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 Design & Validation of Embedded Instruments for Detection of Several Aging Faults, based on IEEE1687 (Deliverable D2.1)

Due date of deliverable: December 31, 2015
Ready for submission date: February 2, 2016

Start date of project: January 1, 2014 Duration: Three years

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

Revision 1.10 (02/2/2016)

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 D2.1 described in contract 619871.

1 Table of Revisions

Version	Date	Description and reason	Author	Affected sections
0.1	Nov 6, 2015	Structure created	H.G. Kerkhoff	All
0.2	Nov 20, 2015	Removed verification & optimization IJTAG	H.G. Kerkhoff	Chap. 5
0.3	Nov 20, 2015	Background	H. Ebrahimi	Chap. 2
0.4	Nov 25, 2015	State-of-the-art	H. Ebrahimi	Chap. 3
0.4	Nov 26, 2015	Aging sensor	H. Ebrahimi	Chap. 4
0.5	Nov 30, 2015	Reviewed and adapted	H.G. Kerkhoff	All
0.6	Dec. 3, 2015	Operational phase	M. Sonza – Reorda	10.3
0.7	Dec 10, 2015	Update material ULUND	E. Larsson, H.G. Kerkhoff	8.2.1, 9.2.2, 10.6.1
0.8	Dec 11, 2015	Update, Uni. Of Tallinn	J. Raik	10.4
0.8.1	Dec 16, 2015	Update, IFAG	P. Engelke	9.2.1, 10.4
0.9	Dec 16, 2015	Update, ULUND	E. Larsson	All
----	December 2015	Review of draft	R. Krenz-Baath	All
0.9.1	Jan 11, 2016	Update, Testonica	A. Jutman	9.7
1.0	Jan 13, 2016	Update draft	H.G. Kerkhoff	All
1.1 – 1.9	Jan 31, 2016	Many update drafts & polish	H.G. Kerkhoff	All
1.10	Feb 2, 2016	Preparing for submission	A. Jutman	9.7; title pages

2 Authors, Beneficiary

Hans G. Kerkhoff, University of Twente (UT)
Hassan Ebrahimi, University of Twente (UT)
Alireza Rohani, University of Twente (UT)
Erik Larsson, University of Lund (ULUND)
Matteo Sonza Reorda, Politecnico di Torino (Polito)
Jaan Raik, TU Tallinn (TUT)
Piet Engelke, Infineon (IFAG)
Artur Jutman, Testonica Lab (TL)
Sergei Devadze, Testonica Lab (TL)

3 Executive Summary

This document reports on the design and validation of embedded instruments for detection of several aging faults employing IEEE 1687 as performed in BASTION. The work is part of the Task 2.1, where the University of Twente, University of Lund, Politecnico Torino, Technical University of Tallinn, Infineon and Testonica Labs have collaborated. It links to deliverable D1.1, dealing with an NFF and aging fault study

First, the background and the state-of-the art are discussed in terms of aging, related embedded instruments and the issues resulting from the usage of the IEEE 1687 (IJTAG) standard.

Next, new contributions are described from the consortium in terms of new embedded instruments, their validation and several solutions to the issues with regard to the usage of IEEE 1687, such as synchronization and interrupts.

The deliverable concludes with a summary of the obtained results, and a list of references.

4 List of Abbreviations

ADC	- Analog-to-Digital Converter
AOI	- Automated Optical Inspection
ASIC	- Application-Specific Integrated Circuit
ATE	- Automated Test Equipment
BIST	- Built-In Self-Test
BSDL	- Boundary-Scan Description Language
BST	- Boundary-Scan Test
BTI	- Bias temperature instability
CAD	- Computer Aided Design (also EDA)
CMOS	- Complementary Metal-Oxide Semiconductor
CPU	- Central Processing Unit, also Processor
CUT	- Circuit Under Test
DPM	- Defects Per Million
DPMO	- Defects Per Million Opportunities
DRAM	- Dynamic RAM
DRC	- Design Rule Check
EDA	- Electronic Design Automation (also CAD)
EMS	- Enhanced Manufacturing Services
FIFO	- First In, First Out (data buffer)
FP7	- European Union's 7th Framework Program
FPGA	- Field Programmable Gate Array
HCI	- Hot Carrier Injection
HW	- Hardware
IC	- Integrated Circuit
ICL	- Instrument Connectivity Language
ICT	- In-Circuit Test
IJTAG	- Internal JTAG, a short name for IEEE 1687 standard and infrastructure collectively
IP	- Intellectual Property (hardware module in FPGA or SoC)
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;
LBIST	- Logic BIST
MBIST	- Memory BIST
MISR	- Multiple Input Signature Register
MPS	- (Coverage metrics based on) Material, Placement & Solder
NBTI	- Negative Bias Temperature Instability
NFF	- No Fault Found or No Failure Found (also NTF)
NoC	- Network-on-Chip
NTF	- No Trouble Found (also NFF)
PBTI	- Positive Bias Temperature Instability
PCOLA/SOQ	- (Coverage metrics based on) Presence, Correct, Orientation, Live, Alignment/Short, Open & Quality
PDL	- Procedural Description Language
PMOS	- p-type Metal Oxide Semiconductor
POST	- Power-On Self-Test

PPVS	- (Coverage metrics based on) Presence, Polarity, Value & Solder
PPRG	- Pseudo-Random Pattern Generator
PUT	- Path-Under-Test
PVT	- Process, Voltage, Temperature
RAM	- Random-Access Memory
RTD	- Research and Technological Development
SBST	- Software-Based Self-Test
SIB	- Segment Insertion Bit
SoC	- System on Chip
SVF	- Serial Vector Format
SW	- Software
TDC	- Time to Digital Converter
TDDB	- Time Dependent Dielectric Breakdown
TDR	- Test Data Register
TRC	- Tunable Replica Circuit
UUT	- Unit Under Test
VHDL	- VHSIC Hardware Description Language
Vth	- Threshold voltage

5 Table of Contents

6 Structure of the document

In this deliverable, results are reported about the research performed by BASTION partners on design and validation of embedded instruments for detection of several ageing faults, in combination with the IEEE 1687 standard (Task 2.2).
These instruments are located at the chip level, and reuse of them will be desirable. The correct communication between embedded instruments has also been investigated in detail.

This report is structured as follows:

First, a background is given on the different subjects, being aging and its embedded instruments for measuring and error detection and the issues related in using IEEE 1687. Subsequently, the state-of-the-art at the start of the project is provided in both these two areas.

Then, the main body of the work performed in this deliverable is presented, being the design and validation of Embedded Instruments (EI) for aging measurements using the standard IEEE 1687. First, aging measurements and test generation and detection of embedded instruments are treated. Then, the design and simulation results of embedded instruments are presented.

Another issue, the re-use of chip-level embedded instruments to detect ageing defects is discussed next. Subsequently, the access of embedded instruments via software in the operational phase is elaborated. As last subject, handling synchronization and interrupts are explained as well as interruption challenges.

Finally, the overall conclusions are provided. The deliverable is completed with a list of references.

7 Background

In this section the background information on aging faults is given, as well as error detection embedded instruments, and error handling via the IEEE 1687 standard. First, aging faults will be introduced, and next the issues with regard to the standard IEEE 1687.

7.1 Aging Faults

Due to aggressive scaling of technology in terms of device dimensions in advanced CMOS technologies, the design of resource-efficient reliable systems is becoming more challenging. There are various degradation mechanisms that can worsen the performance of devices, circuits and their associated electronic systems as a result of this aggressive technology scaling.

The major aging faults can be classified into electro-migration (EM), hot carrier injection (HCI), and time-dependent dielectric break-down (TDDB) and bias temperature instability (BTI). Depending on the structure and the operating conditions, transistor aging may result in more than 20% speed degradation [1].

7.1.1 NBTI-based aging faults

The bias temperature instability (BTI) is a degradation mechanism that occurs in MOS devices as a result of interface traps between the gate oxide and silicon substrate at elevated temperatures [2] and therefore degrade the dependability of associated electronic devices. This degradation mechanism results in device threshold voltage (V_{th}) shift and loss of drive current (I_{on}). The BTI effect is more severe for PMOSFETs than NMOSFETs due to the presence of holes in the PMOS inversion layer that are known to interact with the oxide states.

The BTI in PMOSFETs is referred to as negative BTI (NBTI) due to the negative gate-to-source voltage. NBTI has been shown to be the dominant effect in current process technologies. The NBTI is accelerated by elevated temperature and voltage levels and manifests itself as an increase in threshold voltage and a decrease of the drain current and transconductance [3].

In PMOSFETs, the channel holes interact with the passivated hydrogen bonds in the dielectric resulting into generation of traps and interface states [4]. This results into an increase in absolute threshold voltage (Vth) value and the effect increases at high temperatures. The introduction of new dielectric materials like high-k has enabled the BTI effect in NMOSFETs and is referred to as positive bias temperature instability (PBTI) due to the positive gate-to-source voltage.

NBTI degradation occurs all the time the transistor is turned on. However, it has been noticed that BTI degradation starts relaxing very quickly after the removal of the stress. This recovery process is caused by de-trapping of charges during subsequent removal of stress signal after a stress phase [5]. Recovery after NBTI or PBTI stress in MOSFETs and its dependence on gate voltage, temperature and frequency of stress signal has been a hot topic of research in the past decade [6].

Currently, BTI is one of the most serious and important reliability concerns for both digital and analog circuits. At advanced technology nodes this effect is becoming an increasing challenge due to the reduced voltage headroom, high oxide electric fields resulting from non-constant field scaling, high temperatures due to higher power dissipation and the introduction of new dielectric material.

7.1.2 HCI-based aging faults

Hot Carrier Injection (HCI) is the result of defects being accumulated in the interface between the channel and gate and causes a gradual increase in threshold voltage and reduction in mobility. In HCI, electrons accelerated in the electric field of the channel collide with the gate oxide interface. The collision creates electron-hole pairs. Energetic electrons referred to as *hot* get trapped in the gate oxide layer, causing an increase in Vth. HCI degradation mainly affects NMOS transistors. Moreover, it is dependent on the drain current and predominantly occurs during switching [7].

7.1.3 Electro-migration

Electro-Migration (EM) is one of the causes of permanent failures of interconnect due to aggressive interconnect scaling and heavy current densities over a period of time.
It is caused by the movement in metal ions in conductors, and can eventually lead to the creation of open and short circuits [8].

At high current densities (~10^6 A/cm^2), momentum transfer between electrons and metal atoms becomes important. The transfer, which is called the *electron-wind force*, results in a mass transport along the direction of electron movement.
Once the metal atoms are activated by the electron wind, they are subject to the electric fields that drive the current. Since the metal atoms are positively ionized, the electric field moves them against the electron wind once they have been activated. The interplay of these two phenomena determines the direction of net mass transfer. This mass transfer manifests itself in the movement of vacancies and interstitials. The vacancies coalesce into voids or micro-cracks, and interstitials become hillocks. The voids, in turn, decrease the cross-sectional area of the circuit metallization and increase the local resistance and current density at that point in the metallization. Both the increase in local current density and temperature, increase EM effects.

This positive feedback cycle can eventually lead to thermal runaway and catastrophic failure which will eventually degrade the system dependability.

7.1.4 Time-Dependent Dielectric Breakdown (TDDB) aging

Time Dependent Dielectric Breakdown (TDDB) is one of the major reasons of permanent failures in recent VLSI technologies. It results from a breakdown in the gate oxide [9]. It can lead to an increase in leakage current, or even in worst case causes failure in the transistors' ability to switch. Similar to NBTI, TDDB is driven by gate potential and occurs whenever the transistor is on.

The TDDB is a degradation phenomenon of SiO_2, the thin insulating layer between the control "gate" and the conducting "channel" of the transistor. SiO_2 has a very high band gap (approximately 9eV) and has excellent scaling and process integration capabilities, which makes it the key factor in the success of MOS technology.

Although SiO_2 has many extraordinary properties, it is not perfect and suffers degradation caused by stress factors, such as a high oxide field. The exact physical mechanism of TDDB is still an open question.

The general belief is that TDDB of gate insulating material results from the cumulative effect of insulator trapped charge buildup during short-term and long-term high-field stress. High trapped-charge-induced local fields build up within the insulator creates defects in the volume of the oxide film. These defects accumulate with time and eventually reach a critical density, triggering a sudden loss of dielectric properties [9].

These defects also cause gate leakage and excess noise in MOSFETs. A surge of current produces a large localized rise in temperature, leading to permanent structural damage in the SiO2. One usually refers to *soft-breakdown* (SBD), which is reversible, and *hard-breakdown*, which results in permanent damage. Both will create failures in MOSFETs and hence the dependability of associated electronic systems will degrade.

7.2 *Design and Validation of Embedded Instruments for Aging Measurements using the IEEE 1687 Standard*

7.2.1 The IEEE 1687 standard for fault monitoring

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 [10]. A straightforward scheme to detect and handle such errors is to make use of N-modular redundancy, where the system is replicated N times and the correct response is produced through majority voting. The drawbacks are that (1) already at design time the maximal number of errors, given by N must be known, (2) the voter has to be designed to handle errors, and (3) the system has to be replicated N times, which increases the hardware cost to unacceptable levels in many cases.

An alternative is to embed on-chip instruments capable of detecting errors, and connect the instruments to a fault management that makes decisions based on the collected error statuses. The fault management can be implemented in an on-chip or off-chip processor. To avoid that the network connecting the instruments and the fault management become the bottle-neck, the network must be designed such that the time (latency) from when instruments detect errors and the fault management solution gets aware of the errors is small and deterministic. A small time is needed such that the network does not become the bottle-neck and a deterministic time is important to make the system predictable; the time should not vary significantly, for example due to a high number of concurrent errors.

