Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

**Test Quality Analysis and Improvement for
an Embedded Asynchronous FIFO**

Tobias Dubois

Reg Nr: LITH-IDA/DS-EX--07/002--SE
Linköping 2007

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 93 Linköping, Sweden

Institutionen för datavetenskap
Department of Computer and Information Science

Master's Thesis

Test Quality Analysis and Improvement for
an Embedded Asynchronous FIFO

Tobias Dubois

Reg Nr: LITH-IDA/DS-EX--07/002--SE
Linköping 2007

Supervisor: **Mohamed Azimane**
Erik Jan Marinissen
Paul Wielage
NXP Semiconductors - Research
Clemens Wouters
NXP Semiconductors - Digital Library Technology
Erik Larsson
Linköpings universitet - Department of Computer and Information Science - ESLAB

Examiner: **Erik Larsson**
Linköpings universitet - Department of Computer and Information Science - ESLAB

Department of Computer and Information Science
Linköpings universitet
SE-581 93 Linköping, Sweden

Avdelning, Institution Division, Department Software and Systems (SaS) Department of Computer and Information Science Linköpings universitet SE-581 83 Linköping, Sweden	Datum Date 2007-04-01				
Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ _____	Rapporttyp Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ _____	ISBN _____ ISRN LITH-IDA/DS-EX--07/002--SE Serietitel och serienummer ISSN Title of series, numbering _____				
URL för elektronisk version http://www.ep.liu.se/YYYY/XXXX						
<table border="0"> <tr> <td style="vertical-align: top;">Titel Title</td> <td>Analys och förbättring av testkvalitet för en inbyggd asynkron FIFO Test Quality Analysis and Improvement for an Embedded Asynchronous FIFO</td> </tr> <tr> <td style="vertical-align: top;">Författare Author</td> <td>Tobias Dubois</td> </tr> </table>			Titel Title	Analys och förbättring av testkvalitet för en inbyggd asynkron FIFO Test Quality Analysis and Improvement for an Embedded Asynchronous FIFO	Författare Author	Tobias Dubois
Titel Title	Analys och förbättring av testkvalitet för en inbyggd asynkron FIFO Test Quality Analysis and Improvement for an Embedded Asynchronous FIFO					
Författare Author	Tobias Dubois					
Sammanfattning Abstract <p>NXP Semiconductors (formerly Philips Semiconductors) has created a new embedded asynchronous FIFO module. It is a small and fast full-custom design with Design-for-Test (DfT) functionality. The fault detection qualities of a proposed manufacturing test for this FIFO have been analyzed by a defect-based method based on analog simulation. Resistive bridges and opens of different sizes in the bit-cell matrix and in the asynchronous control have been investigated.</p> <p>The fault coverage for bridge defects in the bit-cell matrix of the initial FIFO test has been improved by inclusion of an additional data background and low-voltage testing. 100% fault coverage is reached for low resistance bridges. The fault coverage for opens has been improved by a new test procedure including waiting periods.</p> <p>98.4% of the hard bridge defects in the asynchronous control slices can be detected with some modifications of the initial test.</p>						
Nyckelord Keywords FIFO testing, FIFO, testing, defect-based testing						

Abstract

NXP Semiconductors (formerly Philips Semiconductors) has created a new embedded asynchronous FIFO module. It is a small and fast full-custom design with Design-for-Test (DfT) functionality. The fault detection qualities of a proposed manufacturing test for this FIFO have been analyzed by a defect-based method based on analog simulation. Resistive bridges and opens of different sizes in the bit-cell matrix and in the asynchronous control have been investigated.

The fault coverage for bridge defects in the bit-cell matrix of the initial FIFO test has been improved by inclusion of an additional data background and low-voltage testing. 100% fault coverage is reached for low resistance bridges. The fault coverage for opens has been improved by a new test procedure including waiting periods.

98.4% of the hard bridge defects in the asynchronous control slices can be detected with some modifications of the initial test.

Acknowledgments

I would like to thank all my supervisors: Mohamed Azimane, Erik Jan Marinissen and Paul Wielage of NXP Semiconductors - Research, Clemens Wouters of NXP Semiconductors - Digital Library Technology and Erik Larsson of Linköping University, for participating in this project and dedicating their time to it. Thanks to all the people of ED&T/Test for your help and support. Thanks to all the students of WAY for making my days more fun and thanks to all the people of Jeroen Boschlaan 142 for late dinners and friendship.

Contents

1	Introduction	1
1.1	Background	1
1.2	Task Description	2
1.3	Report Outline	2
2	Semiconductor Testing	3
2.1	The Manufacturing Process	3
2.2	Testing	5
2.3	Test Access	5
2.4	Classical Memory Fault Models	6
2.5	Defect-Based Testing	7
2.6	Defect-Based Fault Models	9
2.6.1	Bridges	9
2.6.2	Opens	13
3	FIFO Design	15
3.1	Introduction	15
3.2	The NXP Embedded Asynchronous FIFO	15
3.3	Specification	15
4	Defect-Based Analysis Method	17
4.1	Method Overview	17
4.2	Simulation Speed-Up Techniques	20
4.3	Bridge Fault Extraction	21
4.4	Open Fault Extraction	22
4.5	Fault-Resistance Selection	23
4.5.1	Bridges	23
4.5.2	Opens	24
4.6	Windowing	24
4.7	Initial FIFO Test	27
5	Defect-Based Analysis Results	29
5.1	Resistive Bridges in the Bit-Cell Matrix	29
5.1.1	Test Analysis Results	29
5.1.2	Coupling Fault Analysis	33

5.2	Resistive Opens in the Bit-Cell Matrix	35
5.2.1	Gate Opens	38
5.3	Asynchronous Control	41
5.3.1	Bridges	41
5.3.2	Opens	42
5.4	Combining the Tests	43
6	Conclusion	45
6.1	Summary	45
6.2	Future Work	46
	Bibliography	47
A	Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO	49

Chapter 1

Introduction

1.1 Background

An increasing number of integrated circuits (ICs) are utilizing large numbers of First-In First-Out (FIFO) memories. FIFOs are used for intermediate storage, data rate conversion, and clock domain crossing. New design paradigms like Globally-Asynchronous Locally-Synchronous (GALS) and Network-on-Chip (NoC) make extensive use of embedded FIFOs. An individual FIFO is typically small in area size, but due to the large number of FIFOs per IC design, their overall impact on silicon area is significant. Consequently, Philips Semiconductors (now NXP Semiconductors) has decided to add an area-efficient full-custom FIFO module to their design library. The new FIFO is both smaller and faster than its conventional counterparts, which are typically based on an embedded SRAM or made up entirely from standard-cell logic. The new FIFO module is designed as a micropipeline, consisting of a series of asynchronously communicating stages. Each stage is implemented as a register of bit-cell latches and a control slice.

As all on-chip circuitry, this new FIFO needs to be tested for manufacturing defects. The size of an individual FIFO is small, and hence its impact on the IC-level yield and quality is small. However, the typical use scenario is that hundreds of these FIFOs are used in a single IC design. This means that the collective impact of all on-chip FIFOs on IC-level yield and quality is significant. Consequently, effective, yet efficient testing is important.

NXP Semiconductors uses a defect-based analysis method based on analog simulation at transistor level to check the defect detection qualities of a test [3, 2, 1]. For a newly developed module and its test, such as the new NXP asynchronous FIFO, this investigation is even more important, as during the conceptual development of the module's tests, some defects might otherwise easily be overlooked.

1.2 Task Description

The main task of this Master's Thesis project was to analyze the fault detection qualities of a conceptually developed test suite for the new asynchronous FIFO and to investigate possible improvements of that test suite. The investigations should be done by defect-based testing. It should be possible to use this investigation as a basis for creating an effective and efficient test that will be performed at the semiconductor manufacturing plant.

1.3 Report Outline

This report describes the defect-based analysis procedure for embedded memories and how it was applied to the new NXP FIFO module. The analysis led to accurate defect coverage figures and an improvement of the initial FIFO test suite. The remainder of the report is organized as follows. General aspects of manufacturing defects and testing is described in Chapter 2. The NXP FIFO design is described in Chapter 3. The defect-based analysis method used to evaluate and improve the initial test suite is described in Chapter 4. Chapter 5 describes the analysis results and suggest a number of improvements to increase the effectiveness of the test suite. The report is concluded in Chapter 6.

Chapter 2

Semiconductor Testing

This chapter describes what testing of integrated circuits entails and why it is necessary. Some classical memory fault models are presented and the defect-based approach to testing is introduced. The resistive model of bridge and open defects is described in detail.

2.1 The Manufacturing Process

The manufacturing of semiconductors can be described as a sequence of processing steps performed on a batch of semi-conducting wafers. The outcome of a manufacturing step depends on three factors: the process controlling parameters, the layout of the integrated circuit, and random environmental factors, named disturbances [4]. Disturbances can be caused by for example: human errors, equipment failures, instabilities in the process conditions, material instabilities, and lithography spots [8]. In general, all process disturbances cause either a geometrical or an electrical deformation [4]. The deformation can be further classified as either global or local. A deformation has a global influence if a particular parameter, such as the threshold voltage, is affected for the whole wafer. A local deformation is confined to a region much smaller than a wafer. Local deformations are often called defects.

If, for example, there is a dust particle on the wafer it may cause extra metal to be put on the chip, which in turn may cause two metal conductors to be unintentionally connected. Such inadvertent connections are known as shorts or bridge defects. Figure 2.1 shows an example of such a defect.

Other disturbances can cause material to be missing. Missing material in a conductor causes a complete or partial break in the conductor. Missing material in a via or contact similarly causes the connection between two layers to be broken or imperfect. These kind of defects are known as open defects. Figure 2.2 shows an example of an open conductor. There exists several other defect types, but bridge and open defects are the most common in most contemporary processes.

Figure 2.1. A bridge defect shorting two conductors.

Figure 2.2. An open conductor defect.

2.2 Testing

Due to the imperfections in the manufacturing process some of the manufactured circuits will not operate correctly. The fundamental objective of semiconductor testing is to find out which of the manufactured integrated circuits function according to specification and which do not. In well-controlled fabrication environments, global deformation problems are easily identified and kept under control [8]. Therefore, semiconductor testing focuses on detecting the local deformations known as defects. This is done by applying test stimuli to a circuit in such a way that the presence of a defect can be observed on one or more circuit outputs. The faulty behavior can be identified by comparing the obtained test results with the results expected from a fault free circuit. The fault free results come from a “golden” circuit, a circuit that is known to operate according to specification, such as a simulation model.

Testing is the last thing performed at the semiconductor manufacturing plant before the integrated circuit is ready for delivery. The actual testing is performed and controlled by an expensive piece of machinery known as an ATE, Automated Testing Equipment.

Together with the design and the fabrication process, testing is one of the major activities in the development of an integrated circuit. Testing is necessary to guarantee a high-quality product that will operate reliably without errors. It should not be confused with design verification. Verification is the process of verifying that a design is correct according to the design specification. Testing aims to detect manufacturing defects and it works from the assumption that the design has been verified and is functioning correctly.

The new deep sub-micron processes allow a higher level of integration than their older counterparts. This increased integration makes testing more complicated and it also makes the circuits more prone to defects simply because there are more possible defect locations [8]. The small feature sizes entail that even small defects that had no influence on circuits with larger feature sizes, now can cause problems.

The test cost is an important part of the total production cost of an integrated circuit. Its contribution can sometimes be as high as 50%. The test cost consists mostly of the test development cost, including DfT, and the cost for actually performing the test on the manufactured ICs. Test equipment and personnel are expensive. A good test should both be fast and be able to detect most defects. Often, a trade-off between the two is necessary.

2.3 Test Access

The high integration possible in modern VLSI circuits has allowed the inclusion of a large number of internal components, such as microprocessors, standard-logic, ASICs and memories, on a single chip. The external access to these components is restricted by the limited number of chip pins. This is a problem since efficient testing requires full access to the chip components. One way to solve the test access problem is to send test data serially into the chip components from an external

pin. The serial-scan approach does that by using flip-flops with a special scan-mode in the design. The scan-mode allows the flip-flops to be chained together by a dedicated scan line into a scan chain. When the scan-mode is activated it is possible to serially scan data into the flip-flops through the scan line. During testing, the test data is scanned into the scan-chain by the ATE, through an external pin. When all flip-flops have received the test data the flip-flops are set to normal mode and the design is clocked. In this way the internal state of the flip-flops can be externally controlled. To scan out the test-results, the scan-mode is activated again and the test results are scanned out from the chip to be checked by the ATE. The speed of the scan chain is usually limited by the speed of the ATE and the physical limitations of the external pin. Because of the serial nature of the scan chain and the relatively slow scan chain speed, the time between subsequent data inputs to a component can be relatively long.

