

# An Architecture for Fail-Silent Operation of FPGAs and Configurable SoCs

Lee W. Lerner and Charles E. Stroud  
Dept. of Electrical and Computer Engineering  
Auburn University  
Auburn, AL, USA

*Abstract – We present an architecture for fail-silent operation of Field Programmable Gate Arrays (FPGAs) and configurable System-on-Chip (SoC) implementations. The architecture includes replication of the system function separated by a guard band region that guarantees that no single fault can allow interaction between the replicated system functions. The outputs of the replicated system functions are monitored such that, once an error is detected, the FPGA is shut down through a sequence of operations to prevent the transmission of erroneous data. This architecture can detect single event upsets (SEUs) in the configuration memory of FPGAs and FPGA cores in configurable SoCs for subsequent scrubbing operations. The guard band is also particularly effective in isolating operational regions of an FPGA to help to reduce erroneous operation due to SEUs in the configuration memory.*<sup>1</sup>

**Keywords:** fault detection, fault-tolerance, single event upset, programmable routing

## 1. INTRODUCTION

Most non-fault-tolerant applications begin to operate with potential errors at the time of occurrence of a fault and will continue this erroneous operation until the fault is detected by periodic off-line system-level testing. Fault-tolerant applications, on the other hand, seek to correct continue proper operation in the presence of faults either by masking the fault or by detecting the fault via on-line testing or concurrent fault detection techniques and switching to a replicated fault-free unit [1][2]. Fault-tolerant applications that mask faults include  $N$ -tuple Modular Redundancy (NMR) of which Triple Modular Redundancy (TMR) is a particular case where  $N=3$  [3]. A telephone switching system is a classic example of fault-tolerant applications that rely on concurrent fault detection circuits and redundant units [1]. Although less common, there are applications, referred to as fail-silent, that seek to halt any and all operation immediately upon the occurrence of a fault. While most designs implemented in Field Programmable Gate Arrays (FPGAs) have been of the non-fault-tolerant variety, the implementation of fault-tolerant applications in FPGAs has been the focus of recent research and development [4][5]. However, fail-silent architectural implementations in FPGAs, like their applications, have for the most part been overlooked.

The fault-tolerant operation of FPGAs is of particular interest to overcome the affect of single event upsets (SEUs) in the FPGA configuration memory that can occur in high-radiation environments such as space [6][7]. This is of more concern with shrinking transistor features sizes and reduced supply voltages where 3.3V devices have been observed to have a SEU rate five times higher than 5V devices [7]. In most of these implementations, fault-tolerant architectures, such as TMR, are implemented in the system function to be configured onto the FPGA [3]. However, this requires that particular attention be paid to the physical layout and routing to prevent an SEU for inducing a fault that will affect two of the three replicated modules in the TMR structure [8]. A less robust alternative to TMR is to periodically rewrite the contents of the configuration memory, referred to as scrubbing the memory, to eliminate the effects of any SEUs that may have occurred [9].

We present an architecture for fail-silent operation in FPGAs and configurable System-on-Chip (SoC) devices that exploits the programmability and reconfigurability of the device for both fault isolation and for fail-silent operation. We begin with an overview of the basic idea and philosophy of the architecture in Section 2. This is followed by an overview of the architectures of some commercially available FPGAs and configurable SoCs including the Atmel AT94 Field Programmable System Level Integrated Circuit (FPSLIC) [10] and the Xilinx Virtex series FPGAs [11]. Along with the overview of the architecture of each device, we describe how we have applied the proposed fail-silent architecture to those devices in Sections 3 and 4, respectively. The application of the fail-silent architecture to

