Vol. 1(4), pp. 139-147, 3 FEB, 2016
ISSN Print : 2454-1435 © 2015 IJRMMAE
ISSN Online : 2454-1443 © 2015 IJRMMAE
http://www.ijrmmae.in

International Journal of Research in Mechanical, Mechatronics and Automobile Engineering

DESIGN AND DEVELOPMENT OF NEW TECHNIQUE FOR TESTING OF FIELD PROGRAMMABLE GATE ARRAYS

M. Saravanakumar [1 a*], D. Sobya [2 b], B. Sathis kumar [1 c]

[1] Department of Computer Science and Engineering,
Dr. M.G.R. Educational and Research Institute, Chennai- 600 095, Tamilnadu, India

[2] Research Scholar, Department of Electronics and Communication Engineering,
Sathyabama University, Chennai – 600 119, Tamilnadu, India

[a,*] e-mail: navarasamblue@gmail.com, [b] e-mail: sobyadevaraj@gmail.com,
[c] e-mail:sathisnew@gmail.com

Received 29, December 2015 | Accepted 05, Feb 2016

Abstract

Subjects associated to the faults in SRAM-based Field Programmable Gate Arrays have been rigorously studied in modern research studies. This subject topic includes fault detection of FPGA, fault diagnosis of FPGA, defect tolerance of FPGA and fault tolerance of FPGA. A new technique for diagnosis of faults in interconnects and logic blocks of an arbitrary design implemented on an FPGA is presented. This work is complementary to application-dependent detection methods for FPGAs. This technique can uniquely identify any single bridging, open, or stuck-at fault in interconnect as well as any single functional fault in the logic blocks. The number of test configurations for interconnect diagnosis is logarithmic to the size of the mapped design, whereas logic diagnosis is performed in only one test configuration.

1. Introduction

SRAM-based Field Programmable Gate Arrays (FPGA's) are two-dimensional arrays of configurable logic blocks (CLBs) and programmable switch matrices, surrounded by programmable input/output blocks on the periphery. FPGAs are widely used in many applications such as networking, storage systems, communication, and adaptive computing, due to their reprogram ability, flexibility, and reduced time-to-market. The reprogram ability of FPGAs results in faster design and debugs cycle compared to Application- Specific Integrated Circuits (ASICs). However, once the design is finalized and fixed, the programmability becomes useless and costly. This is why FPGAs are very costly for high volume fixed designs compared to ASICs. Nevertheless, the reconfigurability of FPGAs can be readily exploited for defect tolerance at manufacturing level as well as fault tolerance for user applications. Application-dependent testing of FPGAs can be used by the manufacturer for defect tolerance in order to increase the manufacturing yield [Xilinx Easy Path] which in turn results in cost reduction. This is based on the fact that some FPGA chips that do not pass the application-independent test may be still usable for a particular design. In this case, the defects are located in some areas of

the chip not used by that design. This application-specific test flow improves the manufacturing yield, and hence, reduces the costs for high-volume fixed designs. During system operation, application-dependent test and diagnosis play a major role in online self-repair schemes for fault tolerant applications. In these applications, the existence of faults in the system is first identified and faulty resources are precisely diagnosed afterwards. Then, the design is remapped to avoid faulty resources. Because test and diagnosis procedures are performed during system operation (online), the number of test vectors and configurations must be minimized. Note that the test time is dominated by loading test configurations rather than applying test vectors. In this paper, application-dependent diagnosis techniques for logic and interconnect resources are presented. For interconnect diagnosis, the configuration of used logic blocks are modified, and the configuration of interconnects remains unchanged. Any single fault (open, stuck-at, or bridging fault) in interconnects can be uniquely identified in a small number of test configurations. For logic diagnosis, the configuration of used logic blocks remains unchanged while the configurations of the interconnect resources and unused logic blocks are modified. Any single functional fault, inclusive of all stuck-at faults, in logic blocks is precisely diagnosed in only one test configuration. As shown in this paper, these single-fault diagnosis techniques can be extended for multiple faults diagnosis. Recently, the notion of reconfigurable molecular computing at nano scale has been introduced which share lots of similarities with conventional FPGAs Since these devices are more vulnerable to defects at manufacturing and during system operation, defect and fault tolerant techniques are essential parts of these systems. Since the presented techniques are suitable for very large designs, they can be used as detection and diagnosis steps for defect and fault tolerance of reconfigurable molecular computing systems.

