2011 International Conference on System Science, Engineering Design and Manufacturing Informatization

Implementation of a Reconfigurable Computing System for Space Applications

Yimao Cai, Yuanfu Zhao, and Lidong Lan
Beijing Microelectronics Technology Institute
Beijing, China
caiyimao101@163.com, zhaoyf@mxtronics.com.cn, lanld@mxtronics.com.cn

***Abstract*—high reliability is usually the most important requirement in space applications. There is no hardware repair for them. Long mission times and harsh environment are a challenge for electronic circuits, and particular error mitigation techniques have to be implemented in order to be able to cope with the expected error effects. Always, Triple Modular Redundancy (TMR) is relied mostly to prevent SEUs from causing functional errors. And in recent decades, SRAM-based FPGAs are used a lot in space for its high performance and flexibility. However, use of FPGAs in space applications can also be challenging. They are too susceptible to transient faults, such as SEUs. Reconfiguration provides more possibilities for SRAM-based FPGAs into space. Reconfigurable technology solves the disadvantage that the TMR modules cannot heal the error themselves. Additionally, reconfiguration is an improvement in terms of resource utilization and costs. In this paper, TMR and reconfigurable computing are combined to level up the reliability of the space applications. The paper proposed a reconfigurable computing architecture based on a TMR system. A new kind of reconfigurable architecture using SoP (system-on-a-package) technology is presented. Also, a whole space application will be presented.**

Keywords- Reconfiguration; TMR; Fault Tolerance; Space Application

I. INTRODUCTION

As the simplicity and high reliability of TMR, it is widely used for space applications. Although the TMR can improve the reliability of the system, due to the implementation of additional modules, it consumes more hardware resources, power and affects the pace of the work. In addition, the TMR does not have the ability to repair the error itself. All these limit the use of the traditional TMR technology. Dynamic partial reconfiguration enhances space applications with re-programmable hardware and at run-time adaptive functionality. Dynamic partial reconfiguration permits a limited, predefined portion of an FPGA to be reconfigured while the remainder of the device continues to operate. This is especially valuable where devices operate in a critical environment, like space applications, and cannot be disrupted while subsystems are redefined [1]. Dynamic partial reconfiguration can used to level up system reliability by repair the system by itself. This paper will combine the reconfigurable technology and the TMR technology together and the system can get higher reliability. Reconfigurable technology solves the disadvantage that the TMR module cannot heal the error itself.

In the next section, the paper describes fault tolerance concepts. Then we introduce the reconfigurable technology and the system architecture including the TMR system and the detail of the configuration platform in section four. Next section introduces the design flow of reconfiguration. Section six presents our process of the dynamic partial reconfiguration technique with our reconfigurable system. We compare the two systems in section seven. Final section summarizes the text and gives a short outlook to the future work.

II. FAULT TOLERANCE CONCEPTS

A Single Event Upset (SEU) is caused by charged particles losing energy by ionizing the medium which they pass, leaving behind a wake of electron-hole pairs. If this happens in a flip-flop or memory cell, the particle can deposit enough charge to cause the flip-flop or memory cell to change state, corrupting the data being stored. SEUs are still a major concern in SRAM-based FPGAs, even in radiation-hardened devices[2]. In order to solve failures generated by SEUs, a technique called scrubbing has been developed. This technique reconfigures the matrix in order to replace the changed configuration bits with the correct ones. There are different strategies regarding scrubbing. It can be continuous scrubbing [3] or only when it is needed [4].

Triple modular redundancy is among the most fault tolerant technique employed in FPGAs [5]-[7]. In the coarse redundancy approach when a fault is detected in a column, the whole column is marked as faulty and it is then replaced by a spare CLB(configurable logic block) [8], [9]. Scrubbing technique combined with conventional TMR techniques to mitigate SEU effects has been proposed in [10]. In [11], X. Iturbe, etc, have proposed a combination of the TMR and dynamic reconfiguration to mitigate SEU effects and permanent hardware errors. They implement three modules in a FPGA, and we implement a TMR system with three individual same architecture.

