The Essentials of Testing Digital Circuits

Ngene C.U., *Member, IEEE*

Abstract – Testing is an important part of digital devices development life cycle and it takes about 70% of time to market. This paper discusses the various testing concepts as it relates to digital design and how it impacts the reliability of the final product. We also show that making designs testable by using appropriate design for testability techniques considerably reduces testing time and ensures a fine-grained diagnosis of finished product. A three bit counter circuit was used to illustrate the benefits of design for testability by using scan chain methodology.

Index Terms – Reliability, design for testability, faults, defect level.

I. Introduction

The reliability of electronic system used to be the concern of the military, aerospace and banking industries. But today applications such as computers, consumer electronics, telecommunication and automotive industries have joined the league of applications that demands reliability and testing techniques because they are everywhere and their feature sizes have become less and less as the years go by. In addition, their proliferation has led to the tendency of their misuse. An important aspect of reliability is the system's ability to run independently on demand. This requires that the system be fault tolerant.

Poor quality products require more maintenance and repairs which leads to huge expenses on staff and mileage to get staff and spares to outdoor locations [4]. It also affects the manufacturer's image and costs on returned parts and systems.

The three basic engineering activities are design, manufacture and test. Currently testing activities are also carried out at the design stage. This means that testing process is integral to both design and manufacturing activities and cannot be seen as a standalone activity. These activities are done as quick as possible and economically too. Because we want to save time and cost, we should endeavour to ensure that the quality of the would-be product is not compromised. Even while a product is in use testing can also be carried out either as a normal routine service arrangement or to eliminate faults as they occur.

A good quality product must meet the purpose for which it was designed and produced. In addition it must be very reliable meaning that the device should be operational most of the times and rarely fails. The reliability of Digital devices is high. But this reliability can be undermined if the operational conditions are not adhered to. Conditions such as operating temperature, power supply voltages and frequencies, electromagnetic influences and handling can negatively affect the reliability of digital devices. If the room temperature is higher or lower than the recommended for example, the device may over heat and probably damage some of the components which may render the device inoperable.

If we can guarantee 98% fault free circuit at the design and implementation stages, we may not be able to say what happens after packaging and when the component is finally mounted on a board and delivered to the consumer. It is important to note that ICs at the end of the day find there ways onto a circuit board. Even Systems on chip (SoC) end up on a board. While on the board we have to boarder about how well the pins of the various ICs mounted on the board are connected or whether the right IC is in the right position.

Testing encompasses design verification and diagnosis (fault location for purposes of effecting repairs). There are two aspects to test. One is testing the design, or carrying out design verification to make sure the design is correct and conforms to requirements. Design verification also lets you know where you are in the development cycle and how stable the design is [1]. The other aspect of test is testing for physical failures, making sure nothing is been broken and there's no defect from manufacturing. A significant portion of our development cycle time is spent on testing the product design, and that's becoming extremely expensive.

The beauty of integrated design and manufacturing is that it cuts product cycle time, but successful integration hinges on the quality of the design data passed to manufacturing. This paper focuses on the fundamentals of testing at the design stage. The remaining parts of this paper were divided into sections. In section 2 the challenges of product quality will be discussed. Section 3 briefly discusses the design flows with integrated testing. In section 4, this paper reviews faults and test pattern generation, whereas section 5 x-rays ways of making designs testable. A simple example to illustrate the design for testability technique using scan chain methodology was presented in section 6.

II. Testing Challenges

Quality improvement starts at the design stages. Testing starts right from the system level through RTL coding to

Manuscript received February 14, 2012

Ngene C.U. is with the Department of Computer Engineering, University of Maiduguri, Nigeria. He received the MSc. Degree in computer engineering from the Kiev Institute of Civil Aviation Engineers and Ph.D. degree in computer engineering from Kharkov National University of Radioelectronics in 2011 (e-mail: umerahlove@yahoo.co.uk).

fabrication and use of the device in the field. Currently assertions are embedded in codes to help for quick localisation of functional violations during simulation. It is a standard in electronics industry to test chips before they are mounted on a board, test the board before system assembly and finally test the system. This is essentially so because of the rule of ten. If a chip fault is not caught by chip testing, finding the fault costs 10 times as much at the PCB level as at the chip level. Similarly if a board fault is not caught by PCB testing, finding the fault costs 10 times as much at the system level as at the board level. This means that a fault that is not caught at the chip level will now cost 100 times as much at the system level.

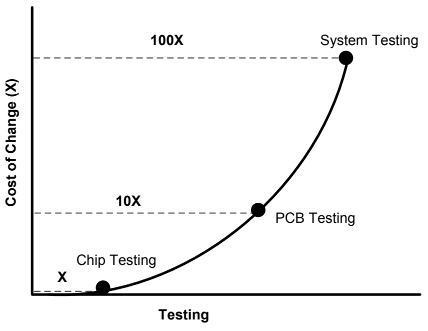

Fig. 1. The Rule of Ten

Some engineers are suggesting that rule of twenty be adopted considering the complex nature of present day ICs. The rule of ten is illustrated in figure 1. Very real costs are associated with inattention to design quality. If errors or omissions in design data are not addressed early, more costly changes are required later in the product development process.

