Manuscript to Appear in the 2005 International Conference on

Parallel and Distributed Processing Techniques and Applications (PDPTA'05), June 2005, Las Vegas, Nevada.

Self-Checking Fault Detection using Discrepancy Mirrors

R. F. DeMara and C. A. Sharma
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450

Abstract

A method for robust detection of faults is developed based on pairwise parallel evaluation using Discrepancy Mirror *logic. Discrepancy Mirrors provide coverage for the fault detector elements within the same mechanism used for the functional logic under test. The detector logic is self-testing and propagates functional outputs with adherence to a* single fault-secure *property so that erroneous outputs from any single fault are not propagated. Within the detector, bitwise equality comparisons are employed directly without additional data encoding/decoding schemes to determine the validity of the output. Fault handling is performed using the underlying data throughput so that additional test vectors are not required. The circuit was implemented for a Xilinx Virtex II Pro FPGA platform and fault-secure operation was verified using ModelSim-II for exhaustive stuck-at scenarios. Results indicate fault isolation in a pool of 100,000 resources using an expected value of 17.6 to 64.1 pairings when as little as one half of the inputs applied articulate the fault.*

Keywords: fault-handling, concurrent error detection, self-checking fault detection, fault-secure circuit, Xilinx Virtex FPGA

1 Introduction

Robust fault detection is central to the problem of enhancing the fault-handling capabilities of digital circuits. A common limitation facing many fault detection schemes is that the failure detector itself may fail. A fault involving the checker may be undetectable or result in the corruption of otherwise valid outputs. Traditional approaches to fault-detection typically rely on coding-based schemes or redundancy using a single *voter*, *comparator* or *error detector*. Those fault checkers possess a single point-of-failure exposure involving the detector elements, or must rely upon special test-vectors or data encodings to isolate them. Detector components in the reliability path have been referred to as *golden elements* [1] because the fault-handling strategy relies on them to be fault-free. This paper develops an alternative approach to self-checking fault detection based on random pairings and temporal voting to reduce such exposures.

Table 1 lists characteristics of selected fault-handling strategies. Specialized encoding schemes are often required by Concurrent Error Detection (CED) approaches as opposed to Triple Modular

Redundancy (TMR) and the Discrepancy Mirror methods which do not. The number of functional logic elements required by TMR is greater than that of the other schemes. Discrepancy Mirrors provide inherent transient fault coverage with minimal detection latency. They also support fine-grained resolution of the fault location, without interruption to the data throughput flow when a fault occurs. Thus, Discrepancy Mirrors offer improved detection of permanent and transient faults, with reduced time and space overheads. A detailed comparison of existing techniques is provided in Section 1.1 and Section 1.2. Section 1.2 provides the design of the discrepancy mirror approach. Results of simulations and fault location experiments are given in Section 3.1 and Section 3.2 respectively.

	Special Encoding	**Number of Elements**	**Transient Fault Detection**	**Detection Latency**	**Special Test Inputs**	**Fault Isolation Resolution**	**Interrupt-Free Testing**
Temporal CED Schemes	Yes	1	Optional	Algorithm Dependent	Yes	Sub-circuit Under Test	No
Spatial CED Schemes	Yes	≥ 2	No	Algorithm Dependent	Yes	Sub-Circuit Under Test	No
TMR	No	3	No	Minimal	No	Voter Elements	No
Discrepancy Mirrors	*No*	*2*	*Yes*	*Minimal*	*No*	*Individual Logic Resources*	*Yes*

Table 1: Comparison of Fault-Detection Techniques

1.1 Concurrent Error Detection Schemes

CED schemes include a wide array of *coding-based* [2] [3] and *redundancy-based* mechanisms [4]. Coding-based mechanisms typically employ mathematically robust encoding and decoding schemes. Redundancy-based mechanisms generate concurrent output values using multiple instances of the functional logic which are then checked for equality. CED schemes utilize the functional throughput that realizes the output $f(i)$ in response to input i to evaluate a diagnostic characteristic computed by an alternate yet predictable function, $f_d(i)$. The diagnostic characteristic used could be the parity of the output, conformance to specified encodings such as Berger codes [5], equality between $f(i)$ and $f_d(i)$, or a specific *m-out-of-n* code. They can be generated using dedicated logic to implement $f_d(i)$ directly or parallel evaluation with a comparator realizing a bitwise `XNOR` equivalence logic function. The redundancy within CED schemes can be *spatial*, where the hardware circuit-under-test is duplicated, or *temporal*, where the outputs are computed in succession for later comparison.

