2014 IEEE 28th International Parallel & Distributed Processing Symposium Workshops

FPGA Redundancy Configurations: An Automated Design Space Exploration

Jahanzeb Anwer, Marco Platzner and Sebastian Meisner
University of Paderborn, Germany
{jahanzeb.anwer, platzner, sebastian.meisner}@upb.de

***Abstract*—With ever-decreasing CMOS transistor sizes, integrated circuits are becoming more and more susceptible to errors. A commonly used approach to improve the reliability of digital circuits is triple modular redundancy (TMR). TMR instantiates three copies of a circuit plus additional voter circuits to take majority decisions on the output values. Prior research has studied variations in TMR voting structures that bring about improvements in performance factors such as area utilization, power consumption and latency at the price of slight degradation in output reliability. In this paper, we extend previous studies by utilizing different redundancy configurations and voter-insertion algorithms to observe variation in these performance factors for FPGA designs. To maintain an automated tool flow for redundancy insertion into a digital design, we enhance the functionality of the previous BYU-LANL TMR tool to include alternative TMR and cascaded TMR redundancy configurations. Design space exploration experiments with different ISCAS circuit benchmarks show that the choice of an appropriate redundancy configuration and voter-insertion algorithm has a strong impact on optimizing performance factors. To support a designer with selecting a redundant implementation, we present a design space exploration tool flow that takes a circuit as input and identifies Pareto-optimal implementations with respect to the four objectives reliability level, utilized FPGA area, latency and dynamic power consumption.**

***Keywords*—Redundancy configurations, Voter-insertion algorithms, Reliability, BYU-LANL TMR tool**

I. Introduction

The scaling of CMOS transistor technology to nanometer dimensions is increasing the unreliability of electronic devices. Nano-devices are unreliable as they are prone to radiation effects like radioactive decay and cosmic rays [1], [2], [3]. In FPGAs, the radiation effects are manifested in the form of soft errors. These soft errors could be single-event upsets (SEUs) causing a permanent change of logic level in the storage element, e.g. configuration memory or they could be single-event transients (SETs) which are temporary logic level glitches [4]. In either case, the error occurred at any random node could be serious enough to cause a system failure or moderate enough to be absorbed within the circuit architecture.

The soft errors could appear in different resources of an FPGA, e.g. combinational and sequential elements, device configuration elements or in clock and global routing. Among these resources, the errors in configuration memory are particularly disastrous as the whole functionality of the FPGA is represented by a bitstream which is stored in configuration memory. Scrubbing is a popular technique that refreshes the data of the configuration memory at regular intervals to avoid accumulation of soft errors [5]. In addition, the resources of the FPGA including configuration memory can be made redundant so that the error occurred at any node could be absorbed within the circuit architecture in contrast to refreshing memory that adds high latency to the circuit design [6], [7].

The FPGAs produced by the leading manufacturers such as Xilinx indeed suffer from radiation-induced errors. The Rosetta experiment [8] and device reliability reports published by Xilinx [9] prove that FPGAs are prone to radiation effects particularly at higher altitudes. Therefore, FPGA manufacturers are introducing error mitigation techniques along with their products to stay below the acceptable failure-in time (FIT) rates. The simplest solution is to add redundancy in the RTL design which is quite tedious to be done by hand especially when the design is huge and multi-module. However, automated tools for redundancy-insertion are available, e.g. Xilinx TMR tool [10], Precision Hi-Rel software [11] and BYU-LANL TMR tool [12], though only the BYU tool is available as open-source. The basic idea behind these softwares is to add triple modular redundancy in the design by converting the synthesis-generated EDIF to a replicated EDIF netlist for further processing by the mapping stage [13].

In this paper, we present a design space exploration tool flow for creating circuit implementations with variations in the reliability level and the three performance factors area consumption, latency and dynamic power consumption. Our claim is that since different redundancy techniques pose different trade-offs of reliability against performance factors, such a design space exploration should be performed for each individual HDL circuit. To create the design space exploration tool flow, we have extended the functionality of the BYU-LANL TMR tool to support different redundancy configurations.