Another solution to the test access problem is to integrate a Built-In Self Test (BIST) on the chip. A BIST is a dedicated circuit that tests a specific chip component by applying test patterns and checking the results on-chip. The BIST provides the test stimuli itself, so the test data does not have to be scanned into the chip. This saves test time. The test results are then checked directly on the chip. There is no need to transfer test data on or off the chip. A BIST enables high-speed testing at the maximum operating frequency of the circuit, since the BIST test speed is not limited by a scan chain or by the chip access speed of the ATE. The main disadvantage is the increase in silicon area usage.

2.4 Classical Memory Fault Models

Testing a circuit by testing its normal functionality is known as functional testing. The circuit is tested by performing standard operations without the help of special test features.

In the early days of semiconductors, circuits were tested through exhaustive functional testing. Exhaustive functional testing means that all possible functions of the circuit are tested. A simple 4-bit adder, for example, can be tested exhaustively with $2^4 = 16$ test vectors. For modern Very Large Scale Integrated (VLSI) circuits this approach is not feasible because of the increased circuit complexity. A small memory of 256 bits has $2^{256} = 1.2 \cdot 10^{77}$ internal states. Testing such a memory exhaustively at 1 GHz would take more time than the current age of the universe. Clearly, a more efficient test strategy must be used.

The exponential increase in test cost and the inefficiency associated with exhaustive functional testing led to the development of structural testing. In structural testing fault models are used to represent faulty circuit behavior in a form suitable for simulation and test generation. The objective of the test is no longer to exhaustively test all the functions of the chip, but to detect the faulty behavior represented by the fault models. This effectively limits the number of necessary test vectors.

A fault is defined as a model of the effect a physical defect has on the circuit. One of the most well-known fault models is the *stuck-at* model. It models defect

behavior by assuming that faulty behavior will show up as if a net in a circuit is stuck at a logic zero or a logic one. These two possible faults are known as a stuck-at zero and a stuck-at one. The model can be used at the gate-level or, for better accuracy, at the transistor-level. In memory testing a stuck-at fault means that a memory cell is locked in either a logic zero or a logic one state.

A *transition fault* in a memory cell means that only one type of data transitions are possible. For example, transitions from zero to one may be possible, but not transitions from one to zero. The cell can initially be in either the one or the zero state, but once it is written to the state where no transitions are possible, it is stuck. In this sense it is similar to a stuck-at fault.

A faulty memory cell can be coupled to neighboring cells. The coupling makes the cells dependent on each other so that if one cell is written to a certain state the coupled cell also changes state. This is known as a *coupling fault*.

If a memory cell has a *data retention fault*, the data in the cell will slowly degrade and eventually be lost due to current leakage.

2.5 Defect-Based Testing

The classical fault models are good as long as they accurately represent faulty circuit behavior. However, it has been shown that functional fault models, such as stuck-at, transition and coupling faults, are insufficient to properly model the effects of defects occurring in current technologies.

In Defect-Based Testing (DBT), circuit-level faults are derived directly from potential particular physical defects. The first step is to identify possible defects by analyzing the circuit layout for failure sites. For example, two metal conductors that are routed close to each other are susceptible to bridge defects. A contact or via may cause an open defect. The defects found are then mapped to an appropriate fault model. A fault is an abstract model of a defect. The effect of an individual fault on the circuit can be simulated by creating a faulty circuit based on the fault model of the fault. By simulating the effect of the test-patterns on faulty circuits, it is possible to determine if the test-pattern can detect the fault or not. By doing this for all the possible faults, the effectiveness of the test in detecting the faults can be evaluated.

There are several ways of performing defect-based testing. Inductive Fault Analysis (IFA) is perhaps the most well-known defect-based approach for determining what faults are likely to occur in a circuit. Shen et al. defines the IFA procedure in three major steps: Step (1): generation of possible physical defects using statistical data from the fabrication process. Step (2): extraction of circuit-level faults caused by these defects, and Step (3): classification of fault types and ranking of faults based on their likelihood of occurrence [9].

The circuit layout contains information on the probability of occurrence of defects. For example, the likelihood of having a bridge defect between two parallel nets depends on the length of the nets, the distance between the nets and the size of possible defects. One method to estimate this probability is to use the concept of critical area. The critical area of two nets is defined as the area where the

center of an imagined circular defect of a certain size has to be located to create a short between the nets, see Figure 2.3. To get the relative defect probability for the defect site, the critical area is calculated for all defect sizes and weighted with the probability of occurrence of defects of those sizes. For a standard defect size distribution, the probability (P) of a bridge defect between two parallel conductors can be shown to be proportional to the length of the nets (l) divided by the distance between them (d), see Eq. 2.1. In a similar way the relative probability of open defects can be estimated based on the length and width of a net.

$$P \propto l/d \quad (2.1)$$

Figure 2.3. Critical area of a circular bridge defect.

Higher level fault models, such as the classic memory fault models described above, usually assumes all defects to be equally likely which is not the case in reality. In a defect-based layout-level model, defect likelihood information can be used to more accurately estimate the effectiveness of a test.

Defect-based testing gives a very accurate way of calculating the effectiveness of a test since all the faults are realistic faults based on fabrication defect types known to be present on the physical circuits. Fault models based directly on defects are much closer to reality than models at higher abstraction levels that are only indirectly based on defects.

The effectiveness of a test can be estimated by calculating the *fault coverage*. The standard fault coverage is defined as the number of detected faults (n_{det}) divided by the total number of faults (n_{tot}), see Eq. 2.2. If probability information is available for the faults, the *weighted fault coverage* can be calculated. The weighted fault coverage is defined as the sum of the relative probabilities (P_r) of the detected faults divided by the sum of all relative probabilities, see Eq. 2.3. The weighted fault coverage gives a more accurate estimate of the relative effectiveness of a test than the unweighted fault coverage.

$$\text{Fault Coverage} = \frac{n_{det}}{n_{tot}} \quad (2.2)$$

$$\text{Weighted Fault Coverage} = \frac{\sum_{i \in F_{det}} P_{r_i}}{\sum_{i \in F_{tot}} P_{r_i}} \quad (2.3)$$

Defect-based fault models based on bridge and open defects includes the behavior of many higher level models such as those mentioned earlier. For example, a stuck-at fault can be represented with a short to one of the power lines. A data retention fault can be modeled as an open in a memory cell. But the defect-based models also exhibit other types of behavior that are not properly modeled by traditional higher level models.

The drawback of defect-based testing is the high computational requirements for simulating the often large number of faults that are identified during the fault extraction process.

2.6 Defect-Based Fault Models

The predominant defects in modern deep sub-micron VLSI circuits are opens and bridges. That is why these defects are the ones investigated in this project.

2.6.1 Bridges

A bridge defect is an unwanted electrical connection between two conductors on a chip. The electrical properties of the bridge can include both linear and non-linear resistance, and capacitance. Because the linear resistive behavior usually is dominant and for sake of simplicity, a bridge defect is usually modeled with a linear resistance. The model is known as a resistive bridge fault model. An unintentional short between two metal conductors on a chip is modeled by a resistance inserted between the two nets in question. By varying the bridging fault resistance, different types and sizes of bridge defects can be modeled. Figure 2.4 shows an example of a bridge defect connecting two conductors and how it is modeled with a resistance between two nets.

Figure 2.4. A bridge defect and its fault model, a resistance.

Figure 2.5 shows an example of two standard library CMOS inverters driving two nets, **Net 1** and **Net 2**, connected by a bridging fault. Figure 2.6 shows the

resulting voltage levels in the nets for different bridge fault resistances when the input to the inverters are a logic zero and a logic one as shown in Figure 2.5.

If both **Net 1** and **Net 2** are initialized to the same voltage level the bridge will not influence the circuit and it cannot be detected. For detection to be possible, the two nets connected by the bridge needs to be driven with opposite logical values. The resulting different voltage levels creates a current flow between the nets through the bridge resistance. This causes the voltages on the two nets to deviate from the correct value. For the input values shown in Figure 2.5, current will flow from **VDD** to **Ground** through the PMOS1 transistor of the top inverter, the bridge resistor, and the NMOS2 transistor of the bottom inverter. The current path is indicated by the red line in Figure 2.5. Figure 2.6 shows the resulting node voltages for different fault resistances.

Figure 2.5. Two CMOS inverters connected by a bridging fault.

If the bridging fault is a perfect short with zero resistance, both net voltages will be the same. A bridge with non-zero resistance will cause a voltage gap between the two nets. The size of the gap will depend on the size of the bridge resistance. The bigger the resistance, the smaller the effect of the bridge. For low bridge resistances the voltage deviation in one of the nets will be large enough to cause an incorrect logic output in a gate connected to the net. For a symmetric gate the switching-threshold voltage is $VDD/2$. If the deviation is larger than $VDD/2$, the following gate will have an incorrect logic output. The highest bridge resistance that still causes an incorrect logic output at the following gate is known as the critical resistance of the bridge. For the inverter example, the critical resistance for **Net 1** is just below 10 k Ω as indicated by the vertical line in Figure 2.6.

Figure 2.6. Resistance versus voltage for Net 1 and Net 2 for $V_{DD} = 1.2$ V.

Net 1 is affected more by the bridge than Net 2 because of the relative strength of the conducting transistors. For bridge resistances above the critical resistance, the bridge cannot easily be detected by logic testing. It may still be detected through I_{DDQ} testing, a method which detects bridges by measuring the increase in quiescent current caused by the bridge. There are however other problems with I_{DDQ} testing, but they will not be discussed in this text.

When the supply voltage is lowered, the on-resistance of the transistors increases, but the resistance of the bridge does not change. Consequently, the resistance ratio between the bridge and the transistors becomes smaller, making the effect of the bridge more severe. This can be seen as an increased critical resistance as shown in Figure 2.7 where the supply voltage to the inverters has been lowered to 1.0 V. Low voltage testing is an effective way of detecting bridges that can not be detected at nominal voltage [3].

Figure 2.7. Resistance versus node voltage at Net 1 and Net 2 for $V_{DD} = 1.0$ V.

Is it really necessary to detect a bridge with a resistance higher than the critical resistance, since such a bridge will not cause a logic error? The answer is yes, it can be necessary since the bridge defect resistance and other circuit parameters can change with time and operating conditions. Hence, a bridge that does not cause a problem at the time of manufacturing still may cause a problem later

in the product life cycle. A bridge also causes a short circuit current that will increase the circuit power consumption. Furthermore, the bridge may degrade the chip performance and lower the noise margin.

The resistance of actual bridge defects in a CMOS circuit was measured in [5]. It was found that most bridges have a resistance below 500 Ω , but there exists defects with resistances up to 20 k Ω . However, these results are quite old and it is unclear how they apply to modern deep sub-micron copper based processes.

2.6.2 Opens

An open defect is a complete or partial break in an electrical connection on a chip. It can be a break in a conductor or a broken via or contact. The defect can exhibit both resistive and capacitive behavior. For simplicity it is often modeled as a resistance that splits the affected net into two parts. Figure 2.8 shows an example of a partially broken conductor open defect and how it is modeled with a resistance inserted in the net corresponding to the conductor. Resistive opens cause an increased signal delay in the affected net. This is because the increased resistance increases the RC-delay.

Opens with resistances higher than 10 M Ω are classified as strong. They usually cause a hard failure in the circuit, which is easy to detect. Opens with a resistance lower than 10 M Ω are known as weak opens. These kind of opens may allow the circuit to function correctly, although with degraded performance [6]. To detect this defect type the increased delay has to be detected. Increasing the test frequency is one way of doing this. In [3] it was shown that SRAM testing at a voltage level higher than the nominal voltage and high frequency testing can improve the detection of open defects.

Figure 2.8. An open defect and its fault model, a resistance.

The distribution of open defect resistances in a 0.18 μm CMOS process was investigated in [6]. It was found that more than 65% of open conductors, contacts and vias have a resistance larger than 10 M Ω and can be classified as strong, but there are also a substantial amount of defects with a resistance lower than 10 M Ω , so these defects should also be considered.

Because of the many interconnection layers and the accompanying vias and contacts between the layers, the significance of via and contact opens has significantly increased in recent technologies. Intel has reported open/resistive vias to be the most common root cause of test escapes in deep-sub-micron technologies [7].

Chapter 3

FIFO Design

3.1 Introduction

A First-In First-Out (FIFO) memory is a specialized type of memory buffer. It has one write port and one read port. Data written at the write port can only be read out at the read port in the same order it was written. This implies that the first data written to the FIFO is the first that can be read out. Hence the name First-In-First-Out. There is no data addressing mechanism so random data access is not possible.

Embedded FIFO memories are used in a wide variety of applications for intermediate storage, data rate conversion, and clock domain crossing. New design paradigms like Network-on-Chip (NoC) and Globally-Asynchronous Locally-Synchronous (GALS) use embedded FIFOs extensively. FIFOs are very useful building blocks in a lot of different designs and their importance will probably stay high, if not increase, in the near future.

