Center for Reliable Computing

TECHNICAL REPORT

Permanent Fault Repair for FPGAs with Limited Redundant Area

Shu-Yi Yu

TR 01-2

May 2001

Center for Reliable Computing
Gates Building 2A, Room 236
Computer Systems Laboratory
Dept. of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Abstract:

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 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 degradation on availability are presented. Analytical results show that our schemes have an improvement on availability compared to a module removal approach.

Funding:

This research was supported by the Advanced Research Projects Agency under Contract No. DABT63-97-C-0024.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Abstract

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 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 degradation on availability are presented. Analytical results show that our schemes have an improvement on availability compared to a module removal approach.

1. Introduction

Field-Programmable Gate Arrays (FPGAs) provide solution to permanent fault repair in finer granularity because of their regular structure and reconfiguration capability. In FPGAs, a faulty module can be repaired by reconfiguring the chip so that the damaged configurable logic block (CLB) or routing resource is removed from a design. Many techniques have been presented to provide permanent fault removal for FPGAs through reconfiguration. One approach is to generate a new configuration after permanent faults are detected in computing systems. CAD tools and algorithms for fast FPGA re-routing and re-mapping are developed [Emmert 98] [Dutt 99]. Another approach is to generate pre-compiled alternative FPGA configurations and store the configuration bit maps in non-volatile memory, so that when permanent faults are present, a new configuration can be chosen without the delay of re-routing and re-mapping [Lach 99] [Huang 01a].

With the repair schemes mentioned above, permanent faults in an FPGA can be repaired as long as there are sufficient routable fault-free elements on the chip so that designs can avoid using faulty elements. However, for designs with high FPGA utilization or very long mission times, all routable fault-free elements could be exhausted due to reconfigurations for permanent fault repair. In this case, further permanent damage can no longer be repaired. To solve this 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 module level, such as TMR/Simplex [Mathur 75] or self-purging redundancy [Losq 76]. However, a system with redundant modules removed may have much lower availability than the original system. For example, consider a TMR system with error detection and a duplex system. When a fault occurs in each system, the TMR system can still function, but the duplex system has to be stopped for recovery or repair. 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 defines the problem we are addressing. Section 3 presents three design candidates for permanent fault repair with graceful degradation of system availability. Section 4 describes an analytic model of a system with error recovery, and describes metrics we use to evaluate the designs. Section 5 shows the analytical results and compares these design candidates. Section 6 concludes the paper.

2. Problem Definition

For an FPGA-based system, as permanent faults happen in the chip, the damaged area can be so large that the design no longer fits on the FPGA. Hence, the original structure has to be remapped into another fault tolerant design with smaller area. The problem we are solving here is how to find a configuration for a computing module given a limited FPGA area, so that the degradation of system reliability and availability from the original design is small. Since TMR is widely used in fault tolerant systems [Siewiorek 00], we focus on degradation of TMR systems.

To explain features required of an FPGA-based design, Fig. 1 illustrates the error recovery steps for FPGA-based systems. First, concurrent error detection (CED) mechanisms detect an error. If an error happens for the first time, it is treated as a transient error; if the error persists, it is treated as a permanent error. When a transient error occurs, the system recovers from corrupt data and resumes normal operation. 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.

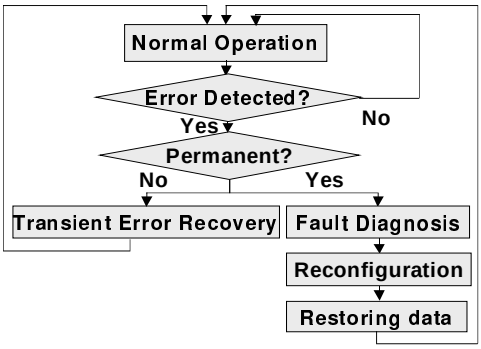


Figure 1. Error recovery in FPGAs.

Possible transient recovery schemes include *rollback* [Chandy 72] and *roll-forward* recovery [Long 90] [Pradhan 92] [Yu 01]. In rollback recovery, the system state is backed up to some point in its processing, which is called a *checkpoint*. When an error is detected, the system will restore its state back to the previous checkpoint and re-compute. However, re-computation has time overhead. In roll-forward recovery, the system will recover the corrupt data by copying correct data from fault-free redundant modules to the faulty module. Hence, re-computation delay is avoided.

From the error recovery flow, it is clear that CED and transient error recovery schemes are required for FPGA-based designs. We assume that fault diagnosis and reconfiguration control

mechanisms are available off-chip. The fault diagnosis and reconfiguration control mechanism can be implemented in another FPGA for fault tolerance and recovery purposes [Huang 01b]. CED and transient recovery mechanism are built on-chip within an FPGA-based design. In this paper, we present designs degraded from TMR with error recovery capability.

