FP7-ICT-2013-11-619871

BASTION

Board and SoC Test Instrumentation for Ageing and No Failure Found

Instrument: Collaborative Project
Thematic Priority: Information and Communication Technologies

Report on developed reconfigurable instruments for ageing fault evoking and high-performance test and diagnosis (Deliverable D2.2)

Due date of deliverable: December 31, 2016
Ready for submission date: March 20, 2017

Start date of project: January 1, 2014 Duration: 40 months

Organisation name of lead contractor for this deliverable: University of Twente (UT)

Revision 1.4

Project co-funded by the European Commission within the Seventh Framework Programme (2014-2016)		
Dissemination Level		
PU	Public	☒
PP	Restricted to other programme participants (including the Commission Services)	☐
RE	Restricted to a group specified by the consortium (including the Commission Services)	☐
CO	Confidential, only for members of the consortium (including the Commission Services)	☐

Notices

For information, please contact Hans G. Kerkhoff, e-mail: h.g.kerkhoff@utwente.nl

This document is intended to fulfil the contractual obligations of the BASTION project concerning deliverable D2.2 described in contract 619871.

Table of Revisions

Version	Date	Description and reason	Author	Affected sections
0.1	Nov 1, 2016	Structure created	H. Ebrahimi	All
0.2	Nov 10, 2016	UT Contribution	A. Rohani, H. Ebrahimi	Section 2, 3
0.3	Nov 20, 2016	Overall changes	H. Ebrahimi	All
0.4	Nov 30, 2016	Draft in SVN	H. Ebrahimi	
0.5	Dec 22, 2016	LUND's contribution	F. Zadegan, E. Larsson	Section 4
0.6	Jan 17, 2017	Torino's contribution	Matteo Sonza Reorda	Section 5
0.7	Jan 16, 2017	Testonica's contribution	A. Jutman	Section 6.1
0.8	Jan 23, 2017	Infineon's contribution	P. Engelke	Section 6.2
0.9	Jan 25, 2017	TUT's contribution	J. Raik	Section 6.3
1.0	Feb 03, 2017	Combined all contributions	H. Ebrahimi, H. Kerkhoff	All
1.1	Feb 07, 2017	Added contributions	H. Kerkhoff	All
1.2	Feb 13, 2017	review	K. Rene	All
1.3	Mar 3, 2017	Last review	H. Ebrahimi	All
1.4	Mar 20, 2017	Preparing for submission	A. Jutman	Title page, Info pages, Summary, Fig. 6-3

Author, Beneficiary

Jaan Raik, Technical University of Tallinn
Piet Engelke, Infineon Technologies AG
Matteo Sonza Reorda, Politecnico di Torino
Riccardo Cantoro, Politecnico di Torino
Artur Jutman, Testonica Lab
Sergei Devadze, Testonica Lab
Sergei Odintsov, Testonica Lab
Erik Larsson, University of Lund
Farrokh G. Zadegan, University of Lund
Hans G. Kerkhoff, University of Twente
Hassan Ebrahimi, University of Twente
Alireza Rohani, University of Twente

About

This BASTION deliverable D2.2 is a report on developed reconfigurable instruments for aging fault evoking and high-performance test and diagnosis. It includes design and validation of embedded instruments for aging faults at chip level, possible reuse and inter-instrument synchronization with interrupts. It presents results coming from performed research tasks T2.2 (*realistic fault-evoking instruments*) and T2.3

(*instruments for high-performance test and diagnosis*). It shows the achievement of the goal of milestone MS6, where reconfigurable instruments for aging fault evoking and high-performance test and diagnosis are available and problems of their reuse adaptation and integration for board test have been studied. Almost all BASTION partners have contributed to this deliverable.

List of Abbreviations

ADC	- Analog to Digital Converter
ASIC	- Application-Specific Integrated Circuit
BIST	- Built-In Self-Test
CMOS	- Complementary Metal-Oxide Semiconductor
CSU	- Capture Shift Update
DNL	- Differential Nonlinearity
DUT	- Device Under Test
EMR	- Error-code/mask shift-register
FIB	- Focused Ion Beams
FIFO	- First In First Out
FP7	- European Union's 7th Framework Program
FPGA	- Field Programmable Gate Array
FSM	- Finite State Machine
FSO	- From Scan-Out
FT	- Functional Test
GIT	- Generalized Integral Transforms
HW	- Hardware
IC	- Integrated Circuit
IF	- Intermittent fault
IJTAG	- Internal JTAG, a short name for IEEE 1687 Standard and Infrastructure Collectively
INL	- Integral Nonlinearity
IRF	- Intermittent Resistive Fault
ITRS	- International Technology Road-map for Semiconductors
JTAG	- Joint Test Action Group; also Boundary Scan; often used as a short name of the IEEE 1149.1 standard and respective infrastructure including test access port and header on the board;
LBDR	- Logic-Based Distributed Routing
MIPS	- Microprocessor without Interlocked Pipeline Stages
MSB	- Most Significant Bit
NBTI	- Negative Bias Temperature Instability
NFF	- No Fault Found or No Failure Found (also NTF)

PCB	- Printed-Circuit Board
RC	- Resistive-Capacitance
RSN	- Reconfigurable Scan Network
RTD	- Research and Technological Development
RTR	- Run-Time Reconfiguration
SAR	- Successive Approximation Register
SIB	- Segment Insertion Bit
SoC	- System on Chip
SSWA	- Sustained Switching Activity
SW	- Software
SWA	- Switching Activity
TAP	- Test Access Port
TCK	- Test Clock Cycles
TDO	- Test Data Out
TDR	- Test Data Register
TRF	- Timing Related Fault
TSI	- To Scan-In
TSO	- To Scan-Out
TSV	- Through-Silicon Vias
TUE	- Total Unadjusted Error
UUT	- Unit Under Test

Table of Contents

Table of Figures:

Table of Tables:

Executive Summary

In this part, the research activities and achievements of the partners in tasks T2.2 (realistic fault-evoking instruments) and T2.3 (instruments for high-performance test and diagnosis) are discussed. It reports on the design and validation of embedded instruments for evoking of several aging and intermittent faults employing IEEE 1687, and in addition instruments for high-performance test and diagnosis. First, the state of the art is discussed, then the achievements of the partners in these tasks are presented in the beyond state of the art; it concludes with cooperation.

Summary of state-of-the-art

In this section, the state-of-the-arts at aging and intermittent fault evoking are presented shortly. The state-of-the-art in this area has been elaborated in detail in Section 1.3.

There are some publications in evoking intermittent faults at board level. However, intermittent fault evoking at chip level is a new challenge that has not been investigated or published. The activation of intermittent resistive faults (IRFs) or other faults acting on the internal circuit can in specific cases be achieved by increasing the temperature inside of chips. Thermal change can be accomplished by maximizing the switching activity. Previously, maximizing the switching activity has been used for example to validate the power characteristics of a design [6], or for performing reliability characterization [7], or to validate the mechanisms implemented to dissipate power and maintain a processor below a certain temperature threshold [8] [9].

With regard to fault localization, hierarchical IEEE 1687 networks have been used in fault management schemes to connect instruments to a fault manager [13]-[15]. In [13], methods for optimized design and calculation of error localization time are presented for the proposed fault management scheme. The work in [14] extends [13] by elaborating on how the fault manager can react faster to new faults while the instrument access network is in use for other purposes and how multiple faults can be addressed, but presents no time analysis method or experimental results for such cases. In [15], a simulation-based platform for experimenting with fault injection and fault management is elaborated, but no time analysis or network optimization method is presented.

Beyond state-of-the-art

In this deliverable, we have shown through experimental results that intermittent resistive faults can be evoked by changing the local temperature. Based on this approach two fault-evoking methods have been proposed for intermittent fault evoking at chip level.

Also in this deliverable, for the first time a software framework for intermittent fault evoking at chip level has been introduced. In this framework, the power consumption of local blocks is extracted, and using the layout information of the chip, an analogous electrical network of heat transfer is produced. The proposed approach can define the temperature profile for any granularity, as long as the power consumption of the desired modules is measurable. It is very important to note that there is no linear dependency between the power consumption and the temperature of each module.

In addition, we have proposed another way of intermittent fault evoking at chip level by maximize the sustained switching activity within a processor. We have proposed a method for identifying the sequence of instructions able to maximize the sustained switching activity of a module within a processor. Experimental results gathered on a MIPS-like processor show the effectiveness of the method.

With regards to fault localization, we have proposed a new self-reconfiguration IEEE 1687 which speeds up fault localization by introducing a new segment insertion bit (SIB). We validated the idea of self-reconfiguring networks through post-layout simulations of one such network. Comparison with a previous similar work shows at least 2.6 times reduction in localization time for a single fault and 6.2 times reduction in case of multiple faults.

In terms of instruments for high-performance test and diagnosis, new instruments have been designed for high-performance test and diagnosis at board and chip level. At board level, two developed FPGA-based embedded instruments are designed to detect timing related faults. In order to enhance high-performance test at chip level, optimizations of the BIST method has been presented. The concept of "Fault injection for validating error checkers" in deliverable D1.2 has been followed here with proposing a fault evoking instrument for the purpose of in-field testing and validation of the fault management infrastructure across different layers. The instruments have been implemented in a demonstrator ASIC with the goal to run fault-injection experiments in real-time validation.

Cooperation

With regard to intermittent and aging faults as well as evoking at chip level there was a tight cooperation between the University of Twente and Politecnico di Torino. This cooperation has resulted in two publications [35][41]. In addition, Testonica Labs and the University of Twente have cooperated on detecting intermittent resistive faults at board level, experimenting with eye-diagram software and a developed IRF detector.

1 Introduction

In this deliverable, results are reported about the research performed by BASTION partners on design and validation of techniques and embedded instruments for evoking of intermittent faults, in combination with the IEEE 1687 standard. The evoking techniques at chip and board level have been investigated.

1.1 Structure of the document

This deliverable reports on the design and validation of embedded instruments for evoking of several aging and intermittent faults employing IEEE 1687, in addition instruments for high-performance test and diagnosis. This report is structured as follows. First, background and the state-of-the-art per subject are discussed. Next, experimental results on fault evoking at board level are presented. It follows with introducing a software framework for fault evoking at chip level. Next, a reconfigurable IEEE 1687 network has been proposed to enhance fault localization. The last Section provides the proposed instruments for high-performance test and diagnosis. Finally, conclusions are provided.

1.2 Background

Interconnection reliability issues become extremely important as semiconductor technology scales. Smaller interconnect dimensions, shrinking geometries, and reduced voltage and noise margins threat the dependability of electronic integrated systems. In electronic systems like system chips and printed-circuit boards, interconnection wiring is heavily dominating the infrastructure and therefore faults in these parts are extremely important.

One of the most challenging interconnect reliability issues that threat dependability of highly dependable systems are intermittent resistive fault (IRF). IRFs are a specific category of No Fault Founds (NFFs). Like other types of NFFs, they result in many product returns in car and avionic industries; moreover, evoking and detection of these faults are highly time and cost consuming [1].

Marginal or unstable interconnections are the most likely cause of IRFs. In advanced integrated circuits, as well as printed-circuit boards, there are a high number of interconnection wires and vias. For emerging 3D chips many very deep and stress-sensitive Through-Silicon Vias (TSVs) are being used as interconnections. Electro migration, corrosion, temperature and mechanical stress cause more increased instability.

IRFs manifest themselves as a sequence (burst) of low-level resistance changes in an interconnection. IRFs might occur randomly during system operational time in any interconnect. Also, IRFs emerge repetitively in a location and they gradually become more severe during the life time of the system. Finally they can evolve in a permanent fault [2]. Therefore, it is very important to detect and repair these faults before they become permanent and result in a system failure especially in safety-critical systems.

Intermittent fault (IF) detection is very challenging. IFs are not deterministic and may not appear during testing. Therefore, for IF detection, highly time and cost consuming

techniques such as periodic testing [3] or online monitoring are required. Before one can be able to detect IRF, it is required to use a method to evoke these faults. As shown by our and other researchers' experiments; IFs can be active or evoked by temperature changes (see Section 2). One of the methods to evoke IFs is changing the temperature in locally in different modules. There are some conventional methods to manipulate temperature for boards like PCBs. One of the most common methods is temperature cycling.
Another approach for evoking IRFs (and other defects as well) lies on devising suitable input stimuli, able to artificially induce a high level of switching activity in a given module, or to create special distributions of switching activity in the different modules comprising a circuit/system. This approach requires some effective methods to generate the required stimuli.
After local thermal control, the next step in fault evoking process is fault localization. IEEE 1687 network are one effective way for fault localization. In this deliverable, we introduce an enhanced network witch significantly reduces the fault localization time.

1.3 State of the art

This Section comprises of the state-of-the-arts in different aspect of fault evoking and localization. First, the state-of-the-arts in temperature cycling for interconnection have been introduced. Then, the state-of-the-arts in on-chip local thermal control have been presented. Finally, the state-of-the-arts in fault localization and management with IEEE 1678 have been introduced.
Regarding temperature cycling acceleration, the authors of [4] presents a schedule-based technique that integrates temperature cycling acceleration with testing procedure. The cycling acceleration is achieved by mixing heating sequences and cooling intervals with test sequences in an efficient order. Furthermore, tests and heating sequences are reordered so that a rapid testing and acceleration process is achieved. The existing approaches are based on temperature chambers and can be impractical for 3D-SICs due to their unaffordable costs and limitations.
In [5], a multi-temperature testing is used to detect the temperature dependent defects. It focuses on core-based designs that allow parallel test of different cores. Testing different cores in parallel drastically reduces the overall test time. Their approach enables also multiple cores to be tested at different temperatures simultaneously by ordering and combining the normal tests and the temperature-dependent tests.
In contrast to these works witch are structural thermal control, in this deliverable two functional approaches have been investigated. Beside simple implementation functional thermal control provides more flexibility.

When addressing the activation of IRFs or other faults acting on the internal circuit activity, one has to maximize the switching activity, which is the opposite goal than the one of many works that can be found in the literature.
However, devising functional stimuli able to maximize the power consumption of a circuit has been already investigated, mainly to validate the power characteristics of a design [6], or for performing reliability characterization [7], or to validate the mechanisms implemented to dissipate power and maintain a processor below the temperature threshold [8] [9]. In [10] the authors showed that proper temperature gradients between different parts in a circuit (or cores in a SoC) may allow the detection of defects that could hardly be targeted in other ways. In that paper the authors also proposed to use suitable stimuli to create these gradients, i.e., to control the temperature

of each module via its activity, thus devising a mechanism similar to the usual Burn-In test; however, temperature control is obtained through suitable stimuli, instead of via external heating. Similarly, the author of [11] proposes to use input vectors able to maximize the heat dissipation in a combinational block in order to reduce the duration of Burn-In campaigns. The work in [12] focuses on IRFs (Intermittent Resistive Faults) caused by temperature.
When the target is to generate functional stimuli able to maximize the switching activity of the whole device (or of single modules within it) is important and to maintain it at the achieved high level for a desired duration, different solutions can be adopted.
A first solution has been proposed in [9], where the usage of a SAT solver was proposed: this solution is particularly aimed at maximizing the switching activity within a single module, which can be more easily managed by formal techniques. The authors of [9] focused on a similar problem, targeting generic sequential circuits. In our work within BASTION we targeted processors, and developed a method to extract from existing programs those parts that are most suitable to maximize the switching activity in a given module, and to maintain it at a high level for a given duration.

Hierarchical IEEE 1687 networks have been used in fault management schemes to connect instruments to a fault manager [13]-[15]. In [13], methods for optimized design and calculation of error localization time are presented for the proposed fault management scheme. The work in [14] extends [13] by elaborating on how the fault manager can react faster to new faults while the instrument access network is in use for other purposes and how multiple faults can be addressed, but presents no time analysis method or experimental results for such cases. In [15], a simulation-based platform for experimenting with fault injection and fault management is elaborated, but no time analysis or network optimization method is presented.
Along with the IEEE 1687 network, [13]-[15] use a fully combinational error propagation network which propagates error flags to the highest hierarchical level of the IEEE 1687 network. A simplified representation of the hierarchical networks used in [13] is shown in Figure 5-2(b) where the error propagation network is marked by the dashed lines. The advantage is that by reading the ErrorFlag register in the highest level the fault manager gets informed of any error in the system without checking each and every instrument. To guide fault localization, [13]-[15] added ErrorFlags at every level, resulting in dramatic increase in fault localization time. Also, fault localization in [13]-[15] involves a number of CSUs to open hierarchical levels, each CSU performed over a scan-path longer than the scan-path for the previous CSU, increasing the fault localization time.

2 Evoking Experiments at Board-Level

In this Section our measurements based on thermal cycling are provided. The experiment has been carried out on one of the main causes of intermittent faults in boards i.e. cold solders. Behavior of a several cold solders under thermal cycling has been investigated.

Our measurement setup is shown in Figure 2-1. A constant resistance R1 is in series with a cold solder and the resistance network is connected to a DC supply voltage. By measuring V1 by a multimeter and using Ohms' law we can easily calculate the value of the resistance for the cold solder joint.

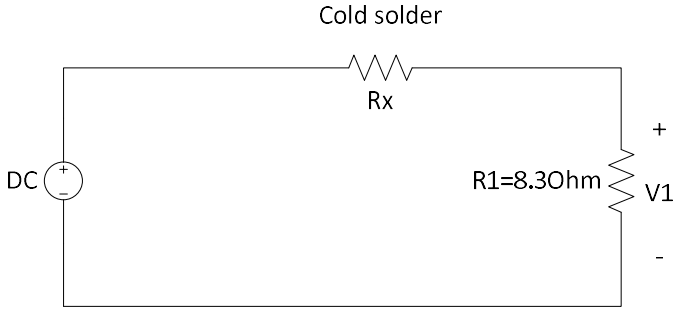


Figure 2-1. Cold soldering measurement setup

We measured the resistance of the cold solder under a harsh environment, which is induced by thermal cycling. The profile of this thermal cycling is shown in Figure 2-2. The solder joint was heated up from room temperature to 120 °C. The highest temperature (120 °C) is selected to be lower than the melting point of the solder (160 °C) otherwise it was possible that the cold solder became a proper solder joint and we would be unable to observe the intermittent behavior of the cold solder.

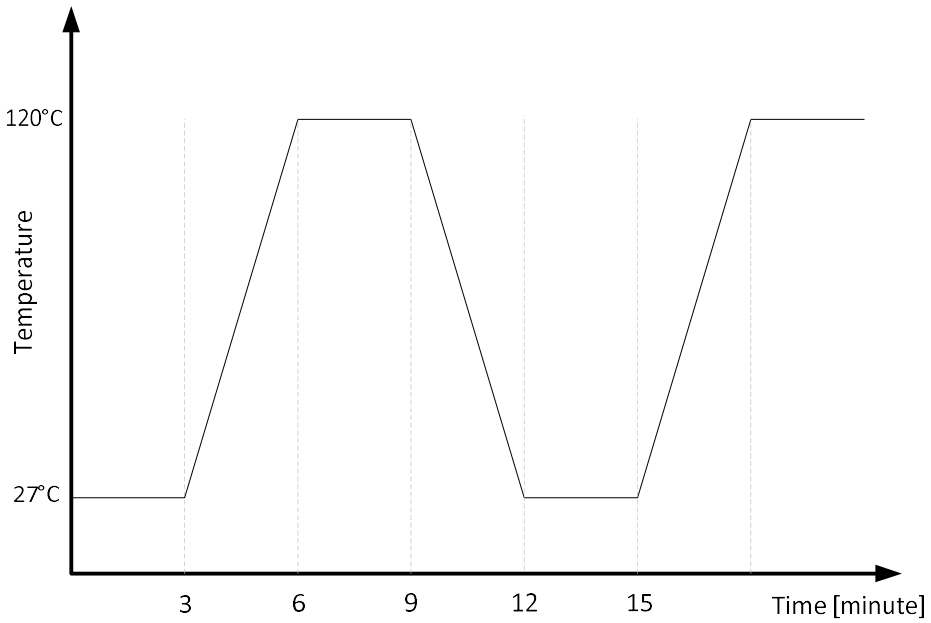


Figure 2-2. Thermal cycling profile

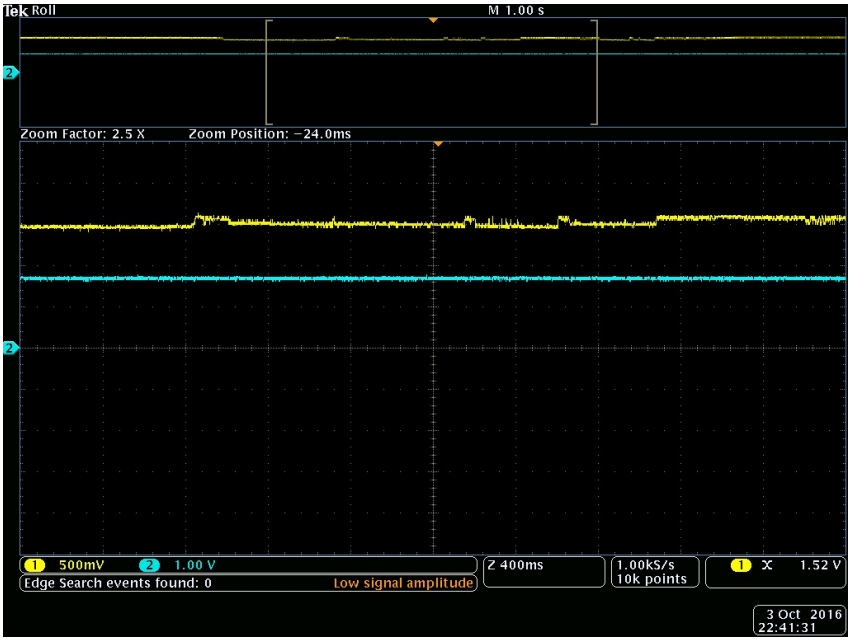


Figure 2-3. Voltage measurement of a cold solder joint (early-stage)

Figure 2-3 shows the voltage measurement of a cold solder joint at the start of temperature cycling. In this Figure, the yellow signal is the measured voltage of the constant resistance and the blue signal is the voltage supply of the test setup, which is 1 volt DC voltage. If the circuit operates near room temperature there is not any resistance change in the cold solder joint, but after several temperature cycles the digitizer detects a voltage change. The voltage changes to reveal that there are resistance changes in the cold solder. It means the *induced temperature changes can evoke cold solder to behave as an intermittent resistive fault*.