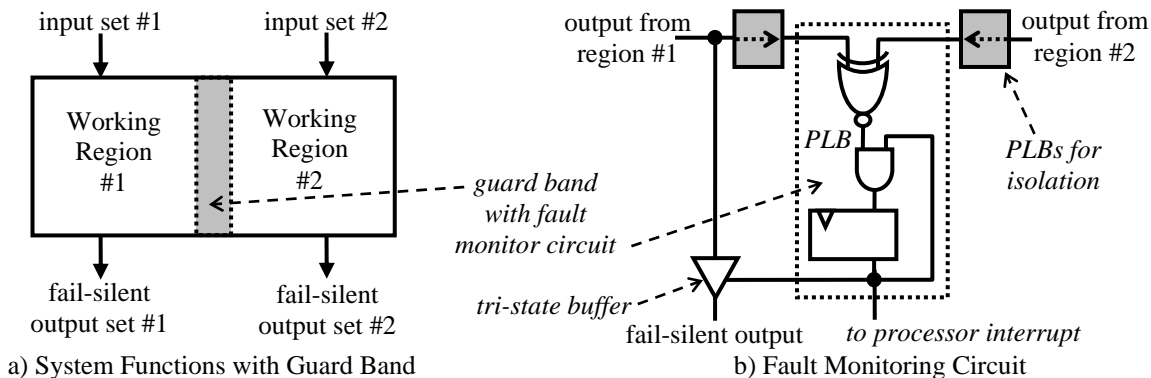
---

<sup>1</sup> This work was sponsored by the National Security Agency under contract H98230-04-C-1177.

the problem of SEUs in the FPGA configuration memory is discussed in Section 5. The paper concludes in Section 5 with a summary of the capabilities and limitations of the overall approach.

## 2. BASIC FAIL-SILENT ARCHITECTURE AND OPERATION

The programmable logic and routing resources in FPGAs and FPGA cores in configurable SoCs also provide the ability to isolate faults once these resources have been tested and found to be fault-free. This can be done by establishing a *guard band* region in the FPGA to separate working system functions. The basic idea behind guard bands is to create a region of isolation in the FPGA such that no single fault in the guard band can allow interaction between two working circuits as illustrated in Figure 1a. As a result, the size of the guard band will be a function of the architecture of the programmable interconnect network in any given FPGA. It should be noted that these faults can occur during system operation as a result of SEUs in the configuration memory [6].



**Figure 1. Fail-Silent Architecture**

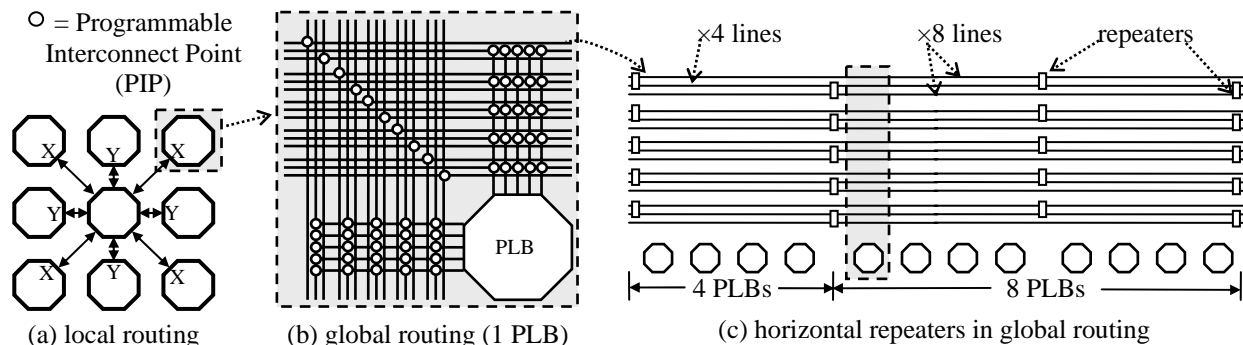
The fail-silent architecture includes a fault monitoring circuit, illustrated in Figure 1b, for each output of the two working regions for the separated circuits. The architecture assumes that two identical sets of inputs come into the device through input buffers associated with each working region such that the two sets of inputs are separated, and isolated, along with their respective working region by the guard band. The outputs of the two working regions represent those signals destined to leave the device. Assuming that only one set of outputs will exit the device, the outputs of one of the working regions drive the data inputs of tri-state output buffers. The outputs of both working regions are passed to the fault monitoring circuit where they are pair-wise compared with any mismatches latched as a logic 0 in the fault monitoring circuit, as shown in Figure 1b, assuming an active-high tri-state enable on the buffer. The output of each latch drives the tri-state control of the output buffer associated with the output monitored by the respective fault monitoring circuit. When a mismatch is encountered, the output buffer is immediately tri-stated on the next clock cycle. As a result of the feedback in the fault monitoring circuit, the output will remain tri-stated until the fault monitoring circuit is reset. An alternate approach, not shown in Figure 1b, is to configure the output buffers as bi-directional buffers with the output of the input portion of the buffer used as the input to the fault monitoring circuit. In this way, faults on the output pins of the device as well as on the signal nets on the printed circuit board can be detected by the fault monitoring circuit to activate the fail-silent mode of operation. In the event of multiple outputs, there can be a fault monitoring circuit for each output with the outputs of the latches for all fault monitoring circuits ANDed together with the single output driving the tri-state controls for all output buffers. This scenario assumes an active high enable on the tri-state buffer; an active low enable would require the dual circuit in Figure 1b with outputs ORed together.