The rest of the paper is organized as follows. Section II elaborates how triple modular redundancy is practiced on FPGAs, in particular with the BYU-LANL tool assisted redundancy insertion. Afterwards, we give examples of redundancy configurations in addition to conventional triple modular redundancy. Section III presents the stages of our tool flow. In Section IV, we provide experiments on benchmarks to show how the performance factors vary with the reliability configuration type. Finally, Section V concludes the paper.

II. Related Work

The following two subsections briefly review major research work related to implementing TMR in FPGAs followed by the variation in voting structures of TMR and its cascaded versions.

978-1-4799-4116-2/14 $31.00 © 2014 IEEE
DOI 10.1109/IPDPSW.2014.37

A. Triple Modular Redundancy in FPGAs

The concept of triple modular redundancy (TMR) is straight-forward as it triplicates a logic design and takes the primary output of the design from a voter placed at the outputs [14]. The function of the voter is to sample three logic outputs and forward the majority result. The limitation of this architecture is the single point of failure, i.e., an error occurring in the voter renders the TMR technique useless. To avoid the single point of failure, we can create a more reliable architecture involving triplicated voters in addition to triplicated logic modules. This architecture removes the single point of failure and runs the three branches in parallel unless they are to be interfaced with the outside world of the FPGA where they can be converged using reducing voters or could be interfaced in the form of three outputs.

The TMR technique can be used in an FPGA by simply triplicating the inputs, outputs and logic modules, inserting buffers and placing the triplicated voter as shown in Figure 1. There are some practical considerations due to which this straight-forward implementation is not suitable. Firstly, TMR is able to counter one error among the three redundant branches, and a larger length of each branch increases its probability of being erroneous more than once. To deal with this fact, there is a need to break the logic of the branch at regular intervals and place the triplicated voters as shown in Figure 2. Thus, an error occurring in one partition of the logic will not be forwarded to the next partition due to the error-mitigation effect of the triplicated voter. In addition, there are certain locations on an FPGA called illegal-cut locations which should not be triplicated due to the FPGA architecture, e.g., dedicated route connections in a slice [13]. Moreover, voters should not be placed on high-speed carry chains in order to not deteriorate the timing performance of the design. Most importantly, voters should always be added in the feedback paths to avoid data corruption at the outputs of sequential elements being forwarded into the feedback paths [13], [14]. These voters are commonly denoted as synchronization voters.

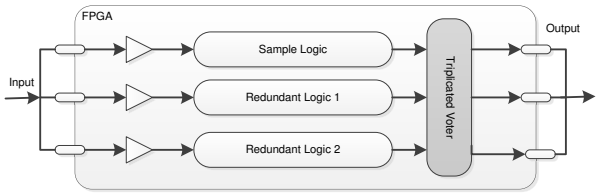

Fig. 1: TMR implementation in FPGA

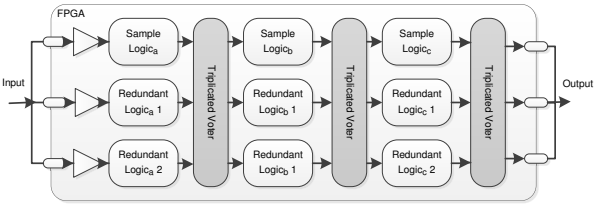

Fig. 2: TMR implementation with logic partition

The process of automatic TMR insertion into a circuit design can be done, for example, via the BYU-LANL tool. We are using the BYU-LANL tool for our analysis because it is the only open-source tool and, thus, modifiable in contrast to other commercial TMR tools. The BYU-LANL tool is able to triplicate the design, insert voters and use built-in algorithms to take care of the constraints explained above. It is up to the discretion of the circuit designer to request the desired *redundancy configuration*, e.g., TMR with only single voters or with a mixture of single and triplicated voters. Moreover, there is a choice of eight algorithms that decide the placement of voters in the triplicated design. These algorithms are termed as *voter-insertion* algorithms. Depending on the type of algorithm, the sets of nets are determined where the voters should be inserted, e.g., using feedback edge set (FES) algorithms or decomposition of strongly connected components (SCCs) in the circuit graph [13]. The algorithms used in the BYU-LANL tool are abbreviated as follows:

