have been proposed in order to cope with the increased complexity. Among them, the P1500 standardization effort [4] is the most important. The standard defines the method how cores are accessed in test mode via well-defined wrapper-cells; however, it leaves sufficient freedom for the test engineer to choose the test strategy.

Chapter 3 has presented scan-test strategies for different types of synchronizers required to connect certain synchronous cores. In order to have a coherent test strategy, the presented techniques must be incorporated in the overall test strategy at the system level as suggested in the previous section.

From the design point of view, the synchronous-to-synchronous synchronizers can be seen as black-box modules, ready to be used in a multiple-clock domains system. From the test point of view, they can be seen either as stand-alone cores with scan capabilities or as being part of a certain core. This is illustrated in figure 5.2, where **Sync.2** can be considered part of **Core 2**, and **Sync.1** part of **Core 1**. This inclusion will decrease the number of cores considered at the system level, while preserving the test strategies.

The inclusion of the synchronizers in the cores is possible since the test clock is usually the same as the receiving clock.

A flow chart comprising the main steps necessary to test the SoC at system level is presented in figure 5.3.

The first test to be applied is the conventional scan-test. All clock domains are first identified by tracing the clock signals of the flip-flops. This step is necessary in order to keep a better track of the synchronizers that are "buffering" cores having different clock domains. The synchronizers are subsequently identified. Identification can be manually or automatically, aided by the previous steps. In order to be able to generate test vectors using commercial ATPG tools, the synchronizers are then replaced with their "remodeled-for-ATPG" counterparts. After this replacement, scan-test vectors can be generated. These vectors will be later applied to the

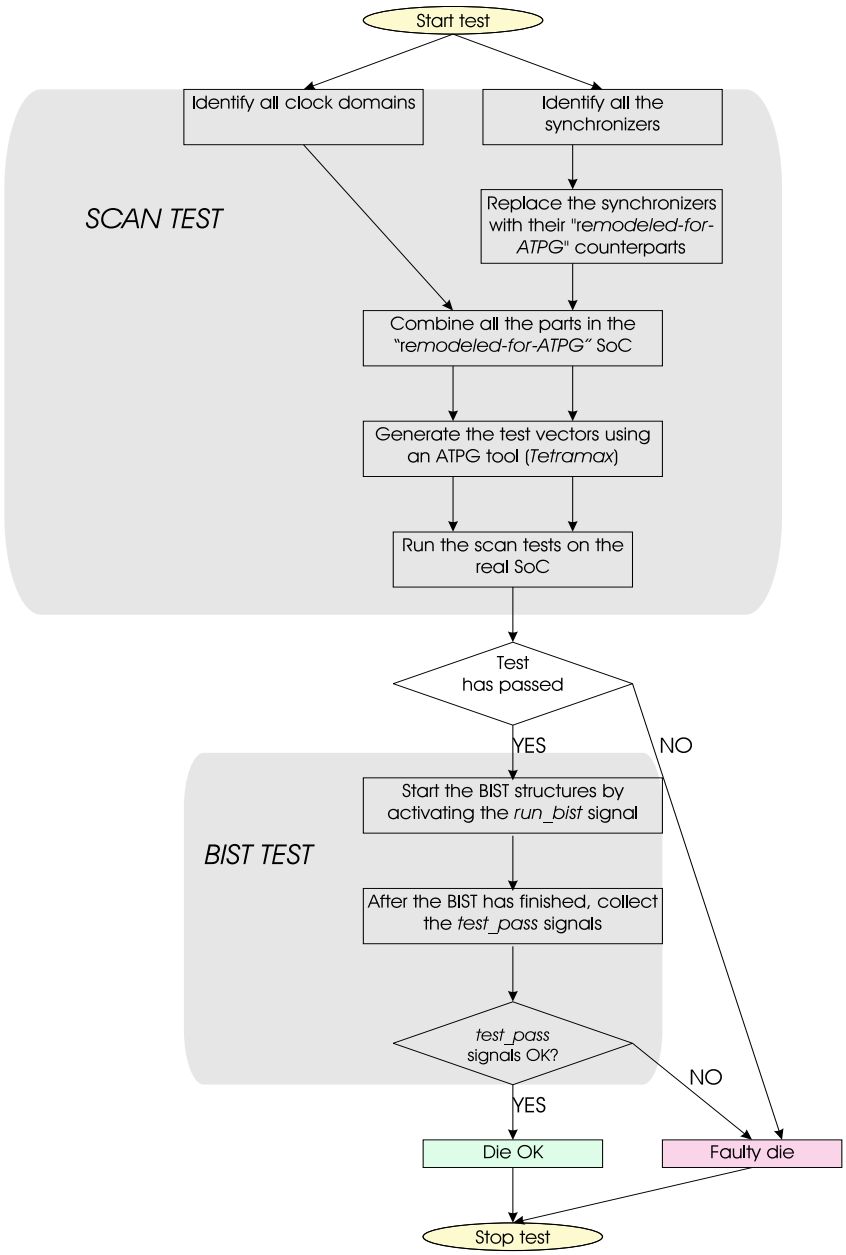


Figure 5.3: Flow chart for testing a complete SoC comprised of synchronous cores only

whole SoC. If the scan-test fails, there is no need to continue with the BIST section. If, however, the scan-test is successful, the BIST mode will be activated. There could be numerous BIST structures that are testing different parts of a SoC design. Among them one can mention, logicBIST [1, 5, 8], which could be used in blocks where the internal scan-chain might not be available, memBIST [11] used to test memory blocks, PLL-BIST [11, 13, 17] employed to test phase-locked loops, and BIST for ADC/DAC [2, 6, 11] used to test analog-to-digital (or vice versa) converters, etc.

A real SoC may comprise some or all of the above mentioned BIST blocks. Some of these BISTs can be run in parallel and some not. It is the task of the test engineer to schedule the exact test strategy. Our presented flow chart of figure 5.3 shows only one generic BIST step.

The BIST test step presented in figure 5.3 refers to our BIST strategies developed in chapter 3, and it is referred to as "synchronizers BIST".

After completion of the BIST, a statement with regard to the correctness of the SoC can be made.

5.3 Unified Test Strategies for Asynchronous - Synchronous Core-Based Design

There is no doubt that future SoCs will embed an increasing number of IP (Intellectual Property) cores and many of them will be fully or partially asynchronous. The synchronization strategies, presented briefly in chapter 4 and in [9, 10], will help integrate the asynchronous and synchronous design styles. The scan-test strategy, shown in the same chapter, will minimize the time-to-market requirements. Since most of todays IP cores are scan-chain compatible, an extension of this strategy to the SoC level is required.

Subsection 5.3.1 will analyze the modifications required for the ASIs in order to accommodate multiple interactions between synchronous and asynchronous cores. As will be demonstrated, no fundamental changes have to be carried out during this step.

In subsection 5.3.2, a hypothetical future SoC, containing some asynchronous cores, will be presented together with the required steps to test it.

5.3.1 Multiple Interactions between Asynchronous and Synchronous Cores

The two ASIs presented in sections 4.1 and 4.2 of chapter 4 represent the kernel designs for these types of synchronizers.

In an actual SoC, there is usually more than one asynchronous producer sending data to a synchronous consumer. In order to accommodate this situation, some annotations to the clock-generation procedure should be performed. The resulting new scheme is presented in figure 5.4.

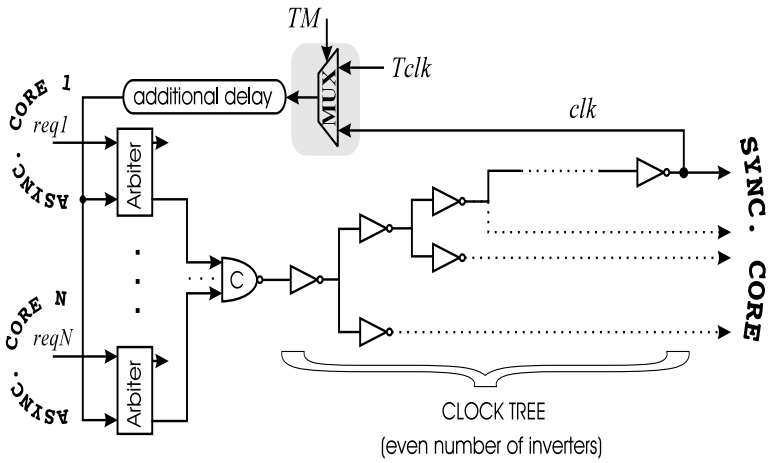


Figure 5.4: Multiple asynchronous requests for a single clock domain

For each asynchronous core sending data to the synchronous core, an arbiter has been added. These arbiters are connected to only one C-element.

In this figure, the DfT hardware has also been taken into account, being represented by the MUX encompassed in the shaded area. This MUX is necessary to control the clock in test mode. The test strategy for this type of configuration will remain the same as the ones presented in section 4.1.

The other case where there is only one synchronous producer and many asynchronous consumers does not pose any problems, because no redesign of the ASIs should take place as opposed to the previous case just shown. Therefore the test strategies remain the same.

5.3.2 System-Level Testing of Asynchronous-Synchronous Core-Based Designs

The scan-test strategy presented in sections 4.1.2 and 4.2.2 can be used to test more complex systems comprised of many asynchronous and synchronous cores. Each synchronous core will have its own clock generated on-chip. The clocks might be also stopped as was explained in sections 4.1.1 and 4.2.1.

Because of recent progress in testing asynchronous cores [7, 14–16], using scan-test techniques, it is now possible to efficiently test SoCs comprised of synchronous and asynchronous cores.

All asynchronous and synchronous cores can be transformed in scan-compatible cores in test mode. The glue logic and the synchronization blocks are usually synchronizers as the ones presented in chapter 4. Therefore, they can also be transformed in scan-compatible blocks.

With everything scan-compatible, it is relatively easy to construct a coherent scan-test strategy at the system level. The problem of testing SoCs comprised of asynchronous and synchronous cores is thus being transformed to a MCD (Multiple-Clock Domain) testing problem for which scan-test solutions are already available [12, 18].

From the test point of view, the SoC level can be seen as the one illustrated in figure 5.5. In this figure, two asynchronous cores and one synchronous core have been considered together with all possible combinations of communication between the cores. In this particular case, six synchronizers modules (4 of them are ASIs) have been represented. However, in a real SoC an excessive number of synchronizers can be encountered between cores. The exact number is application specific. This in return decreases the ability to test them as stand-alone cores. Rather, they will be integrated in the surrounding cores as was also suggested for the synchronous core-based design presented in the previous sections. The synchronizers will be included in the 'receiving' cores as highlighted in figure 5.5. The inclusion of the synchronizers in the receiving cores is driven by the fact that the clock domains of the synchronizers are derived from the receiving clock and not from the transmitting one.

A flow chart including the main steps to test the complete SoC is presented in figure 5.6. These steps are only the basic ones necessary to test the entire SoC. However, because of the complexity of an SoC, other tests or

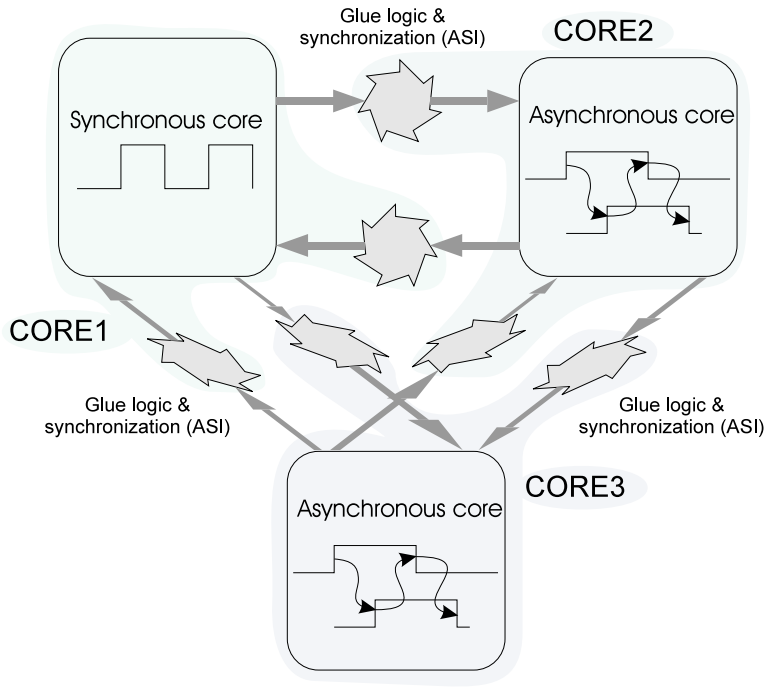


Figure 5.5: A hypothetical SoC design using synchronous and asynchronous cores

preparation steps should be carried out before the scan-test can be applied. These are SoC specific and they are highly dependent on the chosen system-test strategy.

One should notice that apart from the synchronizers being replaced with their scan counterparts, the asynchronous circuits should also be substituted by their "remodeled-for-ATPG" schemes.

5.4 Conclusions

Our combined scan and BIST test strategies presented in chapters 3 and 4, have been integrated at the system level. The integration is simple and highly dependent on the overall test strategy developed by the test engineer. The two key features of the integration of the test strategies at the system level are the scan-test compatibility and the BIST concept. The synchronizers should be seen as "black-boxes", or better "library-cells",

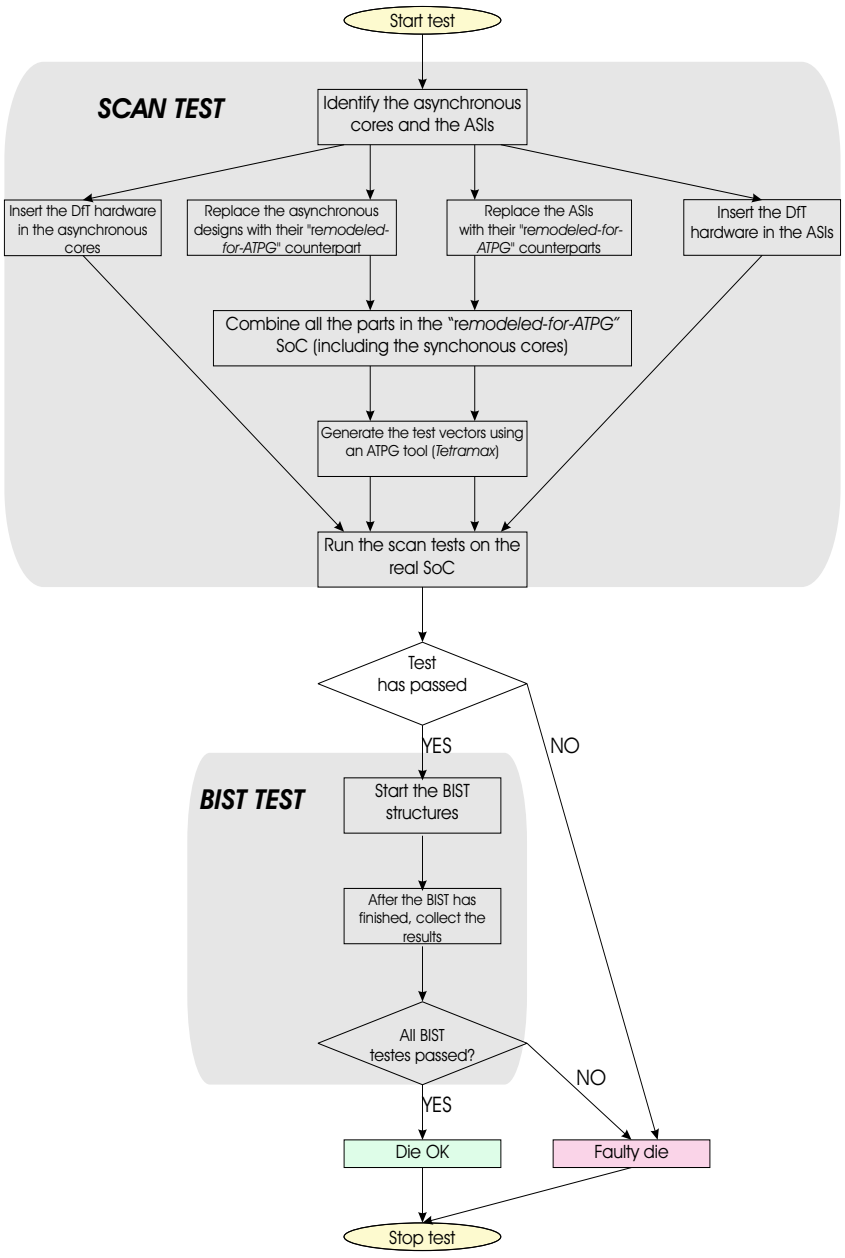


Figure 5.6: Flow chart for testing the complete SoC comprised of synchronous and asynchronous cores

performing the synchronization operation between different types of cores. They will have an associated scan model, necessary for generating ATPG test vectors, and they should be integrated in the system BIST strategy. All synchronizers should also be included in the receiving cores in order to reduce the number of cores considered in test mode.

