

Reconfigurable Computers in Space: Problems, Solutions and Future Directions

Neil W. Bergmann, and Anwar S. Dawood

Cooperative Research Centre for Satellite Systems

Queensland University of Technology

GPO Box 2434 Brisbane 4001, AUSTRALIA

n.bergmann@qut.edu.au, a.dawood@qut.edu.au, <http://www.crcss.qut.edu.au/>

Abstract

This paper explores the use of SRAM-based FPGAs as a system component in future generation spacecraft. It describes the major advantages which reconfigurable computing can provide in reduced space mission cost, and describes the problems which are encountered when conventional SRAM-based FPGAs are used in space applications. Solutions are presented which will enable successful use of conventional SRAM-based FPGAs in space applications. Additional features, which could be incorporated in FPGA chips to make them more suitable for use in space, are also described. Finally, we describe an experimental Adaptive Instrument Module, based on reconfigurable computing technology, which will be flown on the FedSat low earth orbit satellite to test out our ideas.

I. INTRODUCTION

FPGAs are now an established ASIC implementation technology for terrestrially based commercial and military systems. Since 1992, annual scientific conferences in the USA (FCCM, RAW) and Europe (FPL) have attracted many researchers who have demonstrated significant potential performance improvements by the use of reconfigurable computers, i.e. computing machines where FPGAs are combined with conventional processors to give systems which have programmable hardware and software resources.

There is now significant interest in the application of reconfigurable computing techniques to spacecraft, and the ground systems to support them, as evidenced by a new workshop in this area called Military and Aerospace Applications of Programmable Logic Devices (MAPLD). Groups at Los Alamos National Laboratory [1] and NASA's Goddard Space Flight Centre [2] are actively engaged in the design of reconfigurable computing systems for the ground-based processing of satellite data. NASA's Office of Earth Science [3] has described how reconfigurable computers could form an important enabling technology for future generation spacecraft. NASA have embarked on a program to develop a radiation hard, SRAM-based FPGA by about year 2000 [4]. Xilinx, a major producer of SRAM-based FPGAs, have announced the availability of radiation-tolerant, space-qualified versions of their FPGAs [5]. Actel, the major supplier of

one-time programmable anti-fuse FPGAs for space applications have announced plans to release commercial SRAM-based FPGAs in 1999, with space-qualified versions to follow.

At this stage, then, we note a number of convergent trends:

- Terrestrial reconfigurable computers based on SRAM-FPGAs are an active research topic, with many successful problem solutions demonstrated, and under development in the commercial and military domains.
- Space-science researchers are aware of the benefits of these reconfigurable computing technologies, and are beginning to use them for ground-based processing of spacecraft data.
- Because of the potential for application of SRAM-based FPGAs, a number of manufacturers and research organisations have announced the development of space-qualified devices for use in future generation spacecraft. The first such devices have become available from late 1998.

This paper first describes the major advantages which reconfigurable computing can provide in reduced space mission cost. These are different to the major advantages normally claimed for terrestrial systems.

Next, the paper describes the problems which are encountered when conventional SRAM-based FPGAs are used in space applications.

Thirdly, solutions are presented for these problems which we believe will enable successful use of conventional SRAM-based FPGAs in space applications.

Next, additional features are described which could be incorporated in SRAM-based FPGA chips to make them more suitable for use in space.

Finally, we describe an experimental Adaptive Instrument Module (AIM), based on reconfigurable computing technology, which will be flown on the FedSat low earth orbit satellite, due for launch in 2000.

II. ADVANTAGES OF FPGAS FOR SPACE APPLICATIONS

Unmanned scientific spacecraft electronics represent a unique class of computing system. During operation, the systems are physically remote from their operators, and all control of the spacecraft and telemetry data returned by the spacecraft must be transmitted over a wireless radio link. Furthermore, this radio link is low bandwidth, may be unavailable for extended periods (eg. when a satellite is out of line of sight of its ground station), and often has high error rates.

