Reduced Triple Modular Redundancy for Tolerating SEUs in SRAM-based FPGAs

Vikram Chandrasekhar Sk Noor Mahammad V Muralidaran V Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai
kama@iitm.ernet.in

N Vijaykrishnan
Department of Computer Science and Engineering
Pennsylvania State University
USA

Abstract

This paper proposes an efficient alternative to Triple Modular Redundancy for SEU mitigation in SRAM-based FPGAs. The new technique, Reduced Triple Modular Redundancy (RTMR), operates on a lookup-table (LUT) network obtained after the technology mapping stage. The entire set of LUTs is classified on the basis of sensitivity *into* sensitive, internally insensitive *and* last-level insensitive *LUTs. Instead of triplicating the entire design, only the first and third category of LUTs are triplicated and tri-buffer based majority voters are used. The classification is first performed using signal probability propagation and later more successfully by fault simulation. By utilizing the unused flip-flops of the CLBs in combination circuits, certain* last-level insensitive *LUTs can just be duplicated instead of being TMRed. The RTMR technique is tested on the MCNC '91 benchmarks and the results for the 8 largest combinational circuits is presented here. On an average RTMR requires only **99.61**% additional number of LUTs compared to the 200% additional LUTs required by a standard TMR. Even with such a low area requirement, the circuits produced by the RTMR technique are observed to have a very high SEU immunity.*

1. Introduction

Modern SRAM based FPGAs provide high performance at a very low NRE (Non-Recurring Engineering) cost. The reconfigurability feature of the SRAM-based FPGAs make them very suitable for applications that involve frequent modifications to the design after deploying it on field. For example, space applications benefit a lot by the reconfigurability feature by allowing in-orbit design changes, with the aim of reducing the mission cost by correcting errors or improving system performance after launch.

A Single Event Upset (SEU) is a fault that is quite common in the SRAM-based FPGAs. SEUs are circuit errors caused due to excess charge carriers induced primarily by external radiations. Radiation directly or indirectly induces a localized ionization capable of upsetting internal data states. While these errors cause an upset event, the circuit itself is not damaged. These errors are particularly troublesome for memory elements (Routing configuration bits and LUT entries) as the stored values of the bits are changed,[15].

2. Previous Work and RTMR

Many solutions for mitigating SEUs in FPGAs have been proposed in the past, [8], [5],[2],[9],[16],[3],[4]. Other methods for hardening logic circuits against SEUs have also been proposed in [10],[14].

TMR (Triple Modular Redundancy) and its variants are most commonly used for SEU mitigation,[7],[11],[6]. [6] describes a combination of an error correcting code and TMR in protecting an FPGA from SEUs. The main disadvantage of Triple Modular Redundancy(TMR) is the excessive area overhead. The hardened design has 200% more area than the original circuit. A technique known as Selective Triple Modular Redundancy (STMR) is proposed in [12] that employs partial redundancy to introduce SEU immunity. The STMR technique operates on a gate-level circuit, identifying sensitive gates and triplicating only such gates. However, it must be noted that the LUT network obtained after the technology mapping stage, is very different from the gate-level circuit. Hence, the area savings at the gate-level circuit are diminished after the technology mapping stage. Moreover, in most FPGA CAD tools, optimization techniques during the technology mapping stage are designed to remove redundancy in the circuit before obtaining the final LUT network.

In this paper, an SEU mitigation technique known as Reduced TMR (RTMR) is proposed that hardens an LUT-based design against SEUs. In the LUT network obtained from a circuit's technology mapping. Redundant copies of only a reduced set of LUTs are created to introduce SEU immunity. This reduced set of LUTs is first identified using an extension of a standard gate-level technique described in [12]. However, due to the failure of such a classification at the LUT-level, an alternative fault simulation technique is proposed. This alternative technique is shown to produce very accurate classifications resulting in a high SEU immunity at a much lower cost of area. The proposed RTMR algorithm is then further refined for combinational circuits by exploiting an architectural feature available in standard FPGAs. By implementation on several large MCNC benchmark circuits, the refined RTMR technique is shown to require a much lesser area redundancy compared to a standard TMR for nearly the same SEU immunity.

