Forms, solutions and effects regarding pattern resistant logic

MIRELLA AMELIA MIOC
Computer Science Department
"Politehnica" University of Timisoara
Bd. V. Parvan 2, RO-300223
ROMANIA
mmioc@cs.utt.ro http://www.cs.upt.ro/ro/Staff/~mmioc

Abstract: - Built-in self-tests are the heart of any modern reliability tests. Their applications are ranging from cryptography and bit-error-rate measurements, to wireless communication systems employing spread spectrum or code division multiple access techniques. However the strict time constraints limit the complexity of the tests as such, that multiple compression methods via a parallel LFSR (Linear Feedback Shift Register) signature analyzer exist. This paper's purpose is to raise awareness of the issue to propose a common framework for the identification of pattern resistant logic as well as a means to ensure a more stable and safe fault tolerance using a ROM (Read Only Memory) memory as a look-up table. The alternative proposed method for implementing BIST (Built In Self Test) avoids the use of advanced compression algorithms and needs very little hardware overhead so having small cost and die size.

Key-Words: - Linear Feedback Shift Register, Built In Self Test, Random Pattern Resistant, Pattern Resistant Logic, Level-Sensitive Scan Design, Digital Signature, Cryptosystem, Very High Speed Integrated Circuit Hardware Description Language, Multiple-Input Signature Register, Look-Up Table.

1 Introduction

Pattern resistant logic (PRA) [12] is a growing concern. As embedded systems keep growing (i.e. Networks on Chip) in complexity, the number of pattern resistant circuitry grows as well. This issue can no longer be so easily ignored.

Almost all the reliability tests use the built-in self-test (BIST).

In a typical logic BIST system the component parts are:

- Logic BIST Controller;
- Test Pattern Generator (TPG);
- Circuit Under Test (CUT);
- Output Response Analyzer (ORA);

The figure 1 presents not only the configuration, but also the correlations between blocks. Between the Logic BIST Controller and the other three components there are bidirectional links.

Test Pattern Generator (TGP) communicate with the Circuit Unger Test (CUT) and this one with the Output Response Analyzer (ORA).

The Logic Bist Controller is the main part of this scheme, having a supervisor function.

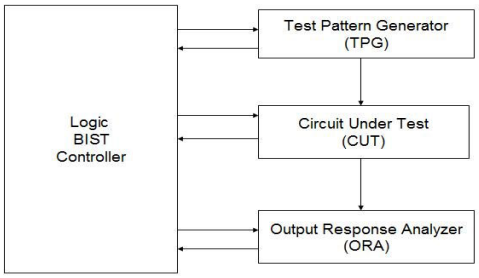

Fig. 1 A typical logic BIST scheme

Having a classic LFSR BIST design, a major potential problem with the LFSR approach is that some logic, such as an NA810 (an 8-input NAND gate), is pattern resistant. When any of the eight inputs is a 0, the output is a 1. Only when all eight inputs are 1s does the logic change state. A manufacturing defect, such as a stuck-at-0 fault on the output of the NA810, is extremely difficult to detect using random patterns. Another problem is that some control signals might not toggle as often as they would when generated by an LFSR.

The Linear Feedback Shift Register is the kernel of any digital system based on pseudorandom bits

sequences and is frequently used in cryptosystems, in codes for errors correcting/detecting [16], in wireless system communication [20], [21].
Linear Feedback Shift Register (LFSR) proved along the years its importance in developing many applications.
In the last years many programming languages and techniques help to simulate the functioning of LFSR and Multiple Input-output Shift Register (MISR) before their hard implementation.
Respecting mathematical ground for increasing the security usually LFSR works in a Galois Field and uses Irreducible or Primitive Polynomials [22], [23].
A serial-input signature register can only be used to test logic with a single output.
The idea can be extended from a serial-input signature register to the multiple-input signature register (MISR).
There are several types of BIST architecture based on the MISR.
By including extralogic an MISR can be reconfigured to be an LFSR or a signature register, named Built-in Logic Block Observer (BILBO).
By including the logic to be tested in the feedback path on an MISR, a circular BIST structures can be developed.
One of these is known as the circular self-test path (CSTP).
Compiled blocks including RAM, ROM and data path elements can be tested using an LFSR generator and a MISR.
To generate all 2N address values for a RAM or ROM it can modify the LFSR feedback path to force entrance and exit from the all-zeros state.
This is known as complete LFSR.
The pattern generator does not have to be an LFSR or exhaustive.
For a better understanding of the functioning of a Linear Feedback Shift Register some simulations using programming languages are useful.
There are two basic implementations for LFSR [17]:

- Fibonacci Implementation and
- Galois Form.