A network for fault monitoring can make use of the functional network or make use of a stand-alone network. The advantage of reusing the functional network is that the existing network is reused. Hence, there is no additional hardware cost. The drawback is that adding traffic to transport error information may impact the performance of the system. It is also difficult at design time to know the amount of traffic information that is to be generated from errors as the traffic depends on how the errors occur. This might reduce predictability. 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 of a standalone network is that it does not impact the performance of the system. A stand- alone network adds hardware cost, if it is added only for fault management. However, most ICs have stand-alone networks, such as IEEE 1149.1 or IEEE 1687, to enable test, diagnosis, configuration, etc., which makes it attractive to *reusing* such network for error handling during operation.

The key feature of IEEE 1687 networks is the reconfigurability, i.e. the possibility to switch segments of the network on and off the JTAG scan-path. One way to add such reconfigurability is by using SIBs. Figure 1 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. A key feature in IEEE 1687 is the programming of the SIBs which is done at the same time as data is shifted through the network. 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.

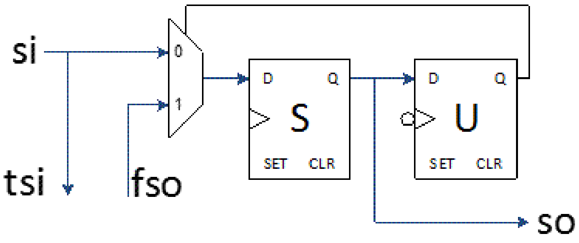


Figure 1. A simplified schematic of a SIB

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, on the other hand, the latched bit is a “1”, the SIB is open and the scan-path includes the segment connected between tsi and fso terminals of the SIB, referred to as the hierarchical port of SIB in this paper. The symbol shown in Figure 2 will be used in the rest of this paper to represent the SIB.

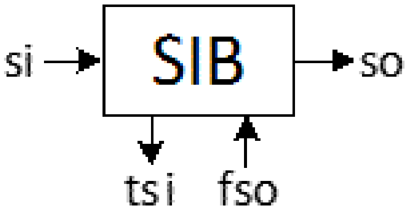


Figure 2. The symbol used for a SIB

Using SIBs makes it possible to design hierarchical IEEE 1687 networks. Figure 3 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 [11], the first hierarchical level of the network (SIB0 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 3, initially, only SIB0 is in the active scan-path. To access e.g. Monitor 1, SIB0 should be programmed to be open, which requires going through Shift and Update states of the JTAG TAP finite state machine (FSM), referred to as capture-shift-update (CSU) cycles. Next, SIB1 should be opened while keeping SIB0 open and SIB2 closed, which requires performing another CSU. At this point, the register for Monitor 1 is placed in the scan-path and can be read from or written to.

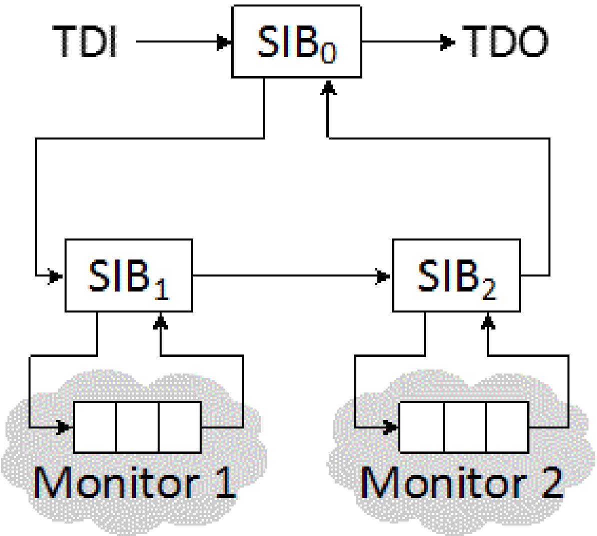


Figure 3. Example of hierarchical IEEE 1687 network with two instruments and three SIBs

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 [12], and that using a hierarchical design can reduce the SIB programming overhead significantly [13]. Other issues with regard to IEEE 1687 are related to timing problems in terms of synchronization and interrupts; they will be discussed later.

8 State-of-the-art

The major mechanisms of aging have not changed recently. As plots from NASA show, new technology nodes sometimes make some aging mechanisms more dominant than the previous node. Some models, like e.g. for NBTI, get more sophisticated, and match better with recent advanced measurements.

With technology scaling, the deviation between predicted path delay after synthesis and actual path delay on silicon increases due to aging and process variation. In digital chips, an increased delay is the major result of aging mechanisms, more specifically resulting from NBTI, currently.

As a second trend, the usage of on-chip measurement (EI) architectures have become more prevalent due to their higher accuracy and lower cost compared to using external expensive measurement devices. Hence, the focus will be on these.

Within BASTION, a strong emphasis has been placed to access the embedded instruments via the emerging IEEE 1687 standard (IJTAG). This includes start, stop, reconfiguration etc., and measurement data transfer. However, the usage of this standard brings also a number of important issues, like triggering, calibration etc. which will also be treated.

8.1 Aging Faults resulting in delays and embedded instruments

One effective approach to measure aging of a digital circuit, resulting from the mechanisms discussed in the previous part (more specifically NBTI), is by measuring its *delay*. There are several approaches toward measuring path delay by using dedicated measurement circuits such as ring oscillator (RO), time-to-digital converter (TDC), and tunable replica circuits (TRC). They will be briefly discussed subsequently.

The ring-oscillator-based architecture can be used more efficiently for path delay measurement by making the path-under-test (PUT) a part of oscillator. The ring oscillator consists of an odd number of inverters, connected into a ring. The ring oscillator frequency can be monitored continuously and any degradation in frequency with time can be attributed to device aging affects.

In [14], the author proposed an on-die aging monitor with the combination of symmetric and asymmetric ring oscillator which can measure both BTI and HCI. Their measurement results of a test chip fabricated in 16nm Fin-FET shows that HCI is almost half the percentage of all aging factors in an asymmetric ring oscillator and PBTI degradation is around 1/10 of the NBTI contribution.

Ring-oscillator based test structures that can separately measure the NBTI and PBTI degradation effects in digital circuits have been presented in [15] for high-k metal gate devices. To decouple the impact of BTI and HCI effects in RO degradation, a novel RO test structure has been proposed in [16].

Although, ROs can simply be used as aging and frequency degradation measurement, however, it has some disadvantage. It is susceptible to self-heating, because of dynamic power dissipation. In addition, it can only distinguish the average delay, not the difference in propagation delay between falling and rising transitions.

A Time-to-Digital Converter (TDC) is a popular on-chip delay measurement circuit. It measures the time interval between two transitional signals. It is typically constructed from registers and delay elements and, like ring oscillators, can be used to measure the effect of process, temperature and voltage variation and aging degradation.

However, as the TDC is an on-chip circuit, it suffers from both process variation and aging effects. Hence, periodical calibration is required to guarantee the accuracy of the delay measurement. A low-area calibration technique for TDCs has been presented in [17]. A TDC circuit allows to accurately measuring its own delay, or the delay of a CUT. The primary limitation is the difficulty in the generating the start and stop measurement signals which must have identical skew.

A Tunable Replica Circuit (TRC) is another method for detecting (aging) errors due to voltage and delay degradation. Process variations and aging affects replicas in a similar way to the critical path. However, the random component of process variation will result in differences between the TRC and the actual critical path [18].

The major drawbacks of TRCs are the complex tuning process and the capability of detecting only the worst-case delay path.

Another approach towards measuring the aging effect is slack monitors accompanied by a guard-band. Most of the aging faults increase delay of transistors and connections. In a synchronous system, a delay increment can result in timing failure. Timing failure occurs when the delayed data does not meet a flip-flop's setup requirement and has a late transition near to the clock edge. Therefore, online delay (or slack) monitoring in an integrated circuit is a suitable metric for measuring the aging of synchronous circuits [18, 19]. The timing slack or guard-band is defined as the delay, between the data arrival time and active edge of the clock, minus the flip-flop setup time. The guard-band assignment is done at the pre-silicon design phase based on the target clock frequency.

In [20]20], the authors have proposed a timing-slack monitoring methodology of inserting monitors at both path ending nets and path intermediate nets. Their experimental results on commercial processors show that with 5% additional timing margin, their methodology can reduce the number of monitors down to 6-8% compared to the number of monitors inserted at the path ending pins.

Recently, a timing slack monitor as well as a sensor insertion flow has been presented in [21]. The output of their slack monitor can be recovered using the IJTAG standard.

8.2 Design & Validation of Embedded Instruments using IEEE 1687 standard

Within the BASTION project, a strong emphasis has been placed to access the embedded instruments via the emerging IEEE 1687 standard (IJTAG). This includes start, stop, reconfiguration etc., and measurement data transfer. However, the usage of this standard brings also a number of important issues, like triggering, calibration, fault handling etc. which will also be treated.

8.2.1 Embedded Instruments Based on Existing Built-in Self-test

Modern SoCs contain various types of components like digital logic, different types of memories as well as various sorts of mixed-signal components. Traditionally these components are only tested during the production test using dedicated test equipment. Driven by customer demands and functional safety requirements, e.g. ISO26262 for the automotive domain, there is more and more the request to perform the device-level testing not only during production, but also in the application. This in-system test cannot use the typical ATE of the production test, but has to rely on the very limited test resources available in the functional system. Therefore, BIST solutions are integrated into the IC allowing the test of chip-level components without the need of additional tester resources.

For embedded memories dedicated BIST hardware [22] is often already implemented for the production test and can be easily reused in the system. For digital logic built-in self-test approaches are known since years.

A typical Logic BIST (LBIST) solution is often based on the STUMPS [23] approach. It uses a pseudo-random pattern generator (PRPG) providing on-chip test data to operate the scan chains of the production test while the test responses are evaluated using a multiple-input signature register (MISR). The LBIST is a feature mainly dedicated to the in-system test, as the reachable test coverage is usually not sufficient to achieve the high quality goals of the production test.

For the emerging amount of *mixed-signal* components the traditional test is performed using functional test sequences and there is usually no structural test solution available. Built-in self-test solutions are also requested and desired for the mixed signal components. Contrary to digital logic however no standard solution exists that covers a range of different mixed-signal components.

8.2.2 The standard IEEE 1687 for fault monitoring

There have been a number of works addressing the network for transporting monitoring data (for transient faults, timing errors, power and temperature estimation, etc.) using a dedicated infrastructure [24]–[26]. The work in [24] stands out as it relies on reusing the existing IEEE 1687 Standard (IJTAG) network [27] for the monitoring purposes. An advantage of a making use of an existing standard is the availability of automation tools (e.g., for insertion and test of the network [13]).

Such hierarchical networks have been used for fault monitoring purposes in fault management schemes proposed by Jutman et al. [25] and by Petersen et al. [26]. The fault management schemes in [25] and [26] make use of an IEEE 1687 network connecting fault monitoring instruments to the fault manager, along with an error propagation network. For example, a simplified representation of the hierarchical network used in [25] is shown in Figure 4.

The key to speed up the error detection in the previous works [25, 26] is the added error-propagation network (marked by the dashed line in Figure 4 in the case of [25]), in which error detection information is immediately reported to the 'ErrorFlag' at the highest hierarchical level. The advantage is that by reading the 'ErrorFlag' the fault management gets informed if there is any error in the system without having to traverse the complete network. To guide error localization, both papers [25] and [26] added ErrorFlags at every hierarchical level. The drawback is that the time for error localization increases dramatically.

In the work by Jutman et al. timing analysis is provided, which is lacking in the work by Petersen et al. A common drawback in the works of both is that error localization, finding where the error is, involves a number of CSUs, which increases the fault localization time.

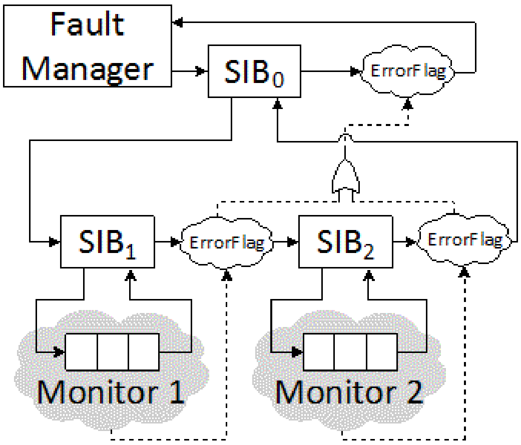


Figure 4. Illustration of the basic ideas in [25] and [26]

Other issues with regard to IEEE 1687 are related to specific timing problems in terms of synchronization and interrupts; they will be discussed later in chapter 9.

9 Design and Validation of Embedded Instruments for Aging Measurements Using IEEE 1687

In this chapter, the research activities of the BASTION consortium are presented regarding the design and validation of several embedded instruments related to aging using the IEEE standard 1687.

9.1 Aging Measurements

To model an aging fault, like e.g. NBTI, the effect of aging on the performance of twenty four Xentium processors [28] under harsh environmental conditions was measured. Figure 5 shows the delay degradation with the operation time. The total time of the (voltage, temperature) stress experiments was more than 2000 hours, but only the first 1000 hours have been plotted.

The mean path-delay values of the processors are drawn by the dashed line. Our measurements confirmed that there is a direct correlation between (NBTI) aging and (delay) performance degradation. Also our NBTI and HCI monitors were simultaneously stressed, evaluated and compared with delay measurements.