3. Design Candidates

We present three fault tolerant design candidates: (1) hybrid TMR-Simplex-CED, (2) duplex with a checker, and (3) duplex with two CED blocks. Figure 2 (a) shows a TMR system, and Fig. 2 (b), (c), and (d) illustrate the design candidates respectively. These designs all contain the features required for error recovery and are able to recover from single faults. Unlike conventional fault tolerant designs, these designs are adjusted according to the available FPGA area. When a fault happens, rollback or roll-forward recovery is used depending on the fault scenario and design structure.

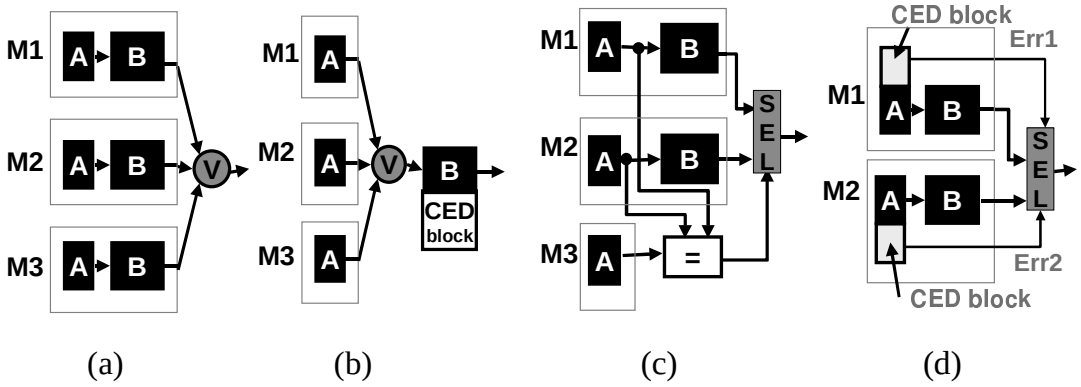


Figure 2. Design candidates degraded from TMR. (a) The original TMR design. (b) A hybrid TMR-Simplex-CED design. (c) A duplex system with a checker. (d) A duplex system with two CED blocks.

In a hybrid TMR-Simplex-CED design, we partition the original simplex design into two parts. As shown in the example of Fig. 2 (b), the design is partitioned into blocks A and B. A is triplicated, and CED, such as parity checker or diversed duplication [Mitra 99], is added to B. The area reduced by the design compared with TMR is approximately the area of block B. The partition depends on the

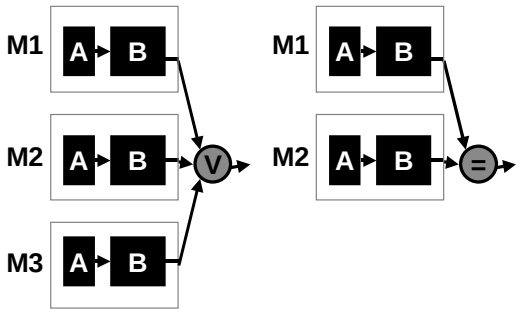
available FPGA area and design constraints. When errors occur in the partition with CED, rollback is used. Otherwise, when errors occur in the partition with TMR, roll-forward is used [Yu 01].

In a real design, some blocks are used more frequently than others. For a frequently used block, it needs to be more reliable since its fault is more likely to affect a whole system. The block usage needs further measurement and is application dependent [Elder 88]. It is not part of our paper.

In a duplex system with a checker, one of the original three modules in TMR is degraded to a checker module. The checker implements a portion of the normal function of a module, and the portion size depends on the available FPGA area. As shown in Fig 2 (c), M3 is degraded to block A, and the area reduced is approximately the area of block B. The data from blocks A of the duplex system 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 modules, 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 modules and they both agree with the checker, rollback is used.

In a duplex system with two CED blocks, error detectors are built into both modules. When the remaining area on an FPGA is no longer sufficient for adding error detection mechanism for a whole module, only part of the design is built with CED. For example, in Fig. 2 (d), only block A is built with CED. Detection coverage is degraded since the detection area is reduced. In the design, when there is a mismatch between the two modules, the module whose CED block 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 CED blocks indicates an error, rollback recovery is used.

We are going to compare our designs with a traditional module removal approach. For a TMR system, one approach to module removal for permanent fault repair is to degrade it to a duplex system, where duplication is used for error detection, and recovery is accomplished by rollback. We denoted the module removal approach as *TMR-Duplex* throughout the paper. Figure 3 illustrates the idea of TMR-Duplex.