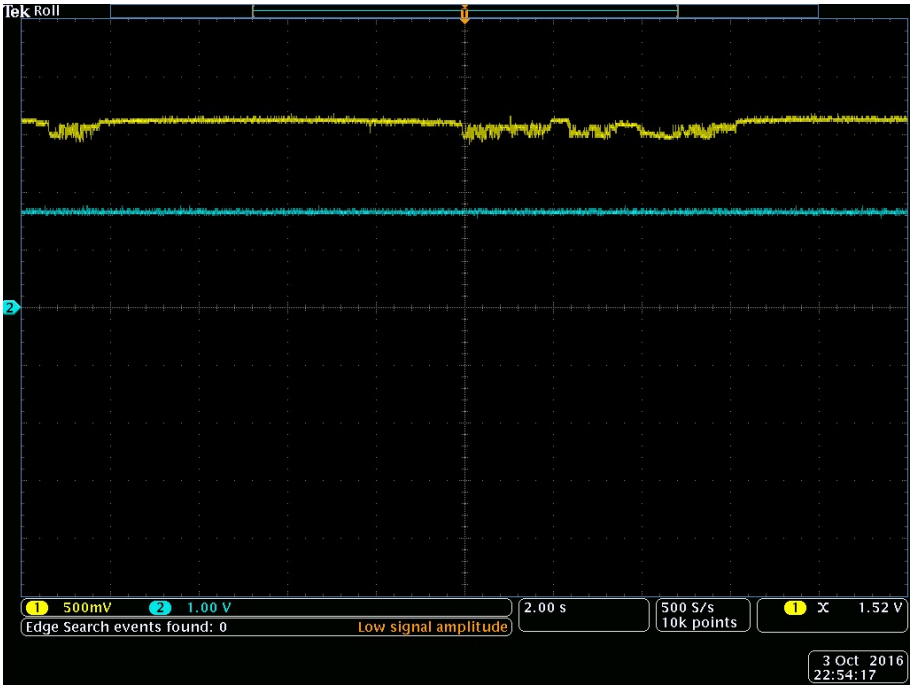


Figure 2-4. Voltage measurement of a cold solder (middle-stage)

Figure 2-4 shows the behavior of the same cold solder after of several minutes of temperature cycling. The comparison of Figure 2-3 and Figure 2-4 indicates the intermittent resistive fault caused by the cold solder became more severe in the presence of temperature cycling. Other measurements show that if temperature cycling has stopped, the intermittent resistive fault witch has been evoked before can become either more or less severe but it will not disappear completely.
In case that the temperature increases higher than the melting point of the solder joint then there is a possibility that the cold solder turns into a good solder joint and subsequently the intermittent resistive fault disappears.
Figure 2-45 shows an Iddt current measurement of a cold solder joint. The X-axis is time axis and the Y-axis is the current using a 25mA scale. It can be interpreted as if the circuit is small like in this case (only two resistances) the current measurement can be useful to detect intermittent resistive faults in the circuit.

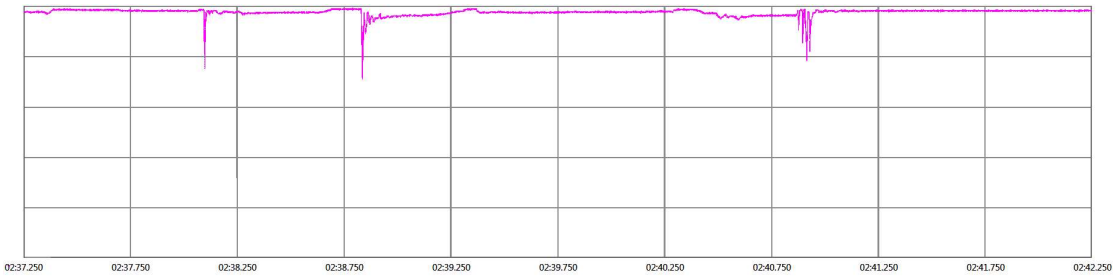


Figure 2-5. Current measurement during an IRF (cold soldering joint) evoking process

Our experimental results show that intermittent resistive faults can be evoked by changing the local temperature. At board-level, any thermal changing can be obtained by using thermal chambers or focused forced hot air. However, evoking IRFs by local thermal changing is not trivial at chip level. In the following section, we propose a framework that enables us to control the local temperature at chip level.

3 Design and Validation of a Framework for Evoking IRFs at Chip Level

In this Section, a framework to evoke intermittent faults at chip level is introduced. This framework helps to evoke existing intermittent faults inside of a chip by controlling local temperatures in the chip [20]. In the following subsections, first a software framework will be introduced to calculate local temperatures in processors, then an approach to generate a thermal model based on a given processor floorplan has been proposed. Finally, a case study has been chosen to show the effectiveness of the framework.

3.1 A Software Framework to Calculate Local Temperatures in Processors

In the following, first there is an introduction about a why a software framework in needed for local temperature calculation inside a processor and then

3.1.1 Introduction

In the development of electronic systems, ranging from Printed-Circuit Boards (PCBs) to VLSI Integrated Circuits (ICs) and System-on-Chips (SoCs) complicated faults have been emerged in electronic circuits. One of these faults is known as Intermittent Resistive Faults (IRFs) which are mostly originating from imperfect interconnections [16] under extreme environmental conditions such as high temperature or extensive vibration. So, during a high temperature, a loose interconnections, either the soldering in PCBs or chip vias or a TSV (Through Silicon Via) in 2.5D, 3D chips can impact the correct operation of a system. The impact is normally appears as bursts of signals, with a random duration, in the output of the affected circuit [16]. However, the reproduction of these faults in a simulation-environment is very challenging because of their random behavior with regard to the time of occurrence. This fact makes the diagnosis of IRFs very challenging since activation of IRFs needs a simulated-controlled environment, which is hard to set-up, especially with regard to temperature. That makes the diagnosis of IRFs a hot topic in industry especially if one notes that IRF rank among the highest in terms of occurrence; in addition the cost of dealing with IRFs is expected to increase in future technology nodes [12][16][17].

One of the methods to evoke IRFs is increasing the temperature in certain modules; however, conventional methods to rise the temperature suffer from non-controllability. In other words, it is almost impossible to control the exact temperature of each module separately while the chip is heating up. This contribution of our method is to present a mechanism to conduct the temperature rising of a chip in a controlled way. The first step is to derive a thermal model that can extract detailed granularity of heat distributions in a chip. Such a thermal model enables the designer to extract the temperature of each local block of a design. An important factor is that the granularity of temperature profile should be sufficiently flexible to be determined by the designer. The core contribution of this section consists of extracting the local temperature of each module based on the executed (software) workload. To be able to do this, first a correlation between local temperature and switching activity (dynamic power) needs to be developed and then based on the imposed activity that a workload imposes on a module, its local temperature can be calculated.

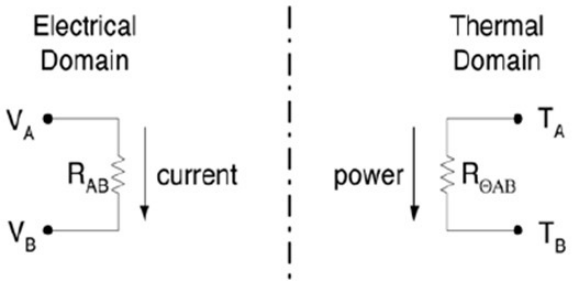


Figure 3-1. Fundamental relationships in the electrical and thermal domains

Our core approach to develop a thermal model based on switching activity of each module is based on the duality between heat transfer and electrical phenomena [19][18], as shown in Figure 3-1. Switching activity incurs dynamic power and dynamic power will produce heat in a circuit. The correlation between switching activity and dynamic power is straightforward but the correlation between dynamic power and heat transfer is more challenging. To be able to do the latter, heat is considered as a **current** which passes through a thermal resistance and creates temperature differences analogous to a **voltage**. The power delivered to the block is considered as a current source. Moreover, thermal capacitances are considered to model the transient distribution of heat across the design.
This model, which is known as the compact model [18][19], considers the heat conduction between neighboring modules. The modules can be of any granularity, which is selected by a designer. The only criterion for the selection of modules is that their power consumption should be measurable. The mechanism to extract these parameters are out of the scope of this report and can be found in the complete paper published in [20].

3.1.2 Related work

The International Technology Road-map for Semiconductors (ITRS) has projected little change in the supply voltage of ICs that will occur in the future. As a result, power densities in ICs are projected to rise faster for the future technologies. As a result, the importance of temperature-aware designs will be more and more significant. The increased temperature has different consequences; i.e. it can significantly decrease the reliability of integrated circuits with regards to soft-errors, IRFs and aging faults, [21][22]. Moreover, it can deteriorate the performance of digital systems by reducing the carrier mobility of CMOS transistors.

In general, thermal distribution models can be categorized into two classes: numerical and analytical models. The numerical models use the duality between heat transfer and electrical circuits to articulate the heat transfer to an *RC* (Resistive-Capacitance) network. The analytical-based methods derive the local temperature of each part of the chip via a statistical approximation of the temperature of the entire chip.

The analytical-based methods are scarce in the thermal community. As one of the few works, the authors in [23] estimate the local temperature via forming a power grid on the desired granularity and subsequently calculate the consumed power in each grid point from the total power. The authors in [24] presented another analytical approach by using the generalized integral transforms (GIT) to estimate the local temperatures. Although their method is accurate, the granularity of temperatures that can be extracted are fixed and cannot be altered by the designer.

Numerical-based methods are the main focus of the thermal design community in recent years. In [25][26][27] the authors present different thermal models. These models all have detailed temperature distribution information across the chip. However, a limitation of the above models is that the thermal package is over-simplified. For example, the PCB that greatly affects chip temperature distribution is not included in the models.

The authors in [28][29][30] considered a more complex model which is able to model heat transfer via a cooling fan, however, the core thermal model is composed of only thermal resistance which neglect the temporal delay of heat transfer between two materials.

Two well-known numerical models to extract local temperature are called TEMPTEST [31] and HotSpot [32]. TEMPSET uses duality between electrical circuits and heat to model temperature; however, it contains one equivalent RC for the entire chip. As a result, extracting local temperature for each part of the chip is not very accurate. Hotspot which is presented in [32] uses analogy between electrical phenomena and heat transfer in order to model temperature. The main focus of the authors in [32] is to propose a run-time power management framework to avoid local hotspots while our contribution is to develop a controllable mechanism to impose temperature in a CMOS processor.

3.1.3 Thermal Model

Our proposed model is based on the analogy between heat transfer and electrical phenomena. The distribution of heat to the neighboring modules is modeled by the thermal resistance. Moreover, a thermal capacitance is being used to capture the time in which a module resists to change to a new temperature.

The values of the thermal resistance and capacitance depends on different parameters such as the distance between modules, specific materials and initial temperature. The power that is delivered to the module is modeled as a **current source**. The temperature of each module is analogous to the voltage of each node which is extracted by solving the analogous electrical equation. Considering that the layout of a chip is fixed, changing the workload affects only the value of the **current source**. As a result, the temperature of each block for different workloads can be calculated rapidly.

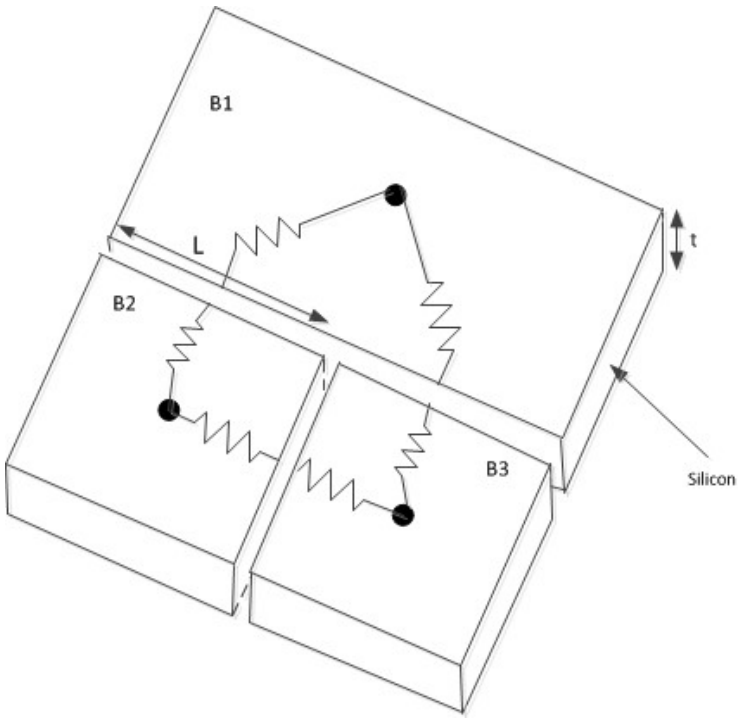


Figure 3-2. Thermal resistances in a chip composed of three blocks

A simple 3D model of an IC with three modules and their thermal resistances are illustrated in Figure 3-2. The IC is composed of three modules, *B1*, *B2* and *B3*.
The power is considered to be delivered at the center of each module (indicated by a dot). This is the dynamic power, which is averaged during the whole execution of a workload and can be extracted from available power analyses tools, such as the Synopsys power analyzer. To calculate the thermal resistance, the following parameters should be considered:

A_s: contact area of the heat source
A_p: contact area of the neighboring block
t: thickness of the chip (shown in the Figure 3-2)
k: thermal conductivity of the heat sink

For each block, the first three parameters can be extracted from place-and-route tools. For example, the area between *B1* and *B2* in Figure 3-2 can be calculated by the multiplication of L and t. The value for k depends on the material that needs to be extracted from the data-sheet of the material.
The exact equation between the cross-section area of two blocks, k and the thermal resistance is extracted from [33], which is as follows:

$$\frac{1}{R_{thermal}} = \frac{\sqrt{A_p * A_s}}{k * \sqrt{\pi * A_p * A_s}} * \frac{\beta * k * A_p + \tan(\beta * t)}{1 + \beta * k * A_p * \tan(\beta * t)} \quad \textbf{(1)}$$

Where:

$$\beta = \frac{\pi^{\frac{3}{2}}}{\sqrt{A_p}} + \frac{1}{\sqrt{A_s}} \quad \textbf{(2)}$$

This value for all six thermal resistances as shown in Figure 3-2 need to be calculated. A MATLAB routine can quickly calculate these values.

As stated before, thermal capacitances are also involved when different materials interact with each other in heat conduction. If this happens the heat conduction between a chip and its package and/or from a package to the air should be modeled. Driving the thermal capacitance is more straightforward as compared to the thermal resistance as it is proportional to the thickness and contact area of interacting materials. From [23], the following formula can be extracted:

$$C_{thermal} = c * T * A \quad \textbf{(3)}$$

where c denotes the thermal capacitance per unit volume. The idea of capacitance in this work is to mimic the transient change when heat transfers from silicon IC to the package and from the package to the ambient.
Finally, all these values can form an electrical circuit that needs to be solved by an electrical circuit simulator, such as PSPICE. As stated before, the *voltage* of each node represents the *temperature* in the center of that module. The next section shows the application of this model to a thirty-two bit adder circuit.

3.1.4 Temperature Evoking in a Thirty-two bit Adder

In order to show the details of the temperature evoking, a thirty-two bit adder has been selected. The general set-up of the adder is composed of thirty-two full-adders (FA). There are two inputs, *A* and *B*, each thirty-two bits wide. The entire system is modeled by using the VHDL language. The selection of workloads on *A* and *B* are in such a way that they control activities on specific parts of the circuit. For example, if *A* equals to *00000005* and *B* equals to *0000000F*, the least significant bits of the adders will change their values and consequently will have more activity as compared to the rest of the circuit. Frequent changes of *A* and *B* impose different activities in Least-Significant Bits (*S0* to *S10*), Medium-Significant Bits (*S11-S20*) and Most-Significant Bits (*S21-S32*), consequently. The power imposed on the circuit is calculated and will be used to build the analogous electrical network in the circuit.

Table 3-1. Dynamic power results

	Inputs		**Power(mW)**		
workloads	**A**	**B**	**FA-2**	**FA-20**	**FA-30**
W1	0000 0005	0000 000F	8.13	1.02	1.02
W2	0000 FF00	0000 5500	1.07	6.50	1.26
W3	0F00 0001	5F00 0000	1.05	1.04	6.89

A power compiler has been used to calculate the power consumption (dynamic power) of each module (full-adder) per workload. The power consumption will be used in

generating the analogous electrical workload of the circuit and consequently extract the temperature profile of the system.
Table 3-1 shows three different workloads, as well as the dynamic power consumption of three full-adders that are located fairly far from each other. This information was directly obtained by the *power analyzer of Synopsys* tools. As can be seen in this table, each workload incurs a different power consumption on specific blocks. This power information need to be calculated for all full-adders. The next step is to build the analogous electrical network for heat transfer, and using the information of power analyzer to complete the analogous electrical network.

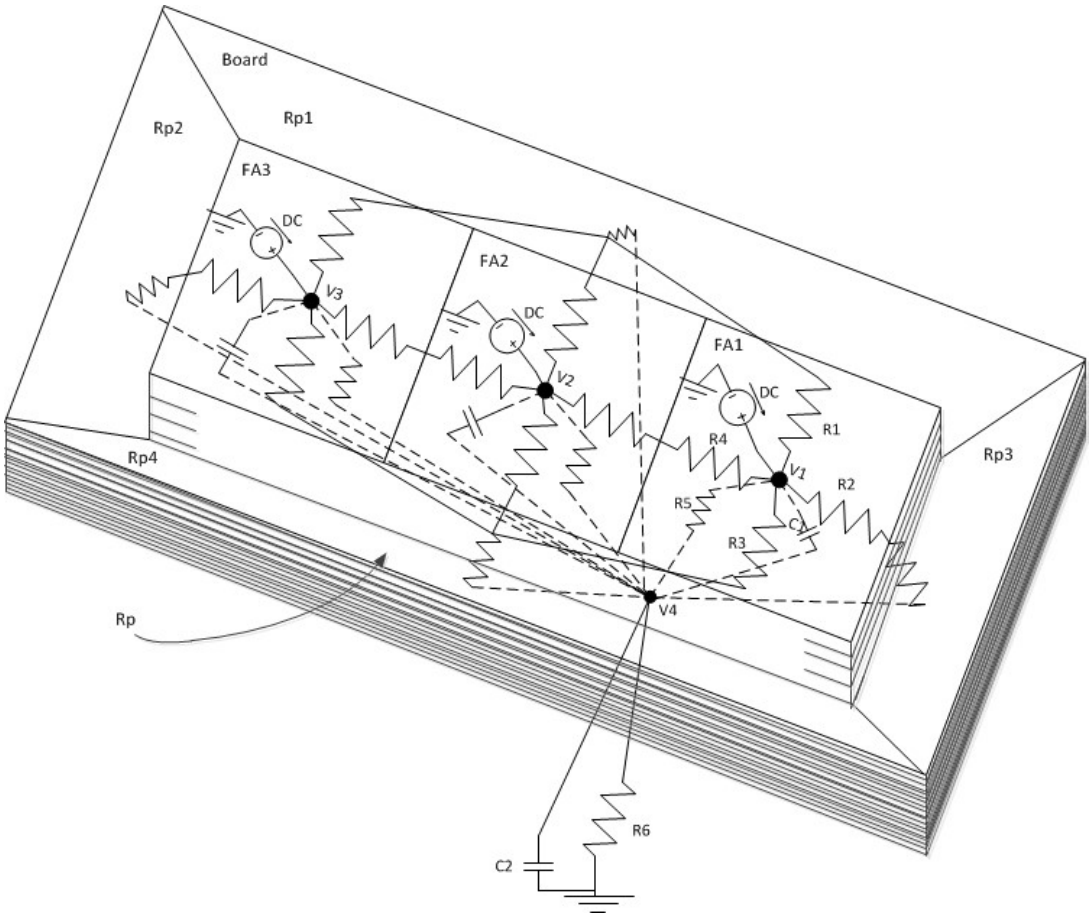


Figure 3-3. The analogous electrical network of the adder

In order to build the analogous electrical network of the circuit, the layout information is also required. That information is obtained via the Cadence place-and-route tool. However, the package which is placed around the adder is also considered.
Figure 3-3 shows the analogues electrical model for first three full-adders. As can be seen in this picture, the package which accommodates the adder is divided into five parts: one corresponds to the area right under the adder *Rp* and four trapezoids for the periphery *RP1*, *RP2*, *RP3*, *RP4*). Each peripheral area conducts the temperature away from one of the blocks while *Rp* area conducts the temperature away from all the blocks of the adder. Finally, the convective heat transfers from the package to the ambient is represented by a single thermal resistance *R6*). If the package covers the top of the chip, another ambient thermal resistance on top of the chip is also required.
As can be seen in this picture, the switching power of each module has been represented as a current source in which their values are derived directly from the power synthesis

results. The heat conduction from each module to the neighboring modules are described by five thermal resistance, named *R1*, *R2*, *R3*, *R4* and *R5*, for the block *FA1*. Notice that *R5* is the heat conduction between the *FA1* and the area of package which is situated right beneath the IC (the dotted lines are situated beneath the IC).

The *R6* resistance is the heat conduction between the package and the ambient temperature. The value of this specific thermal resistance is derived from literature [32]. Moreover, several capacitances are placed to emulate the gradual heat conduction between the IC and the package as well as between the package and the ambient. The value of this thermal capacitance is proportional to the thickness and area of the blocks. These values are also derived from the layout information.
Figure 3-4 shows the temperature raise of *FA1* for three workloads. As it can be seen in the figure, workload 1 (*W1*) imposes the highest temperature raise on this adder. Moreover, the exact temperature of this block after execution of workload W1 can be seen in this figure (the temperature raise needs to be added to the ambient temperature). Figure 3-4 shows that after approximately three seconds, the temperature of block 1 will be around 32.5◦ C, considering the ambient temperature of 21◦ C if workload W1 is executed. While workload W3 will cause a temperature of 31◦ C on this block.
Having the exact temperature raise on each module, the designer can quickly apply a workload to the circuit to raise the temperature of a **specific** module to a certain temperature. This will be very useful to **locate** any temperature-induce faults that arise in certain temperatures.