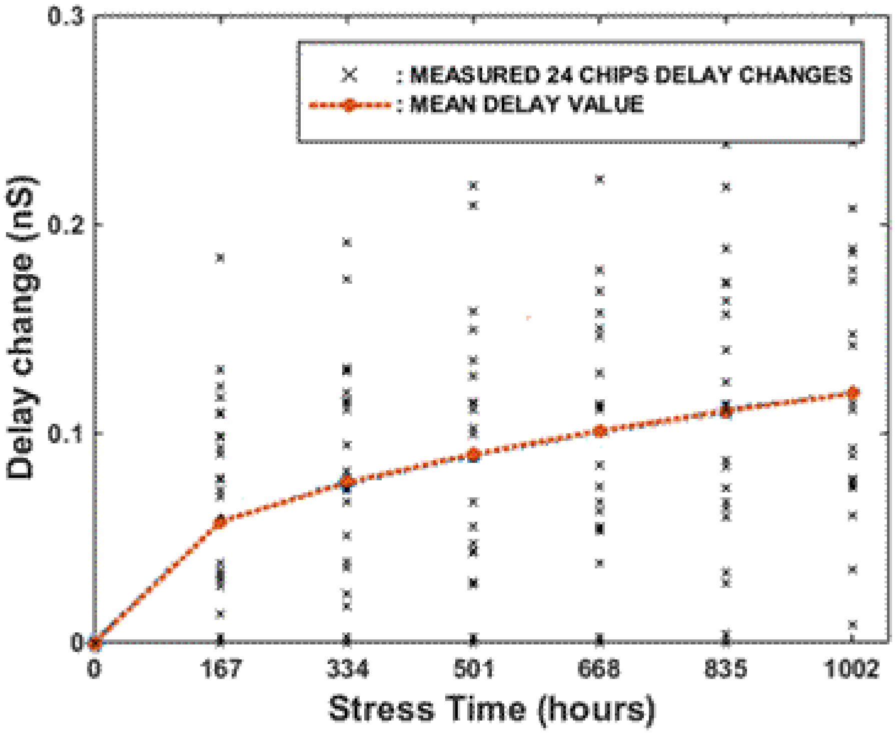


Figure 5. Result of actual delay measurements for 24 Xentium processors

9.2 Test Generation and Detection Embedded Instruments

A positive timing slack indicates that a circuit (e.g. a Xentium processor) is operating safely, with a margin by which the delay can increase before causing failure. Timing slack is an excellent measure for the health of a circuit. Measuring timing slack can allow an early warning of deterioration and trigging pre-emptive actions to avoid failure because of aging. A small or negative quantity of slack is a concern, indicating the circuit is close to, or beyond the point of failure.

It has been shown by experimental results that NBTI is the primary factor leads to timing degradation in current technologies [29]. NBTI has been shown to cause shifts in threshold voltage of up to 50mV over an operating lifespan of 10 years in 65 nm technologies. This translates to more than 20% deterioration in circuit operating speed [30]. To model the delay induced by aging fault (NBTI), the Vth of all PMOSs in the target circuit have been increased by 50mV in our Cadence simulator.

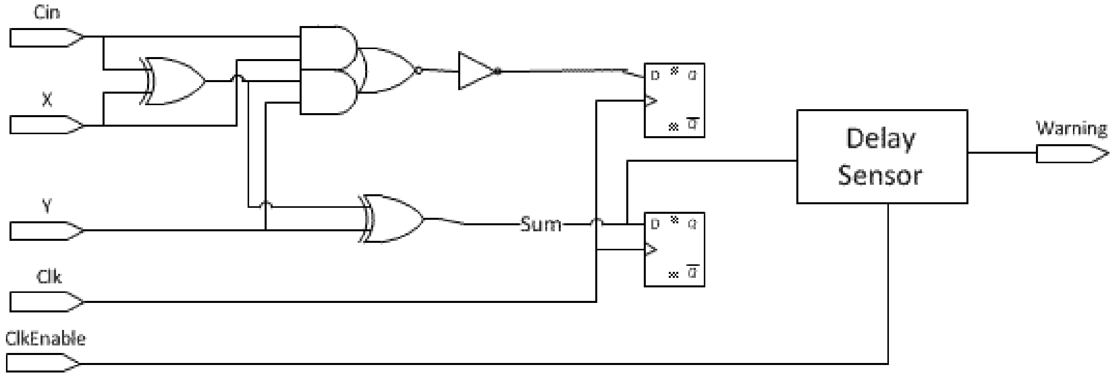


Figure 6. An example of combinational target circuit and delay sensor

To evaluate the proposed delay sensor, a static CMOS circuit in 45nm CMOS technology has been used. The delay sensor monitors output of the circuit before it becomes captured by a flip-flop. The circuit operates at a clock frequency of 2.5GHz. The logic scheme is depicted in Figure 6; its transistor implementation is provided at a lower hierarchical level, but is not shown here for simplicity.

9.3 Design and Simulation Results of Embedded Instruments

An online slack measurement sensor to measure the timing slack (delay-related) at critical nodes in the circuit has been designed to directly measure the effects of delay variation resulting from NBTI aging.

The proposed sensor provides a warning if a specified guard-band (safety margin) has been violated, indicating an impending timing failure. It consists of four D-flipflops (see Figure 7). Each flip-flop receives its clock from the previous output of an inverter, except the first flipflop in which case its clock is connected to the ClockEnable line. Hence, each D-flipflop clock is delayed by the delay of an inverter. All D-inputs are connected to the Data line.

If there is not any late transition in the data line, then all flip-flops capture the same value. In the case the target circuit (see Figure 6) delay is sufficiently increased, the first flipflop will latch the previous value of data before the transition; comparison of the outputs of *all* the flipflops by the NAND gates will then indicate a guard-band violation. The ClockEnable signal determines the guard-band window.

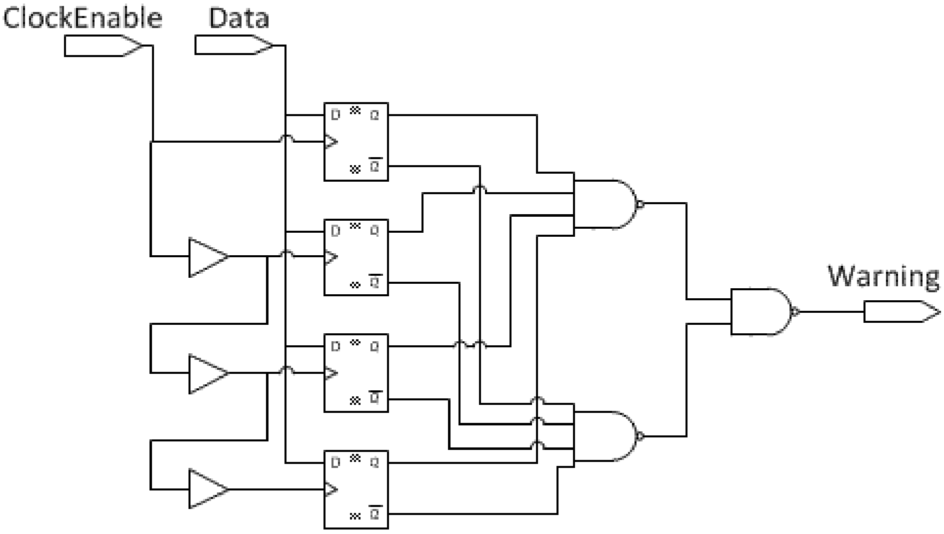


Figure 7. The proposed slack monitor

Figure 8 shows the Cadence Virtuoso simulation environment. At the top, a part of the critical path can be seen of the target circuit (see also Figure 6), and at the bottom the proposed slack/delay monitor can be seen.

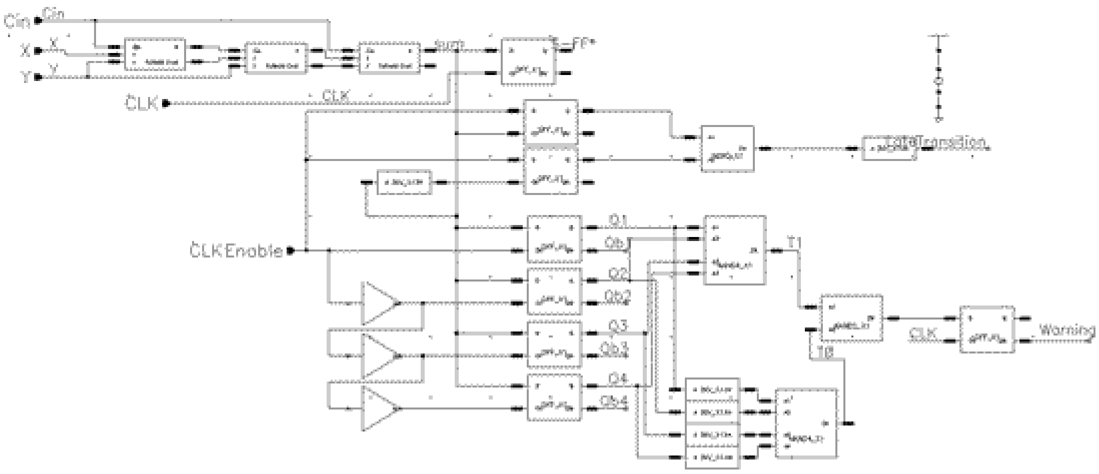


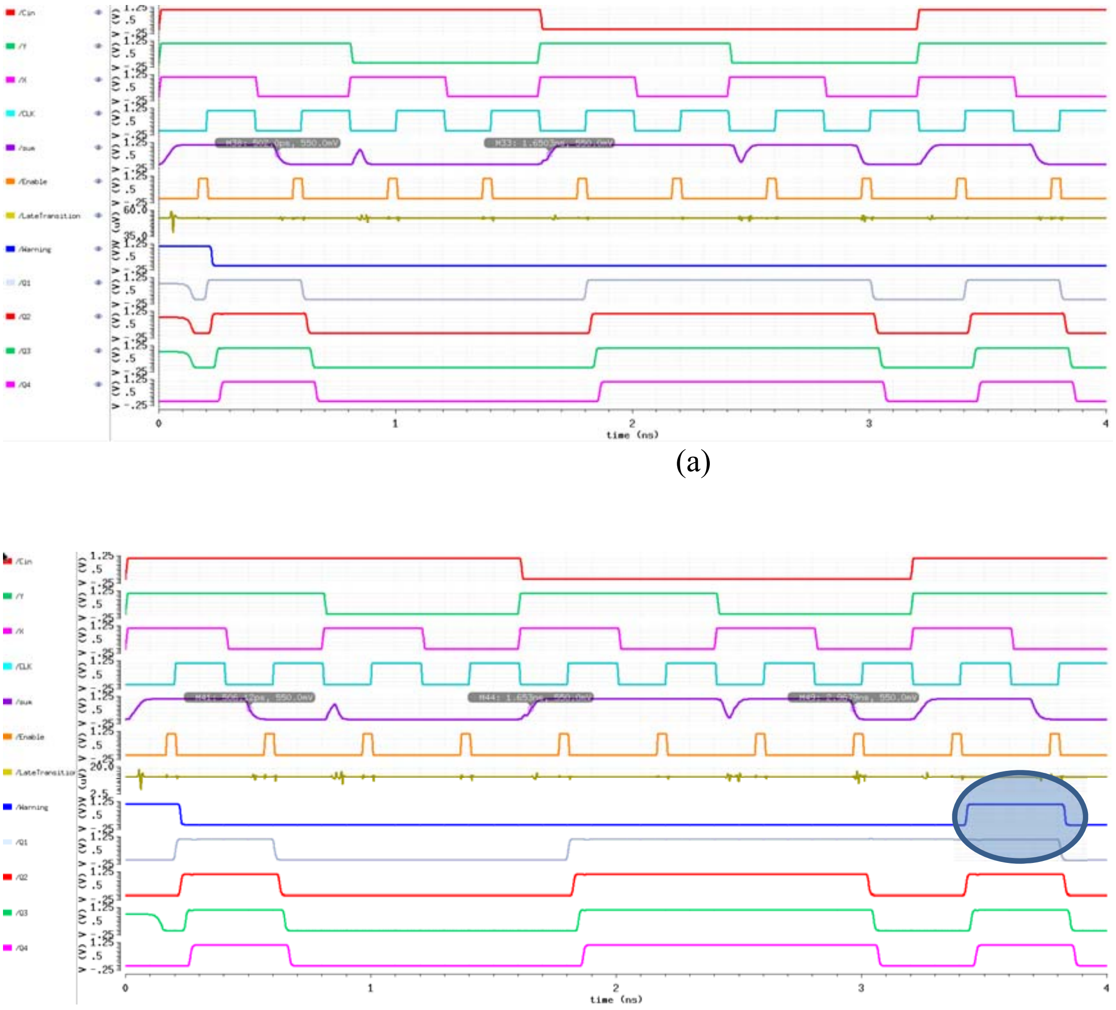
Figure 8. Schematic of the proposed delay monitor

Figure 9 shows the Cadence Virtuoso simulation results of a combinational circuit where its transistors are subjected to *aging* faults. The output (logic) behavior of the circuit as well as the proposed delay aging sensor has been evaluated.

In Figure 9a, the input signals (Cin, X, Y), clock of system (CLK), output signal (Sum), guard-band signal (ClockEnable), output signal of the proposed sensor (warning) and the outputs of the D-flipflops (Q1, Q2, Q3, Q4) are shown from top to down.