1) CC: Connectivity cutset
2) AFC: After flipflop cutset
3) BFC: Before flipflop cutset
4) BD: Basic decomposition
5) HFC: Highest fanout cutset
6) HFFC: Highest flipflop fanout cutset
7) HFFIC: Highest flipflop fanin input cutset
8) HFFOC: Highest flipflop fanin output cutset

B. Variation in Voting Structures of TMR and Cascaded TMR

Besides the conventional single and triplicated voter configurations, there are configurations proposed in Lee et al. [15] with single/double voters in the alternate stages. Using Monte Carlo simulations, the authors proved that these alternate configurations are slightly less reliable than triplicated voter configuration though they save the overhead of increased number of voters. Hence, in situations where the radiation environment is not very strong and area consumption is an important issue, the alternate configurations are highly useful. In addition there is a concept of Cascaded TMR which results in more reliable configurations than TMR, though consuming more area [16]. For our analysis, we use the level-1 CTMR. It can be noted that the CTMR configuration has a single point of failure as it is the extension of TMR with a single voter. While CTMR can also be improved by using the triplicated voter strategy, we assume that CTMR is always superior to TMR for the time being. Overall and based on the results presented in [14], [15], [16], we rate the reliability of the discussed configurations in the following ascending order:

1) SV: Single Voter ([14])
2) OAV: One alternate voter ([15])
3) TAV: Two alternate voters ([15])
4) TV: Triplicated voter ([14])
5) CTMR: Cascaded TMR- Level 1 ([16])

It has to be noted that all configurations have to resort to single voters for illegal-cut locations. In particular, also the configuration TV which is the default configuration of the BYU-LANL tool combines triplicated and single voter structures.

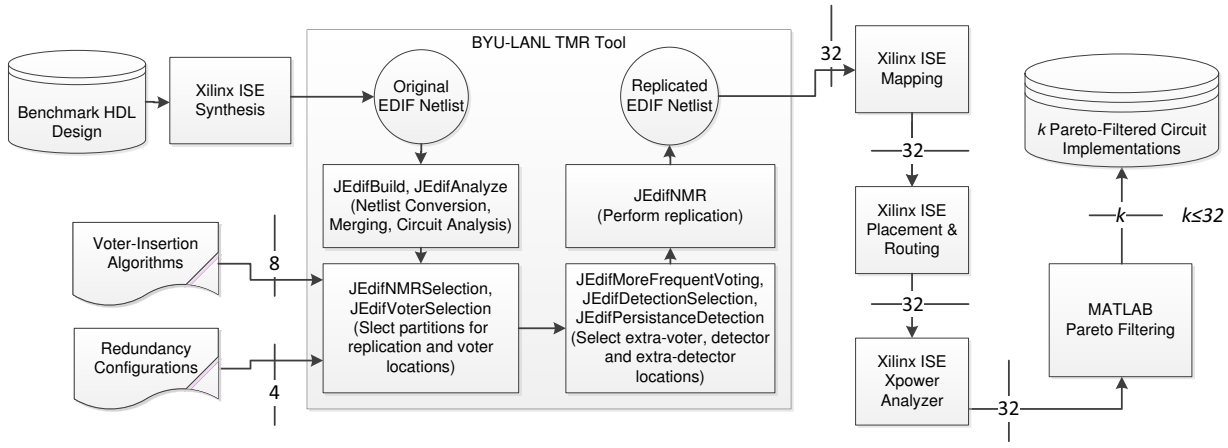

Fig. 3: Extended BYU-LANL TMR tool

III. Design Space Exploration Tool Flow

In this section, we list the stages of our design space exploration tool flow. The tool flow is illustrated in Figure 3 and converts a benchmark HDL design into a set of 4-dimensional Pareto-filtered implementations rated by the reliability level, area consumption, latency and dynamic power consumption.