3.2 The NXP Embedded Asynchronous FIFO

The design and DfT of the NXP Embedded Asynchronous FIFO, which have been used in this project, has been thoroughly described in the paper “*Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO*” by Wielage, Marinissen, Alzheimer and Wouters of NXP Semiconductors [11]. The paper is included in Appendix A and I suggest the reader to read it in full to be able to follow the rest of this report.

3.3 Specification

Table 3.1 shows some limiting values for the FIFO.

The maximum operating frequency of the FIFO approaches 1 GHz.

Parameter	Min	Typical	Max
Supply voltage	0.90 V	1.20 V	1.30 V
Temperature	-40°C	25°C	150°C

Table 3.1. Limiting values for voltage and temperature.

Chapter 4

Defect-Based Analysis Method

This chapter describes the defect-based analysis method that was used to evaluate the effectiveness of the FIFO test-suite. A number of techniques used to speed up the fault simulations are presented. The defect extraction and fault generation process is described in detail. Furthermore, the initial test suite is introduced.

4.1 Method Overview

NXP Semiconductors have developed a practical defect-based method to evaluate and design test suites for embedded modules, such as embedded memories [2, 1]. The method, which is based on Inductive Fault Analysis (IFA) [9], consists of a number of steps. The first step is to use layout information to find possible defects and estimate their probability. Bridge and open defects are considered since they are thought to be the most probable defects in the investigated circuits. The identified defects are used to create a fault list of resistive bridge and open faults¹. This fault generation step can be done automatically by an in-house tool known as FAULTGEN. In the second step, analog simulations are performed on transistor-level netlists of the module, augmented with different faults, to determine if the faults can be detected by a given test suite. This step has been largely automated in the form of an in-house tool called MEMSIM. The tool injects all identified faults, on by one, into the transistor-level netlist of the module, simulates them with a test-bench based on the test suite, and reports if the faults have been detected or not. From this information it also calculates the fault coverage and, if defect probability information is available, the weighted fault coverage. The next step is to investigate the undetected faults. This analysis often results in modifications and additions to the test patterns and/or the stress conditions [3]

¹Note the use of the terminology here. A *defect* is an actual physical defect on the chip, while a *fault* is a simulation model of the defect.

in order to improve the fault coverage. The fault simulation step is then repeated to get new fault coverage numbers for the updated test.

The flow chart shown in Figure 4.1 gives an outline of NXP's defect-based analysis method. The top part of the flow, with dark-gray boxes and white lettering, indicates the part of the flow which is only executed once. The bottom part of the flow consists of two nested loops. The medium-gray boxes depict what is part of only the outer loop, while the light-gray boxes indicate which operations are performed in the inner loop.

The flow starts with the layout of the module under consideration. From this layout, a transistor-level netlist is extracted, including parasitic elements such as resistances and capacitances. The parasitics can also be used to extract location and probability information for possible defects, based on critical area analysis. The parasitic capacitances contain information about the critical area of bridges [10]. Similarly, the parasitic resistances can be used to extract information about open defects. From this information, a fault list of potential bridge and open faults is created. The parasitic extraction is done by a standard extraction tool (ASSURA in this case). The fault-list generation is automated by the in-house tool FAULTGEN. This part of the flow is only performed once.

The second part of the flow is performed iteratively. First, a simulation test-bench based on the test suite is defined for the module netlist. It provides the test stimuli and includes parameters such as supply voltage, temperature, and write and read clock frequency. An analog simulation is performed on the fault-free (golden) transistor-level netlist to make sure that it operates properly. The simulation result is stored for later reference. Subsequently, for each fault in the fault list, the fault injection operation creates a dedicated faulty netlist, containing the original fault-free netlist augmented with exactly that fault with a certain specified resistive value. One analog simulation is performed for each of the faults in the fault list. The results of these simulations are compared against the simulation results for the fault-free netlist. The faults for which the corresponding faulty netlist simulation leads to a mismatch with the fault-free simulation are considered as detected, while faults which do not cause a difference in the simulation results are marked as undetected. Using this information the fault coverage (detected faults / all faults) of the test can be calculated. The fault simulations are automated by the in-house tool MEMSIM. Only one fault is considered at a time since because of the random distribution of defects it is very unlikely to have several defects within the small area of a single FIFO.

In the inner loop, the fault simulation procedure is repeated for a set of representative fault resistance values. The outer loop is entered only if the fault coverage of the test is unacceptably low. In that case, an attempt to improve the test suite is made by modifying the test stimuli or the test parameters (i.e. supply voltage, clock frequency). Note that the outer loop requires re-simulation of the fault-free netlist.

Figure 4.1. Defect-based analysis flow.

4.2 Simulation Speed-Up Techniques

The NXP defect-based simulation flow gives very accurate information about the behavior of potential defects and the effectiveness of the test suite. The major problem of the flow is that it can be very expensive with respect to compute time, due to the many analog simulations. A circuit with d potential fault locations requires $d + 1$ analog simulations per iteration. The inner loop is repeated for the number of distinct fault resistances. For a set of r different resistances there will be $r \cdot d + 1$ analog simulations in the standard flow. The total number of possible bridging faults in the smallest FIFO instance is greater than 8000. If five different fault resistances are used and each simulation takes 10 min, the total simulation time would be $8000 \cdot 5 \cdot 10 \text{ min} \approx 278$ days for simulating all faults on a single computer. This is too much time to be practical. A crucial aspect of the experiments has been to obtain realistic and relevant results, while keeping the corresponding compute time within tractable limits. To accomplish this, a number of speed-up techniques have been applied.

- *Selection of analog simulator.* An accurate Spice-like in-house analog simulator needed hours of compute time for one golden simulation. As this was considered too long, a switch was made to Synopsis' HSPICE simulator. HSPICE is a hierarchical simulator, capable of exploiting regularities in the circuit to avoid duplication of calculations. The accuracy was tuned to a level sufficient for the investigation purposes. In this way a simulation time of less than 5 minutes was obtained for the transistor-level netlist including parasitic capacitance using the standard test-bench.
- *Parallel computation.* The MEMSIM tool supports distributing the fault simulations among multiple computation servers. The number of simultaneous fault simulations possible was mostly limited by the number of HSPICE licenses available at any given time.
- *Abort-on-first-fail.* The simulations of the defective netlists are compared on a pattern-by-pattern basis with the golden-simulation results, and aborted as soon as a mismatch is found.
- *Selection of simulation instance.* The smallest FIFO instance of 16×19 bits was selected for the analog simulations. The transistor-level simulation netlist of this instance contains around 3500 transistors and 8000 parasitic capacitances. The two other instances is similar to the 16×19 FIFO in every way except for the number of bit-cells. Hence, the results obtained for the smallest FIFO will be valid also for the larger FIFOs.
- *Windowing.* In order to reduce the number of defect locations d , the regular character of the FIFO layout was exploited by considering only defects in a small window. For the bit-cell matrix of the FIFO, the focus was on bridges and opens within one typical static latch and bridges between the latch and its direct neighbors. For the asynchronous control of the FIFO, the focus was on one typical slice. First a "typical" bit-cell and control slice was selected.

The assumption is that a fault inserted at the corresponding position in other cells and slices will cause the same type of failures. If the fault mechanisms for one cell can be understood, this information will be valid also for the other cells. To verify this assumption and to take some extreme cases into consideration, some extra cells and slices were later added.

- *Small set of resistances.* In order to reduce the number of distinct defect resistances r , only a small number of fault resistances were considered, while still covering the entire range from “hard” to “weak” bridges and opens.
- *Implied detection.* Another possible speed optimization is to exploit the fact that for bridges, faults with smaller resistances cause a much larger impact on the circuit than faults with larger resistances. This implies that for a certain fault location a fault with a smaller resistance will always be easier to detect than a fault with a larger resistance. This can be exploited by simulating the larger fault resistances first. If a fault is detected at a certain fault location for a large fault resistance, this fault will also be detected for all resistances smaller than the one simulated and it is unnecessary to simulate this fault for any of the smaller resistances. If the test quality for several different resistance values are to be investigated, a lot of extra simulations can be avoided. The same principle works for opens as well, but here the small fault resistances are the ones the most difficult to detect.

4.3 Bridge Fault Extraction

The probability of having a bridge at a certain position in a circuit layout is usually calculated by means of the critical area of defects as described in Section 2.5. According to the theory of critical area the probability (P) of having a defect between two parallel wires is proportional to the length (l) of the wires divided by the distance (d) between them (see Equation 2.1), if a standard defect size distribution is assumed. This relative probability is everything that is needed to calculate the weighted fault coverage.

In the NXP flow the critical area of bridge defects is calculated indirectly by using the parasitic capacitances in the design [10]. If two conductors on a chip are routed next to each other there will be a parasitic capacitance between them. According to the parallel plate capacitance model this capacitance is described by Equation 4.1.

$$C = \frac{\epsilon_0 \epsilon_r A}{d} \quad (4.1)$$

The conductor Area (A) can be divided into height (h) and length (l) of the conductor since $A = h \cdot l$. The permeability for vacuum (ϵ_0) is constant and if the height and relative permeability (ϵ_r) are also constant, P will be proportional to C , and C can be used as a measure of the relative probability of a defect. This is exactly what the tool FAULTGEN does. It generates a fault from each parasitic capacitance and assigns it a probability proportional to C . The advantage of this

method is that fault extraction from the layout can be done with a standard parasitic extraction tool so there is no need to develop a special tool for critical area calculations. Parasitic extraction tools are usually fast and parasitic extraction is something that is done anyway.

However, there are also a few problems with this method.

- The parasitic extraction tools does not use the simple parallel plate capacitance model in Equation 4.1 for calculating capacitance. Instead they use a more advanced model including things like fringing capacitance. Equation 4.1 is therefore not completely accurate in this case.
- ϵ_r may not be constant between all layers, so to get more accurate results the capacitance could be divided by ϵ_r of the respective layer.
- The extraction tool has a setting for the minimum parasitic capacitance that will be included in the extraction, which means that some possible bridges may be excluded in the fault list generation. This is not a big problem since the minimum parasitic capacitance number can be adjusted and if it is low, the bridges that may be skipped should be very improbable.
- It is possible to have a single bridge that connects more than two conductors. These types of bridges are not considered in the NXP flow. However, bridges between two of the conductors will be included and they are more difficult to detect. It is assumed that if a bridge between two of the conductors is detected, a bridge that includes these two conductors and more will also be detected.
- Another problem is the difference between inter-layer defects and intra-layer defects. Equation 2.1 is really only valid for inter-layer defects. For intra-layer defects it is often assumed that the defect probability is proportional to the overlap area of the two conductors in different layers. The inter-layer capacitance is proportional to this area and it can be used as a reasonable approximation of the relative probability of intra-layer bridges. The problem occurs when relative inter-layer probabilities and intra-layer probabilities should be related to each other. Usually inter-layer bridges are up to two orders of magnitude more likely than intra-layer bridges. To make the probability approximation more accurate inter-layer capacitances and intra-layer capacitances should be given a different weight.

Because of the problems mentioned the capacitance method is not as accurate as measuring the critical area directly in the layout, but it is fast, easy to implement, and it produces a useful estimation of the relative probability and the corresponding weighted fault coverage.

4.4 Open Fault Extraction

For opens, critical area is defined as the area within which a hole defect of a certain size must be situated to create a complete break in the affected conductor.

This means that the critical area will increase with the length of the conductor and decrease with the width of the conductor. The critical area will be the same as the critical area for a bridge between two parallel conductors with a distance between them equal to the width of the conductor affected by the open. If the size distribution for opens has the same shape as the size distribution for bridges, the relative probability of an open defect will be proportional to the length of the conductor divided by the width of the conductor.

Hence, the parasitic resistances can be used as an estimate of the relative open probabilities in the IC, although the results should be corrected for the different resistivity of metal layers and the poly-silicon layer. Open vias and contacts have a different probability than breaks in conductors and they should be handled separately.

Using the parasitic resistances as a way of locating possible open defects and estimating their relative probability is not supported by the current version of FAULTGEN and thus this method has not been used in this project.

Currently the FAULTGEN tool extracts open faults based only on the transistors in the netlist. The tool creates an open fault for each transistor terminal: source, drain, gate and base. The base opens were excluded because they can not be reliably simulated by the current transistor models. Since all opens are connected to a specific transistor, open faults that affect several transistors at the same time are not included. But these kind of opens are generally much easier to detect than opens affecting only one transistor terminal and if the single transistor opens are detected, the multiple transistor opens will probably also be detected. Another drawback of this method, as opposed to the critical area method, is that it provides no likelihood information.

Since the bit-cells are simple and small, and they are also responsible for a major part of the faults, it was decided to extract all opens in a bit-cell by hand, including multiple transistor opens, to get a more accurate treatment.

