FP7-ICT-2013-11-619871

BASTION

Board and SoC Test Instrumentation for Aging and No Failure Found

Instrument: Collaborative Project
Thematic Priority: Information and Communication Technologies

Description of instrument-assisted fault injection and sensitization solutions (Deliverable D1.2)

Due date of deliverable: December 31, 2015
Ready for submission date: February 16, 2016

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

Organisation name of lead contractor for this deliverable: Testonica Lab

Revision 1.3

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 Artur Jutman, e-mail: artur@testonica.com

This document is intended to fulfil the contractual obligations of the BASTION project concerning deliverable D1.2 described in contract 619871.

Table of Revisions

Version	Date	Description and reason	Author	Affected sections
0.1	November 19, 2015	Structure created	A. Jutman	Contents
0.2	November 30th, 2015	Contribution UT	H.G. Kerkhoff	All
0.3	December 10th,2015	Contribution ASTER	C. Lotz	Introduction, background
0.4	December 16th,2015	Contribution ULUND	F. Ghani Zadegan	3.5
0.5	December 16th,2015	Timing-related fault injection sections added	S. Devadze	Chapter 3
0.6	December 18th, 2015	TRF injection updated, 1687 design errors injection added	S. Devadze	Sections 3.3 and 3.4
0.7	December 28th, 2015	References added/formatted	S. Devadze	Section 2.2-3.3.3
0.8	January 6th, 2016	Improving the background section	A. Jutman	Section 2
0.8a	January 7th, 2016	General formatting, editing and polishing	A. Jutman	All sections
0.9	January 15th, 2016	Reviewing, addressing comments	C. Laudert, S. Devadze, A. Jutman	All sections
1.0	January 18th, 2016	Adding TUT's contribution	J. Raik	Section 3.8
1.1	January 22nd, 2016	Update of introduction and conclusions. Polishing of other parts	A. Jutman	All sections
1.2	February 8-9th, 2016	Improving introduction and Summary	A. Jutman	Sections 1, 4
1.3	February 15-16th, 2016	Preparing for submission	A. Jutman	All

Author, Beneficiary

Artur Jutman, Igor Aleksejev, Sergei Devadze, Testonica Lab
Christophe Lotz, ASTER Technologies
Carsten Laudert, Infineon
Hans G. Kerkhoff, Alireza Rohani, Hassan Ebrahimi, University of Twente
Jaan Raik, TU Tallinn
Farrokh Ghani Zadegan, Erik Larsson, University of Lund
Matteo Sonza-Reorda, Politechnico di Torino
Rene Krenz-Baath, HSHL

Executive Summary

This document reports on BASTION activities in *fault injection techniques* covering the full spectrum of BASTION research topics including aging, IRFs, performance related faults (e.g. bit slip), as well as various faults and design errors in IEEE 1687 networks.

List of Abbreviations

AOI	- Automated Optical Inspection
ADE	- Cadence AD
ASIC	- Application-Specific Integrated Circuit
BIST	- Built-In Self-Test
BSDL	- Boundary-Scan Description Language
BST	- Boundary-Scan Test
CAD	- Computer Aided Design (also EDA)
CMOS	- Complementary Metal-Oxide Semiconductor
CPU	- Central Processing Unit, also Processor
DDR	- Double Data Rate synchronous memory
DPM	- Defects Per Million
DPMO	- Defects Per Million Opportunities
DQ	- Data input/output in DDR
DQS	- Data Strobe Signal in DDR
DRAM	- Dynamic RAM
DRC	- Design Rule Check
EDA	- Electronic Design Automation (also CAD)
EMS	- Enhanced Manufacturing Services
FIFO	- First In, First Out (data buffer)
FPGA	- Field Programmable Gate Array
FP7	- European Union's 7th Framework Program
HCI	- Hot Carrier Injection
HTOL	- High-Temperature Operating Life
HW	- Hardware
IC	- Integrated Circuit
ICL	- Instrument Connectivity Language
ICT	- In-Circuit Test
JEDEC	- Joint Electron Device Engineering Council
IJTAG	- Internal JTAG, a short name for IEEE 1687 standard and infrastructure collectively
IPIRF	- Intellectual Property (hardware module in FPGA or SoC)
	- Intermittent Resistive Fault
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;
MPS	- (Coverage metrics based on) Material, Placement & Solder
MIG	- Memory Interface Generator

MGT	- Multi-Gigabit Transceiver
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)
PCOLA/SOQ	- (Coverage metrics based on) Presence, Correct, Orientation, Live, Alignment/Short, Open & Quality
PCB	- Printed-Circuit Board
PCBA	- Printed-Circuit Board Assembly
PDL	- Procedural Description Language
PMOS	- p-type Metal Oxide Semiconductor
POST	- Power-On Self-Test
PPVS	- (Coverage metrics based on) Presence, Polarity, Value & Solder
PVT	- Process, Voltage, Temperature
RAM	- Random-Access Memory
RD	- Reaction-Diffusion (model)
RTD	- Research and Technological Development
SBST	- Software-Based Self-Test
SERDES	- Serializer/Deserializer
SIB	- Segment Insertion Bit
SoC	- System on Chip
SVF	- Serial Vector Format
SODIMM	- Small Outline Dual In-Line Memory Module
SW	- Software
TDR	- Test Data Register
TRF	- Timing Related Faults
TSMC	- Taiwan Semiconductor Manufacturing Company
UUT	- Unit Under Test
VHDL	- VHSIC Hardware Description Language
VHSIC	- Very High Speed Integrated Circuit

Table of Contents

1 Introduction

This deliverable describes the research performed in BASTION task T1.3 “Instrument-assisted fault injection and sensitization”. The faults, which are being experimented with in regard to injection and sensitization, are majorly related to research outcomes of T1.1 and T1.2. The former task was mainly connected to *board and system level* (collection of boards or the product as a whole), while the latter one mainly considered the *IC level*. Therefore, it is important to keep in mind the difference between the levels while reading this document: different target faults are considered at different levels. We also detail these aspects in Chapter 2.

In addition to faults contributing to NFF and aging, we have added to this document, a few classes of faults related to instruments (checkers) and instrumentation networks (IEEE 1687 a.k.a. IJTAG). It is important to address them as they are also part of reliability and fault management strategy in BASTION.

To summarize, the *IC-level* faults considered in this document are:

- aging faults;
- faults in checkers;
- faults and design errors in IJTAG networks.

The *board and system level* faults and defects considered in this deliverable are:

- timing-related faults (e.g. transition delays and bit slips);
- signal integrity problems (bit errors and crosstalk);
- intermittent resistive faults (IRF).

Respective results are reported about the research performed by BASTION partners on fault injection and sensitization.

1.1 Structure of the document

This report is structured as follows. First, in Section 2, we provide the background information about target faults along with motivation for the respective research activities. Section 3 details, per fault category, the architecture of fault injection solution, the experimental setup and in some cases the experimental results obtained with the setup. Section 4 summarizes the document.

2 Background

The faults identified in WP1 and listed in D1.1 as those important from the perspective of NFF and aging are major candidates for fault injection and sensitization experiments reported in this deliverable. These faults as described later are falling into different categories and considered at different integration levels (IC or board). In addition to that, fault injection for IEEE 1687 networks has been also considered. In this section, we give background information on target errors, faults and the methodology of their injection and sensitization.

The root causes of both system-level and IC-level NFF have been extensively studied in frames of Task 1.1 as a result of the following activities:

- analysing the industrial best practices facilitated by the BASTION survey;
- the study of state-of-the-art *board-level* test coverage metrics;
- QuadDPMO experiments, on big data from several OEM companies;
- experimenting with IRFs.