(a) (b)

Figure 3. Permanent fault repair through a module removal approach, TMR-Duplex: (a) the original TMR design, and (b) duplex.

4. Evaluation Metrics

In this section, we describe metrics we use to evaluate the designs in Sec. 3. A Markov chain is used to model a general design with recovery mechanism and to derive equations to compute metrics. The model is established in a way that it can be used for all of the three designs.

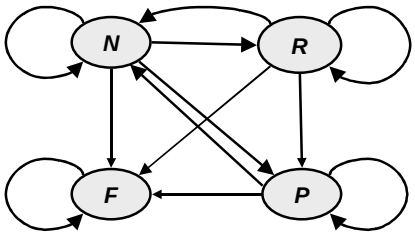


Figure 4. A Markov chain model of an FPGA-based system with recovery mechanism.

Consider a computing system with both rollback and roll-forward recovery capability. When a transient fault occurs, the choice of rollback or roll-forward depends on the location of the fault. Figure 4 shows a Markov chain model for the system. We define a *checkpoint period* as the period between two checkpoints. The system transits among states at the beginning of every checkpoint period.

The states are defined as follows:

State *N*: Normal state. In this state, the system functions normally.

State *R*: Re-computation state. The system rolls back to the previous checkpoint and re-computes to recover from transient faults.

State *P*: Permanent fault repair state. The system is stopped. Fault diagnosis and reconfiguration are done in the state.

State *F*: Fail state. The system fails and repair is not possible.

The system remains initially in State *N*. If no fault occurs during a checkpoint period, it will remain in the state. If a transient fault occurs but the system is recovered by roll-forward recovery, no re-computation is needed, so the system will stay in State *N* after the checkpoint. If a transient fault occurs and the system is recovered by rollback recovery, the system will go to State *R* for re-computation. In State *R*, during re-computation, if no fault occurs or a transient fault occurs and is recovered by roll-forward recovery, the system will go back to State *N* for the next checkpoint period. If a transient fault occurs and rollback is evoked again, the system will stay in State *R*.

From both State *N* and State *R*, if a permanent fault occurs, the system will go to State *P* for repair. When multiple faults occur and the system is unable to be either recovered or repaired, it fails and goes to State *F*. For a system under repair, when the repair is finished, the system will go back to its normal function and move from State *P* to State *N*. However, if repair fails, the system will move to State *F*.

We use the following metrics to evaluate a design:

Probability of rollback: it is the probability of system transition from State *N* to State *R*. It represents efficiency of the use of a design. In a real-time application, a high rollback probability is not acceptable, since deadline may be missed because of re-computation.

Mean time to failure (MTTF): it is the expected time of a system moving from State *N* to State *F*.

The derivation of the evaluation metrics is in the appendix.

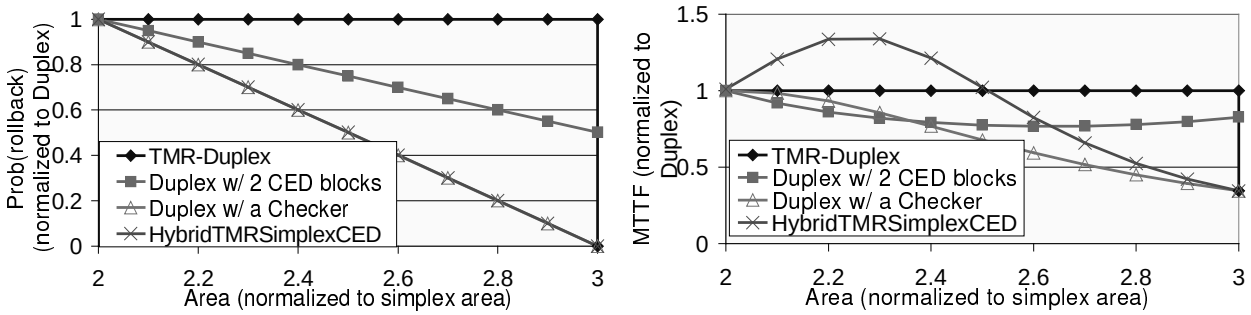
5. Comparison

In this section, we compute the evaluation metrics, rollback probability and MTTF, for different design candidates and compare them with the module removal approach, TMR-Duplex.

