Design for Test of Embedded Microprocessors *

Charles.O.Uagboe (s990420) Mercury Jair Gonzalez (s001069)

Abstract

Embeddeed cores are an ensential part in the realisation of complex System-On -Chip (SOC). Frequently the design of an SOC relies on the use of a microprocessor core designed as an Intellectual Property by some third party. In this paper we discuss some ways that this embedded microprocessor core can be tested particulaly dealing with design for test (DFT) considerations.

Test objectives and (DFT) strategies for an embedded SOC microprocessor will be discussed, we will present the use of IEEE 1149.1 boundary scan design in the design of an embedded core. And lastly we will discuss the concepts of structurally testable delay paths and functionally testable paths

1. Introduction

Due to the advances in the computer industry, we find now that the number of gates that can be put in a chip has grown exponentially. In this mordern age we can compact as much as 10+ million gate on a chip. This has invariably pushed us forward into the area of SOC. This field is the total encapsulation of a systen into a single chip. Thus in the near future we will have the whole electronic, computing, and mechanical blocks (through MEMS- microeletromechanical systems) of a full security system on one chip.

This futuristic eletronic utopia is not without its huddles. In this particular case we are concerned about testing an embedded microprocessor in a system such as the one dicussed above. As we know this miroprocessor might not be off-the-shelf and thus might not hold as much credibility as one. This is because it might be designed by a third party as an IP soft core. For this we have to know how to test these cores if we are working with one.

This paper is divided into 3 sections. Section 1 deals with a general over view of test,particularly its objectives. Also the design for test strategies is talked about for an embedded microprocessor such as the Motorola's ColdFire and the MiniMips-1600 soft core(a 16 bit RISC machine design by the authors in 49280/1). In Section 2 we will explain the use of IEEE 1149.1 boundary scan in the design of an embedded core taking the above named miroprocessors as an example. Section 3 takes us to the field of delay paths with respect to embedded cores. Here, we will go through the difficulties and complexity of testing path delay taking only testable paths. We will also discuss the different kind of testable delay paths and the concepts behind them.

2. Core-based system chip test design

In this chapter we will discuss a proposal for a standard for Embedded core test[5], we will see what are the main challenges and what could be the test strategies, where one of this strategies is to use Boundary scan as test access mechanism. Later we will present a particular case with the Motorola's core microprocessor coldFire. The use of large reusable building blocks namely cores, is a common practice nowadays, this practice has brought new test challenges. Compared to the traditional chip test approach the design process in core-based system chips is conceptually the same, but the manufacturing test process is very different. In core based design, the core is designed first by the core provider and then, the core user (system integration) include it into the design and elaborate a test strategie for the system, then the system is manufacturated and finally tested. Before, the processor designer was in charge of all the process, as shown in the figure 1.

There are 4 challenges clearly identified by core-based test[9], (1)Core internal test, (2) Core test knowledge transfer, (3) Test access to embedded cores, and (4) System Chip Test Integration and Optimization. To overcome this challenges,even though there are some old proposed strategies[6], the Wrapper is a Conceptual architecture is presented by the IEEE P1500 working group[5] that include a quite structurated view of the problem.

*This paper is handed in as an assignment answer in lecture course *49362 Test of Integrated Circuits (1998)* at the Department of Information Technology, Technical University of Denmark.

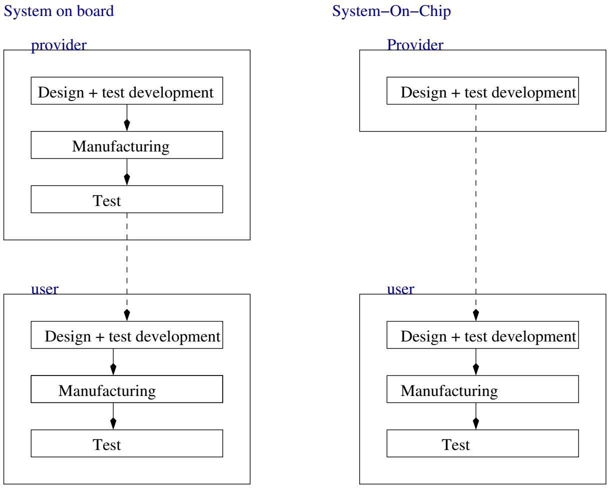

Figure 1. system design process

2.1 Challenges

1. **Core Internal Test.** The core user do not have the details of the core functionality (nor wants to have them), therefore, the core provider shall develop the core test, i.e., the DFT structures and patterns, and deliver them with the core.

