Roving Concurrent Error Detection for Logic Circuits

Sobeeh Almukhaizim
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

Petros Drineas
Computer Science Dept.
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Yiorgos Makris
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

1. Introduction

The continuous increase in the complexity of current designs generates logic circuits that are less immune to soft errors. As the soft error rate increases, Concurrent Error Detection (CED) techniques are becoming ever more essential. A plethora of research efforts have been expended in developing CED techniques that provide high levels of reliability [1, 2, 3]. Yet, the high overhead associated with these CED methods led to little attention in the industry. In a recent trend, low-cost CED methods [4, 5, 6] have been proposed that trade fault coverage for area overhead. Low-cost CED in logic circuits is more appealing to the industry as it increases the reliability of a circuit at an acceptable area overheard.

In this paper, we present a new method for the synthesis of low-cost CED circuitry for logic circuits based on the notion of **test roving**. As illustrated in figure 1.a, multiple substructures that appear twice in a logic circuit are extracted and added to the CED circuitry in figure 1.b. In addition, we duplicate all the gates that are not part of a substructure in the roving CED circuitry to provide complete test of all potential failure points. As the circuit operates, every substructure in the CED circuitry roves -using input and output multiplexors- between the two instances that it matches. An output comparator is utilized to detects any faults in the substructure being tested.

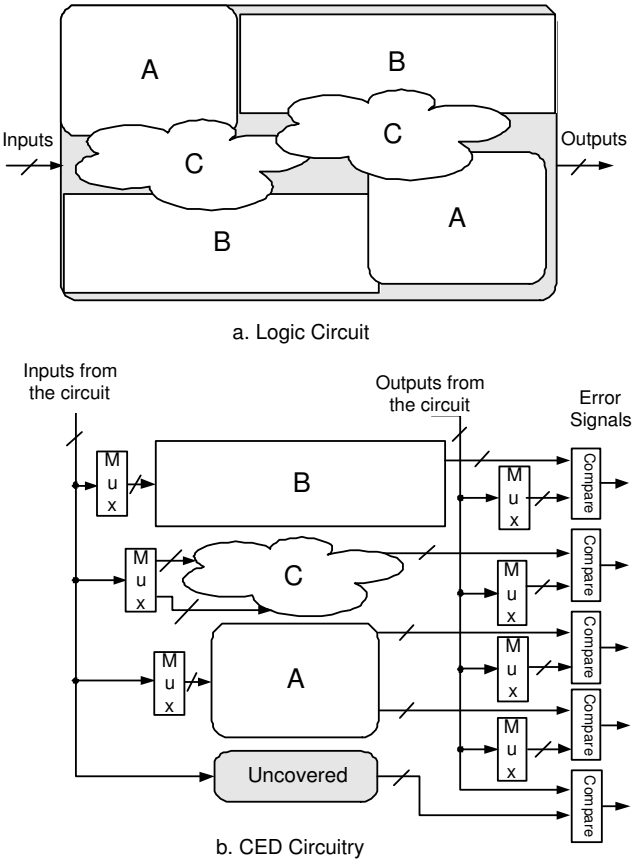


Figure 1. Roving CED for Logic Circuits

2. Prior Work

Previous research efforts within the area of CED have investigated the reduction of area overhead at the cost of fault coverage. A technique was proposed in [4] to *disable* CED for a *preselected subset* of the inputs of the circuit. The parity of the outputs is computed and compared against a predicted parity for the outputs. Disabling the CED comparison introduces don't care conditions in the parity functions that allow further logic optimization of the parity predictor. CED circuitry is added to a circuit in [5] by partially replicating a *subset* of the circuit based on the error susceptibility of the gates. The soft error rate is estimated for every gate based on three masking factors: logical, electrical and the latching-window. A heuristic algorithm is proposed that generates a CED cutset with a pre-specified area overhead and gates with the largest estimated soft error rate. Unfortunately, both of these techniques have an incomplete fault coverage for even *permanent* faults. A fault that is only activated using an input combination that disables CED in [4] or not included in the cutset of [5] will *never* be detected.