In Fibonacci form the weight for any status is 0, when there isn't any connection and 1 for sending back.
Exceptions of this are the first and the last one, both connected, so always on 1.

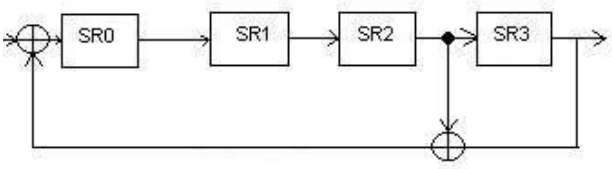

Fig. 2 Fibonacci Implementation

Both this scheme are related to the representation of the same Irreducible Polynomial:

$$x^4+x+1 \qquad (1)$$

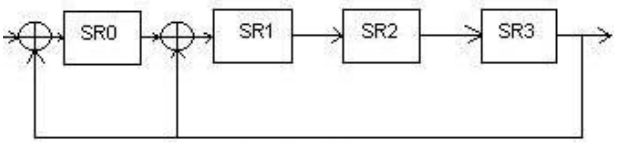

Fig. 3 Galois Implementation

In Galois implementation there is a Shift Register, whose content is modified each step at a binary value sent to the output.
Comparing the two type of representation it is shown that the weight order in Galois is opposite the one in Fibonacci.
For the Galois implementation the program developed in VHDL for the polynomial x^4+x+1 is the following:

```
entity lfsr_a is
 port (i:in std_logic;clk:in std_logic; rn:in std_logic;
      o:out std_logic);
end lfsr_a;
```

The input signal is i, the clock signal is clk, the not reset is rn and the output signal is o.
For the fourth SR's the used register is reg and for keeping the next states is reg_next.

```
signal reg, reg_next:std_logic_vector(3 downto 0);
```

The whole calculus is presented in the next rows.

```
reg_next(3)<=reg(2);  --shift
        reg_next(2)<=reg(1); --shift
        reg_next(1)<=reg(0)      xor      reg(3);
        reg_next(0)<=reg(3) xor i;      o<=reg(3);
-- initialize state
```

For the Fibonacci implementation the program is:

```
architecture str of lfsr_b is
 signal reg:std_logic_vector(3 downto 0);
```

```
 signal reg_next:std_logic_vector(3 downto 0);
 begin
  process (clk,rn)
    begin
     if rn='0' then
       reg<="0000";
     elsif(clk'event and clk='1') then
     reg<=reg_next;
    end if;
  end process;
  reg_next(3)<=reg(2);
  reg_next(2)<=reg(1);
  reg_next(1)<=reg(0);
  reg_next(0)<=i xor (reg(3) xor reg(2));  --porti
xor in cascada
  o<=reg(3);
end str;
```

One of the most famous researchers in this field is Solomon Golomb. He studied the linear feedback shift register sequence commands [18] and explained how the stream ciphers have been used for a long time as a source of pseudo-random number generators.

Important applications of satisfying the Golomb's three randomness conditions are the Berlekamp-Massey algorithm and Bose-Chaudhuri-Hocquenghem (BCH) Decoding [19].

A system reset or a clear terminal is a good example. If this signal toggled all the time, the logic would keep resetting and poor fault detection might result [8]. NAND gates being not all that uncommon in any type of circuit is a serious concern.

There are however approaches to tackle this problem, one such situation is to design random pattern testable circuits to use post-synthesis test point insertion to eliminate random pattern resistant (RPR) faults [5].

The advantages of Random pattern testing are:

- No requirement of test pattern storage;
- No deterministic test set generation cost;
- Obtaining a high suitability for BIST;
- Higher coverage of non-targeted faults.

The number of input patterns that detect the fault divided by the total number of input patterns is the detection probability of a fault.

Random pattern resistant (RPR) are the faults which have a very low detection probability and are hard to detect with random patterns.

A circuit is random pattern testable when it does not have any RPR.

A classical scheme is shown in the following schemes 4, 5 and 6 [1].

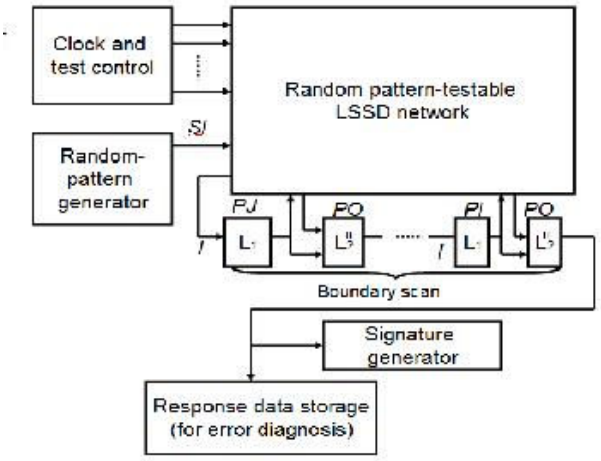

Fig. 4 General structure for random-pattern self-test

The result patterns are given to a Signature Generator which contains most of the compression logic, the digital signatures are The result patterns are given to a Signature Generator which contains most of the compression logic, the digital signatures are.