Figure 9a shows the simulation results of the target circuit before aging. In this case, there is not any timing (guard-band) violation detected by the monitor, as to be expected.

Figure 9b shows the simulation results of the circuit *after* aging. As it can be seen, the proposed sensor has detected a timing violation. The warning signal indicates that there is a timing violation. After timing violation detection, a warning flag is raised to enable the measurement of the path's slack which is captured in the D-flipflops using the iJTAG standard IEEE 1687 [31].



(a)

(b)

Figure 9. Simulated waveform results of the proposed slack monitor (a) before NBTI aging and (b) after NBTI aging

9.4 Re-Use of Chip-level Embedded Instruments to detect Aging Defects

As ageing effects are considered to become more important, the available self-test components can be reused to check if the device is affected by aging. The goal is to check the overall chip area including all major functional components. The identified aging effects are often affecting the device timing. Therefore, it is essential to target not only static defects, but also to enable the in-system test for dynamic fails.

The (Memory–BIST) MBIST for the memories is usually performed using the functional frequency allowing the detection of both static and dynamic defects. The initial (Logic-BIST) LBIST solutions have been mainly focused on stuck-at defects and have now to be enhanced to perform also an at-speed test. The main problem of the at-speed LBIST is not only the provision of the required clocks with the target frequencies, but also the handling of timing exceptions.

Due to the MISR being used for the on-chip test data compression, flip-flops capturing unreliable data from non-functional paths etc. would prevent the LBIST operation as the signature would be corrupted. Therefore, it is essential to prevent timing exceptions wherever possible.

One approach is the introduction of a hierarchical LBIST architecture. When testing smaller parts of the overall design with separate LBISTs, typically LBIST needs to address fewer clock domains and fewer frequencies. In the best case the design can be partitioned in a way that each LBIST needs to be run only with a single functional clock frequency in the capture phase. As most timing exceptions are usually related to the inter-domain logic between different clock domains, the hierarchical test approach significantly eases the problem.

A hierarchical LBIST solution does not only simplify the handling of timing exceptions, but can also be very beneficial for the localization of defects and the related error handling. In addition, the hierarchical test helps also to deal with test-power issues. When timing exceptions can neither be prevented by suitable test partitioning nor can be fixed by over-constraining the design, the paths not fulfilling the timing need to be broken by design measures during the LBIST execution to prevent unreliable capture.

There is no general BIST solution for the large variety of mixed-signal circuits. However, different approaches exist for some typical components used in many designs. Analog-to-Digital Converters (ADCs) are an essential part of many SoCs. They digitize analog values e.g. coming from various sensors and allow a digital post-processing of the analog measurements.

One approach to monitor aging defects would be to measure a parameter affected by ageing. As an example in [32] the authors suggest measuring NBTI degradation of one of the ADC sub-circuits.

Unfortunately, for production designs the overall functionality has to be guaranteed to the customer. The ADC-BIST circuitry, being on-chip, is not only used for production test purposes, but is also triggered and executed by the system software. This is required in order to ensure functional safety. To allow the reuse of the BIST for monitoring ageing defects, the ADC-BIST has been enhanced with a dedicated digital interface allowing the integration into an IJTAG network.

Despite its analog behavior the main purpose of the ADC-BIST is not to measure the ageing of *one* analog parameter. Instead the BIST is executed to check whether the overall ADC functionality is still within the specified limits. To identify ageing effects before the circuit fails it is possible to run the ADC-BIST with stricter limits than functionally required. Another approach driven from safety requirements is to use one or more spare ADCs in the design. The spare ADC is tested while the others are functionally used. After a certain time the design is reconfigured and the ADCs switch roles. The spare ADC becomes functional and one of the other ADCs is tested. Eventually every ADC is tested in a round robin manner.

In case one of the ADCs fails during BIST execution, the system would still be fully functional. Nevertheless, this would be a clear indication that also the remaining ADCs are likely to fail soon and error handling measures need to be performed.

9.4.1 Experimental Results for IJTAG Controlled ADC-BIST

The IJTAG controlled ADC-BIST solution has been tried for a small test case design. Each ADC-BIST uses the functional interface as shown in Table 1 and Table 2.

Table 1. Inputs of the ADC-BIST instance

Inputs	
Clk	Clock
Rst	Asynchronous reset
Interrupt	Synchronous reset
Start	Start signal for BIST execution
capData	Configuration value for capacitor to be charged, 14 bits
counterRef	Expected reference value for BIST counter value, 16 bits

Table 2. Outputs of the ADC-BIST instance

Outputs	
Done	Done signal indicating end of the BIST operation
Status	BIST result status: 1 test completed successfully, 0 failed
DataOut	BIST counter value for read-out, 16 bit

For the evaluation three ADC-BIST instances have been used. In the assumed scenario always two ADCs are operated in functional mode while the third ADC is tested by the BIST. After one ADC has been completely tested, the circuit is reconfigured and the tested ADC becomes functional allowing the test of the next ADC instance.

For each ADC, the BIST verifies the correct operation of the ADC's capacitor network. This network realizes several capacitance configurations that are tested individually by the BIST engine. The result of each test is quantified by a counter value.

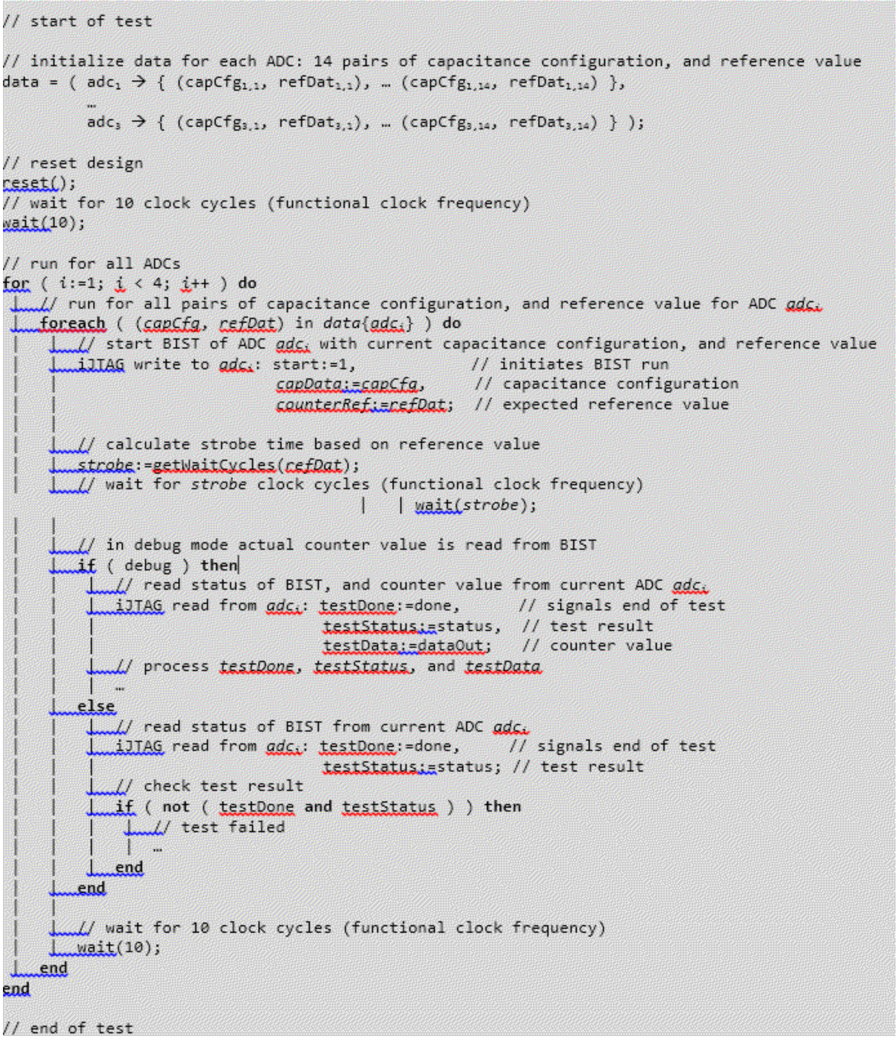
This value is to be compared to a reference (e.g. determined when characterizing the ADC during production test) to check if the respective capacitance configuration is operational. When starting a single test run, both the capacitance configuration to be tested and the expected reference value have to be provided to the BIST controller. Then the test will run for a certain time which is – for an operational capacitance configuration – proportional to the reference counter value. After the test is finished, the BIST's done bit is set to high, and the status bit indicates if the counter value indeed matches the expected reference.

The algorithm in Figure 10 describes the overall flow for testing the three ADCs in a round robin manner. The configurations of the capacitance network along with the respective reference counter values are expected to be available as input to the algorithm.

Additionally a dedicated debug mode can be enabled. In this mode the debug information stored in 'DataOut' is retrieved from the BIST. This information is only relevant for general diagnostics and may be ignored for regular production or in-field test.

The simulation waveform of the relevant BIST signals is depicted in Figure 11 . In particular this figure illustrates, that the three ADCs are indeed tested one after the other.

The default IJTAG network for controlling the ADC-BIST is shown in Figure 12. For each ADC a test register of 32 bits is used. From this test register the BIST is configured and started. The 32 bits controlling the ADC-BIST are: 14 bits BIST configuration, 16 bits expected counter value, 1 bit interrupt, 1 bit BIST start. The same test register is also used to capture the test responses. The test results consist of 18 bits: 1 bit BIST done, 1 bit BIST status, 16 bits actual BIST counter result.



```
// start of test

// initialize data for each ADC: 14 pairs of capacitance configuration, and reference value
data = ( adc_1 → { (capCfg_1,1, refDat_1,1), … (capCfg_1,14, refDat_1,14) },
         …
         adc_3 → { (capCfg_3,1, refDat_3,1), … (capCfg_3,14, refDat_3,14) } );

// reset design
reset();
// wait for 10 clock cycles (functional clock frequency)
wait(10);

// run for all ADCs
for ( i:=1; i < 4; i++ ) do
 └─// run for all pairs of capacitance configuration, and reference value for ADC adc_i
 └─foreach ( (capCfg, refDat) in data{adc_i} ) do
 |   └─// start BIST of ADC adc_i with current capacitance configuration, and reference value
 |   └─iJTAG write to adc_i: start:=1,            // initiates BIST run
 |   |                       capData:=capCfg,     // capacitance configuration
 |   |                       counterRef:=refDat;  // expected reference value
 |   |
 |   └─// calculate strobe time based on reference value
 |   └─strobe:=getWaitCycles(refDat);
 |   └─// wait for strobe clock cycles (functional clock frequency)
                                      |   | wait(strobe);
 |   |
 |   └─// in debug mode actual counter value is read from BIST
 |   └─if ( debug ) then
 |   |   └─// read status of BIST, and counter value from current ADC adc_i
 |   |   └─iJTAG read from adc_i: testDone:=done,      // signals end of test
 |   |   |                         testStatus:=status,  // test result
 |   |   |                         testData:=dataOut;   // counter value
 |   |   └─// process testDone, testStatus, and testData
 |   |   |  …
 |   └─else
 |   |   └─// read status of BIST from current ADC adc_i
 |   |   └─iJTAG read from adc_i: testDone:=done,     // signals end of test
 |   |   |                         testStatus:=status; // test result
 |   |   └─// check test result
 |   |   └─if ( not ( testDone and testStatus ) ) then
 |   |   |   └─// test failed
 |   |   |   |  …
 |   |   └─end
 |   └─end
 |   |
 |   └─// wait for 10 clock cycles (functional clock frequency)
 |   └─wait(10);
 └─end
end

// end of test
```

Figure 10. The flow for testing three ADCs

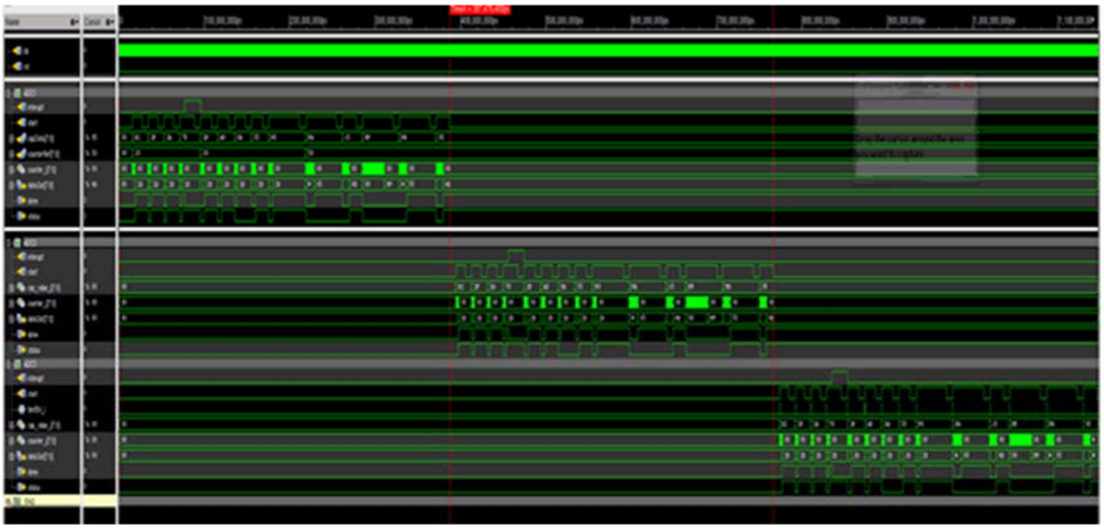


Figure 11. Simulation waveform testing one ADC after the other

To set up the BIST of a single ADC, 36 clock cycles are required in total. This includes 4 cycles to configure the SIBs and 32 cycles to shift the data into the respective TDR. The additional cycles needed for setting up the JTAG TAP are not considered. Similarly 36 cycles are needed if the result of a BIST-run is to be read from a single ADC.