A. Xilinx ISE Synthesis

In the first stage, we synthesize the benchmark HDL design with Xilinx ISE and output an EDIF file to be processed by the BYU-LANL tool.

B. Replication via the BYU-LANL Tool

1) Original BYU-LANL Tool Flow: The BYU-LANL tool performs logic replication in four major stages:

- The first stage comprising *JEdifBuild* and *JEdifAnalyze* performs the technical steps of design flattening, circuit and IOB analysis, etc., and saves the information in intermediate files to be used by the following tools.
- The second stage of *JEdifNMRSelection* and *JEdifVoterSelection* determines the type of configuration and replication to be used, replicates the instances, determines the voter-insertion locations by the specific algorithm used and makes the necessary wire connections.
- The third stage can be run if desired for adding more voters and error-detectors. For our research, we are excluding this stage as an optional one.
- The final stage does the actual replication by reading the intermediate file formats written by the previous tools and by generating the replicated netlist.

2) Extension of BYU Tool: We have made two extensions to the original BYU-LANL tool in order to support the additional redundancy configurations as explained in Section II. First, we have integrated the redundancy configurations number 2, 3 and 5 discussed in section II(B). The tool now supports overall four redundancy configurations: OAV, TAV, TV, and CTMR. The sole configuration with single voters, SV, is not available with the default setting and is thus currently not supported. The BYU-LANL also supports duplication with correction which is out of scope of this paper. Second, we have changed the command-line interface of the *JEdifNMRSelection* tool from replication type to configuration type because there is more than one configuration that uses triplication. As a result, now we can decide on a configuration instead of a replication type. It should be mentioned that our extended tool uses the mixed configuration of single voter, i.e. SV with the rest of the configurations due to constraints governing the replication of resources on FPGA as explained in Section II. We have not altered the decisions taken by the BYU-LANL tool in selecting single or triplicated voters. Therefore, our new configurations can be used with all the eight voter-insertion algorithms.

3) Performing Replication: After running the first stage of the tool, we need to choose one of the four redundancy configurations via the *JEdifNMRSelection* tool and one of the eight voter-insertion algorithms via the *JEdifVoterSelection* tool. The final stage of *JEdifNMR* reads the intermediate file formats of previously run tools and writes the final replicated EDIF netlist. For the time being, we assume that for a single configuration type, the circuit implementations generated by different voter-insertion algorithms have the same reliability level. Overall, we run the extended BYU-LANL TMR tool for 32 possible combinations of redundancy configuration and voter-insertion algorithm, which sums up the design implementation set.

C. Xilinx ISE Mapping, Placement/Routing and Power Analysis

All of the 32 generated implementations of the design are passed through Xilinx ISE mapping, placement/routing and XPower analyzer tools to obtain slice utilization (for area consumption), pad-pad delay/clock frequency (for latency) and dynamic power consumption, respectively. We resort to the dynamic power consumption since the static power almost remains the same throughout the analysis of a single benchmark.

D. Pareto Filtering of Implementation Points via MATLAB

Using the Pareto Front function of MATLAB [17], we reduce the set of implementation points to non-dominated ones to make the selection of trade-off points easier. For our analysis, we have rated the reliability of the four redundancy configurations as shown in Section II-B. This ordering of the reliability levels is sufficient for our Pareto analysis to identify non-dominated design points and, furthermore, an exact mathematical analysis of the configuration reliabilities is rather involved and out of scope of this paper.