The guarantee of fault detection, albeit at an unbounded fault latency, can only be made if every part of a circuit is tested at *some* point in time. The only CED technique that we are aware of for logic circuits that trades fault coverage for area overhead, and yet guarantees the detection of all permanent faults, was proposed in [6]. The idea of roving test has been first proposed in [7] as a built-in test methodology for detecting and locating faults in a digital system. A hardware device, the Roving Emulator (RE), is connected to a set of digital modules by a bus. To test a module, the RE is configured to perform the operation of the module being emulated in an off-line manner. On-line testing using roving test has been proposed in [8] for FPGAs. The FPGA is divided into a roving part and a testing part. The testing part of the FPGA is dynamically reconfigured to test the working

part without disturbing the functionality of the FPGA. The testing part is swaped with a portion of the working part to ensure the correctness of the testing part and, thus, guarantee fault-free and complete testing of the chip.

The CED method proposed in [6] utilizes multiple reconfigurable SRAM-based Lookup Tables (LUTs) to perform duplicate computation for all the logic cones with a prespecified number of inputs. The results of the LUTs are compared against the actual results of the emulated logic cones in the circuit. The LUTs rove between all the logic cones in the circuit to guarantee complete fault detection. The proposed methodology shares some similarities with the technique proposed in [6]. Both methods perform CED for logic circuits using a roving logic block. However, the method proposed here is different in the following:

- The LUTs in [6] have a fixed size and are reconfigured to perform duplicate computation for heterogeneous logic cones. In contrast, the proposed method produces circuit-specific substructures that have variable number of inputs, outputs and internal structure.
- In [6], the area overhead of the multiplexing logic is ignored. In some of the benchmarks considered, a LUT performs duplicate computation for more than a hundred logic cones. In this case, the area overhead of the multiplexors is much larger than the area overhead of the LUTs themselves.
- Since the multiplexing logic was not considered in [6], the time to multiplex the inputs was not considered. Moreover, the delay of a LUT is relatively large compared to the delay of the original circuit. In some benchmarks, the clock cycle time increases by 70% to account for the slow operation of the LUT. The proposed method here controls the delay of the CED circuitry so that it never exceeds the delay of the original circuit.
- The fault latency increases as the ratio of logic cones to LUTs increase. The latency is further extended by the reconfiguration time of a LUT; around a couple of microseconds. The current clock frequencies of logic circuits results in thousands of operations not checked. The roving logic in the proposed technique tests two structures only and requires no reconfiguration time.

3. Roving CED for Random Logic

A logic circuit can be represented as a $typed$ Directed Acyclic Graph (DAG) $G(V, E)$. The set of vertices, V, includes a vertex and type for every gate in the circuit, while the set of edges, E, includes a directed edge for every connection between two gates in the circuit. The circuit in figure 2.a has the DAG representation in figure 2.b and will be used to illustrate the proposed method throughout this section.