No experimental data on permanent error rates have been found in SRAM-based FPGAs yet. However, the experiment in [Ohlsson 98] showed that the permanent error rate is at least lower than 1/5 of the transient error rate for XC4010XL FPGA. In our analysis we choose the permanent error

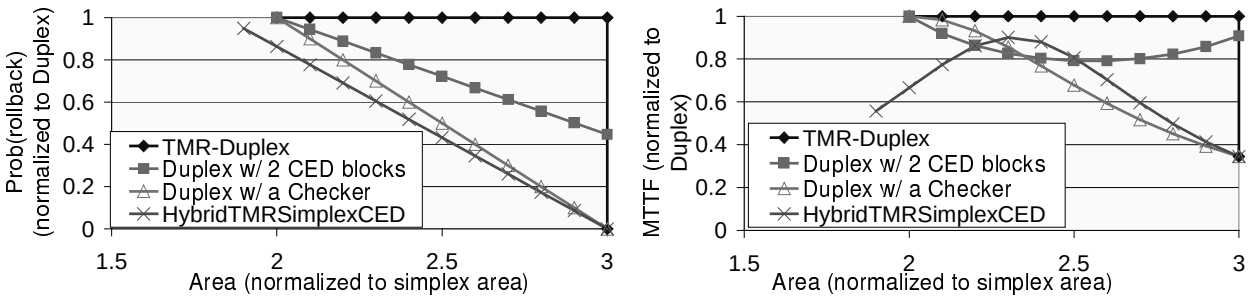
rate to be 1/100 of the transient error rate. CED overhead depends on the error detecting mechanism. For logical operations, parity prediction and duplex is widely used for error detection, and for arithmetic operations, arithmetic error detecting codes such as residue codes and AN codes are more commonly used [Wakerly 74]. The overhead of parity checker can be as high as 90% or even more than 100% [Mitra 00]. CED overhead of arithmetic codes varies a lot depending on the arithmetic function and bit width. In [Sparmann 96], the overhead of residue code with mod 7 checking for a 16-bit multiplier is as low as 31.1%. We examine the design candidates with three different CEDs: (1) duplex technique, (2) parity prediction with 90% overhead, and (3) arithmetic code with 30% overhead. We assume that the CED techniques have 100% single fault detection coverage. To calculate the lower bound of the metrics, we assume that multiple faults are not detectable. In real designs, design diversity [Mitra 99] and multiple-fault detection techniques can be used to increase CED coverage.

Figures 5 though 7 are the metrics plotted with respect to the total available FPGA area. The metrics are normalized to those of a duplex system. The area axis is normalized to the area of a simplex system. All designs are fully utilizing the available area except TMR-Duplex. In TMR-Duplex, the design area is twice the simplex area when the available area is smaller than three times the simplex area.



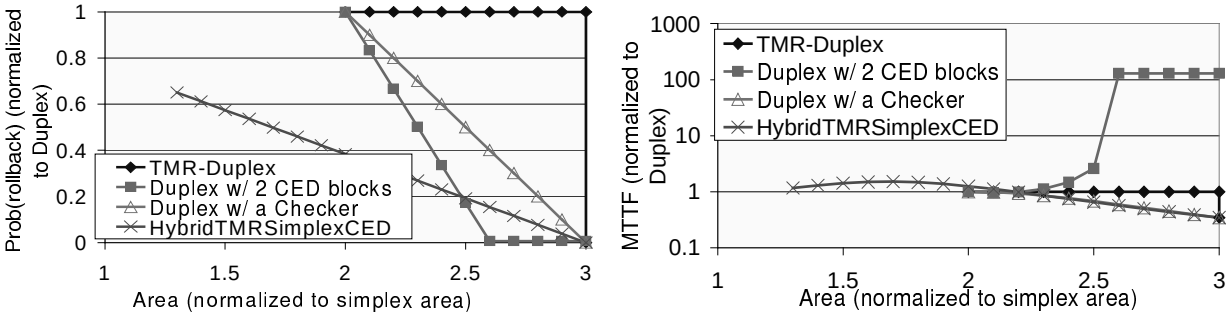
(a) (b)

Figure 5. CED with duplex technique: (a) rollback probability, and (b) MTTF.



(a) (b)

Figure 6. CED with 90% overhead: (a) rollback probability, and (b) MTTF.



(a) (b)

Figure 7. CED with 30% overhead: (a) rollback probability, and (b) MTTF.

From the above analysis results, several observations can be made:

(1) Results are CED dependent: The results vary depending on the CED overhead and the total available FPGA area. Therefore, to choose the most suitable design with given available area, we need to specify the CED techniques and application requirements. For real-time applications, the choice depends on the rollback rate since timing is a critical issue. For non-real-time applications, MTTF is the critical criterion. Table 1 summarizes the choice of design with different available area and CED techniques.

