DC1598

Joining Boundary-Scan to NMR Architectures: the XMR Approach

Jose Miguel V. Santos
FEUP/ ISEP- R. S. Tome, 4200 Porto, Portugal
jmvs@dee.isep.ipp.pt

Jose Manuel M. Ferreira
FEUP- Rua Bragas, 4050 Porto, Portugal
jmf@fe.up.pt

Abstract

As the JTAG 1149.1 infrastructure is becoming a requirement, and IC fabs developed solutions to get reliable scan cells at very small cost, designers can start thinking to use BST for on-line operations.
Exploring the cohabitation of BST to overcome the drawbacks in VLSI designs, seems promising to allow new ASICs and future PLDs get a dependable behavior in Fault-Tolerant applications, at the really feasible and acceptable cost of duplicating the mission logic.

Keywords: Fault-Tolerance, VLSI defects, Concurrent Test, BST, Fault-Location.

1- Stating the Problem

Question: how to design a dependable VLSI IC, tolerant to *permanent* single faults?
Answer: a straightforward solution is not possible because VLSI ICs present specific defects which impact dependability [1, 2].

The objective is to satisfy the *specification* with the minimum possible overhead, reducing the impact on the *mission time (MTTF)*, yield and cost.
Two main design solution are currently available to design *Fault-Tolerant* (*FT*) systems: *Replication* and *Self-Checking* (*SC*) designs [3, 4]. *Error Correcting Codes*, the other known approach, are useful for data transmission and storage, but, for the moment, not for data processing.

SC designs have their specific target, *temporary faults* mainly, and may lead to an interesting hardware overhead (12-80%), depending on the regularity and size of the structures [5]. Known implementations concern data paths mainly, since control and glue logic can hardly be coded, and sequential designs also present a drawback. Given these limitations, in practice, *SC* designs may need additional test methods, as IDDQ and Thermal monitoring.
The above figures do not usually include the logic to confine the errors and provide a *safe*, or at least *known*, output when the application so requires.

Replication can be in *time* or in *space*. Time-replication is only available with microprocessors, obviously not for *permanent* (hardware) faults, and questionable for *intermittent* faults. Here, to deal with hardware faults, we are only concerned with the second case, or *physical* replication of the *mission circuit (mc)*.

A *SC*-based IC can be expected to have a global hardware overhead around 35-60%, say in the order of 50% to the *mc* [6]. By comparison, a duplicated *mc* means a 33% extra overhead, but is much easier to design. Permanent faults, however, require a *SC* design to be Duplex, and compares closely to a Triple Modular Redundant (*TMR*) design, as for overhead. Concerning the fault model, both architectures have their own advantages and limitations, well know when implemented with standard components, but VLSIs pose new challenges [7].
Since their introduction, ICs always showed raising levels of reliability; today, *life-critical* applications as aerospace and avionics, are pursuing designs with *Commercial-of-the-Shelf* Components (*COTS*), aiming low cost and fast results [8]; they also allow a near complete debug provided by many users, but new *FT* and test schemes are being required. Dependable circuits are traditionally obtained with *Fault-Avoidance* and *Fault-Tolerance*; since the use of COTS restricts fault-avoidance, the emphasis is now focused on *FT*.

There are no much choices to design an IC tolerant to hardware faults, which have to be considered as CMOS geometries shrink, leading to ICs more sensitive to electromagnetic radiation and charged particles, the memory elements mainly. VLSI ICs, with many gates available, allow high-redundancy designs to fit in a chip, but we can not really take advantage of it because the reliability of such ICs is impacted by VLSI specific defects, as well as the power dissipation and MTBF, the yield and the final cost in consequence. A duplex *SC* design is hardly feasible in a single IC: the yield will be too low. TMR architectures are an alternative, but a similar overhead impacts the yield much like the same way. What to do then?

2- BST to the Rescue

As the IEEE 1149.1 BST infrastructure [9] is becoming a requirement in new designs, for off-line test, designers may consider using it on-line. Scan-cells having attained a good reliability, the BST infrastructure in VLSI designs is now a promising vehicle to allow new ASICs and PLDs get a dependable behavior.