4.5 Fault-Resistance Selection

4.5.1 Bridges

There are currently no measured information about the distribution of defect resistances in the NXP 90 nm CMOS process. Therefore, it is necessary to rely on defect resistance measurements from other processes and previous experience when selecting fault resistances. In [5], Montañés et al. showed that high-ohmic bridges of 20 k Ω and more do occur in deep sub-micron designs. They also showed that low resistance bridges (< 1 k Ω) is a lot more probable than high resistance bridges. Based on this information and based on previous experience from NXP memory testing it was decided to consider bridges with resistances values of 100 Ω (“hard” bridge), 1 k Ω , 10 k Ω , 50 k Ω , and 100 k Ω (“weak” bridges). This resistance selection gives as an overview of the faulty behavior throughout the resistance spectrum from “hard” to “weak” bridges, although with limited resolution. The focus lies on the smaller, more probable bridge resistances.

4.5.2 Opens

In [6] Montañés et al. showed that deep sub-micron designs can contain opens and that the resistive value of these opens shows a wide spread ranging from “weak” opens with resistances lower than $10\text{ M}\Omega$ to “hard” opens with resistances larger than $10\text{ M}\Omega$. Opens with resistances of $1\text{ G}\Omega$ (“hard” open), $1\text{ M}\Omega$, and $10\text{ k}\Omega$ (“weak” opens) were considered. According to [6] “hard” opens are more probable than “weak” opens. Therefore, the investigations concentrate on “hard” $1\text{ G}\Omega$ opens. The reason why fewer resistances were considered for opens than for bridges is mainly due to a lack of time. More time was spent on bridges since they are thought to be more probable than opens.

4.6 Windowing

Windowing is a technique used to reduce the number of faults that need to be simulated and to simplify the investigations by separating the circuit into different functional parts that can be investigated separately. The different parts of the 16×19 FIFO are described in Table 4.1. A separate fault list is generated for each type of part. Because of the time constraints of this project, it was decided that the focus should be on the bit-cell matrix and the control slices. These two were chosen because they are the two parts with the highest defect probability. Together they account for more than 70% of the parasitic capacitance and hence for more than 70% of the possible bridge defects. This number should be even larger than 70% since most of the capacitance for power lines which is listed under *Other* is actually located within the bit-cell matrix and the control slices. The power line capacitance comes from parasitics between **VDD** and **Ground**.

There are several instances of the memory cells, the asynchronous control slices, and the input and output buffers. All instances of these part types are equivalent and they have a similar behavior with respect to defects. To reduce the number of faults and save simulation time, a selected few of these instances are carefully chosen for fault list generation. The simulation results from those few can then be extrapolated to get the complete picture.

The analysis was started by selecting a “typical” bit-cell as a representative of bit-cell behavior. A cell in the middle of the bit-cell matrix was chosen and possible opens and bridges within this cell and bridges between the cell and neighboring cells were extracted. Later, fault simulations were performed for a few extreme case bit-cells to see if they behaved differently from the “typical” bit-cell in any way. The same procedure was used for the control-slices. First, fault simulations of a “typical” slice was done and then a few extra slices were simulated to verify the simulation results from the “typical” slice. The investigated bit-cells and control slices are indicated in Figure 4.2.

FIFO part	Number	Area	Parasitic Capacitance	Description
Bit-Cell	323	36.4%	45%	Memory latch that stores one bit.
Control Slice	16	23.4%	25%	Controls the pipeline by a 4-phase handshake protocol.
Write Control	1	3.1%	3%	Controls write operations.
Read Control	1	1.5%	3%	Controls read operations and reset.
Full Flag Control	1	2.5%	2%	Creates the full flag.
Empty Flag Control	1	1.4%	1%	Creates the empty flag.
Input Buffers & Mux	19	9.4%	10%	Selects data, buffers the inputs and creates dual-rail data signals.
Output Buffers	19	4.3%	4%	Buffers the data output and converts the data back to single-rail.
Other	1	18.0%	7%	Empty space and power lines.

Table 4.1. C090FIFO circuit parts.

Figure 4.2. Investigated bit-cell and control-slice windows. The “typical” cell and slice are indicated by a thicker line.

4.7 Initial FIFO Test

The original INTEST procedure was conceived before the start of this project based on previous experience in memory testing and scan-chain testing. For an $n \times m$ -bit FIFO (with n even), the initial INTEST procedure consisted of three steps as listed in Table 4.2, to be executed at nominal frequency, temperature (25°C), and supply voltage (1.2 V).

Step	Operation	#ops
1	Reset	1
2	Write $(00 \dots 0; 11 \dots 1)^{n/2}; 00 \dots 0$	$n + 1$
3	Shift-Out $(00 \dots 0; 11 \dots 1)^{n/2}$	n

Table 4.2. Initial FIFO INTEST procedure.

Upon start-up of the SOC, the FIFO is in an unknown state. In order to bring it to a known state, Step 1 resets the FIFO by keeping the **reset** high for one clock cycle. The reset operation flushes the FIFO and initializes it as empty, but it does not reset the state of the bit-cells in the bit-cell matrix. The content of the bit-cells is still undefined even after a reset.

Subsequently, in Step 2, an alternating sequence of $00 \dots 0$ and $11 \dots 1$ words are written into the FIFO. The fact that alternating values are written for a given bit i in subsequent words allows us to test whether the FIFO correctly performs write operations at different levels of being filled. Also single-cell stuck-at and transition faults can be detected in this way. The fact that the bit-cell layout consists mainly of an alternating sequence of neighboring bit and bit-not conductors means that Step 2 fills the FIFO with a physical checkerboard pattern. This is meant to detect coupling faults between neighboring cells. For an n -word FIFO, $n + 1$ words are written, in order to let the last write operation test whether or not the FIFO refuses to write additional data once full. Note that it is *not* allowed to execute read/shift operations simultaneously to these write operations, as this would not lead to a gradual fill of the originally empty FIFO.

Finally, in Step 3, the FIFO content is read out by shifting. Shifting a full FIFO actually exercises some of the worst-case timing paths in the FIFO, as for every individual shift operation, a hole needs to be propagated from the read interface to the write interface through the entire FIFO.

Chapter 5

Defect-Based Analysis Results

In this chapter, detailed analysis results of bridge and open defects in the bit-cell matrix and the control slices are presented and it is shown how the the fault coverage of the initial INTEST was improved. It is also made believable that the test is efficient and effective in the sense that it reaches the highest possible fault coverage in the shortest possible time.

5.1 Resistive Bridges in the Bit-Cell Matrix

5.1.1 Test Analysis Results

Resistive bridge defects, in which two signal conductors are shorted, is the most common defect type in current processes. Figure 5.1 shows an excerpt of the FIFO bit-cell matrix where possible locations for bit-cell bridge defects have been marked. The automatic defect-based fault extraction method based on the parasitic capacitances gave us a total of 18 unique resistive bridge faults within a nominal bit-cell and between the cell and its closest neighbors.

Table 5.1 shows the results of the defect-based analysis for these faults for the initial INTEST procedure. Even in the case of hard ($100\ \Omega$) bridges, one of the 18 faults was not detected, leading to an unacceptably low fault coverage of 94.4%. For weaker bridges, the fault coverage was even lower.

Shorts to the power lines causing stuck-at and transition fault behavior are easily detected by the original INTEST procedure. The alternating bit values that pass through the bit cells during Step 2 and 3 will be disturbed by the stuck-at or transition fault, and henceforth detected upon read-out.

The detection of coupling faults (faults connecting two bit-cells) is more complex. The data values in physically adjacent neighbor cells determine whether a coupling fault is sensitized or not. The idea of the “all-zero” and “all-one” words written in Step 2 of the initial INTEST procedure is to create a physical checker-

Figure 5.1. Possible locations of bit-cell bridge defects in a bit-cell.

Bridge Resistance	Unweighted Fault Coverage	Weighted Fault Coverage
100 Ω	94.4%	99.2%
1 k Ω	94.4%	99.2%
10 k Ω	83.3%	67.9%
50 k Ω	44.4%	46.5%
100 k Ω	0.0%	0.0%

Table 5.1. Fault coverage of resistive bridges for the initial FIFO InTEST procedure.

board pattern in the bit and bit-not conductors of the cells. This is based on the fact that in the two layers of the bit-cell matrix (Poly-Silicon and Metal 1), the words are laid out in an alternating pattern of bit and bit-not conductors. The defect-based analysis showed us that the checkerboard pattern was insufficient to detect all faults, as one bridge fault led to a coupling fault that escaped the initial INTEST procedure. The missed fault is a bridge between two neighboring bit-not conductors in the Metal 1 layer within one word. Figure 5.2 shows the location of the potential coupling between bit-not conductors in neighboring bit-cells within a word.

Figure 5.2. Missed bridges between bit-not (DN) conductors within a word. Words are horizontal.

The analysis also showed possible bridges between bit and bit-not conductors in two different words. These are detected by the initial INTEST even though both conductors have the same value in the steady state. The FIFO transfers data by copying it from one word to another. This implies that at some point, two adjacent words will hold the same data. When the data is the same a bridge between a bit-conductor and a bit-not conductor is sensitized and the fault is detected. It is not necessary to fill the FIFO completely with all ones or all zeros as is often done with memories. Bridges between bit and bit-not conductors in successive words are sensitized and detected by the alternating word pattern.

In order to detect the missing bridge fault between bit-not conductors in neigh-

boring bit-cells within a word, a second test was devised. It is equal to the initial INTEST procedure, but with an additional data background. The requirement on the new data background is that the bit-not conductors within a word should be written to different logic values. This can be accomplished by writing words with an alternating bit pattern. The new test consists of a repetition of the initial test, with two different data backgrounds. It is listed in Table 5.2.

Step	Operation	#ops
1	Reset	1
2	Write $(00 \dots 0; 11 \dots 1)^{n/2}; 00 \dots 0$	$n + 1$
3	Shift-Out $(00 \dots 0; 11 \dots 1)^{n/2}$	n
4	Reset	1
5	Write $(01 \dots 0; 10 \dots 1)^{n/2}; 01 \dots 0$	$n + 1$
6	Shift-Out $(01 \dots 0; 10 \dots 1)^{n/2}$	n

Table 5.2. Modified FIFO INTEST procedure.

Table 5.3 shows the resulting, improved fault coverage for the modified INTEST procedure. The good news is that complete detection of all hard bridges of 100 Ω and 1 k Ω was obtained, and that also detection of weaker bridges of 10 k Ω and 50 k Ω has improved. However, the detection of weak bridges still requires improvement, especially since weak bridges of 100 k Ω are not detected at all yet.

Bridge Resistance	Unweighted Fault Coverage	Weighted Fault Coverage
100 Ω	100.0%	100.0%
1 k Ω	100.0%	100.0%
10 k Ω	88.9%	68.7%
50 k Ω	50.0%	47.3%
100 k Ω	0.0%	0.0%

Table 5.3. Fault coverage of resistive bridges for the modified FIFO INTEST procedure.

To improve the fault coverage of high resistive bridges, different stress conditions was used in the defect-based analysis by changing the supply voltage, temperature, and test frequency. Changing the temperature or test frequency did not have any significant effect on the measured fault coverage. The insensitivity to temperature was expected since the effect of bridges are not influenced much by changes in temperature. Since the FIFO is asynchronous internally, changing the external test frequency does not change the internal speed of the FIFO and hence, changing the test frequency does not cause a major change in the operation of the FIFO and does not improve the detectability for bridge defects in the bit-cell matrix. But low voltage testing significantly improves the detection of high resistance bridges. Lowering the supply voltage from (nominal) 1.2 V down to 0.9 V increased the unweighted fault coverage of bridges of 50 k Ω from 50% to

83% and for 100 k Ω bridges from 0% to 62%. Hence, the modified InTEST with two data backgrounds can be improved by running it at 0.9 V. Table 5.4 shows the test results for the low voltage test and Figure 5.3 shows the fault coverage improvements between the three tests.

Bridge Resistance	Unweighted Fault Coverage	Weighted Fault Coverage
100 Ω	100.0%	100.0%
1 k Ω	100.0%	100.0%
10 k Ω	88.9%	68.7%
50 k Ω	83.3%	68.3%
100 k Ω	61.1%	51.2%

Table 5.4. Fault coverage of resistive bridges for the modified FIFO InTEST procedure at 0.9 V.

Figure 5.3. Fault coverage of resistive bridges for three different tests.