Bibliography

[1] Graham Hetherington et al. Logic BIST for large industrial designs: Real issues and case studies. *International Test Conference (ITC)*, pages 358–367, September 1999. 5.2

[2] J.L. Huang, C.K. Ong, and K.T. (Tim) Cheng. A BIST scheme for on-chip ADC and DAC testing. In *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 216–220, March 2000. 5.2

[3] IEEE JTAG1149.4. *Mixed-Signal Test Bus Working Group*. IEEE. `http://grouper.ieee.org/groups/1149/4/`. 5.1

[4] IEEE P1500. Standard for Embedded Core Test (SECT). `http://grouper.ieee.org/groups/1500/`. 5.1, 5.2

[5] Alexander Irion, Gundolf Kiefer, Harald Vranken, and Hans-Joachim Wunderlich. Circuit partitioning for efficient logic BIST synthesis. *Design, Automation and Test in Europe (DATE)*, pages 86–91, March 2001. 5.2

[6] Huang kai Chen, Chih hu Wang, and Chau chin Su. A self calibrated ADC BIST methodology. In *VLSI Test Symposium (VTS)*, pages 117–122, April 2002. 5.2

[7] Ajay Khoche and Erik Brunvand. A partial scan methodology for testing self-timed circuits. *IEEE VLSI Test Symposium (VTS)*, pages 283–289, April 1995. 5.3.2

[8] Gundolf Kiefer, Harald Vranken, Erik Jan Marinissen, and Hans-Joachim Wunderlich. Application of deterministic logic BIST on industrial circuits. *International Test Conference (ITC)*, pages 105–114, October 2000. 5.2

[9] Simon Moore, George Taylor, Robert Mullins, and Peter Robinson. Point to point GALS interconnect. *Eighth International Symposium on Asynchronous Circuits and Systems*, pages 69–75, April 2002. 5.3

[10] S.W. Moore, G.S. Taylor, P.A. Cunningham, et al. Using stoppable clocks to safely interface asynchronous and synchronous subsystems. *AINT'2000 Asynchronous Interfaces: Tools, Techniques, and Implementations*, pages 129–132, July 2000. 5.3

[11] Benoit Nadeau-Dostie. *Design for AT-Speed Test, Diagnosis and Measurement*, volume 15 of *FRONTIERS IN ELECTRONIC TESTING*. Kluwer Academic Publishers, Boston, September 1999. 5.2

[12] Josef Schmid and Joachim Knblein. Advanced Synchronous Scan Test Methodology for Multi Clock Domain ASICs. *IEEE VLSI Test Symposium (VTS)*, pages 106–113, April 1999. 5.1, 5.1, 5.3.2

[13] Stephen Sunter and Aubin Roy. BIST for phase-locked loops in digital applications. *International Test Conference (ITC)*, pages 532–540, September 1999. 5.2

[14] Frank te Beest. *Full scan testing of handshake circuits*. Ph.D. thesis, University of Twente, May 2003. 5.3.2

[15] Frank te Beest, Ad Peeters, Marc Verra, Kees van Berkel, and Hans Kerkhoff. Automatic scan insertion and test generation for asynchronous circuits. *International Test Conference (ITC)*, pages 804–813, October 2002. 5.3.2

[16] Frank te Beest, Ad Peeters, Marc Verra, Kees van Berkel, and Hans Kerkhoff. Synchronous full-scan for asynchronous handshake circuits. *Proceedings of the European Test Workshop (ETW)*, pages 381–387, May 2002. 5.3.2

[17] Benoît Veillette and Gordon W. Roberts. Stimulus generation for Built-In Self-Test of charge-pump phase-locked loops. *International Test Conference (ITC)*, pages 688–697, October 1998. 5.2

[18] Bart Vermeulen, Maurice Lousberg, and Paul Merkus. Scan test techniques for multi synchronous digital circuits. *Intenational Test Synthesis Workshop (ITSW)*, March 1998. 5.1, 5.1, 5.3.2

Chapter 6

Off-Chip Interaction between SoCs - High Speed Interfaces

During the past years, due to the decrease of the minimum feature size in CMOS technology, the on-chip clock speed has increased dramatically ranging into the GHz domain. This increase has also pushed the need for higher data-transfer rates between these high-speed ICs, in order to optimize the entire PCB system. However, due to high-speed data transfers between ICs, and because of power-supply and temperature variations throughout the normal operation, the clock/data skew can span tens of clock cycles. In order to cope with this new situation, synchronization strategies have been developed [14, 16, 21]. Some of the synchronization techniques rely on delay-lines [1, 8, 13, 18], either controlled or not. In order for the synchronization mechanism to function properly, the tap-delays of the delay-line should be within an acceptable, predefined range. This chapter presents a technique to accurately measure/characterize such tap-delays. It also presents a new delay-line scheme which is better suited for the chosen test method.

The chapter is organized as follows: the next section 6.1 will provide an overview of some existing high-speed inter-chip synchronization strategies. The main part of every synchronization module will be a high-speed multi-tap delay-line. A possible implementation of such a delay-line will be shown in subsection 6.1.2. The DfT hardware requirements for the delay-line will

be presented in the last subsection 6.1.3.

Section 6.2 will present the oscillation test technique. Subsection 6.2.1 will explain how this technique can be used to detect manufacturing faults, being either stuck-at or path delay faults. Next, subsections 6.2.2 and 6.2.3 will present the methods utilized to measure the tap delays of the delay-line.

Section 6.3 will investigate the measurement accuracy of the proposed method. Various measurement error sources, such as jitter, power supply, etcetera, will be analyzed (6.3.1 - 6.3.4) in order to determine the accuracy of the proposed method. Subsection 6.3.5 will present how the proposed oscillation test technique can be used to detect whether stuck-at faults and/or path-delay faults are present in a design. Subsection 6.3.6 will present some additional observations regarding the different values of the tap delays in functional and test mode. In addition, recommendations are given on how to minimize this unwanted effect.

Section 6.4 will present a new delay-line design which is less susceptible in test mode to analogue parameter variations. Comparisons, in terms of the measurement accuracy, between the original and the proposed delay-line will be performed (6.4.1 - 6.4.3).

An IC has been designed and fabricated comprising the new delay-line and our BIST hardware. The scheme and description of the implementation are presented in section 6.5. Measurements on the "DLLINE" IC have been performed and the results are shown in section 6.6.

Finally, the conclusions of the chapter are presented in section 6.7.

6.1 Functional Descriptions of High-Speed Interfaces

SoC design is being facilitated by the development of new deep sub-micron technologies. More and more cores can be squeezed on the same die, resulting in ever more complex systems. While the number of devices per SoC is increasing at an exponential rate, the number of access pins of the SoC is not. A huge amount of information generated/received by the SoC should pass via a small number of pins. In order not to hinder the SoC design strategy, the transfer rate per pin should be increased dramatically.

High-speed synchronization strategies between ICs have been developed to address this issue.

The next subsection will present such possible synchronization strategies while keeping the focus on the testability problems.

6.1.1 High-Speed Synchronization Strategies Between SoCs

A lot of papers [8, 9, 12, 14, 16–18] have been published that address the high-speed data synchronization between SoCs, where the SoCs can be either processors, co-processors, memories, etc.

All of them have something in common, being a delay-line either digital or analogue, controlled or not, tapped or not. The synchronization mechanism is dependent on these delay-lines and any manufacturing fault, either catastrophic or not, will hamper the functionality of the synchronization mechanism.

A generic architecture of a high-speed synchronization mechanism is presented in figure 6.1. The data can be transmitted from one chip to another using low-voltage differential signaling (LVDS) [15]. In figure 6.1, the differential inputs are represented by the "inP" and "inN" signals. The transmitted data is synchronous with the clock signal "clk" but the skew between the data and the clock can span many clock cycles. The differential signal is transformed on-chip in a single-ended signal ("in") by the **AMP** operational amplifier. This signal then passes a tapped delay-line. Different delayed versions of the "in" signal can be selected by the two MUXs. There are two symmetrical paths. These two paths are necessary for the **digital synchronization control** module, seen on the right side of figure 6.1. Following the selection of a certain tap, the data is transformed from serial to parallel by the **deserialiser** module (figure 6.1). This module comprises of only flip-flops, which are clocked by different phases and frequencies of the initial clock "clk". The **clock division block** is the one that takes care of the generation of the clocks for the whole design.

The **digital synchronization control** module is generating all the select signals for the two access MUXs. In this block, the correct sampling point of the incoming data is calculated and the proper tap is selected. We will not go into details about the exact algorithm behind this block. However, it should be mentioned that the **digital synchronization control** module is very sensitive to tap-delay variations of the delay-line. The DfT hardware

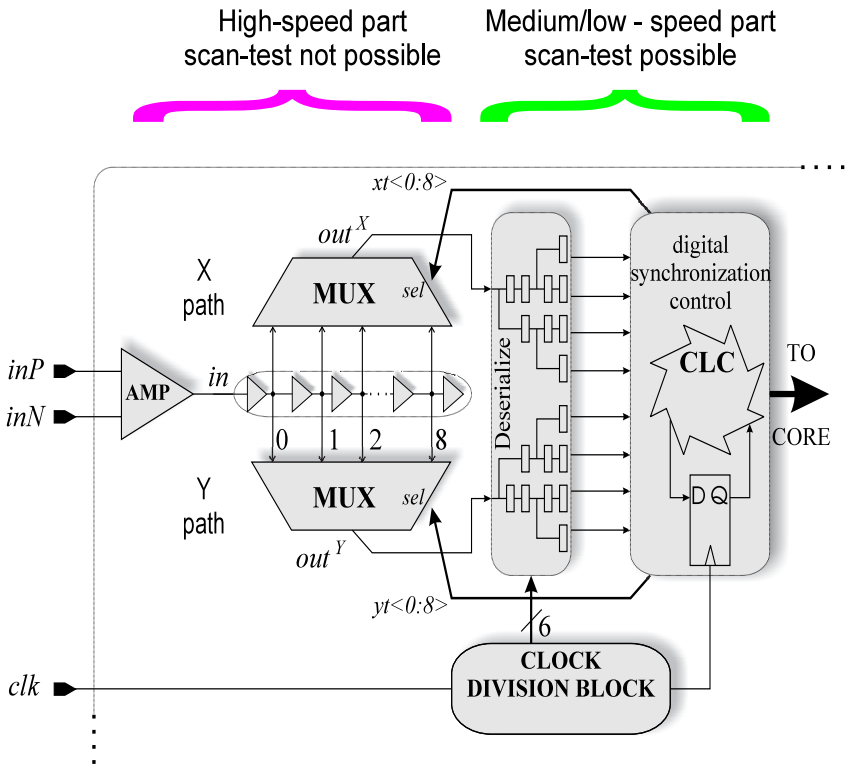


Figure 6.1: Generic architecture of a high-speed synchronization mechanism

requirements of the delay-line module, presented in subsection 6.1.3, are imposed by the **digital synchronization control** module.

Testing these type of interfaces in general, and the delay-lines in particular, is very difficult while using a classical scan-test method. A list of reasons is given below why the scan-test technique cannot be easily applied:

- Most of the time, the high-speed interface design comprises some analogue modules, while the scan-test is intended for digital testing only.
- Even if the synchronization mechanism is pure digital, scan-test cannot be applied due to the negative impact on the speed performance of the interface.
- Scan-test has been mainly designed for stuck-at faults. However, for a high-speed interface especially delay faults are crucial.

- At-speed test using the scan-test strategy can be prohibitively expensive because it can only be performed using costly high-speed ATE systems.
- Scan-test of a delay-line will introduce few undetected stuck-at faults due to the inability to provide different inputs to the access multiplexers (see section 3.3.1).

The following section will present a generic delay-line used as a core part in a high-speed synchronization mechanism.