Based on that, it has been concluded that:

1. Insufficient coverage/escape rate is the most significant contributor to *board-level* NFF (related to final integration and assembly test).
2. Aging is the second contributor, as stress testing is not yet being widely used by *board assembly* companies.
3. Lack of communication between the design and manufacturing departments has a measurable influence on final product quality.
4. Intermittent and timing faults are confirmed as *system-level* NFF contributors, without any possibility to measure the proportions, due to lack of test technique or test coverage metrics.

The insufficient coverage/escape rate important relations between faults, tests and test coverage have been analysed yielding four hypotheses:

a. Insufficient coverage: test available but test coverage is low.
b. Missing test method: test can target the fault, but it is not implemented on the test line.
c. Missing fault model: test can target the fault, but no coverage metrics exist.
d. No test on certain faults: Test cannot target certain faults.

Points c) and d) need a special attention in BASTION and fault injection solutions considered in this deliverable are expected to facilitate further research in this direction.

The overall results of T1.1 (survey and QuadDPMO experiments) push to investigate, on two types of faults, which are key contributors to NFF and which are not being addressed by board-level test coverage.

1. Intermittent faults like Intermittent Resistive Faults (IRF).
2. No coverage for certain faults, like timing related faults (TRF).

The class of NFF faults has also been described in detail in deliverable D1.1 and it has been shown how this type of faults can be modelled and injected into the Cadence simulation environment; in several papers [3,6,14] it has been also shown how IRFs change the behaviour in analog as well as digital CMOS circuits.

This injection model has been the basis from which a *hardware* IRF generator has been designed and implemented. The generator has already been used in preliminary IRF hardware experiments at board level described in [3]. The latter generator will be discussed in detail in this deliverable.

The aging faults category, more specifically the ones that are caused by NBTI has been paid a special attention to in deliverable D1.1 as a result of research performed in T1.2. The injection of these faults has been implemented in the (Cadence) simulation environment, and the developed model by UT [2]. It enables the evaluation of any analog, mixed-signal or digital CMOS circuit under the influence of aging. Applied (stress) voltages, temperatures and time are the used parameters by the designer. It has also been shortly discussed in D1.1.

In the domain of *timing-related and performance faults*, we have been studying two particular fault types potentially contributing to *board-level NFF*: the delay-faults (transition delays) and crosstalk effects. Bit slip and bit-error faults common to high-speed links (multi-gigabit speed) have been also considered. For these fault types we have been carried out simulation experiments and the experiments with fault injection on real hardware. As a final result of this analysis, special fault injection instruments have been developed. These instruments are capable to induce the above-mentioned effects in functional designs and evaluate the behavior of a particular system under stress of the injected faults.

Board-level *delay faults* can be caused by different kinds of physical defects, for example due to presence of major solder voids on PCBA, missing termination, wrong value of an in-line serial resistor, parametric variations or environmental conditions. All these inconsistences can result in delays in arrival of signal transitions at the receiver side of a high-speed communication link.

The crosstalk occurs because of magnetic (inductive) and electric (capacitive) fields between two coupled lines. The closer they are, the stronger the fields and the higher the amount of resulting crosstalk effects. Crosstalk should be considered in three dimensions i.e. in addition to the length of wires, both the height between two adjacent PCB layers and width between the traces play a role in the strength of parasitic coupling.

The line which propagates its signal to adjacent line is called “aggressor” and the line affected by the propagated signal is called “victim”. The crosstalk is also caused by discontinuities in circuit layout, like connectors and vias, where reflections and capacitive coupling is the greatest, and which are critical points that act like antennas [10].

The crosstalk can cause different performance problems such as increased noise levels, undesired spikes, jitter on data edges or reflections of signals. Due to the crosstalk, the transition timing of data is changed at the receiver side and the signal’s eye opening (a measure of signal quality) becomes degraded [12]. More information on crosstalk is provided in [11].

Bit-error and bit-slip are typical timing-related faults occurring on multi-gigabit serial communication links. Bit-errors are single or multiple bits that are being flipped due to distortion of the transmitted signal by physical defects of the link or due to

environmental conditions. Bit-slip is a loss or undesired gain of a single bit during the transmission. After bit-slip is occurred, the receiver starts to incorrectly receive following bits due to misalignment. The correctness of transmission is restored on the next re-alignment procedure.

Typically, a certain amount of bit-errors and bit-slips is allowed by the specification of high-speed communication channel. These errors are normally corrected at the upper layers of the protocol stack that can employ for example ECC correction algorithms and/or perform re-transmission of the corrupted data frames. However, a link is considered to be defective if bit-error rate (BER) exceeds some predefined threshold.

At the same time, fault correction mechanisms often hide problems with link quality from the test engineer who is working on functional/application level. The fault injection instrument described in Chapter 3 helps to assess the consequences of bit-errors and bit-slips at application level as well as to validate functional tests in terms of ability to evaluate link quality.

3 Fault Injection Solutions and Experiments

This chapter describes all fault injection solutions and experimental setups including respective instrument architectures developed in BASTION.

3.1 Intermittent resistive fault injection

In deliverable D1.1, a simulation-based intermittent fault injection in Verilog-A has been used to evaluate IRF effect on digital systems [3]. However, simulation-based fault injection is time-consuming, particularly in case of IRFs. Emulation-based fault injection is an alternative solution for accelerating fault injection. This technique also allows studying the behaviour of a circuit in real time, and also large (PCB-based) systems can be validated in real-time [5].

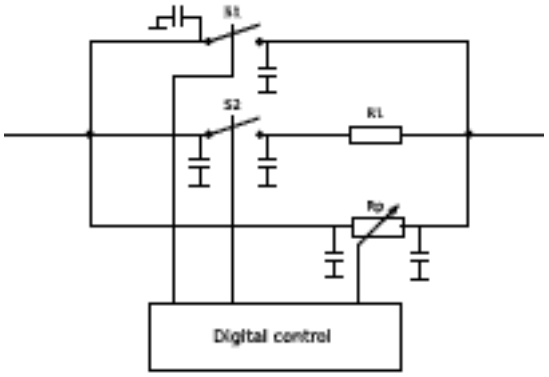

Figure 3.1-1: Basic set-up of the IRF hardware generator.

In Figure 3.1-1, a straight-forward implementation of the IRF hardware generator is shown. It is able to be connected as a programmable wire between *any* two nodes. Of course there are some constraints with regard to the branch voltages and allowed branch currents. The basic idea is to use a network of fixed and programmable resistances. The total range of resistance, as well as the resolution of the smallest resistance change were design parameters. To some extent it resembles a (successive approximation) SAR-like technique. The desired value is split into fixed resistances, and the smaller differences are added via a digital controller.

The most obvious disadvantage of this architecture is that its speed behaviour will be limited by the RC combinations, of which the intrinsic parasitic capacitances are the most difficult to circumvent. However, this was found to be acceptable for the first line of approach. First, its capabilities in a real environment had to be investigated,

An automated hardware/software platform for emulation-based IRF injection has been developed. The proposed emulation-based platform is composed of a host computer, an FPGA and a home-made PCB board which is shown in Figure 3.1-2.

The host computer executes a Matlab script which generates a burst of random resistance values based on the model was described in deliverable D1.1. The

communication screen on the computer, together with the essential parameters can be seen in Figure 3.1-3. The generated burst sequence can be transferred to the FPGA board by a serial communication link.

The FPGA is responsible to synchronize and generate the control signals for the on-board programmable switches and potentiometers on the PCB board. The desired resistance range is provided by a combination of fast digital potentiometers and fixed range resistors.