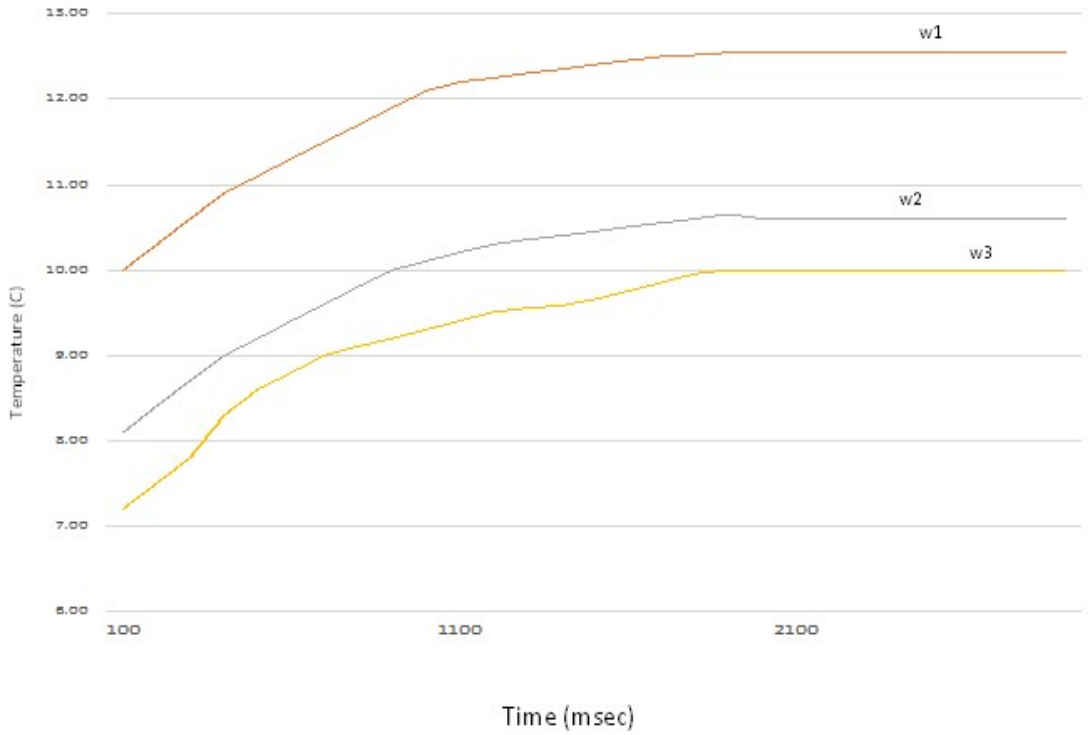


Figure 3-4. Temperature raise in full-adder1 for several different workloads

3.1.5 Case Study

This section introduces the results that have been obtained from applying the introduced framework to a complex processor. The first subsection briefly explains the processor and the second subsection shows the temperature profile of the processor.

3.1.5.1 Xentium DSP Processor

To implement our approach, a high performance Very Large Instruction Word (VLIW) processor, the Xentium processor has been selected. The Xentium processor which is designed by Recore Systems is a high performance processor that is being designed for high performance computing. The data-path architecture of this processor is shown in Figure 3-5 the data-path is based on a VLIW architecture that comprises of ten execution units and five register files. Each execution unit is responsible for a certain class of instructions. *E units* (*E0* and *E1)* perform load/store instructions; *A* and *S units* (*A0*, *A1*, *S0* and *S1)* perform arithmetic and logical operations. *M* units (*M0* and *M1)* are multipliers. *C* and *P* units (*C0* and *P0)* perform instructions which program counter is involved. All functional units can access five register files (*RFA*, *RFB*, *RFC*, *RFD* and *RFE)* in parallel.

This processor was a perfect candidate for us because the dedication of each execution units to certain instruction allows us to adjust a workload quickly in order to impose more/less temperature rise on each execution unit. For example, rising temperature on *M* units imply a workload that has loads of multiplication instructions. *A* set of three different workloads has been applied to this processor. The total activity of each execution unit with regard to each workload is shown in Table 3-2. We will show the temperature profile obtained from applying each workload on this processor.

Table 3-2. Dynamic power consumption Xentium processor

	Switching Activity (%)					
workloads	**E0/E1**	**M0/M1**	**A0/A1**	**S0/S1**	**C**	**P**
W1	80	10	10	10	50	40
W2	10	10	90	90	40	40
W3	20	20	20	20	90	80
W4	10	90	10	10	40	40

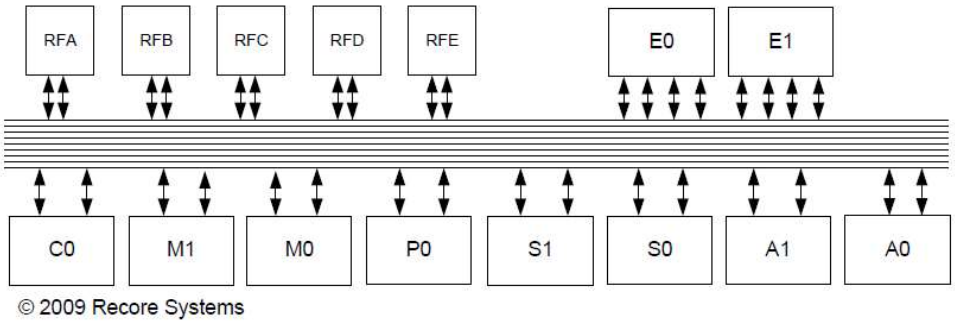


Figure 3-5. The Xentium Data-path Architecture

3.1.5.2 Temperature Profile of the Xentium Processor

The four different workloads have been applied to the Xentium processor and based on the procedure depicted in previous section, the temperature rise in each block has been extracted. Table 3-3 shows the extracted numbers. The hottest block in each workload is also shown in bold. A more detailed expansion of heat in different blocks of the processor are shown in Figure 3-6. The dark red means the hottest temperature.

Table 3-3. Temperature raise for each workload

Unit	W1	W2	W3	W4
C	7	**9**	5	**9**
M	4	9	**11**	4
P	6	11	**15**	9
S	8	**14**	9	5
A	8	**14**	7	5
E	**13**	4	4	4

The first observation from Figure 3-6 is that the temperature rise of each module does not follow a linear dependency with delivered power, but other factors such as the *temperature of adjacent blocks* or the *position of the block on the package* play important roles. For instance, workload *W1* imposes more power on *C/P units* compared to *A/S unit* but the temperature rise on *A/S units* are higher. The reason for this observation is that *A/S units* are placed closer to the hottest units (*E* units). In case of other workloads, there is also no linear correlations between dynamic power and the temperature of that block.

Another observation from Figure 3-6 is that ambient conduction has an important impact on the temperature rise of each block. For example, in the case of *W2* and *W3*, the *E1* unit has the lowest temperature, while the switching activity of this block is in the same range of other blocks. Our detailed investigation showed that *E1* unit loses a lot of heat to the ambient because it has two adjacent sides with the ambient, while it is not the case for *E0* unit.

The extracted temperature profile allows test engineer to have a more clear view about the behavior of a workload on heat generation of a processor. This will help to identify and locate temperature-induced faults more precisely in a processor.

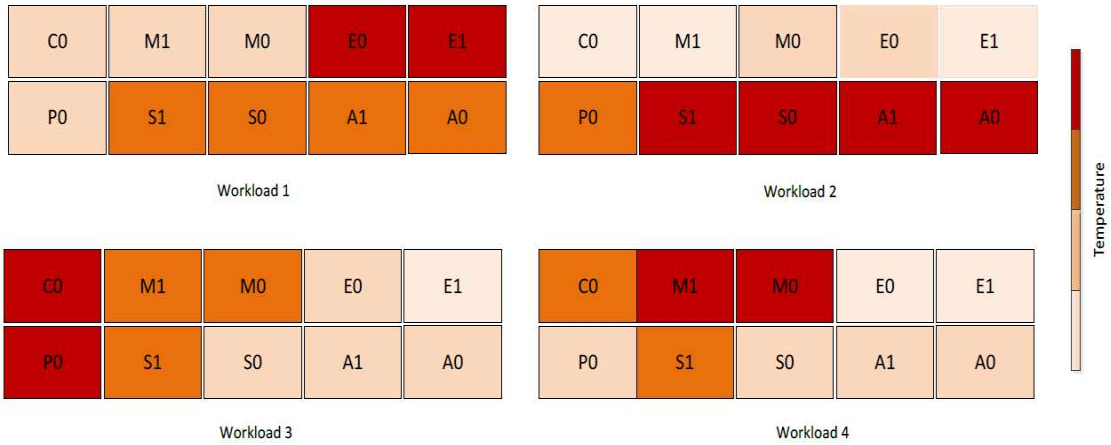


Figure 3-6. Distribution of heat for different workloads (The darker a block is, the hotter it becomes for that workload)

3.1.6 Conclusion

In this Section, a framework to extract the temperature profile of a system has been presented. First, the power consumption of local blocks are extracted, and using the layout information of the chip, an analogous electrical network of heat transfer is produced. Solving the analogous heat transfer can reveal the temperature profile of the system. The proposed approach can define the temperature profile for any granularity, as long as the power consumption of the desired modules is measurable. It is very important to note that there is *no linear dependency between the power consumption and the temperature of each module*.

4 Maximizing the sustained switching activity for accelerating aging fault occurrence

4.1 Introduction

Many techniques have been proposed in the literature (and then sometimes adopted in practice), aimed at minimizing the switching activity in a circuit, mainly to reduce its peak power or energy consumption. Some papers also targeted the reduction of the power consumption during test (e.g., [36]).

In other cases, researchers proposed solutions to identify functional stimuli able to maximize the power consumption of a circuit, mainly to validate the power characteristics of a design [6], or for performing reliability characterization [7]. A method was proposed in [8] to automatically generate special codes, able to maximize the power consumption of a processor, thus allowing to validate the mechanism implemented to dissipate power and maintain the device below the temperature threshold. A similar goal was targeted in [9].

Recently, a few scenarios emerged, where it is important to maximize the switching activity of a given module within a circuit.

For example, the authors of [10] demonstrated that creating proper temperature gradients between different parts in a circuit (or cores in a SoC) may allow the detection of defects that could hardly be targeted in other ways. In that paper the authors also proposed to use suitable stimuli to create these gradients, i.e., to control the temperature of each module via its activity, thus devising a mechanism similar to the usual Burn-In test; however, temperature control is obtained through suitable stimuli, instead of via external heating. This kind of stimuli is clearly based on maximizing the switching activity within the target module. Somehow similarly, the author of [11] proposes to use input vectors able to maximize the heat dissipation in a combinational block in order to reduce the duration of Burn-In campaigns. The authors in [12] proposed a solution to detect IRF (*Intermittent Resistive Faults*) caused by temperature.

Hence, in these scenarios it is important the availability of effective techniques to generate functional stimuli able to maximize the switching activity of the whole device (or of single modules within it) is important and to maintain it at the achieved high level for a desired duration.

A first solution to this problem has been proposed in [9], where the usage of a SAT solver was proposed: this solution is particularly aimed at maximizing the switching activity within a single module, which can be more easily managed by formal techniques.

While the authors of [9] focused on generic sequential circuits, in this section we target processors, and propose a method to extract from existing programs those parts that are most suitable to maximize the switching activity in a given module, and to maintain it at a high level for a given duration.

Experimental results are provided, showing the effectiveness of the method when applied to some modules within a pipelined processor.

4.2 Applications

The current trend of developing high performance integrated circuits has led to electrical components that dissipate massive heat during operation [35]. It is expected that a higher operating temperature is inevitable without an effective heat dissipation

mechanism [37]. However, high operating temperatures will jeopardize the reliability of an integrated circuit, since the reliability will decrease exponentially with the temperature [38]. On the other hand, heating will accelerate the aging of electronic circuits so it is hard to assess the reliability of a component with regard to high temperatures early during the design phase. In this section, it will be shown that one of the potential applications of maximization of switching activity is to provoke errors which will emerge during a raise in temperature. Furthermore, it will be shown that every component of a circuit (a digital processor in this case) can be heated up to a certain temperature while other parts of the design are not. This could find practical application for example to implement a new type of Burn-In experiments, as proposed in [10].

If the switching activity of a module is considered as F_m, the switching power in that particular module, denoted as $Power_d$ can be calculated as:

$$Power_d = 0.5 * F_m * C_m * (V_{DD})^2 \quad \textbf{(1)}$$

where C_m and V_{DD} are the module load capacitance and power-supply voltage, respectively. Maximization of F_m will result to maximum module dynamic power consumption $Power_d$. On the other hand, creating power dissipation in a module generates heat. There are different models to correlate switching power and dissipated heat in a silicon device [39]. In one case thermal conductivity of a material (often denoted by K) will be used in order to correlate power to heat dissipation. Thermal conductivity is the property of a material to conduct heat. Thermal conductivity is measured in watts per millimeter-temperature (W/mm°C). There are a number of methods to measure thermal conductivity in different materials [39][40]. The authors in [42] extracted the thermal conductivity of silicon by thermal analysis of different thicknesses of silicon. A detailed analysis of thermal conductivity is out of the scope of this deliverable, as our intention is to show the usage of thermal conductivity to correlate maximum switching activity and heat dissipation of a component. In this deliverable, it is assumed that a uniform and constant thermal conductivity (K) in all parts of an integrated circuit in all directions exist.

Considering the switching power of a component and its distance to the power source, one can drive the following equation:

$$T_r = T_A + K^{-1} D^{-1} Power_d \quad \textbf{(2)}$$

where T_r is the average temperature of a module, T_A is the ambient temperature, K is the thermal conductivity of the material, D is the distance of the module from the power source (in mm) and $Power_d$ is the switching power of the module. As it stated before, the thermal conductivity is a property of a material that is measured in W/mm°C.

Formula (2) shows that maximization of $Power_d$ will potentially lead to maximum temperature of a module. It can be shown that knowing the distance (D) of a module from a power source, it is possible to deliver a certain amount of power to a module to set it in a predefined temperature during a certain amount of time. This time is proportional to the time in which the workload for maximization of switching activity has been conducted. The parameter D of each module can be easily extracted from synthesis reports. A simple calculation for D can be represented as:

$$D = \frac{\sqrt{A}}{2} \quad \textbf{(3)}$$

where A is the area of the module. This calculation assumes that the shape of the module is squared and the power is uniformly dissipated in the module. As a result, the temperature of a module can be described as:

$$T_r = T_A + K^{-1} * (\frac{\sqrt{A}}{2})^{-1} * 0.5 * F_m * C_m * (V_{DD})^2 \quad \textbf{(4)}$$

where all parameters of equation (4) are known after the synthesis; hence, a predefined temperature can be achieved by employing a specific switching-activity profile.
As a generic example, suppose one has the information reported in Table 4-1 for a processor device and K is determined as K= 1.3 w/cm°C for this specific technology. As a result of equation (4), the temperature of the module will be 45.5°C (25°C + 20.5°C) after 30msec.

Table 4-1. Some characteristics of a module in a processor

Feature	Value
Switching-power	400 mW
Area of the module	90,000 μm^2
Ambient temperature	25°C
Execution-time of a workload	30 msec

The explained procedure can eliminate the use of external heat sources since each module in the processor can be specifically heated up to a specified temperature (limited to 150°C). Furthermore, this internal heating has more controllability over an external heat source since the temperature is diverted only on the module of interest (the module that has the maximum switching activity). In the next section, it will be shown how maximum switching activity can be derived for a module.

4.3 Proposed approach

This section aims first at formally describing the problem targeted in this deliverable. Then it outlines the proposed solution.

4.3.1 Problem formulation

Let us consider a given module within a processor. We assume that a gate-level model of the module is available, resulting from the synthesis with a given technology library. For sake of simplicity, we assume that the module is driven by a single clock signal.
Considering the i-th clock period, we can determine (e.g., by simulation) the number of logical switches $s_{i,j}$ performed by each net j before reaching the final stable value. We can then compute the *total switching activity* SWA_i for the module during the i-th clock period over the N nets composing the module as

$$SWA_i = \sum_{j=0}^{N-1} s_{i,j} \quad \textbf{(5)}$$

The normalized version $nSWA_i$ of SWA_i is defined as

$$nSWA_i = SWA_i / N \quad \textbf{(6)}$$

When considering a sequence of P clock cycles, we can identify the peak switching activity over P as PSA_P as

$$PSA_P = max\ nSWA_i \quad \textbf{(7)}$$

We can now define the *sustained switching activity SSWA* of the module over M consecutive clock cycles as

$$SSWA_M = \frac{\sum_{i=0}^{M-1} SWA_i}{M} \quad \textbf{(8)}$$

as well as its normalized version $nSSWA_M$

$$nSSWA_M = \frac{\sum_{i=0}^{M-1} nSWA_i}{M} \quad \textbf{(9)}$$

Once the value of M has been fixed, $SSWA_M$ corresponds to the switching activity F_M introduced in Sub-Section 4.2. Our goal is to find a program whose execution by the considered processor maximizes the value of *nSSWA* over a given number $M>2$ of clock cycles. The specific value of M may change depending on the specific application of the method.

If the target module we are considering is a combinational block, we can observe that the switching activity SWA_i that is achieved during the i-th clock cycle depends on the inputs applied to the block during the current clock cycle, as well as on the state of the module (i.e., the values of its nets) at the end of the previous clock cycle, which clearly depends on the values applied to the inputs during the previous clock cycle. Hence, achieving a high peak switching activity requires identifying a couple of input vectors able to trigger a maximum number of switches.

On the other side, maximizing the sustained switching activity $nSSWA_M$ requires the identification of a sequence of M input vectors able to maximize the cumulative switching activity over M clock cycles driving the module from an initial state s_0 to a final state s_{M-1}. Each vector v_i in the sequence moves the module from a state s_{i-1} to a state s_i, triggering a given switching activity. M is typically set to the smallest value such that $nSSWA_M$ is maximum and $s_0 = s_{M-1}$. In this way, the same sequence of input values can be applied again, and the module can experience the same sustained switching activity for an endless period.

The reader should note that the *SSWA* maximization is obtained by resorting to functional stimuli, only. Hence, their application cannot damage the device (unless this is affected by some design bug; this case is not considered here).

4.3.2 Proposed method

The method we propose is based on two phases:

- The first phase uses existing programs (corresponding for example to application software, or to programs developed for test and validation purposes) and aims at identifying instruction sequences able to maximize the peak switching activity; the idea is that programs able to maximize the sustained switching activity can be more easily generated once we have been able to identify programs able to maximize the peak switching activity. The output of this phase is a macro, i.e., a short sequence of instructions whose sustained switching activity can be maximized by suitably adjusting the values of the parameters it uses, as well as by possibly (slightly) modifying its structure.

- The second phase aims at optimizing the macro provided by the first phase, transforming it into a short program able to maximize the sustained switching activity.

In the following, the two phases are described in more details.

4.3.2.1 Phase 1

It takes as input some existing programs, which are supposed to stimulate and stress the target processor module. Such programs may for example correspond to application software, or to programs developed for test and validation purposes.

We perform the logic simulation of these programs and gather data about the total switching activity of the module at each clock cycle [41]. We then focus on the clock cycles when the peak switching activity is achieved and identify the instructions executed by the module during these clock cycles, as well as those required to create the conditions allowing to trigger the corresponding switching activity. Based on this analysis, it is typically possible to build a macro, i.e., a sequence of instructions leading to a high and sustained level of switching activity.

As an example to illustrate the goal and behavior of this phase, let consider a pipelined processor, and let assume that the target module is the adder existing in the execution stage. By analyzing some existing programs, we can spot the clock cycles where the peak switching activity is achieved, which clearly correspond to when the execution unit executes an ADD instruction. By further analyzing the code, we could then extract the instructions required to set the input parameters for the ADD instruction, as well as to maintain the switching activity at a maximum level in a sustained way. This is the macro representing the output of the phase.

Clearly, phase 1 can be avoided when (as in the previous example) the identification of the instruction(s) achieving the maximum switching activity is easy. On the other side, phase 1 may be crucial when the identification of the above instruction(s) is not easy.

4.3.2.2 Phase 2

The goal of this phase is to act on the macro produced by phase 1 and optimize it, so that the maximum value of sustained switching activity can be obtained.

This goal can be achieved by resorting to an evolutionary approach (e.g., the one described in [43]), which can modify both the input parameters of the macro and its structure, and evaluate the goodness of each solution via simulation, trying to combine the best ones already found, and further improve them. This approach has proved to be generally effective in generating programs able to maximize a given metric.

The basic points of this solution when applied to our problem are

- Random programs implementing the scheme of the macro from phase 1 are generated; the set of initial programs (or individuals) constitutes the initial population
- Each program is internally represented as a graph, whose vertices are the program instructions
- Each program is evaluated in terms of achieved sustained switching activity resorting to a commercial logic simulator; a fitness value is associated to each program, so that a ranking can be performed out of the programs in the current population
- An evolutionary engine takes the current population and generates new individuals using two operators:

    - Mutation: one random individual is selected and one of its characteristics is randomly changed, i.e., the value of a parameter is changed, a randomly selected instruction is removed, a new random instruction is added
    - Cross-over: two individuals are randomly selected, preferring the individuals with the highest ranking, and combining them (by selecting some instructions from the first, and some from the second).
- The new individuals are added to the current population and evaluated (i.e., a fitness value is associated to each of them); the new population is ranked, and the worst individuals are removed from its, thus maintaining stable its size; in this way, the population evolves from one generation to another.
- The process is repeated until a given stopping condition becomes true, corresponding either to reaching a stable point (the maximum fitness did not improve for a given number of generations) or to spending a maximum amount of CPU time.

4.4 Experimental Results

In order to experimentally evaluate the proposed method, we set up an environment implementing it. For phase 2 the environment exploits the μGP tool [43], which provides a complete framework implementing an evolutionary algorithm aimed at maximizing a generic fitness function. For the *SSWA* computation we resorted to Mentor ModelSim, developing some proprietary tools able to post-process the ModelSim reports in order to compute the metrics introduced in the sub-section 4.3.1, given a certain program.

In order to evaluate a program for the considered processor in terms of its SWA, we set up a proper framework able to profile on top of the Mentor ModelSim logic simulator. The purpose of the profiler is to provide the user with an extended execution trace which is required throughout the proposed method. Given a program to be executed on the processor, and a list of modules within the processor to be monitored, the profiler reports for each clock cycle:

- Information regarding the values of some significant internal registers/signals (e.g., the Program Counter value)
- The *nSWA* values for each module.

The extended execution trace can be parsed in post-processing steps in order to enrich the profile. For example, the program listing file (obtained as additional result of the program compile) reports address and data values for each instruction in the source file; such a file can be used to produce a readable list of the instructions in the pipeline, as shown in Figure 4-1.

We also developed a framework implementing the method described in sub-section 4.3.2. By means of a script for the Mentor Modelsim logic simulator, at each clock cycle a file reporting the toggle count on each net of the circuit is generated; after the report, the counters are cleared and the new clock cycle is simulated. Every time a report file is generated, it is processed by an AWK script counting about 350 lines of code, in charge of parsing it and reporting the profiled information corresponding to the considered clock cycle and the target module(s). The profile of the simulation is parsed by an evaluator counting few lines of bash scripting code; the evaluator calculates the $nSSWA_M$ over a given window of M clock cycles. In the first phase, the instructions able to maximize the *SWA* value are identified and converted into a macro for μGP. In the second phase, μGP generates programs based on the macro, which are profiled as

mentioned before; the *nSSWA* value is used as the value of the fitness function to be maximized.
We then applied the method to a MIPS-like open source processor model [44], counting 27,471 logic gates, and 1,882 flip-flops.

