Using Statistical Models with Duplication and Compare for Reduced Cost FPGA Reliability

Jon-Paul Anderson, Brent Nelson, Mike Wirthlin
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
Brigham Young University
Provo, UT 84602
jonpaul, brent_nelson, wirthlin@byu.edu

Abstract—Although highly reliable for fault mitigation, triple modular redundancy (TMR) in FPGAs comes with the price of increasing the circuit area (3-5x), decreasing the circuit clock rate (20+%), and increasing circuit power (3-5x). Techniques may exist that trade off some of the reliability of TMR for reduced costs in terms of area, timing, and power. This paper proposes one such technique which uses duplicate with compare (DWC) with the addition of a smart detector to predict which of the duplicated circuits is in error to choose the fault free circuit as output. The smart detector proposed in this paper is a simple statistical model with low-resource costs. The model and testing methodology employed is discussed as well as results from fault injection testing, which indicate that the proposed statistical smart detector exhibits 87% to 93% prediction accuracy.[1 2 3]

Table of Contents

1. Introduction

TMR is the technique most often used for redundancy in FPGAs [1] [2] [3]. However, it comes with a significant cost in terms of area (3-5x), timing (20+%), and power (3-5x) [4]. This paper will describe ongoing work designed to answer the question: *What reliability can be obtained using a statistical smart model predictor combined with duplication with compare (DWC), and what is the resulting hardware savings compared to TMR?* A typical scenario where a reduced cost approach may be warranted might be running a science experiment on a space-based FPGA platform where the platform had insufficient FPGA resources for a full TMR implementation. Our hypothesis is that statistical models combined with DWC may provide sufficient reliability for such an application at an attractive cost and power point.

Conventional DWC [5] consists of two copies of the circuit of interest (call the outputs of these two circuit copies A and B). One of either A or B is used as the main circuit output, but an additional single bit, *Not_Equal*, is also produced to indicate on which cycles A is not equal to B, thus signaling to the user that the output data may be in error. DWC does not mask errors, but merely indicates that an error has occurred. Smart detection, the subject of this work, consists of adding a smart model which monitors the A and B data streams and, when they differ, chooses which value it believes is most likely correct and gates it to the circuit's output. Candidates for this smart model include statistical techniques, neural networks, hidden Markov models, or application-specific smart models based on quantities such as SNR, frequency drift characteristics, etc. This paper focuses on the first of these techniques developed and tested: a histogram-based statistical method.

2. Statistical Smart Detector

An example of a smart detector system is shown in Figure 1. The smart detector monitors the outputs from the duplicated circuit and switches the multiplexor according to its determination of which circuit is fault free. One specific instance of the smart detector is a histogram-based model.

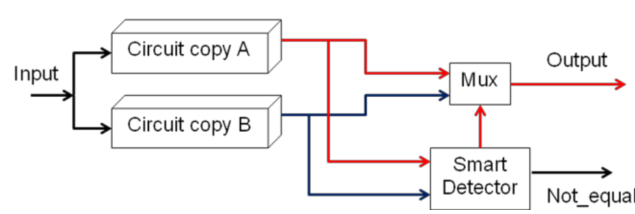


Figure 1- DWC with Smart Detector

The histogram-based model relies on a two-step process. First, representative data is run through a fault free copy of the design and a histogram of output values is constructed for a preselected number of bins. Then, during actual

[1] 978-1-4244-3888-4/10/$25.00 ©2010 IEEE
[2] IEEEAC paper#1504, Version 4, Updated 2010:1:04
[3] This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 0801876.

operation, the histogram is consulted to determine the most likely of the two values output from circuits A and B, where the likelihood is determined by which output maps to the highest value histogram bin. This technique relies on the output data having a non-uniform distribution which can be represented by a histogram. In particular, the data which was used in this study is the output of a QPSK modulator. In this case, the bits being transmitted were random and therefore have no such interesting distribution. However, as will be shown later in this paper, the QPSK modulation process imposes a non-uniform distribution on the data samples.

There are three possible outcomes when consulting the histogram for detection: Correct detection, incorrect detection, and an ambiguous outcome. A simple example will illustrate these three possibilities.

A Simple Example

For this simple example the circuit to be protected is a voltmeter. The voltmeter is observed in fault free operation and the outputs and their counts are gathered into the histogram shown in Table 1. As can be seen, 3V is the most likely value and 2V is the least likely value. The voltmeter is then duplicated as shown in Figure 1 and the histogram is loaded for use in the smart model.