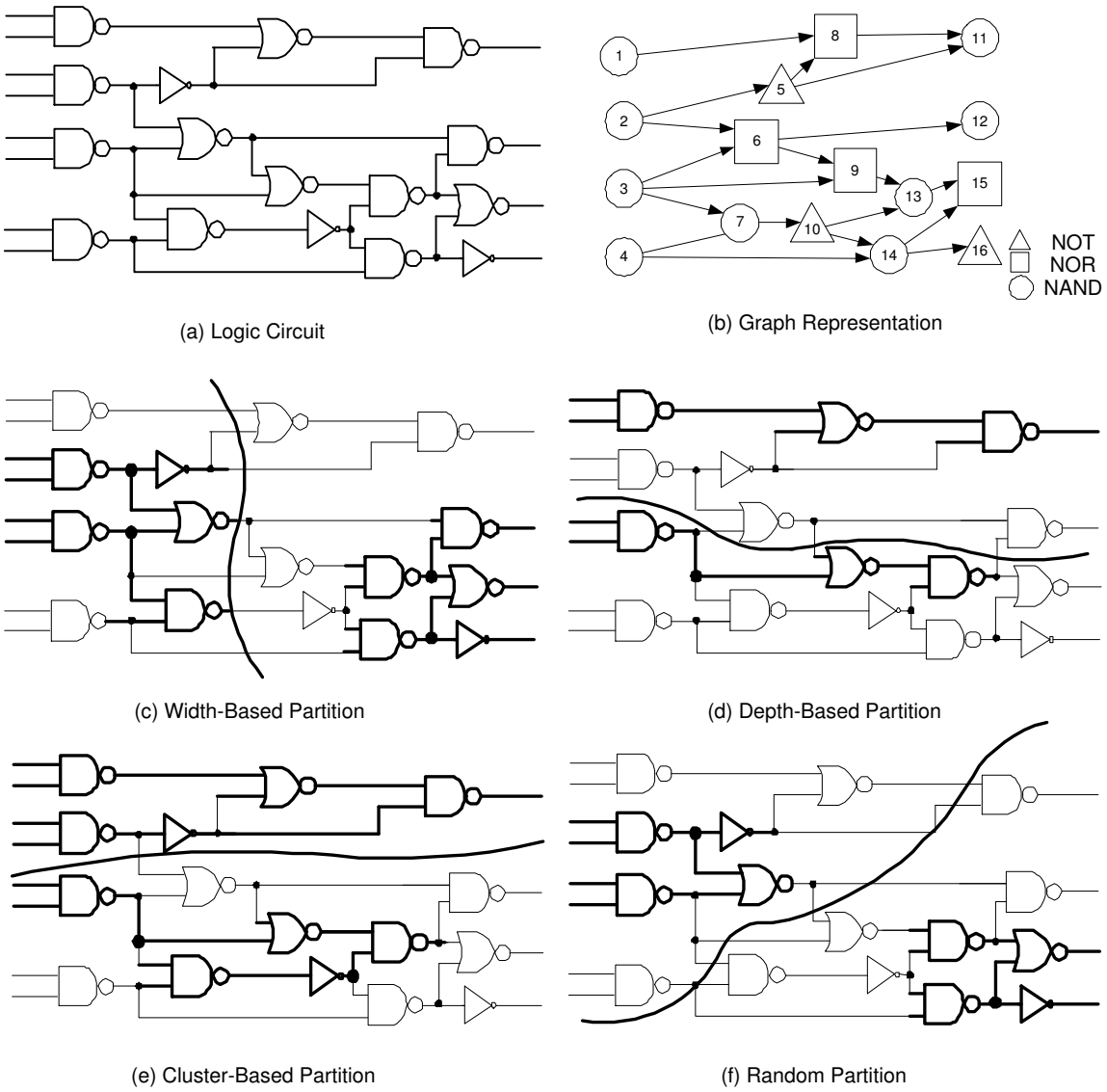
3.1. Maximal Common Substructures Extraction

The inclusion or exclusion of a substructure into the CED circuitry depends on three parameters: the number of inputs and outputs, the number of distinct covered gates and the depth of the substructure. *First*, the number of input and output multiplexors depends on the number of inputs and outputs of a substructure. The cost of a multiplexor depends on the number of instances a substructure will rove between. We utilize pass-transistors [9] as multiplexors in order to reduce the cost of the multiplexing logic. Pass-transistors with many control lines have a large delay and greatly reduces the signal-to-noise ratio. Consequently, we only consider the use of 2-to-1 multiplexors and, hence, every substructure is limited to rove between *two* instances only. *Second*, all of the selected substructures should cover distinct gates so that a maximum saving of 50% can be achieved in the area overhead, excluding the cost of the multiplexors and comparators. *Finally*, a substructure with a short critical path does not increase the delay of the original circuit. Moreover, substructures with a delay lower than half the delay of the original circuit can be used to test both instances within the same clock cycle, as will be discussed later in section 3.2.1.