2. **Knowledge Transfer.** Core distribution requires transfer information regarding the core from the provider to the user, this information should include description of the core-internal DFT, test modes and protocols, fault coverage, test pattern data, etc.

3. **Test Access to Embedded Cores.** A core is normally deeply embedded in a core-based system chip.Therefore, an electronic test access mechanism(TAM) is needed in order to link the test pattern source (either a ATE or a BIST) to the core inputs and then the core outputs to the test pattern sink.

4. **System Chip Test integration and Optimization.** The overall system test integrates the all reused cores, the used defined logic as well as the interconnection logic. this will bring us one of the major challenges, many trade-offs will arise between test quality, area, performance of the system, power consumption and test.

The IEEE P1500 working group, also called Standard for Embedded Core Test(SECT) has proposed a generic access architecture for testing embedded cores.

2.2 The SECT architecture

This section shows a SECT proposal architecture for testing embedded core-based system chips, it consists of 3 elements:

1. *Test pattern source and sink.* The test pattern source generates the stimuli for the embedded core, and the sink compares the response to the expected one. This source and sink can be by means either of an external *Automated Test Equipment* (ATE) or by an internal *Built-in-self-test*(BIST).

2. *Test Access Mechanism (TMA).* The test access mechanism transports test patterns.Either from the *source* to the core inputs or from the core outputs to the *Sink*.

3. *Core Test Wrapper.* It is the interface between the core and the environment, connecting the I/O the the rest of the chips and to the TMA.

The next figure shows the interaction of the elements of the proposed architecture.

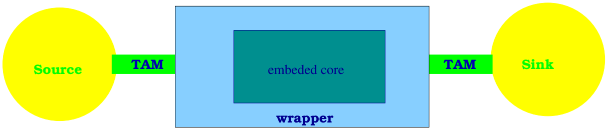

Figure 2. Overview of the elements in a embedded-core test approach.

There are several technique available to implement this elements, the next sections will deal with the description of this techniques, classified according to the three elements.

2.3 Source and Sink

The source generates the stimuli for the embedded core while the sink compares the results to the expected, We have mention that source and sink can be implemented as an Automatic Test Equipment,ATE or as Build-in-Self-Test,BIST, or as well as a combination of both. The choice for certain type of source or sink is determined by the following factors:

1. The type of the circuitry,(1) Logic, (2) Memory, (3) Analogue and mixed-signal.

2. The type of pre-defined test that come with the core.

3. Quality and cost considerations.

The actual choice for a *Source* or a *Sink* is very much determined by a quality-cost trade-off, while BIST techniques provide better performance related with fault coverage, it also increases the the silicon area (seems to be not to important nowadays) as well as power consumption (very important for some applications). Still, on-chip sources and sinks need some kind of ATE, for example for initialization, for report or for final signature checking.

2.4 Test Access Mechanism

The test access mechanism takes care of the **On-chip** test pattern transport, either form the *Source* to the core under test, or from the core to the test pattern sink.this section gives an overview of the several possible strategies for a test access mechanism.

Designing a test access mechanism involves making a trade-off between the transport capacity (*bandwidth*) of the mechanism and the test application cos it induces. The bandwidth is limited by the bandwidth of the *Source* and the *Sink*, as well as by the area needed to be spend on the test access mechanism itself.

The next options are offered when a TAM is being implemented:

- Reuse the existing functionality to transport test patterns.
- The TAM can either go through other core or modules or go around them.
- To have an independent TAM per core or to share it with some other cores.
- It can be a simple channel or contain certain intelligence test control functions.

There are many TAM already in use for testing embedded-core system-base chips in the industry[9], we can mention: Macro test, Transparency of cores, Reusing the system bus, Multiplexed access, A dedicated test bus, Boundary scan, Test scheduler and TestRail. Each one with its own pros and cons, but the favourite one among the companies[4],[3], is the boundary Scan approach(pure or with variations), we will dedicate a later section for it.