5.1.2 Coupling Fault Analysis

Figure 5.4 shows two bit-cells connected by a bridge and how the voltage levels in the two affected nets change with different bridge resistances. The two nets are written to opposite logic values. The supply voltage is 1.2 V. Three different regions with different faulty behavior can be identified in the figure. For low bridge

resistances, the coupling between the nets is perfect and both nets are forced to the same voltage. The nets are always forced to zero since the NMOS pull-down transistors in the cells are stronger than the PMOS pull-up transistors. In this region the resistive bridge model corresponds to the classic coupling fault model. The second region gives rise to a more complicated faulty behavior that can not be described by the classic coupling fault model. The bit-cells keep their data, no cell flips, but one of the bit-conductors will have a voltage level substantially below the nominal logic-high voltage. In region two this voltage is too low to fully turn on the write transistor of the following cell. Hence, the data in the following bit-cell can not be changed. The bridge causes a kind of data dependent transition fault in that cell. Either the transition from zero to one or from one to zero, depending on which bit-conductor is affected, becomes impossible when the bridge is sensitized. In the third region the voltage level is still high enough to change the data in the following cell, but the write will be slower than normal since the write transistor is not fully turned on. Hence, in region three, there is a data dependent delay fault behavior.

The defect-based analysis identified possible bridges between the bit-not-conductors of two successive words. If the bridge resistance is in the perfect coupling region this kind of bridge can be detected by only writing two alternating words that ripples through the FIFO. The bridge fault detection for bridges in region two and three is a lot more complicated. It involves the data of three successive bit-cells in three successive words. If sensitized, the bridge between the first two cells can cause an insufficient voltage level in the bit-not-conductor of the second cell. If this is to be detected, the data in the third cell must be different from the data in the second cell. The data in the two data conductors connected by the bridge must be 0 and 1 with the 1 in the second bit-cell bit-not-conductor since only a “weak” 1 and not a “strong” 0 is possible because of the relative strength of the NMOS and PMOS transistors. To achieve this data configuration which enables detection of the bridge, the FIFO must be filled completely with a pattern that alternates from word to word such as the INTTEST pattern. For higher bridge resistances, filling the FIFO completely is also necessary to detect some bridges between the cells and the enable lines or power lines.

When the the supply voltage is lowered, the size of region one, the perfect coupling region, increases significantly. It can even extend well beyond the 50 k Ω mark. This means that if low voltage is used the more complicated fault behavior in region two and three becomes less important. However, the size of the perfect coupling region will also heavily depend on other parameters such as process conditions.

Figure 5.4 also shows that with the bridge resistance selection used, region two for the simulated process conditions is missed. This means that this type of behavior might not be detected. But faults in region three has the same prerequisites for detection as faults in region two. If faults in region three are detected the faults in region two will also be detected. Furthermore, region two is quite small compared to the other regions.

Figure 5.4. Resulting voltages in two bit-cells versus the bridge resistance of a bridge connecting the cells.

5.2 Resistive Opens in the Bit-Cell Matrix

All opens in a bit-cell were extracted by hand to get more accurate fault information than what is currently possible with automatic fault extraction in FAULTGEN. Figure 5.5 shows the layout of a bit-cell and all the conductor segments, vias, and contacts that are susceptible to open defects. One possible open defect location is indicated for each branch of a conductor, each via and each contact. Broken contacts or vias are thought to be the most probable. Based on the possible defects 27 open faults were identified as indicated by the numbers in the figure. Some conductor opens are electrically equivalent to some via and contact opens and they are combined into one fault. That is why some fault numbers occur more than once. Opens in all layers and the diffusion region have been included. For vias and contacts it is assumed that only the connection between the layers is broken and that the connection within the layers is still intact. The + sign means that the fault type is already included once and that this occurrence can be seen as belonging to an adjacent cell. Fault number 8 is an open in the gate of the N3 enable transistor. In the simulations it is seen as only affecting one single transistor which is not entirely true since a break in the G enable conductor should affect all the following bit-cells. However, this does not matter since the fault is easily detected and the effect of the fault is the same for all cells.

Open faults are localized to a single conductor, via or contact. They do not couple bit-cells together as bridges does. For opens there is no direct coupling between cells, but opens may cause nets to be floating, which makes them more

Figure 5.5. 27 bit-cell open faults.

susceptible to parasitic coupling. However, even if the effects of parasitic coupling is clearly visible in simulations, no case where the coupling between cells causes an open fault to be detectable was found.

The analysis started by applying the initial INTEST. Most of the 1 G Ω and 1 M Ω opens are detected by this test. To improve the fault coverage, the stress conditions (write frequency, supply voltage and temperature) were changed. It was found that changing the stress conditions did not increase the fault coverage of the chosen resistive opens when using the initial INTEST.

The operation of the asynchronous handshake protocol is independent from the data transfers in the bit-cell matrix. No information about if a data transfer is successful or not is sent from the matrix to the asynchronous control circuits. The time constraint on the minimum low time of the acknowledgment signal A [11] in the FIFO four-phase protocol is based on an assumption on the speed of data transfers in the matrix. This time constraint in the control-path implies a similar time constraint for the bit-cell matrix. Because of the asynchronous nature of the FIFO, changing the external write frequency does not influence the time constraints. If a data transfer is slowed down by a resistive open in the data-path, the time constraints of the bit-cell matrix may be violated, causing incorrect data to be latched. To detect violations of this time constraint it is sufficient to use a data background that alternates from word to word. If a data transfer from one cell to another is delayed long enough, old data will be latched. Because of the alternating data background of the initial INTEST this data will be different from the data that should have been latched, and the fault is detected.

When the FIFO is empty, all latches are in the transparent mode, they are all open. This means that data written to the FIFO will move through the bit-cell matrix in a wave-like fashion, independently of the asynchronous control. To make sure that a data transfer has successfully finished before the latch closes, the control slices has a built in timing margin for data transfers. All control slices have the same timing margin built into the protocol, but since the data moves in a wave-like manner, the actual time between the arrival of data at a word and the closing of the word latches will increase with an amount equal to the control timing margin for each stage. This means that during the write phase, the data timing margin that can be violated by an open in the data path will be different for each word. It will be smallest for the first word in the pipeline and then increasingly bigger up to the last word of the pipeline. However, during the read out process, there is no wave-movement of the data and the data timing margin is the same for all stages. The write time margin is smaller than the read time margin for the first few words, for the rest of the words, the read time margin is smaller. What all this means is that the first few words are more sensitive to delay faults and there will be a slightly increased fault coverage for these words. But because of the read time time constraint that is exercised for all words when data is read from a full FIFO, the difference in fault coverage between words is small. However, this further highlights the need to fill the FIFO completely to achieve maximum fault coverage.

In [3] it is shown that high-voltage testing increases the fault coverage of opens for an investigated SRAM. This should probably be true for the FIFO as well. But

when the FIFO was simulated at the maximum recommended operating voltage, 1.3 V, and also above maximum at 1.5 V, no increase of fault coverage was found. This could be caused by the very limited resistance resolution of only three different resistances. It is very possible that an increase in fault coverage would be seen if more resistances were investigated.

5.2.1 Gate Opens

Two of the undetected 1 G Ω open faults in a bit-cell are breaks in the gate of the NMOS transistors of the cross-coupled inverters in a bit-cell. Figure 5.6 shows an example of such a gate open fault. A hard open in the gate of a transistor substantially limits the current that can flow to and from the gate, but it is still possible to charge the gate capacitance through the open resistance, albeit slowly. An open gate also has an increased sensitivity to capacitive coupling.

Figure 5.6. A bit-cell with a resistive open at the gate of transistor N2.

The voltage level on the gate behind the open fault in an inverter NMOS can be changed by writing the cell to a certain value and then wait. After some time, the gate capacitance will be charged or discharged, through the open fault resistance, to the voltage level of the data conductor it is connected to. The charging/discharging time is long compared to the regular write and read speed. If the gate behind the open fault is initialized to a logic one, it will stay at this level even when the data in the cell changes, if the changes are fast. Because of the limited current that can flow through the open resistance it takes a comparatively long time to charge or discharge the gate node. This situation can be seen in the first 150 ns of the graphs in Figure 5.7. The top graph shows the voltage on a data conductor when the FIFO is filled with an alternating data pattern. The bottom graph shows the voltage on the gate behind the NMOS open resistance. As can be

seen in the Figure 5.7, the gate voltage level is affected by capacitive coupling and the gate is slowly charged when the data conductor is written to one. But neither of these effects are strong enough to significantly change the gate voltage. When all writes have been performed, the gate voltage is still so low that the NMOS transistor is turned off.

Figure 5.7. Voltage on the data conductor (upper curve) and on the broken gate conductor (lower curve) connected by a $1\text{ G}\Omega$ open resistance.

If the gate voltage stays low, the NMOS transistor will be turned off. Because data is always written to a bit-cell by forcing a data conductor to zero through the write transistors, the NMOS pull-down transistors in the cross-coupled inverters are never used to change the data in the cell, they are only used to keep the data stable. Even if the inverter NMOS transistor is turned off when it should be on, the cell will still keep its data. However, the cell will be susceptible to disturbances through parasitic coupling since the data conductor is floating. Because of the gate open fault it is also possible to have a logic high voltage on one of the NMOS gates even though the data conductor connected to the gate has a logic low voltage. In this case, both the NMOS and the PMOS of the inverter will be turned on at the same time. Because of the transistor sizing the NMOS transistor is stronger than the PMOS transistor. Hence, the NMOS will force the inverter output down to an incorrect logic zero. This makes it possible to detect the NMOS transistor gate open.

The first step in detecting the NMOS gate open in a bit-cell is to write the data conductor connected to the gate to a logic one and then wait until the NMOS gate has been charged to a logic one through the open resistance. According to the simulations this takes around $1.5\text{ }\mu\text{s}$ for a $1\text{ G}\Omega$ open resistance. When the NMOS gate has been charged, a logic zero is written to the data conductor connected to the NMOS gate. Normally the other data conductor should be charged to a logic one by the PMOS transistor, but since the NMOS transistor is still active because of the logic high voltage on the gate, and since the write time is too short

to change the gate voltage, the bit-cell will flip data when the latch is closed. The incorrect data can be read out of the FIFO to detect the fault.

Table 5.5 shows one possible way of fulfilling the requirements for detection. The FIFO is reset and then filled with a $(00 \dots 0; 11 \dots 1)^{n/2}$ data background and then there is a wait for a couple of μs . This is the first waiting period. There are two NMOS inverter gates in each bit-cell and after this waiting period half of them will be written to a logic one, even in the case of an open fault. The next step is to scan out one word and scan in a word of zeros. When one word is scanned out, all the data words in the FIFO move one stage forward in the pipeline, effectively writing the opposite data into the cells. This enables detection of the NMOS gate open fault for half of the NMOS inverter transistors in the bit-cell matrix. To detect this type of fault for the other half a second wait and a second scan of data is performed. This ensures that all the NMOS gate open faults have been sensitized. If there is a fault it will be detected when the data in the FIFO is read out and compared to the correct data.

Step	Operation	#ops
1	Reset	1
2	Write $(00 \dots 0; 11 \dots 1)^{n/2}$	n
3	Wait $2\mu s$	1
4	Shift-Out $(00 \dots 0)$; Shift-In $(00 \dots 0)$	1
5	Wait $2\mu s$	1
6	Shift-Out $(11 \dots 1)$; Shift-In $(11 \dots 1)$	1
7	Shift-Out $(00 \dots 0; 11 \dots 1)^{n/2}$	n

Table 5.5. FIFO bit-cell inverter gate open test procedure.

Table 5.6 shows the relative increase in fault coverage achieved by the gate open INTEST. For 1 G Ω opens the fault coverage increased by 9%. The gate open fault can not be detected by voltage testing for open resistances of 10 k Ω and 1 M Ω since, for those resistances, the NMOS gate is discharged too quickly to enable detection.

Open Resistance	Initial Fault Coverage	Improved Fault Coverage
10 k Ω	18	18
1 M Ω	92	92
1 G Ω	92	100

Table 5.6. Relative fault coverage improvement of resistive opens for the gate open test procedure. 100 is maximum.

In the current version of the proposed DfT-hardware it is not possible to control the data that is shifted into the FIFO in step 4 and 6 since that data is provided by scan chains. This means that some fault coverage will be lost for the first two words of the FIFO. The good thing about this test procedure is that it is possible to combine it with the INTEST for bridges.

Another method to detect this type of fault that requires fewer writes, is to reset the FIFO, write one word of zeros, reset the FIFO again, wait, write one word of ones, read it out, wait, write one word of zeros. But since the vast majority of the testing time is spent in the waiting periods, saving a few writes probably has a quite small impact on the total test time.

The NMOS gate open can only occur if there is a break in the short piece of the data conductor connected to the gate of the NMOS, see Figure 5.5. No broken contact or via can cause this fault. This means that a single NMOS gate open fault in specific bit-cell probably is very unlikely, but since the bit-cell is repeated many times, the total probability of having a NMOS gate open fault somewhere in the bit-cell matrix is a lot higher. But still, the NMOS gate open fault is probably unlikely and it should be considered if it is really necessary to perform the test with the two waiting periods, especially since the time cost of this test is high.