The fault monitoring circuit(s) can be located in the guard band region provided that local routing resources and programmable logic blocks (PLBs) are used to isolate the fault monitoring circuit from the working regions. This PLB isolation is accomplished by programming an identity function in a look-up table (LUT) of the PLB, such that the LUT output is functionally equivalent to the logic value of the input connected to the LUT. In this way, the LUT provides an isolation buffer between the signal net in the working region and the functionally equivalent signal net inside the guard band. Otherwise, locating the fault monitoring circuit in the guard band compromises the integrity of the guard band in terms of its ability to isolate the two working regions if global routing resources are used to connect the output(s) of the working region to the input(s) of the fault monitoring circuit(s). For example, short between two global routing resources could propagate errors into both working regions. This aspect will be illustrated in further detail in subsequent sections.

The fail-silent operation can be extended beyond the tri-stated output or bi-directional buffer by allowing the output of the fault monitoring circuit to drive an interrupt to an embedded processor core to activate a subroutine in the program memory that will dynamically reconfigure the output buffer as an inactive buffer with pull-up or pull-down such that the tri-state control cannot be enabled again. In addition to reconfiguring the output buffers, the interrupt routine can perform additional functions such as powering down the FPGA core (an option in some SoCs), dynamically reconfiguring the FPGA core (effectively erasing the current system configuration such that the entire FPGA goes into an un-programmed state), and/or giving notification that a failure condition has been encountered. The fault monitoring circuits can be replicated with outputs ANDed (assuming an active high enable) to prevent a stuck-at one fault in the latch from keeping the output buffer enabled during a failure condition in one or both of the working regions. The interrupt produced by the fault monitoring circuit (s) can also be used for handling SEUs as will be discussed in Section 5.

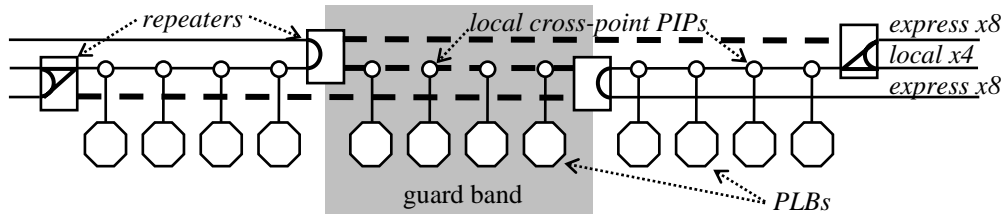
### 3. APPLICATION TO ATMEL AT40K SERIES FPGAS

The Atmel AT94K series SoC consists of an FPGA core, various random access memory (RAM) cores, and an 8-bit Advanced Virtual RISC (AVR) microcontroller core [10]. The FPGA core consists of a symmetrical  $N \times N$  array of PLBs, where  $N=48$  for the AT94K40 device (the largest AT94K series SoC). Each PLB contains two 3-input LUTs, a D-type flip-flop, and other miscellaneous logic resources. Every  $4 \times 4$  array of PLBs has a dedicated 128-bit RAM core that can be configured as a single-port or dual-port RAM. Every PLB has dedicated diagonal (X) and orthogonal (Y) local routing resources to its neighboring PLBs, as shown in Figure 2a. Five sets of vertical and horizontal global routing resources associated with each PLB are shown in Figure 2b and traverse either four PLBs ( $\times 4$  lines referred to as local lines) or eight PLBs ( $\times 8$  lines referred to as express lines) before encountering a bus repeater. Vertical and horizontal bus repeaters are placed at the boundaries of every  $4 \times 4$  array of PLBs (as shown in Figure 2c for the horizontal lines) to prevent signal degradation in lengthy and/or heavily loaded signal nets. Bank clock and set/reset lines are associated with the vertical repeaters running to groups of four PLBs in each column within a repeater boundary. The AVR microcontroller core can write to (but not read from) the FPGA core configuration memory such that the FPGA can be dynamically reconfigured (either fully or partially) by the AVR core during normal system operation [10]. A 32-Kbyte program memory is used to store and execute AVR programs.