(2) Rollback rate improvement: All the three designs are shown to have lower rollback rate than TMR-Duplex. For most of the cases, hybrid-TMR-Simplex-CED has the lowest rollback rate among all. It is because with the same area, it contains the most TMR part than other designs. Its CED costs lower overhead than the checker in the duplex-with-a-checker design when CED overhead is less than 100%, and the duplex with two CED blocks needs two CED blocks rather than one. However, when the CED overhead is small, the duplex with two CED blocks has the

lowest rollback rate. This is because when the CED overhead is small as in the 16-bit multiplier case, the design can still have non-degraded CED even when the available area is smaller than three times the simplex area. With non-degraded CED, the design has similar fault tolerance capability as a quadraplex. Therefore, it has the lowest rollback rate among all.

(3) MTTF results: Unlike rollback probabilities, the MTTF of the designs do not necessarily decrease as the area decrease. The reason is that although designs with larger area have higher CED coverage and fault tolerance capability, they are also more vulnerable to multiple faults. For some designs, the MTTF curves do not increase or decrease monotonically, and maximum or minimum values of MTTF are observed in the curve. The explanation is that these designs can be seen as a combination of two systems, and the MTTF of each system does not increase at the same rate when its area decreases. As the total area decreases, if the MTTF increment of one system is more the MTTF decrement of the other system, then the overall MTTF increases; and vice versa. In addition, the rate of MTTF increment for each system depends on the system area. Therefore, the overall MTTF curve may increase first and then decrease as the total area decreases, so that a maximum value can be observed in the MTTF curve.

(4) Some designs have better MTTF than TMR-Duplex: TMR-Duplex has longer MTTF than all other designs when the CED overhead is large and the area is close to three times the simplex area. It is because the high CED overhead and limited available area restricts fault tolerance capabilities of our designs, and TMR-Duplex has the smallest design area among all and is less likely to have multiple faults. When the total area is close to twice the simplex area, hybrid-TMR-Simplex-CED has better MTTF than TMR-Duplex for some CED techniques. It is because the hybrid-TMR-Simplex-CED is the series connection of a TMR and a simplex-CED, so that it can recover double faults when one of them occurs in the TMR part and the other occurs in the simplex-CED part. On the contrary, double faults can cause failure of a plain duplex design when they occur in different modules. Here we only discuss double faults because they are the most possible among all multiple faults when error rates are low. When the total available area is smaller, we can see the hybrid-TMR-Simplex-CED design as a 5-block design (3 for TMR, 2 for simplex-CED) where each block is small; and a plain duplex design as a 2-block design where each block is large. Hence, the probability of two-block failure, which will cause system failure, in a 2-large-block design can be larger than that in a 5-small-block design. Therefore, a hybrid-TMR-Simplex-CED design can have better MTTF than TMR-Duplex when the total FPGA area is close to twice the

simplex area. For small CED overhead, the duplex with two CEDs has the longest MTTF among all when the area is close to 3. It is because small CED overhead allows the design be less degraded, and a duplex with two CEDs which have high coverage can detect more multiple faults than other designs.

Table 1. Design selection with different available area and CED techniques.

CED	Applications	Area	
		~3×Simplex area	~2×Simplex area
Duplex	Real-time	Hybrid-TMR-Simplex-CED	Hybrid-TMR-Simplex-CED
	Non-real-time	TMR-Duplex	Hybrid-TMR-Simplex-CED
90%	Real-time	Hybrid-TMR-Simplex-CED	Hybrid-TMR-Simplex-CED
	Non-real-time	TMR-Duplex	TMR-Duplex
30%	Real-time	Duplex w/ Two CEDs	Hybrid-TMR-Simplex-CED
	Non-real-time	Duplex w/ Two CEDs	Hybrid-TMR-Simplex-CED

6. Conclusion

In this paper we presented a new permanent fault repair scheme for 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. Hence, they are more suitable for real-time applications where availability is a critical issue. For CED techniques with large overhead, hybrid-TMR-Simplex-CED is the most suitable choice for real-time applications. For CED techniques with smaller area overhead, hybrid-TMR-Simplex-CED is the most suitable for area close to 3 times simplex area, and duplex with two CEDs is the most suitable for smaller area. For non-real-time applications, the module removal approach may be more suitable than the presented three techniques since it has longer lifetime. The choice of designs depends on the CED overhead. For an actual implementation, information of multiple error detectability of CED and design diversity for module redundancy needs to be obtained to select suitable designs according to the available area.

Acknowledgements

The authors would like to thank Dr. Nirmal Saxena, Dr. Subhasish Mitra, Wei-Je Huang, and Siegrid Munda for their useful feedback and suggestions. 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.

[Dutt 99] Dutt, S., V. Shanmugavel, and S. Trimberger, "Efficient Incremental Rerouting for Fault Reconfiguration in Field Programmable Gate Arrays," *Digest of Technical Papers, IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 173-176, 1999.