Several reasons are leading BST expansion:
- dedicated ATE equipment is being replaced by more generic ATE, aiming low cost and flexibility, since production cycles and volumes tend to decrease. Such features are only possible through BST.
- *system-level* interface and ATE interface (to fade implementation details), are presently strong reasons to justify the inclusion of an on-board *BST-controller* (BSμC), compliant to the IEEE 1149.5 std Module Test and Maintenance (*MTM*) [10]. Such a resident controller may also perform tests on power up, on-line monitoring, and report failure situations hierarchically. The BSμC required in our approach is a simpler controller than the std MTM, and we will keep this distinction for the moment, but the BSμC can be merged into the MTM.
- ever increasing, also, is the reusability of IP cores, many of them derived from well known COTS, for which vendors provide a set of test patterns, and testing rules where applicable. Integrated in more complex VLSI designs, *Controllability and Observability* (*C&O*) constraints usually require dedicated scan chains, or scan-BIST structures, to test them.

BST was already proposed to fulfill *SC* requirements [11], but the solution is not 1149.1 compliant. Also for scan-BIST it seems to present interesting features [12]. To help NMR designs on-line, as far as we know, no proposal has been made until now.
NMR designs at logic level, usually assume immediate voting and then N≥3, leading to the well known TMR (or higher) architecture. At system level, however, voting is not immediate most of the times, because of two main reasons:

- the need to synchronize the systems: desirably, they must be not synchronized to minimize the influence of temporary defects,
- voting, if performed by each system to the other 2, means an unavoidable delay.

This delay is not critical indeed; analyzing the majority of *FT* systems, two things become clear:

1. error confinement must be immediate, avoiding errors to spread, possibly being no more detectable.
2. some delay, or *latency*, is usually acceptable to resume operation; Real-Time (RT) systems may allow some milliseconds up to seconds and more.

Back to the initial question, how to join all this to design a dependable VLSI IC? Hardware faults are mostly a result of fabrication defects which pass the tests and degrade lately. To cope with permanent faults a duplicated *mc* is the minimum possible hardware, but has no information enough to decide which *mc* is faulty. This information is traditionally provided through coding schemes or a 3rd *mc* in TMR designs, both weakened by VLSI defects, if integrated in a single IC.

Is it possible to reuse the BST infrastructure on-line (and the on-board BSμC) to allow a 2-*mc* design perform better than simply fail-stop? Such a circuit is highly desirable for dependable applications, alone or in N-plex modules, but to answer this we must satisfy some points:

- confine the error immediately,
- provide additional information to locate and disable the faulty *mc*,
- during the latency delay, outputs must be set to a known state,
- to resume a single-*mc* operation, an additional error detection mechanism must be provided.

In a first, and traditional, approach we may consider a 2-*mc* design, with output comparison to detect errors, and the std BST infrastructure to possibly read inputs and outputs, compare them to "gold" patterns stored in the BSμC ROM, decide which *mc* is faulty, disable it through BST, and feed the outputs with the good *mc*. Theoretically possible, this means a lot of scan operations, requiring a powerful BSμC. Extra logic is also needed to supply a known output, and finally this process can be shown to have a low scan efficiency, if the number of patterns in memory is to be minimized, as desirable concerning ROM size.

There is another, more effective, solution: upgrade POST [13] to deal with a 2-*mc* design. In POST, the 1149.1 infrastructure was enhanced to allow concurrent operation: input pins are observed and compared to the simulation patterns (the "golden" vectors), scanned by the BSμC. An input match triggers the output scan-cells to provide the desired operation, as described in [13]. This on-line monitoring is non-intrusive and the enhanced infrastructure is 1149.1 compliant, the overhead to the std being less than 15%. The more relevant feature here is that, passing to the scan infrastructure the ability to compare patterns, scan requirements are strongly reduced and allows to use a very simple, reliable, BSμC.