Voltage	Count
1V	20
2V	11
3V	37
4V	20

Table 1 - Simple Example Histogram

Assuming only a single event upset (SEU);[4] the possible outcomes are described below:

- The faulty voltmeter reads 2V and the fault free voltmeter reads 3V. Since 3V has a higher count in the histogram, the smart detector will chose this voltmeter as the fault free circuit. This results in *correct detection*.

- The faulty voltmeter reads 3V and the fault free voltmeter reads 2V. Once again, since 3V has the higher count in the histogram, the smart detector will mistakenly chose this voltmeter as the "fault free" circuit. This results in *incorrect detection*.

- In the last example, one voltmeter reads 1V and the other voltmeter reads 4V. It does not matter which circuit is faulty. The histogram has the same values for both readings and therefore cannot make a determination as to which circuit may be right. This is an *ambiguous detection*.

[4] This entire paper assumes single event upsets, meaning that the circuit only contains one fault at a time.

A more detailed look at ambiguous cases, as well as how to handle them, will be treated in Section 4.

An Extension to the Statistical Smart Detector

Figure 2 shows the smart detector as just described where the decision as to which output to choose is determined by a single measurement. Figure 3 shows an extension to the smart detector that was also implemented in this work where a history of decisions was buffered into a shift register and the aggregate decision was used to determine which circuit to choose as the correct circuit. The implementation shown has an 8 deep history implemented as a shift register of values. As can be seen, two of the decisions in the past were for circuit A, but six of the decisions are for circuit B. In this case, the smart model will choose to output circuit B because the majority of decisions were for circuit B. As will be seen later, this use of history has a significant effect on the accuracy of the detector.

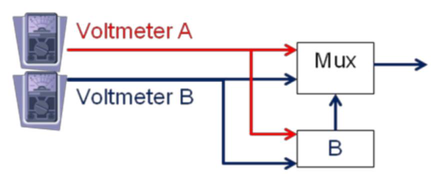


Figure 2 Single Sample Detector

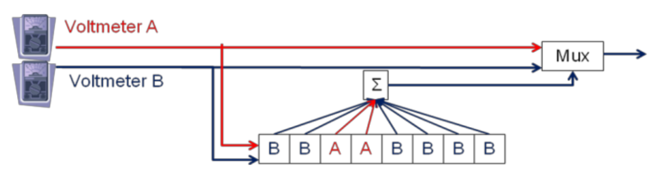


Figure 3 Detector with History

3. INITIAL TESTS

The system used for testing the proposed statistical smart model described above was a downsampler that was constructed by cascading five halfband filters as shown in Figure 4. This subsystem was taken from a QPSK design.

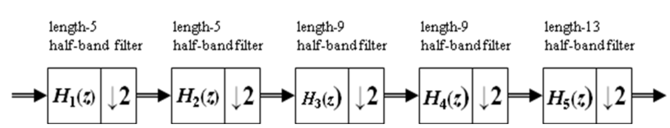


Figure 4 System used for tests

The system was designed using Xilinx System Generator within Matlab. This allowed bit-level accurate simulation and manipulation before synthesis and hardware testing. The system takes in a 12-bit value and, because of bit growth through the multiplications that occur within the filters,