IV. Experiments and Results

In order to analyze the variation in performance factors, i.e., area consumption, latency and dynamic power consumption with respect to changes in the redundancy configuration and voter-insertion algorithm we report on and analyze six benchmark HDL designs from three classes of benchmarks with different circuit architectures [18]: c17 and c3540 from ISCAS'85, s713 and s838 from ISCAS'89, and b17 and b22-1 from ISCAS'99 benchmark suites. We have experimented with overall four redundancy configurations including OAV (one alternate voter), TAV (two alternate voters), TV (triplicated voters), and CTMR (cascaded TMR level 1) from Subsection II-B as well as NR, a non-redundant design for comparison. The device used is a Virtex 5 FPGA, XCVTX150T, with package FF1156. We have set the optimization method for design mapping to *balanced* to ensure the best combination of area and speed efficiency. For latency comparison, we use the maximum pad-pad delay and maximum clock frequency for combinational and sequential circuits, respectively. To determine dynamic power consumption we have applied random testbench signals to each benchmark, though these random inputs do not necessarily account for the maximum possible power consumption.

A. c17 and c3540 (ISCAS'85)

These benchmarks designs are purely combinational circuits and, by default, the BYU-LANL TMR tool inserts only single voters for combinational circuits. Without sequential elements the configurations OAV, TAV and TV always produce the same results since no triplicated or alternate-triplicated voters are used. Also, the choice of voter-insertion algorithm does not matter because voter insertion also varies only with sequential elements. Table I presents the results for the two ISCAS'85 benchmark designs. The reliability configurations OAV, TAV and TV have identical performance factors. The variations in latency, i.e., maximum pad-to-pad delay, is less pronounced for c17 than for c3540, which is a much larger design. For c17, there is even a slight decrease in latency with increasing reliability configurations. This decrease in latency is due to automatic placement and routing which sometimes can find more optimization potential in larger designs. The difference in dynamic power consumption is also less pronounced for the small benchmark. Note that the Pareto filtering is not required for these benchmarks as Table I already lists the minimum set of non-dominated points.

B. s713 and s838 (ISCAS'89)

This series of HDL benchmark designs consists of combinational as well as sequential elements due to which the voter placement algorithms result in variation of the performance factors. Table II lists the results for the two benchmarks and highlights the non-dominated, i.e., Pareto-optimal, implementations of the HDL design. Those implementation points having similar parameter values are highlighted only once. The benefit of Pareto-filtering is obvious as the 32-point set is reduced to 10 and 6 points for s713 and s838, respectively. It can be observed from the table that the span of parameters for OAV, TAV and TV is not high since they only differ in number of voters as compared to NR and CTMR which vary in number of modules as well. Moreover, we can observe that different voter-insertion algorithms can greatly vary the trade-off points. While one would assume that the area utilization always increases in ascending order from NR to CTMR, the experiments prove this assumption wrong. As can be observed for HFC algorithm and the s713 benchmark, the configuration TV consumes less slices than the configuration TAV, albeit the number of voters for TV is higher than for TAV. The explanation for such anomalies lies again in the automatic mapping, placement and routing tools. Sometimes, resources such as flip-flops remain unused in slices to balance the timing constraints and, more generally, the optimization possibilities vary from one design to another. Furthermore, the choice of voter-insertion algorithm within each configuration has a high impact on the area consumption, e.g., for the s838 benchmark the slice utilization varies from 113 to 124 (9.7% variation) for OAV configurations in contrast to 133 to 276 slices (107.5% variation) for CTMR configurations. Similarly, due to different optimization possibilities the maximum clock frequency also does not always decrease with configurations using higher degrees of replication. For example, the maximum clock frequency increased for the s713 benchmark design and the CC algorithm when going from TAV to TV. However, switching from the non-redundant to redundant configurations drastically impacts the maximum clock frequency. For example, when going from NR to CTMR we observe a 35% decrease. The maximum clock frequency also varies considerably with variation of voter-insertion algorithm, e.g., 29% variation for s713 and the TAV configuration. The dynamic power consumption varies minimally for these benchmarks, with a maximum variation of 9.8% observed for s838 and the CTMR configuration.