Another development is the synthesis for different objectives. Early synthesis was aimed at decreasing area and delay. More recently, other objectives have come into play, such as power, noise, thermal control, verifiability, manufacturability, variability, and reliability. Consequently, additional criteria will emerge as new technologies develop, and new models and optimization techniques will be needed to address such requirements [12].

Concept of reliability

Reliability is the probability of no failure within a given operating period. For example, if 50 systems operate for 1,000 hours on test and two fail, then we would say the probability of failure, P_f, for this system in 1,000 hours of operation is 2/ 50 or P_f (1,000) c 0.04. Clearly the probability of success, P_s, which is known as the reliability, R, is given by $R(1{,}000)=P_s(1{,}000) = 1-P_f(1{,}000)=48/50=0.96$.

One can also deal with a failure rate, f_r, for the same system that, in the simplest case, would be f_r = *2 failures/ (50 × 1,000)* operating hours — that is, $f_r = 4 \times 10^{-5}$ or, as it is sometimes stated, $f_r = z = 40$ failures per million operating hours, where z is often called the hazard function. If failure rate z is a constant (one generally uses λ to represent a constant failure rate), the reliability function can be shown as in (1).

$$R(t) = e^{-\lambda t} \quad (1)$$

The mean time between failures (MTBF):

$$\text{MBTF}= \int_0^\infty e^{-\lambda t} dt = \frac{1}{\lambda} \quad (2)$$

The repair time (Rep) is also assumed to obey an exponential distribution and is given by.

$$\operatorname{Re} p(P > t) = e^{-\mu t}. \quad (3)$$

The mean time to repair (MTTR):

$$MTTR = \frac{1}{\mu}, \quad (4)$$

Where, μ is the repair rate. The system availability (failure-free) is the fraction of time the system is operating normally and is given by:

$$\text{System Availability}= \frac{MTBF}{MTBF + MTTR} \quad (5)$$

With the above expression for reliability it becomes evident that the more complex a system is the less is its reliability. For instance if a system board contains n number of components and each component has a reliability of R_c, the reliability of the board (R_{sb}) over time t period of operation without failure is:

$$R_{sb} = [R_c(t)]^n = [e^{-\lambda t}]^n = e^{-n\lambda t} \quad (6)$$

It is therefore clear that the system reliability is very small not minding the fact that the reliability of individual component is high and will reduce further if the reliability of the interconnections were taken into consideration.

The graphical representation of failure rate Z(t) as a function of time can be illustrated by the popular bathtub curve shown in figure 2.

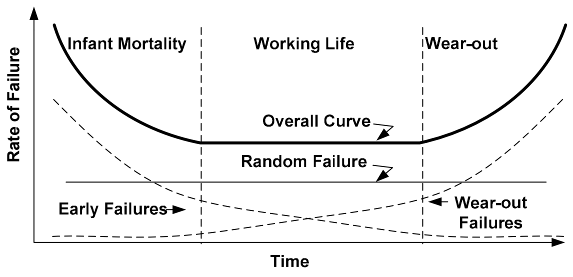

Fig. 2 Failure rate curve

The infant mortality region on the graph depicts failures that are attributed to poor quality as a result of variations in the production process technology. The region on the graph termed “Working life” shows that the failure rate is constant *(Z(t)=λ)*. This is the working life of the component or system and fault occurrence here is at random. The wear out region marks the end-of-life period of a product. For electronic products it is assumed that this period is less important because they will not enter this region due to a shorter economic lifetime as a result of technology advances and obsolescence. It is important to note here that all ICs must be shipped after they have passed infant mortality test periods in order to reduce field failure and subsequent repairs.

III. Design Flows

The design of VLSI follows certain procedure, evolving from the highest level of abstraction down to implementation - Design Specification, HDL Capture, RTL Simulation & Functional Verification, RTL Synthesis, Functional Gate Simulation, Place and Route and Post Layout Timing Simulation.

Every design starts with specification capture. We must determine the functionality of the new design at the onset. Wrong conception at this level could lead to a lot of problems such as poor quality product or overlooked functionality. An idea of what is to be designed is converted into formal document called design specification. In some cases one or more specification documents are created, depending on whether we are creating a component or a system. Design specification is a written statement of functionality, timing, area, power, testability, fault coverage, etc. The following methods are used to specify the functionality – state transition graphs, timing charts, algorithmic state machines and hardware description languages (VHDL and Verilog). Lately the need to capture designs at the highest level of abstraction in what is called Electronic System Level (ESL) using SystemC, System Verilog, etc. is being integrated and pursued vigorously. The specification is then captured using HDL in form of behavioural description. The HDL model of the design is simulated in order to determine functional compliance and to expose any design or coding errors. In order to achieve this, a test plan is developed. This involves writing a test bench for the model and applying appropriate test vectors to verify the design. If the functionality has been verified, then the model is synthesised using appropriate synthesis tools. The objective of synthesis is to produce a netlist (list of modules and their interconnection at the register transfer level stage or at the gate level) of the design for the target technology. Synthesising the design involves optimisation of Boolean functions (minimise logic, reduce area, reduce delay, reduce power, balance speed versus other resources consumed). After the RTL/gate level synthesis, the design is further simulated to determine that the gates used functions properly and meets the overall functionality. If this is achieved then we move on to the placement and routing stage where selected cells are placed on the target technology (CPLD, FPGA or ASIC) and connected in accordance with the netlist. After the placement and routing have been completed the need to further simulate the design arises. In this case we simulate to determine whether the timing (timing back-annotation), speed, physical and electrical specifications have been met. This simulation includes test vector generation to test for inherent fabrication flaws. It is important to note that the design should be correct at this stage, because this is the last stage before the design is signed off for fabrication. You can see that testing is carried out virtually at all of the stages of the design flow. This is important because the earlier an error is detected the better and of course the cheaper.