[Elder 88] Elder, J. H., Osborn, J., Kolasinski, W. A., and Koga, R., "A Method for Characterizing a Microprocessor's Vulnerability to SEU," *IEEE Trans. on Nuclear Science*, Vol. 35, no. 6, pp. 1678-81, 1988.

[Emmert 98] Emmert, J. M., and D. Bhatia, "Incremental Routing in FPGAs," *Proc. of 11th Annual IEEE Int. ASIC Conf.*, pp. 217-221, 1998.

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

[Huang 01b] Huang, W.-J. and E. J. McCluskey, "A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations," *9th ACM Int. Symp. On Field-Programmable Gate Arrays (FPGA'01)*, pp. 183-192, 2001.

[Lach 99] Lach, J., W. H. Mangione-Smith, and M. Potkonjak, "Algorithms for Efficient Runtime Faulty Recovery on Diverse FPGA Architectures", *DFT'99*, pp. 386-394, 1999.

[Long 90] Long, J., W., W. K. Fuchs, and J. A. Abraham, "A Forward Recovery Using Checkpointing in Parallel Systems," *Proc. of the 1990 Int. Conf. on Parallel Processing*, vol. 1, pp. 272-275, 1990.

[Losq 76] Losq, J. "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," *IEEE Trans. Comput.*, C-25, No. 6, pp. 269-78, 1976.

[Mathur 75] Mathur F. P. and P. DeSousa, "Reliability Modeling and Analysis of General Modular Redundant Systems," *IEEE Trans. Rel.*, R-24, No. 5, pp. 296-9, 1975.

[Mitra 99] Mitra, S., N.R. Saxena, and E.J. McCluskey, "A Design Diversity Metric and Reliability Analysis for Redundant Systems," *Proc. 1999 Int. Test Conf.*, pp. 662-671, 1999.

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

[Ohlsson 98] Ohlsson, M., P. Dyreklev, K. Johansson, and P. Alfke, "Neutron Single Event Upsets in SRAM-Based FPGAs," *IEEE Radiation Effects Data Workshop*, pp. 177-80, 1998.

[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.

[Sparmann 96] Sparmann, U., "On the Effectiveness of Residue Code Checking for Parallel Two's Complement Multipliers," *IEEE Trans. on VLSI Systems*, Vol. 4, No. 2, pp. 227-39, 1996.

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

[Wakerly 74] Wakerly, J. F., *Low-Cost Error Detection Techniques for Small Computers*, Stanford PhD Thesis, 1974.

[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.

Appendix A: Derivation of MTTF

In this section, we derive the equation to calculate MTTF of a system with recovery schemes. The Markov chain in Fig. 4 is re-drawn in Fig. 8 for reader's convenience.

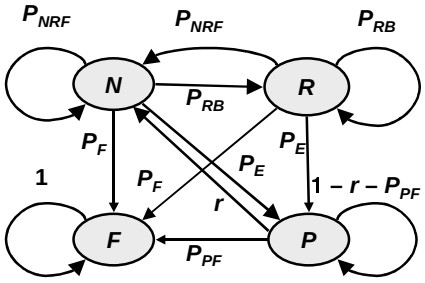


Figure 8. A Markov chain model of an FPGA-based system with recovery mechanism.

We define the probability from State i to State j as $P_{i\to j}$, where $i, j \in \{N, R, P, F\}$. To calculate the evaluation metrics, we define the following parameters:

P_{NRF}: Probability that the system functions correctly during a checkpoint period, or transient errors occur but the system is recovered by roll-forward recovery. When this happens, the system will go to State *N*. Hence, both $P_{N\to N}$ and $P_{R\to N}$ equal to P_{NRF}.

P_{RB}: Probability that transient errors occur but the system is recovered by rollback recovery. The system will go to State *R* for re-computation. Both $P_{N\to R}$ and $P_{R\to R}$ equal to P_{RB}.

P_E: Probability that permanent errors occur and the system raises an indication signal. The system will go to State *E* for repair. Both $P_{N\to E}$ and $P_{R\to E}$ equal to P_E.

P_F: Probability that multiple errors occur so that the system could neither be recovered or raise an indication signal. The system will go to State F in this case. Both $P_{N\to F}$ and $P_{R\to F}$ equal to P_F.

P_{PF}: Probability that the repair process fails. $P_{P\to F}$ equals to P_{PF}.

r: Probability of repair. After repair, the system goes to State N. $P_{P\to N}$ is r, and $P_{P\to P}$ is $1\text{-}r\text{-}P_{PF}$.