III. DYNAMIC PARTIAL RECONFIGURATION CONCEPTS

Dynamic partial reconfiguration (DPR) is a modification of configuration data within a FPGA, while the rest of it is still operational. Partial reconfiguration denotes to modify a limited portion of an FPGA. The use of the dynamic partial reconfiguration is a very active research area. The "Hardware Plugins" concept makes sense with the new huge a partially reconfigurable capable FPGAs [12], [13]. In this approach the system is strongly unitized. It is distinguished

978-1-4577-0246-4/11/$26.00 ©2011 IEEE

between one entire static part and one or more Partial Reconfigurable Modules (PRMs). The PRMs are bounded in Partial Reconfigurable Areas (PRAs) within an FPGA. PRAs can be reconfigured and comprise at least two PRMs with different functionality. The static area is the area within an FPGA except the PRAs and remains unchanged. The partial reconfiguration process of PRAs comprises all logic resources and routing interconnects. If the PRM routing changes, either the communication to the static area or between PRAs could be interrupted. Therefore, persistent routing communication wires between static and PRAs are necessary [2]. Xilinx provides these pre-placed and pre-routed resources with bus macros. DPR is achieved using module-based and difference-based approach. This paper is based on module-based approach.

IV. SYSTEM ARCHITECTURE

A. System Architecture

The architecture of the TMR system is depicted in Fig.1. It represents the implementation of a generic system that mainly includes: three TMR modules and the TMR voter. The control and data signals from each module are voted against each other by a TMR voter. The function of a voter is to decide the real output of a redundant system. It uses the classic majority voting scheme. Thus, taking into account that the presented platform is focused on triple redundancy systems, if the number of '1's in the repeated inputs is equal to or bigger than 2 the output is '1', otherwise the output is '0'.

A communications network is required to support high speed system data flows. We use RS232 to change data between modules.

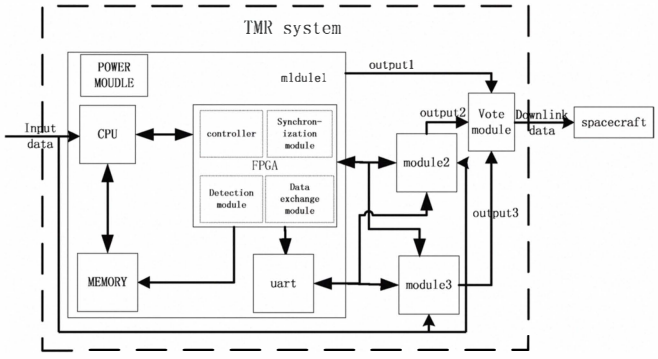

Figure 1. The architecture of the TMR system

The three modules have the same structure including a processor, FPGA, memories and communication portion. The base processor is 32-bit RISC processor based on the SPARC V8 architecture. The main features of the processor core used in our system are a five-stage pipeline, 16KB data cache and 32KB instruction cache, five 24-bit timers, support 1553B bus, 4 A/D converters, watch dog timer, interrupt controller, 32 parallel I/O interface, and 11 interrupt lines organized in two priority interrupt levels. The processor implements both the controller and the scheduler of the given system implementation. Memories are used to store all the partial reconfiguration bitstream data information and store the software programme. There are three memories, 8M Flash, 1M SRAM and 16M SDRAM. Control logic and peripheral IP cores are put into the Xilinx FPGA. Hardware units mapped into the FPGA can be interfaced to the system bus through a peripheral bus. We use general-purpose I/O of the processor to configure the FPGA. Download of the FPGA bitstream is performed at the beginning form the FLASH. To accelerate communication between the configurable hardware and software tasks running on the processor, four interrupt channels can be driven by logic mapped into the FPGA. To allow validation of the FPGA configuration, the bitstream may be read back by the processor. Architecture of one TMR module is as shown in Fig.2.