Verification and Testing occur at different levels of product development. Design verification is a set of activities that is carried out on a circuit before the circuit is implemented physically. These activities are geared toward ensuring that the circuit under design meets its functional and timing specifications. Mapping a design from one phase to another may cause some errors to occur. These errors may be as a result of improper handling of the EDA tools and they must be removed before the next phase. You see that at each stage the design is verified to assert that it is the same design from the previous stage and that it meets the specification. Currently simulation is the most efficient method of design verification. We simulate for functional and timing compliance. Assertion-based verification is gradually gaining in popularity amongst design and verification engineers.

Testing on the other hand is a set of activities designed to ensure that a circuit that has been manufactured complies with the parametric (voltage, resistance, current, capacitance, etc), timing and functional specifications of the design. In other words testing demonstrates that the manufactured IC is error free. Digital testing is performed on the manufactured IC using test patterns that are generated to demonstrate that the product is fault-free. It is important to note that at the logic gate level automatic test pattern generation (ATPG) is used to generate the test patterns and are verified using fault simulators. At higher levels of abstraction (RTL and behavioural) testability measures are used instead.

Rapidly evolving submicron technology and design automation has enabled the design of electronic systems with millions of gates integrated on a single silicon die, capable of delivering gigaflops of computational power. At the same time, increasing complexity and time to market pressures are forcing designers to adopt design methodologies with shorter ASIC design cycles. With the emergence of system-on-chip (SoC) concept, traditional design and test methodologies are hitting the wall of complexity and capacity. Conventional design flows are unable to handle large designs made up of different types of blocks such as customized blocks, pre-designed cores, embedded arrays, and random logic. A key requirement for obtaining reliable electronic systems is the ability to determine that the systems are error-free [6]. Electronic systems consist of Hardware and Software. In this paper we shall be looking at hardware testability issues. What is a system? Semiconductor components are not thought of as systems. A system is a collection of components that forms a complete item that one can procure to do a specific task or function. A system also includes a hierarchy of other systems, which we call subsystems, each of which is a system in its own right. In [1] Hal Carter opined that the basic philosophy is that systems grow as large as our technology will permit and testing complexity also grows. If you take n units and combine them such that they all interact, you'll get $n(n-1)/2$ interconnections, which is a

n^2 product of the communication complexity between the units. If you can decompose that, you can get down to $\log n$ complexity for the number of units actually being diagnosed or tested. Design-for-test and self-test must therefore be involved with components at as many levels as possible. Then system-level testing can actually aggregate those lower level tests in a more streamlined way as they migrate towards the system as a whole [1].

IV. Review of Faults And Test Pattern Generation

With the present deep sub-micron technology which is currently at 20nm [7] ensuring high product reliability has become more daunting. The more transistors/gates we squeeze into a small area of a chip the greater the risk of over heating, crosstalk between interconnections and the more likely the chip is subjected to failure. This has not been the case because of the enormous effort the design and verification engineers spent in testing the would-be IC. The would-be chip is subjected to rigorous testing to expose any fault in terms of functional compliance and power violations. Apart from design errors, faults also result from manufacturing process. Testing continues right after the IC is mounted on a board – system test.

A. Fault types and fault models

A digital circuit whose implementation is different from its intended design is said to be defective. And if the output of the circuit is wrong because of the defect we say an error is observed. When we talk about defects from a higher level of abstraction in terms of circuit function, we refer to them as faults. One is talking about the imperfections in the hardware whereas error refers to the imperfections in the functionality of the hardware. An IC may become faulty not only as a result of incorrect design or manufacturing procedure but also as a result of external influence (electromagnetic influence), mechanical rupture, wear and tear. Hard failures (permanent failures) are usually caused by breaks due to mechanical rupture or incorrect design/manufacturing procedure. Soft failures are transient or intermittent. These are induced by supply fluctuations or radiation. Intermittent failures are caused by the degradation of component parameters.