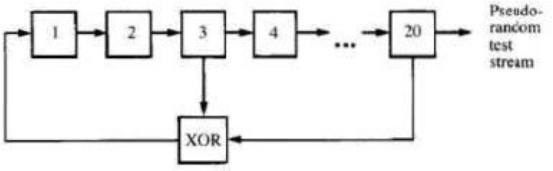

Fig. 5 Random pattern generator

The random pattern generator is a 20-bit maximal-length-sequence linear feedback shift register with taps at bits 3 and 20, and with a test bit-stream output port. There are a lot of tests needing sequential sequences and for all this cases is necessary to use Pseudorandom Generators [9], [14].

The signature generator is a 20-bit maximal-length-sequence LFSR with eight taps and a device test-response-bit input port.

As seen in figure 4, using a LFSR the pseudorandom sequence is generated afterwards it is processed by the LSSD then sent to a memory storage where they will be compared with already existent information thus yielding the result of the error correction code.

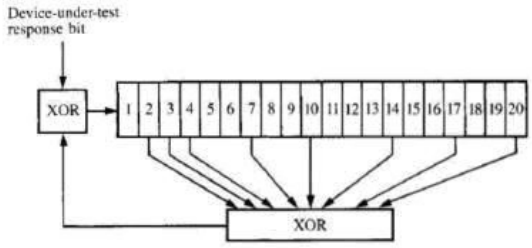

Fig. 6 Signature generator

More advanced systems (i.e. Systems on Chip) with heterogeneous system modules, that usually perform different functions and have considerable differences in design and technology used, have generally four methods of enhancing the fault coverage:
1. weighted pattern generation;
2. test point insertion;
3. mixed-mode BIST;
4. hybrid BIST.
The first three approaches are applicable for in-field coverage enhancement while the fourth approach is applicable for manufacturing coverage enhancement [4].
An interesting application is the Ramp Generator for the ADC BIST [13].

2 Proposed Approach

It is not uncommon for a BIST design [10] to contain a set of test vectors in a ROM (Read Only Memory) [7] and use it by implementing a counter to see exactly which tests to select. Using a minimal hardware overhead of a ROM memory with several control logic to achieve the function of a Look-up Table (LUT), the purpose is to inject at first the "sensitive tests" afterwards letting the BIST logic continue its normal functionality. Sensitive tests are those tests that are the cause of stuck-at-0 or stuck-at-1 in the case of NAND and other pattern resistant logic. Basically any test that would appear in pattern resistant logic is deemed here as „sensitive" and should be treated first, as chances of it appearing in a pseudo-random generation are quite low. Of course a simple routing logic (MUX) is required to ensure at first the entries from the LUT are injected and only afterwards a sequence of PRP will be injected results being afterwards stored and compared. The advantages of this static approach using LUTs is that the time needed to access a LUT entry is less than a clock cycle [6] and the static nature of the ROM(number of entries and their content doesn't change) makes it easy to design and integrate. The number of entries in the LUT as well as the length of one individual entry will be influenced by the number of inputs of the ASIC in question and/or the types of pattern resistant logic. A common framework is of course required.

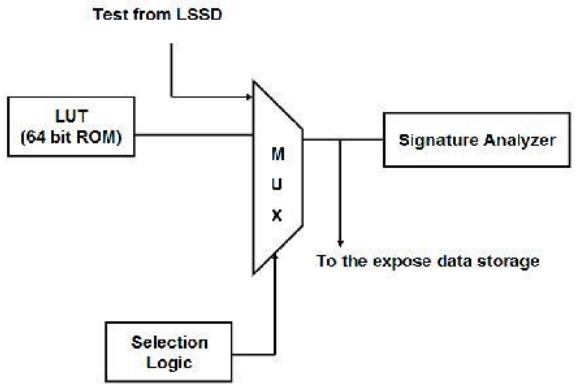

Fig. 7 Proposed enhancement scheme

Several steps should be taken when identifying the pattern resistant logic and generating the set of test that will later be stored in the LUT.
1. Having a map of all the logic in the device (be it sequential or combinational);
2. identify classic combinational PRL: NAND gates, XOR gates with high fan-in etc.;
3. pay special attention to sequential logic and its' transitions;
4. identify any other functional blocks (i.e. DAC/ADC) and make sure it can identify PRL within them;
5. generate a sample space of test vectors;
6. process the correct output;
7. design the LUT in such a way that the most common test vectors will be looked-up first (to gain more time and identify more common entries faster).