In the following we consider the *mc* to be completely debugged, but, as it is monitored on-line, we will call it the *circuit under test*, or *CUT*, according to the common terminology.

3- The XMR Architecture

A *XMR Module* is a 2-*CUT* IC (or part of it, an IP core for example) bounded by a scan chain, and expected to behave as a Fault Containment Region [14] in dependable systems. Concerning the *fault model*, we consider *temporary* and *permanent* single-faults, acceptable after burn-in [15]. Using replication, *common mode faults (cmf)* must also be considered, and will be discussed later.

Enhancing POST to deal with a 2-*CUT* design, we obtained the following architecture:

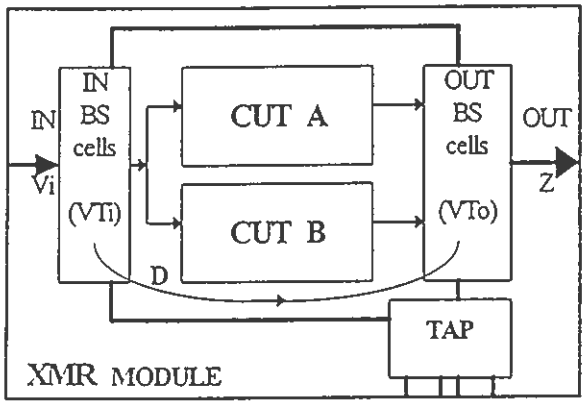
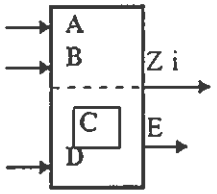


Fig.01 - The XMR architecture

In this scheme we find the duplicated *CUT*s (A,B), a single set of input scan cells and another set of output cells. Input cells can be *std* or *Obs*(ervation)-only, but output cells were redesigned to receive both *CUT* outputs and provide decisions.

The scan operation is similar to POST operation, as described in [13]: the BSμC scans the input simulation pattern (VTi), and also VTo if required; VTi stays in the cells for the time an input match is expected to occur. This match triggers the OUT cells (D= detection), which are 1149.1 std cells redesigned to provide decisions and a TMR-like behavior.

Basically these cells have the output bit (Zi), the error signal (E), and 2 actions:



1. *comparison*
2. *voting*

Comparison of *CUT*s (A, B) confines single-*CUT* errors, has the highest priority and is independent of the BSμC. A *CUT* mismatch sets E, to force the outputs (Zi) into one of 4 known states:

- 0, 1, high-impedance
- to hold the current output.

The 3 first values can be predefined to allow any *known-safe* output pattern (Z). To hold the current output, the BS cell needs an extra latch.

Voting is only possible with the information provided by the BSμC. Triggered by the D signal, meaning that an input match has occurred, the bits of the output simulation pattern (C) are also expected to match the replicas (A, B). Voting allows 2 kind of detection:

- A=B≠C means a *common mode error*, or *cmf*.
- following a *CUT* mismatch, voting allows to locate the faulty *CUT*, and redirects the outputs to resume a single-*CUT* operation.

It must be noticed that both *CUT* are supposed to continue operation during the latency process, which is particularly important for sequential logic.

The mode of operation can be justified by the theory of information applied to digital testing [16]. Let Q be the *quantity of information* in each *CUT* and Q_M the *quantity of information* in the module. The BSμC has the patterns generated for *functional verification* stored in the ROM, and is able to provide additional information; we define $\rho \in [0,1[$ as the fraction of information accessible through BST.