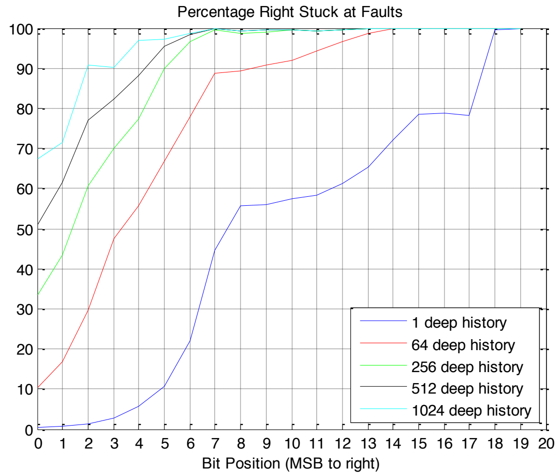


Figure 5

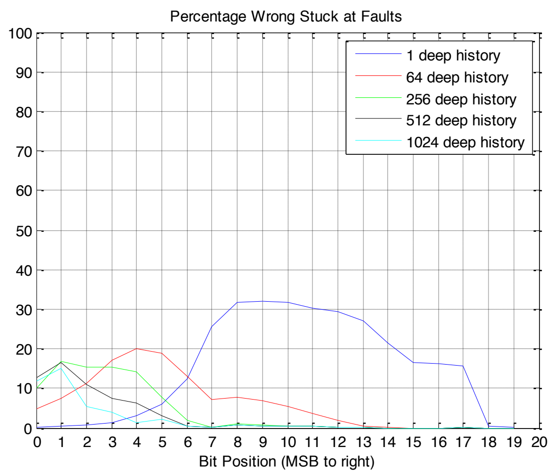


Figure 6

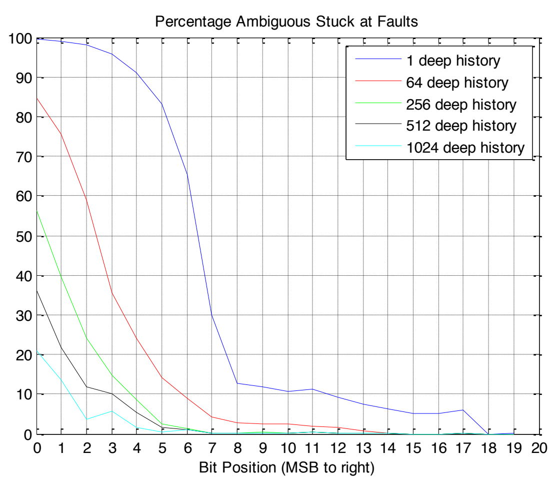


Figure 7

outputs a 20-bit filtered downsampled value. The original sample rate is 100 samples per symbol. The output sample rate is 6.25 samples per symbol.

The initial tests used to evaluate the effectiveness of the histogram approach employed a simple stuck at fault model to mimic SEUs in the system under test. Both stuck at 0 and stuck at 1 tests were performed on each bit of the 20-bit output of the downsampler. In order to approximate an SEU occurring during execution of a running system, the stuck at fault was introduced halfway through the simulation run. The smart detector was then fed the data from the "faulty" circuit as well as data from a clean circuit and prediction accuracy was calculated for each stuck at fault. Simulations were run for a model that only used 1 sample for predictions and for four different history depths. Figure 5 shows the correct detection percentages for the stuck at faults. Figure 6 shows the incorrect detections for the same faults. Figure 7 shows the ambiguous detections for the stuck at faults.

As can be noticed universally, the higher order bits are at or near 100% correct detection for all models. The bit at which the correct detection percentage is below 90% steadily moves to the smaller order bits as the history size increases. For example, a stuck at fault on bit 4 or higher with a 1024 deep history still results in over 90% prediction accuracy whereas the prediction accuracy for bit 4 using just one sample is below 10%. This supports the idea that using history substantially increases the percentage of correct detections.

4. Ambiguous Detection and Histogram Generation

Figure 7 illustrates the issue of ambiguous detection using the histogram method. When using just one sample to make the determination, it can be seen from Figures 5 and 6 that for bits 0 to 4 the percentage of incorrect detections is below 10%, but the percentage of correct detections is also below

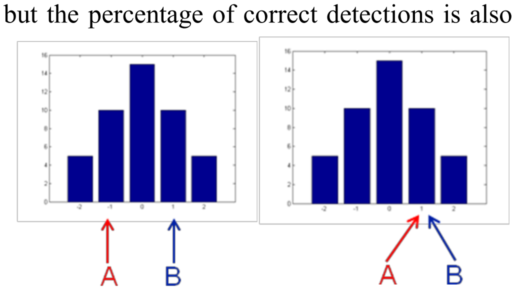
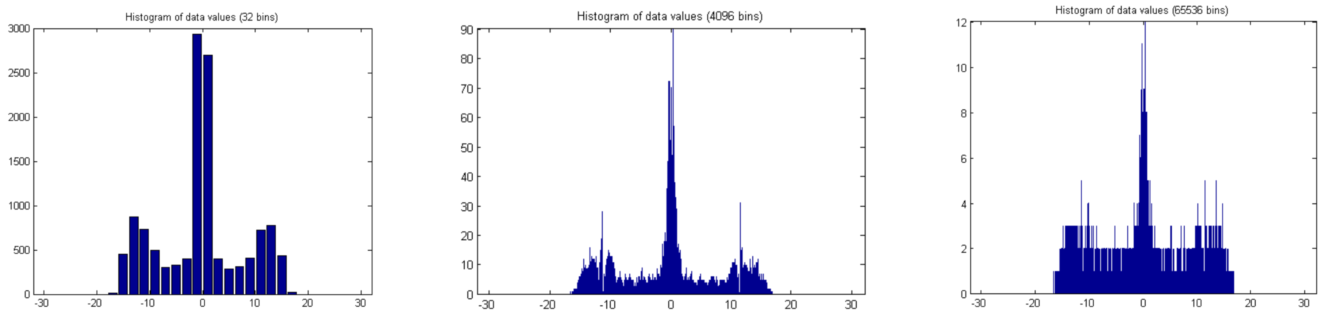