2.5 Core Test Wrapper

The core test wrapper forms the interface between the embedded core and its system chip environment, It connects the core terminal both to the rest of the IC, as well as to the test access mechanism. By definition it is implemented On-chip.

The core test wrapper shall have the next modes of operation:

- *Normal operation.* Namely non-test, the wrapper is transparent between the core and its system-IC environment.
- *Core Test mode.* The test access mechanism is connected to the core, such that test stimuli can be applied at the core's inputs and responses can be observed at the core's output.
- *Interconnect test mode.* Stimuli can is applied at the core's outputs and responses can be observed at the core's inputs.

The core test wrapper connects the core terminals to the test access mechanism. The width of the test access mechanism does not necessarily correspond to the number of core terminals. Then the width adaptation is done in the core test wrapper through serial-to-parallel conversion at the core inputs and parallel-to-serial conversion at the core outputs.

Pre-designed cores have own internal clock distribution system. Different cores have different clock propagation delays, which might result in clock skew for inter-core communication, see figure 3, and also corrupting the data transfer over the test access mechanism, especially if this mechanism is shared by multiple cores. The core test wrapper is the best place to have provisions for clock skew prevention in the test access paths between the cores.

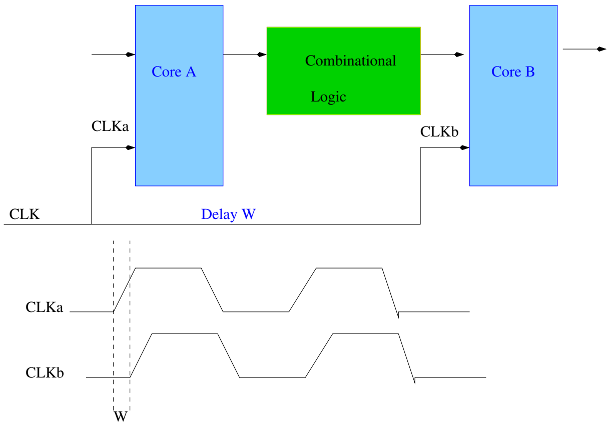

Figure 3. Skew effect.

3 Boundary Scan Test (BST)

The boundary scan approach was first developed for board-level interconnect testing and standardized as IEEE 1149.1, but it has been reused very successfully for testing embedded cores[4], obviously, to be already a standard is a big benefit. As many processor have included the BST already, is logical to reuse all the capabilities included (Instructions for test, debug, emulate, etc). Some drawbacks have been identified, first, a new scheme have to be defined for controlling several TAPs into the same chip, but worst, the BST defines a combined test control and test data access path of only one bit wide via TDI and TDO, no trade-off between bandwidth and test time is possible. Then for small test pattern sets, the small bandwidth is not a problem, but for large scan-testable cores, figure 4 , providing many scan test patterns via only one serial access wire results in a prohibitive long test times.

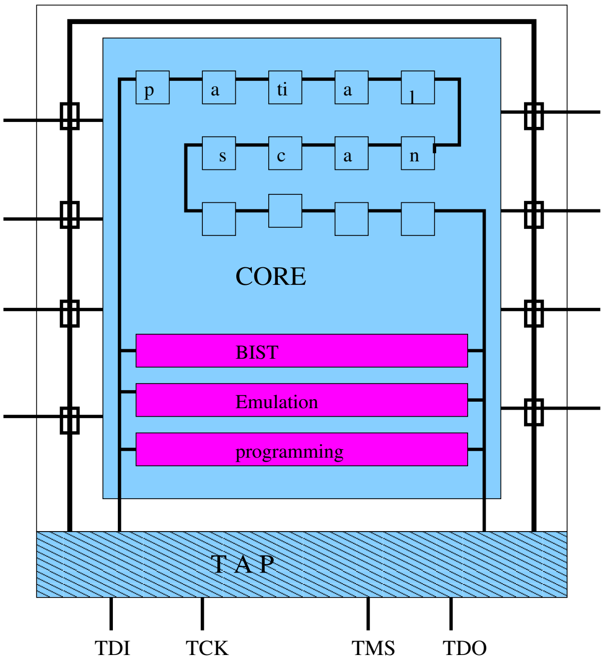

Figure 4. Boundary Scan can offer many options of how it can be configured..