Consider the hardware comparators capable of bitwise output comparison, such as the *equality checker* [6] and the *two-rail checker* [7]. Equality checkers compare two input words to determine bitwise equivalence. These circuits require complemented inputs and a multi-rail representation to be verifiable as a self-testing system. A *two-rail checker* checks that each pair of inputs has complementary values which are used to convert n pairs of signals into one pair of signals that are complementary if and only if all of the n input pairs are complementary. It can be made testable with the addition of a test signal and two XOR gates [6]. Coding-based checkers detect invalid code words at the output of the circuit-under-test for at least one valid code word input to the checker.

1.2 Triple Modular Redundancy Systems

TMR approaches rely on three parallel instances of the functional logic to compute the output in triplicate and a majority voting element to determine consensus. TMR also supports correction of single faults by disregarding the incongruent output. This works satisfactorily in the case of a single-fault assumption, unless the fault induces a *Common Mode Failure* whereby two incorrect outputs agree using *bitwise* [8] or *word-width voters* [9]. If the logic elements within the voter fail then the fault handling system can fail. The use of redundant voters is suboptimal as it does not guarantee that their consensus is computed correctly.

2 Discrepancy Mirror Approach

The Discrepancy Mirror approach is a duplex redundancy technique that utilizes alternate physical configurations from a population of candidate designs that are functionally equivalent. As shown in Figure 1, the technique is composed of three phases, namely *Selection*, *Detection*, and *Preference Adjustment*. The Selection phase selects two candidates, each of which utilize mutually exclusive subsets of the resources under test. The Detection process uses the *Discrepancy Mirror* logic shown in Figure 2 to check for bit-wise equivalence between outputs of the candidates as described below. The Preference Adjustment phase utilizes the results of successive comparisons to update the pairing strategy during subsequent selections. These steps will be explained below in the context of a Field Programmable Gate Array (FPGA) based realization whereby two configurations of the functional logic are loaded in tandem.

2.1 Selection Phase

Candidate designs are selected from a population developed at design time, either manually or via a CAD tool. Random pairings or an adaptive scheme based on the results of Preference Adjustment can be employed. This process is identified as Step *1* in Figure 1. The selected designs are then loaded as the active configurations during Step *2* and Step *3*. Identical input operands are applied in parallel to each configuration and the outputs are redundantly computed for comparison in the next phase.

2.2 Detection Phase

As shown in Step *4* in Figure 1, the discrepancy mirror circuit is used to identify whether the outputs of the two configurations under test agree. A complete instance of the discrepancy mirror is obtained whenever two configurations are loaded, since the discrepancy detector consists of two identical sections as shown in Figure 2. Assertion of `MATCH` output from the discrepancy mirror indicates the absence of a single-fault in the configurations under test, as well as the logic in the