Figure 8 Two possible ways of generating an ambiguous result from the histogram

10%. The majority of detections in this range are ambiguous. Figure 7 shows that even when using history, the percentage of ambiguous detections for the low order bits is still high.

Ambiguous detection occurs in one of three ways:

1. When the values from circuit A and circuit B map to different bins that both contain the same value (as shown in the left side of Figure 8).



(a) (b) (c)

Figure 9 Three histograms with different bin counts

2. When the values of circuit A and circuit B map to the same bin. This can occur if the values are equal, but also if the difference between the two values is not enough to map it to a different histogram bin (as shown in the right side of Figure 8).

3. When using history, as in Figure 3, if the number of decisions for circuit A equal the number of decisions for circuit B.

The first two issues have direct relation to the histogram used. The histogram is generated using representative output from the system in the absence of faults. The histogram is specified by how many bins are used to count the data. For the smart detector, the number of bins is constrained to powers of 2 to simplify the hardware design. Figure 9 shows histograms generated from the same data set with different bin sizes to illustrate the tradeoffs that have to be considered by the designer. As can be seen, all of the histograms have the same general shape. Figure 9(a) shows a histogram with only 32 bins. Although the histogram would fit into a very small ROM, many of the faults would map to the same bin, resulting in an ambiguous detection. This is case 2 described above. Figure 9(c) shows a histogram generated with 65536 bins. Although the designer has significantly decreased the probability of both circuit outputs mapping to the same bin when they differ, two new concerns arise. First is the resource cost to store the larger histogram. The second concern deals with the values in the histogram. The larger histogram seems to have quantized the shape of the histogram. This arises from the fact that no given bin has many data values that map to it. For example, in this histogram many of the bins have the value of 2. This is case 1 described above. By inspection, Figure 9(b) shows what may be an acceptable medium between the two extremes.

Some level of ambiguity will always exist with this technique, because it is not possible to store a histogram that has a bin for every possible output value. As such, since an ambiguous decision can neither be verified as right or wrong, it is necessary to decide beforehand how to report them in our results of accuracy decisions. There are four possible ways to count ambiguous detections:

1. Do not count them at all. Simply remove all ambiguous detections from the calculation.

2. Record all as an incorrect detection. This would constitute a lower bound for the performance of the smart detector system.

3. Record all as a correct detection. This would constitute an upper bound for the performance of the smart detector system.

4. Record half as correct. Assuming a "fair" system, one should guess right half of the time. This is a reasonable metric for the performance of the smart detector system.

In our results, we will report all of these numbers to give a fair view of the system's performance.

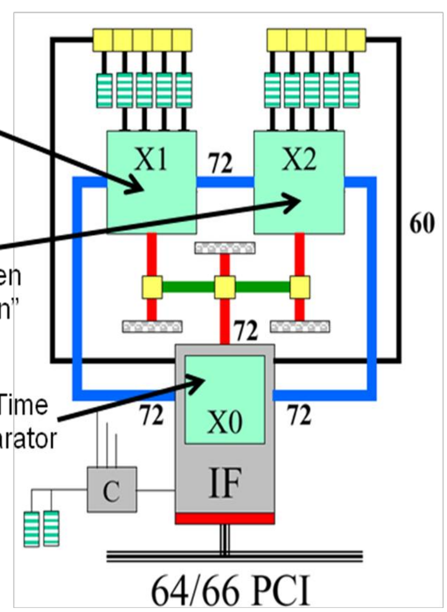


Figure 10 Virtex SEU Emulator (LANL/BYU)