4. Testable Delay Paths

As written in the above sections we can see now that one of the tet measures we have to under take when working with embedded cores is to test for path delays. This is to ensure that the design meets performance specifications. But testing path delays has its own set of problems, this is due again to the high rise in the number of chip and in the clock speed. On-chip clock speed increases dramatically while the tester's speed does not. An example is in [1], here we see that the clock frequency of the ColdFire is 90MHz while that of the tester is only 45MHz. This trend in development leads to an increase loss of good chips; due to the inability of the tester to met up witrh the chips demands.

One way of resolving this problem is using built-in self-test (BIST). BIST solution offers the ability to apply and analyze at-speed (in otherwords offline) test signals on chip yielding a greater accuracy than the tester. Also the circuit under test becomes its own test vector generator and test response detector. A further advantage of this concert is that on-chip tests can be done using the normal system clock(s) and at the full speed of the normal system. Existing BIST techniques belong to the class of structural BIST.

4.1. Structural BIST

Structural BIST is based on scanning techniques and it offers good test quality but requires dedicated test circuitry. This circuitry might include full scans, linear feedback shift registers(LFSR) for pattern generation, MISRs for data analysis and test control circuitry. All these leads to an increase in area and reduction in the performance of the system. Moreover, due to high-swithing random patterns generated by the pattern generators there is an incurrement in the power consumption of the system. Above all the design is quite complex to implement thus increasing the time to market margin.

In our particular case, where we want to discuss structurally testable delay paths, we will see that using a BIST for this test is possible thus:
A path delay is structurally testable if a vector pair can detect the fault using a given sensitization requirement, assuming that an arbitrary vector par can be applied to the circuit. In otherwords, if there exits a structural test for a path that can be applied through a scan-chain then we say that, that path is structurally testable.
Before we go further we are going to give a short example of the structural BIST used in a structural testable delay path. Here we take the ALU of the Parwan microprocessor [8] and describe how we can test for delay path fault in it. Generally the ALU of this core is 8-bit contains an accumulator (AC), and an instruction register (IR). Data inputs A7-A0 and B7-B0 of the ALU are connected to the internal data bus and AC, respectively. The control signals of the ALU are 3-bit wide S2-S0 used for arithmetic and logic operations.

The figure below gives us a general view of the datapath of the Parwan's ALU if this combinational circuit should be tested for path delay faults using the BIST scheme we will have to add more hardware.
The built-in logic block observation (BILBO) is one of the major architectures for BIST. The basic concept of BILBO test method is to partition the circuit under test into blocks of combinatiuonal logic which are each seperated by blocks of sequential logic element. The sequential blocks are rtefered to as BILBO registers, the output test signatures being observed in these block. A general schematic is illustrated below. In the normal mode each BILBO register forms part of the normal sequential logic for the working circuit, with

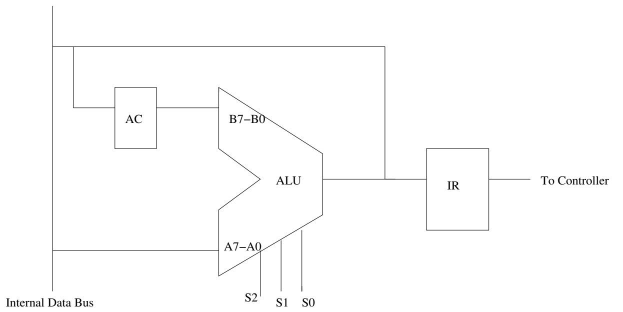

Figure 5. Datapath of Parwan

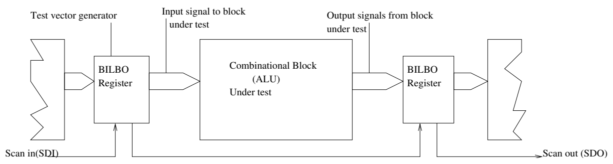

Figure 6. Basic BILBO Concept

the usual secondary inputs and outputs linking ther sequential and combinational logic. In test mode, a pair of BILBO registers,one preceeding and one following each block of combinational logic, is used to test the block, the first reconfigured to provide the input test vectors and the second reconfigured to provide a test output signature from the combinational logic. The test signature can then be read out in serial form for checking by a further reconfiguration of the BILBO registers into a sacn-path reconfiguration. Each BILBO register therefore has the following principal modes of operation:

