FAULT DETECTION AND FAULT DIAGNOSIS IN APPLICATION SPECIFIC FPGAs AND MEMORY ELEMENTS USING BIST TECHNIQUE

H.Senthilkumar,
Assistant professor
K S R Institute for Engineering & Technology.
Email: halansenthil.ece@gmail.com

ABSTRACT

This paper presents a fault detection and fault diagnosis of application specific field-programmable gate arrays (FPGAs) using application dependent fault detection and diagnosis. The proposed technique can test both the interconnect resources and lookup tables (LUTs) in the configurable logic blocks (CLBs).The test pattern generator and output response analyzer are configured by existing CLBs in FPGAs. Thus, no extra area overhead is needed for the proposed BIST structure. This technique can uniquely identify multiple faults in FPGA interconnects (stuck at ,open, bridging fault) as well as multiple functional fault in logic blocks, a fault resulting a change in the truth table of a function, 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 with less than 5% overhead of built-in self diagnosis. By using this technique we will be able to diagnosis the row/column faults ,word faults and stuck at faults in the FPGA memory.

INTRODUCTION

FPGAs are widely used in many applications such as networking, storage systems, communication ,and adaptive computing, due to their reprogrammability ,flexibility, and reduced time-to-market. The reprogrammability of FPGAs results in faster design and debug cycle compared to application- integrated circuits(ASICs). However, once the design is finalized and fixed, the programmability becomes useless and costly if infield further customization and reprogrammability are not required.

In order to reduce the manufacturing costs associated with FPGAs, application-specific FPGAs have been introduced in the FPGA industry which restricts the use of the FPGA device for only one application (design).The cost reduction is mainly due to using devices that may contain defects in the areas not used by the particular application.

Application-dependent diagnosis is also a key player in silicon debug process. Once a particular configuration (e.g., a test configuration at the manufacturing test or an application configuration in the field) fails, the location of defective resources needs to be precisely identified for failure mode analysis (FMA) and yield improvement. An effective application-dependent fault localization (diagnosis) method can reduce the overall silicon debug time and improve its precision and quality.

FAULT DETECTION&DIAGNOSIS

In this paper, application-dependent diagnosis techniques for logic and interconnect resources are presented. This is complements to an application-dependent testing of FPGAs . For interconnect diagnosis, the configuration of used logic blocks is modified, and the configuration of the interconnects remains unchanged. Multiple faults (open, stuck-at, or bridging fault) in the interconnects can be uniquely identified in a small number of test configurations. For logic diagnosis, a built-in self diagnosis (BISD) method is presented in which the configuration of used logic blocks remains unchanged while the configurations of the interconnect resources and unused logic blocks are modified.

Multiple functional fault, inclusive of all stuck-at faults, in logic blocks is precisely diagnosed in only one test configuration.

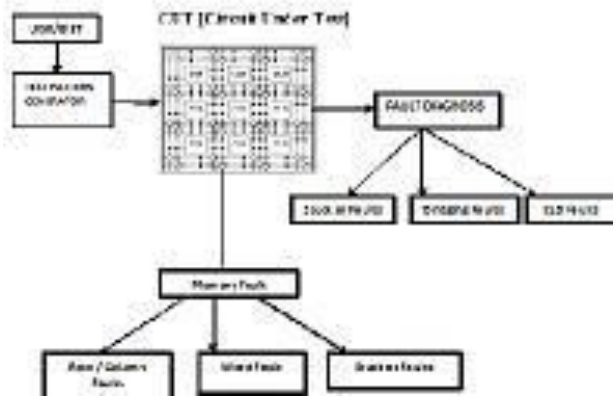

Fig 1: Block Diagram for fault detection

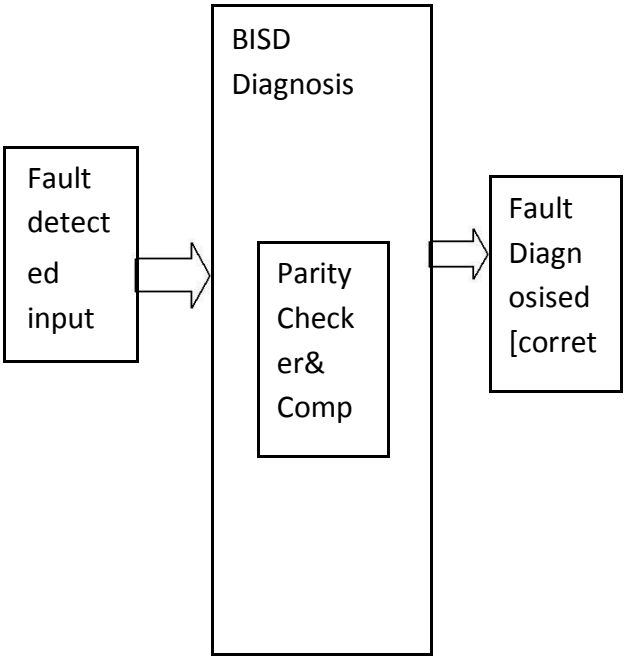

Fig 2: Block Diagram Fault diagnosis

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. 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 For sequential networks, the extra requirement is to set the preset value of each flip-flop to the value of the activating input corresponding to its data input net. The required number of test clock cycles is equal to the longest sequential path of the network.