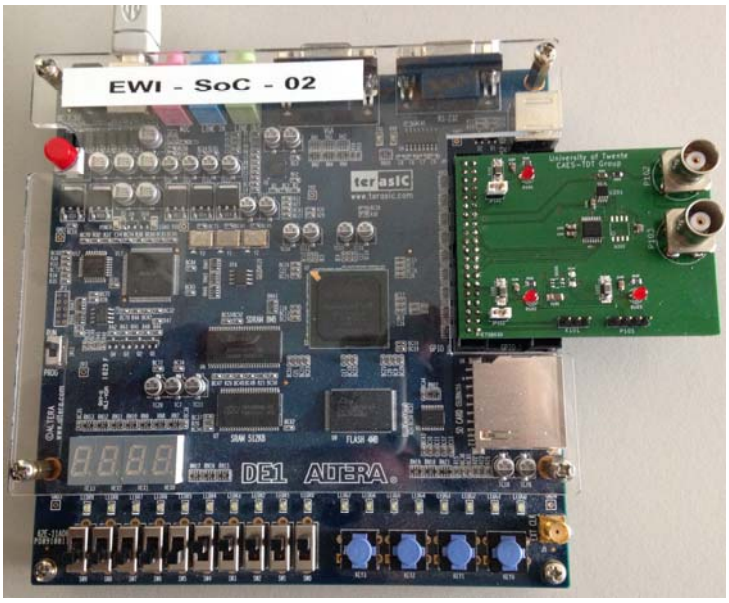

Figure 3.1-2: Actual implementation of the IRF generator. The FPGA houses the digital control, and the right-hand board the switches and resistors.

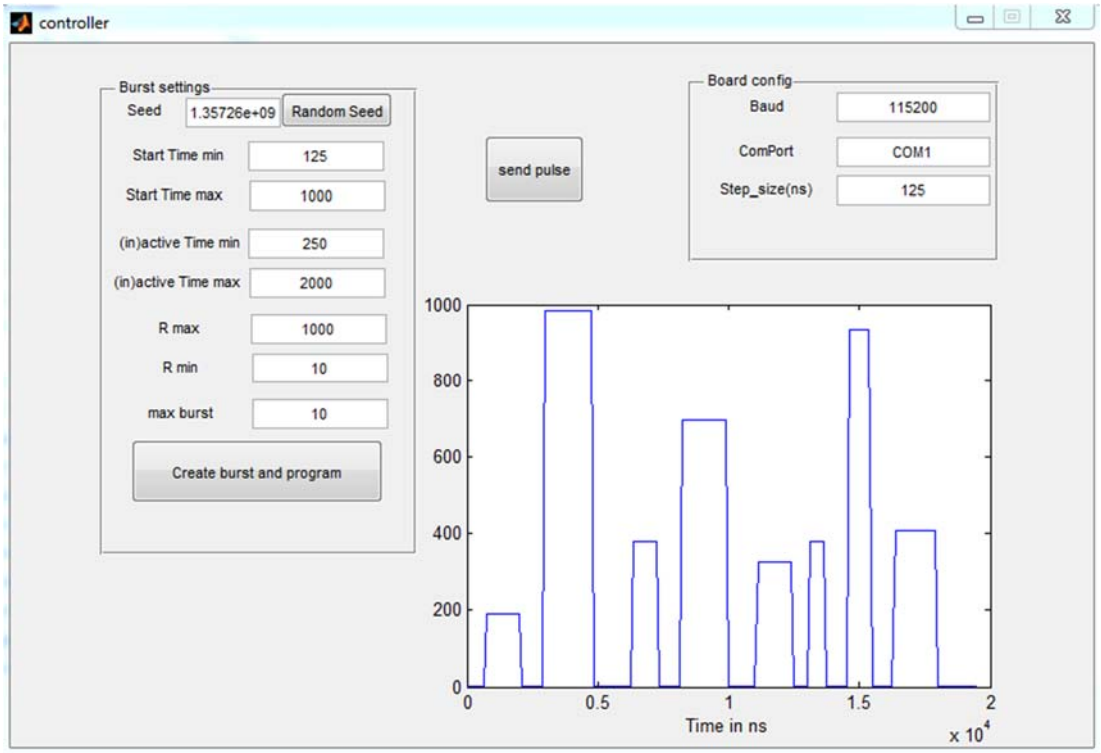

Figure 3.1-3: The communication window on a PC (USB-based) to set the IRF generator parameters.

An example of a generated IRF by our generator is shown in Figures 3.1-4a and 3.1-4b. There are two waveforms in the figures; one is a resistance sequence measured from the PCB board and the other is the expected sequence generated by the script. As it can be seen, the measured resistance sequence approximately follows the expected sequence with an inevitable error because of the parasitic capacitances of the existing switches and potentiometers. As can be seen from Figure 3.1-4a, the actual measured signals are less than the ones set by the program.

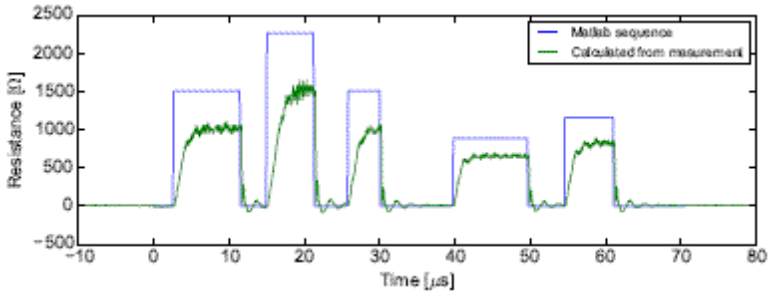

Figure 3.1-4a: The actually measured and Matlab-set pulses generated by the IRF hardware generator.

By means of calibration, these signal values can be improved, and the results are shown in Figure 3.1-4b. It turned out, the noise on top of the measured signals had to do with improper grounding procedures during the measurements. They could be removed after proper grounding.

The weakest point of our current straight-forward design implementation is the relatively low speed, and long durations of the pulses. These are caused by the intrinsic parasitic capacitances in combination with the programmable resistances. To some extent, buffering could help, but for our required short pulses (50ns), a different architecture is compulsory in the future.

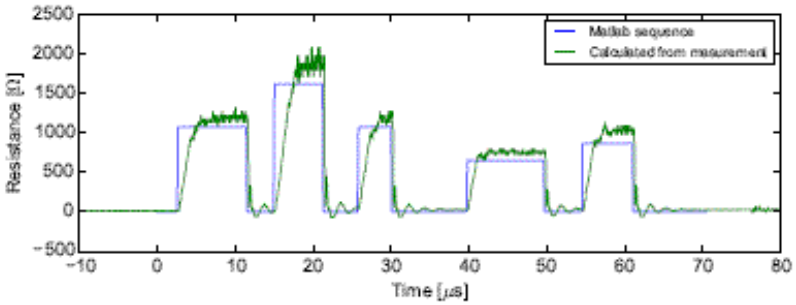

Figure 3.1-4b: The actually measured and Matlab-set pulses generated by the IRF hardware generator *after calibration*.

In our current research activities, the proposed IRF generator is being used to evaluate the influence of IRFs on analogue and digital systems in real time at printed-circuit board (PCB) level. An impression of the test set-up is seen in Figure 3.1-5. The detailed results of this research, however, are part of a future deliverable.