**Figure 2. Atmel AT94K Series Configurable SoC Architecture**

The minimum size guard band that prevents a single fault from allowing interaction between two separated circuits is illustrated in Figure 3 and indicated by the shaded section between the two repeaters. This guard band implies that the logic resources in the shaded area and the routing resources shown in bold dashed lines cannot be used by either of the separated system functions. The unusable routing resources include the local  $\times 4$  line between the repeaters as well as the two express  $\times 8$  lines. It should be noted that  $\times 4$  and  $\times 8$  lines along with the repeaters represent the complete set of five global routing resources. It should also be noted that all vertical routing resources associated with the four PLBs shown in the shaded area are also included in the guard band and, as a result, may not be used by the separated working regions. The repeater resources that may be used by the separated circuits are shown in bold lines within the repeaters in Figure 3. These include loopback connections between  $\times 8$  and  $\times 4$  lines in the repeaters at the border of the guard band and the straight-through, loopback, and diagonal  $\times 8$ -to- $\times 4$  connections at the next repeaters out from the guard band. All other routing and repeater resources in the FPGA can be used by the respective separated working regions. With this guard band arrangement, the column-based bank clock and reset to the PLBs in the two working regions are also isolated.



**Figure 3. Atmel Global Routing Architecture and Guard Bands**

Since only one column of  $4 \times 4$  arrays of PLBs is used as the guard band and the AT94K40 is a  $48 \times 48$  array of PLBs, the two separated circuit regions will not be of the same size – one will be 24 PLBs wide and the other will be 20 PLBs wide. The recommended arrangement will be the 20 PLBs wide working region on the left-hand side of the FPGA core and the 24 PLB wide working region on the right-hand side of the array. This will allow an equal number of RAM cores in both regions to function as dual-port RAMs since the right-most column of RAM cores in the FPGA can only be used as single-port RAMs.

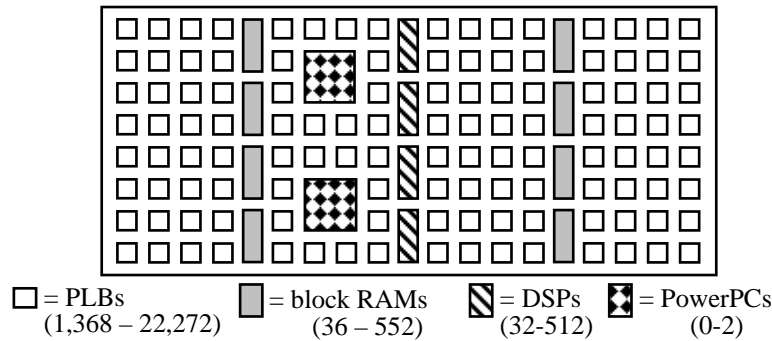
The Atmel Computer-Aided Design (CAD) tool suite capabilities facilitate implementations for this design methodology [11]. Designers have the ability to constrain use to certain regions of the FPGA such that the guard bands can be established with the system function limited to the user defined working region. This is accomplished by synthesizing VHDL or Verilog describing the intended system function as a macro in Mentor Graphic's Leonardo synthesis tool and then importing the macro design into Atmel's Figaro FPGA layout and routing CAD tool. Figaro is then used to place and route the system function macro while restricting the layout and routing to the  $20 \times 48$  working region. The design is then "checked-in" as a macro in the user library to be called up as a component in a hierarchical VHDL description. The same approach can be used for the fault monitoring circuit while restricting them to the guard band with PLBs along the edges used for routing the inputs (outputs from the working regions) to the fault monitoring circuit in the guard band. The top level hierarchical model calls the two system function macros and fault monitoring circuit macros equal to the number of outputs of the intended system function.