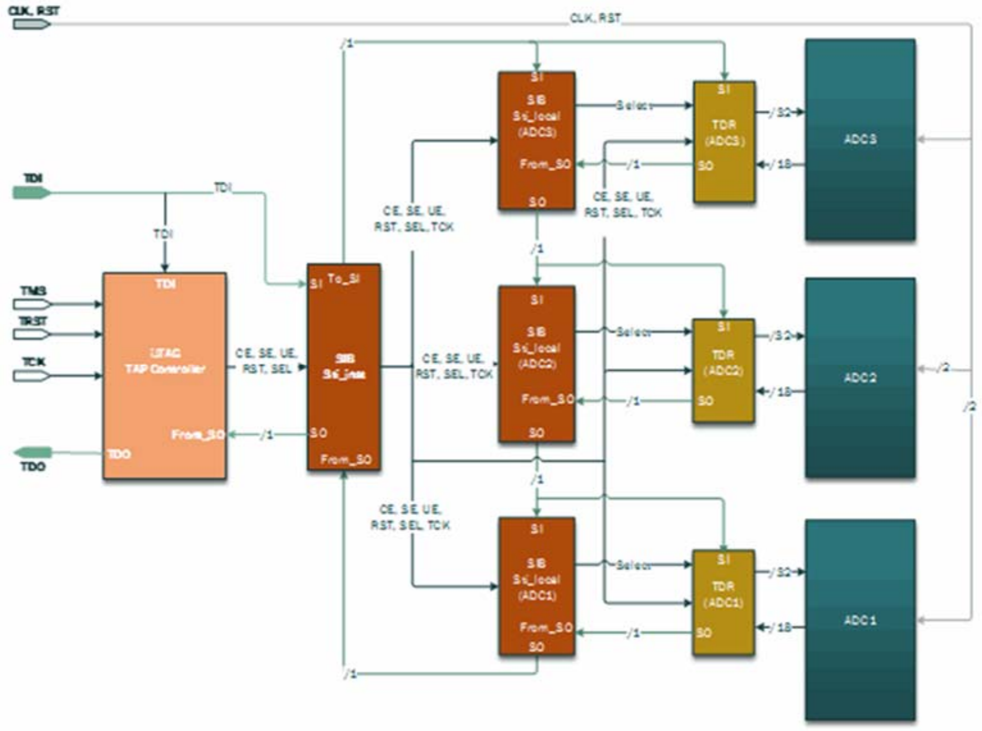


Figure 12. Default IJTAG network controlling 3 ADC-BIST instances

With this network configuration, the ADC-BIST can be fully controlled from off-chip resources. However for production or in-field test the IJTAG configuration can be further optimized. Both applications may only rely on the status of the BIST run, and typically do not require the debug information stored in 'DataOut'. Yet with the default network configuration, 'DataOut' is always retrieved as well, contributing to the 36 shift cycles. As the ADC-BIST runs at the functional clock frequency, the overall test time is dominated by the slow shift cycles of the IJTAG network. Consequently reducing the number of shift cycles may immediately reduce the test execution time.

The IJTAG network in Figure 13 is optimized with respect to production or in-field test. In the optimized network, the single-bit control and status signals are separated from the multi-bit configuration and information signals. For each ADC 'start', 'interrupt', 'done', and 'status' are catered by the same TDR. The signals for 'capData', 'counterRef', and 'DataOut' are joined in a separate TDR.
As a result the number of shift cycles needed to set up a single test run increases slightly to 39. The same number of cycles is needed to retrieve the status of a BIST operation, and the debug information. However, when only reading the status of the BIST run, just 9 shift cycles are required. With this IJTAG network configuration the

ADC-BIST can be controlled efficiently from off-chip resources to enable monitoring of ageing defects.

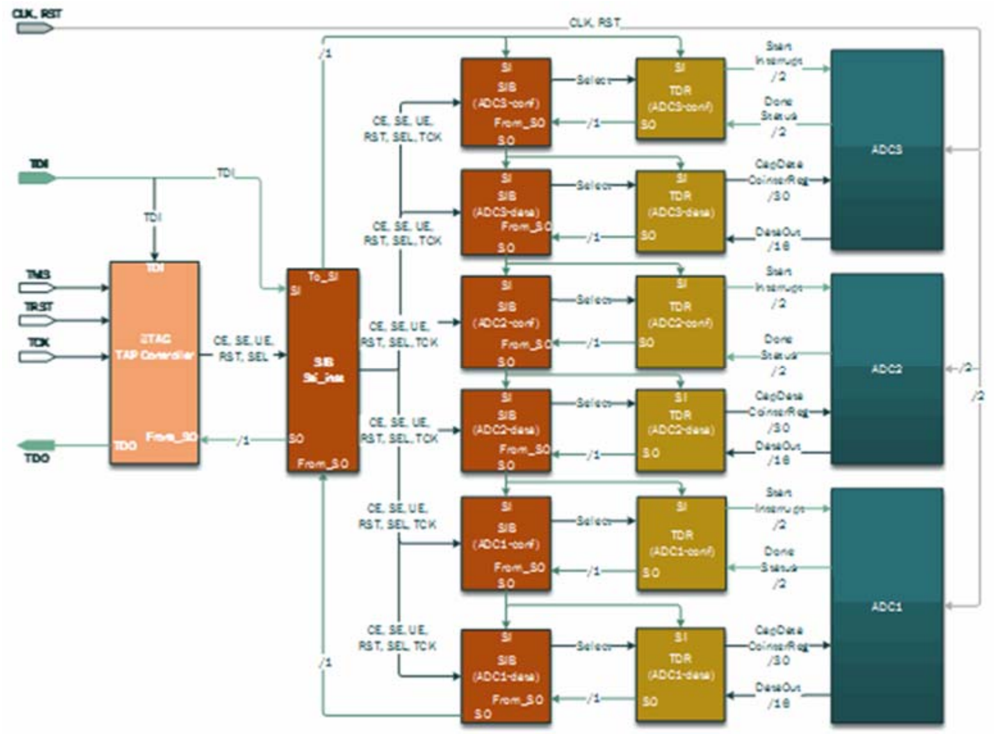


Figure 13. Optimized IJTAG network controlling 3 ADC-BIST instances

9.5 Access of Embedded Instruments via Software in Operational Phase

This section will first deal with issues related to the access options of EIs in the operational phase of a system. In the second part, an example of a digital EI, the error checker, will be introduced as well as a complete flow for implementation and validation.

9.5.1 Access of Embedded Instruments

During the operational phase, test is often managed by the operating system (if any exists in the considered system) or by any high-level supervisor (e.g., acting as a process scheduler)[1].

When considering test, this entity is in charge of different tasks, depending on the specific test solutions which are adopted. Often, they include first of all the access to some instruments existing on board each device, triggering them to provide values (if required), reading, and processing them. Depending on when the test is scheduled, different constraints exist for these activities. BASTION researchers have investigated this and as a result an overview in the following the most common scenarios will be given:

- The test is performed at power-on. This case is common when the system under test is used for short-term missions and the required level of safety is medium-low. For example, this is the solution adopted for several sub-systems on board cars and aircrafts. In this case, the software in charge of performing the test can access and interact with each device in a relatively flexible way, since the application code is still not running. The duration of the test is also less critical in this case with respect to the following ones, although some constraints clearly exist anyway. In some cases, in the test at power-on the test procedure is directly triggered by the reset circuitry.

- The test is performed in the situation that some specific condition is matched (e.g. some error is detected). In this case, the application is temporarily suspended. Hence, the test task has some freedom in this case using the system resources which are not functionally used. However, the status of the system should be kept unchanged as much as possible, thus allowing the normal operation conditions to resume as quickly as possible, if the test does not detect any fault. Similarly, the test duration is an important parameter, although not as crucial as in the following scenario.

- The test is periodically performed, using the *idle* periods of the application. This case is common when the system under test is used for long-term missions or the required level of safety is high. Obviously, the key condition for applying this solution is that the application running on the system has some periodic idle times whose duration is sufficiently long to allow the test task to take the control of the system, run the test, gather some results, and restore the previous status of the system itself. This condition is often verified in safety-critical systems, where often the application code simply corresponds to a single task performing sensor acquisition, basic processing, and actuator driving. Typically, this task is iteratively triggered by a scheduler. In this scenario, the test task can be triggered by the scheduler as soon as the application task ends. By the end of the idle slot, the test task must guarantee that all test related activities are concluded, their (final or intermediate) results

[1] We will not consider here the special case, in which a system has been already labeled as faulty, and a test is performed to more exactly identify the failing component. This scenario is rather different than then ones considered here, since in this case the system is not operational during the test, and the test is often performed in a workshop, possibly resorting to some kind of tester.

gathered, and the status of the system is back to the initial one. Moreover, the test task is often limited by a set of constraints related for example to the amount and location of the memory it can use (which must be compatible with the application constraints).

Access to the instruments can be done in different ways, which are summarized in the following:

- The instruments are accessible as special processor registers. This is for example the solution which is used in some processors to access the performance counters introduced to support silicon debug, software debug and performance evaluation. This mechanism is very flexible and powerful, but requires a strong integration in the processor architecture, and can only be adopted in the case of processors.

- The instruments are accessible as peripheral modules (e.g., using the memory-mapped approach). In this case the software accesses to them following the same scheme used for any other I/O peripheral. In this case, peripherals may also be associated to an interrupt, which is triggered in special conditions (e.g., when unexpected values are read), thus allowing the software to access them only when required. This special case can be effectively used in the frame of the second scenario listed before.

- The instruments are accessible using the IEEE 1149.1 TAP. The standard IEEE 1687 may be used on top of that to make the access more flexible. This solution has the advantage that the access mechanism is completely independent on the functional specifications of the device, but requires the software to be able to access the TAP during the test. If an external test manager exists, this could be a suitable solution to follow.

9.5.2 Error Checkers Validation & Minimization

In BASTION, an automated framework for design, validation and minimisation of embedded error checkers has been developed. The error checkers under consideration are targeting “digital” faults within the system, which may be caused because of wear out due to ageing or by soft errors or electromagnetic compatibility issues. Figure 14. Evaluation and minimization flow for checkerspresents the evaluation and minimization flow for the checkers.

The flow starts with synthesising the checkers from a set of combinational assertions. Thereafter, a pseudo-combinational circuit will be extracted from the circuit of the design under checking. The pseudo-combinational circuit is derived out of the original circuit by breaking the flip-flops and converting them to pseudo primary inputs and pseudo primary outputs. Note, that at this point additional checkers that also describe relations on the pseudo primary inputs/outputs may be added to the checker suite in order to increase the fault coverage.

Subsequently, the checker evaluation environment is created by generating exhaustive test stimuli for the extracted pseudo-combinational circuit. These stimuli are fed through a filtering tool that selects only the stimuli that correspond to functionally valid inputs of the circuit. As a result, the complete valid set of input stimuli that will serve as the environment for checker evaluation is obtained.

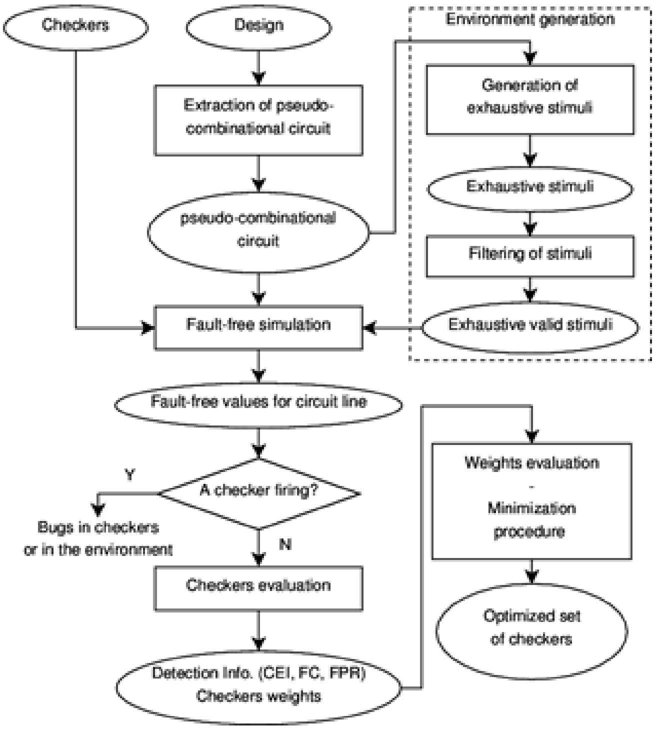


Figure 14. Evaluation and minimization flow for checkers

The obtained environment, pseudo-combinational circuit and synthesised checkers are applied to fault-free simulation. The simulation calculates fault-free values for all the lines within the circuit. Additionally, if any of the checkers fires during fault-free simulation it means a bug in the checker or an incorrect environment.

If none of the checkers is firing in the fault-free mode then checker evaluation will take place. The tool injects faults to all the lines within the circuit one-by-one and this step is repeated for each input vector. As a result, the overall fault detection capabilities for the set of checkers will be calculated. In addition, each individual checker will be weighted by summing up the total number of true detections by the checker.