The identification of a substructure that appears twice in a circuit corresponds to finding a subgraph, H, in G that is isomorphic to at least two instances of H in G. In order to achieve the maximum reduction in area overhead, H's with the maximum number of nodes are selected. The problem reduces to finding the Maximal Common Subgraph Isomorphism (MCSI) that appears twice within G. MCSI is used in various applications such as medical and molecular biology [10, 11]. In these applications, MCSI is used in the similarity search of a database of compounds, represented using molecular graphs, for those compounds that are most similar to some specified compound. More formally, the MCSI between two graphs consists in finding a common subgraph between two graphs which is not a proper subgraph of any other common subgraph between the two graphs [11]. Using the graph representation of the logic circuit G, we *partition* G into two graphs, G_1 and G_2, and search for the MCSI between the two graphs. The partitioning alternatives are discussed in the next section.

3.2. Circuit Partitioning

The substructures of the CED circuitry are identified as a result of the partition choice of the logic circuit. The number of all possible partitions is exponential in the number of gates in a circuit. To reduce the search of common substructures in all these possible partitions, we only consider partitioning the circuit in ways that reduce the CED overhead of the common substructures. Several partitioning choices are described in the following sections.



(a) Logic Circuit

(b) Graph Representation

(c) Width-Based Partition

(d) Depth-Based Partition

(e) Cluster-Based Partition

(f) Random Partition

Figure 2. Circuit Partitioning Alternatives.

3.2.1 Width-Based Partitioning

In order to ensure that the maximum delay of the CED logic does not exceed the delay of the original circuit, partitions are generated with a smaller depth than the original circuit. The delay of a substructure based on this method, illustrated in figure 2.c, can never exceed the delay of the original circuit. The delay of the CED circuit is the sum of the delay of the input and output multiplexors, the delay of the substructure with the longest critical path and the delay of the comparator. If the delay of the CED circuit is less than half the delay of the original circuit, the CED circuit can implicitly rove between the two instances of all the substructures. In this case, the CED circuit is double clocked and the *complete* set of permanent faults in the circuit will be detected every cycle. We only consider partitioning here as a way to reduce the delay of the CED circuit; performing multiple tests within a cycle is part of our ongoing research.

3.2.2 Depth-Based Partitioning

The cost of multiplexing between the two instances that match a substructure depends on the number of inputs and outputs in the substructure. Partitioning the circuit into subcircuits that have a lower number of inputs implicitly reduces the number of multiplexors of the maximum common substructure between the two partitions. A partition example is illustrated in figure 2.d where the common substructures identified has a small number of inputs and outputs.

3.2.3 Cluster-Based partitioning

One way to maximize the size of common substructures is to partition the circuit into two strongly connected clusters (subcircuits) and then perform the MCSI search between the two clusters. Strongly connected clusters, illustrated in figure 2.e, are nearly-isolated from each other, hence, a partition

that cuts between the two clusters would have a higher probability of finding large common substructures. Many methods has been proposed to partition a circuit into two clusters [12, 13] and multiple clusters [14] in order to reduce the layout area of a circuit. These partitioning methods will be applied to partition the logic circuit and search for the largest common substructure between the clusters.

3.2.4 Random Partitioning

The high complexity of the previous partitioning methods motivates partitioning the circuit randomly. A random partition, presented in figure 2.f, can be restricted to generate partitions that are almost equal in size, within a predefined equality metric, in order to maximize the common substructures. In addition, random partitioning is able to escape local minimas that are un-avoided using any of the previous deterministic partitioning techniques.