Furthermore, the remoteness of the system precludes any external repair or upgrading of the system. Often the system is exposed to high levels of ionising radiation, which is damaging to spacecraft electronics. Extensive radiation shielding and redundant circuitry may be precluded by the high cost/kilogram of launching the spacecraft.

Often, only a small number of satellites of a particular design are ever built. Hence Non Recurring Engineering (NRE) costs are high on a per spacecraft basis. The small market size also means specialised electronic components are expensive, and often not state-of-the-art. Very small power budgets on some satellites (maybe as little as 20W total) are also problematic.

If we consider remote-sensing satellites, for example, then until recently the data processing systems of many of these satellites consisted of little more than a sensor, solid-state data recorder and a communications channel to relay this data to ground.

Considerably better satellite operation can be achieved with the use of "smart" sensing instruments, which include some high-performance dedicated data processing resources attached to the sensor. Data can be processed on-board the satellite to extract meaningful data parameters, to compress the data for more efficient storage and downloading, and to filter the data to select the most useful data to store and download.

We contend that such a smart instrument module can be further enhanced by the use of FPGA technology to provide programmable hardware and software resources which can be reprogrammed in flight. We call this smart instrument an Adaptive Instrument Module (AIM).

III. ADVANTAGES OF AN AIM

The adaptability and flexibility of an AIM, which incorporates FPGA technology provides significant benefits at several levels.

Firstly, one can design a generic AIM data processor, with a programmable hardware interface that can be configured for a large number of different sensors to be used on different missions. Physical design costs of the AIM are shared between many instruments and missions.

Use of a generic design shares many NRE costs among several missions.

Secondly, implementation of the AIM means that physical and electrical testing of the system (functional correctness, vibration testing, thermal testing, etc.) can be achieved before the final design of the programmable hardware configuration is complete. This concurrent engineering reduces mission lead-times.

Thirdly, the ability to reconfigure the AIM in flight means that many software and hardware design errors can be repaired while the instrument controller is in flight. This allows reduced mission cost by minimising mission risk, especially for missions with very short lead-times.

Fourthly, the ability to reconfigure the AIM in flight allows the function of the AIM to be adapted to changing mission conditions. For example, ground-based analysis of data received during a mission might lead to development of better data compression algorithms, which can be uploaded during flight. Instrument operation parameters can be adapted to changing mission requirements. In-flight reconfigurability thus provides a significant improvement in spacecraft efficiency.

Fifthly, reconfigurable hardware can be reprogrammed to replace the function of faulty circuit components which might otherwise cause the spacecraft to fail. If some part of an FPGA fails, then circuitry can be redesigned to avoid the faulty part. The FPGA thus reduces mission risk and potentially extends mission lifetime by providing generic operational redundancy. This redundancy is achieved without large additional mission cost, since it does not involve additional on-board circuitry.

Finally, the use of an FPGA to provide circuitry for high-speed data interfacing to different sensors allows use of lower-cost, lower-power-consumption 8-bit or 16-bit microcontrollers rather than more expensive, high-power-consumption 32-bit RISC processors to execute general system software. In many cases, we believe significant power savings can be made.

IV. PROBLEMS WITH FPGAS IN SPACE

One-time programmable, anti-fuse FPGAs have been widely used in spacecraft for many years. However, for reconfigurable computing applications in space, we need to use SRAM or EEPROM technologies. We particularly consider SRAM-based FPGAs here, but many of the same comments apply to EEPROM FPGAs.

Space electronics are subjected to high mechanical and acoustic vibration during launch, microgravity, near vacuum, and a wide temperature range for operation. In most cases, appropriate packaging technology enables industrial or military temperature range chips to successfully operate under these conditions.