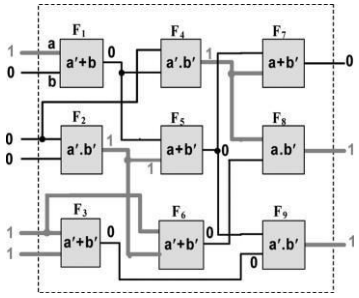

Fig3: Logic network of single term function.

The test vector must remain unchanged during all these test clock cycles. This condition guarantees that all blocks will receive their activating input in the fault free circuit (sensitization of all activated faults) and the value captured in the first flip-flop rank will be propagated and observed at the primary outputs (fault observation). This condition is particularly important for sequential designs containing sequential loops. In such circuits, without proper presetting of flip-flops it may not be possible to justify activating inputs for some blocks and hence, some faults cannot be detected. However for pipeline sequential circuits, i.e., those contain no sequential loops, presetting flip-flops is not necessary. In that case by applying one extra clock cycle, the flip-flops can be set by appropriate values and the fault effect can be propagated to primary outputs.

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. This is a 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 [log2n] pass/fail outcomes are required. Diagnosis procedures are categorized into adaptive and nonadaptive approaches.

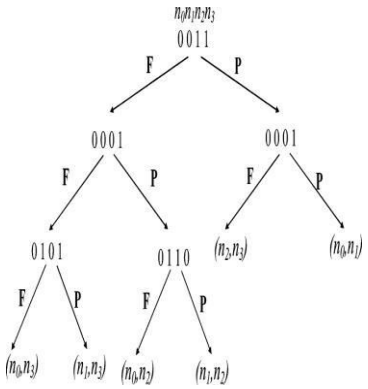

Fig4: Adaptive steps for four nets requiring three steps.

The decision tree including the activating input values of the nets (from which LUT configurations can be derived) depending on fail (F) and pass (P) outcomes of each diagnosis step is shown. In an adaptive procedure, the choice of the next step (test configuration) is based on the pass/fail outcome of the previous step. In a non-adaptive procedure, all steps are performed first (all test configurations applied and tester outcomes are gathered) and the faulty elements are determined based on the failing pattern.

The failing pattern for test configurations contains bit, each bit position is 1 if and only if the corresponding configuration fails at tester.Non-adaptive approaches are preferable over adaptive ones since the total test application time for non-adaptive procedures are typically smaller than that for adaptive ones.

LOGIC BLOCK DIAGNOSIS

Built-In Self Diagnosis (BISD)

In 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.In this BIST scheme, each used logic block will be exhaustively (or super-exhaustively, i.e., all possible transitions) tested while all these logic blocks are tested concurrently.The global interconnect is reprogrammed in such a way that the test signals are routed to each logic block. A linear feedback shift register (LFSR) or a binary counter for generating test vectors is connected to the inputs of all used logic blocks.

The logic block outputs are observed through an internal response compactor (e.g., an XOR tree). The response compactor can be combined with a response (parity) predictor, as will be explained shortly, such that a unique pass/fail signal can be generated. The LFSR and the XOR tree are implemented in the available unused logic blocks. Since the (de Bruijn) LFSR or binary counter 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.

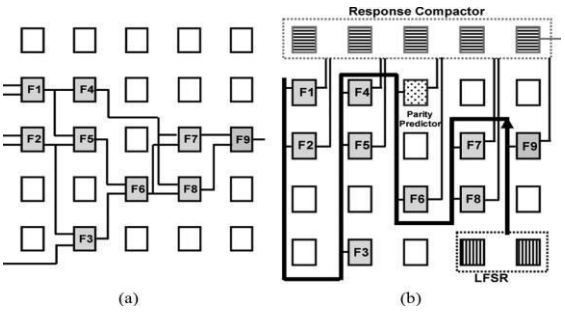

Fig5:Application-dependent self-test architecture for logic blocks. (a) Originalconfiguration. (b) BIST configuration.