Number of samples considered for decision	Percent decisions with ambiguous outcome	No ambiguous decisions counted	All ambiguous counted as wrong	Half ambiguous counted as right	All ambiguous counted as right
1	34.16%	85.03%	55.98%	73.06%	90.14%
64	15.96%	84.78%	71.25%	79.23%	87.21%
256	9.80%	84.75%	76.44%	81.34%	86.24%
512	7.55%	84.65%	78.26%	82.04%	85.81%
1024	6.03%	84.45%	79.35%	82.37%	85.38%

Table 2 Correct Decisions for 1024 bins

Number of samples considered for decision	Percent decisions with ambiguous outcome	No ambiguous decisions counted	All ambiguous counted as wrong	Half ambiguous counted as right	All ambiguous counted as right
1	28.58%	82.04%	58.59%	72.88%	87.17%
64	10.69%	81.70%	72.97%	78.31%	83.66%
256	5.98%	81.90%	77.00%	79.99%	82.98%
512	4.35%	81.89%	78.33%	80.50%	82.67%
1024	3.36%	82.35%	79.58%	81.26%	82.95%

Table 3 Correct Decisions for 4096 Bins

Number of samples considered for decision	Percent decisions with ambiguous outcome	No ambiguous decisions counted	All ambiguous counted as wrong	Half ambiguous counted as right	All ambiguous counted as right
1	25.59%	81.99%	61.01%	73.80%	86.60%
64	5.98%	84.98%	79.90%	82.89%	85.88%
256	2.47%	86.89%	84.74%	85.98%	87.21%
512	1.63%	87.26%	85.84%	86.66%	87.47%
1024	1.15%	87.34%	86.33%	86.91%	87.48%

Table 4 Correct Decisions for 16384 bins

5. Hardware Fault Injection Tests

The hardware fault injection tests performed in this work use the BYU/LANL fault injection tool. It is based on the SLAAC-1V board, which is a PCI card populated with three Virtex 1000 FPGAs. The block diagram of the system is shown in Figure 10. This tool has been previously validated using radiation testing [6] [7]; that is, it has been shown to accurately emulate SEUs caused by high energy particle strikes. The design is loaded into the X1 and X2 chips. X2 remains fault free and X1 has every bit of its configuration bitstream flipped, one at a time, to emulate a SEU. Every time a difference is noted between the output of X1 and X2, the bit is recorded as a "sensitive configuration bit". There were 73146 sensitive bits recorded for the cascaded half band downsampler design shown in Figure 4.

Test vectors were created for the design under test by feeding random numbers through a QPSK modulator in Matlab. This vector was then run through the design without injecting any faults to gather a golden output. From the golden output, histograms of various sizes were created for testing. The new input vector with a different random seed was then run through the design for each of the sensitive bits and the output was captured. To simulate real world conditions, the fault was inserted roughly halfway through the execution to give a certain amount of fault free operation. Matlab was then used to implement the smart detector and to analyze the results with varying sizes of history, as well as with varying sizes of histograms.

6. Results

Tests were run using history depths of 1, 64, 256, 512, and 1024, and with histogram sizes of 1024, 4096, and 16384 bins. The results are tabulated in Tables 2, 3, and 4. The number of decisions with an ambiguous outcome is also shown so the impact of the different history and histogram sizes can be shown. In Table 2, the number of ambiguous decisions when only considering one sample is 34.16%. This can be attributed to the small number of bins and faults being mapped to the same bin as the golden output. When a history depth of 1024 is used, the number of ambiguous outcomes drops by almost a factor of six. Still, the lower