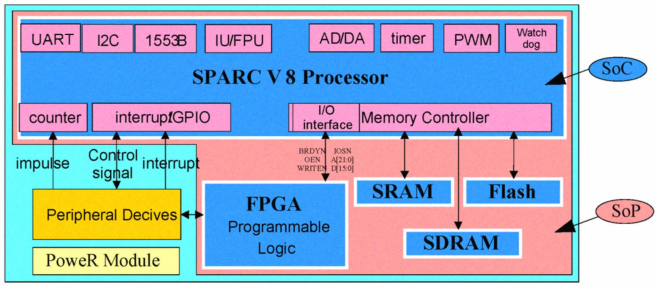

Figure 2. Architecture of one TMR module

V. THE CONFIGURATION UNITS

A. Diving

Now we begin the process of partial dynamic reconfiguration in FPGA. As three modules are the same, we only take one module for example. The whole system is divided into two modules: the fixed module and the reconfigurable module. Fixed module including the peripheral bus and some IP cores will not be reconfigured at run-time. The reconfiguration modules which are placed in the reconfigurable area are used for different algorithms hardware accelerator or IP cores for communication. FPGA resource partitioning and module placement is as shown in Fig.3.

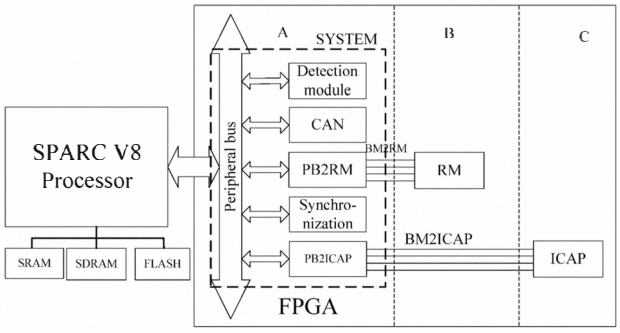

Figure 3. FPGA resource partitioning and module placement

In Fig.3, area A, C is fixed area that cannot be reconfigured; area B is a reconfigurable area where we can place reconfig-urable modules. All connections across different areas connect each other through the presences of the Bus Macro, through which the reconfigurable modules connect to the static part of the design and other modules. BM2RM across the region A and B is the interface between the reconfigurable module and the bus.BM2ICAP across the whole region is the connecting of A and B.

B. Module Designing

The design methodology is based on modular design concept. This feature allows designs to split into portions that are independently synthesized, coded, placed, routed, and mapped. There are three main modules in whole system:

System module: System module, area A as shown in Fig. 3, is the core of the whole system, including connect bus detection module, data exchange module, synchronization module and required peripherals to run the system. For the address bus and data bus are bi-directional and the bus macro can only communicate one-way, the interface between peripheral bus and bus macro is needed for modules that need to write back or read back. PB2RM and BM2ICAP are this kind of interface as shown in Fig. 3.

ICAP module: ICAP module can finish self-reconfiguration of the FPGA. According to device requirements, this module can be only placed in the lower right corner of FPGA, so a separated area C is created. Communication between ICAP and the system module follows the peripheral bus communication protocol. ICAP module is used to read/write a configuration from/to the BRAM to/from a specific reconfigurable module. When a new reconfigurable module has been mapped and it starts its computation, the ICAP informs the processor that the reconfiguration action ended with success. After that the controller enables all the communications interrupted by the reconfiguration.

Reconfigurable modules (RM): These are reconfigurable modules in the system. The implantation files of these modules should be checked in FPGA complier to ensure that each module is limited in designated area. The bus macro is also under the constraint file into the specified location and the length of the bus macro is always occupied the same in the files generated by each module. We realize a CAN receiver in RM module.

At last, we create all the bitstreams needed to implement the system description onto an FPGA through the dynamic embedded reconfiguration and verify the function of the system. This flow facilitates the modular design concept and the process is easy to follow.