3. Sensitive and Insensitive LUTs

The concept of sensitivity is well known in the context of logic gates. The same concept can be easily extended to a 4-input look-up table. An LUT is defined to be *sensitive* if a change in any one of its current input values changes the current output value. The opposite kind of LUT is an *insensitive* LUT which does not allow a change in a current input value to affect the current output value. all the five bits for the function in an LUT have the same value, the LUT is defined to be *insensitive*. An insensitive LUT acts as a block against any SEU effects that are propagated up to its inputs.

If the effect of an SEU, as seen at any LUT, is only a single input value, *insensitive* LUTs should be quite effective as SEU-stoppers. The original LUT network is converted into a redundant network with additional copies of only sensitive LUTs. If there is a chain of *sensitive* LUTs between two insensitive LUTs, two additional copies of the chain are created. A tri-state majority voter is placed at the end of the three chains, whose output is connected to the insensitive LUT. If an SEU occurs inside a *sensitive* LUT chain, the voter stops the wrong value from propagation. If an SEU affects the output of an *insensitive* LUT, the error is stopped at the next *insensitive* LUT on the path to a primary output.

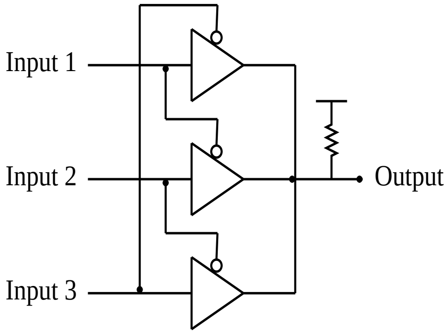

Figure 1. Tri-state majority voter

We define a *last-level insensitive* LUT as an *insensitive* LUT that can reach a primary output(PO) through a chain of only *sensitive* LUTs. Such LUTs need to be triplicated since an SEU that affects the output of a *last-level insensitive* LUT will propagate to a primary output.

Thus, the required redundancy for SEU mitigation can be restricted to only the sets of *sensitive* and *last-level insensitive* LUTs. In later sections, we show how even the redundancy required for *last-level insensitive* LUTs can be reduced in some cases.

4. Classification of LUTs

The partitioning of the original set of LUTs into *sensitive* and *insensitive* LUTs is a very critical step in deciding the *area* vs *SEU immunity* trade-off. A technique used in previous works concerning sensitivity of logic gates[12] is *signal probability propagation*. First, we show how this technique is extended to the case of 4-input LUTs. But this technique fails in the case of 4-input LUTs resulting in either erroneous or in very high-cost classifications. A simpler alternative, fault simulation, is proposed to perform the sensitivity classification. This technique is shown in later sections to achieve very accurate low-cost classifications. Once the classification is performed, the RTMR technique can be used to create SEU-immune circuits.

4.1. Signal probability propagation

The problem of computing signal probability (CSP) is commonly encountered during the analysis and testing of faults [1],[12]. Here, we show how the concept is extended from the gate-level to the LUT-level. The signal probability (SP) of a line is defined as the probability that the line will carry a value '1' at any point of time. Given the SPs of the primary inputs of an LUT network, the SP of every net in the network can be determined by propagating the SPs level-by-level till the primary outputs. Once this step is over, a threshold probability must be fixed in order to estimate the most probable value carried by a line using its SP. If a line's SP is greater than the threshold probability, the line is expected to assume a logic value 1, else it is expected to assume the logic value 0. The calculation of the SP of an LUT's output, given its input SPs, is dependent on the function stored in the LUT. Once again, the focus is on a 4-input LUT storing a 16-bit truth table in a 16-bit SRAM.

Let the input signal probabilities of an LUT, storing the function $F(I_1, I_2, I_3, I_4)$ with inputs I_1, I_2, I_3,I_4, be P_1, P_2, P_3 and P_4. The SP of the LUT output is simply equal to the probability of an accessed SRAM cell of the LUT holding a logic value 1, is computed. Let $M_i \in 0, 1$, for $1 \leq i \leq 4$, and V = $F(M_1, M_2, M_3, M_4)$

$$S(M_1, M_2, M_3, M_4) =$$

$$\begin{cases} R(M_1, P_1) \times R(M_2, P_2) \times R(M_3, P_3) \times R(M_4, P_4) \\ \quad\quad \text{if } V = 1 \\ 0 \quad\quad \text{if } V = 0 \end{cases}$$

where

$$R(A, B) = \begin{cases} B & \text{if } A = 1 \\ 1 - B & \text{if } A = 0 \end{cases}$$

If $F(M_1, M_2, M_3, M_4)$ is equal to 1, then $S(M_1, M_2, M_3, M_4)$ = 0. The output SP is calculated as

$$\sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} \sum_{l=0}^{1} S(i, j, k, l)$$

.