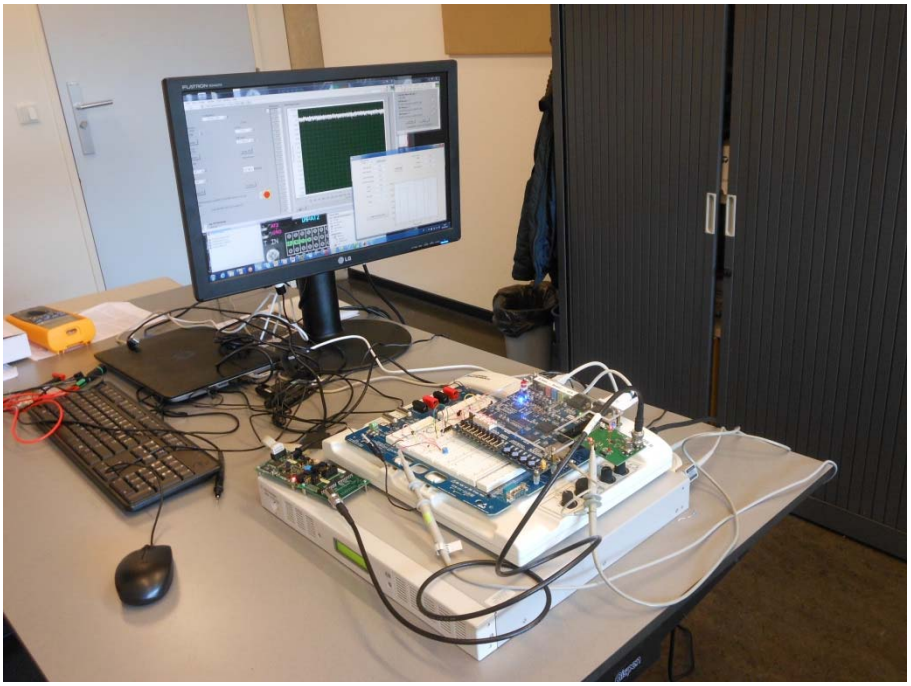

Figure 3.1-5: The application of the IRF generator in an NFF PCB measuring experiment.

3.2 Aging fault injection

The injection of NBTI-based faults has already been shortly discussed by the University of Twente in deliverable D1.1. It can be easily extended for a combination of NBTI and HCI–based faults. It is building on a new model, supported by measurements, that solves the well-known Reaction-Diffusion (RD) equations in a smart way. It is referred by us as the Compact NBTI model.

It was incorporated in Cadence ADE using Verilog-A and the Spectre simulation environment. Several analog as well as digital circuits in 65nm TSMC were evaluated in terms of aging by simulations in this way. Delay is confirmed to be the major contributor at system level to NBTI-based faults. This has been confirmed by actual aging (HTOL) measurements. Several publications (e.g. [6]) have resulted from this.

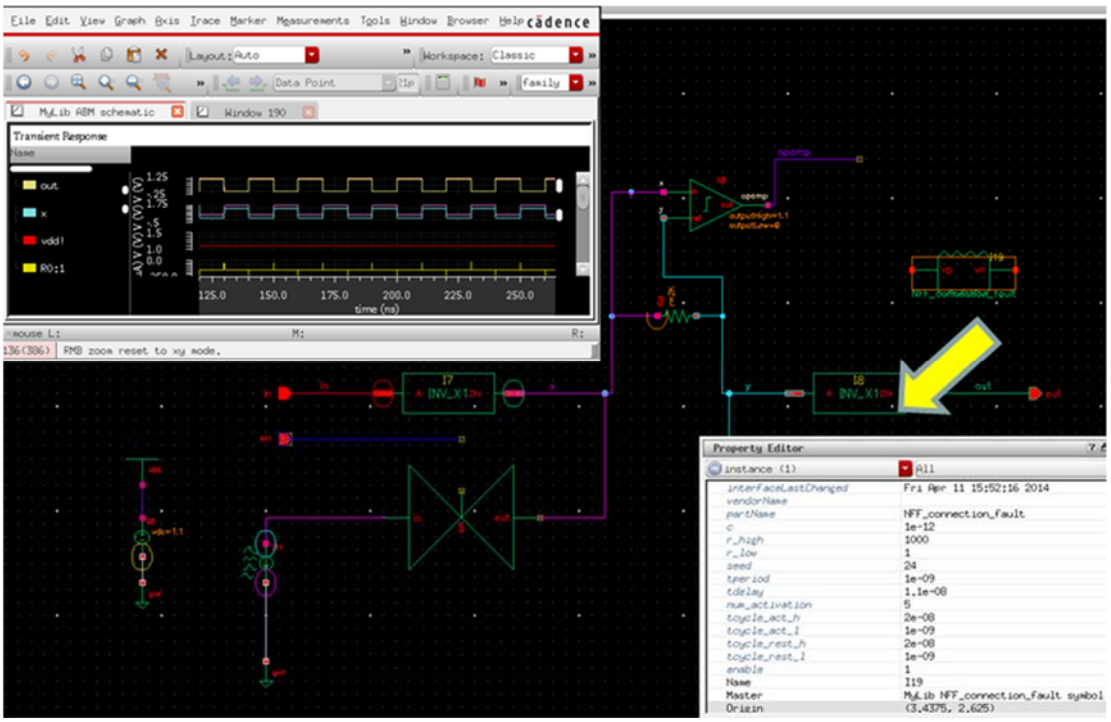

Figure 3.2-1: The Cadence-based IRF generator for aging simulations.

An example of the environment is shown in the figure above (Figure 3.2-1). On the right-hand bottom side, the communication window of the IRF generator the six parameters can be filled in. In the center is the transistor scheme of the digital CMOS circuit (Cadence), while in the left-upper corner the simulation results of aging of the circuit can be seen.

3.3 Delay fault injection

On *board level* various manufacturing defects can result in delays in signal transitions on a high-speed bus. A defect causing transition delay fault on the board cannot be detected just by executing test stimuli statically or at low test application speed. However, the currently available PCBA structural test methods like Boundary Scan cannot achieve high speed of test application and thus are unable to detect timing-related faults. As an illustration, the figure below (Fig 3.3-1) presents the results of simulation that shows how the resistance of a transmission line affects shape of the transmitted signal. The signal rate of 20MHz has been intentionally selected to be close to the typical upper bound for the board-level structural test methods.

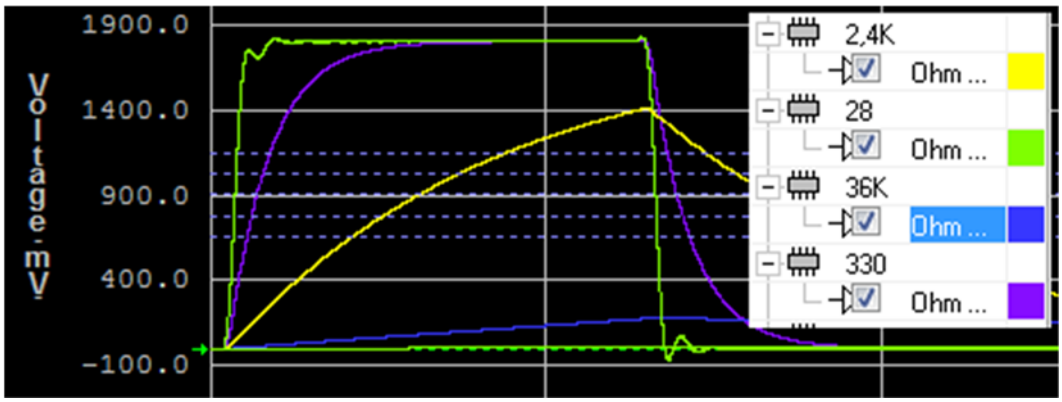

Figure 3.3-1: Shape of 20MHz signal depending on transmission lie resistance

The Fig 3.3-1 was generated using HyperLYNX PCB Analysis tool of Mentor Graphics [7]. As the source data for simulation we have selected standard characteristics of DDR2 data transmission (defined by JEDEC). The simulation has shown that if, for instance, the resistance of a line has been increased from the nominal 28 Ohms (typical for DDR2 SODIMM memories) to 330 Ohms (e.g. due to a manufacturing defect), the shape of the 20MHz signal will not change much. Thus if a test is running at 20MHz pattern application rate – it will not detect delays in signal transmission. However, in operational mode, the same signal is being transmitted at the full speed of DDR2 (e.g. of 400MHz) and its shape may be distorted even by a minor increase of line resistance. The latter can lead to partial corruption of the transmitted data and needs to be reliably tested in production.