Faults play a great role in helping test engineers detect defects in ICs. In another word we can say that faults are models that help us to understand physical defects. A fault model is a representation of the effects of defects on chip behaviours. A fault model may be described at logic, circuit, or physical levels of abstraction. Examples of fault models include stuck-at faults, bridging faults, stuck-open faults, and path delay faults [13]. Several defects can be mapped to a single fault model. Some defects may also be represented by more than one fault model. In view of the fact that faults are models, they may not really be a perfect representation of the defects, but are useful for detecting the defects. There are so many fault models for representing defects at behavioural, functional or structural levels. The most commonly used fault model at the structural level is single stuck at fault (SSA). This is a situation whereby a line in a circuit is permanently at logic 1 or 0 levels. So we say that a line has a fault stuck-at-1 or stuck-at-0. Though SSA fault has been used widely for defects representation, it has become increasingly imperative to use other models especially with the current complexity of digital circuits. Examples of SSA include a short between ground (s-a-0) or voltage (s-a-1) and a signal; an open on a unidirectional signal line; any internal fault in the component driving its output that it keeps a constant value.

B. Fault Simulation

Fault simulation consists of simulating a circuit in the presence of faults. Comparing the fault simulation results with those of the fault-free simulation of the same circuit simulated with the same applied test, we can determine the faults detected by that test. Faults are simulated in order to achieve the following:

- To evaluate the quality of a test set (i.e. to compute its fault coverage.
- Reduce the time of test pattern generation. A pattern usually detects multiple faults and fault simulation is used to compute the faults accidentally detected by a particular pattern.
- To generate fault dictionary. This is necessary for post test diagnosis.
- To analyze the reliability of a circuit.

C. An example of fault detection and test pattern Generation

In order to illustrate how SSA fault model can be used to detect defects and possibly use the patterns to locate them we shall use a simple 2-input XOR gate figure 3. Table 1 shows the function of an XOR gate under various conditions. Column 2 of the table shows the normal response for fault free nodes, whereas columns 3 upwards show faulty responses of the gate under faulty conditions. A fault is said to have occurred when the circuit's normal response is different from the faulty response for the same set of input combinations i.e. $F \neq F_f$ .This can also be expressed as follows: $F \oplus F_f = 1$.

With the above expression in mind and a closer look at the table indicates that faults are not always observable. For instance, with lines A/0 for input combinations 00 and 01, $F = F_f$. The only time the fault free response differs from the faulty response was when the input combinations AB=10 and AB=11 were applied on the circuit. These input combinations can be considered as the test pattern that detects line A stuck-at-0. Because the two patterns detect

A/0 either AB=10 or AB=11 can be chosen as the test pattern. Let us now consider faults that are detected by specific input combinations.

AB=	00	detects	A/1, B/1 and F/1
	01	detects	A/1, B/0 and F/0
	10	detects	A/0, B/1 and F/0
	11	detects	A/0, B/0 and F/1

From the above we can see that the same input combination detects more than one fault. The first test pattern from the above is AB=00 which covers faults A/1, B/1 and F/1. The next pattern is 01 which detects A/1, B/0 and F/0. With these two patterns we have detected five faults namely A/1, B/0, B/1, F/0 and F/1. We are left with one fault i.e. A/0 to be detected. Any of the patterns AB=10 or AB=11 detects this fault. The set of test vectors that will detect all SSA faults for a 2-input XOR gate are: 00, 01 and 11. This means that if want to test a 2-input XOR fig. 3.2 gate it is sufficient to apply all three of these patterns on the inputs of the gate. The fault coverage in this case is 100%. It is important to observe that this example is a trivial one indeed and oversimplification of testing and test pattern generation procedure.

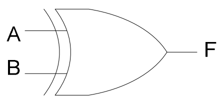

Fig. 3 2-input XOR Gate

TABLE I
XOR GATE RESPONSES UNDER VARIOUS CONDITIONS

Inputs	Fault Free Response	Faulty Response					
A B	F	A/0	B/0	F/0	A/1	B/1	F/1
				F_f			
0 0	0	0	0	0	1	1	1
0 1	1	1	0	0	0	1	1
1 0	1	0	1	0	1	0	1
1 1	0	1	1	0	0	0	1

In practice it is a more daunting task as we have to deal with circuits with millions of gates and different interconnection structures. For example, if we have a tester that is capable of applying test pattern every 100ns, then we can calculate the test time as shown in table 2.

TABLE 2
EXHAUSTIVE TESTING TIME

No. of Inputs	No of tests required for exhaustive testing	Test time
10	2^{10}	102 µSec
20	2^{20}	0.1 Sec
40	2^{40}	30.5 Hours
60	2^{60}	3656 Years

The computational complexity of exhaustive testing is in the order of 2^n. It can be seen in table 2 that the number of tests quickly gets out of hand as the number of inputs increases and therefore it is only of use where there are a very small number of inputs. This testing strategy is also very inefficient since most of the test patterns are actually redundant.

Test pattern generation for sequential circuits is very tedious and less straightforward than for combinational circuits. There are many techniques for test pattern generation, but their discussion is beyond the scope of this paper.

