Permanent Fault Repair in FPGAs through Graceful Degradation

Shu-Yi Yu and Edward J. McCluskey
CENTER FOR RELIABLE COMPUTING
Stanford University, Stanford, CA94305-4055
{syu, ejm}@crc.stanford.edu

Abstract

Conventional FPGA fault repair schemes remove faulty elements from designs through reconfiguration. In designs with high FPGA utilization, routable fault-free elements may not be available for permanent fault repair. We present a new permanent fault repair scheme, where the original design is reconfigured into another gracefully degraded fault tolerant design that has smaller area, so the damaged element can be avoided. Three new schemes that fully utilize available fault-free area and provide graceful availability degradation are presented.

1. Introduction

Permanent fault repair in FPGAs is accomplished by reconfiguring the chip so that faulty elements are not used. Many techniques have been presented to provide permanent fault removal through reconfiguration [Huang 01]. With these schemes, permanent faults in an FPGA can be repaired as long as there exist enough routable fault-free elements on the chip so that designs can avoid using faulty elements. However, for designs with high FPGA utilization or long mission time, all fault-free elements that are routable can be exhausted due to permanent fault repair. In this case, further permanent damage can no longer be repaired. To solve the problem, the design has to be reconfigured into a new fault tolerant design with smaller area, so that the permanent faults can be avoided. A traditional way is to remove design elements at the modular level, such as TMR/Simplex or self-purging. However, modular removal may introduce large degradation of system availability. For example, a duplex system has lower availability than a TMR system. In this paper, we present a new method of permanent fault repair in FPGAs to gracefully degrade system availability by reconfiguring the original design into a new fault tolerant design that fully utilizes the available fault-free FPGA area.

This paper is organized as follows. Section 2 describes error recovery flow in FPGAs, to explain needed features of an FPGA design. Section 3 defines the problem we are addressing, and presents three design candidates for permanent fault repair with graceful degradation of system availability. Section 4 compares these design candidates and selects a suitable one depending on application requirements. Section 5 concludes the paper.

2. Error Recovery in FPGAs

Figure 1 illustrates a general flow of error recovery in FPGAs. First, concurrent error detection (CED) mechanisms detect an error. Error history is used to differentiate permanent and transient faults. When a transient error occurs, the system recovers from corrupt data and resumes normal operation. Possible recovery schemes include rollback [Chandy 72] and roll-forward [Pradhan 92][Yu 01]. Rollback uses re-computation to obtain correct data. Roll-forward uses data in redundant modules for recovery; hence, it has much smaller delay than rollback. When a permanent fault occurs, the system halts. Fault diagnosis determines the location of the damaged resource, and a suitable configuration is chosen according to the available area. Then computation is resumed.

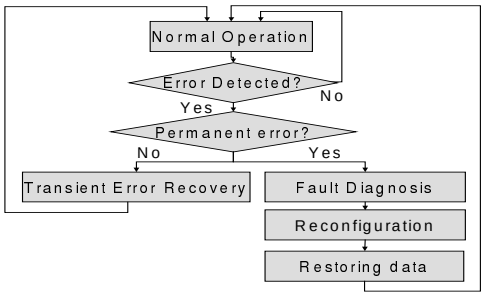

Figure 1. Error recovery in FPGAs.

3. Design Candidates

The problem we are solving here is how to find a configuration for a computing module given a limited FPGA area, so that system reliability and availability are gracefully degraded. Since TMR is widely used in fault tolerant systems [Siewiorek 00], we focus on degradation of TMR systems. For a TMR system, one way of permanent fault repair through module removal is to degrade it to a duplex system. We are going to compare our gracefully degraded designs with the module removal approach, which is denoted as TMR-Duplex throughout the paper. We present three different design candidates: (1) hybrid TMR-Simplex-CED, (2) duplex with a checker, and (3) duplex with two CEDs. These designs all contain the features needed for error recovery and are able to recover from single errors.

In a hybrid TMR-Simplex-CED design, we partition the original simplex design into two parts. One is triplicated, and the other is protected by CED. The partition depends on the available FPGA area and design constraints. As the area decreases, the design degrades from TMR to a simplex design with CED. When errors occur in the TMR part, roll-forward scheme is used for

©2001, Yu and McCluskey

recovery. When errors occur in the simplex part, rollback recovery is used.

In a duplex system with a checker, one of the original three modules in TMR is degraded to a checker. The checker implements a portion of the normal function of a module, and the portion size depends on the available FPGA area. We call the portion of the modules that is also implemented in the checker as the *checked portion*. The data from the checked portion in the duplex are compared with the data from the checker. And the two modules are compared with each other. When there is a mismatch between the duplex, the module that agrees with the checker is selected, and roll-forward is used to restore data in the other one. If there is no mismatch between the two modules but they disagree with the checker, roll-forward is used to restore data in the checker. When there is a mismatch between the duplex and they both agree with the checker, rollback is used.