2. Literature Review

FPGA provides a solution for applications requiring dynamic reconfigurability, short time to market, and low nonrecurring engineering (NRE) cost. Many FPGAs are complex systems on chip; they contain not only configurable components, but also embedded processors, memories, and versatile peripherals. Unfortunately, making the FPGA bigger leads to longer critical paths. In the worst case, signals may have to travel across the chip, from one corner to another. The interconnection can comprise more than 50% of the critical path delay [1, 2] and 60% of the dynamic power consumption [3]. To enlarge the capacity without suffering performance loss, reducing the path length using a 3-D architecture becomes an attractive alternative. The 3-D FPGA can effectively cope with the long critical path problem. Several 3-D architectures [4, 5] have been investigated in academic studies, signalizing future directions for product development. Testing the 3-D FPGA is quite different from testing the 2-D FPGA. Unable to directly probe all pads after die stacking, it seems more difficult to implement the test method. New failure modes may incur during the 3-D integration process. Open in a TSV [6], bridge between adjacent TSVs, and short between a TSV and the substrate are permanent faults that could occur in the 3-D FPGA. TSV void and micro bump misalignment will affect the delays of certain paths in the 3-D FPGA, rather than forming permanent faults. Therefore, testing for delay faults to meet timing specification is vital, especially for those attributed to the interconnect [7].

An automatic test pattern generator for open, short, and delay faults on 3-D FPGA interconnects by exploiting the regularity of switch matrix topology and forming repetitive paths with finite steps and with loop-back were presented [8, 9]. We can provide universal logic functionality with all logic and signal restoration

operating at the nanoscale. The key properties of this architecture are its minimalism, defect tolerance, and compatibility with emerging bottom-up nanoscale fabrication techniques [10, 11]. Abderrahim Doumar provides a guided tour to the approaches related to include FPGA fault detection, FPGA fault diagnosis, FPGA defect tolerance, and FPGA fault tolerance [12]. Also, fan out branches of a net are tested in different test configurations, i.e. dependent logic cones are tested in different configuration, resulting in a few number of test configurations. Due to complexity of configuration generation algorithm, it cannot be applied to large designs.

3. Interconnect Diagnosis

The interconnect resources in FPGAs can be categorized as *inter-CLB* and *intra-CLB* resources. Inter-CLB routing resources provide interconnections among CLBs. Inter-CLB resources include programmable switch blocks and wiring channels connecting switch blocks and CLBs. Intra-CLB resources are located inside each CLB. Intra-CLB interconnects include programmable multiplexers and wires inside CLBs. Diagnosing faults in inter-CLB routing resources is addressed in this section. For inter-CLB interconnect test and diagnosis, the configuration of routing resources remains unchanged while the configuration of logic resources is modified. Test and diagnosis of intra-CLB interconnects along with logic resources are discussed in Sec. 4. For this purpose, the configuration of used logic resources (inclusive of intra-CLB interconnects) is kept unchanged whereas the configuration of inter-CLB interconnects as well as unused logic resources are changed. The separation between inter-CLB and intra-CLB is made because in contemporary FPGAs the programmable logic resources are not limited to LUTs; other logic resources such as carry generation propagation logic and cascade chains are included in CLBs. For inter-CLB interconnect test and diagnosis, these logic elements, if used in the original configuration, will be bypassed.

3.1 Single-Term Logic Networks

A *single-term function* is a logic function which has only one minterm or only one maxterm. In other words, the value of only one term in the truth table is different from the value of all other terms. The general form of a single-term function is a logic OR or logic AND function with some inversions (not necessarily) at the inputs and/or the output. The input corresponding to this specific minterm (or maxterm) is called the activating input. For a single-term function, if the applied input vector is the activating input, all sensitized faults are detected. An example is shown in figure 1, which is an OR function with inversions at the second and fourth inputs. This function has only one maxterm. Since the activating input (0101) is applied, A/1 (A stuck-at 1 fault), B/0, C/1, and D/0 are detected. Moreover the bridging faults between A and B (ABFB), ABFD, BBFC, and CBFD are also detected. This interesting testing property holds for any network of single-term functions. Consider a network of single-term functions N, and the input pattern V. If the values appear at the inputs of each gate (singleterm function) are the activating inputs of that single term function, all activated faults are detected. In other word, for each net n with value Vn, n stuck-at Vn is detected, and for each pair of nets, ni and nj, with Vni Vnj, the bridging fault between ni and nj is detected.