1. Normal-mode operation
2. Reconfiguration into a scan-in/scan-out path
3. Reconfiguration into an output test-vector generator providing psuedorandom test vectors
4. Reconfiguration into an output test signature register

It can be seen now that the BILBO registers are quite complex circuits to provide this resource. Figure 3 shows the usual circuit arrangement. From the above schematic, it is quite obvious that the design for test using the approach is not only area inefficiect; it is also power consuming and difficult to implement. Imagine using this procedure to DFT for an ALU, below is a schematic [7] from page 357.This is a BIST structure in the ALU of a communications processor. Thus because of the above mentioned reasons, the structural test for delay path faults is not efficient and definately, when it comes to mobile embedded systems, the method we not yeild benefitable results.

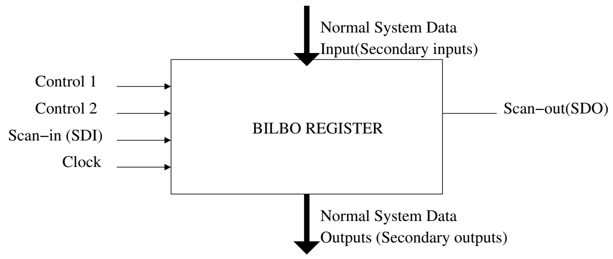

Figure 7. Block Schematic and Overall Circuit Arrangement

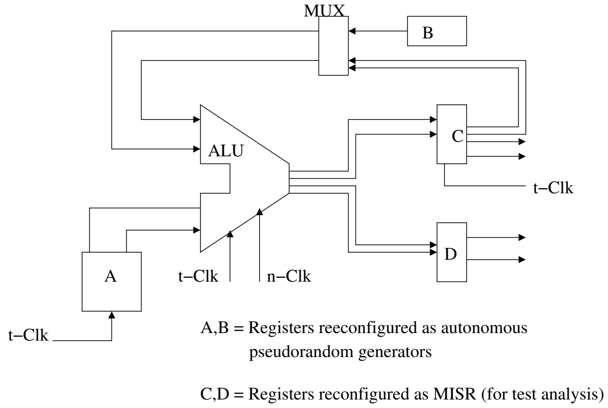

Figure 8. Stuctural Path Delay Testing of an ALU.

4.2. Classification of Path Delay Faults

The classification of path delay faults is illustrated below: Faults are classified into three categories [8]. They are:

1. Enhanced-scan struturally testable.
2. Full scan stucturally testable.
3. Functionally testable.

Note that functionally testable paths is a subset of full scan structurally testable; full scan structurally testable is a subset of enhanced san stucturally testable.
What we have discussed above is basically the full scan structurally delay path testing.Due to it's redundancies when it comes is to circuit optimizations we tend to look at other techniques.

4.3. Functionally Testable Paths

If a test can be applied in the normal operation of a circuit, it is called a functional test. A path is functionally testable

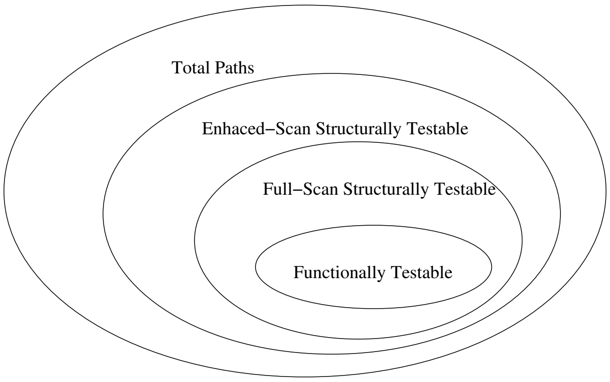

Figure 9. Classificaton of a Path Delay Fault.