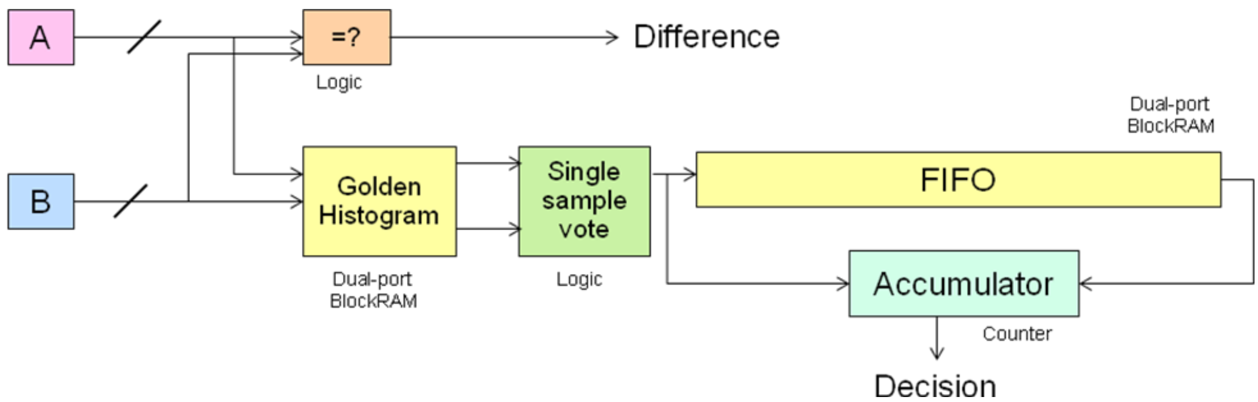


Figure 11 Block diagram of hardware implementation

bound of performance using this histogram is only 79.34% and the upper bound is 85.38%. Table 3 shows the test runs using a histogram size of 4096, the histogram size we considered in Section 4 as a good design trade-off. The percentage of ambiguous decisions drops a factor of 8.5 when moving from no history to a history buffer of 1024 samples. Table 4 shows the results for a histogram size of 16384. With four times the histogram size as the 4096 example, this smart detector might start using too many resources depending on the application, but the reduction of ambiguous decisions from 1 sample to 1024 samples is a factor of 22, and the lower bound on this system's performance is a respectable 86.33%. Since the number of ambiguous decisions is so small, the performance improvement when taking the upper bound is small.

7. Further Experiments

Pratt, et al. [8] have shown that SEUs in a communications system can be characterized into 4 classes. Class 1 and class 2 faults are characterized as non-catastrophic and have effects similar to adding noise to the system. Since communications systems are built to work in the presence of additive noise, these faults are not absolutely necessary to mitigate. Class 3 and class 4 faults cause extremely high bit error rates and are characterized as catastrophic. In the case of class 4 faults, the bit error rate is ½, meaning every other bit is in error. These are the faults that need to be mitigated.

Using the 16-bit filter with a roll-off factor of 1.0 described in [8], experiments were run to determine the performance of the histogram-based smart detector against catastrophic faults. This is a 25-tap FIR filter used in a BPSK demodulator system. As noted, the total number of sensitive bits in this design numbered 42978. The catastrophic bits were 5.9% of this total, or 2537. The experiments were run on a new hardware fault injection platform where the design under test was implemented in a Virtex 4 FPGA. Histograms of 16384 bins were generated for these experiments. As the signal through a real communications system will always have some degree of noise associated with it, histograms were generated from data with a signal to noise ratio (SNR) of 5dB and 10dB, respectively. The smart detector was then tested with these two histograms against class 3 and 4 faults using data that was statistically independent, but also had SNRs of 5dB and 10dB. For the class 3 faults, the histogram-based smart detector had an accuracy of 93%, which is a promising increase over the results in Section 6. However, for class 4 faults, the accuracy matched the results in Section 6 with accuracy in the 86^{th} percentile. This is a curious result and deserves more attention to discover the reason that the accuracy does not meet or exceed the results seen with the class 3 faults. The aggregate accuracy when used against catastrophic faults is 90.9%.

8. Hardware Implementation

Figure 11 shows a block diagram of the hardware implementation for the smart detector. The smart detector consists of some logic, a small number of block RAMs, and an accumulator. Table 5 shows resource costs for an implementation of the smart detector on a Virtex 4 with a history depth of 1024 and a histogram size of 16384 bins with a word width of 8 bits. The left column numbers are for

Logic Utilization	Smart Detector	Downsample Filter	Duplicated Downsample Filter	Triplicated Downsample Filter
# of Slices (24576 available)	48	1075	1598	2757
# of Slice FFs (49152 available)	43	630	1220	1810
# of LUTs (49152 available)	89	1310	2604	3898
# of RAMB16s (320 available)	9	0	0	0

Table 5 Logic Utilization in Virtex IV

a non-triplicated version of the smart detector. A fully triplicated smart detector will be approximately three times as large; thus, the solution using a smart detector requires approximately 1750 slices, while a fully triplicated design is approximately 2750 slices.