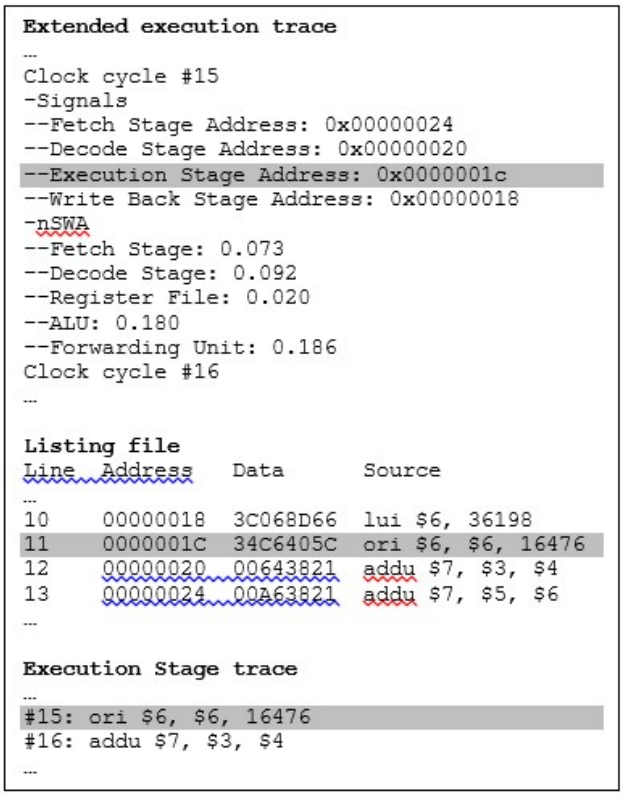


Figure 4-1. Example of program profiling.

All the experiments were performed on 10 cores of a system composed of a dual Intel Xeon CPU running at 2.40GHz and 64 GB of RAM.
We first focused on the Adder existing in the Execution Unit of the MIPS-like processor (counting on 253 gates), and used some existing program as a starting point. This program was generated for test purpose, and targeted the stuck-at fault model, achieving a Fault Coverage of about 88.13% on the whole processor. Table 4-2 summarizes its characteristics. We then used our ModelSim-based environment for computing the relevant metrics, which are also shown in Table 4-2. In this very specific case both the peak and sustained values of the switching activity are equal to 1, since it is possible to find instructions able to force all the nets to switch at each clock cycle, without glitches. A graphical example of the behavior of the initial program in terms of normalized sustained *SWA* over a 10 clock cycle windows ($nSSWA_{10}$) is reported in Figure 4-2.
We then applied the method described in sub-section 4.3.2 and created a small program, whose characteristics are shown in the rightmost column of Table 4-2. As it can be seen, the method is able to achieve significantly higher values of sustained switching activity, equal to the one of the peak switching activity of the initial program.

We reported in Fig. 3 the behavior of the $nSSWA_{10}$ for the final program: the reader can easily see the improvement produced by the method. The initial ramp is due to the instructions to set up the values of the `Add` instructions. Once the maximum value of switching activity is achieved, it can be maintained by repeating the same couple of ADD instructions, each with the proper input values.
The CPU time required by the method is dominated by the one for Phase 2, and corresponds to 11 hours.
The method was then applied to the Decode Unit (composed of 1,073 gates) of the same MIPS-like processor. In this case Phase 1 was crucial, since it was hard to identify a priori which are the instructions which may produce the highest *nSSWA* values. We started again from the same initial program, identified the most promising instruction sequences, and then run Phase 2 to optimize the resulting macro. Table III reports the corresponding results. Once again, the method was able to generate a final program with a maximum value of *nSSWA* much higher than in the initial one, and not far from the peak *SWA* value. Figure 4-4 and Figure 4-5 show the behavior of $nSWWA_{10}$ in the initial and final program. Figure 4-5 also shows that the maximum value for *nSSWA* could be achieved with a sequence composed of two instructions (with proper input parameters) that alternate. These two instructions are `Nor` and `Sltui` (unsigned less-than comparison with a constant). Clearly, their identification was far from trivial. The CPU time required by the method in this case is equal to 50 hours.

Table 4-2. Characteristics of the Initial and final programs for the MIPS adder

	Initial program	**Final program**
Size [#bytes]	1,468	124
Duration [#clock cycles]	16,730	31
PSA_P	1.000	1.000
Max $nSSWA_{10}$	0.700	1.000

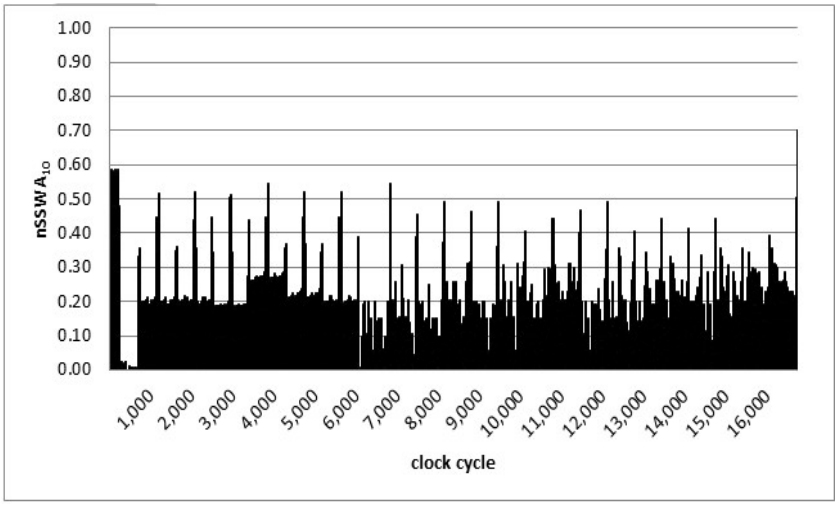


Figure 4-2. Sustained Switching Activity of the original program on the MIPS adder.

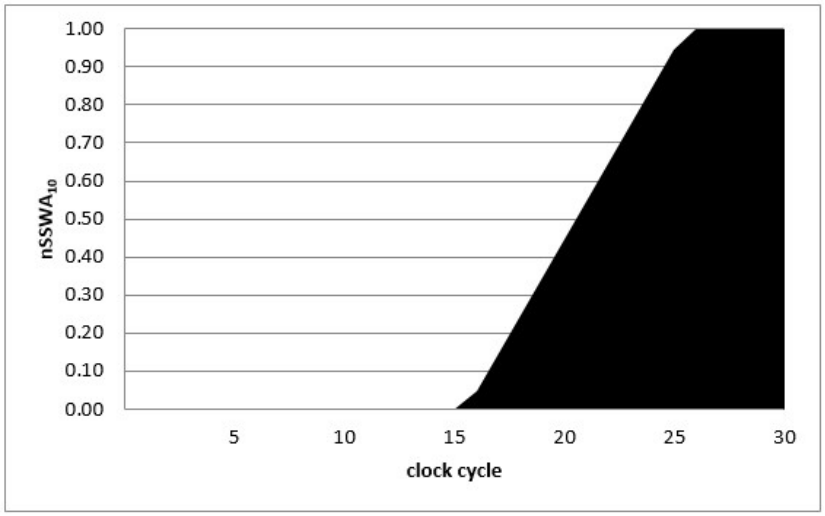


Figure 4-3. Sustained Switching Activity of the final program on the MIPS adder.

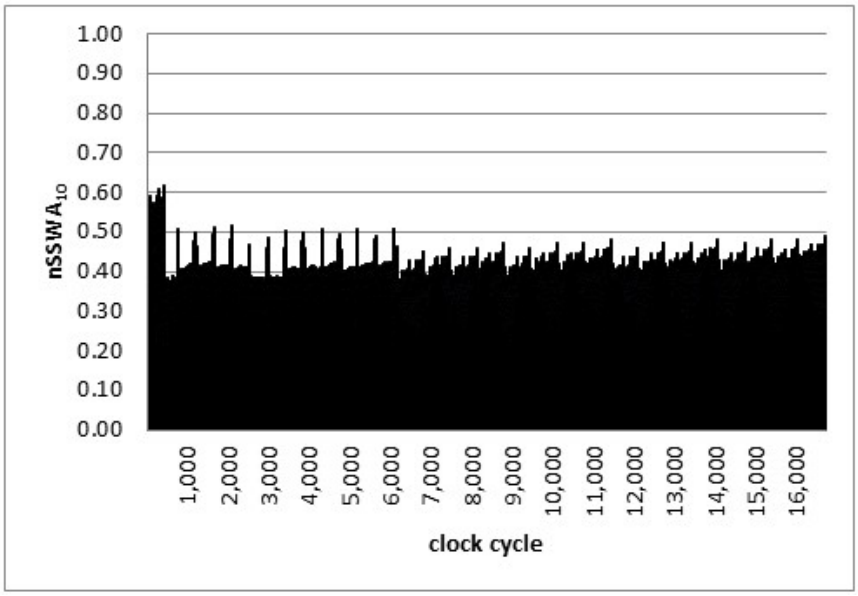


Figure 4-4. Sustained Switching Activity of the initial program on the MIPS decode unit.

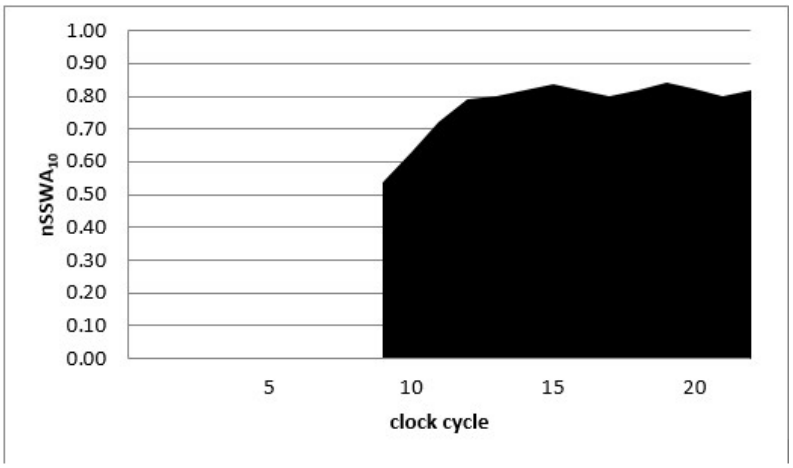


Figure 4-5. Sustained Switching Activity of the final program on the MIPS decode unit.

4.5 Conclusions

Some new applications require maximizing the switching activity of a given module within a circuit, for example to increase the temperature of the corresponding silicon area in a controlled manner, (e.g., to trigger possible intermittent faults).
This Section has described a method for identifying the sequence of instructions able to maximize the sustained switching activity of a module within a processor.
Experimental results gathered on a MIPS-like processor show the effectiveness of the method.

5 Self-Reconfiguring Network

5.1 Introduction

While the semiconductor technology development enables integrated circuits (ICs) with increasing transistor count and decreasing features sizes, it has become crucial to address reliability to handle errors that occur when the IC is in operation [45]. One way to address reliability issues is to embed on-chip instruments capable of detecting errors [46]. Such instruments can be connected to a *fault manager* that makes decisions based on the collected error statuses. The fault manager can be implemented in an on-chip or off-chip processor. To avoid that the *fault monitoring network*, which connects the monitoring instruments and the fault manager, becomes the bottleneck, the network must be designed such that the interval (latency) from when instruments detect errors to when the fault manager gets aware of the errors is low and deterministic. It is also important that the fault manager localizes errors in a short and deterministic time (fault *localization* time) in order to launch suitable recovery actions.

A fault monitoring network can be either *stand-alone*, or part of an existing *functional* infrastructure such as network-on-chip or system bus. The advantage of using the functional network is that no additional hardware cost is needed. One drawback is that adding traffic to transport fault monitoring information may impact the performance of the system. It is difficult at design time to know the amount of traffic information that is to be generated from errors as the traffic depends on when the errors occur. Another drawback is that the predictability of the system is reduced, as the traffic on the functional network might also affect the latency of the fault monitoring information. To be on the safe side, the network might be over-designed to ensure that performance is kept high. This is however costly. The advantage with a stand-alone network is twofold: it does not impact the performance of the system, and simplifies achieving a deterministic fault localization time. The downside of using a stand-alone network is adding extra hardware cost, if it is added only for the purpose of fault management. However, most ICs have stand-alone networks, such as IEEE 1149.1 (JTAG) [47] or IEEE 1687 [48], to enable test, diagnosis, configuration, etc., which makes it attractive to reuse such network for fault monitoring and error handling during operation.

There have been a number of works on networks for transporting monitoring data (for transient faults, timing errors, power estimation, etc.) using a dedicated infrastructure [13][14][15][49][50]. The works in [13][14][15] stand out as they rely on reusing the existing IEEE 1687 network for monitoring purposes.

In this deliverable, for the hardware platform, we assume an IC equipped with embedded monitoring instruments that can detect errors and produce error codes accordingly. Additionally, we assume these on-chip monitoring instruments to be interfaced to an IEEE 1687 network. We propose a scheme where the IEEE 1687 network is self-reconfigured (while maintaining standard compliance) to automatically include the instrument registers containing error codes in its scan-path. The proposed scheme enables very fast error detection, and achieves significantly faster fault localization compared to similar IEEE 1687-based fault monitoring approaches.

The main contributions are as follows:

- a *modified* Segment Insertion Bit (SIB) which can be configured not only in the standard way but also by an embedded fault monitoring instrument

- time analysis on the time needed to detect and localize an arbitrary number of errors
- design guidance for the self-reconfiguring IEEE 1687 network such that the error detection and localization time for a single fault is minimized

To validate our proposed self-reconfiguring networks, we implemented one such network and performed post-layout simulations in 65nm technology. To evaluate the efficiency in terms of time we compare the proposed self-reconfiguring scheme against previous IEEE 1687-based fault management schemes.

5.2 Background and Prior Work

In this section, basics of IEEE 1687 are detailed, and prior work on fault management using IEEE 1687 as the fault monitoring infrastructure is discussed.

The key feature of IEEE 1687 networks is the reconfigurability, i.e., the possibility to switch segments of the network on and off the scan-path. One way to add such reconfigurability is by using SIBs. Figure 5-1(a) shows a simplified schematic of a (possible implementation of a) SIB, in which *select* and control signals (namely, *capture*, *shift*, and *update*) are not shown. The SIB has a shift (S) flip-flop, an update (U) flip-flop, and a two-input scan multiplexer. SIBs in the network are programmed by shifting a bit into their S flip-flop and latching that bit into the parallel U flip-flop. If the latched bit is a "0", the SIB is **closed** and the scan-path is from the *si* (scan-in) terminal, to the *so* (scan-out) terminal via the S flip-flop, bypassing the segment between the *tsi* (to scan-in) and *fso* (from scan-out) terminals. If the latched bit is a "1", the SIB is **opened** and the scan-path includes the segment connected between *tsi* and *fso* terminals of the SIB—referred to as the hierarchical port of a SIB here. The symbol shown in Figure 5-1(b) will be used in the rest of this deliverable to represent a SIB.

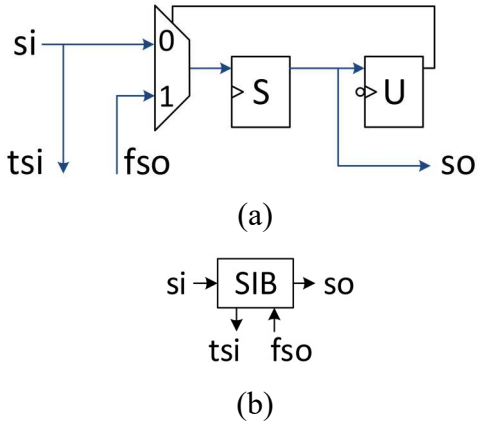


Figure 5-1. SIB: (a) simplified schematic, and (b) symbol

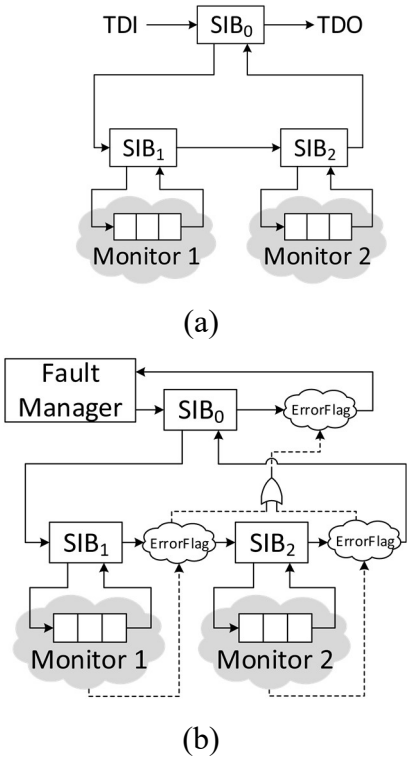


Figure 5-2. (a) Example of IEEE 1687 network, and (b) a simplified representation of the basic idea in [13]

Using SIBs makes it possible to design hierarchical IEEE 1687 networks. Figure 5-2(a) shows a hierarchical IEEE 1687 network consisting of three SIBs and two 3-bit instrument shift-registers, named Monitor 1 and Monitor 2. To access the network via the JTAG TAP, the top (highest) hierarchical level of the network (SIB_0 in this example) is connected as a custom test data register (TDR) between the test-data-in (TDI) and test-data-out (TDO) terminals of the JTAG TAP. In the network shown in Figure 5-2(a), initially, only SIB_0 is on the *active* scan-path. To access, e.g., Monitor 1, SIB_0 should be programmed to be opened, which requires going through Shift and Update states of the JTAG TAP finite state machine (FSM)—referred to as a capture-shift-update (CSU) cycle. Next, SIB_1 should be opened while keeping SIB_0 opened and SIB_2 closed—which requires performing another CSU. At this point, the register for Monitor 1 is placed on the scan-path and can be read from or written to. It can be seen from this example that accessing instruments in an IEEE 1687 network produces two types of overhead: (1) the clock cycles needed to operate the JTAG FSM and (2) the clock cycles needed to shift SIB programming data. It has been shown that the JTAG TAP FSM overhead is negligible compared to the SIB programming overhead [51], and that using a hierarchical design (such as those in Figure 5-2) can reduce the SIB programming overhead significantly [52].

Hierarchical IEEE 1687 networks have been used in fault management schemes to connect instruments to a fault manager [13]-[15]. In [13], methods for optimized design and calculation of error localization time are presented for the proposed fault management scheme. The work in [14] extends [13] by elaborating on how the fault manager can react faster to new faults while the instrument access network is in use for

other purposes and how multiple faults can be addressed, but presents no time analysis method or experimental results for such cases. In [15], a simulation-based platform for experimenting with fault injection and fault management is elaborated, but no time analysis or network optimization method is presented.

Along with the IEEE 1687 network, [13]-[15] use a fully combinational error propagation network which propagates error flags to the highest hierarchical level of the IEEE 1687 network. A simplified representation of the hierarchical networks used in [13] is shown in Figure 5-2. (b) where the error propagation network is marked by the dashed lines. The advantage is that by reading the ErrorFlag register in the highest level the fault manager gets informed of any error in the system without checking each and every instrument. To guide fault localization, [13]-[15] added ErrorFlags at every level, resulting in dramatic increase in fault localization time. Also, fault localization in [13]-[15] involves a number of CSUs to open hierarchical levels, each CSU performed over a scan-path longer than the scan-path for the previous CSU, increasing the fault localization time.

To address the fault localization time, we consider a fault management scheme similar to those in [13]-[15] and propose self-reconfiguration. We show that by adding self-reconfiguration it is possible to reduce the fault localization time significantly while keeping conformity to the IEEE 1687 rules.

In this work, we use the terms fault and error interchangeably even though in practice these two concepts are different, i.e., an error is a manifestation of a fault.

5.3 Self-Reconfiguring Network

In this section, we describe the hardware structure of the self-reconfiguring networks, as well as how to detect and localize errors in the proposed structure.

The basic idea in self-reconfiguration is that when a fault is detected by a fault monitor, the corresponding error code register is automatically included in the active scan-path so that its contents can be readily shifted out and analyzed. Such scheme, improves the speed of fault localization via (1) avoiding to open layers of hierarchy one layer at a time, and (2) using only one single-bit ErrorFlag register instead of placing multiple such registers at each hierarchical level.

5.3.1 Hardware Structure

In this work, we assume a hierarchical IEEE 1687 network interfacing all embedded instruments (test, debug, fault monitors, etc.) in a system to a Fault Manager, which has the purpose of detecting and localizing errors that may occur in different components of the system over time, such that it can initiate necessary fault handling actions. The novelty of this work relies on the fact that part of the hierarchical IEEE 1687 has the feature of self-reconfiguration.

Figure 5-3(b) shows an example of a self-reconfiguring network. Among all the instruments, we assume that there is a set of fault monitoring instruments. In the top level of the hierarchical network, the fault monitoring instruments are connected through a dedicated SIB, denoted with SIB_0, while all the other instruments (test, debug, etc.) are connected through another SIB, denoted with SIB_{ins}. The top level also includes a one bit shift-register (ErrorFlag) to indicate if any errors are detected by any of the fault monitoring instruments.

We assume that a fault monitoring instrument has a *fault flag* output terminal that is set to logic "1" in case a fault is detected. The *fault flag* stays active until it is acknowledged via a *clear flag* input terminal. The fault flag signal will be used as an input to reconfigure the network, such that an access to the fault monitoring instrument

is enabled. Furthermore, the fault flag signal is propagated across the hierarchical levels and is finally captured by the ErrorFlag register in the top level of the hierarchical network.

Additionally, we assume that a fault monitoring instrument produces an error-code which is parallel-loaded during the capture phase into an error-code/mask shift-register (EMR) interfacing the instrument to the IEEE 1687 network. An EMR is assumed to have capture and update features (similar to standard JTAG TDRs) and it contains an error-code field (written by the fault monitor) and a mask field (written by the Fault Manager). Error masking is used to stop a permanent fault from constantly raising the fault flag. To be compliant with the IEEE 1687 standard, error masking should be enabled by default at reset to disable self-reconfiguration of the network. When the EMR of a fault monitoring instrument is selected and data is shifted through it, the *clear flag* is asserted to indicate that the fault from the fault monitor has been acknowledged. In Figure 5-3(b), the 3-bit registers, namely EMR_1 and EMR_2, are the EMRs associated to Monitor 1 and Monitor 2, respectively.

To enable self-reconfiguration, we propose a *modified* SIB, which is the core component in a self-reconfiguring network. A *modified* SIB, while being IEEE 1687 compliant, can additionally be opened asynchronously via a dedicated terminal. The symbol shown in Figure 5-3(a) will be used in the rest of this deliverable to represent a *modified* SIB. In Section 5.7, we detail the circuitry of the proposed *modified* SIB.