if there exits a functional test for that path. Otherwise, the path is functionally untestable.
When delay testing an embedded processor using its functional vectors such as its instruction sequence, the input signals to the embedded block of the processor are determined by its functionality. This imposes temporal and spatial correlation among registers and flip fllops and therefore among inputs and outputs of the embedded core. These correlation results in a much smaller set of vector pairs applicable for delay testing. Also due to the irregular pattern of instruction code execution (due to loops,jumps e.t.c), after the apllication of certain vector pairs, some registers may not be enabled in the immediate following cycle, and thus delay fault effects propagated to them cannot be latched and will be lost. Therefore it is possible that some paths could be structurally testable but functionally untestable. A functionally untestable delay path can not be tested by any instuction sequence. Existence of dely faults on the functionally untestable paths will not cause any chip failure; unless the delay fault exits after 2 clock periods. And if this happens , it will most likely be captured by test for transition faults. Because of this we do not have to target functionally untestable faults for delay testing.
Below is an illustration of the Parwam datapath showing the functionally testable delay path and the functionally untestable delay path.
There are quite some advantages in identifying functionally untestable paths. Some of them are thus:

- it provides a more realistic fault coverage measure.
- it reduces the subsequent test generation effort, dealing with the functionally testable path only.
- it can guide the path selection process to avoid functionally untestable paths.
- it helps designers to identify true, functionally, testable, critical path.

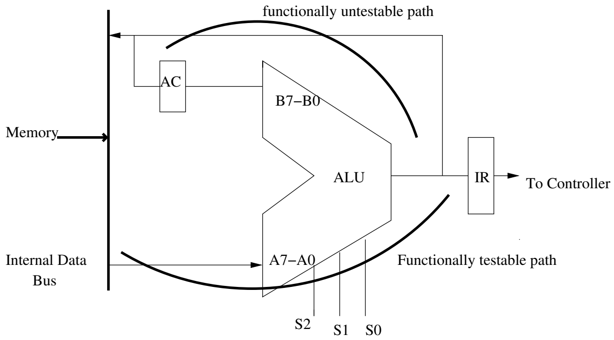

Figure 10. ALU Showing the 2 Delay Paths

- it can guide the synthesis process to generate an implementation of the circuit with a better performance.

The methodology proposed by Wei-Cheng Lai [8] identifies a tight superset of functionally testable paths in an embedded core. Here the set of constraints capturing correlations among input/output signals and registers/flip-flops of the datapath and the controller is extracted at first. Then after this is done they developed an efficient algorithm that identifies functionally testable paths.

4.3.1 Extracting the constraints

In this paper we are only going to consider the data path. This is because it is easy to partition the data path into a set of combinational blocks and registers. By analyzing the RTL description of the Parwam processor,and looking at the schematic of the ALU,all possible vector pairs applicable to the data path at speed is symbolically derived.An example is shown below

	DATA INPUTS				
ADDITION	A7–A0	B7–B0	OPCODES	IR	AC
Cycle1	V1a	V1b	100 (nop)		
Cycle2	V2a	V2b	101 (add)		Latch
Constraints	X...X	C...C	OZR		Care

Figure 11. Signal Flows to the ALU

From the table we can see that, if a NOP is executed before the ADD then in the first cycle the input to A7-A0 and B0-B7 are V1a and V1b respectively; and neither IR or AC is in the latching mode but when the ADD is executed, AC latches it's result. From this an output constraint is then extracted to show that AC is in the care mode (thus a care output).Now inputs to A7-A0 and B7-B0 changes to V2a and V2b respectively. But V2b=V1b, this is because no transitions can be created at B7-B0 (due to the accumulator register holding the previuos value of B7-B0). Therefore, these paths are functionally untestable for this particular

sequence.
The constraints on B7-B0 is denoted as C (which implies unknown but stable); those on A7-A0 are not constrained since from the instruction sequence they are assigned arbitrary vectors.The constrains at the opcodes (or control vector) are represented as O, Z, R,and F denoting transitions 11, 00, 01, and 10 respectively.