9. Conclusions and Future Work

In this paper, we have discussed smart detection using a simple histogram. As can be seen from the results presented, this technique offers a relatively high degree of accuracy with very low resource costs. Although the reliability is not as high as with TMR, it succeeds in its stated purpose of trading off a small amount of reliability for a large reduction in resource requirements. Specifically, a non-triplicated version of the smart detector paired with the duplicated circuit is 60% the size of the triplicated downsample filter. A conservative estimate of the triplicated smart detector at 6 times the resource costs paired with the duplicated downsample filter is still 70% the size of the triplicated circuit. As would be expected, the resource efficiency of this technique increases as the size of the circuit it protects grows. If 100% fault mitigation is not needed, this is an attractive technique with high reliability for a very low resource cost—particularly for catastrophic faults where the accuracy exceeds 90%.

The filters used for testing in this work were extracted from larger demodulator systems and are the simplest blocks in that system. Future work will focus on applying smart detection methods to an entire demodulator system. An important question will be to determine the best locations within the design to probe with the smart model. Also, other possibilities exist besides the histogram method for the implementation of the smart model. Further research is investigating machine learning with neural networks and Bayesian networks as the smart model.

References

[1] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Xilinx Corporation, Tech. Rep., November 1, 2001, xAPP197 (v1.0).

[2] Kastensmidt, F.L.; Sterpone, L.; Carro, L.; Reorda, M.S., "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 1290-1295 Vol. 2, 7-11 March 2005.

[3] Morgan, K.S.; McMurtrey, D.L.; Pratt, B.H.; Wirthlin, M.J., "A Comparison of TMR With Alternative Fault-Tolerant Design Techniques for FPGAs," *Nuclear Science, IEEE Transactions on* , vol. 54, no. 6, pp. 2065-2072, Dec. 2007.

[4] N. Rollins, M. J. Wirthlin, and P. S. Graham, "Evaluation of power costs in triplicated FPGA designs," in *Proceedings of the Military and Aerospace Programmable Logic Devices International Conference (MAPLD)*, Washington, D.C., September 2004.

[5] Johnson, J.; Howes, W.; Wirthlin, M.; McMurtrey, D.L.; Caffrey, M.; Graham, P.; Morgan, K., "Using Duplication with Compare for On-line Error Detection in FPGA-based Designs," *Aerospace Conference, 2008 IEEE*, pp. 1-11, 1-8 March 2008

[6] Wirthlin, M.; Johnson, E.; Rollins, N.; Caffrey, M.; Graham, P., "The reliability of FPGA circuit designs in the presence of radiation induced configuration upsets," *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*, pp. 133-142, 9-11 April 2003

[7] Johnson, E.; Caffrey, M.; Graham, P.; Rollins, N.; Wirthlin, M., "Accelerator validation of an FPGA SEU simulator," *Nuclear Science, IEEE Transactions on*, vol. 50, no. 6, pp. 2147-2157, Dec. 2003

[8] B. Pratt, M. Fuller, M. Rice, and M. Wirthlin, "Reliable communications using FPGAs in high radiation environments – Part 1: Characterization," submitted to ICC 2010

Biography

Jon-Paul Anderson received the B.S. degree in electrical engineering from Brigham Young University, Provo, UT in 2000. After completing the B.S. degree, he was a System Engineer at Northrop Grumman Electromagnetic Systems

Laboratory in San Jose, CA. He is currently pursuing the Ph.D. degree in electrical engineering from Brigham Young University.

Brent Nelson is a professor in the Department of Electrical and Computer Engineering at Brigham Young University and program head for the Computer Engineering program there. He received his Ph.D. in computer science in 1984 from the University of Utah in the area of VLSI CAD. His current research interests focus on two main areas: high-end computing applications of reconfigurable computing and CAD for the design of FPGA-based applications. He currently serves as co-director for the NSF Center for Reconfigurable High Performance Computing (known as CHREC) and as director of the BYU site within that center.

Michael J. Wirthlin received the B.S. and Ph.D. degrees from Brigham Young University, Provo, UT in 1992 and 1997, respectively. After completing the Ph.D. degree, he was Staff Research Engineer with the Systems Architecture Laboratory, National Semiconductor Corporation in Santa Clara, CA. He is currently an Associate Professor with the Department of Electrical and Computer Engineering at Brigham Young University. His research interests include configurable computing systems, FPGA reliability, fault-tolerant computing, high-level synthesis, and computer-aided design for application specific computing.