Though this technique is shown to produce very successful classifications with gates in [12], the extended technique for LUTs does not prove to be effective. The problem is that the SP values computed by the method described above vary over the entire [0-1] range. Thus, fixing a threshold probability in order to determine the sensitivity of the LUTs becomes very difficult. Even after experimenting with a number of threshold probabilities, we find that the resulting hardened circuits with the RTMR technique have poor SEU immunity compared to the high area cost. We also tried the use of two thresholds for determining the probable values of the lines. Let L and H be the two thresholds. If SP ¡ L, the line is assumed to carry '0' and if SP ¿ H, the line is assumed to carry '1'. Even in this case, a large number of LUTs are left unclassified though some LUTs can be correctly classified. These LUTs cannot be considered insensitive as this results in an even poorer SEU immunity in the final circuit. These problems faced with the signal probability propagation technique at the LUT-level indicate the need for an alternative sensitivity classification technique.

4.2. Fault simulation

In later sections, we describe an LUT-level SEU simulator that was designed to determining the SEU immunity of a circuit. However, in this section we use the same fault simulator to classify LUTs as either *sensitive* or *insensitive* without much additional effort. The basic idea is that if over a large number of fault simulations, a particular LUT is repeatedly *sensitized*, then the LUT can be classified as *sensitive*. This method though quite simple and seemingly brute-force, is shown later to be very effective. For a given circuit, a large number faults are simulated, each fault with its own random primary input values. For every simulated fault, a list of *sensitized* LUTs was obtained. The list comprises of all LUTs whose outputs took a different value in the presence of the induced SEU with the same primary input values. A simple measure of an LUT's sensitivity is the number of *sensitized* LUT lists that contain the given LUT. Again, we see that a threshold value is required. However, this threshold value is found to be quite independent of the circuit and directly affects the SEU immunity of the circuit. Therefore, this threshold value can be used to trade off the total number of LUTs for SEU immunity in the new redundant circuit. A higher threshold allows a smaller number of LUTs to be labelled as *sensitive* but this leads to a lower SEU immunity.

5. Proposed Reduced Triple Modular Redundancy

In this section, the algorithm for the RTMR technique is described. A three pass method is adopted for the sake of simplicity and ease of extension to sequential circuits which is part of our future work.

5.1. Generic RTMR algorithm

Using the fault simulation technique described in the previous section, the entire set of LUTs is partitioned into the following,
S - Sensitive LUTs
L - Last level insensitive LUTs
I - Internal insensitive LUTs
We define $T = L \cup S$, which is the set of LUTs to be triplicated

```
for all LUTs $L \epsilon T$ do
    Create two more copies $L$ and a majority voter
    Connect the outputs of $L$ and its copies to its voter
end for
for all $L \epsilon I$ do
    for all inputs $P \epsilon T$ do
        Connect voted output of $P$ to $L$ instead of $P$
    end for
end for
for all $L \epsilon T$ do
    for all inputs $P$ of $L$ do
        if $P \epsilon S$ then
```

```
13:       Connect each copy of $P$ to corresponding copies of $L$
14:     else if $P \epsilon L$ then
15:       Connect voted output of $P$ to $L$ and its two copies
16:     end if
17:   end for
18: end for
19: Remove all unconnected LUTs and voters
```

Steps 1-4 form the first pass which triplicates all the LUTs that are *sensitive* or *last-level insensitive* LUTs. Steps 5-18 form the second pass which involves the connection of nets to the new copies as well as the voters. For an *insensitive* LUT, each *sensitive* or *last-level insensitive* input is replaced by its voter's output. For a *sensitive* LUT, if an input is also *sensitive*, each copy of the input are connected to the corresponding copy of the LUT. However, in the case of a *last-level insensitive* input, the voted output is connected to the LUT in place of the input. Finally, all the additional LUT copies and voters that were not connected as inputs to any other LUTs, are removed from the circuit. This algorithm produces a hardened circuit which a high level of SEU immunity by triplicating only a subset of LUTs.