The experiments with injection of delay faults into real system have confirmed the simulation results. In these experiments, effect of delay fault has been be emulated by increasing the resistance of a particular net on PCBA (i.e. by changing value of in-line resistor). The experiments were carried out using Xilinx Zynq ZC702 [9] and Xilinx Virtex5 ML505 [8] FPGA development boards. In both cases delay faults have been injected into high-speed interconnection lines between FPGA device and DDR2/DDR3 memories.

The results of experiments are presented in Table 3.3-1. In the first case (Zynq board), we have changed the signal characteristics of the memory clock line (CK) by introducing a resistive short with ground (by means of extra pull-down resistor). For the Virtex5 board, a resistance of data line (DQ) has been increased. In both cases, the interconnection test has been executed at different speeds (to emulate slow Boundary Scan test and high-speed test). For high-speed test, different types of memory write/read accesses have been used (single or burst read/write mode). In single access mode, we executed a single read or write operation followed by a long idle state on memory bus. In burst mode, several consequent accesses have been carried out which caused signals on data lines to change on every clock edge.

Injected Fault	Resistor value	Boundary Scan test results	High-speed test results	
Resistive short with GND on memory clock line (CK)	0 (short)	Fault detected	Fault detected	
	5.6 Ohm	Fault undetected	Fault detected	
	10 Ohm	Fault undetected	Fault detected	
	15 Ohm	Fault undetected	Fault undetected	
	22 Ohm	Fault undetected	Fault undetected	
			Single read/write	Burst read/write
Increased resistance of data line (DQ)	Open	Fault detected	Fault detected	Fault detected
	330 Ohm	Fault undetected	Fault undetected	Fault undetected
	2.4 kOhm		Fault undetected	Fault detected
	36 kOhm		Fault detected	Fault detected
			Fault detected	Fault detected
	1 MOhm		Fault detected	Fault detected

Table 3.3-1 – Experimental results: delay fault injection on real hardware

The table above shows that only high-speed test is capable to detect delay-related defects ("fault detected" *shown in green*) while low speed tests as well as incorrectly prepared test patterns will not succeed in delay fault discovery ("fault undetected" *shown in red*). In particular, the burst read/writes have to be used to force data change on each clock edge. As for Boundary Scan, this technique has shown inability to detect timing-related faults even if the delays injected into transmission line were caused by tremendous increase of line resistance (1 MOhm).

Instrument for delay fault injection

The developed instrument is an IP core that is capable to imitate delay faults (permanent or intermittent) on data (DQ) or data strobe (DQS) lines between FPGA and memory device. The instrument IP is targeted to be used in functional design in conjunction with Memory Interface Generator (MIG) IP which is a standard controller for DDR2/3 memories in Xilinx FPGAs. As a result, this instrument can be inserted into ready-made functional designs (Fig 3.3-2) that utilize MIG IP for accessing on-board DDR memory.

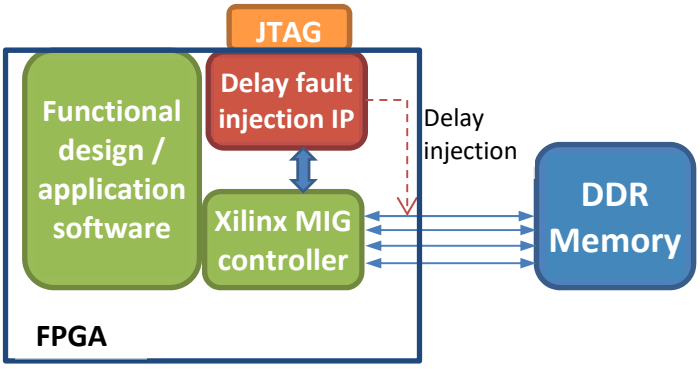

Figure 3.3-2: Architecture of delay fault injection instrument

The instrument is capable to insert extra delays on selected data or strobe lines. In particular the IP exploits capabilities of Xilinx I/O buffers (IOB) to add/adjust fine-grain delays on FPGA I/O line. The instrument is controlled using JTAG bus. The latter allows its on-the-fly activation, deactivation and configuration without the need to stop/re-configure FPGA entirely or pause execution of running functional design.

The following parameters of fault injection can be configured:

- Target line: data line - DQ or differential data strobe – DQS
- Amount of extra delay in 1/32th fraction of source signal frequency for DQ line and 1/128th in case of DQS
- Type of injection:
  - Permanent (fault stays injected permanently)
  - Single-shot (fault is injected just once)
  - Continuous (faults are continuously injected (periodically))
- Fault sensitization/activation condition:
  - Manual (using JTAG-based control software)
  - By trigger (trigger line has to be connected with functional design)
- Fault activation hold-off (period between received activation signal and actual activation)
- Duration of stress (in 4ns periods)

All these parameters are accessible and configurable via ChipScope JTAG interface while the functional design is operating.

3.4 Bit slip and bit error injection

On FPGAs, high-speed communication is normally implemented using built-in multi-gigabit transceiver (MGT) hard IP cores. The developed fault injection instrument is capable to interface with MGT and induce faults (bit-errors and bit-slips) in the received serial data stream. In particular, single bit-error can be injected by simple flip of a selected bit at parallel port of SERDES inside build-in gigabit receiver.

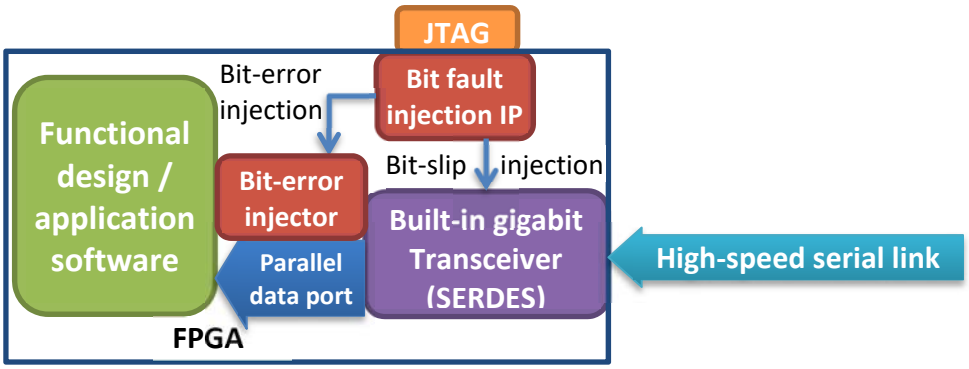

Figure 3.3-3: Architecture of bit-slip and bit-error fault injection instrument

Bit-slip is a loss or gain of a single bit during the transmission. For injection of bit-slip faults, the instrument exploits a special feature of MGT normally used for channel alignment. Channel alignment is a special procedure carried out at link initialization/recovery phase. In alignment mode, a transmitter continuously sends a pre-defined data-word over high-speed channel. At the same time, receiver tries to align itself along the received data stream to match this data-word. For that purpose, MGT receiver logic is capable to slip a single bit of received data (i.e. shifts the received data stream by one position) when a special signal is asserted. Normally, in channel alignment mode, the bit-slipping is repeated until the correct word is caught by the receiver.

In case of fault injection, the same functionality is used to induce temporal misalignment and imitate a bit-slip error. The fault injection instrument asserts the respective signal inside MGT receiver which causes a single bit-slip in the received data stream. As a result, subsequent sequence of bits becomes invalid until link re-initialization or link recovery is performed by OS/driver.

