Improving Soft-error Tolerance of FPGA Configuration Bits

Suresh Srinivasan, Aman Gayasen, N. Vijaykrishnan, M. Kandemir, Y. Xie, M.J. Irwin
Dept. of Computer Science and Engineering, Pennsylvania State University, PA 16802
{ssriniva,gayasen,yuanxie,vijay,kandemir,mji}@cse.psu.edu

Abstract— **Soft errors that change configuration bits of an SRAM based FPGA modify the functionality of the design. The proliferation of FPGA devices in various critical applications makes it important to increase their immunity to soft errors. In this work, we propose the use of an asymmetric SRAM (ASRAM) structure that is optimized for soft error immunity and leakage when storing a preferred value. The key to our approach is the observation that the configuration bitstream is composed of 87% of zeros across different designs. Consequently, the use of ASRAM cell optimized for storing a zero (ASRAM-0) reduces the failure in time by 25% as compared to the original design. We also present an optimization that increases the number of zeros in the bitstream while preserving the functionality.**

I. Introduction

SRAM-based field programmable gate arrays are being increasingly used to start new designs because of their growing density and speed, short-design cycle and cost-effectiveness. In such devices the configuration bitstream determines the routing and functionality of the design. However, an inadvertent change in the value of one of the SRAM cells can potentially modify the functionality of the design or even cause a permanent damage to the cell [9]. A major reason for such inadvertent bit flips is due to soft errors caused by cosmic radiation or the radiation emanating from materials used in packaging and processing. When these particles impact the silicon substrate, they result in the generation of excess carriers, which when deposited on the internal capacitances of a circuit node can result in upsetting the data value stored. The lowering supply voltages and the nodal capacitances with newer technologies have increased the possibility of such excess carriers of creating a bit flip.

Due to this increasing concern, there are several approaches that have been proposed for tackling the soft error problem [4]. One method of SEU correction is to use readback to detect when an upset to the configuration memory has occurred and reload only the particular frame. A simpler method for SEU correction is to omit readback and detection of SEUs and simply reload the entire CLB Frame segment at a chosen interval. This technique is called scrubbing. However, not all the bits in a FPGA are important as only a portion of these reconfiguration bits are actually used by the design. These bits are defined as "care-bits" [11]. Consequently, SEU correction and scrubbing can be performed only for segments or portions of the FPGA containing the care-bits. Modular redundancy techniques have also been investigated for protecting the design against soft errors [5] [10].

Our work is based on the use of an asymmetric SRAM cell (ASRAM) that has a lower leakage and a better soft-error immunity when storing a preferred data value. Since our experimentation with different designs shows that 87% of the configuration bits are zero we propose to use ASRAM-0 cells for the configuration bits. A reduction of error rate by 25% on an average was observed which was further reduced 4-8% by our proposed optimization scheme.

In addition to reducing soft error rates, the use of ASRAM-0 cells reduces leakage energy consumed by the configuration SRAMs by a factor of 18 as compared to using standard SRAM cells. This leakage reduction is significant, considering that 38% of overall leakage in a 90nm FPGA is consumed by the configuration bits.

II. FPGA architecture

We have chosen island-style SRAM-based FPGAs for our study. Xilinx and Altera's FPGAs fall into this category, and all our experimentation was performed using Xilinx's Virtex-II FPGAs.

Two of the major components that are configured using (configuration) SRAMs are the LUTs and routing multiplexers. Note that both of them are multiplexers (muxes), the key difference being in the role the configuration SRAM plays in them. While in the LUT-mux, the input signals are driven by configuration SRAMs, in case of a routing-mux, configuration SRAMs drive the 'select' signals of the mux. Consequently, a flip in the SRAM cell can change the functionality implemented in the LUT or the connection made by the routing-mux, both of which can disrupt the correct functioning of the design, and therefore must be avoided. In case of the routing mux due to the presence of fully decoded mux in the first level, this problem can become more severe when a flip in the SRAM bit creates a short between '1' and '0' and potentially damage the device.

Note that SEUs in only those configuration SRAM bits that drive used resources on the device are critical. It has been shown that the number of critical configuration SRAM bits (care-bits) is significantly less than the total configuration bits and was reported by [11] to be less than 40% of total configuration SRAM bits.