D. *Test quality components*

Fault coverage (7) is a measure employed generally to determine the quality of tests. It is expressed as a ratio of faults detected (covered) by the test pattern to the total number of faults possible for the given fault model. Because of the difficulty in testing ICs exhaustively some of the faulty ones may escape detection leading to yield and defect level problems. Process yield (8) is a fraction of the manufactured ICs that is defect-free. The process yield is approximated by the ratio of the good ICs to the total number of ICs. Process variations, such as impurities in wafer material and chemicals, dust particles on masks or in the projection system, mask misalignment; incorrect temperature control, etc. affect the process yield. It suffices to note that testing cannot improve process yield. However, process diagnosis and correction can improve process yield. This method involves the location of defects in the failed parts and tracing them to specific causes, which may be defective material, faulty machines, incorrect human procedures, etc. Once the cause is eliminated, the yield improves.

When some of the faults escape detection for some components or parts the defect level increases. Defect level (9) is the fraction of faulty chips among the chips that pass the test, expressed as parts per million (*ppm.*). A defect level of 100 PPM or lower represents high quality. This means that among the so-called good parts or ICs there are bad ones. It is well known fact that the quality is a function of user's satisfaction. To a user the highest quality product is one that meets requirements at the lowest possible cost. Testing (functional) checks to ensure that final product conforms to its requirements and the reduction of cost is achieved by enhancing the process yield. The relationships between fault coverage (FC), yield (Y) and defect level (DL) are as shown in the expressions below:

$$FC = m/n \tag{7}$$

$$Y = (1-p)^n \tag{8}$$

$$DL = 1 - Y^{(1-FC)} \tag{9}$$

Where: *n* is the total number of faults, *m* is the number of detected faults $m \le n$, *p* is the probability of any fault occurring.

The following assumptions were made.

1. Stuck-at-fault model is assumed,

2. The probability (p) of any fault occurring is independent of the occurrence of any other fault. That is to say that the faults are mutually exclusive.

For more detailed information on how they were derived please refer to page 15 of [11]. With 100% fault coverage as in the example 4.3 the defect level is 0, meaning that none of the components that passed the test is defective. If the coverage is less than 100% it then means that some faults may still exist.

V. Making Designs Testable

Testing is an expensive activity in terms of generating the test vectors and their application to the digital circuit under test. Because of the complexity of testing processes, design for testability (DTF) approaches was developed. The DTF approach is aimed at making digital circuits more easily testable such that these circuits are more controllable and observable by embedding test constructs into the design. There is no formal definition for testability. An interesting attempt was given in [9] as: “A digital IC is testable if test patterns can be generated, applied, and evaluated in such a way as to satisfy predefined levels of performance (e.g., detection, location, application) within a predefined cost budget and time scale”. One of the key words is “cost.” It is probably the cost of testing that deters semiconductor manufacturers from doing as much testing as is really needed to ensure reliable products [10].

There are many facets to this cost, such as the cost of:

1. Test pattern generation (automatic and/or manual) time. Test pattern generation is an NP-complete problem since it is difficult to find a polynomial solution.
2. Fault simulations and generation of fault location information,
3. Test equipment (Automatic Test Equipment).
4. Test application which includes the process of accessing appropriate circuit lines, pads or pins, followed by application of test vectors and comparison of the captured responses with those expected; time required for detecting and/or isolating a fault.
5. Undetectable faults; unpredictable production schedules and an uncertain level of product quality delivered to the customer. When many actual faults are not detected by the derived tests, it is often reflected in terms of loss of customers.

The cost associated with undetected fault could be high, see figure 1, but sometimes difficult to quantify. Although this fault is difficult to quantify, it influences the other costs by imposing high fault coverage requirement to ensure that fault escape is kept below an acceptable threshold [11].

In view of the fact that these costs can be exorbitant and in most cases exceed design costs, it is therefore, necessary to keep them within acceptable limit. And this is the reason why design for testability has become imperative. It is a proven way of reducing testing costs. A fault is testable if there is a well-specified procedure to expose it, which can be implemented with a reasonable cost using current technologies. And a circuit is testable with respect to a fault set when each and every fault in this set is testable. As there is price for everything in this world, DFT carries its own penalty - silicon real estate and performance penalties. This is mainly because of the extra circuitry employed for implementing the DFT.

Testability, on the other hand, is introduced at the design stage, where it dramatically lowers the cost of test and the time spent at test. Properly managed, testability heightens your assurance of product quality and smoothes production scheduling.

A. DFT at the Design stage

Modern design approach has brought test engineering closer to the design activities in that the test program development for an electronic circuit occurs at an early stage in the product development process and requires a basis in design. This overcomes the problems encountered when design and test activities were separate and distinct, an unnecessary barrier between two interrelated activities. In this DFT approach, test activities can influence how a design is created by identifying testability issues and improving test access to specific circuitry within the design. Specialist engineers in both design and testing are supported by a generalist DFT engineer, shown in Figure 4 who bridges the gap between them. The need for specialists is based on the need for in-depth knowledge of specific design and test issues, roles which a single person could not realistically be expected to undertake. [5]