TABLE I: Design space exploration results for the benchmarks c17 and c3540 from the ISCAS'85 benchmark suite.

		# Slices	Max Pad-Pad Delay (ns)	Dynamic PD (W)
c17	NR	2	5.461	0.165
	OAV TAV TV	3	5.337	0.176
	CTMR	6	5.171	0.176
c3540	NR	90	15.137	2.499
	OAV TAV TV	253	17.283	3.115
	CTMR	447	22.135	3.415

TABLE II: Design space exploration results for the benchmarks s713 and s838 from the ISCAS'89 benchmark suite.

			CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
s713	NR	# Slices	42							
		Max Freq (MHz)	336							
		Dynamic PD (W)	1.476							
	OAV	# Slices	93	78	86	93	81	85	85	85
		Max Freq (MHz)	233	262	249	233	263	273	255	273
		Dynamic PD (W)	1.997	1.988	1.986	1.997	1.993	1.993	1.988	1.993
	TAV	# Slices	101	97	97	101	98	106	93	106
		Max Freq (MHz)	205	265	218	205	220	231	246	231
		Dynamic PD (W)	1.993	1.993	2.006	1.993	1.990	1.999	1.986	1.999
	TV	# Slices	102	117	94	102	86	117	92	117
		Max Freq (MHz)	227	207	222	227	240	207	233	207
		Dynamic PD (W)	2.002	1.996	2.001	2.002	1.988	1.996	1.998	1.996
	CTMR	# Slices	211	214	223	211	173	214	223	214
		Max Freq (MHz)	219	232	228	219	219	232	228	232
		Dynamic PD (W)	2.083	2.073	2.079	2.083	2.050	2.073	2.079	2.073
s838	NR	# Slices	33							
		Max Freq (MHz)	260							
		Dynamic PD (W)	0.287							
	OAV	# Slices	113	117	124	113	117	117	124	117
		Max Freq (MHz)	222	205	228	214	205	205	228	205
		Dynamic PD (W)	0.317	0.305	0.316	0.317	0.305	0.305	0.316	0.305
	TAV	# Slices	147	131	133	147	131	131	133	131
		Max Freq (MHz)	217	212	205	217	212	212	205	212
		Dynamic PD (W)	0.317	0.320	0.322	0.317	0.320	0.320	0.322	0.320
	TV	# Slices	166	173	140	166	173	173	140	173
		Max Freq (MHz)	203	227	208	203	227	227	208	227
		Dynamic PD (W)	0.328	0.327	0.325	0.328	0.327	0.327	0.325	0.327
	CTMR	# Slices	276	133	212	276	133	133	212	133
		Max Freq (MHz)	203	219	203	203	219	219	203	219
		Dynamic PD (W)	0.336	0.306	0.321	0.336	0.306	0.306	0.321	0.306

C. b17 and b22-1 (ISCAS'99)

This series of HDL benchmarks contain complex state machine designs with immense feedback loops. Table III shows a high variation in all the performance factors compared to the last two benchmark categories. The maximum slice utilization increase from OAV to TV is noted as 25.6% and 31% for b17 and b22-1 respectively. Looking at the importance of the voter-insertion algorithm for this benchmark category, we observe a variation of the maximum clock frequency from 54 MHz to 99 MHz (83%) for b22-1's and the TAV configuration. The dynamic power consumption can vary from 0.823 W to 1.424 W (73%), as can be seen for b22-1's and the TV configuration. An anomaly is the decrease in dynamic power consumption of the b22-1 benchmark when going from OAV to TV, though it is expected to increase on the basis of increased area consumption. Hence, if power is a strict consideration in a design and area can be compromised to some extent, such a trade-off could be appealing. Pareto-filtering reduces the number of reasonable designs from 32 to 17 and 13 for b17 and b22-1, respectively, which covers nearly all the available voter-insertion algorithms in one or more redundancy configuration.