An interesting characteristic of configuration care-bits is that a majority of them are zero. For the designs we used for our experimentation, we observed that on an average 75% of the care-bits were zeros, out of which, 90% were the bits

0-7803-8702-3/04/$20.00 ©2004 IEEE.

configuring routing muxes. Such a large number of zeros in the routing configuration bits can attributed to the fact that, most of the routing multiplexers comprise of one level of fully decoded multiplexers.

Since most of the configuration bits are zero, we would want to use an ASRAM-0 cell to reduce leakage in configuration SRAM. It has been shown that 38% of the leakage in a 90nm FPGA occurs in the configuration SRAM [12]. Therefore, reducing its leakage will reduce the total FPGA leakage significantly.

At the same time, using ASRAM-0 cells for the configuration bits will also reduce the susceptibility to soft errors of those configuration bits that store a value of 0, as will be detailed in the next section. LUT bits do not show such a strong preference for value '0'. It was observed that on an average only 42% of the LUT SRAM bits were '0'. Consequently, we try to maximize the number of zeros in LUT bits by restructuring the logic of the design, as will be explained in section IV.

III. Asymmetric SRAM

The asymmetric SRAM (ASRAM) design was originally proposed for leakage reduction [2]. In ASRAM, we carefully select the V_t of the transistors in the SRAM cell for the leakage reduction. Two different SRAM cells are proposed that are optimized for leakage while storing a value of zero, referred to as ASRAM-0 (see Fig 1) and while storing a value of one, referred to as ASRAM-1. Figure 1 shows the ASRAM-0 cell in which only the transistors that are normally leaking for a stored value of zero are made high V_t. Similarly, in case of the ASRAM-1 cell the transistors that leak for a stored value of one are made high V_t. The resultant design is asymmetric with respect to V_t; hence it is called an asymmetric SRAM. When used with a specialized sense amplifier design, the asymmetric cells can reduce the leakage power at the expense of a small performance penalty due to high V_t transistors. The soft error vulnerabilities of these cells can be potentially influenced by the presence of the high threshold voltage transistors [6].

We used a formulation developed in [7] to estimate the SER in SRAM circuits. This model shows that the soft error rate is an exponential function of $Q_{critical}$ and Q_s. $Q_{critical}$ is the critical charge that should be generated by the incident radiation event to cause an upset, Q_s describes the total charge collection efficiency at the point of the impact.

In Table I, the $Q_{critical}$ of conventional SRAM and ASRAM is compared. These values are obtained from custom layout of the designs in 70nm Berkeley Predictive Technology Model [3]. The methodology presented in [6] is used in obtaining the critical values.

Note that the $Q_{critical}$ is different for the bit flip on different nodes of the ASRAM cells unlike that of the normal SRAM cells due to the assymmetry in the ASRAM cell. However for both the nodes when they store their preferred values, the $Q_{critical}$ at both Q and !Q are higher than that of the standard

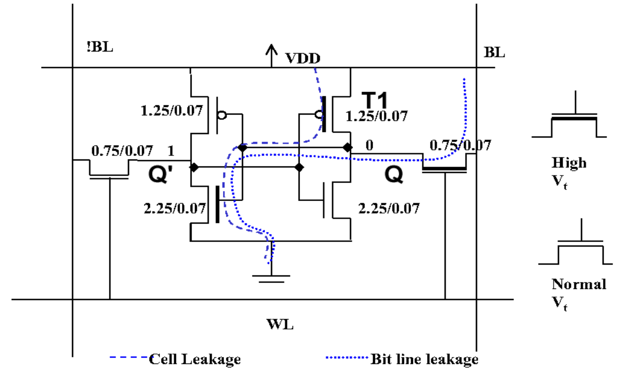


Fig. 1. Asymmetric SRAM cell optimized for leakage of 0 (high V_t = 0.4V, low V_t = 0.2V

TABLE I

NODAL $Q_{critical}$ FOR SRAM DESIGNS

Node	Flip	SRAM fC	ASRAM-0 fC	ASRAM-1 fC
Q=0	0→1	325.482	1225.016	510.79
!Q=1	1→0	15.104	22.31	10.479
Q=1	1→0	15.104	10.479	22.310
!Q=0	0→1	325.482	510.795	1225.016

TABLE II

FIT/MBIT FOR DIFFERENT SRAM DESIGNS

	SRAM	ASRAM-0	ASRAM-1
Storing 0	1000	586.4	1408.6
Storing 1	1000	1408.6	586.4

SRAM. This is due to the change in the drive strength of the high V_t transistors that enables to reduce the leakage in the preferred state. This difference also makes it more difficult to flip the preferred state.