Finally, the weighting information will be exploited in minimizing the number of checkers, eventually allowing outlining a trade-off between fault coverage and the area overhead due to the introduction of checker logic. The work presented in this subsection has been partly supported by the Horizon 2020 project IMMORTAL. Further details about the framework have been published in [33].

9.6 Handling Synchronization and Interrupts

In the parts of background and state-of-the-art in this deliverable, the status of the emerging standard IEEE 1687 has been presented and explained. In this section several issues will be discussed of *new* research that has been carried out in the area of self-reconfiguring networks related to IEEE 1687 for fault monitors.

9.6.1 Self-Reconfiguring Network

Below, the hardware structure is described of 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 an (embedded) fault monitor, the corresponding error indication flag/register is automatically included in the active scan-path so that in the next shift-out phase, its contents can be immediately 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 ErrorFlag register instead of placing multiple of such registers at every hierarchical level.

9.6.1.1 Hardware Structure

It is assumed that there is 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 the hierarchical IEEE 1687 has the feature of *self-reconfiguration*.

Among all the instruments, it is assumed that there is a set of fault monitoring instruments. At the top level of the hierarchical network, the fault monitoring instruments are connected through a dedicated SIB (see previous section of this deliverable), denoted with SIB_0, while all the other instruments (test, debug, etc.) are connected through another SIB, denoted by 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.

It is further assumed 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 it is captured by the ErrorFlag register at the top level of the hierarchical network.

Additionally, it is assumed 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 *instrument*) 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 IJTAG standard, error masking should be enabled by default at reset to disable self-reconfiguration of the network. In the case 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.

To enable the feature of self-reconfiguration, we propose a modified SIB, which is the core component in a self-reconfiguring network. The proposed modified SIB can be opened asynchronously by means of an additional terminal. The symbol shown in Figure 15. Symbol of the modified SIB will be used to represent the *modified* SIB. In a following section, we will detail the circuitry of the proposed modified SIB.

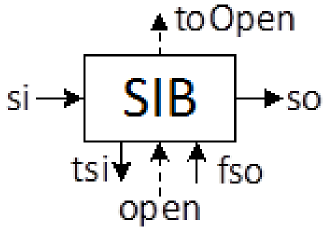


Figure 15. Symbol of the modified SIB

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 or (2) the OR-ed output of the “toOpen” terminals of all modified SIBs attached to it (placed one hierarchical level below). If the “open” terminal is asserted (pulled high), it changes the state of the SIB to open only if the SIB is not already part of an active-scan path. This signal is gated using (an inverted copy of) the select signal to make sure that the state of the SIB does not change (from close to open) when it is part of an active scan-path (e.g., when the Fault Manager is accessing the network). This implies that the fault flag signal should be kept active by the monitor circuitry until acknowledged by the Fault Manager (as discussed above). The “toOpen” terminal propagates the gated open signal to the SIBs in the higher levels of hierarchy. The ErrorFlag register in the top level receives a signal which is obtained by OR-ing the “toOpen” signals from all the modified SIBs directly attached to SIB_0. It should be noted that the suggested modification does not change the behavior of the SIB from the IEEE 1687 standard point of view.

A requirement for the modified SIB (as well as for SIB_0 and SIB_{ins}) is to have its shift (S) flipflop placed after the hierarchical mux port (similar to what is shown in Figure 2). 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 algorithm to recognize the current configuration of the network.

Figure 16 shows an example of a self-reconfiguring network with two monitoring instruments. The 3-bit registers which interface the two modified SIBs to the instruments follow our assumptions mentioned in the beginning of this section.

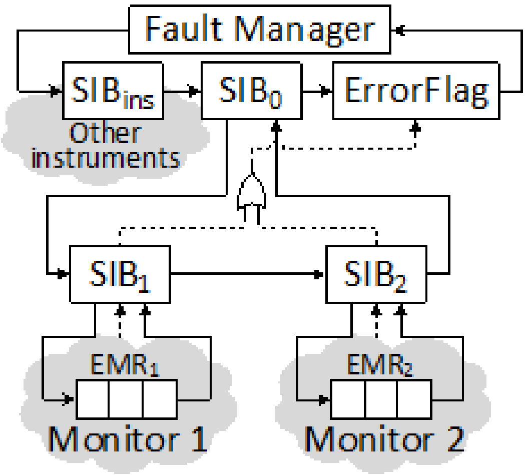


Figure 16. An example of a self-reconfiguring network (dashed line represents the fault propagation network)

The Fault Manager constantly checks the content of the Error-Flag register. The ErrorFlag captures, during Capture phase, the faults through the propagated fault flag (i.e. the ORed "toOpen" signals from SIB_1 and SIB_2). During the Shift phase, the Fault Manager can observe the latest value captured by the ErrorFlag register. A captured value "1" indicates that at least one monitoring instrument has raised an error flag and therefore, the localization phase should be launched. Note that if 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 will be properly configured.

9.6.1.2 Fault Detection and Localization Method

Now, the fault detection and localization method will be explained using the example network shown in Figure 16 and the timelines shown in Figure 17 up to Figure 20.

The following scenarios are being considered:

(1) no error has occurred,
(2) an error occurs when the Fault Manager is not localizing another fault,

(3) a fault occurs while the Fault Manager is localizing another fault, and
(4) two faults occur in a short span of time when the Fault Manager is not localizing another fault.

For these scenarios, it is assumed that whenever a value is shifted out from the network, a zero is shifted in. The effect of this is that all open SIBs are always closed and all error-code registers are set to zero.

For the first (1) 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 by constantly looping 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 detected, the polling takes six test clock cycles (TCK), the interval between t_0 and t_3, as two shifts are required: one for SIB_0 and one for ErrorFlag. This looping is illustrated in **Error! Reference source not found.**7.

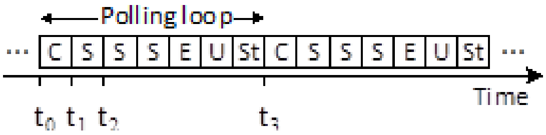


Figure 17. Constant polling to check for faults (C: Capture, S: Shift, E: Exit, U: Update, St: Select)

The value of the fault flag is captured at t_1 into ErrorFlag register and can be observed at t_2. The looping continues as long as the shifted out bit corresponding to the ErrorFlag register is a logic zero.

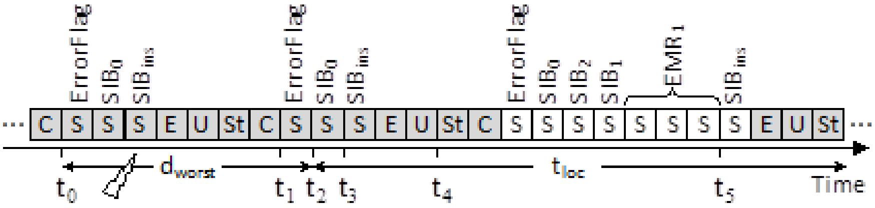


Figure 18. An error (lightning symbol) is detected and localized

For the second (2) scenario (see Figure 18), consider that a fault happens at the interval between t_0 and t_1 and is detected by Monitor 1. The reason this interval is chosen 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 as the *worst-case fault detection time* (if no other error is being localized) and refer to it as d_{worst}. However, it can be seen that it will not take more than seven TCKs to detect a fault. At the moment this value shifts out at t_2 (which belongs to the ErrorFlag) it is a logic "1", and 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 reset the SIBs and error registers for the next round) to localize the fault, as follows:

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 last three bits following are the contents of the 3-bit error code register which interfaces Monitor 1 to the IJTAG network. At this point (t_5), the error is localized and the error information is retrieved. In this work, we also include in the localization time (t_{loc}) the next three TCKs needed to go back to the capture phase such that new faults can be detected. The total worst-case error localization time of a localization round, can then be written as: $t_{worst} = d_{worst} + t_{loc}$.

To also take the time from the fault-propagation network into account, d_{worst} is extended to also include the additional time from a fault-monitoring instrument signal until it reaches the ErrorFlag. This propagation delay is denoted by δ and notice that if the fault monitor signals the error later than t_0 - δ, it is not captured at t_0. Therefore, d_{worst} is given as: $t_2 - t_0 + \delta$ which is equal to $(7/f_{TCK}) + \delta$ 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 19 which is the continuation of the timeline shown in Figure 18 (at t_4 in Figure 18). An error is detected and localized start at t_4 in Figure 19.

A second error happens when the previous one is being localized.

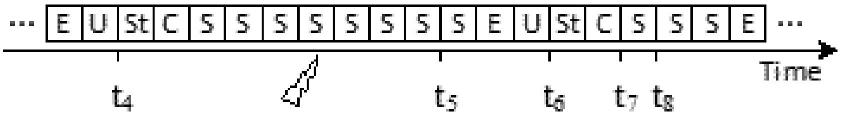


Figure 19. A second error (lightning symbol) happens when the previous one is being localized.

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. Between t_4 and t_6 (when SIB_0 is closed again) SIB_2 is selected (though closed) and, therefore, cannot be opened by a fault flag signal from Monitor 2. That is, any fault happening after t_4, is captured at t_7 and detected at t_8. It should be noted that 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 the 3-bit shift-register connected to SIB_2. The detection and localization of the last error, is performed in the next localization round.

For the last scenario, consider the timeline in Figure 20Figure 1, where both monitors detect faults in the interval between t_0 and t_1.

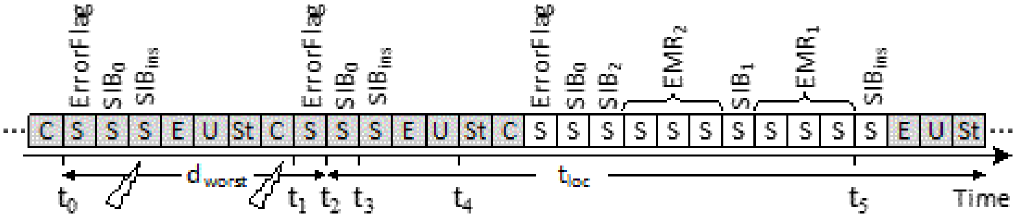


Figure 20. Two faults (lightning symbols) are detected together

In this case, both faults are detected at t_2. In comparison to the scenario for one fault (see Figure 18), the localization procedure takes a longer time as SIB_2 is also open and its associated register is also included in the scan-path.

As a final note in this section, one observes from Figure 18 and Figure 20 that the shaded states are traversed no matter how many faults are being detected and localized. In the rest of this work this constant overhead will be denoted by J_{OH}.

Based on the observations made above, we present in the following section a time analysis on the worst-case fault localization time, for the case of a single error, and the case of multiple concurrent errors. As part of the worst-case error detection and localization time (t_{worst}) is the constant J_{OH}; the presented analysis only focuses on the number of shift states in t_{loc}, which is the variable part of t_{worst}.

9.6.1.3 Time analysis

In this section analyses are presented for the worst-case error localization time in a self-reconfigurable network designed in a balanced hierarchical manner, for two cases: (1) when a single fault occurs, and when (2) multiple faults occur such that they are all addressed in one localization round (see the discussion on Figure 20).

Consider the network shown in Figure 21 in which (except for the last level) each SIB has a number of k SIBs connected to its hierarchical port. Assuming that we have $N = k^h$ instruments in total, the network resembles a balanced k-ary tree whose root is the SIB directly connected to the Fault Manager.

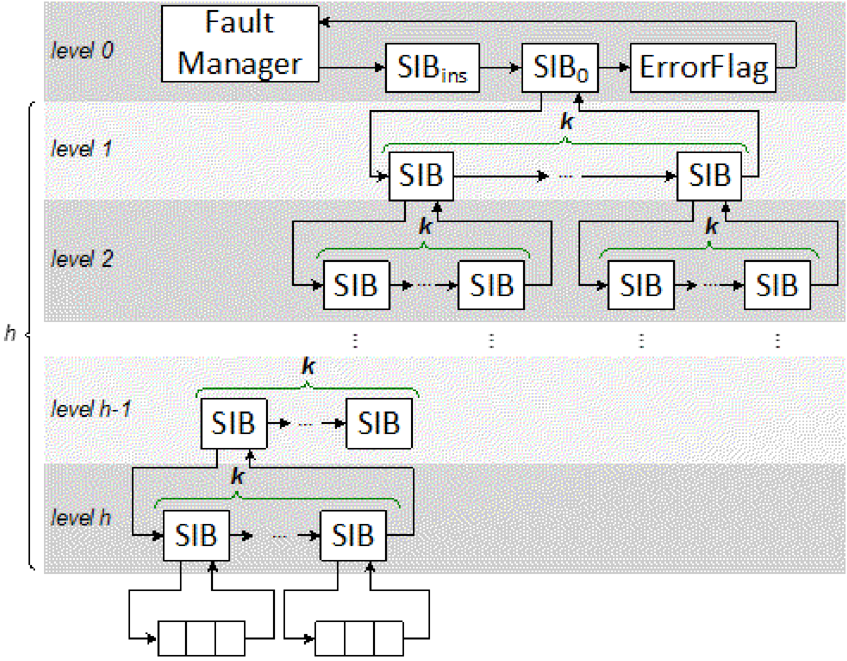


Figure 21. A balanced tree hierarchical network

9.6.1.3.1 Localization time in the case of a single fault

Given the network in Figure 16, and assuming that only one monitor has raised a fault flag causing all SIBs on its hierarchy to change their state to open, there are s = 1+k×h SIBs on the path to each instrument where 1 represents the SIB at level 0. The total shift time t_s is therefore the sum of s and the length of shift-register interfacing the instrument to the network (denoted by L) plus one for the ErrorFlag.