The biggest problem with space electronics is the high level of ionising radiation to which circuitry is subjected, with two major effects. Firstly, ionising radiation striking MOS transistor structures causes soft-errors in circuit operation. Generated carriers can discharge stored charge on floating gates, or cause temporary increases in drain-source conductance. Such soft-errors are most commonly manifested as bit-flips in memory storage cells. For space-qualified SRAM FPGAs in low-earth orbit, these bit-flips occur at the rate of a few per chip per day [6].

Secondly, accumulated crystal damage to the circuit eventually causes permanent failure of the device by rendering part of the circuitry inoperative.

These two characteristics mean that we need special techniques to detect and repair configuration and data-bit errors, and also detect hard-errors in FPGA cells, and redesign circuitry around these.

Other problems with FPGAs occur when we attempt to reconfigure the system in-flight. For circuits such as Xilinx 4000 series FPGAs, programming is achieved by downloading a complete configuration bit stream into the device. It is not practical to supply this configuration bit-stream from ground control each time it is required, given the length of the bit-stream (approx 800,000 bits for a Xilinx 4036) and the low bit-rate communications. Special software is required on-board to store and manage the various configurations which might be needed in-flight for different tasks, and to check these stored configurations for possible radiation-induced errors. Only a limited number of configurations can be stored – many instrument controllers may have only a few tens or hundreds of kbytes of RAM available. Most efficient use of this memory requires techniques which are quite different to those for conventional reconfigurable computers.

V. ENABLING THE USE OF FPGAs IN SPACE

We now look at possible approaches to overcome the problems with using FPGAs in spacecraft as outlined in the previous section, so that we can enjoy the potential benefits of this technology as described in section II.

In this section we look particularly at enabling the use of the space-qualified Xilinx XQR4000 series of SRAM-based FPGAs, since these form the basis of our experimental Adaptive Instrument Module.

Functionally, the XQR4000 series of FPGAs are identical to conventional Xilinx 4000 series FPGAs. The chips are, however, built on a special process using an epitaxial substrate to reduce susceptibility to ionising radiation, and packaged appropriately for space use.

A. Soft Errors

Firstly, we examine possible methods for detecting radiation-induced soft-errors in configuration and data memory cells.

Xilinx FPGAs provide the capability to read-back the current configuration bitstream without interrupting circuit operation. This bitstream can then be examined to determine if any errors have occurred.

Some researchers suggest a triple-redundancy scheme for detecting soft-errors [6]. Three identical FPGAs are operated in parallel, with triple voting circuits used to combine the outputs of the FPGAs. The configuration data bit-streams are read-back simultaneously from the three FPGAs and compared. If a single-bit error is detected in one FPGA, then the configuration data can be reloaded.

This technique has the advantage that it can compare both configuration memory bits (whose correct values are known) and data memory bits in the programmable flip-flops (whose values change during execution). Additionally, the circuit can continue operation after the soft-error is detected. In most cases, circuit operation needs to be interrupted to reload the faulty FPGA, since there is no general way to ensure that the data memory bits in the faulty FPGA can be resynchronised with the continually changing memory bits in the correct FPGAs unless their operation is frozen temporarily, at a convenient point in the execution.

The triple-redundancy technique, however, is obviously costly in terms of extra circuitry, requiring a 3-4 times overhead.

We propose a less costly approach, which will be satisfactory in many cases, where continuous circuit operation is not essential.

A single FPGA is used, and its configuration bitstream is readback at regular intervals. The readback values of data memory cells are not used, since their correct values are not known. Either the readback configuration bitstream is compared with the correct configuration bitstream, or alternatively a cyclic redundancy check is made and compared with a correct value. In either case, if an error is detected, then circuit operation can be halted, any output data since the last successful check marked as suspect, and the configuration reloaded.

With this technique, user data bits are not checked, but these are typically less than 1% of the total memory bits in the FPGA for circuits which use only flip-flops and not user-RAM structures.

Some refinements can be made to further improve this algorithm. If two copies of every data memory bit are included in the programmable circuitry with identical inputs, then the readback value of these bits can be compared, and an error detected. Xilinx X4000 series FPGAs have two flip-flops per LUT – this technique would obviously reduce the density of the programmable circuitry, but would improve error detection slightly. Note

that this technique would not detect errors once they had propagated past the site of the original bit-flip.