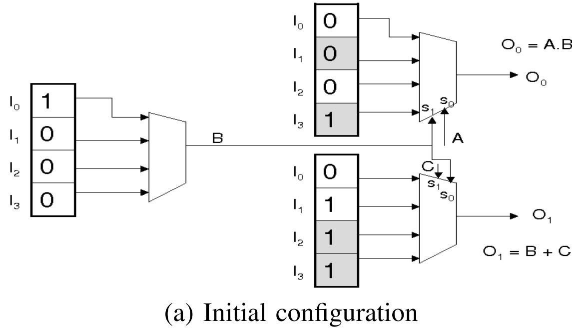
We derive the failure rates of each design using the model presented in [7]. The conventional SRAM's failure is assumed to be 1000 FIT/Mbit, where 1 FIT is 1 failure in 10^9 hours of operation and Q_s is derived from [7] to be 12fC for a 70nm. The SER for various designs are given in Table II. We observe that the ASRAMs storing preferred states are more robust.

IV. Methodology

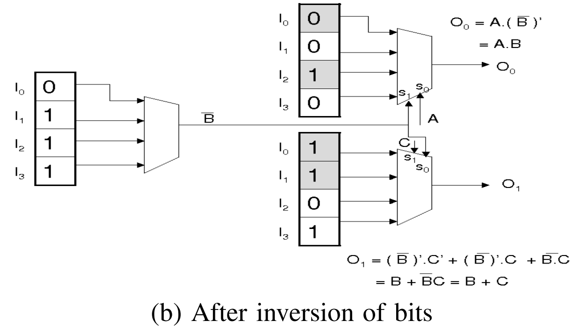
Due to a large number of 0's among the care bits in routing configuration SRAMs, we propose that using an ASRAM-0 cell to implement them will reduce their SER. On the other hand, reducing the SER in LUT SRAM bits is not straightforward because they do not show any particular inclination to value 0 or 1.

In order to reduce the SER in LUT bits, we propose the bit-inversion technique which reduces the number of 1's in LUTs shown in Fig. 4. If an LUT contains more 1's than 0's, then its bits are flipped (to increase number of zeros) and the bits in LUTs that are being driven by this LUT are rearranged (to maintain correct functionality). Note that this technique prohibits inversion of bits in those LUTs which are connected to outputs of the design, either directly, or through a flip-flop.

The list of all LUTs used by a design was extracted from



(a) Initial configuration



(b) After inversion of bits

Fig. 2. Inversion of bits in LUT to maximize zeros

```
boolean internal_LUT(LUT) {
  if ((LUT's output drives a design output) or
   (LUT's output drives a flip-flop driving design output))
     return FALSE;
  return TRUE;
}
invert_bits() {
  used_luts = list of all used LUTs;
  for all LUT ∈ used_luts
      if (internal_LUT(LUT))
          if (numZeros(LUT) < numOnes(LUT)) {
             flip_bits (LUT);
             for all lut ∈ fanout(LUT)
                rearrange_bits(lut); }
}
```

Fig. 3. Algorithm for maximizing 0's in LUT bits

the ASCII description of the implemented design (XDL) which describes only those LUTs that were actually used in the implementation. Therefore, this list could be created in $O(n)$ time, where n is the number of *used* LUTs. Furthermore, the invert_LUT procedure makes one loop over the entire list of used LUTs, which makes it possible to complete the entire inversion of bits in O(n) steps.

V. Experimentation

In order to evaluate our idea of inverting LUT-bits, we selected a set of applications and used the Xilinx Virtex-II FPGA as our target hardware. Table III provides the number of slices used by each application and the target FPGA device used for implementing it.

Xilinx ISE 6.1 tools were used to synthesize, map, place and route the design. Then bitgen was used to generate the configuration bitstream file and the routed design was also converted to an ASCII file in XDL format using xdl tool. The XDL file was used to find which LUTs are used, and the information about partially used LUTs.