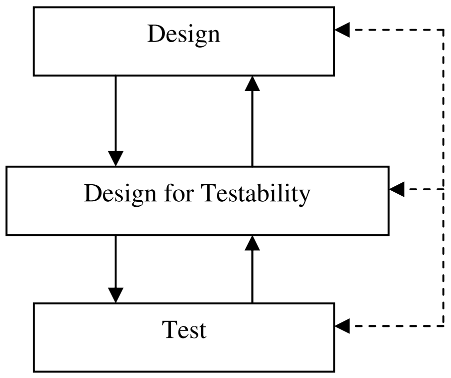

Fig. 4. Integrated designs for testability

B. DFT Methodology

There are several methods of making designs testable. None of these methodologies can solve all VLSI testing problems nor can a single technique guarantee effectiveness of testing for all kinds of circuits. Generally DFT techniques have the capability to increase the circuit real estate on chip which results in complexity of logic circuits. Increased complexity leads to increase in power consumption and decrease in yield. With all these challenges in mind, there is need to select a technique for a particular kind of circuit that balances these trade-offs (benefits and challenges). If a circuit is modified to increase its testability by the addition of extra circuitry, it therefore means that another mode of operation apart from the normal mode has been included. This new mode of operation is called test mode. In this mode the circuit is configured for testing alone. DFT methods include the following: Ad-hoc methods; Scan, full and partial; Boundary scan; Built-In Self-Test (BIST).

The goal of DFT is to increase controllability, observability and/or predictability of a circuit. The DFT discipline started with the ad-hoc technique which involves the insertion of test points, counters/shift registers, partitioning of large circuits, logical redundancy and breaking of global feedback paths. Many of these ad-hoc techniques were developed for printed circuit boards and some are applicable to IC design. These methods referred to as ad hoc (rather than algorithmic) because they do not deal with a total design methodology that ensures ease of test generation, and they can be used at the designer's option where applicable. The detailed description of these techniques can be found in [8].

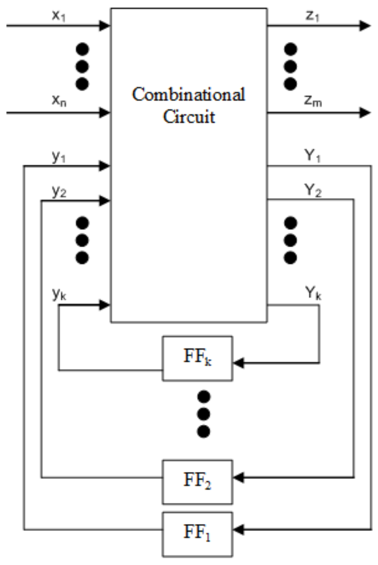

Fig. 5. General Model of FSM

Scan path is a scheme that facilitates the testing of finite state machines (sequential circuits). Automatic test pattern generation for sequential circuits is very tedious and in most cases do not achieve the required test coverage. This arduous task is as a result of the difficulty in controlling and observing the inputs and output states of the flip flops respectively. In this technique the flip flops (FF) or latches are designed and structured in such a way that allows the circuit to be operated in either of the two modes (normal or scan). Figure 5 shows the structure of the FFs when the circuit is operated in the normal mode. In the test or scan mode, all the FFs are disconnected and reconfigured as one or more shift registers called scan chains or scan registers. In the test mode all the state inputs (y_1, y_2,… y_k) become pseudo-primary inputs to the circuit. The state inputs to the combinational circuit are the present states of the FFs and the state outputs of the combinational circuit (Y_1, Y_2, …,Y_k) are the next states of the FFs. When developing tests for the FSM we assume we have only combinational circuit with the following inputs: x_1, x_2,…,x_n and y_1, y_2,… y_k; and outputs: z_1, z_2,…z_m and Y_1, Y_2, …,Y_k.

During test application, the FFs are initialised to put them in a known state. After initialisation the test patterns are applied to the primary inputs of the circuit, the results are latched at FFs and they are propagated to the output by placing the circuit in the test mode and clocking enough times to capture the results. This configuration makes the pseudo primary inputs as control inputs and the input (pseudo outputs) to a FF an observation point. To switch between normal operation and shift modes, each flip-flop needs additional circuitry to perform the switch

Boundary scan method was developed primarily for the testing of circuit boards and is defined by the core reference IEEE standard 1149.1-2001 "Test Access Port and Boundary-Scan Architecture". The idea to bring back the access to device pins by means of an internal serial shift register around the boundary of the device is accredited to European test engineers under the aegis JETAG (Joint European Test Action Group). When North American test engineers joined the group was named JTAG (Joint Test Action Group). It was this group that converted the ideas into an International standard, the IEEE 1149.1-1990 Standard first published in April 1990. The ICs that are compliant to this standard must incorporate extra hardware (Shift-Registers – Boundary scan registers) to facilitate communication between them and the board during testing. This idea is illustrated in figure 6.

