

**RECONFIGURABLE SINGLE-CHIP ON-BOARD COMPUTER
FOR A SMALL SATELLITE**

Daixun Zheng
Dr. Tanya Vladimirova
Hans Tiggeler
Prof. Martin Sweeting

Surrey Space Centre
Surrey Satellite Technology Limited
University of Surrey, **Guildford, Surrey, GU2 7XH, UK**
Tel: +44 1483 879278 Fax: +44 1483 876021
Daixun.zheng@ee.surrey.ac.uk, T.Vladimirova@ee.surrey.ac.uk, H.Tiggeler@ee.surrey.ac.uk
www.sstl.co.uk

Abstract

Small Satellite electronic components are generally unavailable for physical upgrade or repair after launch. Run-time Reconfigurable (RTR) computing technology of SRAM-based Field Programmable Gate Arrays (FPGAs) allows to overcome this problem - application of RTR can facilitate new hardware circuits to be uploaded via a radio link without an interruption of service in the FPGAs. In this project these advances in microelectronics are used to produce a key enabling technology for a new generation of low-cost commercial, scientific and military small satellites. This paper introduces a reconfigurable single-chip on-board computer (RSC-OBC) and proposes two schemes for its on-board reconfiguration. Design issues related to reliable operation of the RSC-OBC in space are addressed.

Introduction

Originally proposed in 1963, Reconfigurable Computing (RC) could not be implemented at the time because of technology limitations. While some impressive research applications have been developed, RC technology is still a relatively new field of study for space applications. Space environment is different from terrestrial systems in that incident radiation can cause bit flips in memory elements and ionisation failure in semiconductors. This kind of hardware faults cannot be debugged and repaired, requiring high-reliability manufacture, assembly and operating techniques.

Nowadays driving force and technological base of RC are reprogrammable logic chips with gate densities exceeding millions of gates and capable of supporting Run-Time-Reconfiguration (RTR). The use of RTR in space will allow to modify on-board hardware by replacing faulty/outdated designs at different stages of a mission. Some example applications are: rectification of design faults, improvement of processing algorithms, alteration of system functionality in response to changed mission requirements, change of hardware configurations to reduce weight and power characteristics, etc. Furthermore, FPGA circuitry damaged by Single-Event Upsets (SEUs) can be reprogrammed and recovered through partial RTR.

The aim of this paper is to present the results of a feasibility study on a reconfigurable single-chip on-board computer (RSC-OBC) based on a programmable logic array. The main objective of the project is to propose a reliable partial RTR methodology for a small satellite single-chip on-board computer targeted at a XILINX Virtex FPGA. This work forms a part of the Surrey Space Centre (SSC) long-term research programme codenamed ChipSat aiming to miniaturise further the small satellite platform.

The paper is structured as follows. The next section details the system specification of a single-chip OBC. After that a description of the design flow is given. It is followed by two sections. The first one discusses a basic scheme for on-board RTR. The second one proposes a client-server scheme for on-board RTR, whereby

the RSC-OBC can be partially run-time reconfigured in a remote way. The last section concludes the paper.

Design of a Single-Chip OBC

A System-on-a-Chip implementation of a simplified version of an on-board computer (OBC) is in a process of development at the Surrey Space Centre using a XILINX Virtex XCV800 prototyping board. The block-diagram of the SoC is shown in Figure 1. The decision on the components and structure of this RSC-OBC has been made as a result of translating the structure and functionality of a reference OBC, developed by SSTL, OBC386¹, into an equivalent organisation consisting of intellectual property (IP) cores.² The IP cores of the RSC-OBC are listed in Table 1, which gives information about the equivalent parts in the reference design.

The SoC consists of a SPARC V8 microprocessor core “LEON” connected to a ROM bootstrap loader and a memory error-detection-and-

correction (EDAC) unit via the system bus, an ARM’s Advanced Microcontroller Bus Architecture (AMBA) bus and a number of peripheral modules. The AMBA bus is the peripheral bus interface in the single-chip OBC.

The High-level Data Link Control (HDLC) controller, the network interface, the true IDE interface and the COordinate Rotation Digital Computing (CORDIC) mathematical co-processor are connected with LEON through the AMBA Advanced high-Performance Bus (AHB). The CAN core is connected via the AMBA Advanced Peripheral Bus (APB).

The LEON processor is based on a new SPARC V8 core developed by ESA.³ It does not require any custom macro-cells apart from synchronous SRAM used for the caches and the register files. The model is extensively parametrical: register window size, cache size, fault-tolerance functions and clocking scheme can be defined through a single configuration file.

Table 1. Soft IP cores of Single-Chip OBC

IP Core	Description	Equivalent Part on OBC 386
ESA LEON	SPARC V8 microprocessor core	386EX Processor
Bootstrap	Bootstrap loader	EPROM version bootloader
EDAC	Memory error-detection-and-correction unit	EDAC FPGA
ESA Hurricane (CAN)	Controller Area Network (CAN) interface	CAN TTC node and CAN Peripheral
HDLC	High-Level Data Link Control (HDLC) interface	HDLC controller chip
Network Interface(FIFO)	Bus LVDS parallel-to-serial converter	10BASE-2 Network chip
IDE interface	The interface to the microdrive	IBM microdrive ⁴ was selected to replace the Solid State Memory (SSM) in OBC 386
CORDIC	Mathematical Coprocessor	387SL Coprocessor

Combined Design Method for Dynamic Reconfiguration

The design flow shown in Figure 2 describes the design process and development tools that have been used for implementation of dynamic

reconfigurable logic. It consists of a static and a dynamic part.⁵ The static part is used to implement that part of the logic, which is not dynamically changed. Furthermore, it initialises the dynamic part of the design or reserves an empty area for the dynamic part to be placed.