For a given FPGA configuration, many of the configuration bits do not directly affect the input-output behaviour of the circuit, such as configuration bits for unused LUTs or wiring channels. For a given application, it would be possible to identify those bits where a random bit-flip would not adversely affect circuit behaviour, and these could be masked out of any detection scheme. We do not have good figures for the percentage of “don’t care” bits for a given configuration, but we suspect that it may often exceed 50% of the configuration bit-stream. If so, adoption of this technique could reduce the need to reconfigure the FPGA by half.

An alternative technique for detecting errors does not use the readback of the configuration bit-stream, but rather uses additional redundancy in the programmable circuitry design. There is a good body of knowledge on how to design redundant self-checking circuits [7]. Typically such circuits code data bits as a complementary pair of signals, A and A' , and logic gates are designed with the properties that any single-stuck-at fault results in an illegal output pair encoding, and any illegal input pair encoding to a gate results in an illegal output pair encoding. Hence a circuit fault can be detected by observing overall system outputs for illegal encoding schemes.

A similar approach can be adopted with FPGA circuits, but here the fault model is a bit-flip of a configuration bit for LUT function or communication wiring. This is a new area of research which to our knowledge has not previously been investigated since this “bit-flip” phenomenon (and hence its use as a fault model) is restricted to high-radiation environments. Previous work by other researchers suggests that there is an increase in gate count of 2-5 times for a given function by using these techniques [7].

B. Hard Errors

Having looked at techniques for detecting and fixing soft-errors, we now turn to hard-errors, where some part of the FPGA is rendered inoperative.

There are clearly some errors which will render the FPGA immediately inoperative, such as a fault on an output pin driver or in the configuration-loading circuitry, or an unacceptable rise in leakage current. For other hard-faults, it will be possible to reconfigure the FPGA to work around the fault. This requires the location of the fault to be identified, reported to the local processor, and transmitted to the ground station. The faulty cell has to be notified to the FPGA CAD system during revised placement and routing process, and the new configuration uploaded. The complexity of current FPGA place and route software probably precludes autonomous circuit repair in the form of revised place and route on board the satellite. Limited repairability may be possible by the use

of redundant circuitry, and a simple means to switch out faulty circuit components, but again there will be approximately a 2 times cost in circuit density. Triple redundancy voting circuit structures will allow continued operation in the presence of some hard faults, without the need to reconfigure the circuit, but at higher circuit cost.

C. Configuration Management

The next class of problems to be addressed involves the efficient storage and transmission of new FPGA configurations.

Given the limited communications capacity between the spacecraft and ground, and the limited storage capacity on-board the spacecraft, it is inefficient to transmit and store configurations as raw bitstreams. We suspect that significant savings can be made by using data compression techniques. Configurations could be coded as the difference between an “exemplar” configuration, and the desired configuration, with this difference file compressed using techniques such as entropy coding, predictive coding or run-length coding. Work is planned to determine which of these coding schemes is most efficient. Additionally, we need to add some CRC checking to ensure that stored configurations have not been corrupted. We suspect that the existing check-bits, which Xilinx adds to their configuration bit streams will be sufficient for this purpose.

VI. USEFUL MODIFICATIONS TO FPGAS

Some of the problems addressed in the previous section result from the use of FPGAs designed for terrestrial use but deployed in a high radiation environment. If use of such devices is successful, then it is possible that FPGA chip architectures may be revised to produce FPGA variations with particular enhancements for space use.