6.1.2 Digital Delay-Line Design: the High-Speed Part

The design of a delay-line for which it is intended to measure the tap-delays is presented in figure 6.2. The intrinsic buffers, having the propagation

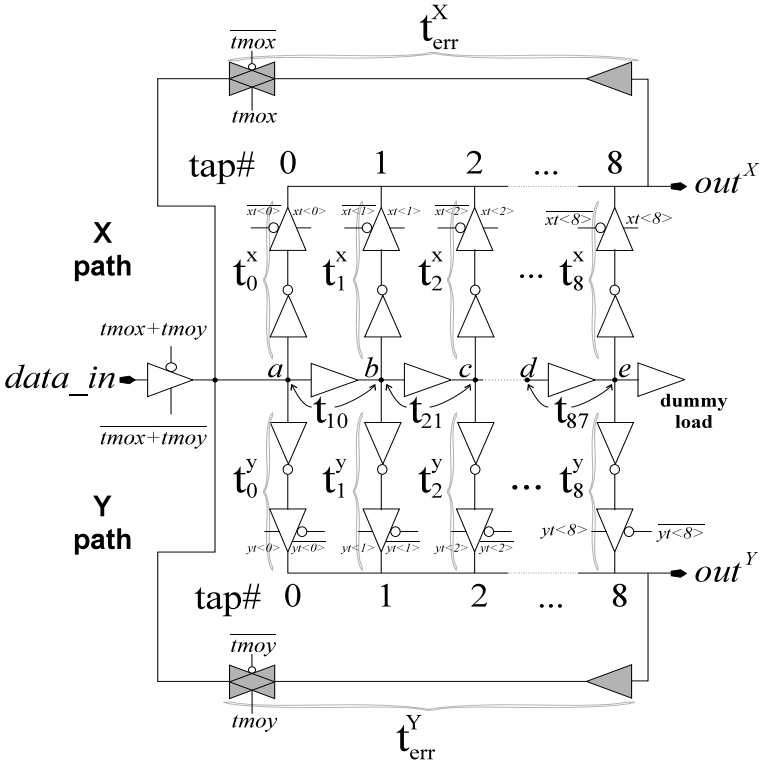


Figure 6.2: Basic delay-line as used in the synchronization mechanism. (The shaded components belong to the DfT hardware)

delays t_{10}, t_{21}, ..., t_{87}, are comprised of an odd number of inverters, where the exact number is application specific. Usually the number of inverters inside a buffer is 2; therefore the delay associated with these tap-delays is very small as can be seen later on in figure 6.3. Each path, either X or Y, has only one enabled output-buffer at any time, driving the out^X or out^Y signals respectively.

In functional mode, the signal "*data_in*" is propagating to the output signals "out^X" and "out^Y" via a certain number of taps selected by one of the signals "*xt<0>*", ..., "*xt<8>*" and "*yt<0>*" ... "*yt<8>*". Only one of the signals "*xt<0>*", ..., "*xt<8>*" can be logic one at a certain moment. This also holds for the signals "*yt<0>*", ..., "*yt<8>*".

The basic DfT hardware used to make the circuit testable using the oscillation technique (section 6.2.1), is shaded in figure 6.2. As it will be shown later in section 6.2.2, measuring the oscillation frequency at different taps will help identifying the tap-delays. During the test mode, described in 6.2, one of the signals "*tmox*" or "*tmoy*" is logic one.

The notations t_{10}, t_0^x, t_0^y, t_{err}^X, t_{err}^Y, etc., will be explained in section 6.2.2 when the basic equations for measuring the tap-delays are presented.

6.1.3 DfT Hardware Requirements

The delay-line design is straightforward. However, the synchronization algorithm is very sensitive to variations of the tap-delays. A maximum acceptable deviation of $\pm 5\%$ has been assumed for the designed tap-delay value. This tap-delay value accuracy is required by the synchronization mechanism following the delay-line.

Measuring a tap-delay with an accuracy of at least 5% is not a trivial task, since the required accuracy is in the same range as the rising and falling times of the signals. This can be seen from the simulation results of the delay-line as presented in figure 6.3.

In figure 6.3, the dotted-plot signal is the delayed version of the continuous-plot signal due to the buffer between taps 1 and 2. Beside the rising/falling times of the signals "*b*" and "*c*" (derived from the "*data_in*" signal), both rising and falling propagation delays of the t_{21} delay-element are shown (see figure 6.2). However, only the rising propagation delay is calculated being approximately 285ps. Measuring this tap-delay with an accuracy of

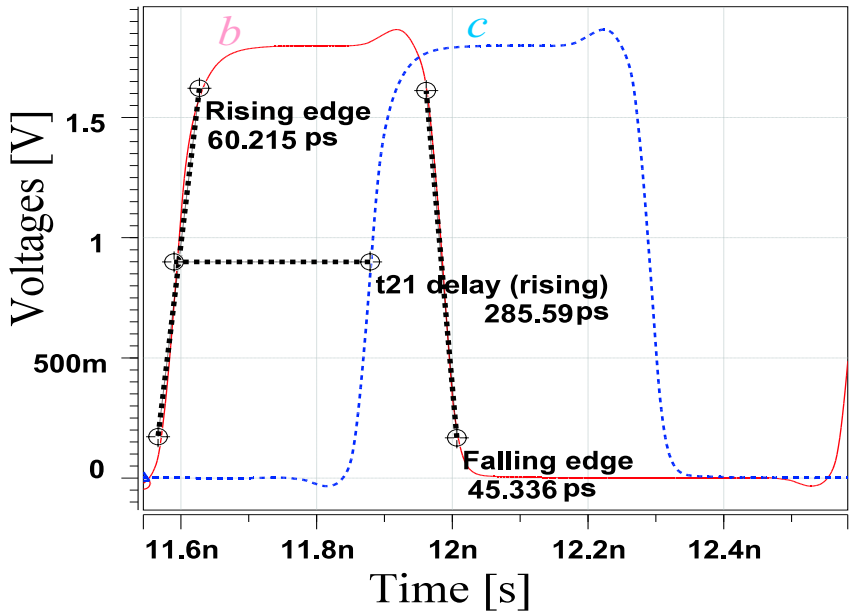


Figure 6.3: Simulated rising/falling time and t_{21} propagation delay (rising) of the digital delay line as presented in figure 6.2.

$\pm 5\%$ is equivalent with an acceptable measurement error of $\pm 14ps$. One may notice that the rising time of the signal is 60ps, which is more than 4 times the accepted measurement accuracy.

The oscillation-test technique [2, 4] is able to provide the necessary measurement accuracy if the oscillation period can be accurately determined. The following sections of this chapter will present the oscillation technique together with the method employed for measuring the tap-delays.

6.2 The Oscillation-Technique Test Method for High-Speed Interfaces

Oscillation techniques have been extensively used in the past to characterize, measure and analyze delays. The ring oscillator was probably the first design utilized to characterize the speed of a new technology using the oscillation technique. Since then, the same oscillation technique has been also used to test analog and digital circuits [2–4]. The power of this

technique lies in the fact that the *average* delay in the oscillation loop is equal with the oscillation period. Therefore, if the oscillation period or the frequency is precisely known, the average delay of the oscillation loop can be calculated.

6.2.1 General Oscillation-Technique Method Description for Identifying Manufacturing Faults

The principle of the oscillation technique for detecting manufacturing faults is well described in [2–4]. However, for the sake of completeness we will present a very brief description of the technique. Since the oscillation method will be applied to a digital design, the description in this chapter will limit itself to combinational logic circuits (CLCs).

Let us considering a simple CLC as the one shown in figure 6.4.

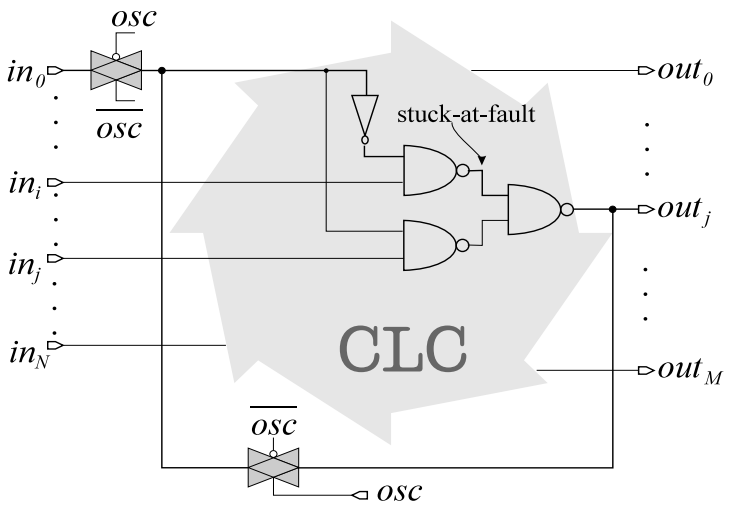


Figure 6.4: The CLC set-up for an oscillation test approach

Let us also consider that there is a stuck-at-fault, either s-a-0 or s-a-1, in one of the connecting wires between the two NANDs as presented in figure 6.4. By closing the feedback loop via the signal "*osc*", an oscillation will start if, and only if, the input signals in_i and in_j have "1" and "0" values respectively, and there is no stuck-at fault present in the oscillation path. It should also be observed that a s-a-0 on in_i or a s-a-1 on in_j will also block the oscillation. Therefore, by selecting and sensitizing all the relevant

paths in the CLC, the stuck-at faults can be immediately identified due to the absence of the oscillation.

The oscillation method can also detect delay-faults since the oscillation period is equal to twice the sum of the average falling and rising propagation delays. If a delay-fault is present in the sensitized path, then the oscillation period will have a different value than the expected one. If the oscillation period, or the oscillation frequency, can be precisely measured, then the delay deviation in the oscillation path can be detected.

6.2.2 Tap-Delay Measurements using an Oscillation Technique: Basic Equations

Equation 6.1 describes the first step of the method for determining the tap-delays. The circuit in figure 6.2 will be forced to oscillate by manipulating the control signals "*tmox*" and "*tmoy*" of the transmission gates. During the oscillation-test mode, the shaded transmission gate, belonging to the DfT hardware of path Y in figure 6.2 is conducting ("*tmoy*" is logic one), while the transmission-gate counterpart of path X is non-conducting ("*tmox*" is logic zero). Moreover, the buffer in front of the "*data_in*" signal will be in the high-Z state.

The parameters t^Y_{err}, $t_{(j)(j-1)}$ and t^y_k for $j = 1, \ldots, 8$ and $k = 0, 1, \ldots, 8$ appearing at the left of the equal sign in equation 6.1, are depicted in figure 6.2. They represent the average delays between falling and rising propagation delays.

Parameter T^Y_k for $k = 0, 1, \ldots, 8$ denotes the oscillation period associated with the Y path when the output-buffer corresponding with the tap number k is enabled.

$$t^Y_{err} + \sum_{j=1}^{k} t_{(j)(j-1)} + t^y_k = \frac{T^Y_k}{2}, \quad k = 0, 1, \ldots, 8 \tag{6.1}$$

The above equation is only referring to the Y path. For path X, equation (6.1) can be easily adapted. The remaining of this section will perform the analysis for the path Y only since both paths, X and Y, are equivalent.

By subtracting two adjacent oscillation periods using equation 6.1 one can

deduce:

$$T_i^Y - T_{i-1}^Y \stackrel{def}{=} T_{(i)(i-1)}^Y = 2(t_{(i)(i-1)} + t_i^y - t_{i-1}^y), \quad i = 1, \ldots, 8 \qquad (6.2)$$

In equation 6.2, $T_{(i)(i-1)}^Y/2$ represents the value of the tap-delay between *tap#i* and *tap#i-1* at the Y path access points, and not the intrinsic tap-delay ($t_{(i)(i-1)}$). From a functional point of view, the tap-delays at the Y path access points are of more importance than the intrinsic tap-delays, since the signals out^Y and out^X are the ones used afterwards by the synchronization mechanism. However, in a fault-free situation, one expects $T_{(i)(i-1)}^Y/2$ to be very close to the $t_{(i)(i-1)}$ value. This because the inverters and the tri-state buffers on the path Y at *tap#i* and *tap#i-1*, which are a source of uncertainty, will be very well matched in the layout, and therefore $t_i^y \simeq t_{i-1}^y$, $i = 1, \ldots, 8$.

As can be noticed, the parameter t_{err}^Y does not appear in equation 6.2. Therefore, the additional hardware, represented by the shadowed components in figure 6.2, does **not** influence the computation of the tap-delays.

One observation should be made. The oscillation periods are larger than the tap-delays to be measured, and the tap-delays are obtained by subtracting various oscillation periods. Therefore the error for measuring the oscillation periods should be much lower than the given tolerance for the tap-delay ($\pm 5\%$). From delay-line simulations, it has been found out that the ratio of the oscillation period with respect to the tap-delay can be up to 25. Hence, the error for measuring the oscillation period in such a case should be $\pm 0.2\%$ in order to be within $\pm 5\%$ tolerance for measuring the tap-delay. An example is given below.

For a tap-delay t_{87} of 400ps, the $\pm 5\%$ accuracy is equivalent with $\pm$20ps. It will be further considered that $t_{87} \simeq T_{87}^Y/2$. From equation (6.2), one can see that the tap-delay t_{87}, is calculated as half the difference of two oscillation periods, T_8^Y and T_7^Y respectively. Therefore these oscillation periods must be measured with an accuracy of $\pm$20ps too. As it was stated, $T_8^Y = 25 \times 400$ps, and the accuracy is still $\pm$20ps. Therefore the accuracy to determine the oscillation period T_8^Y should be ($\pm$20ps)/(25$\times$400ps), which is $\pm 0.2\%$.

6.2.3 Block-Scheme for Measuring the Oscillation Period

As mentioned in the previous section, the accuracy of measuring the tap-delays is correlated with the accuracy of measuring the oscillation periods. These periods should also be measured in a digital way if a BIST strategy is envisioned. A possibility for digitally measuring an oscillation period has been presented in [5]. The block scheme used to perform this operation is shown in figure 6.5. The control logic, which among other things takes care

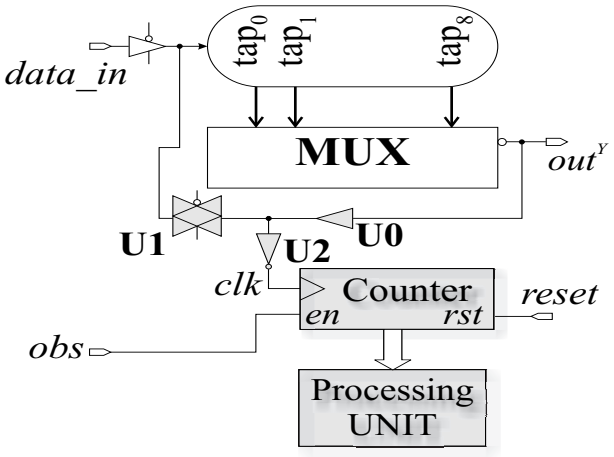


Figure 6.5: Basic scheme for measuring the tap-delays

of the proper selection of the taps, is not shown in this figure for the sake of simplicity. Also, not all signals have been shown in order for the reader to concentrate on the method and not on schematic. Section 6.5 will provide the detailed implementation of the design.

The shaded modules in figure 6.5 are part of the DfT hardware. The rest is the initial design of the delay line. The **U0** buffer and the transmission gate **U1** are added in order to make the circuit oscillate. The waveforms can be seen in figure 6.6, where N is the number stored in the counter.