3 Implementation

In order to present the proposed approach some general information from testing are useful to be remembered.
"Testing" is the evaluation on the reliability and quality of a digital IC.
It is composed of distinct phases which are mostly kept separate both in the industrial practice and in research.
There are three main parts for "testing":

a. Verification
b. Testing

c. Parametric testing

a. Verification is the initial phase in which is verified the correctness of the design to ensure that the first prototype chips matches their functional specification.

This verification checks that all design rules are addressed to, from layout to electrical parameters.

Two conditions are necessary:

- The circuit implements what it is supposed to do and
- Does not do what it is not supposed to do.

This type of evaluation uses a variety of techniques as:

- Logic verification with the use of hardware description languages
- Full functional simulation
- Generation of functional test vectors.

b. Testing is the phase for ensuring that:

- only defect-free production chips are packaged and shipped and
- detect faults arising from manufacturing and/or wear-out.

The specific methods must respect three conditions:

- to be fast enough for applying to large amount of chips during production
- to take into consideration whether the industry concerned has access to large expensive external tester machines, and
- to consider whether the implementation of Built-In-Self-Test (BIST) proves to be advantageous.

In BIST it's possible to signal directly, during testing, it's possible failure status, because the circuit is designed to include its own self-testing extra circuitry.

So, a certain amount of overhead is involved in area and some trade-offs must be considered.

c. For ensuring components meet design specification for delays, voltages etc. is used parametric testing.

The number of I/O remains small, while the density of circuitry continues to increase. This produces a major escalation of complexity and so testing is becoming one of the major costs to industry.

Integrated circuits should be tested periodically during operation, before and after packaging and after mounting on a board.

There are different methods developed for each case.

The main purpose of testing is the detection of malfunctions, and the fault diagnosis or only subsequently one may be interested in the actual location of the malfunctions.

Usually the testing techniques are designed to be applied to combinational circuits only. In practice the idea to design a sequential circuit be partitioning the memory elements from the control functionality is frequently used, so a circuit can be reconfigured as combinational at testing time.

This method is one of the most used approaches in design for testability (DFT).

Figure 8 presents the general division for algorithms in testing.

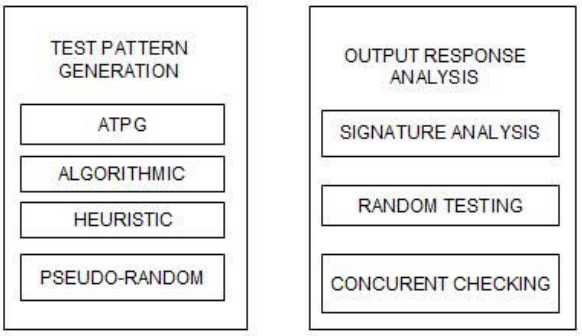

Fig. 8 Taxonomy of testing.
General division for algorithms

Test pattern generation consists of generating an appropriate subset of all input combinations, such that a desired percentage of faults is activated at the outputs.

The analysis of the output response implies methods for capturing only the output stream. The circuit is stimulated by either a random or an exhaustive set of input combinations.

This taxometry of testing methods is presented in the figure 9.

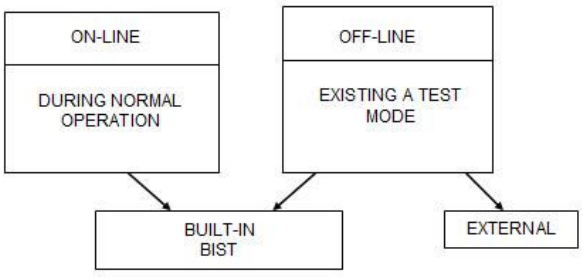

Fig. 9 Taxonomy of testing methods

Analyses of the methods can divide them in on-line and off-line methods.

In the on-line method each output word from the circuit is tested during normal operation.
In the other one the circuit must suspend normal operation and enter a „test mode" which determines the apply of the appropriate method of testing.
So, it can be executed either through the use of BIST or through external testing.
Also in on-line testing the circuit contains some coding scheme embedded in the design of the circuitry.
The clock port (CLK) is used in this implementation for external synchronization and the switch 0 port (SW0) is where the control signal that commands the injection of test vectors from the LUT will be activated.