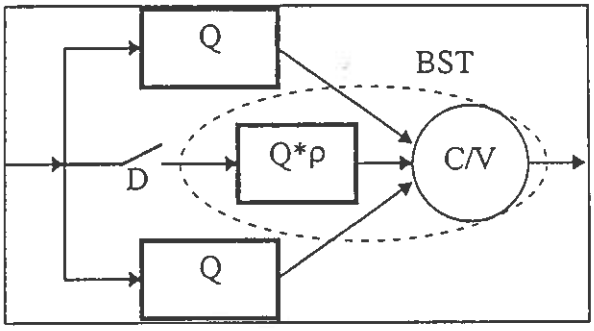


Fig.02- XMR module: the TMR equivalent

Every time an input match occurs (simulated by the switch D), the BST infrastructure acts as the 3rd *CUT* in a TMR, allowing the XMR module to deal with $Q_M=Q*(1+1+\rho)$, and provide a *TMR-like* behavior with discrete voting.

A XMR module, with $X \in [2,3[$, has 4 modes of operation:

a) if the BSμC is not available, then $\rho=0$ and, as *stand-alone*, only error confinement and a known-safe output are possible.

b) when supported by BSμC, the XMR module starts working in the *normal mode* $\{Q_M=Q*(1+1+\rho)\}$, providing error confinement and a known output until operation is resumed. In the case of a permanent fault, operation will resume as *single-CUT* $\{Q_M=Q*(1+\rho)\}$, the right *CUT* supervised in POST mode to provide a timely error detection. If the second *CUT* also fails, there is a last *survival* mode $\{Q_M= Q*\rho\}$, in which a simulation pattern (ROM) is uploaded and VTo injected, when Vi matches VTi. Obviously restricted, this mode goes a step further *fail-stop* designs.

4- XMR-scan: Reliability and Implementation

Here we discuss the reliability of the scan-cells and infrastructure, concerning FPGA and ASICS, as well as XMR extension to NMR architectures. In the above presentation, it is always assumed a reliable and proper BST infrastructure. This will be discussed now.

First of all, it is known that 90-99% or more of the faults are temporary. Since both *CUT* continue operation during the recovery process, combinatory *CUT* are expected to recover naturally, and we are now exploring the possibility to reload on-line the memory elements of a faulty sequential *CUT*.

Currently XMR is implemented in a FPGA, which is not the ideal target because:

- *CUT* separation is difficult to guarantee, and trying to do so impacts the logic optimization process.
- POST-XMR logic needs to access some nodes inside the scan cells; as this is not allowed to the embedded BST chain (the one used to program the FPGA), a second BST infrastructure was necessary.

To avoid this undesirable overhead, future FPGAs may have a native XMR-BST infrastructure. The first point may also justify new FPGAs with logic programmable independently. As obvious, both problems are easily solved when designing ASICs.

Since XMR works concurrently, inputs have the natural excitation, allowing *Obs*-only scan-cells; cells providing Controlability (the *std* cell) are mandatory to the outputs only. *Obs*-only cells have 3 benefits:

- they add no delays to the signal path, contrary to the 2 gate delay of a std cell,
- the cell logic is about 1/2 of the std cell,
- lower probability of introducing faults, since no gates are inserted in the signal path.

There is however a drawback: the two latches of a std cells allow a new pattern to be scanned while the preceding one is searched; the lower TCK frequency allowed improves BST reliability.

The truth table to redesign the decision cells is coherent with a TMR, assuming the XMR-BST has a reliability *not lower* than a *CUT*. Certainly, future experience only may allow to get more precise information, but we based our reasoning as follows:

Known figures show the BST infrastructure to overhead 5-7% on average [17,18]. The 1149.1 std cell, redesigned for XMR decisions, increases by 50%; the other XMR circuitry adds 6-7% to the std TAP controller. Then, XMR global overhead to the 1149.1 std points to 20%, for 40-50 cell chains.

As a result the XMR infrastructure will be about 1.2*7%, or near 10% of a single *CUT*.

The consequence is that, for the hardware, a XMR module has X<2.1.

The other point is the BSμC; the one required to support XMR operation is a Finite State Machine only, with no software, 7 mandatory pins plus 1 pin per *CUT*, and 1-4KB of ROM to store the test vectors. This kind of controller is really simple, can be designed fail-stop and may easily reach a complete debug.