We implemented a prototype circuit using this approach where the system function consisted of a Linear Feedback Shift Register (LFSR) with primitive polynomial and the most significant bit of the LFSR used as the primary output of the system function. The two identical LFSR macros were located on the left and right hand sides of the guard band containing a fault monitoring circuit. In this implementation, we used bi-directional buffers with the output of the input portion of the buffer used as input to the fault monitoring circuit. To prevent automated routing in Figaro from using the global horizontal routing resources in the guard band, we extended the fault monitoring circuit macro into the left and right hand working regions to force the use of local direct routing only within the guard band. The output of the fault monitoring circuit supplied the tri-state control on the fail-silent output and also a falling edge interrupt to the embedded AVR processor. An active interrupt initiated the execution of an AVR program that first reconfigured the output buffers to an inactive state (as input buffers) and then cleared the configuration memory by writing the default, un-programmed values. The interrupt was generated whenever a mismatch in the outputs of the two working regions was detected as a result of a fault. We verified proper operation of the fail-silent architecture by injecting faults in the FPGA core from an AVR program that writes to the configuration memory to turn on bits that will emulate faults the logic and routing resource.

#### **4. APPLICATION TO XILINX VIRTEX-4 SERIES FPGAS**

The general architecture of the Virtex-4 FPGA is illustrated in Figure 4 and consists of columns of PLBs, block RAMs, and digital signal processors (DSPs) [11]. Each PLB consists of four slices, each containing two 4-input look-up tables (LUTs) and two flip-flops along with other specialized logic. The LUTs in two of the four slices can also function as small RAMs or shift registers. The block RAMs are 18Kbit dual-port RAMs programmable in a variety of modes of operation including first-in first-out (FIFO) modes. The DSPs include an  $18 \times 18$ -bit signed multiplier and a 48-bit adder/subtractor with registers for accumulator operation. The numbers of columns of PLBs, block RAMs, and DSPs varies, as does the number of rows, with the size and family (LX, SX, or FX) of Virtex-4 FPGA. The ranges for the total number of PLBs, block RAMs, and DSPs is given in the legend in Figure 4. The Virtex-4 FX family includes one to two PowerPC 405 embedded RISC processor cores and all Virtex-4 FPGAs support the synthesis of soft embedded processor cores such as the 32-bit Microblaze processor [11]. The Virtex-4 also provides the ability to perform dynamic partial reconfiguration of the FPGA configuration memory. This can be

done internally by any embedded processor core via an internal configuration access port (ICAP) in the FPGA; this capability is also supported in the Virtex-II and Virtex-II Pro series FPGAs.



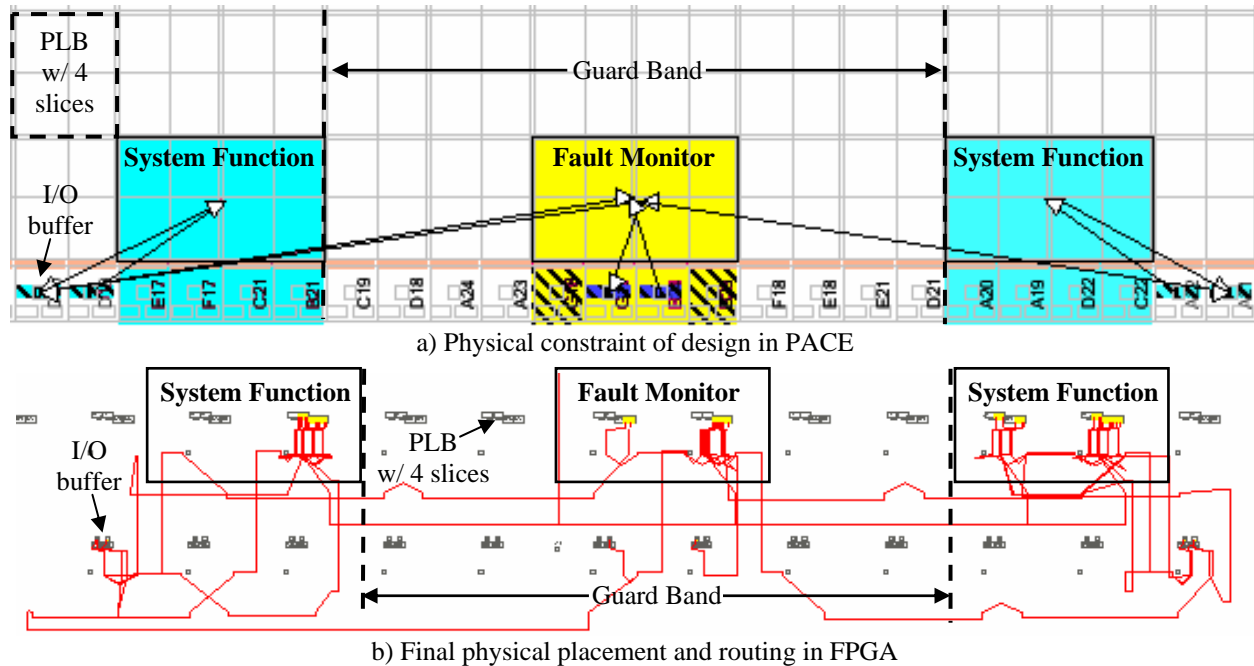
**Figure 4. Basic Virtex 4 Architecture**