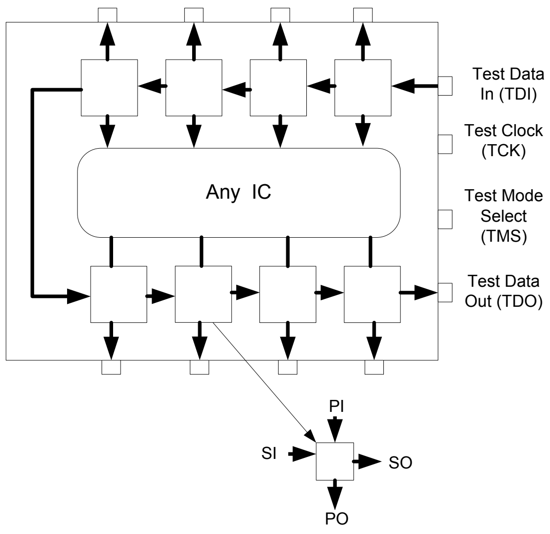

Fig. 6. Generic Boundary Scan Architecture

It is important to note at this point that the use of boundary scan has found their ways in internal testing and running of BIST. Apart from BISTs boundary scan is very useful in testing System on chips (SoC) in a new testing environment that enable systems with IP cores to be easily tested.

Up to this point we have considered techniques that require external generation and application of test patterns by an external device like automatic test equipment (ATE). BISTs are true DFT technique. It encompasses test generation, test application and response verification. It is very useful for current technology which requires testing at speed with due consideration to interconnect delays. Where SAF model fails, BIST succeeds. BISTs can detect faults that otherwise would not have been detected using SAF models – delay faults. In this methodology, test patterns are generated and test responses are analyzed on-chip.

The test pattern generator (TPG) in a BIST is implemented with linear feedback shift registers (LFSR) which is a finite state machine. It is a shift register with feedback from the last stage and other stages. The outputs of the flip-flops form the test pattern. It consists of FFs and XOR gates. The number of FFs and XOR gates depends on the characteristic polynomial of the LFSR. The generic BIST architecture is shown in figure 7. The responses of the circuit under test (CUT) could be large. Consequently the output responses are compacted by the response compactor (RC) to generate a signature at the end of the test application since we are interested on how the circuit responded to the various test patterns from the LFSR.

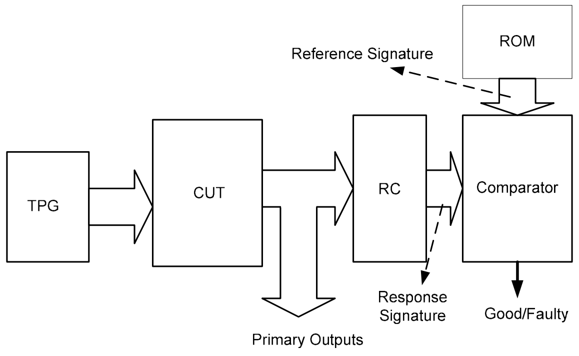

Fig. 7. General BIST Architecture

The generated signature is compared with the reference signature (signature of the fault-free circuit) to know whether the CUT is faulty or not. The detailed information on test generation and response compaction is beyond the scope of this paper. For more detailed information refer to [1], [8], [10] and [11].

VI. A Simple Example Of DFT Technique Using Scan Chain Methodology

As earlier mentioned DFT techniques help increase the testability of fabricated circuit by enhancing the controllability and observability of the various nets of the circuit. To show how DFT enhances the testability of a circuit, let us consider a simple counter circuit as shown in figure 8. The circuit is divided into two parts: combinational and sequential. The part containing the AND and XOR gates is the combinational circuit. The circuit has the following parts accessible to the outside world: outputs q0 to q2, Clock, Enable and Clear inputs. As it is now it will be difficult to properly test this circuit since we have no access to the internal nodes. If node n4 is stuck-at 1 or 0 there is no way we can know about this since we can neither control nor observe the node.

We are going to make this circuit testable by introducing some extra hardware and increasing the input and output ports. Firstly we replace the three flip-flops (FF) with a different type of FFs that has a multiplexer at the D input. By this action, additional three ports have been added namely: Scan-In, Scan-Out and Scan enable. The new sequential circuit is shown in figure 9.