VI. DETAIL OF RECONFIGURATION

The following paragraphs will describe our approach to implement self-repairing. A summary of the Self-repairing sequence is as follow:

A. Error detection and localization

Error Detection and Localization is a crucial part of system. It is supervisory layer that detects faults and raises dedicated countermeasures. Error localization is done by halting faulty unit and turning down all the related out buffers. Next step is to start reconfiguration processes followed by synchronization process.

Transient fault detects will be accumulated to detect permanent faults. Other than notifying the application, nothing else will be done to correct transient faults. We can rely on higher layers to deal with the effects of transient faults. Permanent faults will be dealt with by reprogramming the FPGA with an alternate pre-compiled spatial variant of the same application [14].

B. Reconfiguration

After an error has been detected and its location is known, the repair process can be started. Let us first assume that a soft error has occurred. In this case it is sufficient to reconfigure the FPGA. We use partial reconfiguration reconfigures the defective part of the circuit and therefore can be performed very fast. For hard errors, i.e. hardware defects, simply reloading the bit-stream will not be sufficient - a new place&route process has to be started. The circuit has to be moved away from the defective area and mapped to previously unused resources in the FPGA [15]. In contrast to synchronous logic, where the critical path might become too long with a different routing, delay insensitive circuits can seamlessly adapt to this new situation. An overview on reconfiguration techniques can be found in [16].

C. Recovery

After reconfiguration, the newly placed device is completely reset and has to be synchronized with rest of redundant devices. There are at least four synchronization strategies which are feasible for implementation in reconfigurable avionics test platform. The details are introduced in [17].

After the recovery, the error module is absolutely synchronized as other modules. With dynamic FPGA reconfiguration incorporated in system, single faults can be continuously repaired, rendering system reliability almost constant in time.

VII. TEST AND RESULTS

The TMR system is implemented, the synchronization error is in 30ns and the transfer rate to other module can reach 8Mbps. Each TMR module consists of several elements include a processor, memories and FPGA. We use SoP technology to package them together. The implementation of one module is shown in Fig. 4. The size of the system is 31mm×31mm and it is small enough for embedded system and space application.

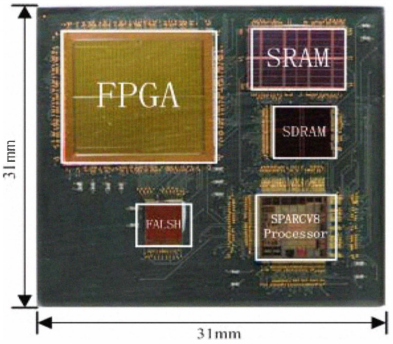

Figure 4. Implementation of the system

The following table (Table 1) reports the results of the tests performed with the original TMR system and with the approach based on the reconfigurable architecture. We can know that the new system has a higher performance of 150MIPS(Million Instructions Per Second). The power consumption reduces 86.7% and the weight reduces 90%. It is a great improvement for the space application. The chip used in the system is much less than before, so our system reliability is much higher.

TABLE I. COMPARISON OF TWO SYSTEMS

	Original TMR system	***Reconfigurable TMR system***
Computing performance	ARM7 60MIPS	SPARC V8 150MIPS 50FLOPS
Weight	7.6kg	750g
Power consumption	27W	3.6W
Peripheral device	76 chips	13 chips

VIII. CONCLUSION AND FURTHER WORK

In this contribution, we propose the design and implementation of a reliable and flexible approach for space application successfully. The proposed system is a valuable step in the process of integrating dynamic reconfiguration in TMR system. In this context, another contribution has been presented in this paper: a system with fault tolerance and self-repairing capabilities that takes advantage from the dynamic reconfiguration capability of the FPGA devices. We have successfully demonstrated the applicability of the dynamic reconfiguration to a real process control task. In addition, through system integration technology, we make the system low consumption of power, volume and mass, and adequate reliability.

As we show in the experiment, this architecture cut weight and power consumption a lot compared to other TMR systems. Future work in this field includes: sufficient fast dynamic reconfiguration and the examination of hardware and software concepts in whole system. Also, we have to achieve more experimental results.