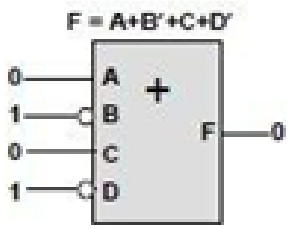

Fig. 1 Single-Term Function with Activating Input Pattern

For sequential networks, the extra requirement is to set the present value of each flip-flop to the value of the activating input corresponding to its data input net. In other words, the initial state of the circuit should be the same as the combinational circuit (if each flip-flop is replaced by a wire) with the conditions described above. In this case, the required number of test clock cycles is equal to the maximum *sequential depth* of the network. The test vector must remain unchanged during all these test clock cycles. This condition guarantees that the value captured in the first flip-flop rank will be propagated and observed at the primary outputs. An example of a sequential circuit with single-term functions which satisfies all these conditions is shown in figure 2. The preset values of the flip-flops are also shown. Here, two test clock cycles must be applied and the test vector (1101) must remain unchanged during these two clock cycles.

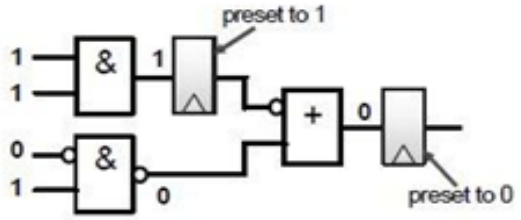

Fig. 2 Sequential Logic Network of Single-Term Functions

3.2. Test Configuration Generation

As explained in Sec. 3.1, single-term functions guarantee the detection of all activated faults. However, some mechanism is required to activate faults with respect to the fault list. We implement single-term functions in all used LUTs in the design. The question of which single-term function to implement in each LUT is addressed in this section, which gives us test configurations for inter-CLB interconnect testing.

Here, the idea of bus interconnect testing is used for activating the faults. In conventional board-level interconnect testing, only $\mathbf{log}2(M+2)$ test vectors are used for testing all possible stuck-at, open, and pair wise bridging faults for M interconnect bus lines], provided that each bus is directly controllable and observable. Figure 3 shows the test vectors for 6 bus lines ($\mathbf{log}2(6+2)=3$). These vectors can be converted to the activating inputs of LUTs implementing single-term functions. Note that if the values of all input nets and the output net for an LUT are known, the single-term function to be implemented in that LUT is also known.