The programmable interconnect resources in the Virtex-4 are similar to other Virtex families and consists of local and global routing resources of various length wire segments and programmable interconnect points (PIPs). Four different types of PIPs are found in the routing architecture: break-point PIPs, cross-point PIPs, multiplexer PIPs, and switch-box PIPs. A group of switch-box PIPs is referred to as a switch matrix that provides connectivity between the adjacent and non-adjacent PLBs. Associated with each PLB are a group of *Double* lines ( $\times 2$  lines) that span two PLBs, a group of buffered *Hex* lines ( $\times 6$  lines) that span six PLBs, and *Long* lines that span the width and the length of the PLB array [11]. The *Long* lines can defeat the fault isolation capabilities of the guard band. However, if the *Long* lines can be avoided in terms of system function usage, a guard band can be established that is six PLBs wide or high, depending on whether the guard band runs vertically or horizontally. The size of the guard band, six PLBs in this case, corresponds to the length of the *Hex* lines ( $\times 6$  lines) in the global routing resources. Due to the column orientation of the block RAM and DSP resources, horizontal guard bands are best suited for the Virtex-4 architecture. It should be noted that unlike Figure 4, the Virtex-4 has more rows than columns with an average ratio of 2-to-1, which is also favorable to horizontal guard bands.

The CAD tool suite associated with Xilinx FPGAs also supports capabilities that facilitate the implementation of the fail-silent architecture [13]. Similar to the approach described above for the Atmel device, the system function can be modeled in VHDL or Verilog and synthesized. However, the physical layout can be better controlled by the physical constraints editor (PACE) in the Xilinx CAD tool suite since the hierarchical VHDL or Verilog model can be constrained on a per module basis without the need to place and route each function as a macro, as was the case in Atmel. This can be seen in Figure 5a where the complete hierarchical model is constrained in PACE for the same example as implemented in the Atmel device. The resultant physical placement and routing of the final implementation in the FPGA, taken from FPGA Editor, is illustrated in Figure 5b, where it can be seen that the two system functions, the fault monitoring circuit, and the respective I/O buffers are contained within the constrained areas from PACE. Therefore, implementation of the fail-silent architecture requires fewer steps in the Xilinx CAD tool suite than the previous example of implementing placed and routed macros for each component in the fail-silent architecture for the Atmel device. It should be noted that using PLBs to isolate the signal nets in the working regions from the functionally equivalent signal nets inside the guard band was not included in the example in Figure 5b but should be included in an actual implementation of the architecture. However, the Xilinx CAD tools do support the implementation of placed and routed macros, similar to the Atmel example, which can be used for isolating signal nets between the system function working regions and the guard band. It should also be noted that the figures have been rotated 90 degrees counter-clockwise from the actual implementation which implements a horizontal guard band as we have recommended for Virtex-4 FPGAs.