5.2. MPV technique

The main sources of redundancy seen in the RTMR algorithm above are the *sensitive* and *last-level insensitive* LUTs. Here, an existing provision in the architecture of standard FPGAs is exploited to reduce the redundancy due to *last-level insensitive* LUTs. *Insensitive* LUTs, as seen in the signal probability propagation technique are also LUTs which have mostly have a fixed value as the output. Interestingly, during the process of fault simulation, it is observed that LUTs that are classified as *insensitive*, almost have a constant output value. This observation is used to replace the triplication process of *last-level insensitive* LUTs by a duplication process. A basic logic element (BLE) consists a 4-input LUT as well as a flip-flop that can also be connected to any output pin of the Configurable logic block. The flip-flop as seen in a standard Virtex architecture [13] has an asynchronous set/reset control which is part of the FPGA configuration memory. Though a sequential circuit has predefined initialization values for these flip-flops, a combinational circuit does not use these flip-flops at all. This very flip-flop, in the case of a *last-level insensitive LUT* can be used as part of the SEU mitigation logic. A TMR for such an LUT would consist of three LUTs having the same set of inputs connected to a majority voter circuit. Instead, we use two such LUTs with the flip-flop of one LUT containing the most probable output value (MPV) of the LUT. The clock signal to this flip-flop is not connected at all to the system clock as it is a combinational circuit, which ensures that the stored value is not re-written. The flip-flop is programmed to hold the desired value by simply specifying the asynchronous appropriate set/preset control in the configuration memory. The outputs of the two LUTs as well as the flip-flop output are connected to a majority voter circuit as in TMR. If an SEU affects the output of one of the *insensitive* LUTs, the output of the majority circuit is the MPV of the LUT. In the absence of an SEU, the outputs of the two LUTs will be the output of the voter, regardless of the MPV of the LUT. This simple addition to the generic RTMR algorithm, which we call as the MPV technique, causes a significant drop in the required redundancy for the same level of SEU immunity. This is explained by the fact that a large percentage of insensitive LUTs are *last-level insensitive* LUTs due to techniques such as depth optimization employed during technology-mapping. Since many *last-level insensitive* LUTs are observed to have a constant output value, the MPV technique is very effective in reducing the required redundancy.

However, not all *insensitive* LUTs identified by fault simulation display a constant output value during the entire simulation. Hence an additional set of threshold values is required for finding *constant-output last-level* LUTs. However, these threshold values are much easier to fix unlike the signal probability threshold values.

The RTMR algorithm for combinational circuits incorporates a few additional steps into the previous RTMR algorithm.

The entire set of LUTs is partitioned into the following,
S - Sensitive LUTs
L - indeterminate last level insensitive LUTs
D - determinate constant-output last level insensitive LUTs
I - Internal insensitive LUTs
We define $T = L \cup S$, which is the set of LUTs to be triplicated

```
for all LUTs $L \epsilon T$ do
  Create two more copies $L$ and a majority voter
  Connect the outputs of $L$ and its copies to its voter
end for
for all LUTs $L \epsilon D$ do
  Duplicate $L$, fix the flipflop of $L$ to the MPV
  Create a majority voter for $L$
  Connect $L$, its copy and the flip-flop to the voter
end for
for all $L \epsilon I$ do
  for all inputs $P \epsilon T$ or $P \epsilon D$ do
    Connect voted output of $P$ to $L$ instead of $P$
  end for
end for
for all $L \epsilon D$ do
  for all inputs $P$ of $L$ do
    if $P \epsilon T$ then
      Connect voted output of $P$ to $L$ and its copy
```

```
        else
            Connect $P$ to the additional copy of $L$
        end if
    end for
end for
for all $L \epsilon T$ do
    for all inputs $P$ of $L$ do
        if $P \epsilon S$ then
            Connect each copy of $P$ to corresponding copies of $L$
        else if $P \epsilon L$ then
            Connect voted output of $P$ to $L$ and its two copies
        end if
    end for
end for
Remove all unconnected LUTs and voters
```