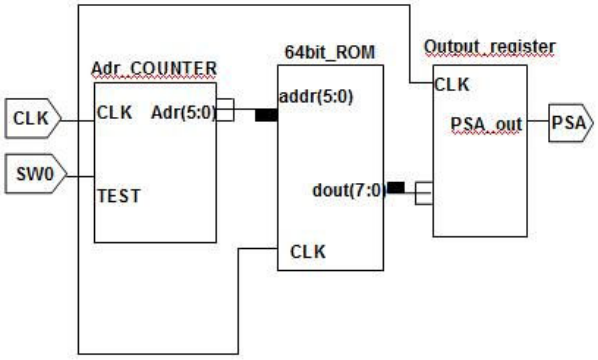

Fig. 10 The schematic (hardware) view of the enhancement

The counter selects the next address of the entry to be selected and on the next clock cycle(s) an entry from the ROM is read. In the following clock cycle the test vector is loaded in the output registry, which is meant to function as a buffer. When the buffer is full the tests are written in the PSA'S (Parallel Signal Analyzer) internal memory.
A traditional fully functional ROM block is used with a size of 64 bits.
Regarding the ROM I/O ports only the address, clock and data output ports are used.
In the figure 11 the used ports are marked.
ADDR is an input port marked as being used and also it can be seen the direction of pointing.
CLK is used for the function identically named and is also an input port.
From the output ports only DOUT is used and the printed direction indicates this.

In the classical form of a Read Only Memory (ROM) block all the other ports aren't useful for the functioning.

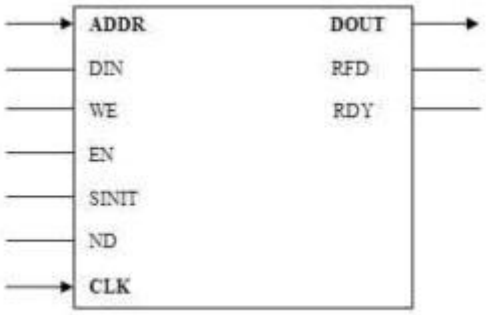

Fig.11 Read Only Memory block

An area of processor technology is embedded computing.
The same technological drivers towards multicore apply here too.
The cores are typically integrated onto a single integrated circuit die, named chip multiprocessor (CMP).
Usually multicore and dual-core referred to some sort of central processing unit(CPU), but sometimes applied to dual-core referred to some sort of central processing unit(CPU), but sometimes applied to digital processors[4](DSP) and System-on-a-chip(SoC).
Typically the embedded software is developed for a specific hardware release, making issues of software portability, legacy code or supporting independent developers.
So, it is easier to adopt new technologies and to develop a greater variety of multicore processing architectures and suppliers.
The number of cores in a die is increasing at a high rate.
Permanent of intermittent hardware faults, caused by defects in the silicon or metallization of process package and wear out ever time, lead to "hard faults".
The "soft errors", which cause random bit values to change erroneously, may be caused by electrical noise or external radiation.
Many of the soft errors that can occur inside a core are handled by CMPs.
With the hand faults is a more difficult task, having problems with the cost and space.
For solving the fault tolerance issues some special techniques are developed in the field of Reconfigurable Computing. [32]

Designing architectures for embedded systems using reconfigurable hardware is becoming more popular.
As building blocks for reconfigurable computing the following types are used:

- Field Programmable Gate Arrays (FPGAs);
- Field Programmable Transistor Arrays (FPTAs);
- Complex Programmable Logic Devices (CPLDs).

For the proposed implementation in this paper the counter and register implementation is simulated using a FPGA (Field-Programmable Gate Array) and the description language used is VHDL while the ROM is considered to be a standard block (i.e. the available ROM on the FPGA). The VHDL code (strictly behavioral) for the counter used is:

```
library ieee ;
USE ieee . std_logic_1164 .all ;
USE ieee . std_logic_arith .all ;
USE ieee . std_logic_signed .all ;
entity Adr_counter is
    Generic(N : natural := 64);
port ( Clk: in std_logic ;
  Test: in std_logic ;
  Adr: out std_logic_vector (5 downto 0));
end Adr_counter;

architecture str of Adr_counter is
  signal Count : std_logic_vector(N
        downto 0) := (others =>'0') ;
begin
        Adr  <= Count(N downto N−5) when
        Test = '1 ';
 process p(Clk)
 begin
        if (event 'Clk and Clk ='1') then
        Count <= Count + 1;
         end if ;
 end process;
end str ;
```

The VHDL code (strictly behavioural) used for the register is:

```
library ieee ;
USE ieee . std_ logic_ 1164 .all ;
USE ieee . std_ logic_ unsigned .all ;
USE ieee . std_ logic_ signed .all ;
entity reg  is
      Generic(n: natural :=8);
       port (reg in : in
    std_logic_vector(n−1 downto 0);
    Clk: in std_logic ;
PSA_out: out
  std_logic_vector(n −1 downto 0));
end Output_ reg;

architecture str of Output_reg is
signal PSA_out_tmp:
std_logic_vector(n−1 downto 0);
signal load: std_logic ;

begin
process(reg_in , Clk, load)
begin
load:= '1 ';−− the entry is always loaded

if (Clk ='1' and Clk'event)  then
          if  load = '1' then
          PSA_out_tmp <= reg_in ;
end if ;
end if ;
end process;
−− concurrent statement
PSA_out <= PSA _out_tmp;
end str ;
```