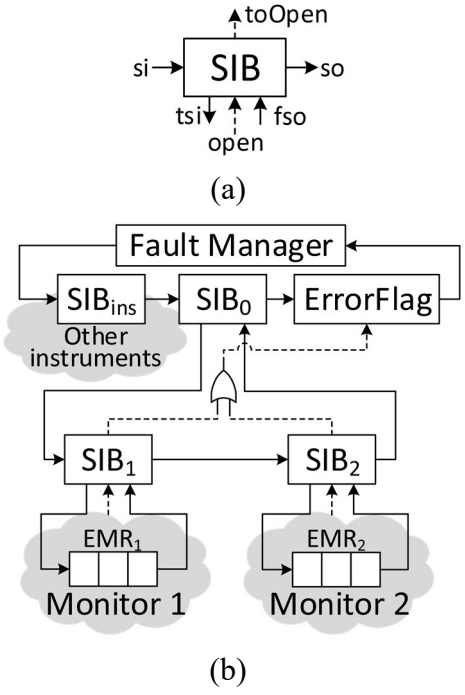


Figure 5-3. (a) Symbol for the *modified* SIB, and (b) An example self-reconfiguring network (the dashed line represents the fault propagation network)

All fault monitoring instruments in the network are connected to the top-level SIB_0 through a network of *modified* SIBs. The main difference between a regular SIB and a *modified* SIB is the pair of terminals "open" and "toOpen". The "open" terminal of a *modified* SIB is connected either to (1) the fault flag of the monitoring instrument—see

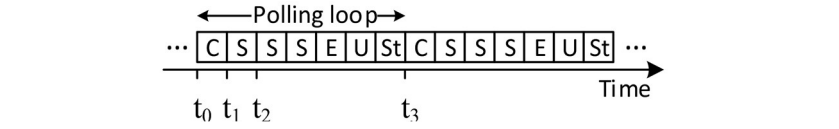
SIB_1 and SIB_2 in Figure 5-3(b)—or (2) the ORed output of the "toOpen" terminals of all *modified* SIBs attached to it (placed one hierarchical level below). When the "open" terminal is asserted (pulled high), it changes the state of the SIB to opened only if the SIB is not already part of an active-scan path. The signal from the "open" terminal is gated internally using (an inverted copy of) the select signal to make sure that the state of the SIB does not change (from closed to opened) when it is part of an active scan-path (see Figure 5-8 for details on the *modified* SIB). The "toOpen" terminal propagates the internally gated signal (from the "open" terminal) via an OR gate either to (1) the *modified* SIB in the hierarchical level above, or (2) the ErrorFlag register in the top level—see Figure 5-3(b). Note that when the fault flag has managed to propagate to the ErrorFlag register, all the *modified* SIBs on the path from the fault monitor raising the flag to the top level SIB_0 are properly configured, i.e. the network has self-reconfigured.

A requirement for a *modified* SIB (as well as for SIB_0 and SIB_{ins}) is to have its shift (S) flip-flop placed after the hierarchical mux (similar to what is shown in Figure 5-1(a)). Such placement, while being fully standard compliant, ensures that during shifting, the state of the SIB is always shifted out first. This is required by the fault-localization method to determine the current configuration of the network.

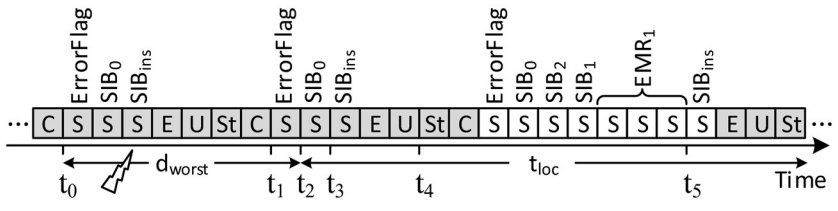
5.3.2 Fault Detection and Localization Method

In this section, we explain the fault detection and localization method with the help of the example network shown in Figure 5-3(b) and the timelines shown in Figure 5-4. The following scenarios are considered: (1) no error has occurred, (2) an error occurs when the Fault Manager is not localizing another error, (3) an error occurs when the Fault Manager is localizing another error, and (4) two errors occur in a short span of time when the Fault Manager is not localizing another error.

For the first scenario, when no error is reported, the Fault Manager constantly checks the status of the system by polling the value captured by the ErrorFlag register. The Fault Manager does the polling via looping constantly through the *Capture*, *Shift*, *Exit1*, *Update*, and *Select* states in the DR branch of the TAP FSM. Since SIB_0 is closed when no errors are being localized, the polling takes seven test clock cycles (TCK)—the interval between t_0 and t_3 in Figure 5-4(a)—as three shifts are required: for SIB_{ins}, SIB_0, and ErrorFlag. The value of the fault flag raised by monitoring instruments is captured at t_1 into ErrorFlag register and can be observed at t_2 (see Figure 5-4(a)). The polling continues as long as the shifted out bit corresponding to the ErrorFlag register is a "0". During polling, zeros are shifted in to keep SIB_0 and SIB_{ins} closed.



(a) C: Capture, S: Shift, E: Exit, U: Update, St: Select



(b)

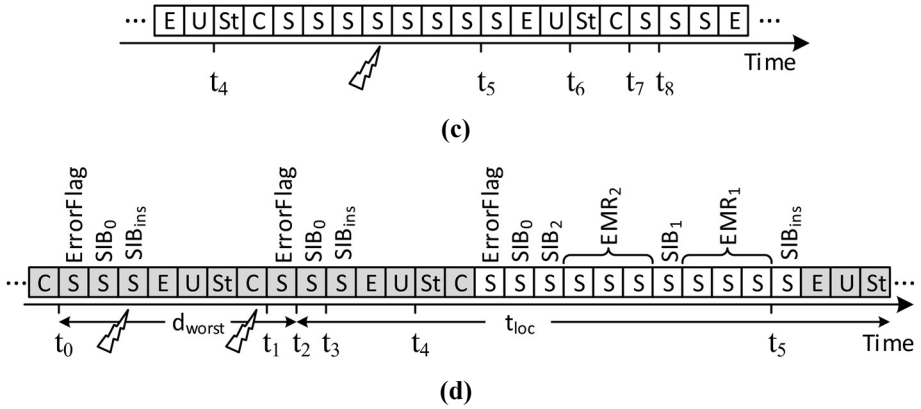


Figure 5-4. The detection and localization method: (a) constant polling to detect a fault, (b) an error is detected and localized, (c) another error happens when the previous one is being localized, and (d) when two faults are detected together.

For the second scenario (see Figure 5-4(b)), consider that a fault happens at the interval between t_0 and t_1 and is reported by Monitor 1. The reason we chose this interval is that no matter when in this interval a fault occurs, it will not be captured until t_1 and will therefore not be detected until shifted out at t_2. We refer to the interval between t_0 and t_2 (which is eight TCKs long) as the worst-case fault detection time (when no other error is being localized) and denote it by d_{worst}. When the value shifted out at t_2 (which belongs to the ErrorFlag) is a "1", the localization procedure is launched by shifting a "1" into SIB_0 at t_3 which takes effect at the following Update phase (t_4). Once SIB_0 is open, as the rest of the network is already self-reconfigured, the Fault Manager starts shifting out data from the network (while shifting in zeros to close the SIBs and reset EMRs on the active scan-path) to localize the fault: The first two bits shifted out are the contents of ErrorFlag and SIB_0. The third bit is the contents of SIB_2 for which a value of zero indicates that SIB_2 is closed and the fault is not reported from the network segment connected to the hierarchical port of SIB_2. The next bit is the contents of SIB_1 which is "1" meaning that SIB_1 is open and the fault is reported from the segment connected to it, i.e., Monitor 1 in this example. The next three bits are the contents of the 3-bit EMR_1 which interfaces Monitor 1 to the IEEE 1687 network. At this point, i.e., at t_5, the error is localized and the error information is retrieved. In this work, however, we include in the localization time (denoted by t_{loc}) the next four TCKs needed to shift-in one more zero for SIB_{ins} and take the TAP FSM back to the capture phase. The worst-case error detection and localization time can then be written as

$$t_{worst} = d_{worst} + t_{loc} \quad \textbf{(2)}$$

In practice, for the above scenario, d_{worst} should be extended to include the time that it takes a fault flag signal to propagate from the fault monitoring instrument to the ErrorFlag. We denote this propagation delay by δ and note that if the fault monitor signals the error later than t_0-δ, it is not captured at t_0. Therefore, d_{worst} should be written as t_2-t_0+δ which is equal to $8/f_{TCK}$+δ where f_{TCK} is the maximum frequency that the JTAG TAP can be operated at.

For the third scenario, when an error happens while the Fault Manager is localizing a previous error, consider Figure 5-4(c) as continuation of the timeline in Figure 5-4(b). As discussed for the second scenario above, at t_4 SIB_0 is opened which puts SIB_1 and SIB_2 on the active scan-path. SIB_0 is closed at t_6 meaning that between t_4 and t_6 SIB_2 is selected (though closed) and, therefore, cannot be opened by a fault flag signal from Monitor 2. That is, any fault reported by Monitor 2 after t_4, is captured at t_7 and detected at t_8. Since SIB_2 is closed, the fault flag from Monitor 2 is not acknowledged and therefore remains active until SIB_2 is opened and the error code from Monitor 2 is captured into EMR_2.

For the last scenario, consider the timeline in Figure 5-4(d), where both monitors detect faults in the interval between t_0 and t_1. In this case, both faults are detected at t_2. In comparison to the scenario for one fault (see Figure 5-4(b)), the localization procedure takes a longer time as this time SIB_2 is also opened and EMR_2 is also included in the scan-path.

As a final note in this section, we observe from Figure 5-4(b) and Figure 5-4(d) that the shaded states are traversed no matter how many faults are being detected and localized. We denote this constant overhead of 18 TCKs by J_{OH}, and write Eq. (2) as

$$t_{worst} = J_{OH} + t_s \quad \textbf{(3)}$$

where t_s denotes the number of shift cycles in t_{loc} and varies by the number of faults being localized.

5.4 Time Analysis

In this section, we present analyses for the worst-case error detection and localization time (t_{worst}) in a self-reconfiguring network, for two cases: when a single fault occurs, and when multiple faults occur such that they are all detected by the Fault Manager at the same time (see the discussion on Figure 5-4(d)).

As shown in Eq. (3), t_{worst} has a constant part J_{OH} and a variable part t_s. For the analyses we focus on calculating t_s.

We present time analyses for balanced tree networks such as the network shown in Figure 5-5. In such networks, each SIB, except for the ones in the lowest level, has k SIBs connected to its hierarchical port. The SIBs in the lowest level are interfaced to the instruments. For $N=k^h$ instruments, the network resembles a balanced k-ary tree whose root is SIB_0.

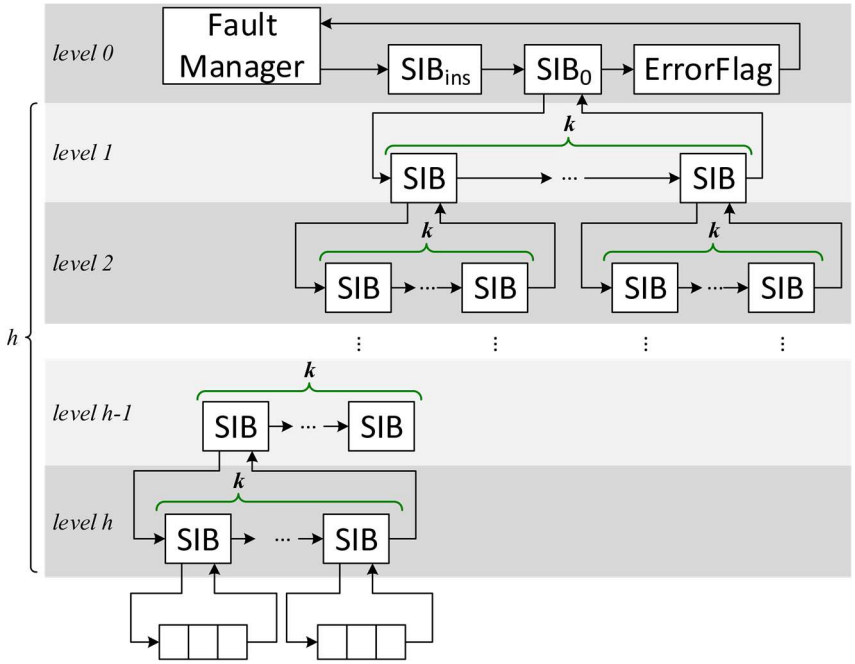


Figure 5-5. A balanced tree hierarchical network

5.4.1 In Case of a Single Fault

Given the network in Figure 5-5, assuming that only one monitor has raised a fault flag causing all SIBs on its hierarchy to change state to open, there are $s = 2 + k \times h$ SIBs on the path to each instrument, where 2 represents SIB_0 and SIB_{ins}. The total shift time t_s is therefore the sum of s and the length of the EMR (denoted by L) plus one for ErrorFlag:

$$t_s = 2 + k \times h + L + 1 = 3 + k \times \log_k N + L \quad \textbf{(4)}$$

5.4.2 In Case of Multiple Faults

Assume that F faults ($F \leq N$) are to be localized at the same time. To calculate t_s, we consider monitors detecting these faults to be spread in the network such they cause maximum possible number of SIBs to be opened (maximizing the length of the active scan-path, thus leading to the longest localization time). As an example, when F=k faults happen in the system monitored via the network in Figure 5-5, the localization time is maximized when each of these k faults happen in the subtree of each of the k SIBs in level 1. Another observation is that for localization of $F \geq 1$ faults, the SIB at level 0 is opened, for $F \geq k$, all SIBs in level 1 are opened, for $F \geq k^2$, all SIBs in level 2 are opened, and so on. The number of these SIBs, which are on the scan-path to all F monitors (i.e., shared by all of them), is captured by $\sum_{i=0}^{r} K^i$ where $r = \lceil \log_k F \rceil$ is the number of upper levels in which all the SIBs are open. Next, to calculate the number of SIBs exclusively on the path to each of the F monitors, we note that $h-r$ remaining lower levels are open exclusively for each fault, each having k SIBs—thus $F \times k \times (h - r)$ is the total number of SIBs exclusively opened for the F monitors. To sum up, the total number of SIBs on the scan-path for the F faults is $s = 1 + \sum_{i=0}^{r} K^i + F \times k \times$

$(h-r)$, where 1 is for SIB_{ins}. To calculate t_s, we need to add to this number of SIBs, the total length of EMRs on the scan-path (i.e., $F \times L$) as well as one for the ErrorFlag, as follows:

$$t_s = 1 + \sum_{i=0}^{r} K^i + F \times k \times (h-r) + F \times L + 1 \quad \textbf{(5)}$$

5.5 Network Design

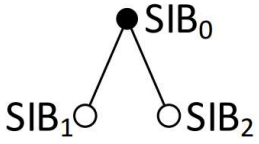
In this section, we describe a design method for the proposed self-reconfiguring networks such that t_{worst} for a single fault is minimized. As t_{worst} has a constant part J_{OH} and a variable part t_s (see Eq. (3)), minimizing t_{worst} reduces to minimizing t_s. For a single fault, t_s is calculated using Eq. (4). Assuming that the number of instruments N is given, to find k which minimizes t_s, we set the first derivative of t_s w.r.t. k to zero:

$$t_s = 3 + \ln N \times \frac{k}{\ln k} + L \;\Rightarrow t_s' = \ln N \frac{\ln k - 1}{(\ln k)^2} \quad \textbf{(6)}$$

$$t_s' = 0 \Rightarrow \ln k = 1 \Rightarrow k = e \quad \textbf{(7)}$$

Since $e \cong 2.72$, we should either choose k=2 or k=3 to minimize t_s. However, given an arbitrary number of instruments N where N is not a power of two or three, it is not possible to construct a balanced tree. In such cases, a straightforward way to construct the network is to create a balanced tree for $k^{\lceil \log_k N \rceil}$ instruments, where k=2 results in a binary tree and k=3 results in a ternary tree, and prune the tree (after placing the N instruments at the leaf nodes). Pruning can be done by removing the internal nodes to which one or no instrument is connected. After pruning, one can compare the results from the pruned binary and ternary trees and pick the better one. In the following, we present a network construction method that by mixing binary and ternary subtrees yields similar or better results compared with each of the pruned binary and ternary tree alternatives.

Let us now switch to a simpler network representation which is more suitable for the discussion in this section. The tree in Figure 5-6(a) captures the hierarchical relation (and not the data connections) between the SIB components in the network shown in Figure 5-3(b). The instruments are not shown (as the length of instruments' shift-registers has no effect on SIB shifting overhead) and those SIBs directly connected to instruments are represented by empty circles. In Figure 5-6(a), node SIB_0 is parent to sibling leaf nodes SIB_1 and SIB_2. When a parent SIB is opened, its children are on the scan-path no matter if they are opened or closed. In other words, when a node is on the scan-path, all its siblings are also on the scan-path.



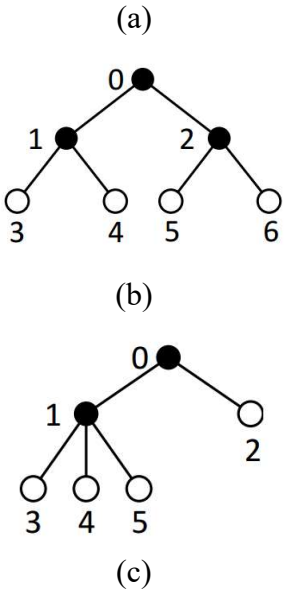


Figure 5-6. Alternative representation of networks, where filled circles represent the SIBs which are not directly connected to instruments, empty circles represent SIBs connected to instruments, and edges represent the hierarchical relations: (a) representation of network in Figure 5-3(b), (b) and (c) networks for four instruments

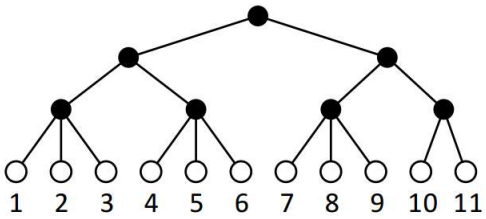


Figure 5-7. Representation of a self-reconfiguring network for 11 instruments

As the proposed network construction method is based on bundling instruments in groups of three, we would first like to make an observation for when the remaining number of instruments is one, i.e., when $N \bmod 3=1$. Figure 5-6(b) and Figure 5-6(c) show two networks constructed for four instruments. When in the network in Figure 5-6(b) a fault is detected at instrument connected to the SIB at node 3, that SIB (node 3) as well as the SIB at node 1 are opened. This means that in total five SIBs are on the path (namely, nodes 0–4) and it therefore takes five clock cycles to read their status. In this case, as all instruments have the same number of SIBs on their scan-path, the average-case and the worst-case fault localization time is the same for all of them. This, however, is not the case for the network represented in Figure 5-6(c) in which for the instrument connected to node 2 three shift clocks are needed while for those connected to nodes 3–5 six shift clocks are needed—averaging to $(3\times6+1\times3)/4=5.25$ clock cycles. It can be seen from this example that the network represented by Figure 5-6(b) results in both better average-case and worst-case shifting time.

Based on the above observation, we propose the following construction algorithm. For given N instruments, we bundle the instruments into clusters of three instruments each. When N is a multiple of three, we will have $c=N/3$ clusters. If N is not a multiple

of three, one or two instruments will remain. Following the observation made for Figure 5-6(b), when the number of remaining instruments is one, we make $c=\lfloor N/3 \rfloor - 1$ three-instrument clusters plus two two-instrument clusters. If, however, the number of remaining instruments is two, we make $c=\lfloor N/3 \rfloor$ three-instrument clusters plus one two-instrument cluster. Assuming each cluster to be an instrument, the same procedure described above can be applied to the created clusters, creating clusters of clusters until the network is complete. Figure 5-7 shows this procedure for 11 instruments.

5.6 Comparison with Similar Approaches

We have compared our proposed self-reconfiguring IEEE 1687 network with the work presented in [13] (which uses a regular IEEE 1687 network for monitoring) with regards to t_{worst} (see Eq. (2)). In this section, we present the results of the comparison for the case (1) one fault occurs (discussed in Section 5.4.1), and the case (2) multiple faults detected by the Fault Manager at the same time (discussed in Section 5.4.2).

5.6.1 For a Single Fault

Table 5-1 shows the results of comparison with the approach proposed in [13]. For the construction of the proposed self-reconfiguring network, we compared four alternatives: (1) using the network construction method for regular IEEE 1687 networks presented in [52], (2) a binary tree with pruning, (3) a ternary tree with pruning, and (4) the construction method proposed in Section 5.5. To calculate the number of SIBs on the scan-path for the self-reconfiguring networks, pre-order tree traversal is employed. A fixed number of $J_{OH}+2=18+2$ TCKs is added to the calculated shift time to account for the constant overhead (see Section 5.3.2), ErrorFlag, and SIB_{ins}. Moreover, another three TCKs are added to account for the length of the fault monitor's EMR (i.e., $L=3$).

From the results, it can be seen that by using the proposed self-reconfiguration scheme (regardless of the considered network tree construction method), at least 2.6x reduction in localization time is achieved compared to [13]. The reason for this improvement can be attributed to opening many hierarchical levels in a single CSU and having only one single-bit ErrorFlag register directly on the scan-path.

Among the construction methods examined for the self-reconfiguring network, the one described in Section 5.5 performs up to 17% better than the method in [52] and results in better or equal t_{worst} compared to binary and ternary trees.

Table 5-1. t_{worst} for a single fault (in TCKs)

Number of instruments	[13]	Self-reconfigurable networks			
		tree from [52]	binary tree	ternary tree	this work
25	90	35	34	33	33
50	118	37	36	35	35
100	158	39	38	38	37
200	206	42	40	39	39
500	266	52	42	42	42
1000	326	53	44	44	44

5.6.2 For Multiple Faults

The work in [13] has not presented analysis and results on multiple faults. On the other hand, our calculations for multiple faults (see Section 5.4.2) are for balanced k-ary trees, and cannot be directly used for the network structures presented in [13]. Therefore, to perform the comparison, we (1) used constraint programming (by using the constraints

formulation in [13]) to get the optimal network structure for the number of instruments suitable for our analysis (see below), and (2) developed time analysis for multiple faults for the networks presented in [13] based on their time analysis for a single fault and our analysis for multiple faults.

As for the number of instruments, we chose numbers which are powers of three resulting in networks resembling balanced ternary trees. For each of these networks, we calculated t_{worst} for 1 to 10 faults. For each pair of network and number of faults, we calculated t_{worst} using Eq. (3) where J_{OH}=18 and t_s is calculated using Eq. (5). The results are presented in Table 5-2 where the shaded rows present the numbers obtained for the network structure type proposed in [13]. The last row, presents the average improvement ratio achieved over [13] in case of multiple concurrent faults which ranges between 6.2 to 7.8 times improvement. As was the case for single faults, the reason for this improvement can be attributed to opening many hierarchical levels at once and having only one ErrorFlag register directly on the scan-path.