The instrument supports the following parameters of fault injection:

- Type of fault: bit-error or bit-slip
  - Bit-error mask (defines which bits in data-word have to be flipped)
  - Number of bit-slips to produce (single or double)
- Type of injection:
  - Permanent (fault stays injected permanently)
  - Single-shot (fault is evoked just once)
  - Continuous (faults are evoked periodically after specified amount of received data words)
- Fault sensitization/activation condition
  - Manual (using JTAG-based control software)
  - By trigger (trigger line has to be connected with functional design)
- Fault injection hold-off (period between the received activation signal and the actual activation; the period is counted in number of received data words)

The developed instrument allow to experiment on how bit-slips and bit-errors influence on the overall functionality of running application. It is also possible to evaluate how well a functional test is prepared and running on application level of a particular system can detect problems with high-speed speed data transmission over serial communication channels.

3.5 Crosstalk Fault injection

The effect of crosstalk was simulated in HyperLYNX environment by placing two signal routes very close to each other (Fig 3.3-4, left side). The simulation results are presented in Fig 3.3-4 (right side).

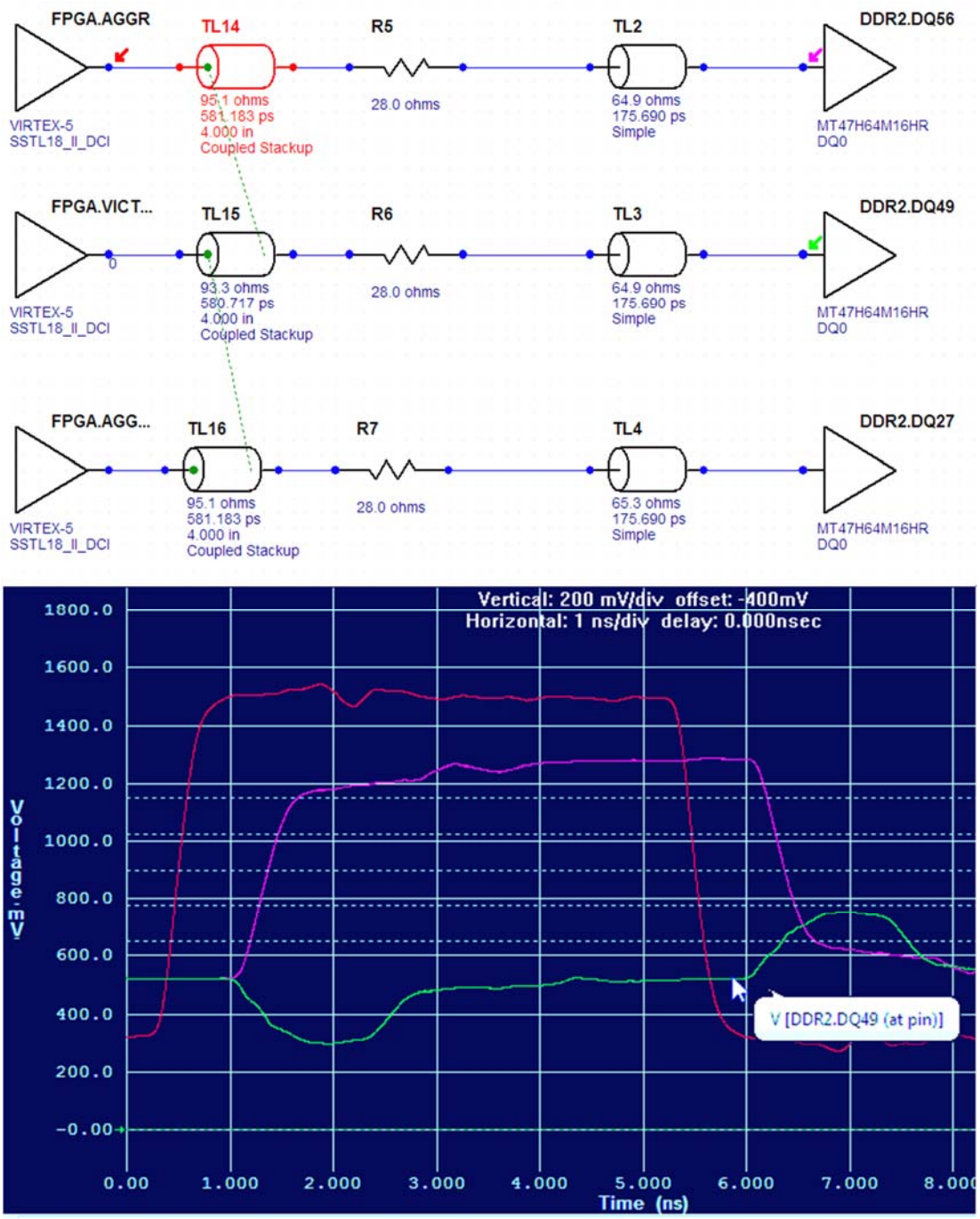

Figure 3.3-4. Simulation of crosstalk in HyperLYNX environment

The overall effect on signal transmission could be different:

- Glitch (caused by static transmission mode)
- Rising/falling delay (caused by odd transmission mode)
- Rising/falling speed-up (caused by even transmission mode)

The effect of crosstalk can be clearly seen in the right side of the picture. The signal transition on two aggressor lines (shown by red and purple colors) causes a glitch on the transmission line located between them (the green-colored signal).

The experiments on crosstalk detection were performed on Virtex ML505 evaluation board with SODIMM DDR2 memory installed. The crosstalk effect was emulated by

injecting capacitors (4.7pF each) between a selected memory data line ("victim") and two adjacent data lines ("aggressors"). In this setup different sets of test patterns have been executed at-speed. The results of the experiments are outlined in table below.

Pattern set #	Stimuli / Expected measurement			Measured	Crosstalk detected	Comment
	DQ56 (Aggr)	DQ 49 (Victim)	DQ27 (Aggr)	DQ 49 (V)		
#1	0	0	0	0	NO	No crosstalk expected
	0	0	0	0		
#2	0	0	0	0	NO	No crosstalk expected
	0	1	0	1		
#3	0	1	0	1	NO	No crosstalk expected
	0	0	0	0		
#4	0	0	0	0	NO	1 aggressor; (test for "glitch")
	1	0	0	0		
#5	0	1	0	1	YES	1 aggressor (test for "falling delay")
	1	**0**	0	**1**		
#6	0	1	1	1	NO	1 aggressor, 1 helper (test for "falling delay")
	1	0	0	0		
#7	1	1	1	1	NO	2 aggressors (test for "glitch")
	0	1	0	1		
#8	0	0	0	0	YES	2 aggressors (test for "glitch")
	1	**0**	1	**1**		

Table 3.3-2 – Experimental results: crosstalk faults injection on real hardware

The first three experiments (pattern sets #1-3 in Table 3.3-2), consisted in executing patterns that would not induce crosstalk even if the defect is present. These experiments expectedly produced correct results on the "victim" line despite the parasitic capacitance between the two adjacent nets.

The next pattern set #4 could induce crosstalk however the experiment shown that single aggressor is not always enough to make the crosstalk effect be significant enough to produce detectable glitch. When the second aggressor has been added (pattern set #8), the defect was detected by observing glitch on the victim signal line.

The experiments with sets #5 and #6 show that crosstalk can be abated due to the presence of a "helper" i.e. a signal on the line adjacent to the victim line which is transitioning in the same way as the victim signal (pattern set #6). Because of the "helper" line the defect has not been detected in the experiment #6. However the same defect has been detected by pattern set #5 (where no "helper" is present).

The experiment #7 did not detect glitch while aggressor signals have been switching from high to low. We can suggest that depending on hardware and nature of the defect receiver may tend to interpret incorrect signal value (i.e. below V_{IH} but above V_{IL}) as "logic-1" thus even with 2 aggressors the test was unable to detect "0-glitch".