As long as the enable signal "*obs*" of the counter is high, the number stored in the counter will be increased by one every rising edge of the signal "*clk*". It can be concluded that:

$$T_{obs} \simeq N_k^Y \cdot T_k^Y, \quad k = 0, 1, \ldots, 8 \tag{6.3}$$

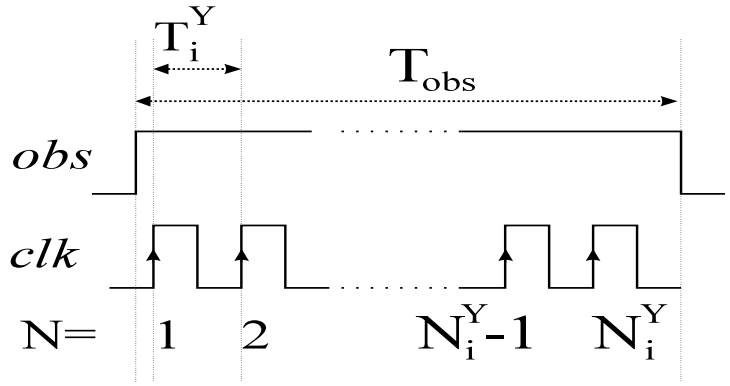


Figure 6.6: The waveforms associated with the scheme in figure 6.5, if tap# i is selected

From equations (6.2) and (6.3), the tap-delay $t_{(i)(i-1)}$ is calculated to be:

$$t_{(i)(i-1)} \simeq \frac{T^Y_{(i)(i-1)}}{2} \simeq T_{obs}\frac{N^Y_{(i-1)} - N^Y_i}{2N^Y_i N^Y_{(i-1)}} \tag{6.4}$$

The processing unit (**PU**) in the scheme of figure 6.5 can be either on-chip or off-chip. Its function is to calculate the tap-delays using the numbers stored in the counter together with the period information about the signal "*obs*".

6.3 The Oscillation-Period Measurement Accuracy

This section deals with the measurement accuracy, using the scheme shown in figure 6.5. This analysis is important, because the subsequent digital synchronization mechanism relies on equidistant tap-delay values within a predefined range. Any delay fault present in the delay-line can disrupt the expected functionality of the synchronization mechanism.

The measurements of all oscillation periods T^X_i and T^Y_i for $i = 0, 1, \ldots, 8$ require two steps:

1. After resetting the counter, it will be enabled by the signal "*obs*". Subsequently, the counter will count the number of rising edges of the oscillation period to be measured.

2. If the signal "*obs*" becomes low, a number will be stored in the counter. This number is used to calculate the particular oscillation period, using for example equation (6.3).

As can be seen from these steps, finding the value of any oscillation period implies in fact a measurement and a calculation. Hence, during the remaining of the chapter the term *measurement* will mean the first step while the term *calculated* will indicate the second step.

The main sources of the measurement error can be divided into four categories:

A. *digital rounding* - due to the fact that the oscillation period (e.g. T_i^Y) and the observation time (T_{obs}) are not correlated.

B. *observation time affected by random jitter*

C. *systematic error when generating the observation time* (T_{obs})

D. *analogue parameters* - like power supply and temperature that can vary during adjacent measurement periods.

These four effects will now be carefully analyzed in the following subsections.

6.3.1 Digital-Rounding Measurement Error

As described earlier, the technique used to measure the oscillation periods $T_i^Y,\ i = 0, \ldots, 8$ is quite simple [5]. The induced periodic signal will be the clock signal for a digital counter. The counter will also have an enable signal which has to be provided externally, and duration of the logic one level will be referred to as *the observation time* (T_{obs}). Because the periodic signal of a ring oscillator is not correlated with the observation time, there will always be a measurement error. This is given by the following equation:

$$T_{obs} = N_i^Y \cdot T_i^Y + u \cdot T_i^Y, \quad i = 0, 1, \ldots, 8 \tag{6.5}$$

where u is a random variable having an uniform distribution between $(-1, 1)$, and N_i^Y is the number stored in the digital counter. N_i^Y represents the number of rising edges during the time when the enable signal "*obs*" of the counter is high. It should be noted that before every oscillation-period measurement, the counter should be reset. In figure 6.7, two extreme situations for u are shown.

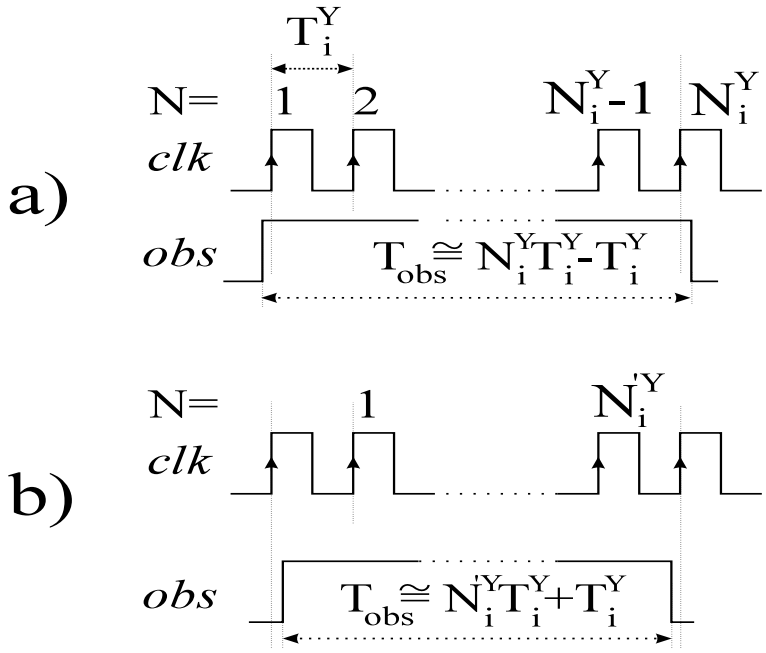


Figure 6.7: Measurement error due to digital rounding - two extreme cases for u. a) u = -1 and b) u = 1

Rewriting equation (6.5) results in:

$$T_i^Y + \frac{uT_i^Y}{N_i^Y} = \frac{T_{obs}}{N_i^Y}, \quad i = 0, 1, \ldots, 8 \tag{6.6}$$

In the previous equation, the value T_{obs}/N_i^Y is our calculated value of T_i^Y, since there is knowledge of T_{obs} and N_i^Y, while the real value is T_i^Y. One may further deduct that the measurement error (me) is uT_i^Y/N_i^Y. Since u is unknown at the time of measurement, the maximum measurement error is T_i^Y/N_i^Y. A simple conclusion could be that by extending the number of bits of the counter, and hence increasing T_{obs}, the measurement error as a result of the digital rounding, can be made as small as desired.

E.g.: Consider measuring an oscillation period that has the nominal value $T_8^Y = 25 \times 400ps$ with an accuracy of $\pm 20ps$. Therefore $20ps \geq T_8^Y/N_8^Y$. Hence $N_8^Y \geq 500$. The observation time should be $T_{obs} \geq 5\mu s$. As can be seen, keeping the oscillation period as small as possible, will also decrease the observation time, while keeping the absolute measurement accuracy the

same. A new delay line scheme will be presented in section 6.4 which will make use of the above observation.

The measurement error cannot be as small as desired if the value of T_{obs} is changing between adjacent oscillation periods measurements. The following subsection will illustrate this.

6.3.2 Measurement Errors Resulting from Jitter

The effect of jitter [7, page 403] on our observation time T_{obs} is illustrated in figure 6.8. In this figure, the peak-to-peak representation of jitter is considered. The random variable j is assumed to have a Gaussian distribution. The $max(|j|)$ value is considered to be at 3σ.

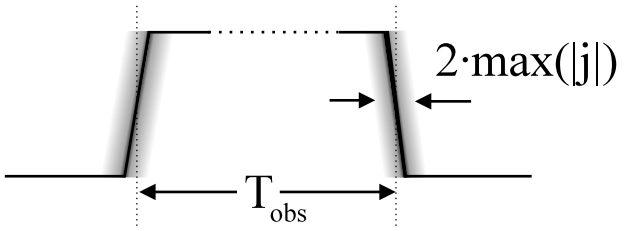


Figure 6.8: Jitter affecting the observation time T_{obs}

It is further considered that j_i^Y is the value of the jitter, if one wants to measure the oscillation period T_i^Y, $i = 0, 1, \ldots, 8$. The value of j_i^Y is measured in seconds, and is within the peak-to-peak jitter range shown in figure 6.8. In other words, j_i^Y is a particular realization of the random variable j.

Therefore, when measuring T_i^Y, the observation time T_{obs} has the following value: $T_{obs} + j_i^Y$. When measuring T_{i-1}^Y, the observation time T_{obs} has the value $T_{obs} + j_{i-1}^Y$. Because of these tiny variations, the measured value of the oscillation periods will be corrupted with a certain error.

The value of the observation time affected by jitter when measuring the oscillation period T_i^Y will be denoted as $T_{obs\,i}^Y$. The values of $T_{obs\,i}^Y$ ($i = 0, 1, ...8$) are different for each measurement but the average value is T_{obs}.

In order to simplify the notation of future relations, the jitter will be con-

verted into percentages of T_{obs}. By definition h_i is defined as:

$$h_i^Y \stackrel{def}{=} \frac{j_i^Y}{T_{obs}} \tag{6.7}$$

Hence, the influence of the jitter on the observation time T_{obs}, when measuring the oscillation period T_i^Y, is given as:

$$T_{obs\,i}^Y = T_{obs}(1 + h_i^Y), \quad i = 0, 1, \ldots, 8 \tag{6.8}$$

The value stored in the counter, as a result of the T_i^Y oscillation period measurement while having an observation time $T_{obs\,i}^Y$, affected by jitter, will be represented by N_i^Y. Rewriting equation (6.5), by replacing T_{obs} with $T_{obs\,i}^Y$ results in:

$$T_{obs\,i}^Y = N_i^Y T_i^Y + uT_i^Y, \quad u \in (-1, 1) \tag{6.9}$$

The value *calculated* for T_i^Y, which will be denoted $T_{i\,calculated}^Y$, is in fact T_{obs}/N_i^Y since T_{obs} and N_i^Y are known.

Therefore $T_{i\,calculated}^Y$, using equation (6.9), can be expressed as:

$$T_{i\,calculated}^Y = \frac{T_{obs}}{N_i^Y} = \frac{T_i^Y T_{obs}}{T_{obs\,i}^Y - uT_i^Y} \tag{6.10}$$

Hence the difference between the value of the *calculated* oscillation period ($T_{i\,calculated}^Y$) and the real value of the oscillation period (T_i^Y) is:

$$\frac{T_{obs}}{N_i^Y} - T_i^Y = T_i^Y \left(\frac{T_{obs} - T_{obs\,i}^Y + uT_i^Y}{T_{obs\,i}^Y - uT_i^Y} \right) \tag{6.11}$$

The expression above represents the error measurement for T_i^Y in seconds. A representation in terms of percentage is much more suggestive. This is obtained by dividing equation (6.11) with T_i^Y. Denoting the result by *'l'*, which represents the measurement error in percentage, it can be concluded:

$$l = \frac{T_{obs} - T_{obs\,i}^Y + uT_i^Y}{T_{obs\,i}^Y - uT_i^Y} \tag{6.12}$$

Rearranging equation (6.12) using (6.8) one finds:

$$l = \frac{-T_{obs}h_i^Y + uT_i^Y}{T_{obs}(1+h_i^Y) - uT_i^Y} \tag{6.13}$$

In the above relation, h_i^Y has in fact an unknown value. *'l'* should be as small as possible and therefore the worst-case value for *'l'* (*'l'* is large), is if h_i^Y is negative and $|h_i^Y|$ is maximum. Denoting with *'h'* the maximum absolute value of $|h_i^Y|$, it can be concluded:

$$l = \frac{T_{obs}h + uT_i^Y}{T_{obs}(1-h) - uT_i^Y} \tag{6.14}$$

Equation (6.14) provides a lot of information about the error of measuring/calculating any oscillation period T_i^Y.

Two limiting cases may be identified:

1. $T_{obs}h \ll uT_i^Y$
   In this case *'l'* may be approximated by $uT_i^Y/(T_{obs} - uT_i^Y)$. This is the particular case discussed in subsection 6.3.1, if there is no jitter in the observation time. In this case *'l'* can be improved (meaning decreased) by increasing T_{obs}. However $T_{obs}h$ should stay less than uT_i^Y. Therefore, for a very low-jitter observation signal "obs", one can obtain a very small measurement error.

2. $T_{obs}h \gg uT_i^Y$
   In this case *'l'* may be approximated by $h/(1-h)$ and this is common for large observation times or a high-jitter observation signal "obs". For small values of *'h'*, and this is always the case, it can be inferred that $l \simeq h$. Hence, the measurement error for the oscillation periods T_i^Y, $i = 0, 1, \ldots, 8$, cannot be better than *'h'*.

The values of the oscillation periods have increasing values, while the absolute accuracy to measure them remains the same. Hence T_8^Y demands the lowest jitter requirements in T_{obs}.

Now consider that one wants to measure the oscillation period $T_8^Y = 25 \times 400ps$ with an accuracy of $\pm 20ps$. This is equivalent of having an accuracy of $\pm 0.2\%$. If the observation time T_{obs} is sufficiently large, then the second limiting case should be considered. In this case, the accuracy of measuring

T_8^Y is $l \simeq h$. Therefore $h = 0.2\%$, which in turn requires a maximum peak-to-peak value of jitter $2 \cdot max(|j|) = 2 \cdot 0.2\% \cdot T_{obs}$ (see figure 6.8 and relation (6.7)). For a T_{obs} value of $10\mu s$, $2 \cdot max(|j|) = 40ns$ (see figure 6.8).

6.3.3 Systematic-Errors

Another source of uncertainty when measuring the oscillation periods, is the systematic error of the generated frequency used as the enable signal ("obs") for the counter in figure 6.5. The T_i^Y values of the oscillation periods are directly proportional with the value of T_{obs}, as can be seen in equation (6.4). Any uncertainty in T_{obs} will directly influence the results. If one defines the percentage of the systematic error of the generated frequency by k, then it can be written:

$$T_{obs\,generated} \stackrel{def}{=} T_{obs}(1+k) \tag{6.15}$$

$T_{obs\,generated}$ is the observation time generated by the pulse generator. Since this section analyzes the systematic error, the factor k is assumed to be constant during all measurements of the taps.

The value calculated for T_i^Y, which has been denoted $T_{i\,calculated}^Y$, is in fact T_{obs}/N_i^Y since the values of T_{obs} and N_i^Y are known. For the moment, it is assumed that the digital rounding in equation (6.5) is zero. This allows us to focus on the measurement error as a result of the systematic error of the observation period. The calculated value of an oscillation period T_i^Y can be written as:

$$T_{i\,calculated}^Y = \frac{T_{obs}}{N_i^Y}, \quad i = 0, 1, \ldots, 8 \tag{6.16}$$

The real value of the oscillation period T_i^Y using equation 6.15 and ignoring the digital rounding, is:

$$T_i^Y = \frac{T_{obs\,generated}}{N_i^Y} = \frac{T_{obs}}{N_i^Y}(1+k) \tag{6.17}$$

The oscillation-periods calculus represents the first step while determining the tap-delay. As was seen earlier, a tap-delay is half the difference of

adjacent oscillation periods. Using relation (6.16), the calculated value for $T^Y_{(i)(i-1)}$, which will be denoted $T^Y_{(i)(i-1)\,calculated}$, is:

$$T^Y_{(i)(i-1)\,calculated} = T_{obs}\left(\frac{1}{N^Y_i} - \frac{1}{N^Y_{i-1}}\right), \quad i = 1, 2, \ldots, 8 \tag{6.18}$$

while the real value of $T^Y_{(i)(i-1)}$ is:

$$T^Y_{(i)(i-1)} \stackrel{def}{=} T^Y_i - T^Y_{i-1} = T_{obs}(1+k)\left(\frac{1}{N^Y_i} - \frac{1}{N^Y_{i-1}}\right) \tag{6.19}$$

From relations (6.18) and (6.19), the percentage measurement error (m) of tap-delay $T_{(i)(i-1)}$ can be calculated:

$$m \stackrel{def}{=} \frac{T^Y_{(i)(i-1)} - T^Y_{(i)(i-1)\,calculated}}{T_{(i)(i-1)\,calculated}} = k \tag{6.20}$$

From relation (6.20), it can be seen that the percentage measurement error m for any tap-delay is the same as the systematic error k when generating the observation period T_{obs}.