Fig 5 shows an example of this scheme. In Fig5(a) the original design, with used logic blocks F1 to F9 with original interconnections, is shown. In the BIST configuration, the original interconnections are modifies such that LFSR outputs, implemented in unused blocks, are connected to the inputs of all used blocks

F1-F9 in parallel. The outputs of used blocks along with the parity predictor block are connected to the response compactor, which is also implemented in the available unused resources. The logic blocks in contemporary FPGAs contain other logic components besides LUTs, such as carry generation and propagation logic, cascade chains, and programmable multiplexers. Since exhaustive test patterns are applied to the inputs of each logic block, the exact faulty resource(s) inside the failing logic block(s) (all the resources in the user logic blocks, inclusive of all logic resources and local (intra-CLB) interconnect) can be uniquely diagnosed based on a fault dictionary.

MEMORY FAULTS DIAGNOSIS:

By using BISD technique multiple faults in FPGA memory (faulty words, faulty rows, and faulty columns) can be diagnosised. The faulty row/column is the continuous faults on the same row/column.

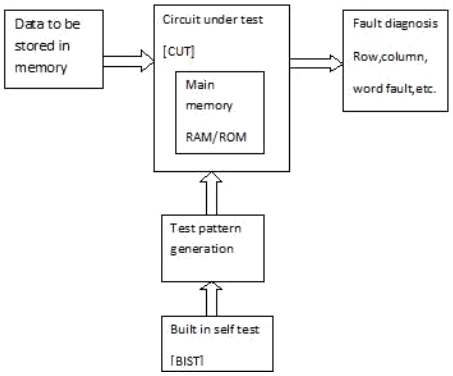

Fig 6:Memory fault diagnosis diagram

Our approach reduces the amount of data that need to be transmitted from the chip under test to the ATE (automatic test equipment). It therefore reduces the ATE occupation time and the required ATE capture memory space. It also simplifies the analysis that has to be performed on the ATE. Moreover, the proposed method does not increase the test time for the fault-free memories. A defect in different parts of thememory may lead to different faults and/or fail patterns. Fault identification is not trivial, and can be aided by using the fail pattern information.

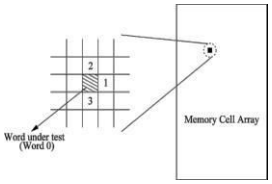

Fig. 7.Memory cell array being tested.

Fig. 7 shows the memory cell array being tested. The shaded region represents the word under test (WUT). If the WUT has a different output than the expected value when we read it, then the word fails and a fault is detected. In a typical BISD design, when a fault is detected the test process pauses, and the fault data is either registered or shifted out before the test process resumes. However, whether it is shifted out immediately or registered and then shifted out later, the cost (time complexity and ATE capture memory size) can be high if there are many faults.We use a more advanced approach, i.e., identify the faulty rows,faulty columns, and faulty words simultaneously during the test process.

Faulty Row : When theWUTis faulty, we test the next word in the same row, i.e., Word 1 as shown in Fig. 7. If Word 1 is also faulty, we continue to test the next word in the same row until we reach a fault-free word or the end of the row.

Faulty Column : Identification of a faulty column, assuming the WUT has been tested faulty, consists of several condition-checking steps.Word 1 is tested fault free, so a faulty row can be excluded.The word above the WUT in the same column (i.e.,Word 2 as shown in Fig. 1) is tested fault free, otherwise the WUT has been covered by the previous faulty column test.The word under theWUTin the same

column (i.e.,Word 3 as shown in Fig. 7) is tested faulty.We continue to test the subsequent words in the same column until we reach a fault-free word or the end of the column.

Single faulty word : When the WUT is faulty but not in a faulty row or column, i.e., Word 1, Word 2, and Word 3 are all tested fault-free, we consider the WUT as a single faulty word.

CONCLUSION

In this paper, fault detection and fault diagnosis of application specific field-programmable gate arrays (FPGAs) using application dependent fault detection and diagnosis. was presented. For interconnect diagnosis, multiple faults (open, stuck-at, or bridging fault) can be uniquely identified. As shown in the paper, the number of total test configurations for diagnosis of interconnects is logarithmic to the size of the design. For logic block diagnosis, a BISD approach is presented in which multiple faults can be uniquely identified in only one test configuration. In memory fault diagnosis the row/column faults, word faults and stuck at faults in the memory devices was diagnosised.

REFERENCE

[1] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, "Built-in Self-test of FPGA interconnect," in Proc. Int. Test Conf., 1998, pp.404–411.

[2] C. Stroud, E. Lee, and M. Abramovici, "BIST based diagnostics of FPGA logic blocks," in Proc. Int. Test Conf, 1997, pp. 539–547.

[3] M. B. Tahoori, "Application dependent testing of FPGAs," IEEE Trans.Very Large Scale Integr. (VLSI) Circuits, vol. 14, no. 9, pp. 1024–1033, Sep. 2006.

[4] R.-F. Huang, C.-L.Su, C.-W.Wu, S.-T.Lin, K.-L.Luo, and Y.-J.Chang, "Fail pattern identification for memory built-in self-repair," in Proc. 13th IEEE Asian Test Symp. (ATS), 2004, pp. 366–371.