In general, the experimental results have shown that in order to detect board-level timing-related faults, the tests should be not just applied at-speed, but also appropriate test patterns have to be selected, which is not a common practice today for PCBA test, that resorts mainly on the functional test.

3.6 Design error injection in IEEE 1687 networks

The following fault injection instruments are capable to inject design errors into IEEE1687/IJTAG instrumentation networks. The main purpose of error injection is to provide a way to evaluate IEEE1687 test and validation algorithms and check their ability to catch design errors.

The overall scheme of IJTAG fault injection is presented in Fig 3.4-1. The green part (on the bottom side of picture) represents the original 1687 network where the dashed lines show the functional connections of wires prior to insertion of fault injection instruments. The middle part shows two types of fault injection instruments inserted into the design (the types are discussed below). Depending on the need, each of the instruments can be activated (i.e. activates fault injection) or be remain in de-activated state. The activation is controlled by a separate IJTAG network (orange colored in Fig 3.4-1) which consists of a series of sequentially connected test data registers (TDRs). Each TDR controls activation of the corresponded fault injection instrument.

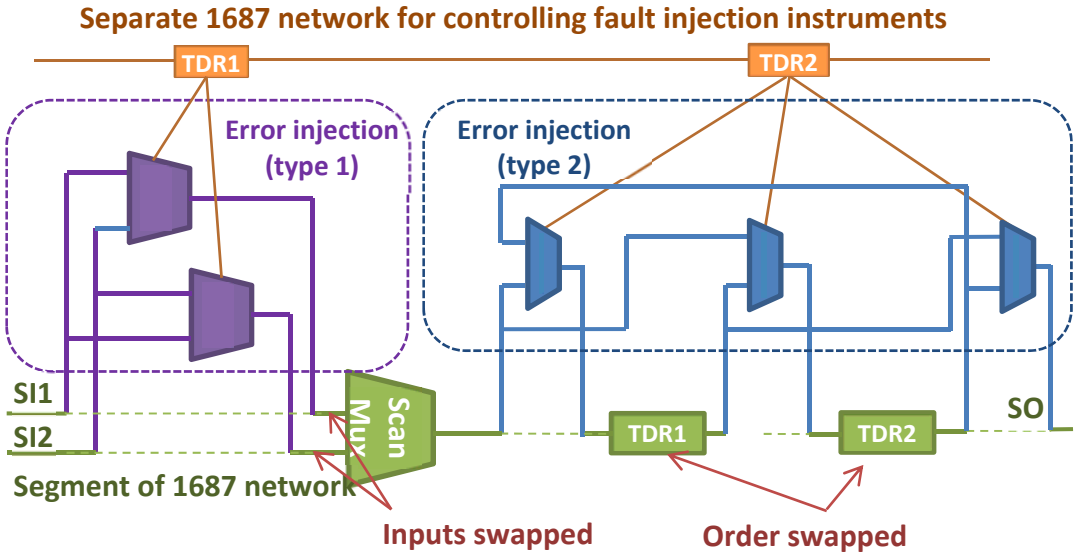

Figure 3.4-1. Instruments for design-errors injection into IEEE1687 network

Two types of IEEE1687 design-error injection mechanisms have been developed. The first instrument (marked by violet color in Fig 3.4-1) has to be attached to selected scan multiplexor (ScanMux) of IJTAG network. When the instrument is activated it switches (exchanges) connections to the inputs of the multiplexor. This emulates a typical design error i.e. mixed-up wiring of multiplexor inputs or erroneous assignment of its control signal.

Another type of instrument (blue part in Fig 3.4-1) has to be attached between two sequentially connected segments of the network. When activated the instrument swaps the order of these segments. Such type of error corresponds to the case, when order of scan-segments differs in hardware implementation and in the specification (ICL file).

3.7 Defect injection in IEEE 1687 networks

The purpose of defect injection is to emulate the effect of manufacturing defects on IEEE 1687 networks. The idea is that by controlled injection of defects into a "network under test" we evaluate the effectiveness of our test and diagnosis tools. Defect injection is done via another IEEE 1687 network, referred to as "Defect injector network" in this work. Figure 3.5.1 shows how the network under test and fault injector networks are added as custom test data registers (TDRs) to the JTAG circuitry.

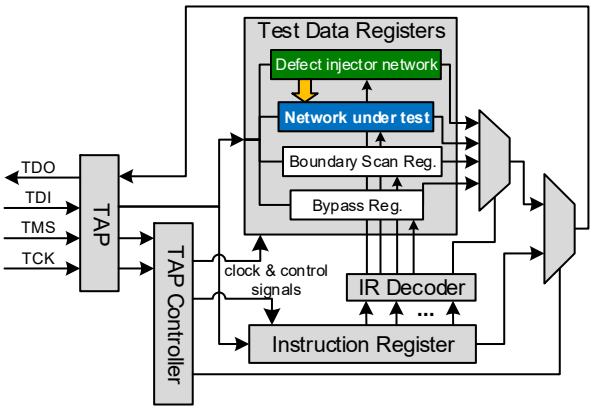

Figure 3.5.1 Two IEEE 1687 networks are added as custom TDRs to the JTAG circuitry: (1) a network under test, and (2) a fault injector network

To inject the defects a number of basic component will be used. Figure 3.5.2 shows some sample component which can be used for adding Stuck-At and Bridging faults. These components will be inserted into candidate locations in SIBs and other IEEE 1687 network blocks, and by default (i.e., when their control input is not pulled high) act as a wire (with a bit of delay). To activate the intended defect type, the control signal is pulled high via the defect injector network. In practice, to keep the number of register bits in the defect injector network low, especially when many defect injector components are to be used, decoder circuits can be employed (i.e., instead of direct one-to-one connection from the register bits in the defect injector network to the control inputs of the defect injector components).

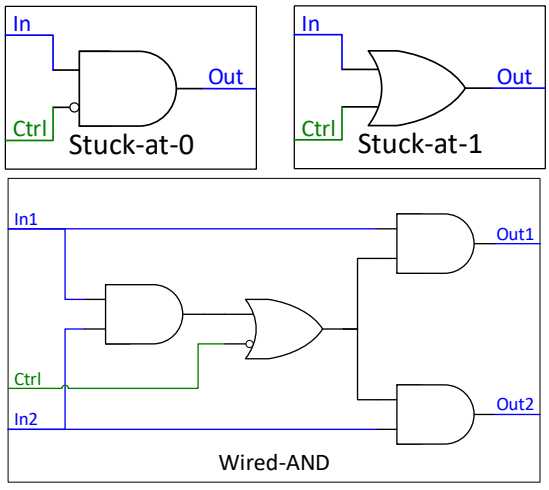

Figure 3.5.2 Defect injector components

As an example of IEEE 1687 network blocks, Figure 3.5.3 shows schematic of a Segment Insertion Bit (SIB). A SIB is a programmable component which is used to switch a shift-register (or another network segment) connected between its TSI and FSO ports on and off the scan-path. The colored dots on the schematic mark some candidate points where the defect injector components can be inserted to emulate different defect behaviors.