The systematic error k can be less than 1% for many pulse generators. Since $m = k$, the systematic measurement error does not significantly influence the tap-delay measurements because the required accuracy is $\pm 5\%$.

All the error sources presented - digital rounding, random jitter and the systematic error when generating the observation time T_{obs}- should be added together in order to find the total oscillation-period measurement error. Out of the three sources of error, the jitter seems to be the most important one because the digital rounding can be made as small as possible, and the error of the generated observation time is usually small and does not significantly affect the final result. The next sections present the influence of the power-supply (VDD) and temperature on the measurement-error.

6.3.4 Analogue Parameter Variations: Power Supply and Temperature Variations

Power-Supply Variations

It is well known that the power supply (VDD) has a direct influence on the oscillation period of inverter-based oscillators. In this subsection, variations of T_i^Y $(i = 0, 1, \ldots, 8)$ with VDD will be investigated. Therefore $T_i^Y = T_i^Y(VDD)$. Expanding T_i^Y using the Taylor series results in:

$$T_i^Y(VDD_i^Y) = T_i^Y\big|_{VDD_{i-1}^Y} + \left(VDD_i^Y - VDD_{i-1}^Y\right) \left.\frac{dT_i^Y}{dVDD}\right|_{VDD_{i-1}^Y} + err_{VDD} \quad (6.21)$$

where:

1. VDD_{i-1}^Y is the power-supply at the moment the previous oscillation period T_{i-1}^Y is being measured.
2. VDD_i^Y is the power-supply when the oscillation period T_i^Y is being measured.
3. err_{VDD} is the error resulting from the Taylor expansion. This error is zero for a linear dependence of T_i^Y with VDD.

As can be noticed, only the first derivative has been included in the Taylor expansion since the oscillation period has approximately a linear relation with VDD for small deviations. Hence, err_{VDD} can be considered $\simeq 0$. This can be seen in figure 6.9, where i=8.

The second term in equation (6.21) represents the drift of the oscillation period due to the power-supply variation during adjacent measurements. This term can be seen as a measurement error for T_i^Y, which should be smaller than an accepted error $t_{accepted\,error}$ as stated below:

$$\left| \left(VDD_i^Y - VDD_{i-1}^Y\right) \left.\frac{dT_i^Y}{dVDD}\right|_{VDD_{i-1}^Y} \right| \leq t_{accepted\,error} \quad (6.22)$$

From relation (6.22), the restriction on VDD variations between adjacent measurement periods can be derived. This is presented in equation (6.23).

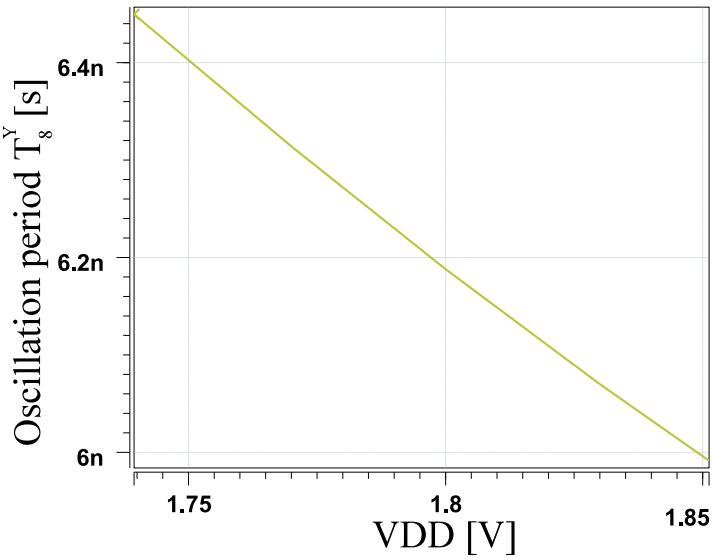


Figure 6.9: Variations of the T_8^Y oscillation period with the power-supply (VDD) - simulation results

$$|\Delta VDD| \leq \left| \frac{t_{accepted\,error}}{\frac{dT_i^Y}{dVDD}\Big|_{VDD_{i-1}^Y}} \right| \tag{6.23}$$

where $\Delta VDD = VDD_i^Y - VDD_{i-1}^Y$.

The value of $t_{accepted\,error}$ is derived from initial requirements.

Consider $t_{accepted\,error} = 10ps$ and $VDD_7^Y = 1.8V$. From the simulation in figure 6.9, one can deduce that $\left| dT_8^Y/dVDD|_{VDD_7^Y=1.8V} \right| = 4 \times 10^{-9}[s/V]$. Therefore, $|\Delta VDD\ | \leq 2.5mV$.

As can be noticed, the allowed VDD variation for a 10ps measurement error is small. In section 6.4, a new scheme is presented which relaxes the VDD requirement. However, a voltage-stabilized source is still needed.

Temperature Variations

The temperature effect on the oscillation period can be seen in figure 6.10.

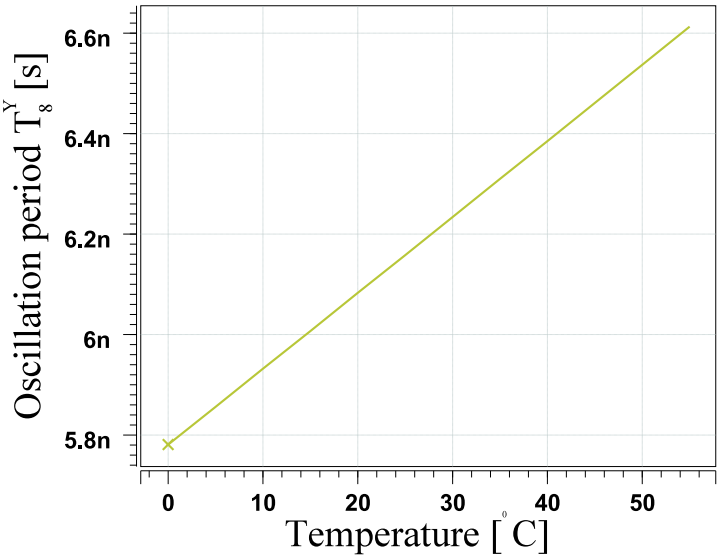


Figure 6.10: Variations of the T_8^Y oscillation period with the temperature - simulation results

Expanding again T_i^Y using the Taylor series one obtains:

$$T_i^Y(Temp_i^Y) = T_i^Y\big|_{Temp_{i-1}^Y} + \left(Temp_i^Y - Temp_{i-1}^Y\right) \left.\frac{dT_i^Y}{dTemp}\right|_{Temp_{i-1}^Y} + err_{Temp} \tag{6.24}$$

where:

1. $Temp_{i-1}^Y$ is the chip temperature while the oscillation period T_{i-1}^Y is being measured.
2. $Temp_i^Y$ is the chip temperature at the moment the oscillation period T_i^Y is being measured.
3. err_{Temp} is the error resulting from the Taylor expansion. This error is zero for a linear dependence of T_i^Y with temperature.

In this case, err_{Temp} is very small since the oscillation period varies almost linearly with the *temperature*, as can be seen in figure 6.10.

From equation (6.24) the restriction on *Temp* variations between adjacent period measurements is derived. This is presented in equation (6.25):

$$\Delta Temp \leq \frac{t_{accepted\,error}}{\left.\frac{dT_i^Y}{dTemp}\right|_{Temp_{i-1}^Y}} \tag{6.25}$$

where $\Delta Temp = Temp_i^Y - Temp_{i-1}^Y$.

Looking at figure 6.10, $dT_8^Y/dTemp|_{Temp_7^Y} = 1.91 \times 10^{-11}[s/^oC]$ and let us assume $t_{accepted\,error} = 10ps$. It can then be calculated that $\Delta Temp \leq 0.52^oC$.

For the sake of completeness, the effect of both power supply (VDD) and temperature (Temp) variations can be combined and expressed in equation (6.26):

$$\begin{aligned} T_i^Y(Temp_i^Y, VDD_i^Y) &= T_i^Y\big|_{Temp_{i-1}^Y, VDD_{i-1}^Y} + \\ &+ \left(Temp_i^Y - Temp_{i-1}^Y\right) \left.\frac{\partial T_i^Y}{\partial Temp}\right|_{Temp_{i-1}^Y, VDD_{i-1}^Y} + \\ &+ \left(VDD_i^Y - VDD_{i-1}^Y\right) \left.\frac{\partial T_i^Y}{\partial VDD}\right|_{Temp_{i-1}^Y, VDD_{i-1}^Y} + err \end{aligned} \tag{6.26}$$

Parameter *err* is the introduced error resulting from the Taylor expansion and is smaller than the sum of the two previously defined errors err_{VDD} and err_{Temp}. As can be seen, the temperature and power-supply variations during measurements play a vital role when determining the oscillation periods. If the power supply and/or temperature variations are too high for the accepted measurement tolerance, then a redesign of the delay line should be carried out. A possible implementation of such a delay line is presented in section 6.4.

6.3.5 Stuck-at/Delay Fault Debugging Capabilities Using the Oscillation Technique

In figure 6.11 the values of the oscillation periods are shown on the horizontal axis if no faults are present.

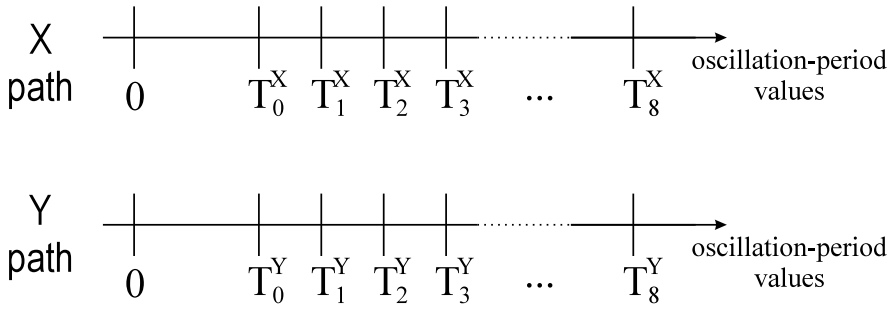


Figure 6.11: Oscillation period values if no fault is present

It can be seen in figure 6.11 that the difference between T_i^Y and T_{i-1}^Y, for $i = 1, 2, \ldots, 8$ is always the same. Next, different locations for the delay faults will be considered. They may appear either in the intrinsic buffer paths or in the path of the inverters and the tri-state buffers as shown in figure 6.2.

Delay Faults in the Intrinsic Buffer Paths

Any delay fault that may appear in the intrinsic delay-line path, corresponding with our notations $t_{(i)(i-1)}$, for $i = 1, 2, \ldots, 8$, will have an effect similar to the one as presented in figure 6.12.

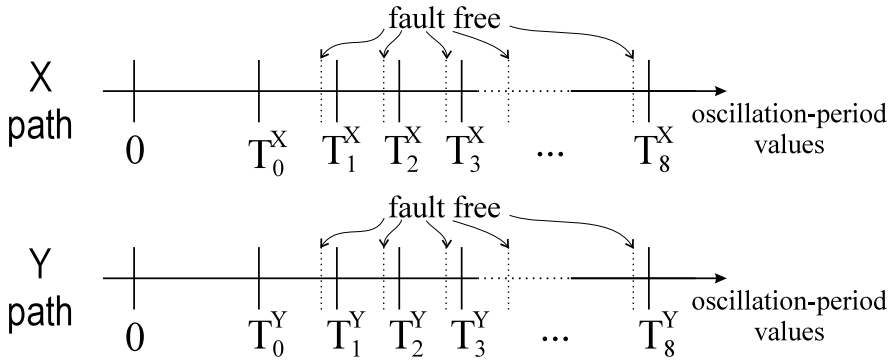


Figure 6.12: A delay fault present in t_{10} increases all oscillation-period values

As can be seen, all the T_i^Y marks after the delay fault occurs, are shifted to the right. The dotted lines represent the values of the oscillation periods if no faults are present. For such a delay-fault location, only one difference

of type $T_i^Y - T_{i-1}^Y$ is different from the others, and should appear in both paths: Y and X at the same tap number. The delay-fault location in the intrinsic tap-delay can therefore be pinpointed.

Delay faults in the inverters or the tri-state buffers

Any delay fault in the inverters or in the tri-state buffers, as shown in figure 6.2, corresponding with our notations t_i^Y or t_i^X, $i = 0, \ldots, 8$ will shift only a certain T_i^Y or T_i^X mark to the left or right depending on the delay as can be seen in figure 6.13. For such a delay-fault, two differences, of type

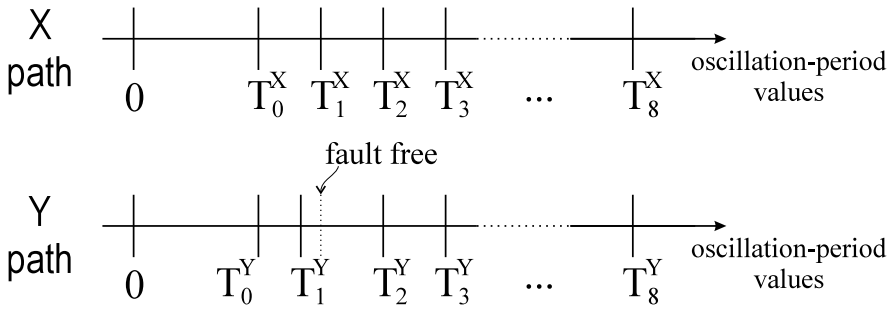


Figure 6.13: The effect of a smaller t_1^y delay on the oscillation period values

$(T_i^Y - T_{i-1}^Y)$ or $(T_i^X - T_{i-1}^X)$, will differ from the rest, and they should appear either in path Y or X.

Considerations with regards to fault masking and multiple delay faults

If one considers that only a single delay-fault may appear, then that delay fault can be identified as explained earlier.

If more than one delay fault appears, the problems are getting more complicated. Basically, the calculated tap delays will indicate were the faults are, by just comparing them. This is possible since a lot of taps are available.

The stuck-at-faults are easily detected since the oscillations cease to exist if such types of faults occur.