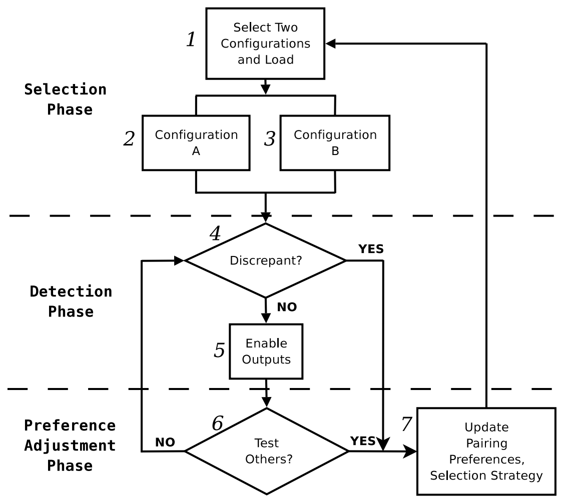

Figure 1: Discrepancy Mirror-based Scheme

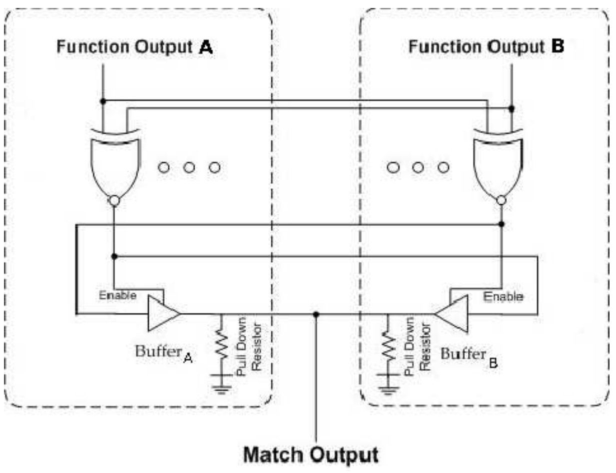

Figure 2: Discrepancy Detection Circuit

discrepancy mirror. The data output is enabled if and only if no faults are detected as shown in Step *5* in Figure 1.

The inputs to the Discrepancy Mirror shown in Figure 2 as "Function Output A" and "Function Output B" are generated independently. If there is a fault in a resource utilized by either of these configurations, then a discrepancy is observed at the output. The truth table shown in Table 2 describes the operation of the circuit shown in Figure 2. Outputs from the function configurations A and B are applied as inputs to both the XNOR gates. The output from each XNOR gate acts as the `ENABLE` signal for the tristate buffer in the same half, as well as the input to the tristate buffer in the other half of the discrepancy mirror. The tristate buffer outputs are tied together to form a Wired-OR connection which provides the `MATCH` output signal. The pull-down transistors hold the signal to a logic '0' level when the tristate buffer output is in a high-impedance state. In an active-high non-inverting tristate buffer, the input is buffered at the output only when the `ENABLE` signal is high. When the `ENABLE` signal is low, the output of the buffer is in a high-impedance state.

As listed in Table 3, the response of the circuit is robust to several possible fault scenarios. If either of the XNOR gates fail, then one of the two tristate buffers will be disabled and the other will have an input of zero, thus `MATCH` will be a '0', signifying discrepancy. If the tristate buffers fail, producing a high impedance output, the pull down resistors in the circuit will hold the signal to '0'. The wired-OR connection reduces single points of failure to a stuck-at fault exposure for the `MATCH` output.

Table 2: Discrepancy Mirror Truth Table

A	B	$XNOR_A$	$XNOR_B$	ENB_A	ENB_B	TRI_A	TRI_B	**Match**
0	0	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1

2.3 The Preference Adjustment Process

Step *6* and Step *7* comprise the Preference Adjustment process. When the Discrepancy Mirror returns a `MATCH` output, alternate configurations can be loaded for testing or the resident fault-free configurations can be used. The output from the discrepancy mirror over a period of time indicates the relative fitness of the different configurations. This information can isolate the fault location and aid in regeneration of lost functionality through the identification of alternate resources. The cumulative discrepancy information from diverse pairings over time can be used in Step *7* to modify the selection preferences for the configurations in the population.

3 Results

3.1 Hardware Simulation and Verification

The operation of the discrepancy mirror circuit was verified on a Xilinx Virtex-II Pro FPGA platform using ModelSim-II. The pull-down resistors were emulated using digital components as shown in the Xilinx data sheet [10]. The waveform for the `MATCH` output was asserted whenever the inputs to the discrepancy mirror were in agreement. The simulation waveform showed a `LOW` signal whenever the `MATCH` output was a '0'. In the Xilinx Virtex-II Pro FPGA, when pull-down resistors are emulated, a `LOW` signal is the equivalent of a logic-0 output. The circuit was also simulated using Cadence SPICE. The entire circuit was also realized using a total of 44 p and n-channel MOS transistors using a 1.5 micron minimum width technology with a length of 600 nm. A total of 44 CMOS transistors were utilized to realize the circuit. The widths of the pMOS transistors in the XNOR circuit were selected to be thrice the widths of the nMOS transistors to shape the waveform rise and fall times, to develop the required timing characteristics. The simulation results and waveforms obtained indicated behavior conforming to Table 2 and Table 3.