V. Conclusion and Further Work

This research highlights the need for design space exploration for selecting redundancy configurations having different reliability levels. Each redundancy configuration demands a trade-off for performance factors such as area consumption, latency and dynamic power consumption of the design. Computing the design space for different redundancy configurations and different voter-insertion algorithms results in implementations having different performance factors. In this paper, we have described an extension to the BYU-LANL open source TMR tool that allows us to generate a larger set of implementations at more redundancy configurations. We have outlined our design space exploration tool flow combining Xilinx ISE tools with the extended BYU-LANL tool and MATLAB for Pareto filtering. The result is a design tool flow that can be used to select a suitable implementation depending on constraints in area consumption, latency, power consumption and the required reliability level.

Experiments conducted on various benchmarks supported our case of performing a design space exploration for each individual HDL circuit. Our experiments have shown that the magnitudes of the performance factors do not linearly increase with the reliability level. Instead, the final performance factors strongly depend on the FPGA mapping, placement and routing tools. A slight increase in area consumption might even lead to dramatic decrease in power consumption and/or latency. Similarly, an increase in the reliability level could even improve the performance factors. Moreover, our experimental observation shows that the trend in variation of performance factors strongly depends on the circuit benchmark suite. Therefore, it is highly beneficial to generate all the circuit implementations with respect to reliability levels and voter-insertion algorithms and filter the non-dominated ones.

TABLE III: Design space exploration results for the benchmarks b17 and b22-1 from the ISCAS'99 benchmark suite.

			CC	AFC	BFC	BD	HFC	HFFC	HFFIC	HFFOC
b17	NR	# Slices	2266							
		Max Freq (MHz)	151							
		Dynamic PD (W)	0.486							
	OAV	# Slices	7414	6316	6427	7313	6163	6125	6516	5993
		Max Freq (MHz)	78	112	112	84	92	107	112	112
		Dynamic PD (W)	2.437	1.800	1.698	2.415	2.044	1.957	1.988	1.818
	TAV	# Slices	8191	6722	7220	8158	6687	6403	6982	6251
		Max Freq (MHz)	74	115	112	77	90	108	77	103
		Dynamic PD (W)	2.944	2.093	1.973	3.066	2.350	2.187	1.842	2.079
	TV	# Slices	9318	6835	7526	9403	6832	6758	7401	6797
		Max Freq (MHz)	72	111	108	75	93	112	101	100
		Dynamic PD (W)	3.318	2.302	1.889	3.389	2.431	2.327	1.897	2.278
	CTMR	# Slices	14098	11614	13502	14941	11991	11597	12806	11734
		Max Freq (MHz)	66	71	74	70	76	85	64	72
		Dynamic PD (W)	4.667	3.762	3.360	4.712	4.035	3.792	3.423	3.752
b22-1	NR	# Slices	1240							
		Max Freq (MHz)	116							
		Dynamic PD (W)	0.939							
	OAV	# Slices	4851	3627	4173	4913	4165	4155	4063	3755
		Max Freq (MHz)	60	97	81	68	81	99	80	95
		Dynamic PD (W)	2.314	1.677	2.031	1.806	1.878	2.270	2.067	1.886
	TAV	# Slices	4877	4281	3757	5190	4086	3764	3946	4300
		Max Freq (MHz)	99	99	84	54	67	86	83	84
		Dynamic PD (W)	1.260	1.268	1.251	0.990	1.442	1.439	1.241	1.388
	TV	# Slices	6354	4044	4462	5678	4821	4357	4521	4084
		Max Freq (MHz)	80	86	77	55	73	90	83	90
		Dynamic PD (W)	1.018	0.823	0.836	0.959	1.424	1.283	0.829	1.246
	CTMR	# Slices	12158	9923	10410	11780	9302	9923	10410	8780
		Max Freq (MHz)	42	46	48	42	44	46	48	46
		Dynamic PD (W)	3.693	4.139	4.547	3.944	3.590	4.139	4.547	3.719