6. SEU simulation

This section describes the SEU fault simulator that was designed in order to both support as well as verify the RTMR technique. The fault simulator uses the open source Icarus Verilog compiler and the Virtual Verilog Processor (VVP) simulator. Using the technology mapping of the circuit, a Verilog model is constructed, which consists of a network of 16-bit registers, 16-to-1 4-select multiplexers and flip-flops. A single 16-bit register and a multiplexer represent a 4-input LUT. Though the primary purpose of the simulator is to observe the SEU immunity of the circuit, other information such as *sensitivity lists* as described in earlier sections must also be gathered. An SEU is simulated by forcing a randomly chosen net to the opposite of its value observed in logic simulation without the SEU. When an SEU is simulated on a particular net, only those LUTs encountered on any path to a primary output, as well as the primary outputs, need to be monitored. A single fault simulation consists of the following steps

1. Randomly assign values to primary inputs
2. Randomly choose the net to be affected
3. Perform logic simulation and record monitored values
4. Force chosen net to opposite value
5. Perform logic simulation and record monitored values
6. Construct *sensitivity list* for the fault

7. Experimental Results

The proposed RTMR technique was implemented on standard MCNC benchmark circuits to evaluate the immunity of the RTMRed circuits as well the savings in redundancy. The sensitivity classification of LUTs was performed by simulating 10,000 faults for each circuit. The SEU immunity of the RTMRed circuits is compared to that of the original circuits by simulating 1000 faults in each case and observing the number of faults propagated to the primary outputs. The results for only 8 of the largest combinational MCNC benchmarks is shown here due to a lack of space. Its worth noting that each of these circuits have more than 1000 LUTs which makes RTMR a very scalable technique. Table1 shows the results for the generic RTMR algorithm without the use of the MPV technique. The number of propagated faults out of the 1000 applied faults, is shown for both the original as well as the RTMR circuit. Even without the MPV technique, the RTMR algorithm requires only

Circuit	No. of LUTs in original circuit	No. of faults	No. of LUTs in RTMR circuit	No. of faults	% of extra LUTs
alu4	1522	281	3004	25	97.37
apex2	1878	349	4054	5	115.87
des	1591	691	3967	71	149.34
misex3	1397	509	2935	30	110.09
pdc	4575	238	9593	24	109.68
seq	1750	412	3482	21	98.97
spla	3690	477	8070	27	118.7
ex5p	1064	394	2532	53	137.97

Table 1. Comparison of the area overheads of TMR and RMTR circuits

Circuit	No. of LUTs in original circuit	No. of faults	No. of LUTs in RTMR circuit	No. of faults	% of extra LUTs
alu4	1522	281	2959	7	94.42
apex2	1878	349	2974	0	58.36
des	1591	691	3850	68	141.99
misex3	1397	509	2723	17	94.92
pdc	4575	238	8661	13	89.31
seq	1750	412	3371	7	92.63
spla	3690	477	7222	13	95.72
ex5p	1064	394	2442	21	129.51

Table 2. Comparison of the area overheads of TMR and RMTR circuits

117.25% additional redundancy compared to the 200% additional redundancy requirement of a standard TMR. However, the tradeoff comes at the expense of a marginal loss of SEU immunity due to the high threshold for identifying *sensitive* LUTs from the *sensitivity lists*. Table2 shows the new results for the same benchmark circuits with the addition of the MPV technique to the RTMR algorithm. Here the threshold for the identification of *sensitive* LUTs is very low. As a result, a larger number of LUTs are classified as *sensitive* compared to the generic RTMR technique. However, by using only a duplication process instead of triplication for the *constant-output last-level insensitive* LUTs, the total number of additional LUTs is lowered. The new RTMR technique requires only **99.61**% additional redundancy compared to the 200% requirement of standard TMR. Since the number of LUTs classified as *sensitive* has been increased, the SEU immunity of the hardened circuit is much better than that achieved by the generic RTMR algorithm. It can be seen that for any benchmark circuit, the number of propagated faults in Table2 is significantly lower than the number of propagated faults in Table1.