3.2 Fault Location Experiments

Two sets of experiments were performed to analyze the fault isolation latency. Both experiments sought identify the number of iterations required to identify the faulty resource in the case of single fault. A monte carlo simulator was constructed using a C-language program for simulating the Selection, Detection, and Preference Adjustment phases. The inputs to the simulated mirror were obtained using random number generators. Random input values were applied to two configurations chosen at random from the pool of competing configurations. More formally, let U denote the set consisting of all the logic resources in the FPGA, S denote the pool of resources suspected of being

Table 3: Discrepancy Mirror Fault Coverage and Response

Component	**Fault Scenarios**				**Fault-Free**
Function Output A	*Fault*	Correct	Correct	Correct	Correct
Function Output B	Correct	*Fault*	Correct	Correct	Correct
XNOR_A	Disagree(0)	Disagree(0)	*Fault :Disagree(0)*	Agree (1)	Agree (1)
XNOR_B	Disagree(0)	Disagree(0)	Agree (1)	*Fault: Disagree(0)*	Agree (1)
Buffer_A	0	0	High-Z	0	1
Buffer_B	0	0	0	High-Z	1
Match Output	*0*	*0*	*0*	*0*	*1*

faulty, and $C_i \subseteq U$ denote the set of resources used by the i^{th} configuration. Initially, $|S| = |U|$. A process of m successive intersections among the subsets $C_j \bigcap C_{k \neq j} (1 \leq j, k \leq m)$ are performed. Each successive intersection reduces $|S|$ until after the m^{th} intersection at time $t = m$ eventually $|S| = 1$, completing the fault-location process. Each experiment was conducted with $|U| =$ 1,000, 10,000, and 100,000. The expected number of iterations to isolate the fault are reported for the mean values observed over 100 trials of the simulator. An individual logic resource is the equivalent of a *Configurable Logic Block (CLB)* in an FPGA so the range of resource pool sizes reflect a realistic device scenario.

3.2.1 Analysis with Perpetually Articulating Inputs

In this experiment, the inputs applied consistently articulate any fault in the logic resources used by the configurations under test. Thus, a match output indicates that the logic resources used by the configurations being compared are completely fault-free. A discrepancy between the configurations' functional outputs indicates the presence of at least one resource fault. Assertion of the `MATCH` output exonerates all logic resources currently being used, and a de-assertion of the `MATCH` output implicates the subset of logic resources currently being used as suspect. The faulty resource is isolated after m pairings through a process of successive intersection. Figure 3 shows the faulty resource can be identified using an expected value of 11.1 pairings when $|U| =$ 1,000 and half of the resources are utilized by each configuration. When $|U| =$ 100,000, the mean number of pairings required to locate the fault increases by much less than a factor of ten to a value of 17.6. Under more demanding parameters, when $|U|$ =100,000, and when only 5% of the resources are being used by each configuration, a mean value of 63.7 pairings are required to isolate the faulty resource.

