Dynamically Reconfigurable Softcore Processor of Fault-Tolerance Technique by Picoblaze Core

[1]R. Sivaranjani ,[2]G. Vinoth
PG student, Assistant Professor
Department of Electronics and Communication Engineering
DMI College of engineering, Chennai-602013, India

***Abstract* - In this paper, A new enhanced Lockstep scheme built using a pair of Micro blaze cores is proposed and implemented on Xilinx Virtex-6 FPGA.Lockstep scheme allows to detect and eliminate its internal temporary configuration upsets without interrupting normal functioning. Faults are detected and eliminated using a configuration engine built on the basis of the picoblaze core which, to avoid a single point of failure, is implemented as fault tolerant using triple modular redundancy. The new enhanced lockstep scheme requires significantly shorter error recovery time compared to conventional lockstep scheme and uses significantly smaller number of slices compared to known TMR based design. A soft-core processor can recover from configuration upsets through partial reconfiguration combined with roll-forward recovery. SEUs affecting logic which are significantly less likely than those affecting configuration are handled by check pointing and rollback.**

Keywords— Error recovery, fault injection, fault-tolerance, FPGA, lockstep, reconfigurable system, single-event upset (SEU),softcore processor

1. INTRODUCTION

In past a lockstep scheme using two hardcore Power PC processors embedded in Xilinx Virtex-6 Pro FPGA, which could be considered for application in soft core processors as well. However, this scheme is rather a handshake scheme than a lockstep one because, to perform consistency checks, the processors execute the same program but not simultaneously. As a result, the overall time overhead is relatively large (besides context saving and restoring, if needed), because of the sequential execution of two identical tasks on two different processors, which might be prohibitively long in some real time applications. Moreover, using check pointing and rollback for context recovery results in an extra time overhead. Modern FPGAs, besides customary reconfigurable resources, offer the designers the possibilities of implementing programmable processors having features of Commercial Off-The-Shelf (COTS) components .Soft-core processors use reconfigurable resources, so their number that can be actually implemented depends on the device size only. Xilinx FPGAs use SRAM-based technologies which are known to be very susceptible to radiation and electromagnetic noise .The major effects caused by them are known as Single-Event Upsets (SEUs) or soft errors, because only some logic state(s) of memory element(s) are changed but the circuit/device itself is not permanently damaged.

In FPGAs, SEUs may directly corrupt computation results or induce changes to configuration memory; the latter can cause changes in the functionality and performance of the device. Due to their flexibility, FPGAs are attractive for mission-critical embedded applications, but their reliability could be insufficient unless some fault-tolerance techniques capable of mitigating soft errors are used. These techniques should allow for online error detection or/and correction during system operation, very fast fault location, quick recovery from temporary failures, and fast permanent fault repair through reconfiguration. Here, we are interested in designing and implementing a fault-tolerant (FT) soft-core processor using Virtex-6 FPGA. Therefore, one option to implement larger number of lockstep modules in a single FPGA is to employ soft-core processors that can use all available reconfigurable resources of the device.

2. PRELIMINARIES

In this section, we will present some basic FPGA features that are essential to support fault tolerance in the designs proposed here, the fault and error model of SRAM FPGAs,and the survey of the fault-tolerance techniques commonly used in FPGA-based systems.

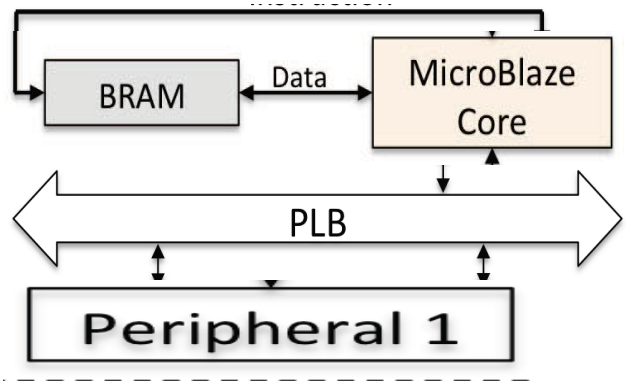

Fig. 1. MicroBlaze softcore processor structure

2.1 MicroBlaze Softcore Processor Structure

Besides hardcore embedded processors like PowerPC, that are hardwired in the die, FPGAs can also implement softcore processors offered by manufacturers, like MicroBlaze, PicoBlaze, and Nios, as well as those proposed by the open source community,like LEON3 and LEON4 . Softcore processors are Intellectual Property (IP) blocks written in hardware description languages (HDL) like VHDL or Verilog, to be implemented using reconfigurable resources of FPGAs. Fig. 1 shows a typical MicroBlaze processor system that consists of a 32-bit MicroBlaze core, the program memory BRAM with its data and instruction buses, and the Processor Local Bus (PLB) the central bus of the MicroBlaze core with some peripherals connected to it.Many aspects of the MicroBlaze can be user configured: cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces can be customized. The area-optimized version of MicroBlaze, which uses a 3-stage pipeline, sacrifices clock-frequency for reduced logic-area. The performance-optimized version expands the execution-pipeline to 5-stages, allowing top speeds of 210 MHz (*on Virtex-6 FPGA family.) Also, key processor instructions which are rarely used but more expensive to implement in hardware can be selectively added/removed (i.e. multiply, divide, and floating-point ops.) This customization enables a developer to make the appropriate design tradeoffs for a specific set of host hardware and application software requirements.