Table 5-2. t_{worst} for multiple faults (in TCKs)

# instruments	Number of faults									
	1	2	3	4	5	6	7	8	9	10
27	33	42	51	57	63	69	75	81	87	90
	90	126	162	162	162	162	162	162	162	162
81	36	48	60	69	78	87	96	105	114	120
	146	202	258	314	370	426	426	426	426	426
243	39	54	69	81	93	105	117	129	141	150
	334	538	742	946	1150	1150	1150	1150	1150	1150
729	42	60	78	93	108	123	138	153	168	180
	298	486	674	862	954	1046	1138	1230	1322	1414
2187	45	66	87	105	123	141	159	177	195	210
	394	662	930	1118	1306	1494	1682	1870	2058	2246
Average ratio	6.2	7.1	7.5	7.8	7.8	7.5	7.1	6.7	6.4	6.3

5.7 Practical Issues

To validate our proposed self-reconfiguring networks, we implemented one such network and performed post-layout simulations (in 65nm technology) for the scenarios discussed in Section 5.3.2. Figure 5-8 shows our implementation of the proposed *modified* SIB. Before discussing Figure 5-8, we should mention that depending on the available standard cell library, simpler designs with the same functionality are possible, and that our implementation is affected by our ASIC vendor's library.

In Figure 5-8, the clock signal is not shown to avoid clutter. The *Reset* signal is the synchronous active-low reset from *Test-Logic-Reset* state in the JTAG TAP FSM. The self-reconfigurability revolves around the *U′* flip-flop which is D-type with asynchronous active-high *set*. The *set* input of *U′* is connected to a gated copy of the *Open* terminal of the SIB. The *Open* signal is gated via the *Select* signal so that the self-reconfiguration only happens when the SIB is not selected (i.e., not part of the active scan-path). The *Q* output of *U′* is used to open the SIB—i.e, include the segment connected between *TSI* and *FSO* terminals in the scan-path. The output of *U′* is captured

into the *S* flip-flop (as required by the localization method described in Section 5.3.2) when the TAP FSM goes through the capture phase. *U′* is cleared when the TAP FSM goes through the *Update* phase or through the *Test-Logic-Reset* state during initialization.

To give an idea of the propagation delay (denoted earlier by δ) in a large network, we constructed a network with 2187 instruments (which is a balanced ternary tree with seven layers) and performed synthesis and place & route (optimized for a 100MHz TCK) using a 65nm technology. The reported delay between each of the 2187 instrument fault flags and the parallel input of the ErrorFlag register (i.e., through all seven layers) shows a minimum delay of 1.73ns, average delay of 2.1ns and maximum delay of 2.62ns (still shorter than one TCK period).

As an example, assuming an on-chip Fault Monitor which can operate the network at 100MHz, the worst-case localization time (in seconds) for the case of one fault happening in a network with 2187 instruments (see Table 5-2) is calculated as $45\times 1/100\times 10^{6} + 2.62\times 10^{-9}$ which is 452.62ns.

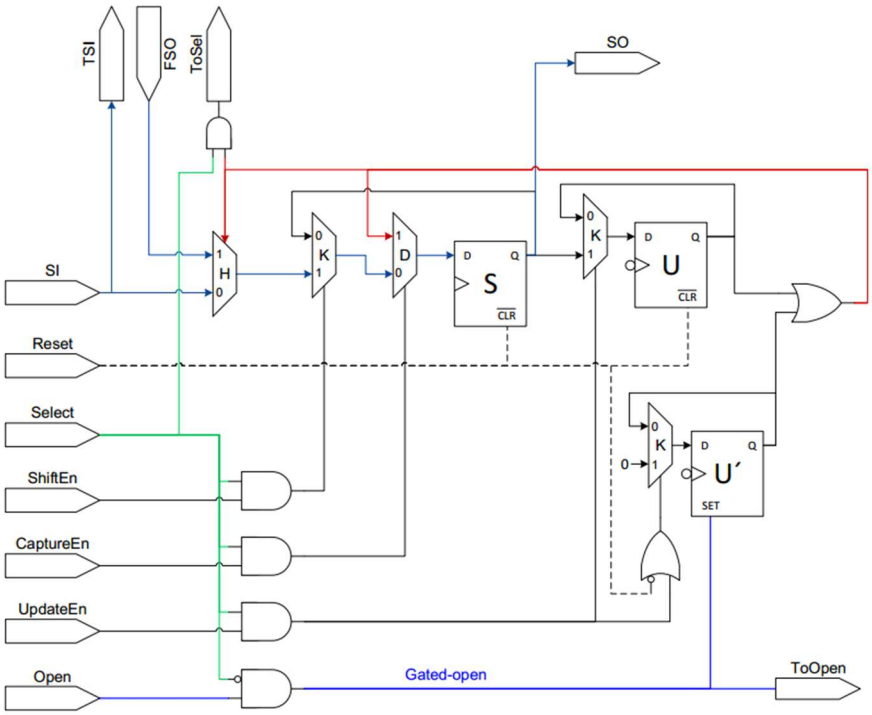


Figure 5-8. Schematic of the proposed *modified* SIB

5.8 Conclusion

Fault localization time is reduced by introducing a segment insertion bit (SIB) that enables self-reconfiguration of IEEE 1687 networks. We presented timing analysis for single and multiple concurrent faults, as well as a construction method for the proposed self-reconfiguring network. We validated the idea of self-reconfiguring networks through post-layout simulations of one such network. We compared the proposed scheme with a previous similar work and observed at least 2.6 times reduction in localization time for a single fault and 6.2 times reduction in case of multiple faults.

6 Instruments for High-performance Test and Diagnosis

The importance of timing related faults (TRFs) detection during a board level test was reported in D1.3 and [53]. This set of dynamic faults may seriously influence the behavior of end-product leading to a complete stop of a system in a bad scenario. At the same time, contemporary test techniques such as Boundary Scan or Functional Test (FT) are not targeted to catch TRFs during a test leaving a place for undiscovered defects [54]. The effective test application speed (F_{app}) of Boundary Scan is simply not sufficient to support modern protocols and communicate with high-speed devices [55]. Functional test on its turn applies test patterns at-speed but it does not produce measurable coverage of structural faults (including TRFs) and is not suitable for debug and diagnosis [56]. As a third alternative, ad-hoc FPGA design that is capable to send test patterns at-speed can be used [57]. This solution is relatively easy to implement by using native controllers available from FPGA vendors for communication with devices under test (DUTs). However, in this case, a user typically has no guidance over low-level communication protocol. Thus, the user has no control over selection of test patterns and how they are applied to DUT, which makes the approach in most cases ineffective for TRFs detection.

In the next sections, we present two developed FPGA-based embedded instruments that are designed to detect timing related faults and provide a diagnosis based on achieved test results.

6.1.1 Universal Virtual Instrument for Transition Delay Fault Test on FPGA Pins

This subsection describes the architecture of a fully-flexible reconfigurable instrument capable of delivering at-speed transitions on any combination of FPGA pins. The instrument belongs to the class of virtual instruments where a hardware part is fixed and only software defines the operations of instrument [58]. These instruments are designed in a special way that allows run-time reconfiguration (RTR) of the instrument to test the particular product. The key advantage of the RTR approach is that such instruments can be instantly used for every SUT and does not need to be recompiled for a new product or after a product change [59][60].

The schematic architecture of developed test system is shown in Figure 6-1. In our approach, we assume that the FPGA-based embedded instrument is controlled by an external tester via the JTAG link. The tester is responsible for the generation of test patterns and an analysis of test results (diagnosis). The role of embedded instrument is to apply all the patterns in a strict pre-defined order. This is required to create specific conditions on a communication bus, which are mandatory to detect timing related faults as it was reported in D1.3. Besides the correct order of test pattern application, the instrument executes test patterns at high speed meeting timing requirements of DUT specification. In this way, our instruments enable a test for pin-level and net-level transition faults described in D1.3.

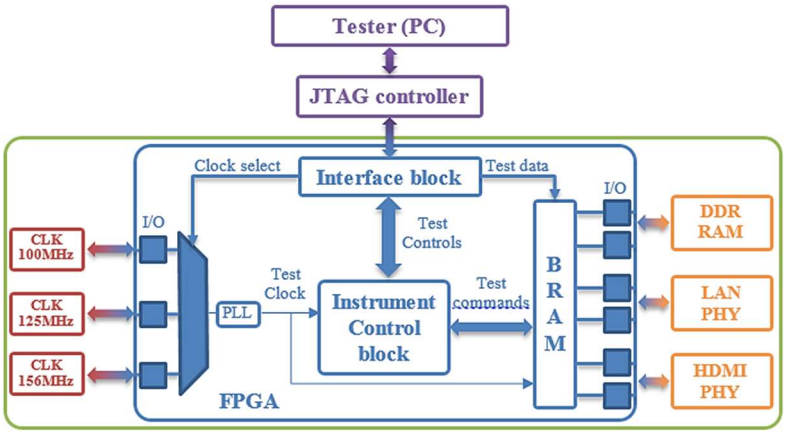


Figure 6-1. Schematic the architecture of the instrument

The developed embedded instrument consists of four main blocks:

- "Interface block". It is mainly responsible for test data communication between external tester and instrument blocks. It receives data via JTAG channel and depending on the operation passes the information to the corresponding block.
- "Clock selection block". This block combines the external clocks coming to FPGA and based on the provided control signal selects the required clock for the core logic. Before entering core logic the clock signal goes also through a PLL which is used to align the clock and if necessary multiple the input frequency.
- "Application block" is used for storage and application of test patterns to devices under test. The key features of this block are execution of test patterns according to the specified order, possibility to drive and measure signals simultaneously and apply the patterns at high speed.
- "Instrument control block". This block is a central operational core of the instrument. It controls the behavior of "Application block" by setting it to different functional modes in respect to a particular test case.

The simplified working principle of the instrument can be described as follows. At first, we collect test patterns into an on-chip memory and then apply them at once to a device under test at high speed. All major FPGA vendors (Xilinx, Altera and Lattice) provide the capability to use FPGA internal memory blocks (BRAM) inside a user design. The total size of these memories varies depending on the size of FPGA device starting from only several kilobytes ending up to tens of megabytes per FPGA. The amount of memory defines the depth of the buffer, i.e. how many patterns can be preloaded at once. Sometimes, in order to perform the full test, all test patterns have to be split into chunks that fit into BRAM buffer size. In this case, the test patterns in each chunk should perform the finished action. For example in case of a memory test, a sequence of writing and reading of test data to the particular memory addresses should be performed.

Each on-chip memory block is configurable to different data width and memory depth relatively. Physically it consists from two independent access address ports (port A and B) with their corresponding output data ports. The RAM contents are shared between these two ports and can be addressed simultaneously. Both ports are able to perform read or write operations depending on the *Write_enable* input port. During the

write operation memory block can also simultaneously perform read operation. In our instrument, we configure BRAM as a dual-port memory. One port is used to write new test patterns or measured responses from DUT into the memory. The second port is configured as a read-only and is used for test pattern application and for reading back the DUT responses. Separate read and write ports give a possibility to both drive and capture measured test values simultaneously.

Both read and write operations are synchronous and require only one clock edge. Thus, the test application speed (F_{App}) of instrument corresponds to the F_{TEST_CLK} supplied to the memory block. This test clock is provided from the "Clock selection block" and if F_{TEST_CLK} equals to normal operational speed of DUT, then the patterns are applied at-speed. For each device under test the nominal clock speed is different. For example, 100Mbps or Fast Ethernet standard assumes communication at 25MHz, while 1000Mbps or Gigabit Ethernet at 125 MHz and HDMI at 156.25MHz clock.

It is worth to mention that the presented architecture is limited with physical characteristics of FPGA I/O and memory blocks. In consequence, the maximal test application speed rarely exceeds 200MHz, which is enough for communication with above-mentioned devices, but too slow to work with contemporary high-speed devices, like DDR3/4 memories. For these devices, the test will not be applied at-speed. Although such test will also cover part of timing-related faults (see experimental results in D1.2), the test does not guarantee marginal defects detection.

The instrument's architecture is also supplemented with a built-in comparator for just-in-time comparison of DUT responses with the expected ones. The patterns of expected responses are stored in a part of the memory block beforehand. During pattern application, the comparison of DUT responses with the expected ones is performed and in case of mismatch, the error signal is set.

The memory block is supervised by "Instrument control block" which defines the address, data and control signals. In our instrument, we have implemented the following operational modes:

- *Load* mode. This mode is used to preload test patterns into the memory. During this mode, the test data from the external tester is stored in RAM contents one by one until the buffer limit is over. If needed, the expected DUT responses are also passed to the buffer.
- *Execution* mode. This is the key mode when all the preloaded test patterns are applied to the DUT while the responses of DUT are concurrently stored back to the memory. If built-in comparison is used, the responses of DUT are compared with the expected ones and the error signal is set in case of mismatch.
- *Read-out* mode. This mode is used to read back the DUT responses from the memory for external analysis.
- *Error signal read-out* mode. This mode is used to determine whether error signal was set during test pattern application (in *Execution* mode). If there is no error flag, we can continue to perform the test. If the error flag is triggered, then we read back the measured values for future analysis before continuing the test.

The test execution flow for the instrument is the shown in Figure 6-2. First, the correct clock source is selected depending on a device under test. Then, the external tester generates all the test patterns according to the fault models described in D1.3 and sends them one by one to the embedded instrument via JTAG link. When the memory contents are full, these patterns are being applied to the device under test. The responses of DUT are captured and written back into the memory simultaneously during the application of test patterns. When the application is finished, these responses can be read out to the external tester for analysis. Afterwards, next patterns are loaded to the buffer and then executed. This cycle is continued until all test patterns have been applied to the DUT.

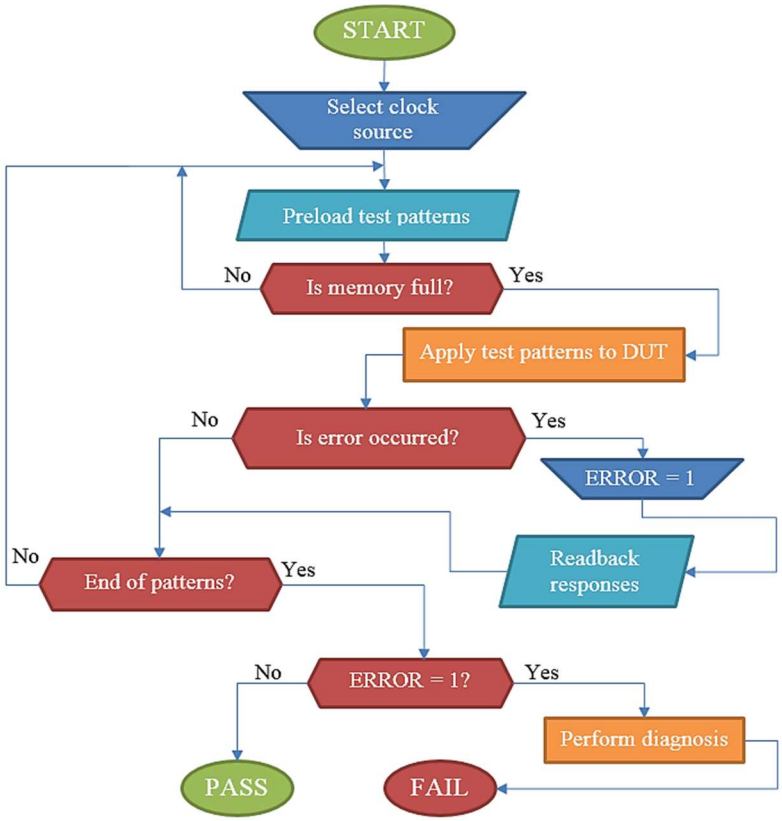


Figure 6-2. Execution test flow

As it was stated in the beginning of the section, the developed instrument belongs to the class of Universal Virtual instruments and support the described functionality on all I/O pins of FPGA. This provides an ability to use single implementation of instrument for any kind of test setup (location of DUT relatively to FPGA pins).

6.1.2 Instrument for Marginal Timing Related Fault Detection on DDR4 Interface

Due to limitations of the method described in previous sub-section in testing DDR memory interfaces, we have developed a dedicated embedded instrument for DDR3/4 memory interface.

Since TRFs are essentially dynamic type of faults, only high-speed (at-speed) test methods can be used to identify them. Moreover, testing at-speed does not guarantee catching all TRFs since the combination of test patterns should be carefully chosen.

Certain classes of TRFs (e.g. crosstalk) manifest themselves only at full speed and with the proper set of test patterns (which may be unlikely appear during normal operational of the system). All these facts make testing the system for TRFs a complex and challenging task.

The situation becomes even worse when considering another category of faults - Marginal TRFs (MTRF). A defect, causing MTRFs typically does not prevent the correct operation of an interconnection line on the PCBA. Instead, such defects lower the quality of a transmitted signal, to the degree which is close to the operational threshold. Unlike conventional TRFs (e.g. delay-faults or cross-talks), MTRFs are not detectable by regular test methods (including at-speed functional and structural tests). Indeed, if a signal line is fully operational while PCBA is being checked at test station, then any combination of test patterns will lead to the correct test result. However, MTRFs can manifest themselves in the field, especially under the pressure of ageing and/or environmental conditions (e.g. temperature, mechanical stress, etc.). But even then, an occurred fault may appear as a single event which may not be reproducible during troubleshooting. As a result, MTRF can lead to sporadic failures of electronic products in the field.

The identification of MTRF defects requires checking the quality of the interconnect and thus implies measurement of an analog waveform of a digital signal. The direct measurement is not possible in most cases, since high-speed digital interconnect signals are normally not accessible externally. Instead, indirect measurements can be carried out by construction of so-called BERT diagram of the signal.

In particular, it is possible to use a special calibration mode of high-speed DDR3/4 memories, which is normally intended to fine-tune bus timings, with respect to the particular PCBA type. During calibration procedures, a memory controller is shifting the sampling point on a particular data line (by phase), while it tries to find the optimal sampling offset. To construct a BERT diagram, we can apply a transition stimulus to an interconnection line and acquiring a signal value with the shifted sampling point. By iterative measurement of the same test stimuli (signal transition) in all possible sampling points, within the signal's unit interval and repeating these measurements many times, a BERT diagram can be recorded for a signal line. Such BER diagrams show if the sampled data is always correct in an optimum sampling point and also how much margin is left before communication will fail.

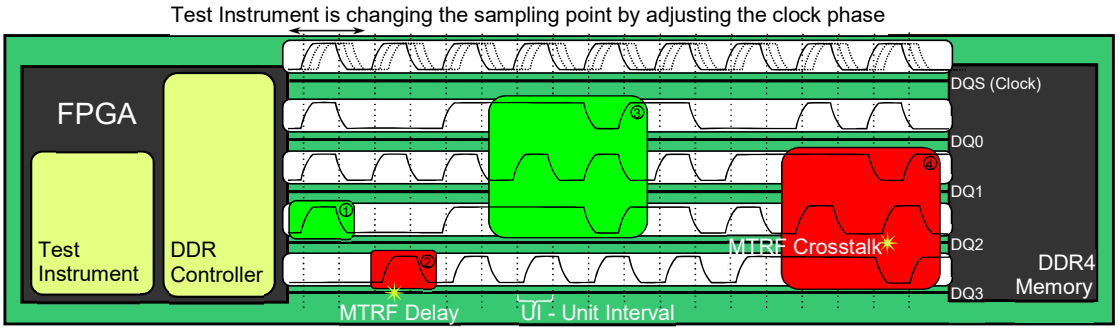
By analysis of constructed BER diagrams, it is possible to identify possible Marginal TRFs on interconnection lines of DDR memories. Several approaches of such analysis have been proposed.

6.1.2.1 Suspicious deviation detection

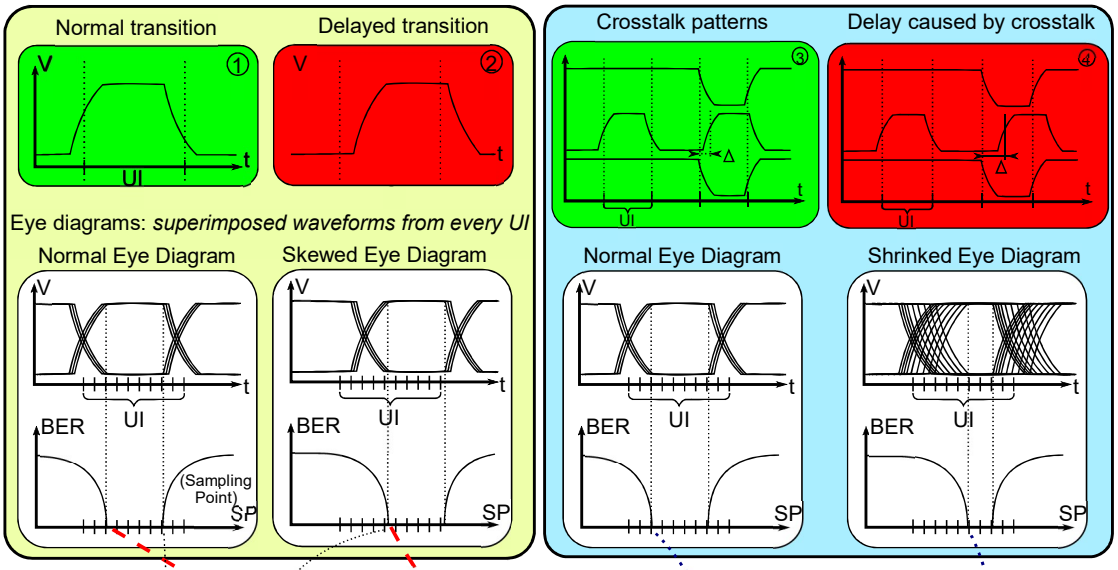
The suspicious deviation approach is based on analysis of measured transaction moments, on a large quantity of manufactured PCBAs. With help of statistical distribution of interconnection line delays, it is possible to locate suspicious outliers. The outlier delays can be caused by either marginal process variations or independent defects that are adding unusual delay or crosstalk susceptibility. The drawback of this approach is that it requires recording of measurement results for sufficient samples of PCBAs of the same type.