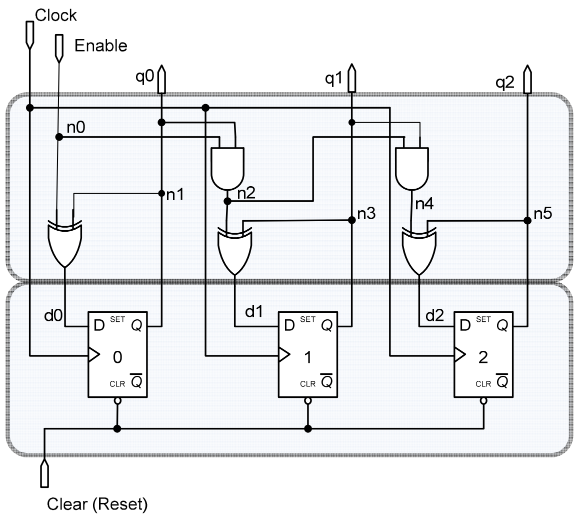

Fig. 8. A simple Counter Circuits

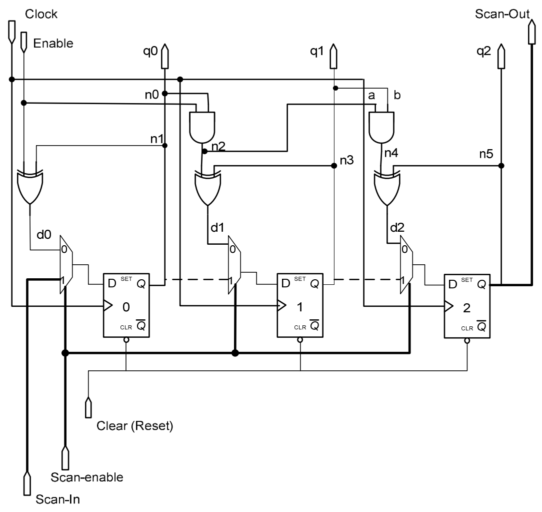

Fig. 9. A simple Counter circuits with DFT

With the new configuration the FFs form a shift register. The bit sequence can be shifted into the FFs through the scan-in input pin with the scan-enable signal set to high (logic 1) and the bits shifted out of the shift register can be observed at the scan-out output pin. Under normal operation of the sequential circuit the scan-enable signal is set to low (logic 0). The only change here is that our circuit can operate in two modes – normal and test modes. We can now develop and generate tests pattern for the combinational part to test the whole circuit the FFs inclusive. Let us assume that the node n4 is stuck-at-0. We can control input lines ‘a’ and ‘b’ to logic ‘1’ and set n5 to ‘0’ and observe the output at scan-out pin. The purpose of setting n5 to ‘0’ is to propagate the fault n4 stuck-at-0 to the output d2 of the XOR gate. Let us now look at how we can detect the fault stuck-at-0 at line n4.

Reset all FFs to 0

Set line ‘a’ =1 by setting enable input =1 and
n0=0 (FF0 was earlier reset to 0) d0=1,
Subsequently, FF0 output will be set to 1.

With enable=1 and FF0=1 => n2=1
Set line 'b' =1, by setting FF1 output to 1.
If n2=1, then d1=1 => FF1=1.
Set n5=0. Since n5 is the same as the FF2 output n5 is already 0.

With the above settings we are supposed to have logic 1 at the output. If however, the output is 0, then node n4 is stuck-at-0.

It is important to note that the functionality of the sequential circuit is not affected by the extra circuitry that implements the DFT technique. The major advantage of this modification is that testing of this circuit has become a combinational problem rather than a sequential one. The down side is that the circuit area has been increased, though not significantly.

VII. Conclusions

In this paper it has been shown that product quality depends to a greater extent on the thoroughness of verification and testing processes during its development. Testing of digital components/system is time consuming, expensive and can negatively affect time to market. The example given in this paper has clearly demonstrated that design for testability greatly eases the process of testing without a serious consequence on the area and delay issues of the would-be chip.

References

[1] A D&T roundtable System Test – What, Why and How? *IEEE Design and Test of Computers,* vol. 7, pp. 66 – 72, 1990.

[2] Schmid D, Wunderlich H, et al "Integrated Tools for Automatic Design for Testability", *in Proc. Conference on Tool Integration and Design Environments, Amsterdam:* Elsevier Science Publishers B.V.(North Holland), IFIP, 1988, pp. 233-258.

[3] Fang H, Chakrabarty K, Hideo Fujiwara H, "RTL DFT Techniques to Enhance Defect Coverage for Functional Test", *Journal of Electronic Testing: Theory and Applications* (JETTA) vol. 26, pp.151–164, 2010.

[4] Yu-Ting Lin, Williams D, Ambler T, "Cost-effective designs of field service for electronic systems", *in Proc. International Test Conference,* 2005, pp.460 – 467

[5] Grout I., *Digital Systems Design with FPGAS and CPLDS*. London: Newness-Elsevier, 2008.

[6] Breuer M.A., Friedman A.D., *Diagnostics and reliable design of Digital Systems Computer.* New York: Science Press, 1976.

[7] Taiwan Semiconductor Manufacturing Company (TSMC), TSMC Announces Move to 20nm Process. 2010, May 14. Available: http://www.tsmc.com/tsmcdotcom/PRListingNewsAction.do?action=detail&newsid=4741&language=E. Accessed

[8] Abramovici M, Breuer M.A, Friedman A.D., *Systems testing and testable design*. New York: IEEE Press, 1990.

[9] Bennetts R.G. *Design of Testable Logic Circuits*. Reading, MA: Addison-Wesley, 1984.

[10] Mourad S., Zorian Y., *Principles of Testing Electronic Systems.* New York: Wiley, 2000.

[11] Niraj Jha and Sandeep Gupta, *Testing of digital systems.* New York: Cambridge University Press, 2003.

[12] Brayton R, Cong J. "NSF Workshop on EDA: Past, Present, and Future (Part 2)", *IEEE Design and Test Computers,* vol. 27, pp. 62 – 73, 2010.

[13] Cho KY, Mitra S, McCluskey EJ Gate exhaustive testing, *in Proc. International Test Conference*, 2005, pp. 777 – 183.