5.3 Asynchronous Control

There are 16 asynchronous control slices in the 16×19 FIFO. The slices are laid out in eight rows of two slices each. The first slice interfaces with the write circuit and the last slice interfaces with the read circuit. All slices are exactly the same internally, but they have a different neighborhood. Because of the different neighborhood there will be some small differences between slices for bridges between the slices and the neighborhood. Based on the window technique principle, the simulations were focused on a typical control slice in the middle of the FIFO. To verify that the window assumption was correct, some simulations were also performed on the other slice in the same row as the typical slice and on slices in the first and last rows.

Testing the control slices poses some problems since it is not possible to directly control the slice inputs or to directly observe the outputs of the slices, which is possible for the data inputs and outputs of the bit-cells. The asynchronous control slices can only be controlled by doing writes and reads, and the output of the slices can only be indirectly observed through the FIFO outputs (the data outputs and the **Full** and **Empty** flags). Hence, the controllability and observability is limited which makes testing more difficult than for the bit-cells.

5.3.1 Bridges

The automated bridge fault extraction with FAULTGEN generated many more bridge faults for the typical control slice than for a typical bit-cell. Fault simulations with the initial INTTEST resulted in a weighted fault coverage of 98% for 100 Ω bridges.

A thorough analysis of all undetected faults was done to try to find a way to improve the fault coverage. This proved difficult because of the limited controllability and observability.

One undetected bridge was a bridge between the control slice and an internal read signal that signals from the read circuit to the write circuit that a read is in progress. The fault causes the full flag to become low when the write enable is

de-asserted, even though the FIFO is full. The fault was not detected by the initial INTEST since the write enable is de-asserted at the same time as the stream-mode is activated, and in stream-mode the full flag is always low. By making sure that the write enable is low one clock cycle before the stream-mode is activated, this fault could be detected.

By lowering the supply voltage, one additional 100 Ω bridge, was detected. Because of the limited controllability and observability, no way to modify the test to further increase the fault coverage was found.

With the small modification mentioned above and low voltage testing, the resulting weighted fault coverage is 98.4%. To further increase the fault coverage it is probably necessary to build some kind of DfT solution. Another possible solution may be to redo the layout to prevent the undetected bridges from occurring or making them even more unlikely.

The fault mechanisms in the control slices do not depend on the data pattern, but the pattern is still important for detecting errors. Some faults causes data in the following cell to be overwritten. To detect this an alternating data pattern like the one used in the initial INTEST is needed. One fault causes a word to “disappear”. The data can pass through the word, but no data can be written to it. It appears as if though the word does not exist. The FIFO seems to be one word shorter. Writing the FIFO full ensures that this is detected.

5.3.2 Opens

As discussed earlier there is no NXP tool support for making a complete automatic open fault extraction. Because of the size and complexity of the control slices, making a manual open extraction as was done for the bit-cells would be very tedious and time consuming. Therefore, opens in the asynchronous control slices were automatically extracted by FAULTGEN based on the transistor schematics. One open is generated for each terminal of each transistor. This does not give a complete picture of the open fault coverage since opens that affect two or more transistor are left out and some open faults that can not occur in reality may be included. Usually, opens that affect only one transistor terminal are the most difficult to detect so studying only those opens gives a kind of worst case scenario.

From the simulations it can be noted that opens in the cross-coupled inverters are not easily detected. The reason for this is the same as for the inverters in the bit-cell: The inverters are only used to keep the data and not to switch the state of the keeper. Some of the very wide transistors in the control slices are split up into two transistors connected in parallel. An open in one of these transistors is not easily detected either, since the other transistor will still be functioning. The driving strength will be lower which may cause a problem. But these transistors have a lot of redundant contacts so even if one of the contacts is open the transistor will operate according to specification. The probability for having opens in several contacts is very small.

5.4 Combining the Tests

The tests for resistive bridges and opens uses the same data background and it possible to combine the tests. The test for opens presented in Table 5.5 includes the data background of the initial INTEST and it will detect the bridges detected by this test. By adding the additional data background from the test in Table 5.3 a new combined test with the same fault coverage of bridges and opens as the tests in Table 5.5 and 5.3 can be created. But low-voltage testing as is used in the test in Table 5.4 to increase the fault coverage of bridges will decrease the fault coverage of opens. To achieve the maximum fault coverage for opens high-voltage testing should be used. For these reasons the tests should not be combined if low- and/or high-voltage testing is used.

Chapter 6

Conclusion

6.1 Summary

Many IC designs use numerous embedded FIFO memories for intermediate storage, data rate conversion, and clock domain crossing. The usage of embedded FIFOs is expected to grow, as new design paradigms such as Network-on-Chip (NOC) and Globally-Asynchronous Locally-Synchronous (GALS) use FIFOs extensively. NXP Semiconductors has developed a new embedded asynchronous FIFO module, based on an asynchronous micropipeline architecture. Due to the full-custom design, the new FIFO is substantially smaller and faster than SRAM-based and standard-cell-based counterparts.

In this report, it has been described how the fault detection qualities of the initial test procedure for resistive bridges and opens in the bit-cell matrix and in the asynchronous control of the NXP asynchronous FIFO has been investigated and improved.

By including an additional data background 100% fault coverage was reached for 100 Ω and 1 k Ω bridges in the bit-cell matrix. The fault coverage of higher resistance bridges was increased substantially by including low-voltage testing. The probability of high resistance bridges is a lot smaller than the probability for low resistance bridges so achieving 100% fault coverage here is not as important.

Two hard 1 G Ω opens in the gates of the NMOS transistors of the cross-coupled inverters in the bit-cells that were not detected by the initial test were found. A test procedure with two waiting periods was created to detect them.

Achieving 100% fault coverage of 100 Ω bridges in the control slices proved difficult because of the limited controllability and observability. The fault coverage was improved by making a small modification in the initial test procedure and by including low-voltage testing. It was found that the fault coverage could not be further increased by modifying the test patterns. To increase the fault coverage it would be necessary to either change the layout or to include some kind of DfT-solution.

It may be possible to increase the fault coverage of opens in the asynchronous control by including high-voltage testing or maybe a different test procedure with

waiting periods, but there was not enough time to investigate this thoroughly.

6.2 Future Work

To get the complete picture of the fault coverage, fault analysis needs to be done for the write and read interfaces, parts that were not fully investigated in this project. The integration of the test and DfT approach should be validated in the CAT system flow. The final step is to validate the test suite on real silicon.

Bibliography

- [1] Mohamed Azimane et al. A New Algorithm for Dynamic Fault Detection in RAMs. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 177–182, May 2005.
- [2] Mohamed Azimane and Ananta K. Majhi. New Test Methodology for Resistive Open Defect Detection in Memory Address Decoders. In *IEEE VLSI Test Symposium (VTS)*, pages 123–128, April 2004.
- [3] Ananta K. Majhi, Mohamed Azimane, Guido Gronthoud, Maurice Lousberg, Stefan Eichenberger, and Fred Bowen. Memory testing under different stress conditions: An industrial evaluation. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 438–443, March 2005.
- [4] W. Maly, A.J. Strojwas, and S.W. Director. VLSI Yield Prediction and Estimation: A Unified Framework. *IEEE Transactions on Computer Aided Design*, CAD-5(1):114–130, January 1986.
- [5] Rosa Rodríguez Montañés, Eric Bruls, and Joan Figueras. Bridging Defect Resistance Measurements in a CMOS Process. In *Proceedings IEEE International Test Conference (ITC)*, pages 892–899, September 1992.
- [6] Rosa Rodríguez Montañés, José Pineda de Gyvez, and Paul Volf. Resistance Characterization for Weak Open Defects. *IEEE Design & Test of Computers*, 5(19):18–26, SeptemberOctober 2002.
- [7] W. Needham et al. High Volume Microprocessor Test Escapes – An Analysis of Defects Our Tests are Missing. In *Proceedings IEEE International Test Conference (ITC)*, pages 25–34, 1998.
- [8] Manoj Sachdev. *Defect Oriented Testing for CMOS Analog and Digital Circuits*. Kluwer Academic Publishers, 1998. ISBN 0-7923-8083-5.
- [9] J.P. Shen, W. Maly, and F.J. Ferguson. Inductive Fault Analysis of MOS Integrated Circuits. *IEEE Design & Test of Computers*, 2(6):13–26, December 1985.
- [10] Charles E. Stroud, John M. Emmert, James R. Bailey, Khushru S. Chhor, and Dragomir Nikolic. Bridging Fault Extraction from Physical Design Data for

- Manufacturing Test Development. In *Proceedings IEEE International Test Conference (ITC)*, 2000.
- [11] Paul Wielage, Erik Jan Marinissen, Michel Altheimer, and Clemens Wouters. Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO. In *Proceedings Design, Automation, and Test in Europe (DATE)*, April 2007.

Appendix A

Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO

Design and DfT of a High-Speed Area-Efficient Embedded Asynchronous FIFO

Paul Wielage^{1*}

Erik Jan Marinissen¹

Michel Altheimer^{2**}

Clemens Wouters³

¹ NXP Semiconductors
Research - Digital Design & Test
High Tech Campus 48, M/S-02
5656AE Eindhoven
The Netherlands
paul.wielage@nxp.com
erik.jan.marinissen@nxp.com

² NXP Semiconductors
Digital Library Technology
505, Route des Lucioles
Sophia Antipolis, 06560 Valbonne
France

³ NXP Semiconductors
Digital Library Technology
High Tech Campus 46, M/S-11
5656AE Eindhoven
The Netherlands
clemens.wouters@nxp.com

Abstract

Embedded First-In First-Out (FIFO) memories are increasingly used in many IC designs. We have created a new full-custom embedded ripple-through FIFO module with asynchronous read and write clocks. The implementation is based on a micropipeline architecture and is at least a factor two smaller than SRAM-based and standard-cell-based counterparts. This paper gives an overview of the most important design features of the new FIFO module and describes its test and design-for-test approach.

1 Introduction

Embedded First-In First-Out (FIFO) memories are increasingly used in IC design for intermediate storage, data rate conversion, and clock domain crossing. New design paradigms like Network-on-Chip (NOC) and Globally-Asynchronous Locally-Synchronous (GALS) use embedded FIFOs extensively [1, 2]. Despite the fact that a single individual FIFO is not very large, the number of FIFOs in a circuit can be huge, and hence their overall silicon area contribution can be significant. Consequently, it is important that they are implemented in an area-efficient way.

Conventional FIFO designs are based on random access memory for storing the data, e.g. an embedded SRAM or one made up entirely from standard-cell logic. SRAM-based FIFOs offer an area-efficient bit-cell, but small instances suffer from a relatively large SRAM periphery (such as address decoders and sense-amplifiers) in addition to the FIFO control overhead. Standard-cell-based FIFOs omit any kind of analog circuitry, but require a relatively large area to store a single data bit.

This paper describes the design of a new full-custom FIFO module which is now part of the NXP design library. It is a micropipeline, consisting of a series of asynchronously communicating stages, each implemented as a register of latches and a control cell. The FIFO provides area-efficient data storage, a small control path, low energy consumption, and high-speed operation. The latter is achieved by implementing a modified version of the conventional four-phase handshake protocol for communication between the various stages of the micropipeline.

We also describe the Design-for-Test (DfT) approach for this FIFO. The FIFO has a dedicated test, but unlike what is common for most embedded non-logic modules such as memories, we do not equip our FIFO with a full test wrapper or Built-In Self Test (BIST). Instead, dedicated DfT hardware enables the application of test stimuli to multiple (or all) FIFOs in parallel, while for further test access, the FIFOs are integrated into the on-chip scan chains.

The remainder of the paper is organized as follows. The FIFO design is described in Section 2 and its test and DfT approach is detailed in Section 3. The paper is concluded in Section 4.

2 FIFO Design

The architecture of the FIFO design is depicted in Figure 1(a). The FIFO consists of a write interface, a read interface, and a core. The core is an asynchronous micropipeline [3]. The pipeline consists of a chain of stages, where each stage can hold at most one word of data. By means of local handshake signaling between the stages data propagates through the pipeline in a self-timed fashion. The employed four-phase handshake protocol takes care of a fast and coherent movement of data from one stage to the next stage provided the receiving stage is ready to accept data. Data that arrives at the read interface of the FIFO can be read out from the pipeline. Basically, a read operation frees the last stage of the pipeline, allowing all data to shift one position towards the output. However, since the read operation creates only one free position, the whole content is not shifted instantaneously, but word by word. So it looks like the read operation creates a ‘hole’ which ripples from the read side to the write side interface

* Paul Wielage is currently with NXP Semiconductors’ IC Laboratory in Eindhoven, The Netherlands.

** Michel Altheimer is currently with NXP Semiconductors in Crolles, France, e-mail:michel.altheimer@nxpcrolles.st.com.

in a direction opposite to the data. When the FIFO is completely full at the moment of a read operation, this hole must ripple all the way to the first stage before writing becomes possible again. In short, the handshake mechanism causes the data to line up at the read-side of the pipeline and, consequently, the empty positions at the write-side.