3.3. Proposed Algorithm

In order to reduce the CED overhead, the substructures must be disjoint and completely cover all the gates in the original circuit. Since the number of solutions is exponential, we only consider solutions containing the largest possible substructures. Therefore, every iteration of the algorithm, as indicated in Algorithm 1, partitions the graph, based on one of the mentioned partitioning techniques, and finds the MCSI subgraph (substructure) between the two partitions. The process is repeated for a number of times and the *best* subgraph, in terms of the savings in CED overhead, is recorded. The inclusion of the substructures in the CED circuitry is repeated until either the logic circuit is completely covered *or* no new substructure exists that provides any savings. In the latter case, all the gates that are not covered in any substructure are duplicated in the CED circuitry as a separate substructure.

Data : Graph G
Result : Subgraphs H_ℓ's that cover G
Beginning:
$DO\ (i = 1 : 1000)$
  $Partition(G, G_1, G_2)$;
  $Find\_MCSI(G_1, G_2, H_i)$;
  $Record\_Best(H_i, H)$;
if *No Solution Exists* **then**
  $Include\_Uncovered\_Gates(H_\ell)$;
  $Exit$;
else
  $Add(H, H_\ell)$;
  $Mark\_Nodes\_Covered(H, G)$;
  $Goto\ \ Beginning$;
end

Algorithm 1: The overall algorithm

The proposed method explores the tradeoffs between fault coverage, area overhead and detection latency. Moreover, the proposed method controls the delay of the CED circuitry and may permit multiple tests of the matching instances within the same clock cycle. A promising extension would be to select low-power substructures with inputs that have a biased probability of being set to either zero or one. Consequently, the proposed CED method explores fault coverage, area overhead, detection latency, delay of the CED circuitry and power consumption. The purpose of our ongoing research is to gain an insight of the interaction between these parameters on the quality of the final solution. The observations and results of our experiments will be provided in the workshop.

References

[1] S. J. Piestrak, "Self-checking design in Eastern Europe," *Design and Test of Computers*, vol. 13, no. 1, pp. 16–25, 1996.

[2] M. Nicolaidis and Y. Zorian, "On-line testing for VLSI - a compendium of approaches," *Journal of Electronic Testing: Theory and Applications*, vol. 12, no. 1-2, pp. 7–20, 1998.

[3] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?," in *International Test Conference*, 2000, pp. 985–994.

[4] K. Mohanram, E. S. Sogomonyan, M. Goessel, and N. A. Touba, "Synthesis of low-cost parity-based partially self-checking circuits," in *International On-Line Test Symposium*, 2003, pp. 35–40.

[5] K. Mohanram and N. A. Touba, "Cost-effective approach for reducing soft error rate in logic circuits," in *International Test Conference*, 2003, pp. 893–901.

[6] V. V. Kumar and J. Lach, "Heterogeneous redundancy for fault and defect tolerance with complexity independent area overhead," in *International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2003, pp. 571–578.

[7] M. A. Breuer and A. Ismaeel, "Roving emulation as a fault detection mechanism," *IEEE Transactions on Computers*, vol. 35, no. 11, pp. 933–939, 1986.

[8] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, and V. Verma, "Using roving stars for on-line testing and diagnosis of fpgas in fault-tolerant applications," in *International Test Conference*, 1999, pp. 893–901.

[9] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, Addison Wesley, 1993.

[10] H. Grindley, P. Artymiuk, D. Rice, and P. Willett, "Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm," *Journal of Molecular Biology*, vol. 229, pp. 707–721, 1993.

[11] G. Valiente, *Algorithms on Trees and Graphs*, Springer, 2002.

[12] Y. C. Wei and C. K. Cheng, "A two-level two-way partitioning algorithm," in *IEEE International Conference on Computer Aided Design*, 1990, pp. 516–519.

[13] D. Cheng, C.-C. Lin, and M. Marek-Sadowska, "Circuit partitioning with logic perturbation," in *IEEE International Conference on Computer Aided Design*, 1999, pp. 163–167.

[14] J. Garbers, H. Promel, and A. Steger, "Finding clusters in vlsi circuits," in *IEEE International Conference on Computer Aided Design*, 1990, pp. 520–523.