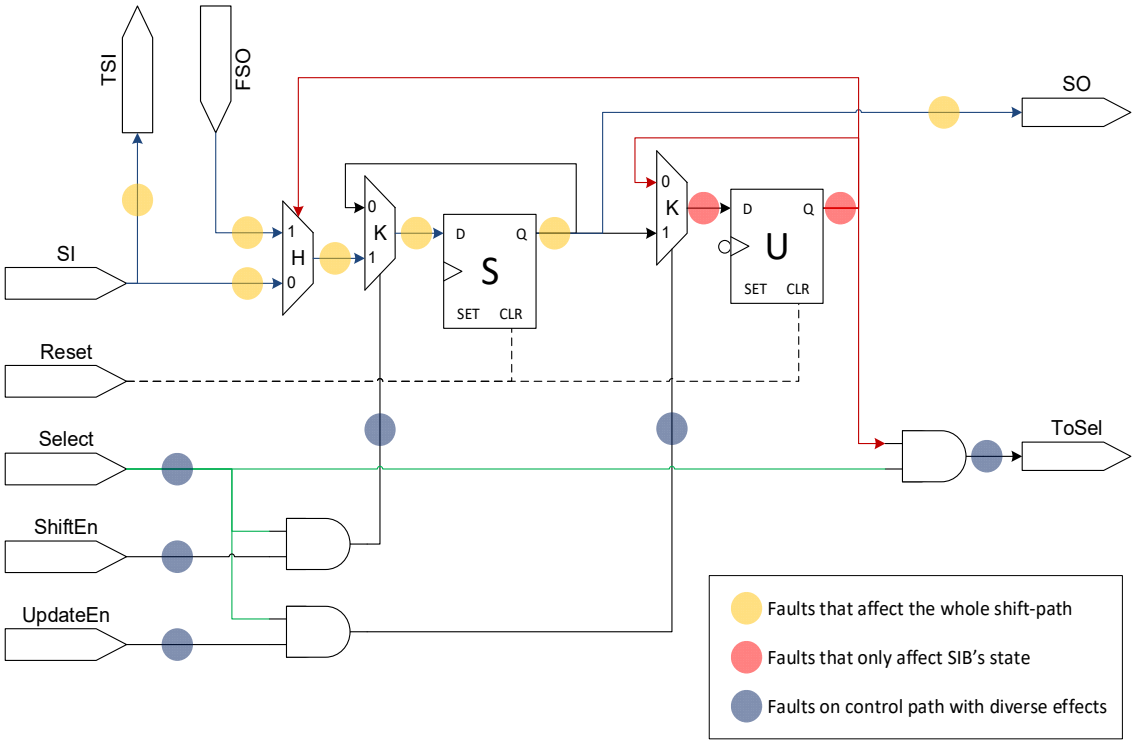

Figure 3.5.3 shows schematic of a possible implementation of a SIB as well as candidate points for fault injection

3.8 Fault injection for validating error checkers

In BASTION, we consider digital error checkers in addition to analog health monitors. These checkers are intended to detect faults related to wear-out effects and to single event upsets during the normal system operation. In order to validate and test the checker infrastructure in the field, a low-area fault evoking instrument solution has been developed.

Figure 3.8.1 illustrates the main concept of the BASTION fault injection architecture for in-field testing of error checkers. The online test for the checkers is launched by the Controller by activating the Fault Evoking instruments (FE) and simultaneously isolates the checker outputs by the Fault Isolation (FI) block. An FE is a simple logic circuit for inserting logic 0 and 1 values to the signal lines monitored by the Error Checker, an FI in turn, is a logic block that sets the value of the error flags to 0 (i.e. to "no error detected").

The controller periodically launches the online tests. In order not to lose the fault detection capabilities of the checker during the normal operation, the test application

takes place at the time steps when the checker is inactive. E.g. if the task of the checker is to perform checks related to the address data it reads, the test will be triggered by handshaking signals indicating the checker is not reading a new address or is currently in an idle mode.

Currently, the cost-effectiveness of the proposed approach in terms of area and achieved fault coverage is being validated on the BASTION demonstrator design. Preliminary experiments on a router design show that the proposed fault injection architecture consumes less area with respect to the error checkers while providing the full fault coverage for the checker circuitry.

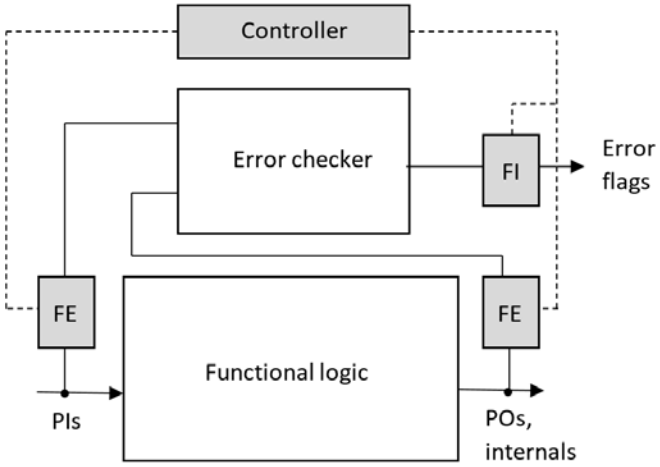

Figure 3.8.1 Fault injection architecture for in-field testing of error checkers

4 Summary

In this deliverable, we have reported fault injection techniques and solutions developed in frames of BASTION. This work facilitates further experiments in WP2, WP3 and WP4 as well as it does contribute to the BASTION demonstrator (WP6).

The following types of faults have been considered in the deliverable:

- Aging and wear-out
  - board-level intermittent resistive faults (IRF);
  - IC-level aging faults due to NBTI.
- Timing-related faults (TRF) and signal integrity issues
  - transition delays;
  - bit slips;
  - bit errors;
  - crosstalk.

Faults injected into checkers are similar to stuck-at faults.

5 References

[1] *Big Trouble with "No Trouble Found" Returns,* Accenture Report, 2008, Available at: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture_Returns_Repairs.pdf

[2] J. Wan, H.G. Kerkhoff and J. Bisschop, *"Simulating NBTI Degradation in Arbitrary Stressed Analog/Mixed-Signal Environments",* in IEEE Transactions on Nanotechnology, November 2015.

[3] H.G. Kerkhoff and H. Ebrahimi, *"Investigation of Intermittent Resistive Faults in Digital CMOS Circuits",* IN Journal of Circuits, Systems and Computers, World Scientific Publishing Company, 2015/2016.

[4] M. Brakels and D-J. van de Sanden, "*Design and Implementation of a hardware IRF generator*", Report of University of Twente, July 2015, 18 pages.

[5] H.G. Kerkhoff and H. Ebrahimi, "*Detection of Intermittent Faults in Electronic Systems based on the Mixed-Signal Boundary-Scan Standard*", in Proc. of the Asian Quality Electronic Design Conference (ASQED), Kuala Lumpur, Malaysia, August 2015, pp. 1-6.

[6] Kerkhoff, H.G. and Ebrahimi, H., "*Intermittent resistive faults in digital cmos circuits*", in: IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2015, 22-24 April 2015, Belgrade, Serbia. pp. 211-216. IEEE Circuits & Systems Society. ISBN 978-1-4799-6779-7

[7] HyperLYNX PCB simulation solutions, https://www.mentor.com/pcb/hyperlynx/

[8] Virtex-5 LXT FPGA ML505 Evaluation Platform, http://www.xilinx.com/products/boards-and-kits/hw-v5-ml505-uni-g.html

[9] Xilinx Zynq-7000 All Programmable SoC ZC702 Evaluation Kit

http://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html

[10] "PCB Crosstalk Fundamentals - What It Is and How You Can Prevent It", Mentor Graphics Webinar, https://www.mentor.com/pcb/multimedia/crosstalk-fundamentals-webinar

[11] "Time Domain Methods for Measuring Crosstalk for PCB Quality Verification", Tektronix Application Note

[12] Stephen Scearce, „PCB Crosstalk Fundamentals and Strategy", Cisco web presentation

[13] X. Bai, S. Dey, "High-level Crosstalk Defect Simulation for System-on-Chip Interconnects", in Proc. of VTS 2001

[14] J. Wan and H.G. Kerkhoff, "The Influence of NFF in Analog CMOS Circuits",International Mixed-Signal, Sensors and Systems Test Workshop (IMS3TW), Porto Alegre, Brazil, 2014.