We used API's provided by JBits(ver 3.0) to get a detailed information about the implemented design from the configuration bitstream file. We wrote a Java program (bitInvert.java) using JBits API's that took the XDL file and bitstream file as inputs and inverted the LUT bits as described in section IV,

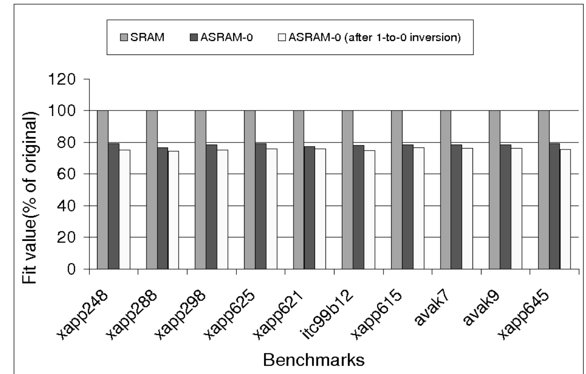


Fig. 4. FIT for SRAM, ASRAM-0 before and after 1-to-0 inversion of LUT bits. Avg. (ASRAM-0): 78.3 Avg. (ASRAM-0 after inversion): 75.5

along with bit shuffling. We did the inversion first to maximize zeros in LUT bits, and then to experiment with ASRAM-1 as configuration memory, we also inverted the LUT bits to maximize ones. This program also estimated the number of care bits in the implemented design and counted the number of zeros and ones in them, as well as the distribution of these between LUT bits and routing configuration bits.

’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

VI. Results

The results of the various simulations performed as per the aforementioned setup are summarized in Table III. One of the observations from the table is that a majority of the care bits are comprised of zeros. This property motivated us to use ASRAM-0 instead of normal SRAM cells to implement configuration memory and therefore minimize the soft error rate in most of the configuration bits. Table III also shows the percentage of zeros in the configuration bits before and after the application of our bit-inversion technique. We also list the FIT values for every implemented design in table III. Note that the FIT values for all designs reduce when ASRAM-0 is used as the configuration memory cell and LUT bits are inverted to maximize LUT 0 bits. The average reduction was found to be 24%. The results reflect an average increase of 27% in the number of zeros in the LUTs after the proposed 1-to-0 bit-inversion of LUT bits. Note that this reflects an average improvement of only 3% in the number of zeros considering

TABLE III

CHARACTERISTICS OF BENCHMARK DESIGNS. NOTE WE CONSIDER ONLY THE USED LUTS IN THE TABLE BELOW. THE UNUSED LUTS CAN BE TRIVIALLY FLIPPED TO EITHER 0'S OR 1'S.

Design	#Slices	Care Bits(CB)	Original implementation			After 1-to-0 inversion			FPGA device
		(%)	0's in CB (%)	0's in LUTs (%)	FIT	0's in CB (%)	0's in LUTs (%)	FIT	
xapp248	95	22.4	75.9	44.5	16.1	79.4	65.2	12.1	xc2v40-cs144
xapp288	54	8.33	77.5	24.4	6	80.9	83	4.5	xc2v40-cs144
xapp298	76	18	75.6	42.7	13	79.6	65.7	9.7	xc2v40-cs144
xapp615	1152	36.9	76.1	40.7	159.3	78.2	65.5	121.9	xc2v250-cs144
xapp621	1387	44.6	76.3	42.5	192.6	79.1	73.7	145.8	xc2v250-cs144
xapp625	221	56	74.9	38.7	40.3	79.6	70	30.5	xc2v40-cs144
xapp645	415	12.43	78.71	43.9	89.5	80.5	62	67.5	xc2v1000-ff896
itc99b12	210	55	76.3	47.2	39.6	80.3	72.3	29.6	xc2v40-cs144
avak7	1312	26.3	75.7	47.9	251.3	78.3	65.5	192	xc2v250-cs144
avak9	2482	43.2	75.8	47.5	586.5	78.5	66.7	447.4	xc2v1000-ff896

all care bits. The reason for such a small improvement is that the proposed bit flipping is only applicable for LUTs, which only comprise around 10-15% of the total care bits, therefore the percentage change in zeros is not a very significant portion of the total care bits. For used LUT bits, note that on one extreme, xapp288 design shows an increase of 58.3% in the number of 0's after bit-inversions while on the other extreme, avak7 shows an improvement of only 17.6%. This large variance in the improvement occurs due to variations in the number of 0's among designs. Xapp288 has only 24.4% of 0's in its LUT bits originally. This enables our bit-inversion technique to flip a large number of 1's in the LUTs to 0's. On the other hand, avak7 has 47.9% of 0's in LUT bits. Due to an almost equal number of 0's and 1's in the design, the bit-inversion technique is ineffective.