Any type of delay-fault or stuck-at-fault can be identified using the procedures presented above. Since the number of taps is usually high (in our

case 18), more than one stuck-at or delay faults may be detected and their location identified. If the "one delay-fault" model is used, then there is no fault masking. However if more than one delay-fault occurs, then the delay-fault identification procedure can provide a false indication.

Let us consider an increased delay in t_{21} (figure 6.2) with the same size as the decrease in t_2^x. The delay between tap 1 and tap 2 in path X will therefore appear to be unaffected.

6.3.6 Gate Delays in the Functional and Test Mode

Our basic assumption at the beginning of the chapter was that the delays of each gate are always the same in both modes, being functional and test mode. Sometimes this is not true. There are two major factors that can influence the mismatch:

1. Different node capacitances in test and functional mode.
2. The absence of a **full-swing** periodic signal at a certain node.

All of these error sources will be treated in the following subsections.

Different Capacitances in Test and Functional Mode

In figure 6.14, a simple inverter is presented together with some ideal waveforms. It is considered that the output load of the inverter is always the same. The propagation gate delays of the inverter, measured between the 50% points of the power supply, are shown in the same figure.

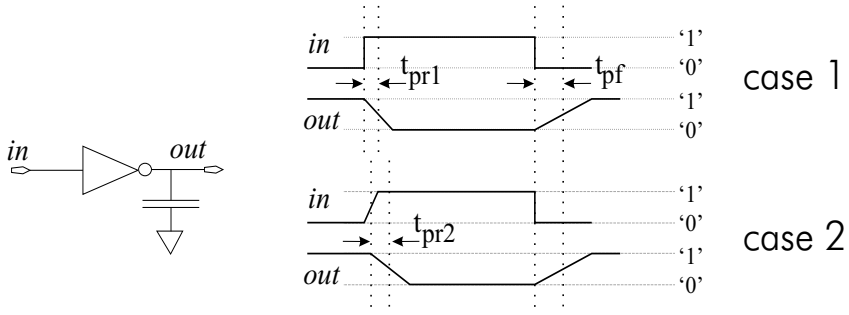


Figure 6.14: Rising/Falling propagation delays of an inverter. case 1) fast rising input signal; case 2) slow rising input signal

It is known that the propagation delay of any gate is highly dependent on the output load. However, the delay of a gate is also dependent on the rising/falling time of the input signal which is dependent on the input load. This effect can be seen in figure 6.14, where the rising propagation time in case 1 is different than the rising propagation time in case 2 ($t_{pr1} \neq t_{pr2}$).

Because the propagation delays of a certain gate are dependent on both input and output loads, it is compulsory to keep these loads the same in both, functional and test mode. If this is not entirely possible then it is required to have the same capacitances when measuring adjacent oscillation periods. As will be seen in section 6.4, the new scheme presented in figure 6.17 is well suited from this point of view.

Even if the *net1* node, in figure 6.17, has different capacitances associated with it in functional and test mode, when measuring the tap-delay 87 (t_{87}), its influence on the measurement of t_{87} is zero. This is because the affected gate delays are not used when calculating t_{87} (see relation 6.2).

In functional mode, in order to have the same capacitance on line "out^X" ("out^Y"), a dummy delay cell plus a dummy transmission gate should be added. The dummy transmission gate should be well matched with the shaded transmission gates in figure 6.17. The same should hold for the dummy delay cell. Any layout mismatch will influence the measurement of all the tap-delays. However, a small influence on the final measurement results is anticipated, since the load at the out^X line is quite high compared to any mismatch difference.

Absence of a Full-Swing Periodic Signal at a Certain Node

If the load on a certain net is very high, for example the load on the "out^X" line, then the oscillation signal on that net will not reach the positive power supply or/and the ground. A typical situation is illustrated in figure 6.15. The signal "out^X" starts changing (see the overshoot) before reaching the power supply (1.8V).

If this situation occurs, the real gate-delay is slightly higher than the one considered in the general equation (6.1) for the oscillation period. Fig. 6.16 shows this phenomenon using ideal waveforms.

It can be noticed that the falling propagation time in case 2 (t_{pf2}) is smaller than the falling propagation time in case 1 (t_{pf1}) because the output ca-

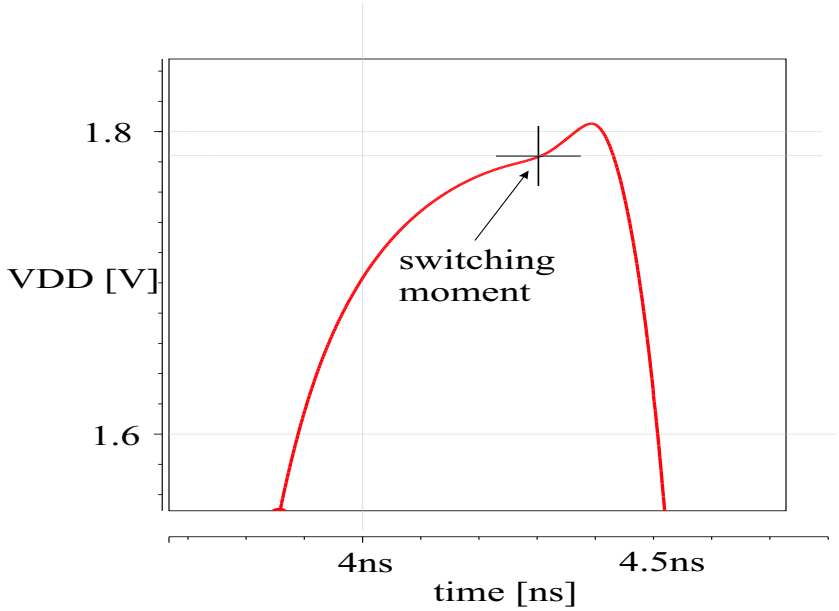


Figure 6.15: Simulated waveform for signal "out^X"

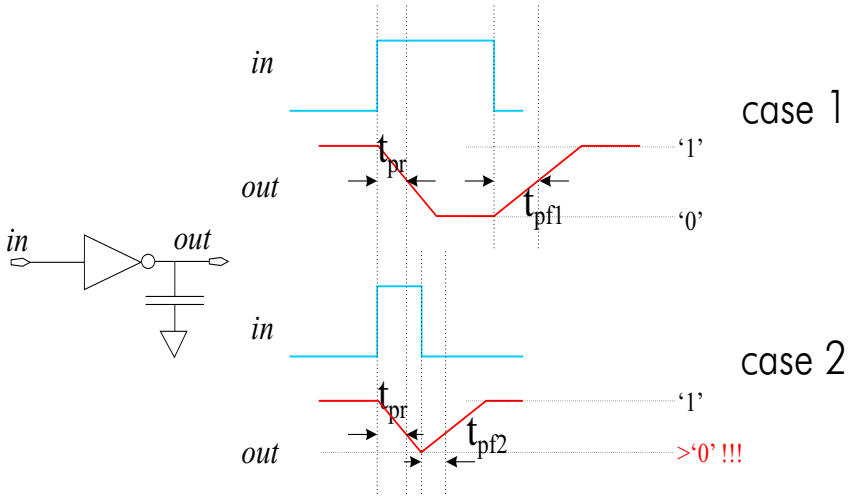


Figure 6.16: Different falling propagation delays due to high load and smaller oscillation periods

pacitance was not completely discharged. This situation should be avoided either by decreasing the load on the problematic node or by adding more gates in the oscillation rings in order to allow sufficient time for the node to settle. All the delays are measured at half the power supply.

6.4 A New Delay-Line Design Comprising Additional DfT Hardware

In this section, a new delay-line design is presented which is more suitable for applying the test-oscillation technique. As was seen in the previous sections (6.3.4 and 6.3.1), the main disadvantage of the initial design is the increase of the oscillation period while the absolute value of the measurement error remains the same. The new delay-line design is able to connect any feed-back signal anywhere within the delay-line. This has been realized by inserting additional transmission gates as can be seen in figure 6.17. The shaded components in the figure are the newly inserted transmission gates.

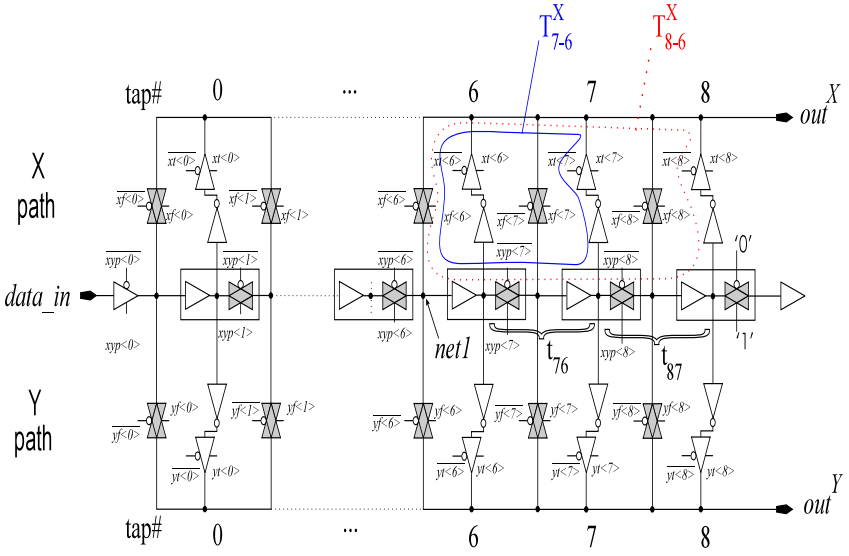


Figure 6.17: A new delay-line scheme for the reduction of the oscillation periods

In the scheme, two new oscillation loops T_{7-6} and T_{8-6} are illustrated. Oscillation loop T_{8-6} is generated by selecting tap number eight (signal "$xt<8>$" is logic one) and closing the loop in front of tap# 6 (signal

"*xf<6>*" is logic one). At the same time the transmission gate in front of tap# 6, controlled by signal "*xyp<6>*" should be disabled ("*xyp<6>*" is logic zero). The original delay-line shown in figure 6.2, did not have this capability of closing the oscillation loop within the delay-line.

This particular configuration is used for determining the tap-delay t_{87} (in fact T_{87}^X). Since shorter oscillation periods can be generated for measuring the tap-delays, the values $dT_i^Y/dVDD$ and $dT_i^Y/dTemp$ in equations (6.21) and (6.24) respectively, are smaller. This is equivalent to state that the susceptibility to power-supply and temperature variations has decreased.

6.4.1 Comparison Between the Initial and the New Delay-Line

It should be mentioned that the disadvantages for the new scheme are a higher area-overhead (around 35%) and increased tap-delays (around 50%) as compared to the original design presented in figure 6.2. However, the synchronization mechanism is able to cope with increased tap-delay as long as they are uniformly distributed among the taps and within a certain predefined range which is application specific. For the new scheme in figure 6.17, the signal "*obs*" (figure 6.5) can have the same period for all oscillation-period measurements. This significantly simplifies its generation procedure. By decreasing the oscillation periods, the test time is also reduced. This is because the observation time, which is proportional to the test time, is quadratically dependent on the oscillation period, $T_{obs} \sim (T_i)^2$, for a certain measurement accuracy. In section 6.3.1 the digital-rounding measurement error (me) is calculated to be T_i^Y/N_i^Y where T_i^Y is the on-chip oscillation. The observation time T_{obs} is equal with $N_i^Y \cdot T_i$. Therefore, the observation time T_{obs} is $(T_i)^2/me$.

The new scheme has also better fault-debugging capabilities since more than one measurement combination is possible for testing the same tap-delay.

Table 6.1 shows a comparison between the initial and the new delay-line design for an accepted measurement error of $t_{accepted\ error} = 10ps$. The values in the table have been derived from simulations in HSPICE [10]. It can be seen that the susceptibility to the power supply and temperature has decreased. The test time in table 6.1 does not include the time for testing the overhead as a result of the control logic. It only represents

Table 6.1: Comparison between the initial and the new delay-line

$t_{accepted\ error} = 10ps$	New Design	Initial design
ΔVDD	4,3mV	2,5mV
ΔTEMP	$1,65^oC$	$0,52^oC$
Test time for determining t_{87}	1.8us	4.3us

the observation time for a certain measurement accuracy. It should be noted that for determining each tap delay, two measurements should be performed for the new delay-line. This is in contrast to the initial delay-line presented in figure 6.2, where only one measurement is required because the previous oscillation period measurements are reused. With the new delay-line presented in figure 6.17 it is possible to measure two oscillation periods in parallel, one in each path. For example it is possible to carry out the measurements of T^Y_{8-6} and T^X_{4-2} in parallel.

The values of ΔVDD and ΔTEMP in table 6.1 are quite small. The following paragraphs will prove that these constrains can be met in practice.

6.4.2 The ΔVDD requirement

Corner simulations have been carried out in order to find the maximum difference in current between adjacent measurements drained from the power supply. This value is well within 10mA for a $0.18\mu m$ UMC technology. Knowing that the maximum ΔVDD allowed should be less than 4.3mV, it can be concluded that the output dynamic resistance of the power supply should be less than 430 mΩ.

Today, there are commercial voltage references which can provide a value almost 10 times lower than the one required above. However, the resistance of the power-supply metal lines are determined by their layout, together with the contact resistance and the output resistance of the voltage reference and should therefore not exceed 430 mΩ.

6.4.3 The ΔTEMP requirement

Table 6.1 also presents the maximum temperature deviation allowed for a measurement error of 10ps. These values are derived from simulations. Because the oscillation frequency of the ring oscillators is quite high, around

0.5GHz, the self-heating of the IC could influence the measurements. Running two measurements in parallel, at different taps for both path X and Y, the maximum current consumption obtained in the worst case was below 15mA. Calculation of the necessary *Junction-to-Ambient* (Θ_{JA}) parameter, in order not to have a higher ΔTEMP as a result of self-heating, was carried out. The value of Θ_{JA} should be lower than 59.26 $^oC/W$. Most of the available packages can provide this value.

Usually other tests, like full-scan, memory BIST, logic BIST, etc., are also carried out as part of the testing strategy. The tap-delay test strategy based on the oscillation technique should be employed first in order not to overheat the IC and thus obtain unusable results. If all taps have been measured, then all other tests can be carried out.

6.5 Design Implementations for the Delay-Line and its DfT

The new delay-line scheme presented in figure 6.17 together with additional DfT hardware similar to the one presented in figure 6.5 has been implemented in a real design. The following subsections, 6.5.1 and 6.5.2 will present the details of the "DLLINE" chip implementations.

6.5.1 Delay-Line Scheme

The top level of the delay-line scheme and the associated DfT hardware is presented in figure 6.18.

Instance **I8** represents the TAP controller of the IEEE standard 1149.1-2001 [11]. Via this TAP controller it is possible to select different taps and feedback loops in the delay-line. The TAP controller will also collect the results from the digital counter and serialize them out via the "*TDO*" signal.