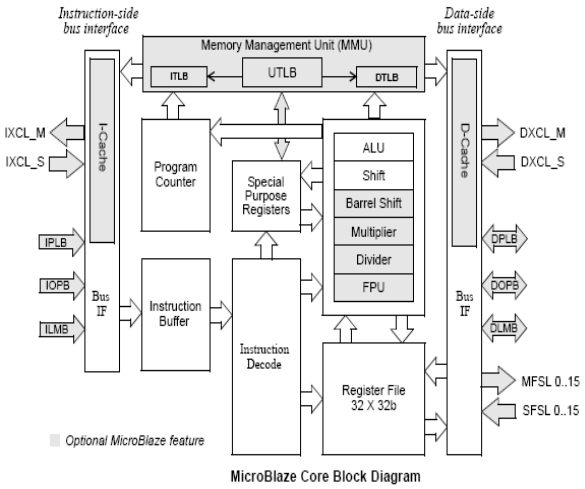

Fig.2 MicroBlaze Processor Block Diagram

3. BASE SYSTEM BUILDER

The Base System Builder (BSB) automates basic hardware and software platform configuration tasks common to most processor designs. If you are targeting one of the supported embedded processor development boards available from Xilinx, or from one of our partners, the BSB lets you pick from the peripherals available on that board, automatically match the FPGA pin out to the board, and create a completed platform and test application ready to download and run on the board. If you are developing a design for a custom board, the BSB lets you select and interconnect one of the available processor cores (MicroBlaz) or PowerPC, depending on your selected target FPGA device) with a variety of compatible, commonly used peripheral cores from the library. This gives you a hardware platform to use as a starting point from which you can add more processors and peripherals if needed, including custom peripherals, using the tools provided in XPS. For a PLB-based design, you can create a single or dual processor system, which can use either MicroBlaze or PowerPC (405 or 440) processors. In the Board and System Selection page of the Base System Builder, select the type of system you want to create. Review the system information that displays when you select a system. The BSB creates a single or dual processor system. You can configure the processor, peripheral set, and some major configuration parameters for the peripherals in the next window of the wizard.

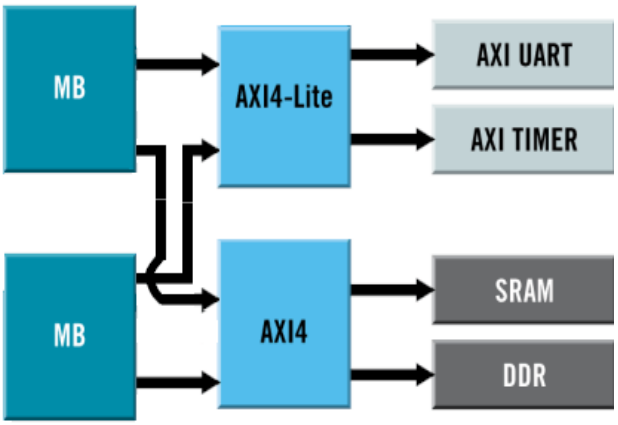

Fig 3 Processor selection

4. .FAULT AND ERROR UPSETS

We assume a commonly used fault model of SRAM FPGAs,which includes temporary faults (SEUs) affecting configuration memory or user memory implementing, e.g., flip flops and registers. Single SEUs will be of our primary concern here, because they are the most likely to occur. Nevertheless, unlike most authors, we will also consider permanent faults affecting configuration memory.

SEUs in configuration memory may result in modifications of the functionalities of the application design the FPGA implements. For a given design, all configuration memory bits can be classified as being sensitive (whose upset induces errors) and non-sensitive. This is because among numerous configuration memory bits only some are actually utilized by the user's design, hence SEUs affecting the configuration bits that are not utilized by a specific design will not affect the behavior of that design. Although of temporary nature, SEUs may have permanent effects until the device is reconfigured, e.g., by readback or scrubbing [21]. In addition to configuration sensitivity, the sensitive bits can be further categorized into two following categories :

. Nonpersistent bits are those configuration bits which, when upset, may induce nonpersistent functional errors which disappear once the device is reconfigured, so that the design can return to normal operation. The nonpersistent bits generally involve purely combinational circuitry of the design.