The performance of the proposed method shall depend on how fast the entry in the LUT is found, thus the LUT size per latency ratio is a crucial parameter.
The latency of a conventional LUT is situated between $\log_4 N$ (minimum) and N−3 (maximum) [3] depending on the technology used (the latency is given in clock cycles). N represents the number of inputs thus the LUT size would be 2N. Further, experimental work supporting the effectiveness of the proposed methods is also detailed.

Tab.1 Experimental results

Number of Inputs	Look-up table size [bits]	Minimum latency O(logN) =$\log_4 N$ [clock cycles]	Maximum latency O(N) =N−3 [clock cycles]
5	32	2	2
6	64	2	3
7	128	2	4
8	256	2	5

4 Related Works

Even in the field of medicine some particular methods from pattern recognition are developed.

For example a modern method used for Anesthesia Monitoring is the Lung Sound Pattern Analysis. The method combines noise cancellation and stochastic pattern recognition to enhance accuracy of diagnosis.
In the same time there are a lot of applications for detecting other respiratory diseases, which require further analysis and investigations.[18]
In this paper was described the structure and the use of a Signature generator related with the proposed enhancement scheme.
Another possible use of Digital Signature is in the field of cryptography.[11]
It was proposed a digital signature scheme that requires the original message in order to verify the signature.
The basis algorithm was El Gamal and a new public key cryptosystem using block upper triangular matrices with elements in Z_p was developed.
Also, the authors: Alvarez, Martinez, Vicent and Zamora presented some applications of this integral security kernel in any digital business protocols requiring security, like A/V content distribution systems, online payment systems, certified email systems, anonymous peer to peer systems and others.
Built-in-self-test is gaining increasing acceptance as an industry-wide test solution.
The design and shrink as process technologies become more complex thru developing a lot of analysis and applications.
Each basis method can be the kernel of different studies and experimental works.
For example based on the well-known scan design and BIST, scan-BIST scheme was developed.

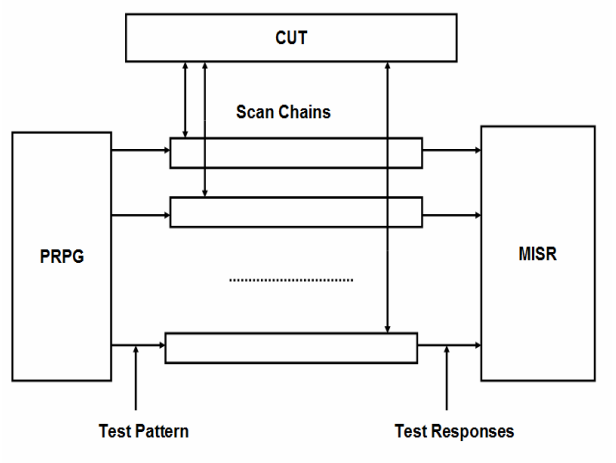

Fig.12 Generic scan-BIST scheme

These techniques apply a large number of patterns from a pseudorandom pattern generator (PRPG) to the circuit under test (CUT) via scan chains.
After that a compact signature is generated using a multiple-input signature register (MISR).
In the Scan-BIST Environment an interesting approach based on the application of overlapping intervals of test vectors was developed. [33]
In scan-BIST diagnosis the problem of identifying the set of failing vectors is a difficult one, because the length of a scan chain in a typical BIST scheme is usually much smaller that the number of test vectors applied to the CUT.
A lower bound on the number of true failing vectors was determined using a simple graph model.
There are many different kinds of implementations for Pseudo-random Pattern Generators depending on dimension, depending on the used hard and others.
An ASIC Implementation using a 32-bit LFSR was developed by Sedaghat and O'Brien.[14]
Usually all applications based on functioning of a LFSR use Irreducible Polynomials and it is proved that for increasing the security is better to use Primitive Polynomials LFSRs are popular mechanisms for built-in Test Pattern Generation (TPG) because they provide a pseudorandom source of bit patterns with low hardware overhead.
Often there are long subsequences of useless LFSR states with the consequence that the required overall test length is overly increased.
For addressing this issue many reseeding techniques were developed.[15]
Any reseeding technique is based on the fact that the LFSR have a new useful starting state (meaning "seed") every W codes.
W is an appropriate parameter defined such that the W subsequent state starting with the seed, contain presumably many other useful states.
Using a reseeding scheme in a BIST context it resulted how to generate the seeds efficiently in hardware.
There are four different types of such kind of Reseeding methods :

1. To keep the reseeding control information in a ROM [26], [28]. Such kind of reseeding approaches can be alternatively viewed as "test compression" approaches, as the long test vectors are represented in terms of shorter LFSR patterns.
2. To generate from the current seed the nest seed [29].