ACKNOWLEDGMENT

The presented work is part of the project Design of the Reconfigurable Chip, funded by Beijing Microelectronics Technology Institute (BMTI). The authors would like to thank Jianyong Wang and the owners of BMTI for their support.

REFERENCES

[1] Björn Osterloh, Harald Michalik, Sandi Alexander Habinc, Björn Fiethe, "Dynamic Partial Reconfiguration in Space Applications," in 2009 NASA/ESA Conference on Adaptive Hardware and Systems, pp.336-343.

[2] Björn Osterloh, Harald Michalik, Björn Fiethe, Frank Bubenhagen, "Architecture Verification of the SoCWire NoC Approach for Safe Dynamic Partial Reconfiguration in Space Applications," in 2010 NASA/ESA Conference on Adaptive Hardware and Systems, 2010.pp.1-8.

[3] C. Charmichael, "Correcting single-event-upsets throught Virtex partial reconfiguration." XAPP216, 2000, Xilinx Inc.

[4] K. Chapman and L. Jones, "SEU strategies for Virtex-5 devices," XAPP864, 2009, Xilinx Inc.

[5] E. Stott, P. Sedcole, and P. Cheung, "Fault tolerant methods for reliability in FPGAs," in Proc. Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on, 2008, pp. 415–420.

[6] S. D'Angelo, C. Metra, S. Pastore, A. Pogutz, and G. Sechi, "Fault tolerant voting mechanism and recovery scheme for TMR FPGA-based systems," in Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 1998, pp. 233–240.

[7] R. DeMara and K. Zhang, "Autonomous FPGA fault handling through competitive runtime reconfiguration," in Proc. Evolvable Hardware, 2005. NASA/DoD Conference on, 2005, pp. 109–116.

[8] A. Antola, V. Piuri, and M. Sami, "On-line diagnosis and reconfiguration of FPGA systems," in Proc. of the The First IEEE International Workshop on Electronic Design, Test and Applications (DELTA'02). IEEE Computer Society Washington, DC, USA, 2002, pp. 291–296.

[9] A. Doumar, S. Kaneko, and H. Ito, "Defect and fault tolerance FPGAs by shifting the configuration data," in Proc. of the 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems. IEEE Computer Society Washington, DC, USA, 1999, pp. 377–385.

[10] [C. Bolchini, A. Miele, and M. D. Santambrogio, "TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs," in 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2007, pp. 87–95.

[11] X. Iturbe, M. Azkarate, I. Martinez, J. Perez, and A. Astarloa, "A Novel SEU, MBU and SHE Handling Strategy for XILINX VIRTEX-4 FPGAs," in 19th International Conference on Field Programmable Logic and Applications (FPL09), Sep. 2009, pp. 569–573.

[12] E. L. Horta, J. W. Lockwood, D. E. Taylor, and D. Parlour, "Dynamic Hardware Plugins in an FPGA with Partial Run-time Reconfiguration," in Proceedings of the Design Automation Conference (DAC'02), New Orleans, LA, Jun. 2002, pp. 343–348.

[13] K. Danne, C. Bobda, and H. Kalte, "Run-Time Exchange of Mechatronic Controllers Using Partial Hardware Reconfiguration," Lecture Notes in Computer Science, vol. 2778, pp. 272–281, 2003.

[14] Clayton Okino, Clement Lee, Andrew Gray, Payman Arabshahi, "Space-Based Autonomous Reconfigurable Protocol Chip," in Aerospace Conference,2005, pp. 1494-1499.

[15] Thomas Panhofer, Martin Delvai. "SELF-HEALING CIRCUITS FOR SPACE-APPLICATIONS," in Field Programmable Logic and Applications 2007, FPL 2007, pp.505-506.

[16] J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for fpgas," ACM Trans. Des. Autom. Electron. Syst., vol. 11, no. 2, pp. 501-533, 2006.

[17] Rafal Graczyk, Marcin Stolarski, Patrick Cormery, "Exploratory Study about the Use of New Reconfigurable FPGAs in Space," in 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), pp.220-226.