As shown in [19], the BSμC can verify the integrity of the BST infrastructure and, having this in mind the decision cells were designed to involve their logic in the capture of the *true* output (Zi), when they are scanned out to read Vo (Z). This test is transparent, may avoid to design *self-testing* cells, and can be made on-line without interfering with comparison, but disables voting. We are investigating possible enhancements.

The signal path in the decision cell has 3 mandatory gates, against 2 in the std 1149.1 cell, and this may slightly impact reliability but, as a whole, the XMR-BST infrastructure may be expected *no less* reliable than a *CUT*, supporting the TMR rule assumption above.

The latency interval, between disagreement and voting, can be estimated as a function of the number of inputs (n). We investigated 2 usual values: 8 and 16 inputs.

The results, assuming an *equi-probable* input distribution and a new input every interval T (T=1/f; f is the operating frequency), show that, for frequencies typical in the RT world, say 1-20 Mhz, the average delay (given by $td=(2^{n-1})*T$) is compatible to the latency interval acceptable in many cases. The chart below shows this average delay for a 16 bit module.

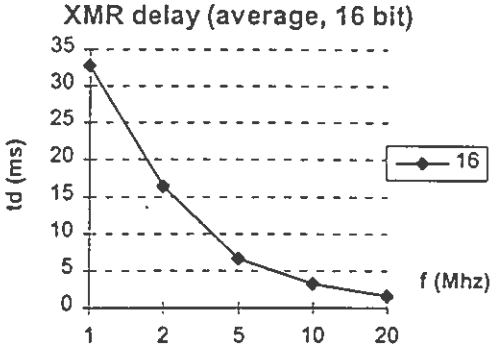


Fig.03- Average latency for recovery

These results show that XMR modules up to 16 input can be used in most RT systems, for which the acceptable delays reported usually fit in the 20-100ms range. An 8 bit module has a similar behavior, the average delay being divided by 256.

Low operating input frequencies, say below 50Khz, can also be monitored: another read-search process allows the patterns to be verified. The TCK required, in all cases, is never higher than 10Mhz.

5-Extension to NMR and Sequential designs

There are examples of recent *FT* designs using ASICs with duplicated logic (total or partial) and internal comparison to provide a fail-stop IC [20]. Joining two such ICs, designers can get a duplex design, *FT* for single-faults. Most of these IC have embedded scan infrastructures, and the upgrade to XMR ICs has a very small cost, the overhead being similar.

As a single BSμC may feed the 2 XMR-modules in parallel with no extra cost or delay, we may find a duplex XMR-IC design behaving as a 6-MR-like architecture, able to degrade successively, down to a "single-IC-single-*CUT*" operation mode, sustaining several possible combinations of permanent faults.
This is a really powerful architecture, as the 2 first failing *CUT*s (permanent) will not affect system operation, no matter where they occur.
More complex designs are also reported, but we believe that, for the moment, they will be not required with XMR designs.

XMR modules with sequential *CUT*s are more complex to design. First we must notice that single-*CUT* errors are always confined. Besides the inputs, memory elements are also required to have *Obs*-cells to locate the fault. Such a kind of approach is also explored for scan-BIST and looks promising, with a low design cost and good fault coverage [12].
As the number of *Obs*-cells strongly impacts the input matching delay, partition has to be done, as for other scan-based approaches. Our experience points to a limit of 16-20 *Obs*-cells, for usual frequencies and latency delays. Assuming a similar number of outputs, on average, each scan chain may have around 40 cells, a value found in other approaches. Multiple scan was also shown not to impact necessarily the overhead [21].

Conclusions

The *XMR* architecture seems promising for NMR architectures, with really a small cost, since the BST infrastructure is now present in virtually all new designs. The implementation in ASICs has no special requirements, but today FPGS and PLD have some limitations, mainly because the embedded BST infrastructure can not be used requiring a second one.
High-safety systems may require physical independence of the replicas, but, even here, replicas built of XMR ICs can be an advantage over the single *CUT* counterpart, and a duplicated channel built with XMR ICs may be even better than a traditional XMR.
The simulation patterns can be derived directly from the specification, and are needed in XMR designs for the recovery process and *cmf* detection only. The *fault coverage* they provide is relevant for *cmf* only.