In figure 6.18, instance **I2**, called dlTopLevel, is the delay-line as presented in figure 6.17. The control signals, for selecting the taps of the path X in test mode are "*xt<0:8>*". The feedback selection of the taps is carried out by the signals "*xf<0:9>*". One may notice an extra signal "*xf<9>*" as compared to the delay line of figure 6.17. This is used in functional mode and has not been shown in figure 6.17. It has no influence on the

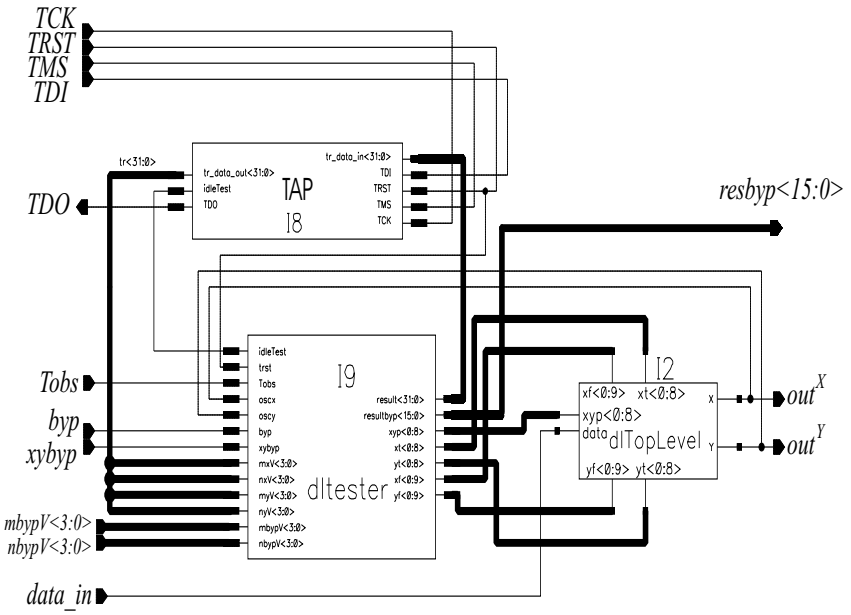


Figure 6.18: Top-level implementation of the "DLLINE" chip

proposed testing method and can be ignored. Signals "*xyp<0:8>*" control the transmission gates in front of the intrinsic buffers (see figure 6.17).

In figure 6.18, together with the delay-line (**I2**) and the TAP controller (**I8**), there is also a block called dltester (**I9**). This block takes care of decoding the signals "*nbypV<0:3>*" and "*mbypV<0:3>*" (explained later) in the control signals "*xt<0:8>*", "*xf<0:9>*" and "*xyp<0:8>*" of the delay-line.

The complete design has two testing modes being "TAP mode" and "backup mode". During the TAP mode, control signals of the delay-line are scanned-in via the "*TDI*" input and the results of the counter are scanned-out via the "*TDO*" output. The backup mode has been provided, as the name suggests it, as a backup solution in case the TAP mode is faulty. The backup mode is activated by providing a logic one in the "*byp*" signal. The signal "*xybyp*" is used to select which path, either X or Y, is tested during bypass mode. The two paths cannot be tested in parallel due to the chip-limited number of pins (48). In backup mode, the delay-line is controlled by "*nbypV<0:3>*" and "*mbypV<0:3>*" input signals via the **dltester** block. The **dltester** block takes care that no conflicting combinations can be ap-

plied to the delay-line, which can damage the design by having two drivers of different values (zero and one) on the same net. The results of the internal counter that is counting the number of rising edges of the internal self-generated ring-oscillations can be observed outside via the signal "*resbyp<15:0>*".

6.5.2 The Delay-Line Layout

The scheme presented in figure 6.18 has been implemented in an UMC $0.18\mu m$ CMOS technology [19].

The layout of the design is shown in figure 6.19. The real implementation of the chip has been called "DLLINE". The top view in figure 6.19 is the layout in the "Virtuoso Layout Editor" of CADENCE [20]. The bottom view of the layout figure is a picture taken after the chip has been manufactured.

The core area of the chip is 0.3mm×0.4mm. The total chip area (including pads) is 1.5mm×1.5mm.

6.6 Tap-Delay Measurements

The "DLLINE" chip measurements have been carried out using a Logic Master ATS1 test system [6].

Figure 6.20 shows a screen dump of a basic test. After the reset signal "*TRST*" is set to one in sequence number 18, the internal test structure is activated. Following two falling edges of the signal "*Tobs*", the results on the "*resbyp<15:0>*" bus signal can be read out. In this particular case, tap number six is selected ("*nbypV*"=6) and its signal is fed back to tap number one ("*mbypV*"=1). Please note that the LSBs for the binary signals "*mbypV*" and "*nbypV*" are at the left in figure 6.20. The signal "*resbyp<15:0>*" is displayed in decimal form in the same figure. The cycle time for the measurements is $4.992\mu s$. The counting happens only once when the "*Tobs*" signal is one. The internal generated ring-oscillation is clocking the internal counter 3676 times during an interval of $4.992\mu s \cdot 4 = 19.968\mu s$ (T_{obs} is logic one for four cycles). The expected column in figure 6.20 should be ignored since simulation results have not been entered in the test system. Hence the oscillation frequency can be calculated as $19.968\mu s/3676 = 5.432ns$.

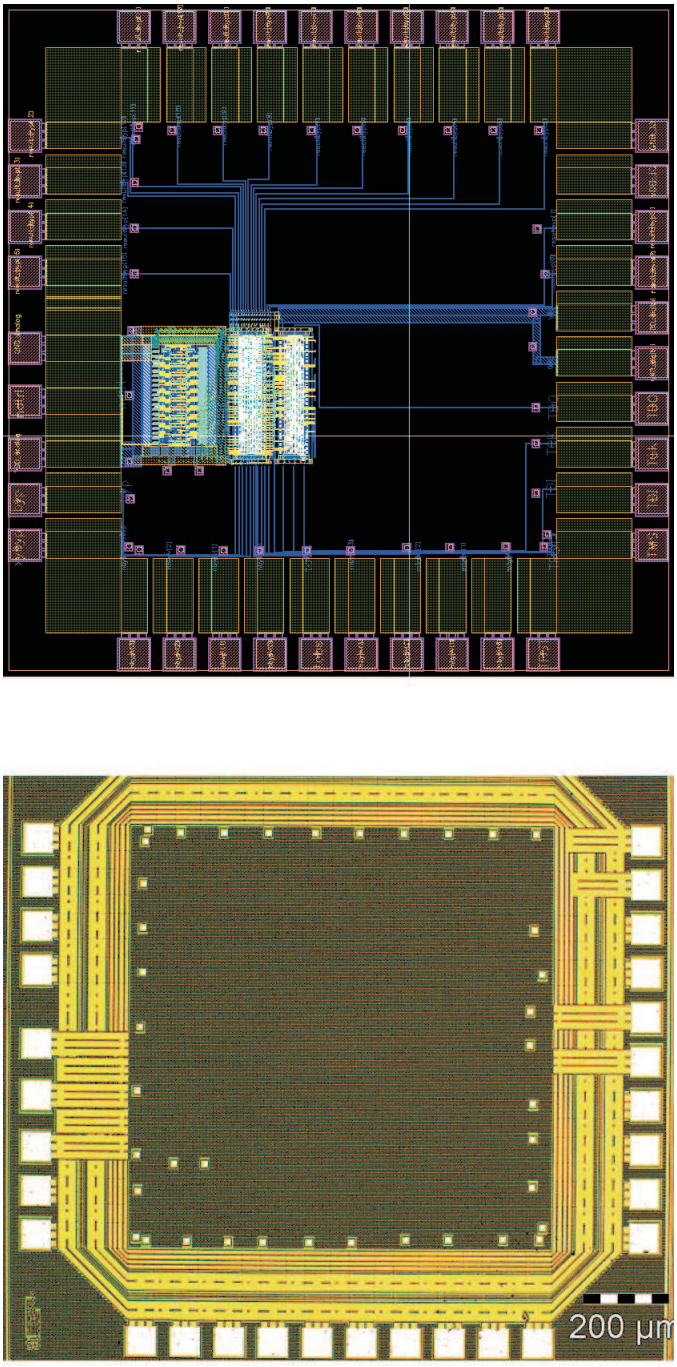


Figure 6.19: "DLLINE" chip layout; top: CADENCE layout; bottom: manufactured chip

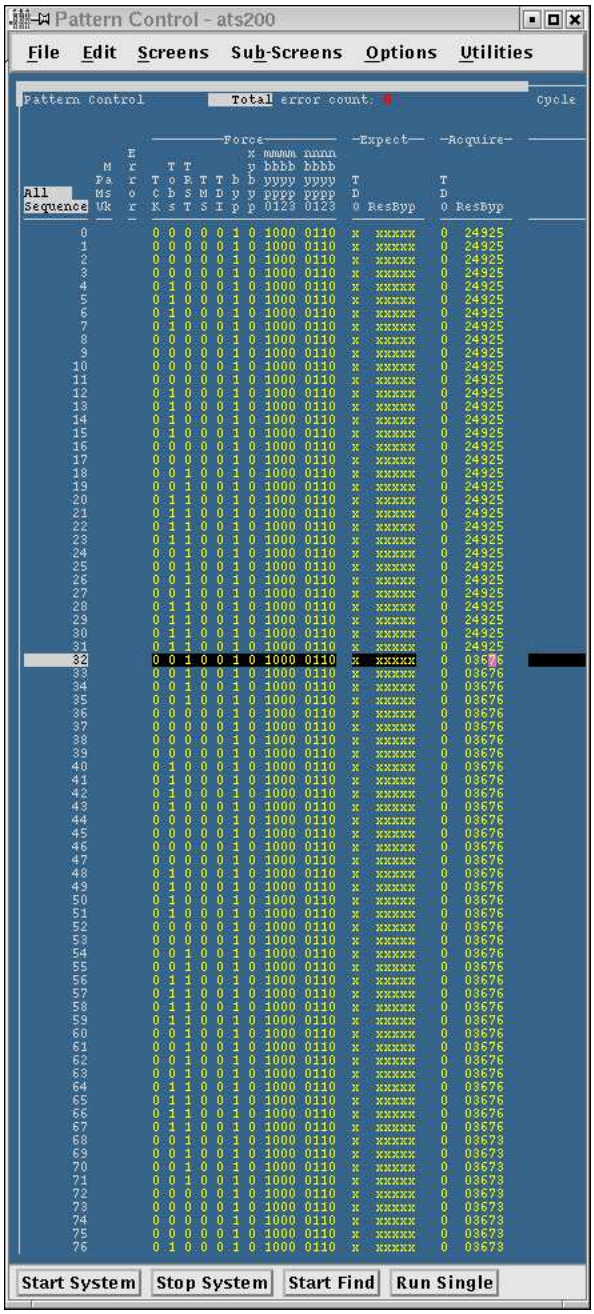


Figure 6.20: IMS screen during an oscillation measurement

Many measurements, as the one presented above, have been performed for determining the delays between taps. The delay between tap 7 and 8 has been chosen to prove that the measurement accuracy is equal or better than what has been presented in section 6.3. The reason for selecting this delay is that it permits the maximum number of measurements (7) for its determination. One of the many ways in which the delay t_{87} can be measured has been shown in figure 6.17. In this figure, tap# 6 has been chosen as feedback. The following steps have been carried out in order to determine the accuracy of the measured delay between tap 7 and 8:

- Oscillation periods have been calculated based on the number present in the internal digital counter for the following setups:
  - tap 8 is selected and fed back in tap 0, 1, 2, 3, 4, 5, 6 and 7
  - tap 7 is selected and fed back in tap 0, 1, 2, 3, 4, 5, 6 and 7

  For each of the above setups, 160 values of the number stored in the counter have been recorded. Therefore, 2560 values have been considered.

- For each feedback tap (0, 1, ..., 7), the values of the delay between tap 7 and 8 have been calculated using relation 6.4.

All of the above values, together with their associated errors are shown in figure 6.21 for path X and in figure 6.22 for path Y.

On the horizontal axis, the tap number used for measuring the tap delay between tap 8 and 7 is shown in figures 6.21 and 6.22; on the vertical axis, the value of the delay between tap 8 and 7 is given, together with the associated measurement error. On the same axis, the minimum and maximum of the accepted error of $\pm 10ps$ can also be seen. As one can notice, all the measurements (with the exception of the measurement via tap 1) are within the accepted tolerance. It can also be seen that the measurement error is improving if the oscillation is smaller as has been predicted in section 6.4. Another effect that can be seen in figures 6.21 and 6.22 is the 'false' increased value of the delay between tap 7 and tap 8 if very short oscillation periods are used (tap 7 is used for measurements). This effect has been analyzed in subsection 6.3.6 (page 133) and the measurements confirm it.

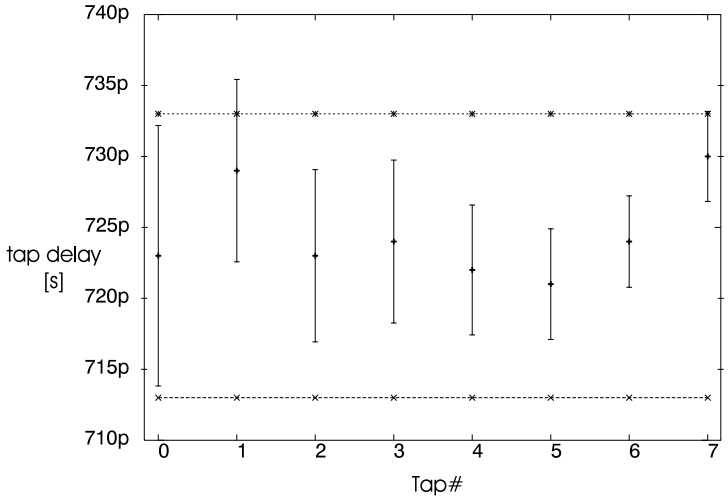


Figure 6.21: Measured values for the delay between tap 8 and tap 7 for path *X*

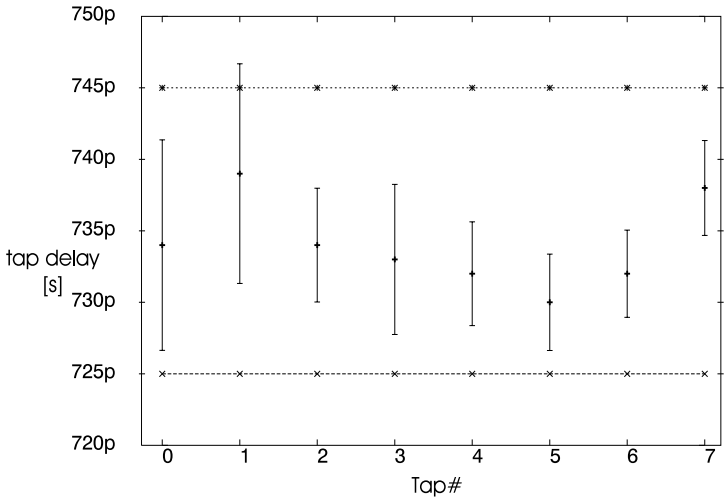


Figure 6.22: Measured values for the delay between tap 8 and tap 7 for path *Y*

6.7 Conclusions

A new method for measuring the tap-delays of a digital delay line used for high-speed data synchronization has been presented. By measuring the tap delays, it is possible to pinpoint where possible delay-faults and/or stuck-at-faults are located. It has also been shown which design equations are to be considered in order to determine the measurement accuracy of the tap-delays. These equations can be extended for other oscillation-based testing techniques which use a digital counter for measuring the oscillation period. In order to obtain a higher measurement accuracy, a new delay-line design was presented, which is capable of reducing the test time and the influence of analogue parameters like power-supply and temperature variations. The design of the new delay-line together with its associated DfT hardware has been implemented in an UMC $0.18\mu m$ technology. By measuring and then calculating the tap delays it has been shown that the measurement accuracy can be within $\pm 10ps$. The results are important since it has been proven that the proposed method can be used to accurately measure the tap delays of any delay-line.