Figure 4 captures the effect on FIT values, when implementing the configuration memory using ASRAM-0 cells for all benchmark designs. It also shows the additional reduction in FIT values achieved by inverting the LUT bits to 0. The graph depicts an improvement of the FIT value by around 25% for all the designs. Further improvement of 4-8% was observed when we applied 1-to-0 bit inversion in the LUT memory bits. The use of ASRAMs for the configuration memory results in reduction of leakage power when the ASRAM cell is in its preferred state. Configuration SRAM leakage in the ASRAM-0 implementation was observed to reduce by a factor of 18 without our LUT bit-inversion optimization, reducing the overall FPGA leakage by 35.9%. After incorporating bit-inversion, this number reduced by a factor of 21 as compared to the original SRAM implementation, reducing the overall FPGA leakage by 36.2%.

VII. CONCLUSION AND FUTURE WORK

In this work, we exploited the large number of zeros in an FPGA's configuration SRAM bits to increase the immunity of the device to soft errors. We proposed the use of asymmetric SRAM cells optimized for value '0' in the configuration memory to reduce the FIT value of the FPGA, and demonstrated a reduction of 25% in the FIT value of the FPGA. In order to increase the number of zeros in the LUT memory bits, we proposed a bit-inversion technique that increased the number of zeros in the LUTs by nearly 27%. In addition to increasing the reliability, the ASRAM-0 implementation also reduced the leakage in configuration SRAMs by a factor of 21 after the bit-inversions.

The current gains of our proposed bit-inversion technique is restricted by the way logic is originally mapped onto the LUTs. In future, we plan to work on developing a mapping technique that minimizes the number of zeros (or ones) in LUT memory bits.

VIII. ACKNOWLEDGEMENTS

This work was supported by NSF CAREER Awards 0093082 and 0093085 and a grant from GSRC-PAS Award.

REFERENCES

[1] J. H. Anderson, F. Najm, and T. Tuan. "Active Leakage Power Optimization for FPGAs". In *Proceedings of ACM/SIGDA International Symposium on Field-programmable gate arrays*, 2004.

[2] N. Azizi, F. N. Najm, A. Moshovos. "Low Leakage Asymmetric-cell SRAM". *IEEE Intl. Symposium on Low Power Electronic Devices*, 2002.

[3] Berkeley Predictive Technology Model, Device Group, UC Berkeley. http://www-device.eecs.berkeley.edu/ ptm.

[4] C. Carmichael, M. Caffrey, A. Salazar. "Correcting Single-Event Upsets Through Virtex Partial Configuration" *Xilinx Application Notes, XAPP216 (v1.0)*, June 1, 2000

[5] C. Carmichael. "Triple Modular redundancy design techniques for Virtex FPGAs" *Xilinx Application Notes, XAPP197* , 2001

[6] V. Degalahal, N. Vijaykrishnan and M.J Irwin. "Analyzing Soft Errors in Leakage Optimized SRAM Design". *Proceedings of 16th International Conference on VLSI Design*, 2003.

[7] P. Hazucha and C. Svensson. "Impact of CMOS Technology Scaling on the Atmospheric Neutron Soft Error Rate" *IEEE Transactions on Nuclear Science*, Vol. 47, No. 6, Dec. 2000.

[8] T. Kamik, B. Bloechel, K. Soumyanath, V. De and S. Borkar. "Scaling trends of cosmic rays induced soft errors in static latches beyond 0.18μ". *Digest of Technical Papers, Symposium on VLSI Circuits*, 2001. Pages: 61 -62.

[9] R. Katz, J. Wang, J. McCollum, and B. Cronquist. "The Impact of Software and CAE Tools on SEU in Field Programmable Gate Arrays Programmable Gate Arrays to Space Projects". *IEEE Nuclear Space Radiation Effects Conference*, 1999

[10] P. Samudrala, J. Ramos and Srinivas Katkoori. "Selective Triple Modular Redundancy for SEU Mitigation in FPGAs" *Proc. of MAPLD*, 2003

[11] P. Sundararajan, C. Patterson, C. Carmichael, S. McMillan, and B. Blodget. "Estimation of Single Event Upset Probability Impact of FPGA Designs". *Proc. MAPLD*, 2003

[12] T. Tuan and B. Lai. "Leakage Power Analysis of a 90nm FPGA". In *Custom Integrated Circuits Conference*, 2003.

[13] Xilinx product datasheets. http://www.xilinx.com/literature.