The read (write) interface of the FIFO module converts the internally employed asynchronous handshake protocol into a read (write) enable signal and an empty (full) status flag that are synchronous to the read (write) clock. This allows the FIFO to be smoothly integrated in a synchronously clocked context, possibly with different clock signals for the write and read interfaces.

The FIFO core is detailed in Figure 1(b). Each stage consists of a data register and a stage controller (SC). The data register is built of latches, with one latch per bit of a word. The stage controller determines whether the register is in hold (full) or transparent (empty) mode by means of signal $G_i = 0$ and 1 respectively. The communication between the stages is via channels based on the concept of *bundled data* [4, 5]. In here, the data is encoded on a bus of bit signals and presence resp. acceptance of valid data on the data bus is encoded by two signals R_i and A_i according to a four-phase handshake protocol. In Figure 1(b) the data busses are depicted by the big arrows.

Figure 1: The FIFO module (a) and its internal asynchronous core (b).

Other micropipeline implementations which exploit bundled data are MOUSETRAP [6], IPCMOS [7], GasP [8], $LP_{SR2/1}$ [9], and LP_{HC} [10]. Our implementation is different in that it (1) employs an optimized data latch consisting of seven transistors and (2) provides very fast propagation of holes from the read to the write interface, which improves the slowest timing arc of a micropipeline design significantly.

The remaining subsections describe the data path, the control path, and the write and read interfaces in more detail.

2.1 Data Path

The schematic of the data path latch is given in Figure 2. Transistors P1, P2, N1, and N2 form two cross-coupled inverters for keeping the state. Transistors N3, N4, and N5 allow changing the state. The bits in the data path are dual-rail encoded, i.e., both the bit value and its inverse are stored. This allows to employ an SRAM-like write technique in the transparent mode of the latch. To that end, transistor N5 must be turned on by a high level on G such that the true or the complement signal of the cell is forced to ground, depending on the data input D (and its complement DN).

Figure 2: The static latch as used in the data path.

2.2 Control Path

Transfer of data between two successive stages in the pipeline is based on a four-phase handshake protocol. The typical way of four-phase handshaking is shown in Cycle 1 of Figure 3, including the dashed arcs. Since we choose for the early data-valid scheme, the rising edges of R_i and A_i , at the end of Phases 1 and 2, indicate presence resp. acceptance of data on the channel [4]. With typical four-phase handshaking, R_i and A_i return to zero as shown in Phases 3 and 4, where the first transition is a condition for the second. This strict ordering guarantees that both the request and acknowledgement have been seen by the other side.

Figure 3: Three possible cycles of the modified 4-phase handshake protocol.

To improve the propagation speed of both data words and holes, we have implemented an alternative four-phase protocol, which is one without the dashed arcs. Now, constant data transfer is guaranteed when two conditions are met. Firstly, $\tau_3 < \tau_1$ which implicitly implements arc (a) and, secondly, the stage controller is (level) sensitive to R_i only when A_i is low which makes arc (b) superfluous. The difference with standard four-phase handshaking is depicted in Cycle 3. In particular, it shows that the modified protocol allows a new request before de-assertion of a previous acknowledgement. Given this protocol, the relation between the request signal to the next stage, the acknowledgement signal to the previous stage and G_i becomes very

simple: $R_i = A_{i-1} = \overline{G_i}$, provided that G_i becomes low after an incoming request event and immediately high after an incoming acknowledgement event.

The danger of omitting the dashed arcs is two-fold: unintended data duplication and deadlock. The first problem is solved by the timing constraint $\tau_3 < \tau_1$. The second problem is solved by setting a lower bound on τ_2 . Furthermore, the latter delay constraint also guarantees that data can be captured robustly in the latch register. By introducing these timing constraints, we effectively change the controller from speed-independent into self-timed [5].

Our modified four-phase handshake protocol is summarized in Table 1. Given the previously introduced encodings for R_i and A_i , data transfer happens during Phase 2. During this phase (1) valid data is present since $G_i=0$ and (2) the receiving stage is empty since $G_{i+1}=1$. In the first and fourth phases, the channel is waiting for these two conditions to become true. Since both events can happen in arbitrary order, Phase 1 has two possible encodings. Finally, Phase 3 is devoted to the immediate de-assertion of R_i to prevent duplication of data as explained.

Phase	R_i / A_i	Action
1	00 or 11	wait for the second data transfer condition
2	10	transfer of data
3	11	de-assert request signal
4	01	wait for a first data transfer condition

Table 1: Modified four-phase protocol with encoding.

To conclude, the proposed handshake protocol allows concurrent signaling of data presence and acceptance of data on the channels between the stages and this contributes to high-speed operation.

2.3 Read and Write Interfaces

The FIFO supports three basic operations: (1) *write*, in which data is written into the FIFO under control of wr_clk , we , and $full$, provided it is not full, (2) *read*, in which data is read out of the FIFO under control of rd_clk , re , and $empty$, provided it is not empty, and (3) *shift*, in which all data words in the FIFO are shifted one stage towards the output. While the write and read operations can be performed truly asynchronous, the shift operation, which is actually a combined write and read operation, only works if wr_clk and rd_clk are synchronous to each other. The shift operation can be used when the FIFO is embedded in a synchronous context, or when the two clock signals are explicitly made synchronous. The latter is the case in our test mode, as described in Section 3.

The write interface implements four functions: (1) conversion of the synchronous timing convention at the interface into the handshake timing convention of the FIFO core, (2) conversion of single-rail data inputs into dual-rail, (3) capturing the data bus at the rising clock-edge, such that intermediate transitions on the data bus in between write operations are blocked, and (4) enabling a shift operation even when the FIFO is full. The write interface is equipped with an additional latch register. This supports Function (3) by allowing the first register stage to effectively behave as a master/slave flip-flop. If signal $STREAMMODE$ is asserted, the same register implements Function (4). During shift operation it can be used as an extra word location to temporarily store a written word while a read-created hole is still making its way towards the write interface.

The read interface implements four similar functions: (1) conversion of the synchronous timing convention at the module interface into the handshake timing convention of the FIFO core, (2) buffering of the data output bus, (3) providing a flush mechanism to reset the FIFO to its empty state, and (4) support of the stream mode. The reset function is implemented as a sequence of asynchronous read operations for as long as the FIFO contains data and the reset is enabled. The support of the stream mode is simply a feed-through of the re signal to the write interface.

2.4 Layout

Three FIFO instances have been developed in a 90 nm CMOS technology. Table 2 lists the three instance sizes and corresponding silicon area, and compares that with SRAM-based and standard-cell-based alternatives.

Words $\times$ Bits	Silicon Area including DfT				
	SRAM		Standard Cell		New
	(μm^2)	($\Delta\%$)	(μm^2)	($\Delta\%$)	(μm^2)
16 $\times$ 19	10,869	+278	12,580	+338	2,875
32 $\times$ 37	20,043	+206	35,814	+446	6,554
64 $\times$ 37	25,421	+122	62,319	+444	11,453

Table 2: Currently available FIFO instances and their area sizes.

The layout of the 16 $\times$ 19-bit instance is shown in Figure 4. The largest layout block is formed by the memory-cell matrix. It contains 17 words (horizontal rows in the figure) of 19 bit cells each; 16 regular words plus the additional latch register that is enabled in the stream mode. The orientation of all memory cells is equal; hence data signal D of one bit is physically next to data-not $\overline{DN}$ of an adjacent bit in the same word. Power lines are routed left and right of the memory-cell matrix, as well as through the middle. The second-largest layout block is formed by the asynchronous stage control cells. The control cells are laid out in eight horizontal rows of two control cells each. The height of one row of control cells corresponds to the height of two rows of memory cells. A similar layout set-up works for all three FIFO instances, as they all have an even number of words. Other layout blocks implement the input and output buffers and write and read interfaces.

Figure 4: The layout of the 16 $\times$ 19-bit FIFO instance.

3 FIFO Test and DfT Approach

3.1 Prior Work in FIFO Testing

Conventional FIFO designs are based on either an SRAM or standard cells. SRAM-based FIFOs are typically tested with a dedicated BIST per FIFO as in the approaches by Barbagallo et al. [11], Van de Goor et al. [12], and Zorian et al. [13]. However, for small FIFOs, the corresponding DfT area is relatively large; a dedicated BIST would take 130% for our smallest $16 \text{ word} \times 19 \text{ bits}$ FIFO. This becomes prohibitively expensive in case a large number of FIFOs are used in one IC design. In order to reduce the DfT area costs, Grecu et al. [14] propose to use one common BIST controller and stimulus generator for multiple FIFOs, while only the response evaluator is dedicated per FIFO. To be efficient, this approach requires all FIFOs to be of the same size, which unfortunately is rather uncommon. Also, the area saved by sharing controller and stimulus generator might be spent again for the wiring between the shared BIST and the various FIFOs [15].

Two different approaches exist to test standard-cell FIFOs. The first approach is to make the latches or flip-flops that constitute the FIFO memory scan-testable and integrate them into the on-chip scan chains. The associated DfT area costs are high, but the FIFO can now be tested by ATPG-generated patterns as integral part of the IC's logic. The second approach is to treat the FIFO as a separate module that requires a dedicated test. Rearick [16] follows this second approach and uses, in a pre-test-wrapper era, scan chains surrounding the FIFO to provide test access; while test generation for embedded FIFO and surrounding logic is separate, the actual execution of both tests is merged into one. A potential drawback of this approach is that there might be logic gates between the embedded FIFO and its surrounding scan chains, that will complicate test generation and access.

Shi et al. [17] describe standard-cell based asynchronous FIFOs and propose a test method based on ATPG that targets both stuck-at and delay faults in the micro-pipeline. As DfT structure, a two-input multiplexer is added at each pipeline stage. This seems an expensive solution, but area costs are not mentioned. The paper also does not discuss if embedded FIFOs are wrapped for test, and how test access to them is achieved.

Van de Goor et al. [12] discuss on a high level defects and fault models. However, actual defects such as shorts and opens are difficult to model at high level. Our test solution, as explained below and in companion paper [18], aims at identifying both hard and weak shorts and opens.

3.2 Modular Test with Partial Test Wrapper

Non-logic modules embedded into an environment consisting of digital standard-cell logic are traditionally tested as separate units by dedicated tests; this was already the case even before the terms ‘SOC’ and ‘core-based testing’ were invented [19]. Examples of such non-logic embedded modules are custom-designed blocks like memories, analog, and mixed-signal blocks. The reason for testing them as separate units is that these modules have circuit structures different from standard-cell logic, which consequently exhibit different defect mechanisms and fault behavior. Our FIFO is also a custom-designed hard macro, and hence, we adopted from the start of the project the approach to use a dedicated test.

The common design-for-test approach that enables separate testing of embedded modules is to encapsulate the module with a *test wrapper* [20]. The test wrapper switches between the functional and test connections of the module. It supports at least three modes: (1) normal mode, in which the functional connections are enabled and the test infrastructure is switched off, (2) INTEST mode, in which the module is tested internally while external influences are blocked, and (3) EXTEST mode, in which the test wrapper participates in testing the circuitry external to the module while blocking signals internal to the module. Test wrappers with this functionality are nowadays quite common [21, 22] and have been standardized as IEEE Std. 1500 [23, 24]. For embedded memories, it is quite common to extend the functionality of the test wrapper with on-chip stimulus generation and response evaluation capabilities, commonly termed *Built-In Self Test* (BIST) [15].

For our FIFO, we decided *not* to implement a full test wrapper. A first-order approximation showed that the silicon area required for implementing only the data portion of such a wrapper (i.e., excluding the Wrapper Instruction Register [23]) for the smallest 16×19 -bit FIFO would in 90 nm CMOS technology take about $1,500 \mu\text{m}^2$. Compared to the size of the FIFO, $2,500 \mu\text{m}^2$, this implies an area cost of well over 50%! The principal reason of existence for our full-custom FIFO is its small area size, compared to implementation alternatives based on SRAM or standard cells. With a conventional IEEE Std. 1500 compliant test wrapper, the FIFO's area advantage would be completely destroyed. The *absolute* sizes of the FIFO and its wrapper are small, and hence, these area costs are bearable in case only one or few wrapped FIFOs are integrated into an SOC design. However, the intended use of the FIFO is especially for SOC designs in which several hundreds of FIFOs are integrated, and in that scenario, the area costs of equipping the FIFOs with a full-fledged test wrapper would become disproportionately expensive.

Instead of implementing a full test wrapper, we implemented only a partial wrapper [25]. We used wrapper cells like IEEE Std. 1500 type WC_SDL_COI [26, 23] to provide both controllability and observability to FIFO inputs `we`, `re`, `streamMode` and `reset` and FIFO outputs `full` and `empty`. Per IEEE Std. 1500, clock signals `wr_clk` and `rd_clk` are exempt from having wrapper cells.