The most obvious of these problems is automatic error detection and correction for configuration memory. Memory systems for space computers typically include additional error detection and correction bits, which are used to detect bit flips. Since the contents of such a memory are only significant when they are read, it is enough to check the current memory word being read with a single error detection and correction circuit. For an FPGA, all configuration bits are used all the time to determine circuit function, and so all bits should ideally be checked in parallel. In this model, each group of 32 bits or so would have an additional few check bits plus circuitry to detect, and perhaps correct bit errors. Note that in the case of user data memory, writing to any one bit also requires updating of the check bits. An alternative solution would be internal circuitry to stream the entire configuration bit stream (and perhaps memory data stream also) through a central EDAC unit at regular intervals.

A few FPGAs already allow partial dynamic reconfigurability, where the function of part of the circuit can be changed while the rest of the circuit continues normal operation. Combined with the ability to readback circuit configuration data, such facilities greatly reduce the time to repair soft-errors, since only the relevant configuration bits need to be reloaded. They also facilitate “differential” circuit configuration where a standard configuration is overwritten only for those areas where a new configuration differs. This also allows small “patch code” to be uploaded to change an existing configuration to account for faulty areas of circuitry.

Other potential enhancements include additional “self-test” circuitry to detect and locate permanently faulty circuit elements. Structures like FPGA-controlled redundant LUTs and wiring channels would ease the task of self-repair by allowing use of the same configuration bit-stream but automatically relocating programmable structures.

Such enhancements are still a long way in the future. First we need to confirm the benefits of using FPGAs in space, and also gain experience of how severe problems like those described in section IV are, and how successful our proposed solutions will be. To do this we plan to launch an experimental reconfigurable computing system on-board a low-earth orbit satellite, FedSat, in the year 2000. The next few sections describe our experiment in detail.

VII. AN EXPERIMENTAL SATELLITE PAYLOAD

Reconfiguration of FPGAs while the spacecraft is in flight represents a new scientific challenge. A generic board, which contains SRAM based FPGA and the necessary control circuitry, is being built as a standalone payload to be incorporated with the FedSat LEO due for launch in 2000 [13]. The aims of the experiment are summarised as follows:

- Determine practical techniques for uploading configuration files of different sizes from ground while the spacecraft is in flight.
- Determine and optimise techniques for managing the reconfiguration process of SRAM based FPGA while the spacecraft is in the operational phase.
- Investigate the practicality of running new applications on-board the spacecraft by utilising the FPGA adaptability for changing mission requirements.
- Evaluate FPGA tolerance for radiation induced upset errors in the form of bit-flips during the reconfiguration process and operation mode.
- Develop techniques for detecting configuration errors and monitoring configuration integrity of the SRAM based FPGA.

- Develop techniques for recovering procedures from radiation induced upset errors.
- Determine best techniques for modifying the algorithms implemented on the FPGA during operation for optimum performance.
- Investigate options for spacecraft self-initiated dynamic reconfiguration of SRAM based FPGA for specific applications depending on real time measurements and on-board processing results.
- Evaluate performance improvement of a satellite system using reconfigurable computing technology for on-board services such as data filtering, data compression, and real time image processing.

The payload consists of a single card with the following major components:

- A microprocessor to provide local control and monitoring for the activities assigned to the AIM card. It coordinates and ensures an efficient use of the FPGA chip.
- A Field Programmable Gate Array (FPGA) chip to provide the platform for the reconfigurable computing tasks.
- A Programmable Read Only Memory (PROM) for storing the operating system of the above microprocessor/microcontroller.
- A non-volatile Electrically Erasable Programmable Read Only Memory (EEPROM) for storing configuration files. At least (1-3) configuration files will be stored in this memory before launching. Other newly created configuration files can be uplinked and stored in this memory at the operational phase. Stored configuration files can be serially downloaded into the FPGA on demand.
- A Static Random Access Memory (SRAM) for storing commands, data, and as a working area (scratch pad) for the FPGA and the local microprocessor during operation.
- A Memory Controller or Multiplexer to control the joint access of the microprocessor and the FPGA to the AIM card memory.
- A standard serial interface for communication with the platform Data Handling System.