6.1.2.2 Bus margin check

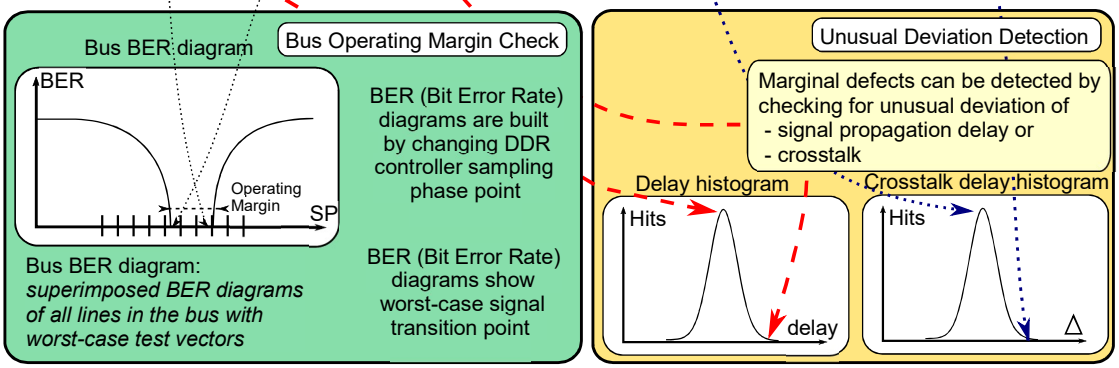
With the bus margin check approach, it is possible to find how much margin (i.e. width of acceptable signal window) is left on an interconnection bus. By subtracting the worst case delay value from the worst case speedup transitions of the next cycle, calculation of the worst case margin can be derived. As a result, a PCBA on which has a measured margin less than defined, will fail the test. However, this method requires a definition of an acceptable margin which could be difficult to assess.



a) Data Communication over Memory Interface: DQS Phase, normal transitions, delays and crosstalk



b) The effect of delays and crosstalk on BER diagrams



c) Detection of outliers

Figure 6-3. Delays and crosstalk on the memory bus; respective eye and BER diagrams

6.1.2.3 Single outlier detection

It is also possible to compare BERT diagrams recorded for DDR interconnection lines on a single PCBA with each other to find out a possible outlier. First of all, the BERT diagrams have to be normalized with respect to the lengths of the corresponded traces on PCBA. Typically, data lines connected to a memory chip are of different lengths (by design) thus source BERT diagrams will have skew against each other.

Then, two properties are extracted for each of normalized BERT diagram: the center of the error-free region on the BERT diagram and its width. By analyzing the results using methods based on standard, or median, deviation, we can observe several possible situations:

1. BER diagrams of lines are identical. No outlier is detected.
2. Error free region in BER diagram of one line is skewed relatively to other lines. This means that this line is delaying signal more than the others and is probably an outlier.
3. Error free region in BER diagram of one line is narrower than for other lines. The line with the more narrow error free region has either a slow-to-rise/fall defect, or is prone to jitter more than others.

6.2 Test Optimization Techniques

In [61] we presented a BIST method for SAR ADCs, as described by McCreary et al. in [62]. The presented method allows ascertaining the satisfactory function and performance of such an ADC. Unlike the state of the art for the test of mixed-signal blocks, the BIST measurements need no stimuli and are conducted internally within the ADC. Thus, they can be used to perform production test as well as checking "system health" by performing them "In the Field". This sub-section presents further optimizations of the BIST method.

6.2.1 A Refresher on the BIST Methodology

The proposed methodology relies on measuring the capacitance ratios of the DAC used in the McCreary ADC. By stimulating all analog parts of the ADC under test to perform the measurements, the BIST method not only allows assessing the performance of the ADC, but it also performs a structural test of all its analog components.

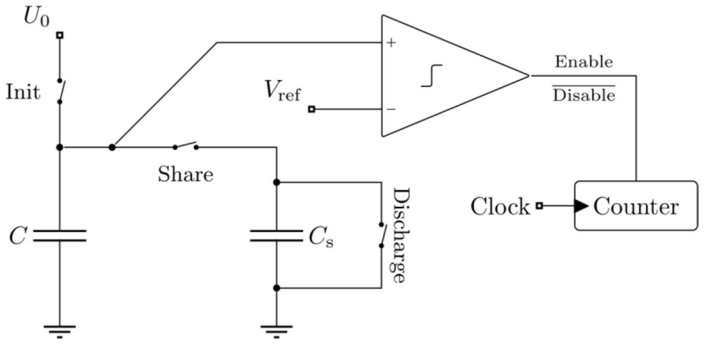


Figure 6-4. Measuring the discharge time of a capacitance

Figure 6-4 depicts the basic principle used in the Capacitance-Measurement-BIST; a small capacitance C_s is used to discharge a capacitance C of unknown value. The time needed to discharge C down to a voltage level of V_{ref} depends on the capacitance ratio $\frac{C}{C_s}$, and is measured by using the ADC's comparator to enable a counter at the start of the measurement, and to disable it when V_{ref} is reached. At the end of the measurement the counter value is proportional to the time needed for the discharge.

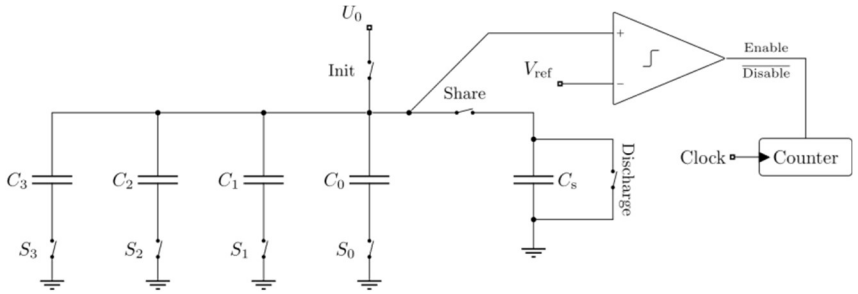


Figure 6-5. Example of a 3-bit ADC with capacitance measurement

Figure 6-5 shows an example of a SAR-ADC that has been extended with the BIST circuitry to perform the measurement. By using the appropriate switch $S_{\{0,1,2,3\}}$ the ratios of the individual capacitances can be measured. The measured ratios allow calculating the INL, and DNL values of the ADC. For further details please refer to [61].

6.2.2 Measurement results of two implementations of the capacitance measurement method

Apart from the already reported measurement results from [61], two additional test chips have been designed processed and measured in the lab, in order to explore the capabilities of the methodology.
One of the test structures was an 8-bit ADC, this allowed us to verify the methodology in the case of a rather coarse ADC. The other consisted of an 11-bit ADC to explore an accuracy of the methodology and its ability to detect errors.

6.2.2.1 Eight bit ADC Measurement Results and Comparison with the Capacitance Measurement Method

The capacitance measurement technique has been implemented in an eight bit ADC. The discharge circuitry used in this case allowed for a linear discharge of the capacitance to be measured, this was deemed necessary because the capacitance ratios used in this ADC would have made it impossible to use the linearization technique discussed in [61].
Measurements of INL and DNL were done in the lab by applying a ramp signal to the inputs of the IC, sampling the code histograms, and calculating the relevant values.
Figure 6-6 shows the characterization results obtained using the "traditional" ADC measurement procedure, e.g. applying a ramp at the ADC input, gathering the histogram data of each code, and calculating the INL and DNL values from the raw data. On the left of Figure 6-6, one can see the applied ramp values and the ADC's output codes, and on the right the calculated values of INL, DNL, histogram contents, and the TUE (Total Unadjusted Error, i.e.: error of the ADC before applying offset and gain correction).

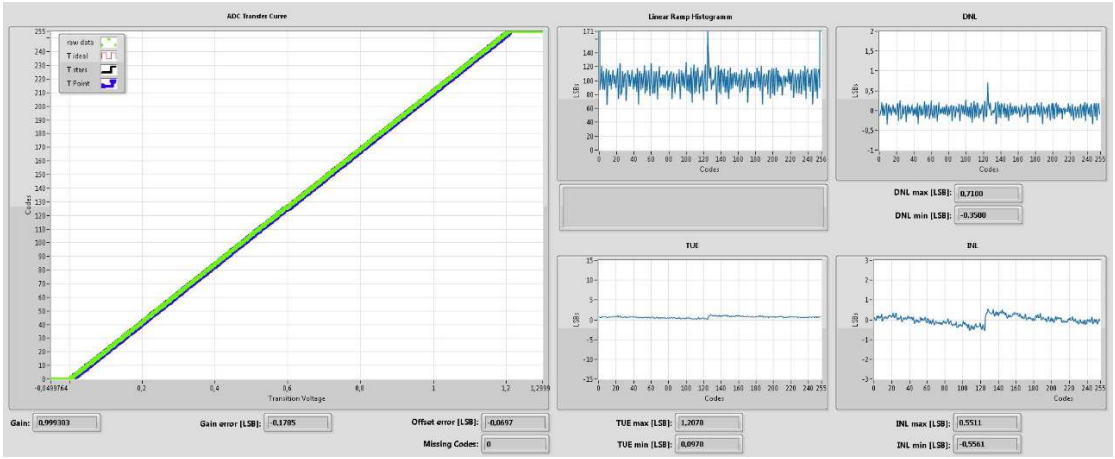


Figure 6-6. Characterization results with externally applied ramp

After performing the capacitance measurement with the proposed BIST, the resulting capacitance ratios were calculated. These ratios were then used to reconstruct the characteristic curve of the ADC with the help of a MATLAB model of the ADC. The model of the ADC was used subsequently to replicate all the measurements done for characterization. Figure 6-7 shows the calculated characteristic curve of the ADC using the measured capacitance ratios, and the resulting INL and DNL values.

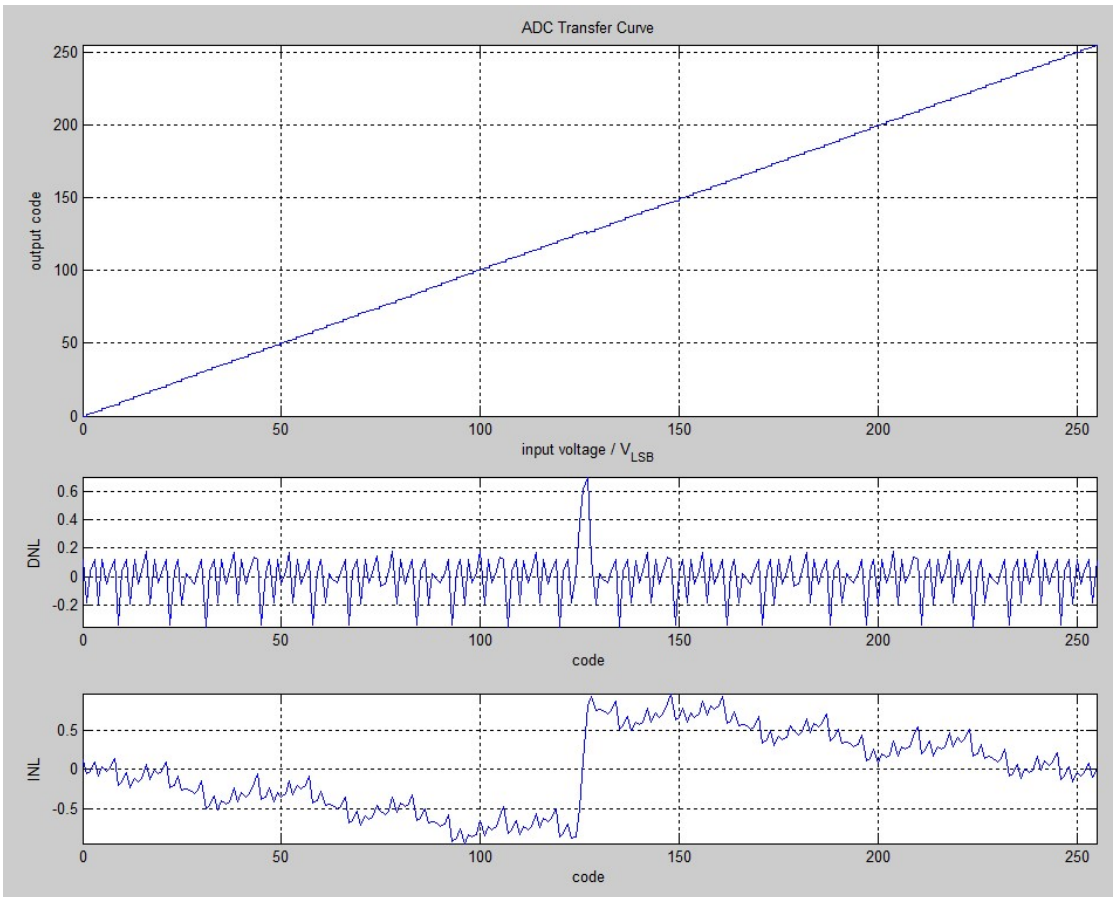


Figure 6-7. Results of ADC model reconstruction of the characteristic curve, and the resulting calculated values of INL and DNL

One can see an almost perfect fit to the lab measurements, allowing us to state that the capacitance measurement is capable of replacing the measurements done with the traditional ramp/histogram method used for characterization. Please note that the upward peak in the middle region of the DNL curve, and the associated jump in the INL curve are faithfully reproduced by the model using the values of the capacitance measurement. The peak stems from a chip layout issue that resulted in the MSB capacitance not having the right value, and was detected by the measurement instruments in the lab as well as with our capacitance measurement method.
It is worth repeating that the results of Figure 6-7 were obtained without resorting to laboratory grade equipment, and without applying a high linearity ramp to the inputs of the ADC. They have been obtained by triggering the measurements within the chip and by reading out the binary values of the result registers. This means that with this method the measurements, when done on this ADC "in the field", can be as precise as those done in the lab, thus allowing a performance check of the ADC during life time.

6.2.2.2 Measurement procedure

During the experiments with the capacitance measurement method, it was noted that performing several measurements and averaging the result did not result in the expected accuracy increase (as seen while performing the capacitance measurements on ADCs with higher resolutions, see [61]). Analysis of the circuits and of the results led to the insight that with such a low resolution, the noise of the ADC itself is not high enough to disturb the voltages by an amount that is higher than the voltage difference between two consecutive discharging steps. This means that the dithering that was happening in the case of 10-bit (or higher) ADCs will not help increasing the accuracy of the capacitance measurement for this eight bit ADC. It is possible to obtain a higher accuracy capacitance measurement if the ADC contains dithering circuitry.

6.2.2.3 Measurement Results of a Faulty 11-bit ADC

An 11-bit ADC, including an exponential discharge measurement circuitry, was designed, implemented in silicon, and characterized. To assess the fault detection capabilities of the capacitance measurement method, chosen capacitances from the capacitance array of the ADC were deliberately disconnected using Focused Ion Beams (FIB). The goal of the experiment was to determine in silicon whether the measurement method would allow detecting these missing capacitances.
As a first experiment parts of the Most Significant Bit (MSB) capacitance were disconnected and the transfer curve of the ADC was recorded. The capacitance measurement were then performed, and as in the case of the eight bit ADC, the values of the capacitance ratios were used in a MATLAB model to reconstruct the characteristic curve of the ADC.
In Figure 6-8 one can see that the transfer curves are almost identical, slight discrepancies originate from the approximation made when calculating the capacitance ratio values, see [61]. It can be stated that the measurement method detects the fault. Were this fault to happen during the life time of the chip, then it would have been unequivocally detected.
As a further experiment, multiple "faults" were emulated by removing capacitance from different parts of the capacitance array, so that not only one bit was affected. Once again the comparison between the characterization measurement and the results of the capacitance measurement show that these faults are detected by the proposed method.

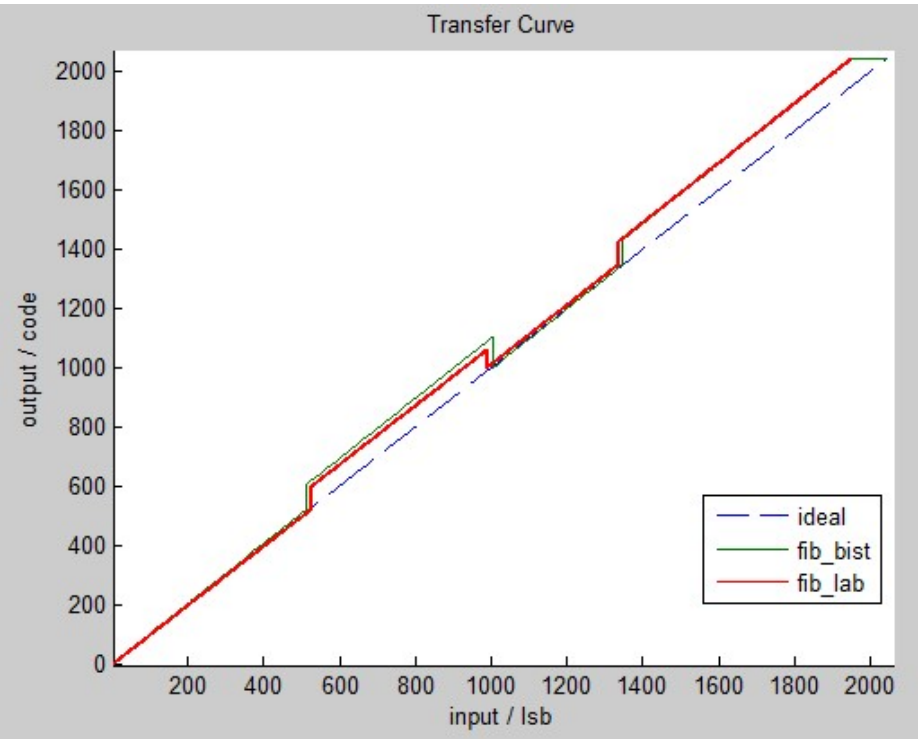


Figure 6-8. 11-bit ADC with a faulty MSB capacitance

Figure 6-9 shows the results. Once again the data from the internal measurement are compared to those acquired in the lab and show a good agreement.

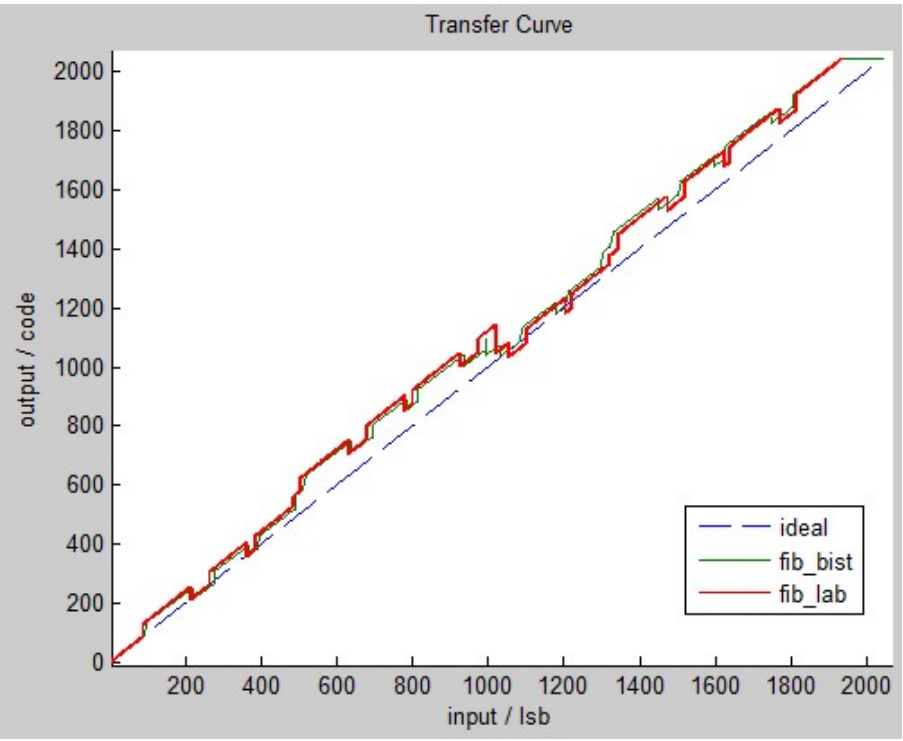


Figure 6-9. 11 bit ADC with multiple faults

6.2.3 Conclusion and Lookout

Chip internal measurement of the ADC capacitances is possible, and is capable of detecting ADC defects. Because the measurement method is incorporated within the chip, and because neither external stimuli nor external measurement instruments are needed, the measurements can be performed during the life time of the chip. This allows the customer of the chip to check the performance of the ADCs and to detect any deviation from the expected performance. This measurement method will now be implemented in all new products that include ADCs. The built in self-test will be used for production test, and it will be available to customers allowing them to reliably check the "health" of the ADCs in their systems.

6.3 Implementation of fault evoking instruments for validating fault management

In deliverable D1.2, the concept "Fault injection for validating error checkers" was presented. This is a hardware architecture for low-area fault evoking instruments for validating and testing the checker infrastructure in the field. The checkers under consideration are intended to detect faults related to permanent and temporary failures during the normal system operation. An additional application of the fault evoking instruments is validation and test of the fault management infrastructure in the field.

6.3.1 Fault Injector (FI)

A fault injector can inject a fault at any of the wires in the circuit under test. The considered fault model in this work is single stuck-at fault. The Fault injector module is parameterizable, based on the data width and address width. It accepts the data as input, along with two input ports, which select whether a stuck-at-1 (sa1) or stuck-at-0 (st0) fault should be injected, or no fault should be injected. There is also an address input, which specifies at which bit location in the input data, the fault should be injected. Figure 6-10 presents the Fault Injector (FI).

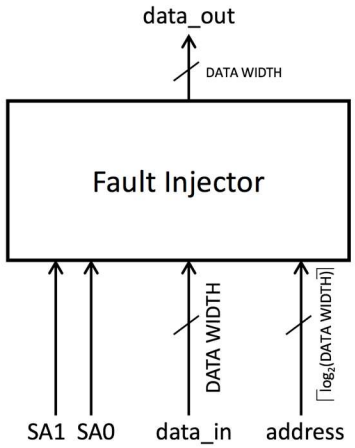


Figure 6-10. An overview of single stuck-at fault injection module (FI)

The locations where faults are injected with respect to each modules inputs/outputs and internal signals are based on the scope of circuit that the checkers can actually cover, as shown by the green circles in Figure 6-11.

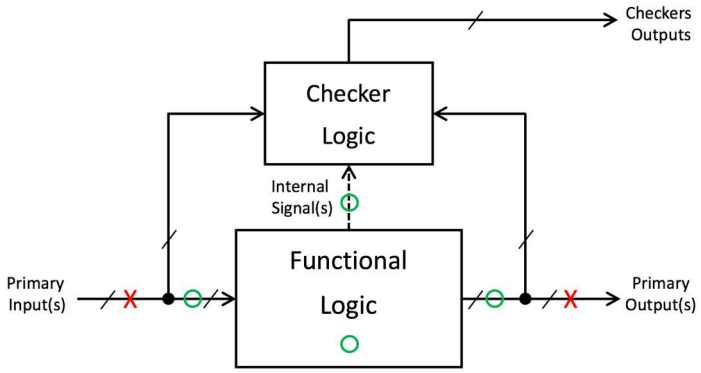


Figure 6-11. Fault injection points for the design to be checked

6.3.2 Reference design: channel of a Network-on-Chip router

In order to validate the concept, a design of a many-core system with fault management capabilities has been designed and implemented in silicon. It is a fault-resilient channel of a network-on-chip router including the fault evoking instruments and concurrent online checkers. Figure 6-12 presents the top-level architecture of the design. In order to save I/O pins, injection of faults by the fault evoking instruments is controlled by a binary code shifted in via a dedicated input pin *Fault_data_in*.