3. To generate the next seed from the XOR sum of the current seed and the last state in the sequence of the previous seed [30].
4. To generate the next seed by appropriate mapping from the last state in the sequence of the current seed[25], [27]. There are some reseeding techniques applied to non-LFSR TPG mechanism, such as counters and accumulators [31].

There are some reseeding techniques applied to non-LFSR TPG mechanism, such as counters and accumulators [31].
The basis problem for designers of multicore architectures and system-on-chip is to achieve effective on-chip fault-tolerance.[32]
Solving problems as faults in BIST resources, BIST-node path failures and split brain a complete and effective realization of on-chip Fault-Tolerance using BIST Resources can be developed.
For applying the proposed method is necessary to test periodically system entities for permanent failures using BIST resources and recovering them.
For any failure a check point of previous system state can be used.

5 Future work

In this paper it has been shown that the proposed alternative method for implementing BIST uses very little hardware overhead and has a small cost and die size. An important aspect is the use of the Look-Up Table (LUT), which can be developed in different implementations.
In the figure 7 the proposed enhancement scheme contains as LUT a 64 bit ROM.
It is possible to change this one with a RAM or EEPROM memory.
Also SRAM and DRAM comparisons should be made.
All this tests need to be made as performances depending on the access time on the LUT.
Some future work can use hybrid techniques combining the main three already existing:

- Software enumeration;
- Hardware enumeration;
- Integer Linear Programming (ILP).

6 Conclusions

The trend nowadays is to create more and more performant compression algorithms. The idea is to use a conventional look-up table to inject the test cases for pattern resistant logic first and afterwards let the pseudo-random number generator continue it's work. It is still uncertain what the impact on process time of the BIST will be, however in terms of cost and die area it is a fair trade off because high level compression algorithms require massive hardware overhead and contribute significantly to die area. Of course hybrid BISTs using the aforementioned technologies are beginning to emerge. This proposed variant for LUT approach should guarantee a fast and easy method to increase fault-tolerance in fields of study where time and the die size of the chip are crucial.
The proposed method reduces production cost, implementation overhead and increases reusability.
Also obvious advantages are obtained and the lifetime of VLSI chips are prolonged.

References:

[1] E. Lindbloom E. B. Eichelberger, Random pattern coverage enhancement and diagnosis for lssd logic self-test, *IBM Journal RES. DEVELOP*. VOL. 27 NO. 3 MAY 1983,pp.265-271;
[2] Andrzej Hlawiczka, Compression of multiplevalued data serial streams by means of parallel lfsr signature analyzer, *Informatik-Fachbericht*, Vol. 84, Ed. Springer-Verlag, London, UK,1984, pp. 406–416;
[3] Oskar Mencer, Wayne Luk, Kubilay Atasu and Tim Todman, Optimal implementation of combinational logic on look-up tables, *Research in Microelectronics and Electronics, 2008,PRIME 2008. Ph. D.*, June 22 2008-April 25 2008, Istanbul,pp.153-156;
[4] Nur A. Touba, Laung-Terng Wang, Charles E. Stroud, System-on-chip test architectures, Elsevier, Inc., 2007;
[5] Edward J. Mccluskey and A. Touba, Automated logic synthesis of random pattern testable circuits, *Proc. of International Test Conference*, 2-6 Oct1994 pp. 174 – 183;
[6] Jeffrey Hugh Reed: Software Radio, A Modern Approach to Radio Engineering, Ed. Prentice Hall, 2002.
[7] CharlesE.Stroud, A designers guide to built-in self-test, Kluwer Academic Publisher, 2002.
[8] Texas Instruments. Self-Test Services, Ambler,Norwell,Massachussets,USA, PA, 1989;
[9] Jose-Vicente Aguirre, Rafael Alvarez, Leandro Tortosa, Antonio Zamora: An Optimized Pseudorandom Genrator using Packed Matrices, *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS Manuscript* received Oct. 22, 2007; revised Mar. 22, 2008;