The write and read data ports `wr_data` and `rd_data` are *not* equipped with wrapper cells. This cuts most of the area costs of a full wrapper and saves 38 (for the smallest FIFO instance) to 74 (for the larger FIFO instances) wrapper cells. A Wrapper Instruction Register (WIR) and associated Wrapper Serial Control (WSC) port, both mandatory per IEEE Std. 1500, are *not* supplied either.

3.3 FIFOs Integrated in Scan Chains

During its shift operation, i.e., when simultaneous write and read operations are applied, an $n \times m$ -bit FIFO (i.e., a FIFO consisting of n words of m bits each) filled with i words ($1 \leq i \leq n$) actually behaves as m parallel scan chain segments of length i bits each. For test, we exploit this fact by including all FIFOs into on-chip scan chains. In this way, the FIFO's write- and read-data inputs become fully controllable and observable from chip pins, and hence are compensated for their lacking wrapper functionality.

Inclusion into the on-chip scan chains requires an $n \times m$ -bit FIFO to be equipped with an m -bit wide scan-in port `si[m-1:0]`, an m -bit

wide scan-out port $so[m-1:0]$ (tapped off from the rd_data output), and a scan enable control input se . Note that the FIFO requires a filling of at least one word (i.e., to be non-empty) in order to be able to shift data. As described in Section 2.3, signal $streamMode$ should be asserted to be able to shift a full FIFO, i.e., a FIFO with a filling of n words.

The on-chip SOC-level TestRail architecture design [21] is not constrained by the inclusion of the FIFO scan chain segments. The SOC integrator can freely design a TestRail architecture, as long as all FIFO scan chain segments are connected into one or more scan chains. It is possible to create dedicated ‘FIFO scan chains’, but it is also allowed to mix and match with regular scan chain segments that run through the surrounding standard-cell logic. It is possible to concatenate the scan chain segments of the FIFO(s) into any smaller number k ($k \geq 1$). This allows our approach to be used even for SOC’s which allow only very few scan chains, such as very-low pin-count SmartCard ICs.

3.4 FIFO InTest Procedure

For an $n \times m$ -bit FIFO (with n even), our INTEST procedure consists of three steps, as listed in Table 3. Note that the reset, write, and shift operations are all normal functional modes of the FIFO.

Step	Operation	#ops
1	<i>Reset</i>	1
2	<i>Write</i> ($00 \dots 0; 11 \dots 1$) $^{n/2}; 00 \dots 0$	$n + 1$
3	<i>Shift-Out</i> ($00 \dots 0; 11 \dots 1$) $^{n/2}$	n

Table 3: FIFO INTEST procedure.

Upon start-up of the SOC, the FIFO is in an unknown state. In order to bring it to a known state, Step 1 resets the FIFO by applying $reset = 1$ for one clock cycle, thereby effectively flushing the FIFO’s content.

Step 2 tests whether the FIFO correctly performs write operations at different levels of being filled. At every distinct fill level, a write operation is confronted with different delay paths, which need to be within certain margins for the FIFO to operate correctly. In order for Step 2 to cover all fill levels, it is important that only write operations are executed, and no read (or shift) operations. In this way, we cover all fill levels, from empty to full. For an n -word FIFO, we write $n + 1$ words, in order to let the last write operation test if the FIFO indeed refuses to write additional data, once full. The alternating sequence of $00 \dots 0$ and $11 \dots 1$ words enables to distinguish between neighboring words of the FIFO. In the layout, each word is in its turn an alternating sequence of bit and bit-not wires, such that the test data fills the FIFO with a physical *checkerboard* pattern. Next to the delay paths at different fill levels, this test also aims to detect single-cell stuck-at and transitions faults, as well as coupling faults between neighboring cells.

In Step 3, we read out the FIFO content by shifting. Shifting a full FIFO actually exercises the worst-case timing paths in the FIFO, as for every individual shift operation a hole needs to be propagated from the read interface to the write interface through the entire FIFO. The FIFO data is shifted to the chip pins, where the ATE compares actual with expected responses. Note that depending on the actual chip-level scan chain configuration (see Section 3.3), the responses scanned out of the FIFO might require propagation through other FIFOs and/or

logic scan chain segments before actually reaching the chip pins.

3.5 Low-Cost ‘Filler’ DfT

In Step 2 of the INTEST procedure, we cannot perform the write operations at different fill levels by simply shifting all FIFOs together as part of the SOC scan chains. Write data could be provided via the scan chains only if the subsequent FIFOs execute and complete their respective Steps 2 one by one. This would require a test control infrastructure that allows us to subsequently activate the FIFOs. This could be implemented by means of a dedicated on-chip counter per FIFO or, alternatively, a dedicated control pin per FIFO; both solutions were considered prohibitively expensive. In a very-low pin-count SOC, such as a SmartCard IC, where we need to concatenate even the scan chain segments through a single FIFO, Step 2 of our test cannot be performed at all by means of scan access, as it would require bit i of the FIFO words to shift, while bit j is only written and does *not* shift (for $1 \leq i < j \leq m$).

To overcome the above difficulties, we have implemented additional write access DfT. This so-called *Filler* is a degenerate version of BIST, which is capable only of providing test stimuli, and only in a restricted way. It consists of three input bits to the FIFO, named $fillerData[1:0]$ and $fillerEnable$. When control input $fillerEnable$ is asserted, the words written into the FIFO are determined by $fillerData$, such that all even bits correspond to $fillerData[0]$ and all odd bits correspond to $fillerData[1]$. This Filler DfT gives us a restricted, but low-cost and efficient way to control the data written into a FIFO. Due to the restrictions imposed by the Filler, test stimuli for the FIFO can now be characterized and denoted by two bits only; one representing all even bits, and one representing all odd bits in the stimulus words.

The envisioned usage of the three additional Filler inputs at SOC integration level is that they will be connected up for all FIFOs per signal, allowing for a broadcast of the same Filler signals to all FIFOs simultaneously. Such a connection scheme would allow to carry out Step 2 of our test for all FIFOs in parallel. Steps 1 (reset) and 3 (scan-out) could already be executed in parallel for all FIFOs, such that now effectively the entire test can be carried out in parallel for all FIFOs.

Figure 5: The FIFO module and its test-specific terminals.

Figure 5 shows the FIFO module and highlights the additional test-specific inputs and outputs. They are $si[m-1:0]$, $so[m-1:0]$, and se for scan access, and $fillerData[1:0]$ and $fillerEnable$ for Filler access. The overall DfT area costs of our solution is 15%, 7.1%, and 4.1% for the three instances respectively.

4 Conclusion

Many IC designs use numerous embedded FIFO memories for intermediate storage, data rate conversion, and clock domain crossing. The usage of embedded FIFOs is expected to grow, as new design paradigms such as Network-on-Chip (NOC) and Globally-Asynchronous Locally-Synchronous (GALS) use FIFOs extensively. We have developed a new embedded asynchronous FIFO module, based on a micropipeline architecture. Due to its full-custom design, the new FIFO is substantially smaller than SRAM-based and standard-cell-based counterparts; the savings are 3.4 mm² and 7.3 mm² in 90 nm technology for 250 FIFOs of 32 words × 37 bits, respectively.

As hard macro, the FIFO is tested with a dedicated test. The initial test procedure is described in this paper, while a companion paper describes the process of analyzing and improving the detection qualities of this test [18]. Contrary to what is common for embedded hard macros, the FIFO is *not* equipped with a full test wrapper, in order to maintain its area advantage. Instead, the FIFO has a partial test wrapper, while the data path has been made controllable and observable during testing by integrating it into the regular on-chip scan chains. A dedicated ‘Filler’ interface allows restricted write access, which is sufficient to test multiple FIFOs in parallel. With our approach, the overall DfT costs range between 4.1% and 15% of the bare FIFO silicon area.

Acknowledgements

The authors thank Mohamed Azimane of NXP Research in Eindhoven, The Netherlands, and Tobias Dubois and Erik Larsson of Linköpings Universitet in Linköping, Sweden for working with us on the improvement of the FIFO test procedure [18]. Furthermore, we thank Erik van Geest of NXP Semiconductors and Ananta Majhi, Bart Vermeulen, and Kees Goossens of NXP Research in Eindhoven, The Netherlands for constructive criticism on an early draft of this paper.

References

- [1] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):21–31, Sept-Oct 2005.
- [2] Jens Muttersbach, Thomas Villiger, and Wolfgang Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 52–59, Eilat, Israel, April 2000.
- [3] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.
- [4] Ad M.G. Peeters. *Single-Rail Handshake Circuits*. PhD thesis, Eindhoven University of Technology, June 1996.
- [5] Jens Sparsø and Steve Furber, editors. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, Dordrecht, The Netherlands, September 2001.
- [6] Montek Singh and Steven M. Nowick. MOUSETRAP: Ultra-High-Speed Transition-Signaling Asynchronous Pipelines. In *Proceedings International Conference on Computer Design (ICCD)*, pages 9–17, Austin, TX, USA, September 2001.
- [7] S. Schuster et al. Asynchronous Interlocked Pipelined CMOS Circuits Operating at 3.3–4.5 GHz. In *Proceedings International Solid State Circuits Conference (ISSCC)*, pages 292–293, San Francisco, CA, USA, February 2000.
- [8] Ivan Sutherland and Scott Fairbanks. GasP: A Minimal FIFO Control. In *Proceedings Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 46–53, Salt Lake City, UT, USA, March 2001.
- [9] Montek Singh and Steven M. Nowick. High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapaths. In *Proceedings Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 198–209, Eilat, Israel, April 2000.
- [10] Montek Singh and Steven M. Nowick. Fine-Grain Pipelined Asynchronous Adders for High-Speed DSP Applications. In *Proceedings of the IEEE Computer Society Workshop on VLSI*, pages 111–118, Orlando, FL, USA, April 2000.
- [11] S. Barbagallo et al. A Parametric Design of a Built-In Self Test FIFO Embedded Memory. In *Proceedings IEEE Intl. Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 221–229, Boston, MA, USA, November 1996.
- [12] Ad J. van de Goor, Ivo Schanstra, and Yervant Zorian. Functional Test for Shifting-Type FIFOs. In *Proceedings European Design and Test Conference (ED&TC)*, pages 133–138, Paris, France, March 1995.
- [13] Yervant Zorian, Ad J. van de Goor, and Ivo Schanstra. An Effective BIST Scheme for Ring-Address Type FIFOs. In *Proceedings IEEE International Test Conference (ITC)*, pages 378–387, Washington, DC, USA, October 1994.
- [14] Cristian Grecu et al. Methodologies and Algorithms for Testing Switch-Based NoC Interconnects. In *Proceedings IEEE Intl. Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 238–246, Monterey, CA, USA, October 2005.
- [15] Rob Aitken. A Modular Wrapper Enabling High Speed BIST and Repair for Small Wide Memories. In *Proceedings IEEE International Test Conference (ITC)*, pages 997–1005, Charlotte, NC, USA, October 2004.
- [16] Jeff Rearick. Practical Scan Test Generation and Application for Embedded FIFOs. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–300, Atlantic City, NJ, USA, September 1999.
- [17] Feng Shi, Yiorgos Makris, Steven M. Nowick, and Montek Singh. Test Generation for Ultra-High-Speed Asynchronous Pipelines. In *Proceedings IEEE International Test Conference (ITC)*, pages 1009–1018, Austin, TX, USA, November 2005.
- [18] Tobias Dubois, Mohamed Azimane, Erik Larsson, Erik Jan Marinissen, Paul Wielage, and Clemens Wouters. Test Quality Analysis and Improvement for an Embedded Asynchronous FIFO. In *Proceedings Design, Automation, and Test in Europe (DATE)*, Nice, France, April 2007.
- [19] Frans Beenker, Karel van Eerdewijk, Robert Gerritsen, Frank Peacock, and Max van der Star. Macro Testing: Unifying IC and Board Test. *IEEE Design & Test of Computers*, 3(4):26–32, December 1986.
- [20] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 130–143, Washington, DC, USA, October 1998.
- [21] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, USA, October 1998.
- [22] Teresa McLaurin and Souvik Ghosh. ETM10 Incorporates Hardware Segment of IEEE P1500. *IEEE Design & Test of Computers*, 19(3):6–11, May/June 2002.
- [23] Francisco DaSilva, editor. *IEEE Std 1500™-2005, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits*. IEEE Standards Association, New York, NY, USA, August 2005.
- [24] Erik Jan Marinissen et al. On IEEE P1500’s Standard for Embedded Core Test. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 18(4/5):365–383, August 2002.
- [25] Nur Touba and Bahram Pouya. Testing Embedded Cores Using Partial Isolation Rings. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 10–16, Monterey, CA, USA, April 1997.
- [26] Erik Jan Marinissen, Yervant Zorian, Rohit Kapur, Tony Taylor, and Lee Whetsel. Towards a Standard for Embedded Core Test: An Example. In *Proceedings IEEE International Test Conference (ITC)*, pages 616–627, Atlantic City, NJ, USA, September 1999.

Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>