In a duplex system with two CEDs, CEDs are built in both modules. The coverage of the CEDs depends on the available FPGA area. The degradation of CED is done by building it for only part of the modules instead of for the entire modules. In this case, the CED coverage decreases linearly as the area decreases. When there is a mismatch between the two modules, the module whose CED does not indicate an error is selected, and roll-forward is used to restore data in the other module. When there is a mismatch and both or neither of the CEDs indicates an error, rollback recovery is used.

4. Design Selection

We evaluate the design candidates with *Rollback Rate* and *Mean Time to Failure* (MTTF). Rollback rate is used to evaluate availability of designs. It is defined as the percentage of the time that a design spends in re-computation. For applications with strict deadlines, rollback delay can be an important issue. An analytic model is built to compute the metrics with respect to the available area. The area is normalized to the simplex area. The CED overhead is assumed to be 90% [Mitra 00], and detection coverage is 100% for single faults. To compute a lower bound of metrics, we assume that CED is not able to detect multiple faults. Errors are assumed to occur independently in each unit area in unit time. The permanent error rate is assumed to be 1/100 of the transient error rate. The results are shown in Fig. 2 and Fig. 3. These candidates are compared with TMR-Duplex. The metrics are normalized to those of a duplex system. As shown in Fig. 2, all design candidates have lower rollback rates than TMR-Duplex. Among all designs, hybrid-TMR-Simplex-CED has the lowest rollback rate. From Fig. 3, it is shown that TMR-Duplex has the longest MTTF among all. It is because although other designs have more fault tolerance features, the duplex design has smaller area; hence, it is less prone to have multiple errors that cause failure.

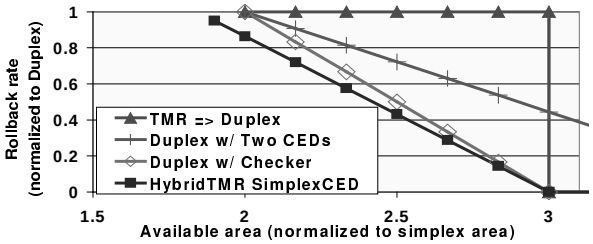

Figure 2. Rollback rate normalized to Duplex.

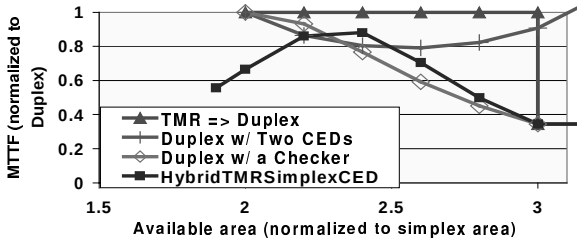

Figure 3. MTTF normalized to Duplex.

5. Conclusion

In this paper we presented a new scheme for permanent fault repair in FPGA-based computing systems. When no fault-free elements are available for permanent fault removal, the design is reconfigured into another fault tolerance design, which needs smaller area. We have examined three design candidates to adapt to limited FPGA area and to provide reliability and availability. These designs are found to have an improvement on availability compared to the module removal approach. For a CED with 90% overhead and single fault detectability, a hybrid TMR-Simplex-CED design is found having the lowest rollback rate; therefore, it is suitable for real-time applications. However, for non-real-time applications, the module removal approach is found to be the most suitable design.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. DABT63-97-C-0024.

References

[Chandy 72] Chandy, K. M. et al., "Rollback and recovery strategies for computer programs," *IEEE Trans. on Computers*, vol. C-21, No. 6, pp. 546-56, 1972.

[Huang 01] Huang, W.-J. and E. J. McCluskey, "Column-Based Precompiled Configuration Techniques for FPGA Fault Tolerance," to appear in *FCCM'01*, 2001.

[Mitra 00] Mitra, S. and E.J. McCluskey, "Which Concurrent Error Detection Scheme to Choose?" *Proc. 2000 Int. Test Conf.*, pp. 985-994, 2000.

[Pradhan 92] Pradhan, D. K., and N. Vaidya, "Roll-forward Checkpointing Scheme: Concurrent Retry with Nondedicated Spares," *IEEE Workshop on Fault Tolerant Parallel and Distributed Systems*, pp. 166-74, 1992.

[Siewiorek 00] Siewiorek, D. P. and R. S. Swarz, *Reliable Computer Systems – Design and Evaluation*, 3rd Ed, Digital Press, 2000.

[Yu 01] Yu, S.-Y. and E. J. McCluskey, "On-line Testing and Recovery in TMR Systems for Real-Time Applications", submitted to *ITC'01*, 2001.