8. Conclusion and Future work

An efficient SEU mitigation technique known as Reduced TMR (RTMR) has been proposed for SEU tolerance in SRAM-based FPGAs. The LUTs are classified based on their sensitivity to SEUs and only a reduced set of LUTs are TMRed. The classification is very accurately performed by fault simulation with a fault simulator designed to evaluate a circuit's immunity to SEUs. Furthermore, the unused storage elements in a CLB of a combinational circuit are used to avoid triplication of a certain set of LUTs. For 8 of the largest combinational MCNC benchmark circuits, RTMR produces very low-cost SEU-hardened circuits.

Our future work will deal with the hardening of sequential circuits to SEUs in a more area-efficient manner than the standard TMR. The tradeoff between Area vs SEU immunity will also be studied in greater detail.

References

[1] G. Asadi and M. Tahoori. An accurate SER estimation method based on propagation probability [soft error rate]. *Proceedings of Design, Automation and Test in Europe 2005*, pages 306–307, 2005.

[2] G. Asadi and M. B. Tahoori. Soft error rate estimation and mitigation for SRAM-based FPGAs. *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pages 149–160, February 2005.

[3] P. Blain, C. Carmichael, E. Fuller, and M. Caffrey. SEU mitigation techniques for Virtex FPGAs in space applications. *[Online] http://www.xilinx.com/appnotes/VtxSEU.pdf*.

[4] M. Garvie and A. Thompson. Scrubbing Away Transients and Jiggling Around the Permanent: Long Survival of FPGA Systems Through Evolutionary Self-Repair. *Proceedings of the 10th IEEE International On-Line Testing Symposium*, pages 155–160, July 2004.

[5] M. Gokhale, P. Graham, E. Johnson, N. Rollins, and M. Wirthlin. Dynamic reconfiguration for management of radiation-induced faults in FPGAs. *Proceedings of the 18th Parallel and Distributed Processing Symposium*, page 145, april 2004.

[6] R. Hentschke, F. Marques, F. Lima, L. Carro, A. Susin, and R. Ries. Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy. *Proceedings of the 15th Symposium on Integrated Circuits and Systems Design*, page 95, September 2002.

[7] F. L. Kastensmidt, L. Sterpone, L. Carro, and M. S. Reorda. On the Optimal Design of Triple Modular Redundancy Logic for SRAM-based FPGAs. *Proceedings of Design, Automation and Test in Europe*, pages 1290–1295, 2005.

[8] F. Lima, L. Carro, and R. Reis. Designing Fault Tolerant Systems into SRAM-based FPGAs. *Proceedings of the Design Automation Conference*, pages 650–655, June 2003.

[9] G. Mojoli, D. Salvi, M. Sami, G. Sechi, and R. Stefanelli. KITE: a behavioural approach to fault-tolerance in FPGA-based systems. *1996 Workshop on Defect and Fault-Tolerance in VLSI Systems*, page 327, November 1996.

[10] M. Nicolaidis. Design for Soft-Error Mitigation. *IEEE Transactions on Device and Materials Reliability*.

[11] N. Rollins, M. Wirthlin, P. Graham, and M. Caffrey. Evaluating TMR Techniques in the Presence of Single Event Upsets. *Proceedings of 6th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, May 2003.

[12] P. Samudrala, J. Ramos, and S. Katkoori. Selective triple Modular redundancy (stmr) based single-event upset (seu) tolerant synthesis for FPGAs. *IEEE Transactions on Nuclear Science*, pages 2957–2969, October 2004.

[13] P. Samudrala, J. Ramos, and S. Katkoori. Virtex-II Platform FPGAs: Complete Data Sheet, DS031(v3.3). *Xilinx Product Specification*, 2004.

[14] Q. Shi and G. Maki. New design techniques for SEU immune circuits. *Proceedings of 9th Nasa Symposium on VLSI Design*, pages 4.2.1–4.2.16, 2000.

[15] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham. The Reliability of FPGA Circuit Designs in the Presence of Radiation Induced Configuration Upsets. *Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, page 133, April 2003.

[16] M. J. Wirthlin, N. Rollins, M. Caffrey, and P. Graham. Hardness by design techniques for field-programmable gate arrays. *Proceedings of the 11th Annual NASA Symposium on VLSI Design*, pages WA11.1–WA11.6, May 2003.