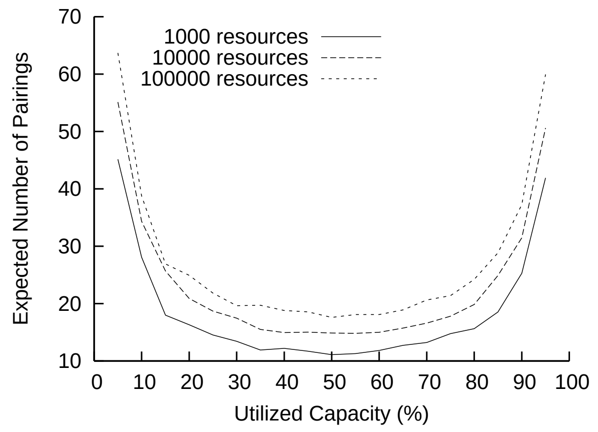

Figure 3: Fault Isolation with Perpetually Articulating Inputs

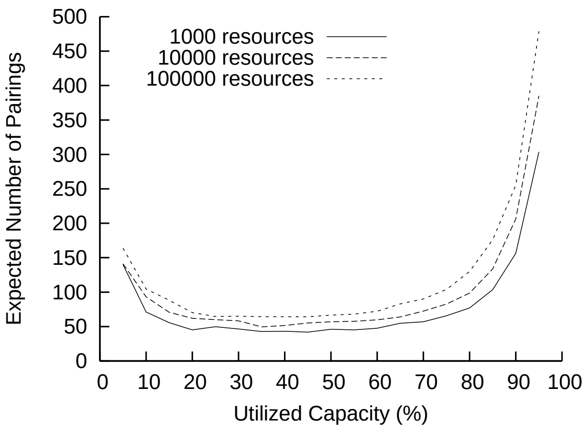

Figure 4: Fault Isolation with Intermittently Articulating Inputs

3.2.2 Analysis with Intermittently Articulating Inputs

Depending on the inputs applied, the fault in the functional logic under test may remain dormant and thus some inputs would not articulate a visible discrepancy. In this case, a match output from the discrepancy mirror cannot evaluate whether all the resources under test are fault-free. A discrepant output is a definitive indicator of the existence of a single-fault. With such *Intermittently Articulating Inputs*, the discrepancy mirror based scheme requires additional random pairings to isolate the single-fault. When $|U| = 1,000$, with resource utilization at 45%, an expected 42 random pairings are required to uniquely identify the faulty resource. When $|U| = 100,000$, the best performance is observed for a utilization of near 50%, where the expected value of random pairings is 64.1.

4 Conclusion

The discrepancy mirror is capable of handling faults in either the functional logic or the detector. If there is a failure in either, then the output of the mirror remains de-asserted indicating the presence of at least one resource fault.It is able to isolate the faulty resources with a expected number of random pairings that is sub-linear in the number of resources under test. It does not depend upon a specific coding scheme or a pre-defined set of inputs. Random pairings of configurations perform successive intersection of the resources under test to isolate the faulty resource. Figure 3 and Figure 4 show that more pairings are required to identify the faulty resource when the utilization of available resources is below 20% or above 80%. In these situations, each successive pairing implicates (or exonerates) a smaller sub-set of resources than when half of the resources are utilized. Finally, using a discrepancy mirror based approach, the number of pairings required for fault location increases sub-linearly with an increase in $|U|$. For example, at 50% utilization, the expected number of pairings to locate a fault within pools of 1,000, 10,000, and 100,000 resources are 11.1, 14.9, and 17.6, respectively, demonstrating the viability of the technique.

References

[1] M. Garvie and A. Thompson, "Scrubbing away transients and jiggling around the permanent: Long survival of fpga systems through evolutionary self-repair," in *Proc. 10th IEEE Intl. On-Line Testing Symposium* (C. Metra, R. Leveugle, M. Nicolaidis, and J. Teixeira, eds.), pp. 155–160, IEEE Computer Society, 2004.

[2] N. A. Touba and E. J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 783–789, July 1997.

[3] K. Mohanram, E. S. Sogomonyan, M. Gossel, and N. A. Touba, "Synthesis of low-cost parity-based partially self-checking circuits," in *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pp. 35–40, July 2003.

[4] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?," in *Proceedings International Test Conference 2000*, p. 985, IEEE, October 2000.

[5] D. L. Tao, R. P. Hartmann, and P. K. Lala, "An efficient class of unidirectional error detecting/correcting codes," *IEEE Trans. Comput.*, vol. 37, no. 7, pp. 879–882, 1988.

[6] E. J. McCluskey, "Design techniques for testable embedded error checkers," *IEEE COMPUTER*, vol. 23, pp. 84–88, July 1990.

[7] J. L. A. Hughes, E. J. McCluskey, and D. J. Lu, "Design of totally self-checking comparators with an arbitrary number of inputs," *IEEE Trans. Computers*, vol. C-33, pp. 546–550, June 1984.

[8] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems:Design and Evaluation.* Digital Press, 1992.

[9] S. Mitra and E. J. McCluskey, “Word voter: A new voter design for triple modular redundant systems,” in *18th IEEE VLSI Test Symposium*, p. 465, April 2000.

[10] Xilinx Inc., *Virtex-II Pro and Virtex-II Pro X Platform FPGAs : Complete Data Sheet*, November 2004.