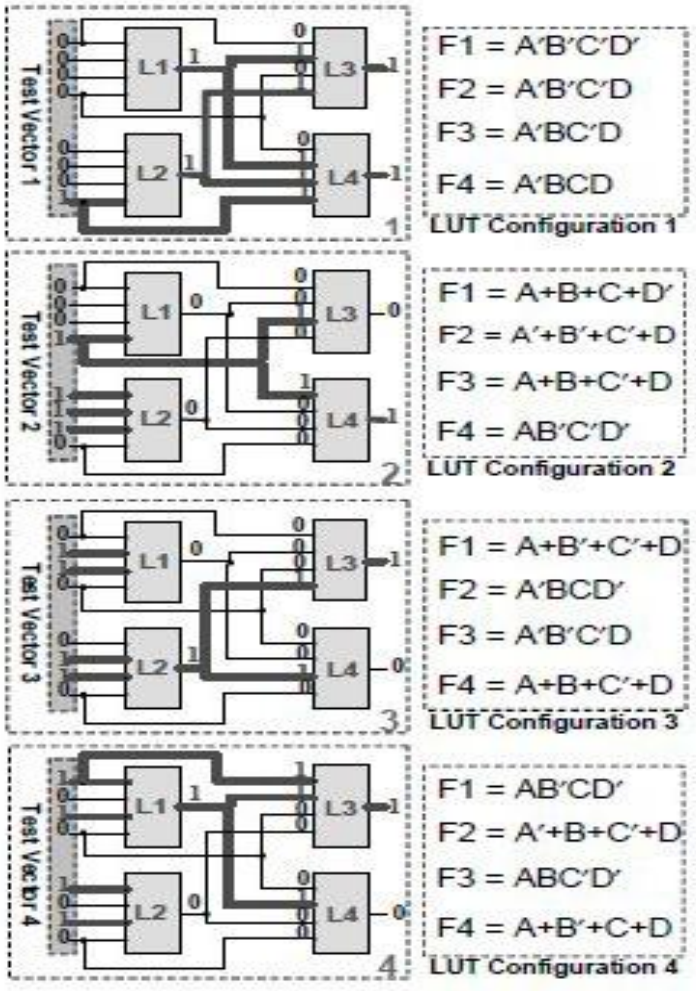

Fig. 3 Test Vectors and Configurations

Since these test configurations target faults in inter-CLB interconnect, all additional logic resources in CLBs, if used, will be bypassed. Hence, CLBs are configured as LUTs followed by flip-flops (if those flip-flops are originally used in the user configuration). Here, we assume that nets extend from an LUT output to LUT input(s).

3.3. Diagnosis Procedure

Assume that the outcome of each test configuration at the tester is a pass/fail result, i.e. the worst case condition is considered in which no further information regarding the failing outputs or test clock cycles is available. Note that this problem is different from conventional diagnosis approaches for bus interconnects in which all nets are fully observable]. This special diagnosis problem in which all (internal) nets have full controllability (using single-term functions, it is possible to set any desirable value to each net) but limited observability (instead of observing the value of each net, only the pass/fail outcome can be obtained). Having considered this assumption, to precisely diagnose one single fault out of n distinct faults, at least $\lceil \log 2n \rceil$ pass/fail outcomes are required

3.3.1 Diagnosis of Stuck-at faults

A circuit with *n* nets has *2n* stuck-at faults. Based on the above assumption, in order to uniquely identify any single stuck-at fault at least **log***22n* = 1 + **log***2n* test configurations are required. For example, consider the circuit shown in Fig. assuming that there is a stuck-at-0 fault on net *n9*. Using four test configurations presented in Fig. , the *failing pattern* is 1001 (first and fourth test configurations fail while second and third pass) which uniquely codes *n9* (9) in (4-bit) binary representation. Note that no two stuck-at-0 faults have the same fault pattern. However, the failing pattern for *n6* stuck-at-1 (1001) is exactly the same for *n9* stuck-at-0. This is why one extra test configuration is required to uniquely diagnose all stuck-at-0 and stuck-at-1 faults. This extra test configuration activates only stuck-at-1 faults. Based on the outcome of this test configuration, it can be determined if the failing pattern is encoding stuck-at-1 or stuck-at-0 faults. The diagnosis procedure for single stuck-at faults is as follows. Note that this procedure is non-adaptive.

1. The first configuration is all OR in which the activating inputs for all LUTs are all-0 (all LUTs implement the OR function). This configuration activates all stuck-at-1 faults.
2. The remaining [**log**2n] configurations are the same as Sec. 3.2

3.3.2 Diagnosis of Open faults

An open fault on a net can be detected by testing for both stuck-at-0 and stuck-at-1 faults on that net. In other words, an open fault behaves as both stuck-at-1 and stuck-at-0 faults on the same net . As mentioned in Sec. 3.3.1, the first step of stuck-at fault diagnosis consists of the all-OR configuration. If the all-AND configuration (the dual of all-OR) is also applied and both of them fail, it can be concluded that there is an open fault (based on single fault assumption). In this case, by applying the remaining [**log***2n*], the failing pattern uniquely identifies the net with open fault

3.3.3 Diagnosis of Bridging faults

The bridging fault list for a circuit with *n* nets contains *n(n-1)/2* distinct pair-wise bridging faults. Hence, at least **log***2*[*n(n-1)/2*] *2***log***2n – 1* test configurations are required for single bridging fault diagnosis. The presented technique in this section is an adaptive approach in which the determination of the next test configuration depends on the pass/fail outcome of the previous test configuration.