Future work will include the extension of the BYU-LANL tool to support the latest Xilinx FPGAs series such as Virtex-6 and 7 models and to look into techniques to quantitatively characterize the reliability of different redundancy configurations. Furthermore, we intend to look into advanced applications of our tool flow. Currently, we give a designer the freedom to choose the circuit implementation conforming to design constraints or reliability levels. In principle, we could store the dominated circuit implementations and create a system that can decide at runtime which implementation to configure and use.

REFERENCES

[1] C. Huang, *Robust Computing with Nanoscale Devices: Progresses and Challenges*. Springer, 2010.

[2] M. Stanisavljević, A. Schmid, and Y. Leblebici, *Reliability of Nanoscale Circuits and Systems: Methodologies and Circuit Architectures*. Springer, 2010.

[3] M. Tehranipoor, *Emerging Nanotechnologies: Test, Defect-Tolerance and Reliability*, ser. Frontiers in Electronic Testing. Springer, 2008, vol. 37.

[4] R. Do, "Automated Triple Modular Redundancy," Webseminar, Mentor Graphics, 2011. [Online]. Available: http://www.mentor.com/products/fpga/multimedia/automated-triple-modular-redundancy-how-and-when-to-use-it

[5] J. A. Maestro and P. Reviriego, "Selection of the Optimal Memory Configuration in a System Affected by Soft Errors," *IEEE Transactions on Device and Materials Reliability*, vol. 9, no. 3, pp. 403–411, 2009.

[6] V. Srinivasan, J. W. Farquharson, S. Member, W. H. Robinson, B. L. Bhuva, and S. Member, "Evaluation of Error Detection Strategies for an FPGA-Based Self-Checking Arithmetic and Logic Unit," in *MAPLD International Conference*, 2005.

[7] K. Iniewski, *Radiation Effects in Semiconductors (Devices, Circuits, and Systems)*. CRC Press, 2010.

[8] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The Rosetta Experiment: Atmospheric Soft Error Rate Testing in Differing Technology FPGAs," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 317–328, 2005.

[9] Xilinx Device Reliability Report, UG116, Ver. 9.1, 2012.

[10] "TMR Tool." [Online]. Available: http://www.xilinx.com/ise/optional_prod/tmrtoolhtm

[11] "Precision Hi-Rel Synthesis Software." [Online]. Available: http://www.mentor.com/ products/fpga/synthesis

[12] "BYU EDIF Tools Homepage." [Online]. Available: http://reliability.ee.byu.edu/edif

[13] J. M. Johnson and M. J. Wirthlin, "Voter Insertion Algorithms for FPGA Designs Using Triple Modular Redundancy," in *Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2010, pp. 249–258.

[14] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs," Xilinx Technical Report, XAPP197 (v1.0.1), 2006.

[15] S. Lee, J. Jung, and I. Lee, "Voting Structures for Cascaded Triple Modular Redundant Modules," *IEICE Electronics Express*, vol. 4, no. 21, pp. 657–664, 2007.

[16] D. Bhaduri and S. K. Shukla, "NANOPRISM: A Tool for Evaluating Granularity vs. Reliability Trade-offs in Nano Architectures," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, 2004, pp. 109–112.

[17] Y. Caol, "Pareto Front." [Online]. Available: http://www.mathworks.de/matlabcentral/fileexchange/17251-pareto-front

[18] "Benchmarks homepage." [Online]. Available: http://www.pld.ttu.ee/maksim/ benchmarks