4.3.2 Classification algorithm

The classification algorithm we are going to talk about is based on that proposed by Cheng [2]. Note that this algorithm target functionally testable path delaty faults for data paths only, we are not going to talk aboutr the controller since this is mostly sequential in nature and thus more difficult to implement.
Given a set of constraints derived as we have explained above,as well as the gate-level implementations if we can find a set of value that do not violate any sensitization requirement or input/output constraints it is then classified as functionally testable otherwise it is functionally untestable.Then it proceed thus:
A wire is associated with a flag, called the c-flag, and this flag indicates whethere the signals on the wire are stable (00, or 01).

- Step 1. Initially,the set of functionally testable faults is empty.
- Step 2. Read an unprocessed constraint from the set of constraints.
- Step 3. Find the corresponding constrained value for each input. If the constraint is an O, Z, C set flag to true otherwise set to false.Then assign the second vector of the vector pair e.g. assign for constraint R (01) the value 1.
- Step 4. Imply the second vector and the c-flag value. Note that the c-flag of a gate output is true iff all the c-flag of all the inputs are true, thus if any of the gate input flag is false then c-flag of the gate output is false.
- Step 5. All the paths are examined in a depth-first fashion.Initially a partial path consist of an input then extended gate by gate. If any gate is included into this partial path, it checks if the c-flag output of this gate is true. If true then the partial path is unsensitized and the algorithm backtracks,and go to the next partial path. Else the c-flag is false,mandatory assignment at the second vector on all the off-input to the gate are set and implied again. If any conflict occurs, the algorithm backtracks. When we find that the path terminates at a care output, we add it (that is the path) to the functionally testable set.
- Step 6. If there any unprocessed constraints, return to step 2. Otherwise,return the set of functionally testable paths.

5 Conclusion

The new trend of using pre-designed cores have brought many benefits, due that individual design effort when reusing the cores. This has also made available the fast design for SoC that are taken over nowadays. But the flexibility is not for free, and one of the costs is the increase of challenges for testing the cores and the overall SoC. Since the beginning of the trend, various guidelines have been given on how to approach the problem of testing this chips, but the biggest effort on this matter is the proposal of the SECT working group for a standard of Test of embedded cores. We have described the IEEE 1500p SECT proposal that present solutions for actual challenges; If this proposal is taken, it will improve the inter-operability between cores and sub-systems of different providers and then help the industry to go better.
.
We have discussed the concept of structurally testable delay paths and functionally testable paths.
Due to the complexity of embedded core and their low controlability and observability we see that the traditional ways or modes of test can not be applied just like that to these system. We have discussed ways of by which test could be imposed especially for the test of delay paths faults.
We have also discussed the relationship between stucturally testable delay paths (which can be done using the BIST mechanisms); and the functionally testable path which on the other hand uses the instuction set of the core's architecture.
Also we have illustrated how not all the functional paths need to be taken in regards because from our discussion we have seen that some of these paths will not affect the operation or performance of the circuit (functionally untestable). So we do not need to test for them. Thus the identification of the testable paths will be a big plus for us since it save a lot of resources and at the same time having a high fault coverage.

References

[1] T. L. M. Alfred L. Croch, Michael Mateja. Test devekopment for a third-version coldfire microprocessor. *IEEE Design and Test of Computers*, pages 29–37, 2000.

[2] K. Cheng and H.-C. Cheng. Classification and identification of nonrobust untestable path delay faults. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 845–853, 1996.

[3] R. R. Collings. Bounbary-scan logic (fully complaint with ieee std 1149.1(jtag)). 2001.
[4] R. R. Collings. Overview of pentium probe mode. 2001.
[5] S. D. Erik Jan Marinissen, Yervant Zorian. Towards a standard for embedded core test-an example. *ITC INTERNATIONAL TEST CONFERENCE*, pages 616–627, 1999.
[6] S. L. Hurst. *VLSI TESTING (digital and mixed analogue/digital techniques)*. IEE London United Kingdom, 1998.
[7] S. L. Hurst. *VLSI TESTING (digital and mixed analogue/digital techniques)*. IEE London,United Kingdom, 1998.
[8] K.-T. C. Wei-Cheng, Angela Krstic. Functionally testable path delay faults on a microprocessor. *IEEE Design and Test of Computers*, pages 6–14, 2000.
[9] S. D. Yervant Zorian, Erik Jan Marinessen. Testing embedded-core based chips. 1999.