[10] A. Al–Yamani II, Logic BIST: Theory, Problems, and Solutions, Stanford University, *RATS/SUM02*, 2002;
[11] Rafael Alvarez, Francisco-Miguel Martinez, Jose-Francisco Vicent, and Antonio Zamora:, A Matricial Public Key Cryptosystem with Digital Signature, *WSEAS TRANSACTIONS on MATHEMATICS Manuscript* received Nov. 28, 2007; revised March 17, 2008;
[12] Fiorence Choong, Faisal Yasin, Shahiman Sulaiman and Mamun Reaz, VHDL Modeling of an Artificial Neural Network for Classification of Power Quality Disturbance, *WSEAS TRANSACTIONS on CIRCUITS AND SYSTEMS*, Issue 3, Volume 3, May 2004, pp.1355-1360;
[13] Wang Yong-Sheng, Wang Jin-Xiang, Lai Feng-Chang, Ye Yi-Zheng, On-Chip Ramp Generator for ADC BIST, *WSEAS TRANSACTIONS on SYSTEMS*, Issue 12, Volume 4, December 2005, pp.2448-2453;
[14] R. Sedaghat, B. O'Brin, "ASIC Implementation of a Pseudo-random Test Pattern Generator Using a 32-bit Linear Feedback Shift Register (LFSR)", Proc. International Conference for Upcoming Engineers, ICUE, 2003;
[15] S. Udar, D. Kagaris, LFSR Reseeding with Irreducible Polynomials, IOLTS 2007, pp.293-298;
[16] W. Wesley Peterson, Error-Correcting Codes, MIT Press, 1970;
[17] M Goresky, A. Klapper: Fibonacci And Galois Representations of Feedback with Carry Shift Registers, December 4, 2004;
[18] G. Solomon, Shift register sequences, Aegean Park Press, Laguna Hills, Canada, 1967;
[19] J. L. Massey, Shift-Register Synthesis and BCH Decoding, IEE Transactions on Information Theory, vol. 15, Issue 1, Jan 1969, pp.122-127;
[20] M. A. Mioc, Study of using Shift Registers in Cryptosystems for grade 8 irreducible Polynomials, WSEAS Conference SMO, 23-25 September, 2008;
[21] M. A. Mioc, A complete analyze if using Shift Registers in Cryptosystems for grade 4, 8 and 16 Irreducible Polynomials, WSEAS Transactions on Computers, vol. 7, Issue 10, October 2008, pp.1805-1817;
[22] M. A. Mioc, Simulation study of the functioning of LFSR for grade 8 Irreducible Polynomials, WSEAS Conference ISPRA, 21-23 February, 2009;
[23] M. A. Mioc, Correlation Formula referring different implementations of a Linear Feedback Shift Register for the same Irreducible Polynomials;
[24] H. Zheng, H. Wang, L. Y. Wang, G. Yin, L:ung Sound Pattern Analysis for Anesthesia Monitoring, 2005 American Control Conference, *Proc. of the 2005 Volume*, June 8-10, 2005, pp. 1563-1568
[25] A. A. Al-Yamani, E. J. McCluskey, Built-in Reseeding for Serial BIST, Proc. VLSI Test Symposium, Apr. 2003;
[26] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, B. Courtois, Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers, IEEE Transactions on Computers, vol.44, no.2, pp.223-233, Feb. 1995;
[27] E. Kalligeros, X. Kavousianos, D. Nikolos, Multiphase BIST: A New Reseeding Technique for High Test Data Compression, IEEE Transactions on CAD, vol.23, no.10, pp.1429-1446, Oct. 2004;
[28] H.-S. Kim, S. Kang, Increasing Encoding Efficiency of LFSR Reseeding-Based Test Compression, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.25, no.5, pp.913-917, May 2006;
[29] B. Koenemann, A Pattern Skipping Method for Weighted Random Pattern Testing, Proc. European Test Conference, 1993, pp.418-425;
[30] J.Savir, W. H. McAnney, A Multiple Seed Linear Feedback Shift Register, IEEE Trans. Computers, vol.41, no.2, pp.250-252, 1992;
[31] A. P. Stroele, F. Mayer, Methods to Reduce Test Application Time for Accumulator-Based Self-Test, Procedings of IEEE VLSI Test Symposium, 1997, pp. 48-57;
[32] S. D. Mediratta, J. Draper, Effective Realization of On-chip Fault-tollerance Utilizing BIST Resources, Proceedings of the 5th WSEAS int. Conf. onn Circuits, Systems, Electronics, Control& Signal Processing, Dallas, USA, 1-3 November, pp. 215-225.
[33] Chunsheng Liu, Krishnendu Chakrabarty, Failing Vector Identification Based on Overlapping Intervals of Test Vectors in a Scan-BIST Environment, IEEE Transactions on Computer-Aided Design of Integrated Circuits, vol.22, no. 5, pp. 593, May 2003.
[34] Rajesh S, Vinoth Kumar C, Srivatsan R, Harini S, A. P. Shanthi, Fault Tolerance in Multicore Processors With Reconfigurable Hardware Unit.