Theorem: For *n* nets, *w0, …, wn, 2*[**log***2n*] *– 1* adaptive steps can precisely identify any single twonet bridging fault.

Proof: It is based on an induction on the number of nets. For the sake of simplicity, assume *n* is a power of two (the proof holds for any arbitrary value for *n*). The first step sets *w0, …, wn/2-1* (subset 1) to 0 and *wn/2, …, wn* (subset 2) to 1. This activates *(n/2)(n/2) = n2/4* of bridging faults and the remaining *n(n-1)/2 – n2/4 = n2/4 – n/2* faults are not activated. If this step fails (i.e. there is a bridging fault between a net in subset 1 to a net in subset 2), the left sub-tree is a complete binary tree to uniquely identify one of *n2/4* faults in other **log***2 (n2/4) = 2* **log***2n – 2* steps (using binary search).

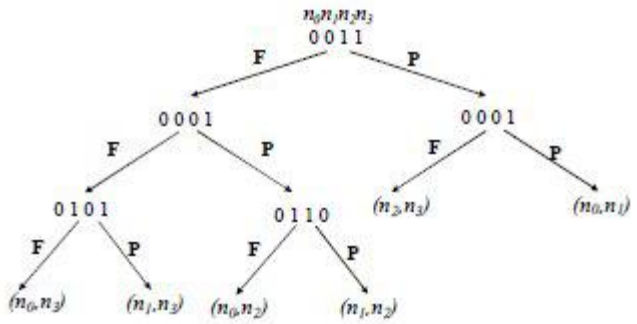

Fig. 4 Adaptive Steps for 4 Nets with 3 Steps

3.3.4 Overall Diagnosis Procedure

The pseudo code for the entire detection and diagnosis flow is shown in figure 4. The pass/fail outcomes of the all-OR and the all-AND test configurations determine the type of fault, namely stuck-at-0, stuck-at-1, open, and bridging faults. In the worst case, the total number of test configurations is equal to $3[\log 2n] + 1$ to detect all open, stuck-at, and bridging faults, as well as diagnosis of all single faults.

1. Apply [**log***2n*] test configurations (Sec. 3.2)
2. Obtain failing pattern *P* = *p*0…*p*_**log***2n*_
3. **if** diagnosis is required **then**
4. Apply all-OR configuration COR
5. Apply all-AND configuration CAND
6. **if** *P*OR*P*AND = 01 **then**
7. P corresponds to net with ***stuck-at-0*** fault
8. **else if** *P*OR*P*AND = 10 **then**
9. P corresponds to net with ***stuck-at-1*** fault
10. **else if** *P*OR*P*AND = 11 **then**
11. P corresponds to net with ***open*** fault
12. **else** /**P*OR*P*AND = 00 */
13. Apply $2[\log 2n] - 1$ adaptive steps to diagnose ***bridging*** fault

Pseudo code for the detection and diagnosis flow

Logic Block Diagnosis

4.1. Linear Compactor for Diagnosis

For logic block (including intra-CLB interconnects) testing and diagnosis, the configuration of the original used logic blocks is preserved while the configuration of interconnects and unused logic blocks are changed to exhaustively test and diagnose all used logic blocks. The idea of application-dependent logic block testing is presented in this technique. A linear feedback shift register (LFSR) or a binary counter is connected to the inputs of all used logic block generating test vectors. The outputs of all these logic blocks are connected to a response compactor (e.g. an XOR tree) to generate the pass/fail signal. The LFSR and the XOR tree are implemented in the available unused logic blocks. Since the LFSR generates all possible patterns (2^n patterns for an n input logic block) and the XOR tree propagates any single fault to its output, any single *functional* fault in the used logic blocks will be propagated to

the output of the XOR tree and will be detected. Functional faults are any faults that change the truth-table of an LUT, including stuck-at faults. An example of this scheme is shown in figure 5.