Mean time to failure (MTTF): it is the expected time of a system moving from State N to State F. To derive the equation of MTTF, let $D_i[n]$ be the probability that the system is in State i at checkpoint n, where $i \in \{N, R, P, F\}$ and $0 \le n$. Assume that the system is initially fault-free. The system stay at State N at checkpoint 0:

$$D_N[0] = 1,\ D_R[0] = 0,\ D_P[0] = 0, \text{ and } D_F[n] = 0.$$

The transitions can be listed as recursive equations,

$$D_N[n+1] = P_{NRF}\times D_N[n] + P_{NRF}\times D_R[n] + r\times D_P[n],$$

$$D_R[n+1] = P_{RB}\times D_N[n] + P_{RB}\times D_R[n], \text{ and}$$

$$D_P[n+1] = P_E\times D_N[n] + P_E\times D_R[n] + (1\text{-}r\text{-}P_{PF})\times D_P[n].$$

MTTF is obtained by calculating the expected time needed to transit from State N to State F. Solving from recursive equations,

$$\sum_{n=0}^{\infty} D_N[n+1] = \left(\sum_{n=0}^{\infty} D_N[n]\right) - D_N[0] = P_{NRF}\times\sum_{n=0}^{\infty} D_N[n] + P_{NRF}\times\sum_{n=0}^{\infty} D_R[n] + r\times\sum_{n=0}^{\infty} D_P[n]$$

$$\sum_{n=0}^{\infty} D_R[n+1] = \left(\sum_{n=0}^{\infty} D_R[n]\right) - D_R[0] = P_{RB}\times\sum_{n=0}^{\infty} D_N[n] + P_{RB}\times\sum_{n=0}^{\infty} D_R[n]$$

$$\sum_{n=0}^{\infty} D_P[n+1] = \left(\sum_{n=0}^{\infty} D_P[n]\right) - D_P[0] = P_E\times\sum_{n=0}^{\infty} D_N[n] + P_E\times\sum_{n=0}^{\infty} D_R[n] + (1-r-P_{PF})\times\sum_{n=0}^{\infty} D_P[n]$$

$$\Rightarrow \quad \sum_{n=0}^{\infty} D_N[n] = \frac{(1-P_{RB})\times(r+P_{PF})}{r\times P_F + P_{PF}\times(P_F+P_E)}$$

$$\sum_{n=0}^{\infty} D_R[n] = \frac{P_{RB}(r+P_{PF})}{r\times P_F + P_{PF}\times(P_F+P_E)}$$

$$\sum_{n=0}^{\infty} D_P[n] = \frac{P_E}{r\times P_F + P_{PF}\times(P_F+P_E)}$$

Therefore, average number of checkpoints to failure is

$$MTTF = \sum_{n=0}^{\infty} D_N[n] + \sum_{n=0}^{\infty} D_R[n] + \sum_{n=0}^{\infty} D_P[n] = \frac{r+P_E+P_{PF}}{r\times P_F + P_{PF}\times(P_F+P_E)} \text{ checkpoint periods.}$$

For permanent error rate much smaller than repair rate and reliable repair process where $P_E \ll r$ and P_{PF} is very small, MTTF can be approximated as

$$MTTF \cong \frac{1}{P_F} \text{ checkpoint periods.}$$

Appendix B. Derivation of Evaluation Metrics for The Designs

In this section we analyze each of the designs and derive equations for evaluation metrics. Several assumptions are made for the following analysis: (1) Hardware overhead introduced by CED and recovery schemes is proportional to the original logic area. (2) Permanent and transient faults occur independently in each unit area in unit time. (3) Error latency is not considered. (4) State restoration time for recovery is short, so that the effect of state restoration failure is negligible. (5) Common logic such as voters and comparators have much smaller area compared with module area and are protected by duplication or TMR, so that failures caused by errors in common logic are negligible in our calculation.

We use the following notations:

A_T: total used area on an FPGA.

A_M: area of a single module, including the area overhead of recovery mechanism.

ced: a proportional factor of area overhead introduced by CED.

p_t: Probability that no transient error happens in a unit area during a checkpoint period.

p_p: Probability that no permanent fault happens in a unit area during a checkpoint period.

cvg: Detection coverage of CED. All possible single and multiple faults are taken into account.

For a hybrid TMR-Simplex-CED system, a part of the module is protected by TMR and the rest is by Simplex-CED. Assume that the fraction of being protected by TMR is *tmr*, and the fraction of Simplex-CED is 1-*tmr*.

$$A_T = 3 \times A_M \times tmr + A_M \times (1- tmr) \times (1+ ced).$$

From the above equation, the fraction of TMR is calculated as

$$tmr = [A_T/A_M - (1+ced)]/(2-ced), \qquad 0 \le tmr \le 1.$$

In this design, when an error in Simplex-CED part is detected, rollback will be evoked regardless if the TMR part is faulty or not. Therefore,