. Persistent bits are those configuration bits which, when upset, induce persistent functional errors, which do not disappear even after the device is reconfigured. The persistent bits generally involve any part of the design that contains the sequential circuitry or BRAM. The frame-based reconfiguration followed by the internal reset could eliminate persistent errors but, unfortunately, it cannot deal with corrupted data written to the BRAM. One feasible solution for the latter problem is the following:

- to reconfigure the whole module (module-based reconfiguration) to eliminate configuration errors and re establish the start up state of the BRAM; then
- to perform an internal reset of the registers and flip flops to the initial correct state; and finally,
- to perform the context recovery.

4.2 Error Handling Techniques in FPGAs

Configuration data of SRAM FPGAs, containing millions of configuration bits, are particularly vulnerable to SEUs. They can be protected against errors by various standard means provided by FPGAs' manufacturers, like configuration readback, configuration scrubbing, and error detecting or correcting codes [8], [21]. These techniques can be applied continuously in the background of a user design. FRAME_ ECC is the Virtex-5 and -6 primitive using single-error correcting/ double-error-detecting (SEC/DED) Hamming error correcting code (ECC) which during configuration readback allows to correct single errors and/or detect double errors in the configuration frame data.

4.3 Error Recovery Techniques

Once an error is detected, the next step is error recovery, i.e., the process of removing errors from the system and bringing it back to the error-free state. The processor context is the set of information needed to define uniquely the state of the processor at a given moment. It could include the states of the processor registers, the cache, the memory, etc.

Saving and restoring all relevant values is essential for effective processor context switching and error recovery. Several error recovery schemes have been proposed ,which can be classified as backward and forward.The backward error recovery techniques, which are based on time redundancy, rely on saving the correct processor context and restoring it once the errors are removed, so that the processor could resume correct functioning at the last saved point (checkpoint). Among various context recovery techniques differing in the moment of saving and restoring the context, the most often used is rollback recovery using checkpoints. The major drawback of rollback recovery using checkpoints is the time overhead (regular saving of the processor context with checkpointing frequency growing with the error rate, computations lost from the last checkpoint followed by restoring of the processor context). Fig. 5 shows the scheme of checkpointing and rollback applied for basic lockstep scheme recovery. Because it is not possible to identify the faulty processor without extra diagnosing support, the error recovery in FPGA designs can be achieved through reconfiguration of both processor cores which, however, can be time consuming.

The latter problems are alleviated in roll-forward error recovery which does not need regular context saving. When an error occurs, it is corrected by copying the correct

state of the processor from a fault-free mirror processor, provided that there are means to identify it. Because a lockstep scheme using roll forward rather than rollback seems to offer better performance without significant increase of hardware resources, we will consider it in the designs proposed here.

5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new architecture of a fault-tolerant reconfigurable system which can be implemented on any SRAM-based FPGA with integrated softcore processors. An Enhanced Lockstep scheme built using a pair of MicroBlaze cores was proposed and implemented on Xilinx Virtex-5 FPGA. Unlike the basic lockstep scheme, ours allows to identify the faulty core using a Configuration Engine which allows to recover from single-event upsets through partial reconfiguration combined with roll-forward recovery technique. As a result, the problem of fault latency is alleviated, because faults are detected immediately, once they cause an error. The Configuration Engine was built using PicoBlaze cores and, to avoid a single point of failure, was implemented as fault-tolerant using triple modular redundancy (TMR)

Future work would include developing the methodology which would facilitate the creation of tiling configurations using Xilinx Design Language (XDL) representation of the circuit accompanied by RapidSmith taking into account the sensitivity and persistence of errors caused by faults of the configuration bits inside the permanently defected zone. We would like also to consider fault injection in user's logic and BRAM to validate the significant advantages of our system in dealing with SEUs directly affecting logic.

6. REFERENCES

[1]. Xilinx, Inc., "Single-Event Upset Mitigation Selection Guide,"- Appl. Note XAPP987 (v1.0), http://www.xilinx.com/support/documentation/application_notes/xapp987.pdf, Mar. 2008.

[2]. Xilinx,Inc.,"Virtex-6 FPGA configuration user guide"UG360(v3.6)April18,2013.www.xilinx.com/support/documentation/user_guide.

[3]. Xilinx,Inc., "MicroBlaze Processor Reference Guide" Embedded Development Kit EDK 13.2, UG081 (v13.2).

[4]. D.K. Pradhan and N.H. Vaidya, "Roll-Forward Checkpointing Scheme: A Novel Fault-Tolerant Architecture," IEEE Trans. Computers, vol. 43, no. 10, pp. 1163-1174, Oct. 1994.

[5] H.-M. Pham, S. Pillement, and D. Demigny, "A Fault-Tolerant Layer for Dynamically Reconfigurable Multi-Processor Systemon- Chip," Proc. Int'l Conf. ReConFigurable Computing and FPGAs, pp. 284-289, Dec. 2009.