More details of the AIM developed jointly with Johns Hopkins University - Applied Physics Lab are given in an accompanying paper [14].

VIII. CONCLUSIONS

Reconfigurable computing technology using SRAM based FPGAs seems very promising technology to be used on future generation of satellite systems for space applications. This technology offers advantages such as

reduced hardware physical design time and cost, flexibility to adapt changing mission requirements through in-flight reconfiguration, ability to provide operational redundancy by replacing faulty circuit components, lower mass and volume budget through hardware resource sharing, and optimised power consumption. There are some expected problems in using this technology in space due to the high level of ionising radiation to which these devices are exposed. We discussed methods to deal with the most critical aspect of susceptibility of the SRAM based FPGAs to radiation induced circuit upset in the form of configuration bit-flips, and proposed detection and correction solutions. We also presented an experimental Adaptive Instrument Module, which will be flown on the FedSat low earth orbit satellite, to test out the practicalities of using reconfigurable computing hardware devices in space environment and evaluate improvements in satellite performance.

ACKNOWLEDGMENTS

The support of the Australian Commonwealth Government and the Queensland State Government, through the Cooperative Research Centre for Satellite Systems, is gratefully acknowledged.

REFERENCES

- [1] S. H. Robinson, M. P. Caffrey, M. E. Durham, "Reconfigurable Computer Array: the Bridge between High Speed Sensors and Low Speed Computing", Proceedings of 8th International Workshop on Field-Programmable Logic and Applications, FPL '98, Estonia, September 1998.
- [2] M. Figueiredo, K. Winiecki, T. Graessle, U. Patel, "Study and Utilisation of Adaptive Computing in Space Applications", NASA GSFC, MAPLD Conference, , September 1998.
- [3] D. Andrucyk, "Achieving the Earth Science Vision", NASA GSFC, MAPLD Conference, September 1998.
- [4] J. McCabe, "Radiation Hard Reconfigurable Field Programmable Array", NASA GSFC, MAPLD Conference, September 1998.
- [5] H. Bogrow, "Field Programmable Gate Arrays Technology", Xilinx Inc., MAPLD Conference, September 1998.
- [6] P. Alfke, R. Padovani, "Radiation Tolerance of High Density FPGAs", MAPLD Conference, September 1998.
- [7] S. M. Kia, S. Parameswaran, "Synchronous TSC/CD Error Indicator for self checking systems", Pacific Rim International Symposium on Fault Tolerant Computing, 1993.
- [8] J. B. Brown, S. J. Gardner, A. Wicks, L. Boland and E. C. Graham, "The FedSat Spacecraft Design", Space Innovations Limited (UK) and CRCSS (Australia), 49th International Astronautical Congress, Melbourne, Australia, October 1998.
- [9] The Xilinx QPRO XQR4000XL Radiation Hardened Field Programmable Gate Arrays Product Specification, Xilinx Inc., October 1998.
- [10] P. Sutton and N. Bergmann, "Custom Computing for Satellite Applications", 49th International Astronautical Congress, Melbourne, Australia, October 1998.
- [11] C. Underwood, V. Asenek, M. Day, and M. Sweeting, "Reliability of COTS Microprocessors On-Board SSTL Micro Satellites In the LEO Space Environment", Surrey Space Centre, 49th International Astronautical Congress, Melbourne, Australia, October 1998.
- [12] N. W. Bergmann, J. C. Mudge, and L. R. Cirroco, "FPGA based Custom Computers", Australian Conference on Microelectronics, Gold Coast, October 1993.
- [13] J. Kingwell, <http://www.crcss.csiro.au/fedsat2.htm>: "AUSTRALIA'S FedSat MICRO-SATELLITE", 1997.
- [14] Richard Conde, Charles Dumont, Ann G. Darrin, Phil Luers, Steve Jurczyk, Neil Bergmann and Anwar Dawood., "Adaptive Instrument Module through Programmable Logic", MAPLD99 Conference, Laurel Maryland, September, 1999