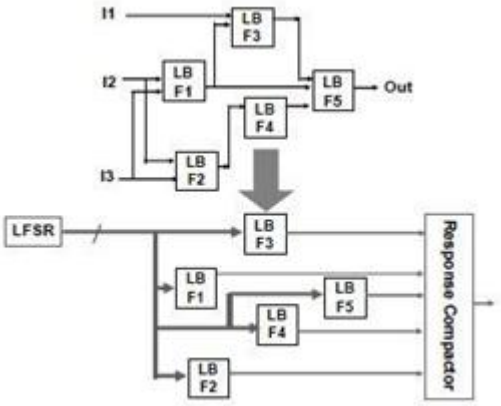

Fig. 5 Application-Dependent Self-Test Architecture for Logic Blocks

5. Conclusions

Based on the above study the following conclusions were made.

1. In this article, an application-dependent diagnosis technique for faults in interconnects and logic blocks of an arbitrary design mapped into an FPGA are presented.
2. For interconnect diagnosis, single faults (open, stuck-at, or bridging fault) can be uniquely identified. As shown in the paper, the number of total test configurations for interconnects diagnosis is logarithmic to the size of the design.
3. For logic block diagnosis, single faults can be uniquely identified in only one extra test configuration. The presented technique can be modified for multiple fault diagnosis.
4. This method can be used for defect tolerance by the manufacturer in order to increase the manufacturing yield, or as a part of online self-repair schemes for fault tolerant applications.

Moreover the presented techniques are suitable for very large designs and it can be used as detection and diagnosis steps for defect and fault tolerance of reconfigurable molecular computing systems which share similarities with conventional FPGAs.

6. References

[1] Gayasen, V. Narayanan, M. Kandemir, and A. Rahman, (2008), "Designing a 3-D FPGA: Switch box architecture and thermal issues," IEEE Trans. Very Large Scale Integr. (VLSI) System, Vol. 16, No. 7, pp. 882-893

[2] Dong, D. Chen, S. Haruehanroengra and W. Wang, (2007), "3-D nFPGA: A reconfigurable architecture for 3-D CMOS/nanomaterial hybrid digital circuits," IEEE Trans. Circuits System, International Fundam. Theory Appl., Vol. 54, No. 11, pp. 2489-2501

[3] L. Shang, A. Kaviani and K. Bathala, (2002), "Dynamic power consumption in Virtex-II FPGA family," in Proceeding of International Symposium in Field Programming Gate Arrays, pp. 157-164

[4] Y. Chen, J. Zhao and Y. Xie, (2010), "3-D nonFAR: 3-D non-volatile FPGA architecture using phase change memory," in Proceedings of ACM/IEEE

International Symposium Low-Power Electronics and Design, pp. 55-60

[5] Y. Y. Liauw, Z. Zhang, W. Kim, A. El Gamal and S. Wong, (2012), "Nonvolatile 3-D FPGA with monolithically stacked RRAM-based configuration memory," in Proceedings of International Solid-State Circuit Conference, pp. 406-408

[6] F. Ye and K. Chakrabarty, (2012), "TSV open defects in 3-D integrated circuits: Characterization, test, and optimal spare allocation," in Proceedings in Design of Autom. Conference, pp. 1024-1030

[7] M. B. Tahoori and S. Mitra, (2004), "Interconnect delay testing of designs on programmable logic devices," in Proceedings of International Test Conference, pp. 635-644

[8] Yen-Lin Peng, Yung-Fa Chou and Cheng-Wen Wu, (2014), "Application-Independent Testing of 3-D Field Programmable Gate Array Interconnect Faults", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 22, No. 2, pp. 207-219

[9] I. G. Harris and R. Tessier, (2000), "Interconnect Testing in Cluster-Based FGPA Architectures", Proceedings DAC, pp. 49-54

[10] A. DeHon, (2003), "Array-based architecture for FET based nanoscale electronics", IEEE Transactions on Nanotechnology, Vol. 2, No. 1, pp. 23-32

[11] S.C. Goldstein and D. Rosewater, (2002), "Digital logic using molecular electronics", Proceedings IEEE International Solid- State Circuits Conference, Vol. 1, pp. 204-209

[12] A. Doumar and H. Ito, (2003), "Detecting, diagnosing and tolerating faults in SRAM-based field programmable gate arrays-A survey", IEEE Transactions on VLSI Systems, Vol. 11, No. 3, pp. 386-405