Bibliography

[1] Donnie Anderson et al. The 68040 32-b Monolithic Processor. *IEEE Journal of Solid-State Circuits*, 25(5):1178–1189, October 1990. 6

[2] Karim Arabi, Hassan Ihs, Christian Dufaza, and Bozena Kaminska. Digital oscillation-test method for delay and stuck-at fault testing of digital circuits. *International Test Conference (ITC)*, pages 91–101, 1998. 6.1.3, 6.2, 6.2.1

[3] Karim Arabi and Bozena Kaminska. Oscillation-test strategy for analog and mixed-signal integrated circuits. *IEEE VLSI Test Symposium (VTS)*, pages 476–482, 1996. 6.2, 6.2.1

[4] Karim Arabi and Bozena Kaminska. Oscillation Built-In Self Test (OBIST) Scheme for Functional and Structural Testing of Analog and Mixed-Signal Integrated Circuits. *International Test Conference (ITC)*, pages 776–786, 1997. 6.1.3, 6.2, 6.2.1

[5] Karim Arabi and Bozena Kaminska. Method of dynamic on-chip dig-

ital integrated circuit testing. Patent No. US 6223314, April 2001. 6.2.3, 6.3.1

[6] ATS200. *IMS Logic Master ATS 1.* Credence acquired Integrated Measurement Systems (IMS) - Credence Systems Corporation 215 Fourier Avenue Fremont, CA 94539. `http://www.ims.com/`. 6.6

[7] William J. Dally and John W. Poulton. *Digital Systems Engineering.* ISBN 0-521-59292-5 (hb). Cambridge University Press, 1998. 6.3.2

[8] Avner Efendovich, Yachin Afek, Coby Sella, and Zeev Bikowsky. Multifrequency zero-jitter delay-locked loop. *IEEE Journal of Solid-State Circuits*, 29(1):67–70, January 1994. 6, 6.1.1

[9] Bruno W. Garlepp, Kevin S. Donnelly, Jun Kim, et al. A portable digital DLL for high-speed CMOS interface circuits. *IEEE Journal of Solid-State Circuits*, 34:632–644, May 1999. 6.1.1

[10] HSPICE. *Golden Accuracy Circuit Simulator.* Synopsys, Inc. 700 East Middlefield Rd. Mountain View, Ca. 94043. `http://www.synopsys.com/products/mixedsignal/hspice/hspice.html`. 6.4.1

[11] JTAG1149.1. *Boundary Scan Testing Standard.* IEEE. `http://grouper.ieee.org/groups/1149/6/`. 6.5.1

[12] Mark G. Johnson and Edwin L. Hudson. A variable delay line PLL for CPU-coprocessor synchronization. *IEEE Journal of Solid-State Circuits*, 23(5):1218–1223, October 1988. 6.1.1

[13] Beomsup Kim, David N. Helman, and Paul R. Gray. A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2-μm CMOS. *IEEE Journal of Solid-State Circuits*, 25(6):1385–1394, December 1990. 6

[14] Sungjoon Kim, Kyeongho Lee, Deog-Kyoon Jeong, and Yunho Choi. A pseudo-synchronous skew-insensitive I/O scheme for high bandwidth memories. *Symposium on VLSI Circuits Digest of Technical Papers*, pages 41–42, June 1994. 6, 6.1.1

[15] LVDS. *Low-voltage differential signaling.* The American National Standards Institute (ANSI)/Telecommunications Industry Association (TIA)/Electronic Industries Alliance (EIA)-644-1995 standard. `http://www.iec.org/online/tutorials/low_voltage/`. 6.1.1

[16] Yasunobu Nakase, Yoshikazu Morooka, David J. Perlman, et al.

Source-synchronization and timing vernier techniques for 1.2-GB/s SL-DRAM interface. *IEEE Journal of Solid-State Circuits*, 34:494–501, April 1999. 6, 6.1.1

[17] Stefanos Sidiropoulos and Mark Horowitz. A semi-digital DLL with unlimited phase shift capability and 0.08-400 MHz operating range. *IEEE Solid-State Circuits Conference; Digest of Technical Papers*, pages 332–333, February 1997. 6.1.1

[18] Stefanos Sidiropoulos and Mark A. Horowitz. A semidigital dual delay-locked loop. *IEEE Journal of Solid-State Circuits*, 32(12):1683–1692, November 1997. 6, 6.1.1

[19] UMC. *Semiconductor foundry.* UMC, No. 3, Li-Hsin 2nd Road, Science Industrial Park, Hsinchu City, Taiwan, R.O.C., Email: foundry@umc.com. `http://www.umc.com/`. 6.5.2

[20] VIRTUOSO. *Layout Editor.* CADENCE - corporate headquarters, 2655 Seely Avenue, San Jose, CA 95134. `http://www.cadence.com/products/custom_ic/veditor/index.aspx`. 6.5.2

[21] Tsutomu Yoshimura, Harufusa Kondoh, Yoshio Matsuda, and Tadashi Sumi. A 622-Mb/s Bit/Frame Synchronizer for High-Speed Backplane Data Communication. *IEEE Journal of Solid-State Circuits*, 31(7): 1063–1066, July 1996. 6

Chapter 7

Conclusions

7.1 Summary and Conclusions

In the first chapter the trends in SoC design have been outlined. It has been indicated that the fully synchronous design style will have to be adjusted in order to cope with the emerging deep-submicron technologies. Future digital SoCs will be comprised of a large number of complex digital cores. Asynchronous and synchronous designs will slowly but certainly find their way in mainstream digital SoC design. Testing will be a crucial issue for future SoCs. Test strategies for synchronous cores are already available, while for the asynchronous cores they are now being developed. Hierarchical test will be the natural trend in testing.

The interface between these cores always requires small modules called synchronizers, which facilitate low-latency data communication between the cores. Their designs are not trivial and their test strategies are not straight forward. Therefore, they were usually bypassed in test mode, resulting in a non-optimal fault coverage of the entire SoC. Despite the fact that the fault coverage gain is not dramatic, due to the small silicon area occupied by these devices, the synchronizers are a crucial component for the correct operation of the whole SoC.

This thesis has presented test strategies for *synchronizers* between on-chip digital cores being either synchronous or asynchronous. Scan-test solutions have been implemented and they have been augmented with built-in self-test (BIST) modules for obtaining a higher fault coverage.

Chapter 3 has addressed the interface testability problems between synchronous cores. The stuck-at fault coverage of these synchronizers have been increased from less than 70% to almost 100%. At the same time, the relevant path delay-faults, belonging to the designed delay-lines within these synchronizers, have been detected by designed BIST structures. The inserted DfT hardware has transformed the synchronizers in scan-compatible modules, ready to be scan-tested with the surrounding scan-compatible cores.

Chapter 4 presented scan-test strategies for asynchronous (using two-phase signaling) to synchronous (with stoppable clocks) interfaces (ASI). These interfaces were previously tested in a functional way or not tested at all. Additional DfT hardware has been introduced to transform the ASIs into scan-compatible modules. Modeled-for-ATPG versions of the ASIs have been developed in order to be able to automatically generate test vectors using commercial ATPG tools. The ATPG test vectors were able to detect all stuck-at faults in the original ASIs, therefore achieving 100% fault coverage.

Since the proposed test strategies are using traditional test techniques such as the scan method, it was possible to integrate these test strategies of the synchronizers to the SoC top level (chapter 5). This integration was simple and highly dependent on the overall test strategy developed by the test engineer. The two key features of the integration of the test strategies at the system level are the scan-test compatibility and the BIST concept. The synchronizers should be seen as "black-boxes", or better "library-cells", performing the synchronization operation between different types of cores. They will have an associated scan model, necessary for generating ATPG test vectors, and they should be integrated in the system BIST strategy.

This thesis has also addressed the testability of a tapped delay-line as often used in off-chip synchronization strategies. By using an oscillation test technique, it has been proven that accurate measurements, within $\pm 1.5\%$ or $\pm 10ps$, of the tap delays can be accomplished. A modified new delay-line has been presented which was better suited for the proposed differential oscillation technique. A prototype chip containing the new digital delay-line and a TAP controller has been designed and implemented in a UMC $0.18\mu m$ technology. Measurements have been performed on the chip and the accuracy mentioned above has been achieved. These measurements are important since the digital synchronization mechanism is relying on accurate tap delays.

7.2 Accomplishments of the thesis

The accomplishments of this thesis can be summarized as:

- Identify the future SoC design trends and the potential testing problems associated with it. As a result of this analysis our attention has been focused on the on-chip and off-chip interfaces between digital cores.
- test strategies comprised of scan and BIST have been presented for all possible synchronous-to-synchronous interfaces.
- a scan-test strategy has been developed and verified for the two-phase asynchronous-to-synchronous (with stoppable clocks) interfaces.
- a method for accurate tap-delay measurements of a digital delay-line used in high-speed synchronization has been presented. An improved design of the delay-line has also been introduced which is better suited for the chosen measurement method.
- a successful implementation of a chip containing the new digital delay-line and the control logic in an UMC $0.18\mu m$ technology has been accomplished. The measurements by a verification test system confirmed an accuracy of $\pm 10ps$ for the tap-delays measurements.

7.3 Recommendations for future research

There is still much work to be done in this research area. The core-based design style is taking off and the problems associated with it must be addressed. One of them is the testing problem. Hierarchical test will be the only way forward. Up to know, the complexity of the digital designs was at such a level that testing could have been performed using off-chip test equipment. No hierarchical test methods have been imposed on the overall test strategy. The P1500 standard addresses this issue; however, it seems that the cores themselves are not well prepared for hierarchical test. Therefore, a good research topic will be to develop hierarchical test strategies by means of making the cores themselves aware of the hierarchy.

Cores themselves cannot communicate efficiently with each other "out-of-the-box". New interfaces are being developed to smooth the communication between the cores. They usually do not obey the well-known syn-

chronous design style and as a result they are difficult to test. This thesis has only tackled a small portion of these new core interfaces. The presented synchronous-to-synchronous interfaces represent only a small portion of the possible designs. Also, there are other asynchronous-to-synchronous interfaces that have not been addressed in this thesis. Whenever such interfaces are being developed, they should also be delivered with the proper test strategy. This is not the case nowadays. Their test strategy should also be integrated with the surrounding cores. DfT area does not pose a problem nowadays. The rate at which the capabilities of integration for the new technologies is evolving is by far greater than the designer's ability to exploit it. DfT should be part of every design and should try to keep the fault coverage as high as possible, even if the area overhead is higher than what we are used at this moment.

Summary

This thesis presents test strategies for on-chip or off-chip high-speed synchronization interfaces. Emerging ULSI technologies demand a hierarchical design approach also known as *core-based design*. Integrating many intellectual property (IP) cores supplied by different vendors, will be the main task of future digital design engineers. Special on-chip interfaces between cores should be designed in order to improve the average transmission latency. The design of off-chip interfaces should also be improved for a complete optimization up to the PCB level.

For every IC design, the test is an important step of the product development cycle. Any future test strategy should have at least two steps. During the first step, test vectors provided by core vendors should be employed to test each core separately. During the second step, the cores should be tested together with other surrounding cores. Failing to perform the second step and thus testing the system as a whole, will undermine the confidence about the quality of the final product. While testing each core separately should not be a difficult task, due to the availability of the test vectors from the core provider and due to the emerging P1500 standard, the test of the complete system poses many problems. One of them is that the newly designed interfaces are not pure digital designs anymore. Testing them using a pure scan-test strategy is impossible. Another testing problem arises as a result of different clocks used by cores. Scan-chains will have to be constructed with care for such designs. Asynchronous cores complicate the testing problem even more due to their clock-less design strategy.

Integrating all digital cores to build a complex SoC system is a difficult task for both the design and test engineers. This thesis tackles some test issues associated with future SoC designs. First, all possible interactions between generic digital cores in use nowadays are analyzed in chapter 2. This classification helps pinpointing test related problems specific to these

interactions. Next, several types of potential synchronous interfaces, designed for a low average transmission latency, are described in chapter 3 and their associated test problems are exposed. Scan and BIST (SaB) test strategies are presented in the same chapter. This test strategies increase the fault coverage of these interfaces and smoothen their integration (chapter 5) in a more general test strategy at the SoC top level. Asynchronous to synchronous interfaces are presented in chapter 4 and their associated scan-test strategies are shown. Comparisons are performed between our proposed test strategies and the existing ones, whenever they are applicable.

Chapter 6 addresses the need for good testing techniques for off-chip interfaces. Digital tapped delay-lines are found to be an important module for most digital synchronization mechanisms. Tap-delay values accuracy is of great importance for the subsequent digital synchronization module. Measuring these tap-delays is not trivial due to their small delay values. By using a differential oscillation technique, it is possible to measure any tap-delays with high accuracy. The technique only relies on a precise off-chip generated frequency and on stable power supply and temperature during the measurement. Equations are given in order to relate the influence of surrounding parameters to the measurement accuracy. An improved design of the delay-line is introduced which is better suited for the chosen measurement method. The design of the digital delay-line and the control logic has been implemented in an UMC $0.18\mu m$ technology. The measurements confirmed a small accuracy when measuring the tap delays.

Biography

I was born on the 6^{th} of March 1975 in Bucharest, Romania. In July 1993 I graduated the theoretical high-school "Matei Basarab", after passing the final exams. Between 1993 and 1998 I followed the courses of the Faculty of Electronics and Telecommunications in Bucharest, Romania, which is part of the University "Politehnica" of Bucharest. In 1998 I received the bachelor degree in "Microelectronics and Computer Systems", specialization "Calculus Systems and Computer Aided Design for Microelectronics System". The graduation project was "Automation extraction of semiconductor device parameters using MATLAB". From 1998 until 1999 I followed the Master studies at the same faculty. At the end, I received the Master of Science degree in "Microsystems". The dissertation project was "Enhancements for the CELP vocoder".

Between 1997 and 1999, I was employed by LSI Design and Integration Corporation (LDIC) as a IC design engineer. I was involved in the front-end and back-end design improvements of the read/write channels used in mass storage devices.

Between October 1999 and December 2003, I have been enrolled in a Ph.D. program at the University of Twente in the Netherlands. The results of this program are published in this dissertation.

Index