9.6.1.3.2 Localization time in the case of multiple faults

Assume that F faults ($F \leq N$) are to be localized at the same time (same localization round). To calculate the worst-case localization time, we consider monitors detecting these faults to be spread in the network such they cause a maximum possible number of SIBs to be opened (maximizing the length of active scan-path, thus leading to the longest localization time).

9.6.1.4 Network design

Next, an optimized design method is described for the proposed self-reconfigurable networks such that the error-localization time for a single fault is minimized.
As the error-localization time has a constant part (the JTAG FSM overhead) and a variable part (the total shift time), to minimize the localization time, one should reduce the total shift time t_s. Assuming that the number of instruments is given, to find the optimal k which minimizes t_s, we set the first derivative of t_s to zero. The result is that k = e (natural exponent). In practice, we must either choose k = 2 or k = 3 to

minimize the time it takes to shift out the error code from the monitoring instrument which has raised the fault flag.

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. We propose the following construction algorithm. Assuming that there are N instruments, we bundle the instruments into clusters of three instruments each. When N is a multiple of three, one will have c = N/3 clusters. If N is not a multiple of three, one or two instruments will remain.

If the number of remaining instruments is one, then c = $\lfloor N/3 \rfloor - 1$ three-instrument clusters plus two two-instrument clusters.

If, however, the number of remaining instruments is two, then c = $\lfloor N/3 \rfloor$ three-instrument clusters plus a 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.

9.6.1.5 Experimental results

We have compared our proposed self-reconfigurable IEEE 1687 network with the work presented in [25] with respect to worst-case and average-case fault localization time. The results are presented of the comparison for the case if (1) one fault occurs, and (2) multiple faults occur concurrently.

9.6.1.5.1 Single Fault

As for the construction of the network tree for the proposed self-reconfigurable network, four alternatives have been compared: (1) using the network construction method presented in [13] for regular IEEE 1687 networks, (2) a binary tree with pruning, (3) a ternary tree with pruning, and (4) the construction method.

To calculate the number of shift cycles for the self-reconfigurable networks, a regular pre-order tree traversal algorithm is employed. A fixed number of $J_{OH} + 1 = 17$ TCKs is added to the calculated shift time to account for the constant overhead and the ErrorFlag. Moreover, another three TCKs are added to account for the length L of the fault monitoring instrument's shift-register (L = 3). Table 3 shows the results of comparison with the approach proposed in [25]. The numbers show the average-case fault localization time for the self-reconfiguring approach. As the average-case was not reported in [25], we calculated them based on the given timing analysis and information on the structure of the generated networks.

Table 3. The time t_{worst} for a single fault (in TCKs)

Number of instruments		Self-reconfigurable networks			
		tree from [25]	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

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 a 2.8x reduction in localization time is achieved as compared to [25]. 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.

Among the construction methods examined for the self- reconfigurable network, the method described above performs up to 20% better than the method in [13] and results in a better or equal localization time as compared to binary and ternary trees.

9.6.1.5.2 Multiple Faults

The work in [25] did not presented analysis and results on multiple faults, and as our calculations for multiple faults are for balanced k-ary trees, they cannot be used for the network structures presented in [25]. Therefore, to perform the comparison, we (1) used constraint programming (by using the constraints formulation in [25]) to get the optimal network structure for the number of instruments suitable for our analysis (see below), and (2) developed a time analysis for multiple faults for the networks presented in [25] based on their time analysis for a single fault.

As for the number of instruments, we chose k = 3 and calculated the fault-localization time for 27, 81, 243, 729, and 2187 instruments and for 1 to 10 concurrent faults for each network. The fault-localization time for the proposed self-reconfigurable network is calculated by adding J_{OH} TCKs for the constant overhead.

Table 4. The time t_{worst} for multiple faults (in TCKs)

# instruments	Approach	Number of faults									
		1	2	3	4	5	6	7	8	9	10
27	Our	33	42	51	57	63	69	75	81	87	90
	[25]	90	126	162	162	162	162	162	162	162	162
81	Our	36	48	60	69	78	87	96	105	114	120
	[25]	146	202	258	314	370	426	426	426	426	426
243	Our	39	54	69	81	93	105	117	129	141	150
	[25]	334	538	742	946	1150	1150	1150	1150	1150	1150
729	Our	42	60	78	93	108	123	138	153	168	180
	[25]	298	486	674	862	954	1046	1138	1230	1322	1414
2187	Our	45	66	87	105	123	141	159	177	195	210
	[25]	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

The results are presented in Table 4, where the shaded rows present the numbers obtained for the network structure type proposed in [25]. The last row, presents the average improvement ratio achieved over [25], ranges between 6.4 to 8.1 times improvements. 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.

9.6.1.6 Practical Issues

To validate our suggested self-reconfigurable networks with the proposed SIB, we implemented one network and performed post-layout simulations (in 65nm CMOS technology).

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 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 shows a minimum delay of 1.73ns, a maximum delay of 2.62ns and an average delay of 2.1ns.

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 occurring in a network with 27 instruments is calculated as $(30 \times 1 + 2.62) \times 10^{-9}$ which is 302.62ns.

9.6.1.7 Conclusions

A method has been presented where fault localization time is reduced by introducing a *segment insertion bit* (SIB) that enables self- reconfiguration of IEEE 1687 networks. Timing analysis was presented for single and multiple concurrent faults and a way has been proposed to construct the suggested self-reconfigurable network. The proposed self-reconfigurable network was validated through post-layout simulations of one such network. The proposed scheme was compared with previous similar work and observed at least a 2.8 times reduction in localization time for a single fault and a 6.4 times reduction in case of multiple faults.

9.7 Instrument synchronization and interruption challenges

As previously discussed, detection of faults, stress conditions and level of degradation of the system sometimes requires synchronization between monitors or instruments, e.g. in order to relate actual measurement results to operating conditions. Some BIST schemes may also require synchronous operation, e.g. a simultaneous or cascaded execution of Pseudo Random Pattern Generator (PRPG) sequences by multiple generators.

Interruption and emergency signaling may be needed when immediate reaction on error or stress conditions is necessary. The IEEE 1687 standardization committee did not include such scenarios into the standard.

In the context of BASTION, a systematic approach to instrument synchronization and interrupting is desirable; the consortium would like to motivate IP core vendors to use the same scheme enabling flawless integration of cores and reliability solutions into a solid system. Below, the research carried out in this area in BASTION is elaborated in detail.

9.7.1 Interrupts from instruments to OS

The actual error detection is taking place in-situ by embedded instruments/monitors. In case of a critical situation, the error signal from the monitor should be immediately delivered to the Operating System (OS) so that the latter could reschedule the failed task as soon as possible to another available resource. In BASTION deliverable D2.3 we have described a flag-based error reporting system where each block or sub-module is provided with a dedicated set of status flags indicating the current fault detection status of the respective resource. All flags collectively form a hierarchical error indication and propagation asynchronous signaling network tied with IEEE 1687 (Figure 22). The status flag register at each module consists of two status bits and one mask bit. Status bit F (Fault) indicates the detection of fault in the module or in a sub-module. The second status bit C (Correction) shows that the detected fault was automatically corrected. Mask bit X is used to disable fault detection for the underlying module to prevent "false alarms" e.g. if the respective resource is out of use.

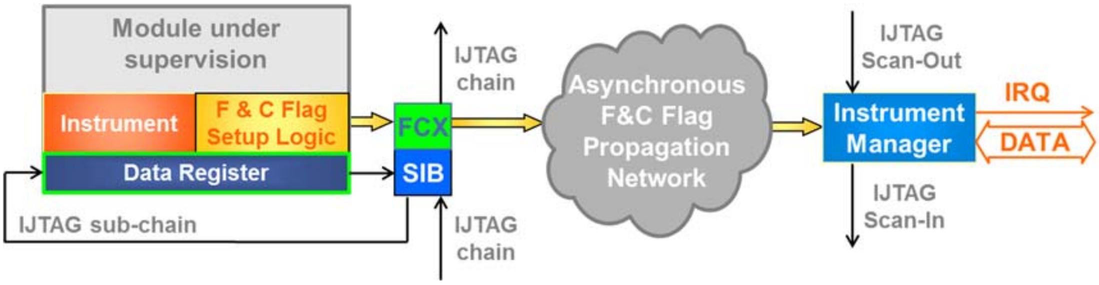


Figure 22. Instrument to OS interrupt via asynchronous emergency signaling network

The asynchronous signals are hierarchically aggregated using logic gates to produce one signal that is, in case of uncorrected error, converted into a system interrupt.

9.7.2 Internal interrupts within the module

Failures in critical resources may result in lost data or lost stability of the system. In practice, such resources are often protected by fault-tolerance mechanisms, like modular or temporal redundancy and hardening techniques. In context of BASTION, fault detection is expected deeper in the unit and earlier in time thanks to embedded instrumentation. If the Fault flag in a module is set by the instrument and the Correction flag is low (fault detected and not corrected) it makes sense to isolate the effect of faulty behavior in a critical resource by blocking further task execution by issuing an internal interrupt inside the unit before a potential damage accumulates. The combination of F and C flags could be used e.g. to block the system clock in that module (see Figure 23). We also foresee that the software or the user may want to enable or disable internal interruption for particular modules. Therefore, a JTAG-configurable Clock Control (CC) bit has to be set to enable such interrupts.

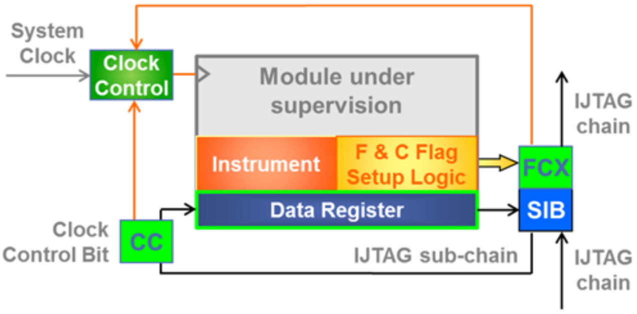


Figure 23. Early blocking faulty task execution by clock gating interrupt

Internal interrupts could be used in double-modular redundancy schemes when the faulty module could be reset while the other one is serving the system.

9.7.3 Calibrated synchronization of instruments

In the IEEE 1687 standard, all actions are synchronized with the test clock and the global Update signal that triggers all instruments. In a large SoC the balancing of Update signal propagation paths can be costly. Therefore, the test clock is usually much slower than the system clock allowing the Update signal to arrive to some parts of the system with a delay without impact to the IJTAG stability.

However, if one needs to run several high-speed instruments simultaneously they need to be synchronized exactly using the fast system clock. A calibration mechanism depicted in Figure 24 is proposed to calibrate delay offsets for Update signal in different instruments so that when needed, all test actions could be aligned to the system clock and start simultaneously as soon as the Update has reached each instrument.

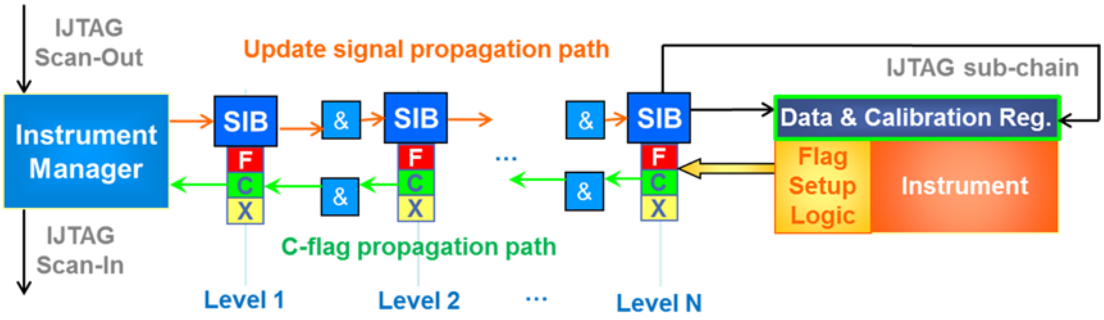


Figure 24. The Update signal delay calibration mechanism

The proposed calibration procedure is based on the fact that the global Update signal propagates asynchronously as a rising front through the whole IJTAG network from the entry point (IM) to each particular instrument. At each level (SIB) in hierarchy, Update passes through the corresponding local enabling logic (AND gate). One can notice that the asynchronous propagation path of the Update signal is nearly identical to the one of the C flag but heading in the opposite direction. A rising transition could be also sent from the instrument through the network to the IM. This fact enables a very efficient calibration procedure consisting of the following steps:

1. IM instructs the instrument to start the calibration procedure;
2. Flag setup logic at the instrument resets the C flag from default state of 1 to 0;
3. IM receives the falling front on C and observes the “illegal” F&C state of 00, which indicates that the instrument performed the requested action;
4. IM starts the calibration by counting system clocks in the calibration counter and simultaneously sends the rising transition on Update signal;
5. When the instrument receives the transition on Update, it sends the rising transition on C flag;

6. When IM receives the rising transition on C it stops the counter that holds the double value of update signal propagation from IM to instrument.

The precision of the calibration procedure depends on the relative propagation path length of the Update signal and C flag, which mainly depends on the numbers of logic gates on each path. In both cases, AND gates could be used and in both cases the rising transition is sent. Hence, guaranteeing an equal propagation path length is not hard to achieve.

The Calibration procedure should be run on all instruments that need to be calibrated separately and relative delay offsets between them would be calculated subsequently.