P_{RB} = Prob (no permanent faults happen; transient faults happen in the simplex block and are detected)

$$= p_p^{3\times A_M\times tmr}\times(1-p_t^{A_M(1-tmr)(1+ced)})\times p_p^{A_M(1-tmr)(1+ced)}\times cvg\,.$$

P_F = Prob (faults happen in the simplex block and are not detected,

or the simplex block is fault-free but two or three blocks of TMR are faulty)

$$= (1-(p_t p_p)^{A_M(1-tmr)(1+ced)})\times(1-cvg)$$

$$+(p_t p_p)^{A_M(1-tmr)(1+ced)}\times(3\times(1-(p_t p_p)^{A_M\times tmr})^2\times(p_t p_p)^{A_M\times tmr}+(1-(p_t p_p)^{A_M\times tmr})^3)\,.$$

In a duplex system with a checker, let the checker area be denoted as A_{CK}.

$$A_{CK} = A_T - 2\times A_M, \qquad 0 \le A_{CK} \le A_M.$$

We call the part of the module that produces the same partial result as the checker *the checked block*, and the rest of the module *the unchecked block*. Then the area of the checked block is the same as the checker area, A_{CK}. The area of the unchecked block is $A_M - A_{CK}$.

In the case when one of the modules has transient faults in both its unchecked block and checked block, and the rest is fault free, we assume rollback happens instead of roll-forward. The reason is that although the checked block of the faulty module has transient faults, the faults do not necessarily produce incorrect partial outputs. Hence, the faults might not be caught by the checker, and in this case rollback will happen instead of roll-forward. We consider the worst case to compute a lower bound of the evaluation metrics. Therefore,

P_{RB} = Prob (One of the modules has transient faults in the unchecked block. Its checked block may or may not have transient faults. The rest are fault-free.) + Prob (Both the checked blocks are fault-free. The checker and one of the unchecked blocks have transient faults)

$$= 2\times((1-p_t^{A_M-A_{CK}})p_p^{A_M-A_{CK}})\times p_p^{A_{CK}}\times(p_t p_p)^{A_M}\times(p_t p_p)^{A_{CK}}$$

$$+2\times(p_t p_p)^{2\times A_{CK}}\times((1-p_t^{A_{CK}})p_p^{A_{CK}})\times((1-p_t^{(A_M-A_{CK})})p_p^{(A_M-A_{CK})})\times(p_t p_p)^{A_M-A_{CK}}\,.$$

P_F = Prob (Both the checked blocks fail, or only one of them fails and the checker fails) + Prob (None of the case above, but both of the unchecked blocks fail) + Prob (The checker is fault-free. One of the duplex fails in the checked block and the other fails in the unchecked block.)

$$= (1-(p_t p_p)^{A_{CK}})^2+2\times(p_t p_p)^{A_{CK}}\times(1-(p_t p_p)^{A_{CK}})^2$$

$$+(1-(1-(p_t p_p)^{A_{CK}})^2-2\times(p_t p_p)^{A_{CK}}\times(1-(p_t p_p)^{A_{CK}})^2)(1-(p_t p_p)^{A_M-A_{CK}})^2$$

$$+2\times(p_t p_p)^{A_{CK}}(p_t p_p)^{A_{CK}}(p_t p_p)^{A_M-A_{CK}}(1-(p_t p_p)^{A_{CK}})(1-(p_t p_p)^{A_M-A_{CK}})\,.$$

In a duplex system with CED in both modules, assume that the fraction of modules being protected by CED is *f*. Then the total area needed is

$$A_T = 2 \times A_M \times (1+f \times ced).$$

Hence, $f = (A_T /(2 \times A_M)\text{-}1) / ced, \qquad 0 \le f \le 1.$

Since only *f* of each module has CED, the area that is covered is $A_M \times f \times (1+ced)$ and the overall module area including CED is $A_M \times (1+ f \times ced)$; overall coverage for each module, *cvg_all*, is

$$cvg\_all = (cvg \times \text{area covered by CED})/(\text{total area}) = (cvg \times f\,(1+ced)) / (1+f \times ced).$$

P_{RB} = Prob (one of the modules has transient faults and is not detected and the others are fault-free; or both modules have transient faults and both are detected)

$$= ((1- p_t{}^{A_M(1+f\times ced)}) p_p{}^{A_M(1+f\times ced)})(1-cvg\_all)(p_t p_p)^{A_M(1+f\times ced)} + ((1- p_t{}^{A_M(1+f\times ced)}) p_p{}^{A_M(1+f\times ced)})^2 cvg\_all^2.$$

P_F = Prob (both modules are faulty but not both of them are detected)

$$= (1-(p_t p_p)^{A_M(1+f\times ced)})^2 \times (1-cvg\_all^2).$$