Unlike the Atmel AT94K series device, the Virtex-4 (as well as Virtex-II) devices do not have dedicated program memory for the embedded processor cores. Instead, programmable logic and routing resources must be used in conjunction with block RAMs in the FPGA to construct a program memory. As a result, additional guard bands (without fault monitoring circuits) should be used to isolate the embedded processor core from the system functions. The layout of the embedded processor core can follow the same procedure as that of the system functions. If the system function requires an embedded processor core, then the larger Virtex-4 FX series (FX40 and higher) devices should be used that contain two PowerPC modules such that one can be included in each system function. If the system function incorporates a soft processor core, such as MicroBlaze or PicoBlaze, then the Virtex-4 LX and SX series

devices can be used. However, we emphasize that the fail-silent architecture can be applied to any FPGA while the size of the guard band will vary as a function of the programmable interconnect architecture.

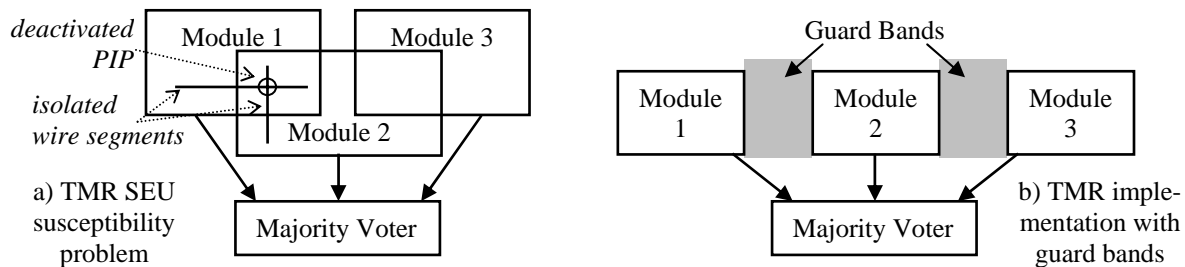


**Figure 5. Xilinx Implementation of Guard Band and Fail-Silent Architecture**

## 5. APPLICATION TO SEUS

Proposed solutions to the problem of SEUs in the configuration memory of FPGAs have included periodic scrubbing of the memory and fault-tolerant approaches such as TMR. Scrubbing the memory essentially erases any SEUs but does not address the problem of erroneous system operation from the time of the SEU to the time of scrubbing. The fail-silent architecture provides a mechanism for detecting the occurrence of an SEU that effects the operation of either system function. Furthermore, the fault monitoring circuit provides an indication that an SEU has occurred which can be used as an interrupt to an internal or external processor to initiate a scrubbing routine to minimize system down time due to the occurrence of an SEU. With the indication of an SEU that effects the operation of the system function, it may be possible to reduce or eliminate periodic scrubbing of the memory.

TMR, on the other hand, can ensure error-free system operation in the presence of one or more SEUs but will eventually fail when many SEUs have occurred. Furthermore, it has been observed that the physical layout of the TMR implementation must be considered to prevent one SEU from disabling two of the replicated modules. An example of this situation is illustrated in Figure 6a where a TMR circuit is synthesized in an FPGA with no constraints in terms of physical layout and routing. Since the synthesis CAD tools do not recognize the redundancy of the system function, the logic used to construct the replicated modules will be interspersed, represented by the overlapping module boundaries in Figure 6a. As a result of the intermingling of the logic of the modules, it is possible that two signal nets in two different replicated modules will be isolated only by a single deactivated PIP, illustrated by the two lines denoting the two wire segments (where the horizontal wire segment is associated with Module 1 and the vertical wire segment is associated with Module 2) and the circle denoting the deactivated PIP. An SEU on the configuration bit that controls the PIP would activate the PIP and short the two wire segments. This single bridging fault could cause both replicated modules to fail, emulating multiple fault behavior. Experimental results obtained from injecting faults in the configuration memory of an FPGA implementing various TMR adders and multipliers indicated that between 9.0% and 13.2% of single faults injected resulted in failure of the TMR circuit as a result of this type of multiple fault behavior [14]. One proposed solution to this physical layout problem has been a specialized place and route algorithm specifically for the implementation of TMR designs in FPGAs. In this place and route algorithm, the replicated modules are allowed to intersperse but the critical signal nets are separated by at least two PIPs [8]. While this approach has shown to be effective in preventing TMR failures that result from SEUs, the software is not readily available or supported for general use with commercially available FPGAs.