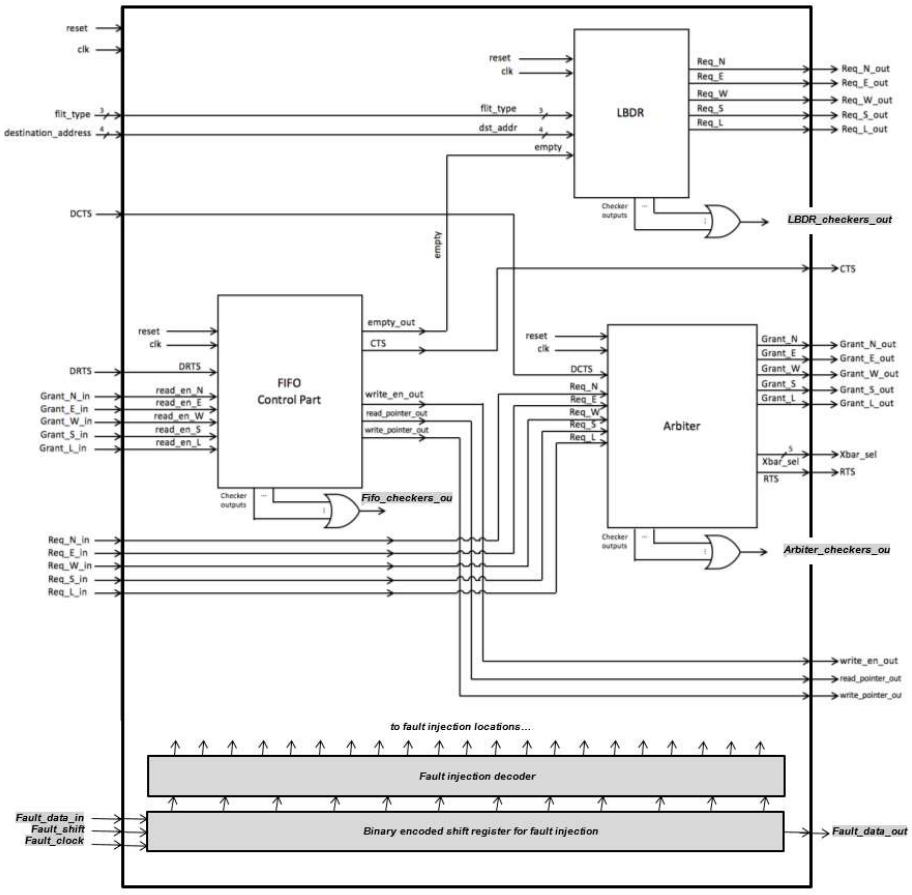


Figure 6-12. Architecture of the fault-resilient channel of a network-on-chip router including the fault evoking instruments and concurrent online checkers

In this project, the system clock and the fault injection clock are separated. In order to shift the fault control data to the circuitry, into each control part module, a shift register with serial input is integrated, which operates on the positive edge of the *fault_clock* signal.

The shift register shifts the data one bit every time the *fault_shift* input signal is set to one. The shift register has the possibility to provide both serial and parallel data on its output. The serial input is connected to the *fault_data_in* input of the top module (the router channel) which is used for sending data input to the shift register of the chain of control part components, starting from FIFO, to LBDR and then to Arbiter. The parallel data output of the shift register with serial input inside each control part module provides a signal, which includes (1) bits for the inputs and outputs of the module where are of interest for injecting single stuck-at fault, (2) bits that show the address for the fault injector (indicating on which bit location to inject the fault) and (3) 2 bits indicating the type of the injected fault (stuck-at 0 or stuck-at-1). The corresponding bits of the signal are connected to the Fault Injector (FI) module inside each control part module.
The Fault Injector inside each control part module, gets the non-faulty values of all the signals related to the locations that fault can be injected on them as input (in our case 28 bits for FIFO's control part signals, 22 bits for LBDR signals and 36 bits for Arbiter signals) and based on the address and type of fault, the fault is injected on one of the respective signals. The values of the faulty signals are then passed to the corresponding signals inside each module.

6.3.3 Implementation of a demonstrator ASIC

Figure 6-13 shows the layout of the demonstrator ASIC. The design was produced as an ASIC by using the AMS C18 A6, 180nm, 6-metal layers technology resulting in the die size of 1.6388mm x 1.63508mm. The first 40 test chips have been manufactured by Europractice in the Fraunhofer Institute in Erlangen, Germany. In order to allow aging measurements, newer, 40 nm TSMC technology version of the chip including a fault resilient many-core system is currently under development.

6.3.4 Conclusions

In BASTION, fault evoking instruments for the purpose of in-field testing and validation of the fault management infrastructure across different layers has been developed. The instruments have been implemented in a demonstrator ASIC with the goal to run fault injection experiments in real-time validation.

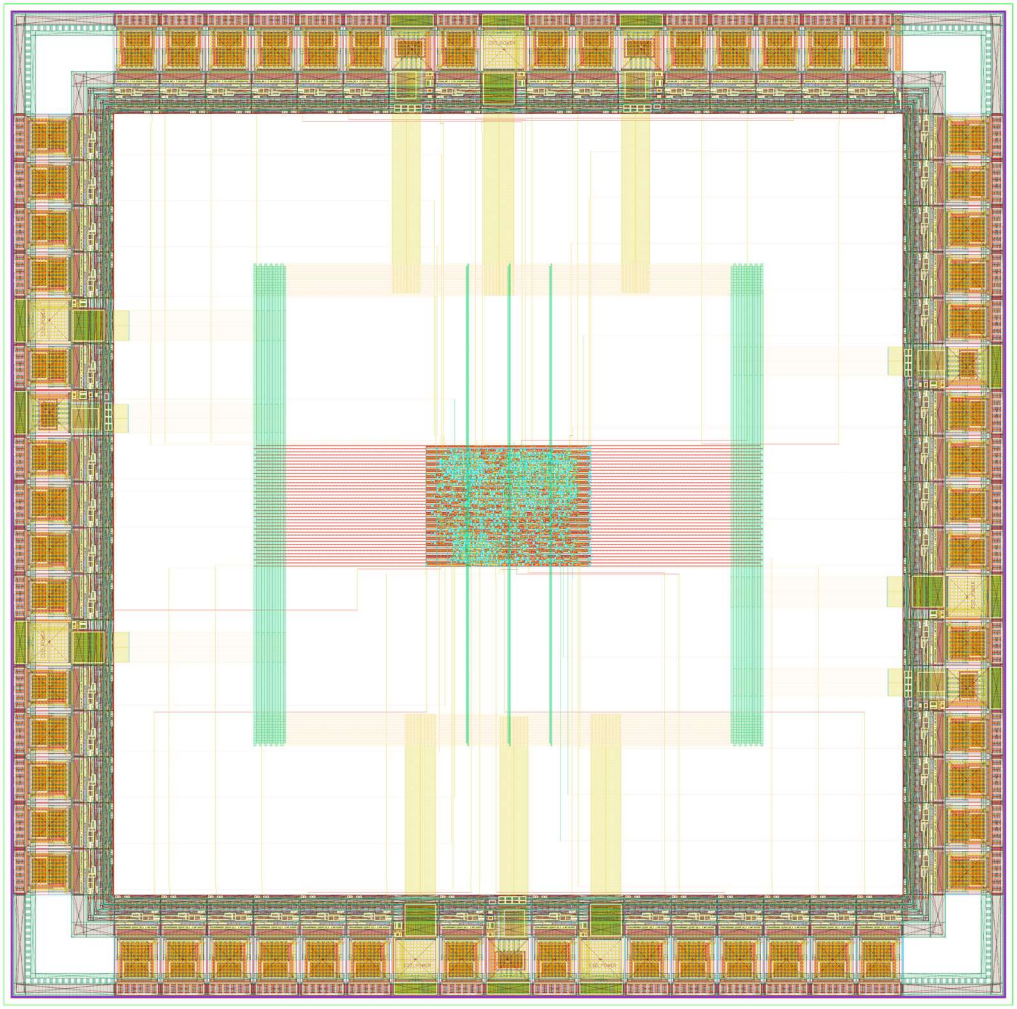


Figure 6-13. Layout of the fault-resilient channel of a network-on-chip router ASIC

7 Summary

In this deliverable we have shown possibilities of evoking intermittent resistive faults at board level as well as chip level. We have set up an experiment to evoke intermittent faults at board level. As a case study, we used a cold solder and measured its resistance changes by voltage and current measurements while the board was exposed to temperature cycling shocks. The result shows the possibility of using temperature cycling for intermittent resistive fault evoking. Second, we introduced a framework to force a local temperature change at chip level. The effectiveness of this framework has been investigated by a case study on an industrial processor structure.

A self-reconfiguration IEEE 1687 has been proposed which speeds up fault localization by introducing a new segment insertion bit. The idea of self-reconfiguring networks through post-layout simulations of one such network has been validated. In comparison to previous similar works the new network enhances localization time at least 2.6 times for a single fault and 6.2 times in case of multiple faults.

With regard to instruments for high-performance test and diagnosis, a new instrument has been designed for high-performance test and diagnosis at board and chip level. Two developed FPGA-based embedded instruments have been designed to detect timing related faults at board level. In order to enhance high-performance test at chip level, optimizations of the BIST method has been presented by partners. The concept of "Fault injection for validating error checkers" in deliverable D1.2 has been followed here with proposing fault evoking instruments for the purpose of in-field testing and validation of the fault management infrastructure across different layers has been developed. In addition, a new test design and ASIC implementing a fault-resilient channel of a network-on-chip router including concurrent online checkers and fault evoking instruments for injecting faults mimicking permanent and temporary effects has been presented.

In terms of KPIs, three contributions have been made by this deliverable:

KPI1: Improvement of efficiency in aging fault detection up to 30%

- The development of the IRF embedded instrument has contributed to the aging fault detection, as these faults could previously not be detected. However, they were also not in the usual defect universe. The monitor coverage is 100%.
- The introduction of on-chip burn in and temperature cycling methods we presented in this deliverable has shown to be enhancing the probability of detection; previously the chance was high *not* to be detected.

KPI2: Introduction of two new classes of NFFs

- The new class of intermittent resistive faults (IRF) has been introduced and proven to exist by measurements, as well as their aging behaviour and root cause.

KPI5: Reducing test-escapes and impact of NFFs, at least a factor of two

- With our developed IRF detection circuit, the test escape can be reduced. How much, depends on the number of detectors employed,

and the parameters of the IRF to be detected (minimum resistance value as well as minimum occurrence time)

- Our evoking developments have shown to be enhancing the detection of all kind of faults in a chip, among them the IRFs. The improvements are somewhat comparable with conventional burn-in tests.
- Instruments for marginal TRF detection presented in this deliverable further contribute to reduction of test escapes. The actual reduction ratio can be estimated based on the new class of TRFs introduced in BASTION deliverable D1.3. Getting real numbers requires however long learning loop that involves data from actual production where the new technologies would have been implemented.

KPI6: Linear complexity with respect to the size of chip of the major monitoring and handling parameters (e.g. error detection latency, faulty resource localization, test initialization, etc.)

- With our techniques, we showed that we have a time overhead for fault localization that is a few hundred clock cycles even in cases with thousands of instruments.

8 References

[1] S. Davidson, “Towards and understanding of no trouble found devices,” in Proc. VLSI Test Symp. (VTS), pp. 147-152, 2005.

[2] Ridgetop Group Inc., SJ BIST, White Paper Presentation, 2013.

[3] N. Kranitis, A. Merentitis, N. Laoutaris, et al., “Optimal periodic testing of intermittent faults in embedded pipelined processor applications,” in the Proceedings of IEEE Conference on Design, Automation and Test in Europe (DATE), pp. 65-70, 2006.

[4] N. Aghaee, Z. Peng, and P. Eles “An integrated temperature-cycling acceleration and test technique for 3D stacked ICs,” in The IEEE Asia and South Pacific Design Automation Conference, pp. 526-531, 2015.

[5] N. Aghaee, Z. Peng, and P. Eles, “Efficient Test Application for Rapid Multi-Temperature Testing,” in the Proceedings of Great Lakes Symposium on VLSI, pp. 3-8, 2015.

[6] A. Calimera, E. Macii, D. Ravotto, E. Sanchez, M. Sonza Reorda, “Generating Power-hungry Test Programs for Power-aware Validation of Pipelined Processors,” in Proceedings of the symposium on Integrated circuits and system design (SBCCI), pp. 61-66, 2010.

[7] M. de Carvalho, P. Bernardi, E. Sanchez, M. Sonza Reorda, “An enhanced Strategy for Functional Stress Pattern Generation for System-on-Chip Reliability Characterization”, in Proceeding of the International Workshop on Microprocessor Test and Verification (MTV), pp. 29-34, 2010.

[8] A.M. Joshi, L. Eeckhout, L.K. John, C. Isen, “Automated Microprocessor Stressmark Generation”, Proceedings of the IEEE International Symposium on High Performance Computer Architecture, pp. 229-239, 2008.

[9] F.A. Aloul, A. Sagahyroon, “Using SAT techniques in dynamic burn-in vector generation,” Proceedings of the 15th IEEE Mediterranean Electrotechnical Conference (MELECON), pp. 1448-1452, 2010.

[10] N. Aghaee, Z. Peng, P. Eles, “Temperature-Gradient-Based Burn-In and Test Scheduling for 3-D Stacked ICs,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015.

[11] A.A. Sagahyroon, “Maximizing Heat Dissipation for Burn-In Testing,” in Proceedings of the IEEE Canadian Conference on Electrical & Computer Engineering, pp. 399-402, 2002.

[12] H. G. Kerkhoff and H. Ebrahimi. Intermittent resistive faults in digital CMOS circuits. In Proc. DDECS, Belgrado, pp. 201-206, 2015.

[13] A. Jutman, S. Devadze and K. Shibin, "Effective Scalable IEEE 1687 Instrumentation Network for Fault Management," Design & Test, vol. 30, no. 5, pp. 26-35, 2013.

[14] K. Shibin, S. Devadze and A. Jutman, "Asynchronous Fault Detection in IEEE P1687 Instrument Network," in IEEE North-Atlantic Test Workshop (NATW), May 2014.

[15] K. Petersen, D. Nikolov, U. Ingelsson, G. Carlsson, F. Ghani Zadegan and E. Larsson, “Fault Injection and Fault Handling: An MPSoC Demonstrator using IEEE P1687,” in IEEE International On-Line Testing Symposium (IOLTS), 2014, pp. 170-175.

[16] F. J. Mesa-Martinez, M. Brown, J. Nayfach-Battilana, and J. Renau. “Measuring power and temperature from real processors,” in EEE International Symposium on Parallel and Distributed Processing, pp. 1-5, 2008.

[17] FJ.T Chang and E. J. McCluskey. “Detecting bridging faults in dynamic CMOS circuits,” in IEEE International Workshop on IDDQ Testing, pp. 106-109, 1997.

[18] K. Skadron, M. R. Stan, W. Huang, and S. Velusamy “Temperature-aware microarchitecture,” in International Symposium on Computer Architecture, pp. 2-13, 2003.

[19] K. Skadron, M. R. Stan, K.Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. “Temperature-aware microarchitecture: Modelling and implementation,” in Transactions on Architecture and Code Optimization (TACO), 1(1), pp. 95-125, 2004.

[20] A. Rohani, H. G. Kerkhoff, and H. Ebrahimi "A software framework to calculate local temperatures in CMOS processors," in International Workshop on Integrated Circuit and System Design Power and Timing Modeling, Optimization and Simulation (PATMOS), pp. 16-22, 2016.

[21] D. Rossi, M. Oman, C. Metra and A. Paccagnella "Measuring Power and Temperature from Real Processors," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23(4), pp. 743-751, 2015.

[22] A. P. Shah and V. Neema "Effect of process, voltage and temperature (PVT) variations In LECTOR-B (leakage reduction technique) at 70 nm technology node," in International Conference on Computer, Communication and Control, pp. 1-6, 2015.

[23] K. Skadron, M. R. Stan, K.Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan "Temperature-aware microarchitecture: Modelling and implementation," in Transactions on Architecture and Code Optimization, pp. 95-125, 2004.

[24] P. Huang and Y. Lee. "Full-chip thermal analysis for the early design stage via generalized integral transforms," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 17(5), pp. 613-626, 2009.

[25] F. T. Wang and C. Chen. "3-d thermal-adi: A linear-time chip level transient thermal simulator," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 21(12), pp. 1434-1445, 2002.

[26] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif "Full chip estimation considering power supply and temperature variations," in Proceedings of the international symposium on Low power electronics and design, pp. 78-83, 2003.

[27] P. Li, L. Pileggi, M. Asheghi, and R. Chandra "Efficient full-chip thermal modelling and analysis," in Proceedings of the IEEE/ACM International conference on Computer-aided design, pp. 5-11, 2004.

[28] J. M. Lasance "Two benchmarks to facilitate the study of compact thermal modeling phenomena," in IEEE Transactions on Components and Packaging Technologies, 24(4), pp. 559-565, 2001.

[29] M. N. Sabry "Compact thermal models for electronic systems," in IEEE Transactions on components and packaging technologies, 26(1), pp. 179-185, 2003.

[30] E. G. T. Bosch "Thermal compact models: An alternative approach," in IEEE Transactions on Components and Packaging Technologies, 26(1), pp. 173-178, 2003.

[31] A. Krum "Thermal management," The CRC handbook of thermal engineering, 2000.

[32] K. Skadron, T. Abdelzaher, and M. R. Stan. Control theoretic techniques and thermal-rc modelling for accurate and localized dynamic thermal management. pp. 17-28, 2002.

[33] N. Rinaldi "Thermal analysis of solid-state devices and circuits: an analytical approach," in Elsevier Journal of Solid-State Electronics, 44(1), pp. 1789-1798, 2000.

[34] Recore systems, December 2012.

[35] J. Alt, P. Bernardi, A. Bosio, R. Cantoro, H.G. Kerkhoff, A. Leininger, W. Molzer, A. Motta, C. Pacha, A. Pagani, A. Rohani, and S. Strasser. "Thermal issues in test: An overview of the significant aspects and industrial practice," in IEEE 34th VLSI Test Symposium (VTS), pp. 1-4, 2016.

[36] I. Pomeranz, "On the Switching Activity and Static Test Compaction of Multicycle Scan-Based Tests", in IEEE Transactions on Computers, 61(8), pp. 1179-1188, 2012.

[37] A. Bar-Cohen, A. Watwe, and K. N. Seetharamu, "Fundamentals of thermal management," in Fundamentals of Microsystems Packaging, McGraw-Hill, 2001.

[38] H. Li, K. I. Jacob, C. P. Wong, “An Improvement of Thermal Conductivity of Underfill Materials for Flip-Chip Packages,” in IEEE Transactions on Advanced Packaging, 26(1), pp. 25-33, 2003

[39] M. Pedram, S. Nazarian, “Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods,” in IEEE Proceedings, pp. 1487-1501, 2006.

[40] P. Bujard and J. P. Ansermet, “Thermally conductive aluminum nitride-filled Epoxy Resin,” in Proceedings of IEEE SEMI-THERM Symposium pp. 126–130. 1989.

[41] R. Cantoro, M. Sonza Reorda, A. Rohani, and H.G. Kerkhoff. “On the maximization of the sustained switching activity in a processor,” in IEEE International On-Line Testing Symposium, (IOLST), pp. 34-35, 2015.

[42] M. Asheghi, M. N. Touzekbaev, K. E. Goodson, “Temperature-Dependent Thermal Conductivity of Single-Crystal Silicon Layers in SOI Substrates,” in IEEE Transaction of the ASME, Vol. 30, pp. 30-37, 1998.

[43] E. Sanchez, M. Schillaci, G. Squillero, “Evolutionary Optimization: the μGP toolkit,” in Springer Science & Business Media, 2011.

[44] miniMIPS. http://opencores.org/project,minimips

[45] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” in IEEE Micro, 25(6), pp. 10-16, 2005.

[46] Sun Microsystems, Inc., “UltraSPARC T2™ Supplement to the UltraSPARC Architecture 2007,” [Online]. Available: http://www.oracle.com/technetwork/systems/opensparc/t2-14-ust2-uasuppl-draft-hp-ext-1537761.html. [Accessed Mar. 2016].

[47] “IEEE Std 1149.1-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture,” IEEE Std 1149.1-2001, 2001.

[48] “IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device,” IEEE Std 1687-2014, 2014.

[49] A. Bouajila, A. Lakhtel, J. Zeppenfeld, W. Stechele and A. Herkersdorf, “A low-overhead monitoring ring interconnect for MPSoC parameter optimization,” in IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), pp. 46-49, 2012.

[50] S. Madduri, R. Vadlamani, W. Burleson and R. Tessier, “A monitor interconnect and support subsystem for multicore processors,” in Proceedings of the Conference on Design, Automation and Test in Europe (DATE), pp. 761-766, 2009.

[51] F. Ghani Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, “Access Time Analysis for IEEE P1687,” in IEEE Transactions on Computers, 61(10), pp. 1459-1472, 2012.

[52] F. Ghani Zadegan, U. Ingelsson, G. Carlsson and E. Larsson, “Design Automation for IEEE P1687,” in Proceedings of the Conference on Design, Automation and Test in Europe (DATE), Grenoble, pp. 1-6, 2011.

[53] I. Aleksejev, A. Jutman, S. Devadze, “On coverage of timing related faults at board level” in Proceeding of European Test Symposium, pp. 1-2, 2016.

[54] K. Zetterberg, “Non-intrusive Board Test Strategy for GreenCity”, in Proceeding of Nordic Test Forum, Sweden, 2009

[55] T. Nirmaier et al. “Efficient High-Speed Interface Verification and Fault Analysis”, in IEEE International Test Conference, pp. 1-9, 2008.

[56] P. Maxwell et al. “Comparing Functional and Structural Tests”, in Proc. of International Test Conference, pp. 400-407, 2000

[57] J. Ferry, J. Scesnak, S. Shaikh, “A strategy for board level in-system programmable built-in assisted test and built-in self-test”, in International Test Conference, pp. 800-807, 2005.

[58] I. Aleksejev, A. Jutman, S. Devadze, “Run-Time Reconfigurable Instruments for Advanced Board-Level Testing" in proceeding of AUTOTESTCON, pp. 1-8, 2016.

[59] I. Aleksejev, S. Devadze, A. Jutman, K. Shibin, “Virtual Reconfigurable Scan-chains on FPGAs for Optimized Board Test” in Latin American Test Symposium, pp. 1-6, 2016

[60] I. Aleksejev, S. Devadze, A. Jutman, K. Shibin, “Optimization of Boundary Scan Tests Using FPGA-Based Efficient Scan Architectures” in Journal of Electronic Testing: Theory and Application, 32 (3), 2016

[61] BASTION, “Test optimization techniques (Deliverable D4.2)”, December 2015.

[62] J.L. McCreary and P.R. Gray, “All-MOS Charge Redistribution Analog- to-Digital Conversion Techniques – Part 1,” in IEEE Journal of Solid State Circuits, 10(6), pp.371-379, 1975.