9.7.4 Triggered execution of tests & measurements

Just like in traditional instruments, some measurement actions inside the chip require a trigger signal so that the instrument being ready for the mission is waiting for proper conditions. Trigger mechanisms could be used for event-driven data logging, cascaded execution of test and measurement tasks as well as for delay testing and performance measurement. A trigger could be also used as an alternative (simpler) synchronization mechanism for adjacent instruments.

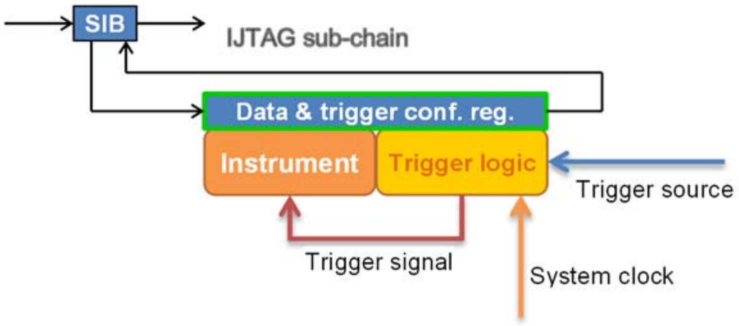


Figure 25. An IJTAG instrument with a triggered execution mechanism

A triggered instrument execution mechanism is presented in Figure 25. In this picture, an IJTAG instrument is supplemented with a trigger logic block that controls the execution of the instrument with a trigger signal line. The trigger logic block has a trigger source input which, when asserted, signals about an incoming trigger event. Then, depending on the configuration of trigger logic, the trigger signal is generated which, in its turn, activates the instrument.

The trigger logic block can be configured via the respective data register placed into the IJTAG scan-chain. The configuration defines how exactly the trigger signal has to be generated in response on assertion of a trigger source input:

- **Direct triggered execution**. The instrument is executed immediately after the trigger source input becomes asserted.

- **Delayed triggered execution**. The instrument is executed after a certain period of time (defined by a specified amount of system clock cycles) after the trigger source is asserted.

Typically, an IJTAG instrument is intended to perform a certain task and does not need to be stopped externally (e.g. BIST). However some instruments can run continuously (e.g. data loggers or performance monitors) and may need to be active only for particular period of time. For those instruments, the stopping condition should also be configured:

- **Triggered deactivation**. Instrument is stopped immediately after trigger source input is de-asserted.
- **Triggered delayed deactivation**. The same as above but with a delay.
- **Manual deactivation**. Trigger signal remains active infinitely long (until it is not cleared explicitly with a special command supplied via IJTAG chain). Such an instrument remains active even if the trigger source becomes de-asserted.
- **N/A or self-deactivation**. The instrument does not require explicit de-activation and will stop after finishing all its tasks.

Finally, the trigger logic block can also support several modes of re-arm:

- **Single-shot**. Trigger is executed only once, on the first assertion of the trigger source signal. For the following executions, the trigger logic has to be explicitly re-armed by an IJTAG command.
- **Auto-rearm**. After execution is finished, the trigger is re-armed again.
- **Delayed auto re-arm**. The same as above but re-arm is performed after certain amount of system clock cycles.

This concludes the discussion of instrument synchronization and interrupt challenges.

10 Summary & Conclusions

In this deliverable D2.1, the research results obtained by the BASTION partners in Task 2.1 have been described. This includes design and validation of embedded instruments for detection of several aging-related faults, in combination with the IEEE 1687 standard. Also issues in the latter in terms of synchronization and interrupts received attention in this research.

After an introduction about aging faults and discussion on state-of-the-art publications, original contributions were proposed towards the BASTION project in terms of embedded instruments.

The proposed approach addresses the design of aging monitors, which enables detecting timing degradation induced in the circuit path because of aging or other timing variations. The sensor's architecture is simple and easy to implement because it only uses standard cells. Therefore, it has relatively less area and power overhead. In addition, the proposed sensor is scalable and its resolution accuracy can be changed by adding or removing a few gates. The circuit has been validated by simulations in a 45nm CMOS process node.

Compatibility with the IEEE 1687 standard makes the proposed sensor more flexible and accessible. Also, it facilitates data logging and fault localization. The experimental results show the capability of our design in terms of detection of aging and measuring timing slacks.

Next, a method where fault localization time is reduced by introducing a segment insertion bit (SIB) that enables self-reconfiguration of IEEE 1687 networks has been presented. Timing analysis was carried out for single and multiple concurrent faults and a way has been proposed to construct the suggested self-reconfigurable network.

The proposed self-reconfigurable network was validated through post-layout simulations of one such network. The proposed scheme was compared with previous similar work and observed at least a 2.8 times reduction in localization time for a single fault and a 6.4 times reduction in case of multiple faults.

Then the possibilities of using Built-In Self-Testing (BIST) for embedded analog-to-digital converters (ADC) for evaluating the aging behavior of this category of IPs were investigated. For that purpose, the ADC-BIST has been enhanced with a digital interface allowing the integration into an IJTAG network. Emphasis is on performance degradation testing due to aging and a flow for this was provided. Three ADCs were used to validate the concept via simulation. The used IJTAG network was optimized for in-field test, and the ADC-BIST can be controlled efficiently to enable the monitoring of aging defects.

Another issue, the access of Embedded Instruments during the operational phase of a system has been investigated. A number of scenarios were tackled, where the duration of a test is critical in several cases. Also a number of access approaches were investigated, each with their own pros and cons; the IEEE 1687 access shows many possibilities, among them, flexibility.

In addition, an automated framework for design, validation and minimization of (EI) error checkers for digital circuits has been developed. It generates a complete valid set of input stimuli. It can minimize the number of checkers, and provide a trade-off between obtained fault coverage and area overhead.

Finally, research was carried out with regard to embedded instrument interrupts, calibrated synchronization of instruments, as well as triggered execution of measurements. Suggestions have been provided to enable and disable internal interrupts via a JTAG-configurable Clock Control block. In terms of calibrated synchronization, a calibration procedure has been proposed which is very efficient. Also a triggered instrument execution mechanism has been presented with many configuration options.

Deliverable D2.1 has significantly contributed to the BASTION project results.

11 References

[1] E. Mintarno, J. Skaf, R. Zheng, J. Velamala, Y. Cao, S. Boyd, R.W. Dutton, and S. Mitra, "Optimized self-tuning for circuit aging," in Design, Automation Test in Europe Conference Exhibition (DATE), pp. 586–591, 2010.

[2] A. W. Strong, E. Y. Wu, R.-P. Vollertsen, J. Sune, G. La Rosa, T. D. Sullivan, et al., "Reliability wear out mechanisms in advanced CMOS technologies," vol. 12, John Wiley & Sons, 2009.

[3] R. Entner, "Modeling and simulation of negative bias temperature instability," Ph.D. Dissertation, Vienna University of Technology, 2007.

[4] T. Grasser and B. Kaczer. "Evidence that two tightly coupled mechanisms are responsible for Negative Bias Temperature Instability in Oxynitride MOSFETs," IEEE Transactions on Electron Devices, vol. 56, no. 5, pp. 1056–1062, May 2009.

[5] T. Nigam and E. B. Harris, "Lifetime enhancement under high frequency NBTI measured on Ring Oscillators," in Proceedings of IEEE International Reliability Physics Symposium, pp. 289-293, 2006.

[6] H. Reisinger, T. Grasser, K. Hofmann, W. Gustin, and C. Schlunder, "The impact of recovery on BTI reliability assessments," in Proceedings of IEEE International Integrated Reliability Workshop (IRW), pp. 12-16, 2010.

[7] C. Guerin, V. Huard, and A. Bravaix. "The Energy-Driven Hot-Carrier Degradation Modes of nMOSFETs," in IEEE Transactions on Device and Materials Reliability, vol. 7, no. 2, pp. 225-235, 2007.

[8] A. Scorzoni, B. Neri, C. Caprile, and F. Fantini, "Electromigration in thin-film interconnection lines: models, methods and results," Materials science reports, vol. 7, pp. 143-220, 1991.

[9] J. H. Stathis, "Physical and predictive models of ultrathin oxide reliability in CMOS devices and circuits," IEEE Transactions on Device and Materials Reliability, vol. 1, pp. 43-59, 2001.

[10] S. Borkar, “Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation,” IEEE Micro, vol. 25, no. 6, pp. 10–16, 2005.

[11] K.P. Parker, “The Boundary Scan Handbook”, 3rd edition, ISBN-13: 978-1402074967, ISBN-10: 1402074964, Springer, 2003.

[12] F. G. Zadegan et al., “Access Time Analysis for IEEE P1687,” IEEE Transactions on Computers, vol. 61, no. 10, pp. 1459–1472, Oct. 2012.

[13] ——, “Design Automation for IEEE P1687,” in Proc. Design, Automation & Test in Europe Conference (DATE), March 2011.

[14] M. Igarashi, K. Takeuchi, T. Okagaki, K. Shibutani, H. Matsushita, and K. Nii, “An on-die digital aging monitor against HCI and xBTI in 16 nm Fin-FET bulk CMOS technology,” in European Solid-State Circuits Conference (ESSCIRC), pp. 112-115, 2015.

[15] T. T. H. Kim, L. Pong-Fei, K. A. Jenkins, and C. H. Kim, “A Ring-Oscillator-Based Reliability Monitor for Isolated Measurement of NBTI and PBTI in High-k/Metal Gate Technology,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, pp. 1360-1364, 2015.

[16] M. H. Hsieh, Y. C. Huang, T. Y. Yew, W. Wang, and Y. H. Lee, “The impact and implication of BTI/HCI decoupling on ring oscillator,” in IEEE International Reliability Physics Symposium (IRPS), pp. 6A.4.1-6A.4.5, 2015.

[17] K. Katoh and K. Namba, “A low area calibration technique of TDC using variable clock generator for accurate on-line delay measurement,” in International Symposium on Quality Electronic Design (ISQED), pp. 430-434, 2015.

[18] K.A. Bowman et al., “Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance,” in IEEE Journal of Solid-State Circuits, vol. 44, pp. 49-63, 2009.

[19] A. Rahimi, L. Benini and R. Gupta, “Application-adaptive guard-banding to mitigate static and dynamic variability,” IEEE Transactions on Computers, vol. 63, 2014, pp. 2160-2173.

[20] L. Lai, V. Chandra, R.C. Aitken and P. Gupta, “SlackProbe: a flexible and efficient in-situ timing slack monitoring methodology,” in IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems, vol. 33, pp. 1168-1179, 2014.

[21] M. Sadi, L. Winemberg, and M. Tehranipoor, “A robust digital sensor IP and sensor insertion flow for in-situ path timing slack monitoring in SoCs,” in VLSI Test Symposium (VTS), pp. 1-6, 2015.

[22] A.J. van de Goor, “Testing semiconductor memories: theory and practice”, John Wiley and Sons, New York, 1991.

[23] P.H. Bardell, W.H. McKenney, and J. Savir, “Built-In Test for VLSI: Pseudorandom techniques”, John Wiley and Sons, New York, 1987.

[24] S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, “A Monitor Interconnect and Support Subsystem for Multicore Processors,” in Proc. Design, Automation & Test in Europe Conference, April 2009.

[25] A. Jutman, S. Devadze, and K. Shibin, “Effective Scalable IEEE 1687 Instrumentation Network for Fault Management,” IEEE Design & Test, vol. 30, no. 5, pp. 26–35, Oct 2013.

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

[27] “IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device,” IEEE Std. 1687-2014, pp. 1–283, Dec 2014.

[28] X. Zhang, “Towards a Dependable Homogeneous Many-Processor System-on-Chip”, Ph.D. Thesis University of Twente, ISBN 978-90-365-3772-8, 213 pages, 2014.

[29] E. A. Stott, J. S. Wong, P. Sedcole, and P. Y. Cheung. “Degradation in FPGAs: measurement and modelling,” in ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 229-238, 2010.

[30] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao. “The impact of NBTI on the performance of combinational and sequential circuits,” in Conference on Design automation (DAC), pp. 364-396, 2007.

[31] K. Shibin, S. Devadze, and A. Jutman, “Asynchronous Fault Detection in IEEE P1687 Instrument Network,” in North Atlantic Test Workshop, pp. 73-78, 2014.

[32] J. Wan and H.G. Kerkhoff, “Reliability of SAR ADCs and associated embedded instrument detection”, 2015 20th International Mixed-Signal Testing Workshop (IMSTW), 24-26 June 2015.

[33] P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan, T. Hollstein, “Automated Minimization of Concurrent Online Checkers for Network-on-Chips,” 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC 2015), 2015.

Additional relevant references, not used in text:

[34] J. Li and M. Seok, “Robust and In-Situ Self-Testing Technique for Monitoring Device Aging Effects in Pipeline Circuits,” in IEEE Design Automation Conference, 2014, pp. 1-6.

[35] F. Firouzi, F. Ye, K. Chakrabarty and M.B. Tahoori, “Representative critical-path selection for aging-induced delay monitoring,” in IEEE International Test Conference (ITC), 2013, pp. 1-10.

[36] A.J. Drake et al., “Single-cycle, pulse-shaped critical path monitor in the POWER7 microprocessor,” in IEEE International Symposium on Low Power Electronics and Design (ISLPED), 2013, pp. 193-198.

[37] A. Bouajila, A. Lakhtel, J. Zeppenfeld, W. Stechele, and A. Herkersdorf, “A Low-Overhead Monitoring Ring Interconnect for MPSoC Parameter Optimization,” in Proc. IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS), April 2012.