REFERENCES

1- I. Koren, A.D. Singh, "Fault Tolerance in VLSI Circuits", *IEEE Computer*, pp73-82, July 1990.

2 - M. D'Abreu, A. Stokes, L. Sassoon, "Defect Analysis-Impact on Yield Improvement and to the Design of Manufacturable ICs", *IEEE ETW'97, Compendium of Papers*, Session 4, Italy, 1997.

3- P.K. Lala, *Fault Tolerant , Fault Testable HW Design*, Prentice/Hall International, 1985.

4- M. Nicolaidis, S. Noraz, B. Courtois, "A Generalized Theory of Fail-Safe Systems", *FTCS-19 Digest of Papers*, IEEE Comp. Society Press, 1989, pp398-406.

5 - R.O. Duarte, I.A. Noufal, M. Nicolaidis, "A CAD Framework for Efficient Self-Checking Data Path Design", *3rd IEEE IOLTW*, Greece, 1997, pp28-35.

6- E. Bohl, R. Stephan, W. Glauert, "The Architecture of the Fail-Stop Controller AE11", *3rd IEEE IOLTW'97*, Crete, Greece, 1997, pp. 47-52.

7- M.d'Abreu, P. Isakanian, S. Hunjan, "Understanding of the Fabrication Process- Key to Design and Test of Mixed Signal Integrated Circuits", *IEEE ETW'98, Compendium of Papers*, pp.156-9, Barcelona, 1998.

8 - DW Caldwaell, DA Rennels, "FTSM: A Fault-Tolerant Spaceborn Microcontroler", *IEEE FTCS-28 Digest of FastAbstracts*, pp 3-4, Munich, 1998.

9 - *IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture*, IEEE Inc, NY, 1990.

10 - *IEEE Standard 1149.5 , Module Test and Maintenance Bus Protocol*, IEEE Inc., NY, 1994.

11- M. Lubaszewski, B. Courtois, "On the design of Self-Checking Boundary Scannable Boards", *Proc. of ITC*, 1992, IEEE, pp.372-81.

12 - G. Kiefer, H-J Wunderlich, "Deterministic BIST with Multiple Scan Chains", *IEEE ETW'98 Compendium of Papers*, pp. 39-43, Barcelona, 1998.

13 - J.M.V Santos, J.M. Ferreira, "An Approach to On-Line Failure Detection Based on BST", *DCIS'97*, Spain, 1997.

14 - J.H. Lala, R.E. Harper, "Architectural Principles for Safety-Critical Real-Time Applications", *Proc. IEEE*, V82 n1, Jan 1994, pp25-40.

15 - R.Roques, A. Correge, C. Boleat " Fault-Tolerant Computer for the Automated Transfer Vehicle", IEEE *FTCS-28 Digest of Papers*, pp. 412-19, Munich, 1998.

16- V.D. Agrawal, "An Information Theoretic Approach to Digital Testing", *IEEE T. Comp.*, V. C-30, pp.582-7, 1981.

17- A.L. Crouch, C. Pyron, "Impact of JTAG/1149.1 Testability on Reliability", *GOMAC* 1989, pp83-90.

18 - *IEEE ETW'98, Industrial Presentations (IP2, IP3) Compendium of Papers*, Barcelona, Spain, May 1998.

19 - F. Jong, F. Heyden, "Testing the Integrity of the Boundary Scan Test Infrastructure", IEEE, *Proc. of ITC*, 1991, pp106-12.

20 - L. Spainhower, TA Gregg, "G4: A Fault-Tolerant CMOS Mainframe", *IEEE FTCS-28 Digest of Papers*, pp 432-40, Munich, 1998.

21 - T.Yang, Z. Peng, "Incremental Testability Analysis for Partial Scan Selection and Design Transformations", *IEEE ETW'98 Comp. of Papers*, pp. 107-12, Barcelona, 1998.