**Figure 6. TMR Implementations in FPGAs**

The guard bands in the fail-silent architecture provide the proper type of isolation for TMR since no single fault resulting from an SEU can result in interaction between the separated system functions. Furthermore, guard bands can be applied to TMR implementations to isolate the replicated system function, as illustrated in Figure 6b, in the same manner as in the fail-silent architecture, without the need for specialized place and route algorithms. As a result, effective TMR implementations for SEUs can be implemented with the available CAD tool suites for the FPGAs, as demonstrated by the implementation examples of the fail-silent architecture in the previous two sections.

## 6. SUMMARY AND CONCLUSIONS

Between the realm of non-fault-tolerant and fault-tolerant applications lies a set of applications where it is desired to halt operation rather than continue erroneous operation due to the presence of a fault. We have presented a fail-silent architecture for the implementation of these applications in FPGAs. A small guard band region of the FPGA is used to isolate two working regions that contain functionally equivalent system functions whose outputs are monitored by fault monitoring circuits contained within the guard band region. When mismatches are observed and latched in the fault monitoring circuit as a result of a fault in one of the two working regions, the outputs of the device are force into the tri-state mode with pull-up or pull-down features programmed in the I/O buffer forcing a fail-safe condition. While fault-tolerant approaches typically require an area overhead of at least three times that of the non-fault-tolerant circuit, the fail-silent architecture requires slightly over two times that of the non-fault-tolerant circuit. Furthermore, the inclusion of the fault monitoring circuits within the guard band region helps to minimize the fail-silent architecture area overhead. When applied to the problem of SEU, the fail-silent architecture does not provide fault-tolerance as in the case of TMR, but it does provide an immediate indication of the occurrence of an SEU to initiate scrubbing of the configuration memory.

## REFERENCES

- [1] B. Johnson. "Design and Analysis of Fault Tolerant Digital Systems." Addison-Wesley, 1988.
- [2] P. Lala. "Fault Tolerant and Fault Testable Hardware Design." Prentice Hall International, 1985.
- [3] C. Carmichael. "Triple Module Redundancy Design Techniques for Virtex FPGAs, Application Note 197." Xilinx, Inc., 2001.
- [4] M. Abramovici, C. Stroud and J. Emmert. "On-Line Built-In Self-Test and Diagnosis of FPGA Logic Resources." IEEE Trans. on VLSI Systems, 12(12): 1284-1294, 2004.
- [5] P. Samudrala, J. Ramos and S. Katkoori. "Selective Triple Modular Redundancy Based Single-Event Upset Tolerant Synthesis for FPGAs." IEEE Trans. on Nuclear Science, 51(5): 713-724, 2004.
- [6] J. Barth, C. Dyer and E. Stassinopoulos. "Space, Atmospheric and Terrestrial Radiation Environments." IEEE Trans. on Nuclear Science, 50(3): 466-482, 2003.
- [7] M. Ohlsson, P. Dyreklev, K. Johansson and P. Alfke. "Neutron Single Even Upsets in SRAM-Based FPGAs." Proc. IEEE Nuclear and Space Radiation Effects Conf., 177-180, 1998.
- [8] M. Sonza Reorda, L. Sterpone and M. Violante. "Multiple Errors Provoked by SEUs in the FPGA Configuration Memory: A Possible Solution." IEEE European Test Symp., 136-141, 2005.
- [9] C. Carmichael, M. Caffrey and A. Salazar. "Correcting Single-Event Upset Through Virtex Partial Reconfiguration, Application Note 216." Xilinx, Inc., 2000.
- [10] "AT94K Series Field Programmable System Level Integrated Circuit." Atmel Corp., 2001.
- [11] "Integrated Development System Technical Reference and Release Notes Version 6.0." Atmel Corp., 1998.
- [12] "Virtex 4 User Guide UG070 (v1.4)." Xilinx, Inc., 2005.
- [13] "Xilinx ISE 7 Software Manuals." Xilinx, Inc., 2005.
- [14] P. Bernardi, M. Sonza Reorda, L. Sterpone, M. Violante. "On the Evaluation of SEU Sensitiveness in SRAM-based FPGAs." Proc. IEEE International On-Line Testing Symp., 115-120, 2004.