The static part is designed using VHDL synthesis in combination with the tools, available from Xilinx for placing, routing and bitstream generation.

The dynamic part of the design controls the configuration of the FPGA during operation of the application, and the tool JBits is used for this. *JBits* is an Application Program Interface (API) to the Xilinx configuration bitstream. This API permits Java applications to dynamically modify Xilinx Virtex bitstreams. *JBits* may be used as a stand-alone tool or as a base to produce other tools, including traditional place-and-route CAD applications, as well as more application specific tools.⁶

With JBits, the programming bitstream of the FPGA can be altered easily with relatively simple commands. The JBits functionality is used in a main JAVA application that implements the user interface of the SoC and controls the reconfiguration. This application has to communicate with the hardware through *Xilinx Hardware Interface (XHWIF)*.⁷ It permits simple porting of JBits to the hardware. It includes methods for reading and writing bitstreams to FPGA's, incrementing the on-board clock and reading and writing to and from the on-board memory. BoardScope is a tool that can read a configuration back from the FPGA, including the state of instantiated flip-flops.

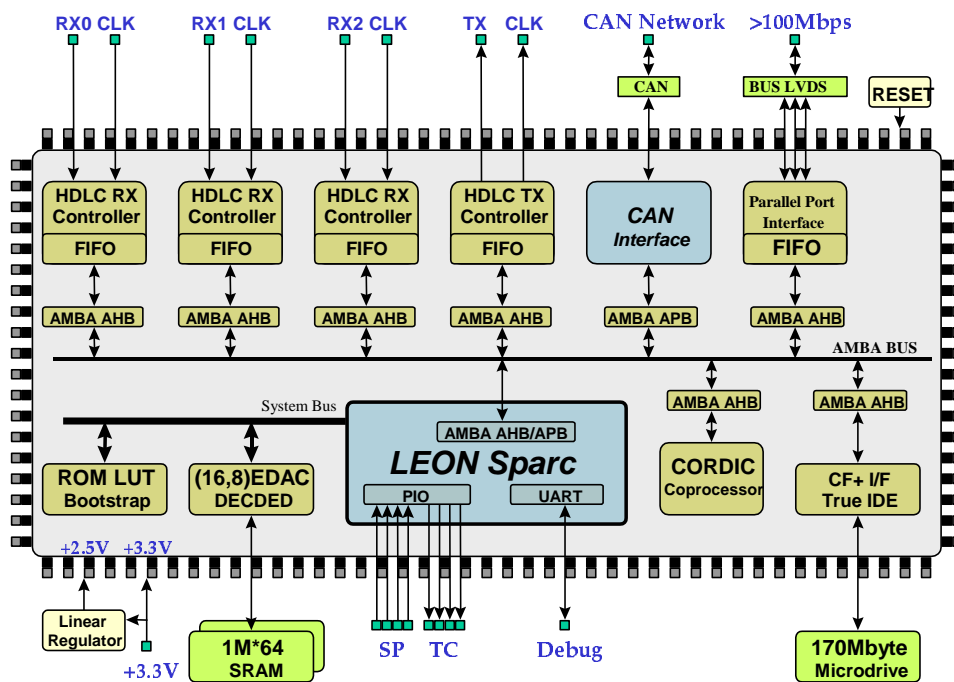


Figure 1. SoC block-diagram

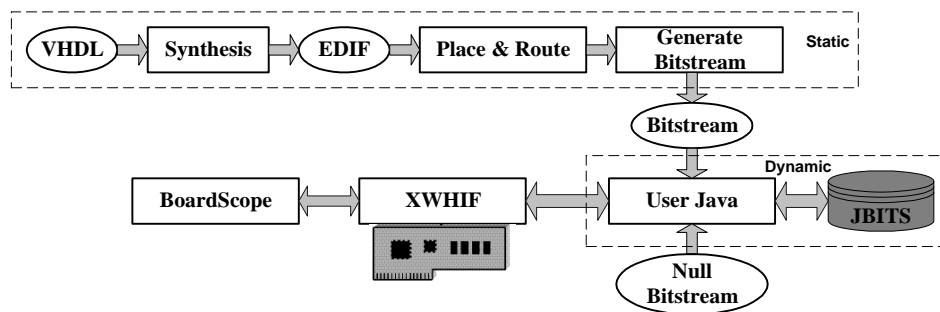


Figure 2. Design flow for dynamic reconfigurable logic

On-Board Run-Time Reconfiguration – Basic Scheme

At the first stage of the development of a reconfigurable SoC a basic RTR scheme will be implemented. It is illustrated in Figure 3. There are many different kinds of architectures designed for using reconfigurable computing. One of the primary variations among these is the degree of

coupling (with a host microprocessor, which is responsible for controlling of the reconfigurable logic⁸). Here, a microcontroller is used to implement this function. A microcontroller is used to configure the Virtex FPGA, control the scrubbing and decompress the reconfiguration bitstream files.

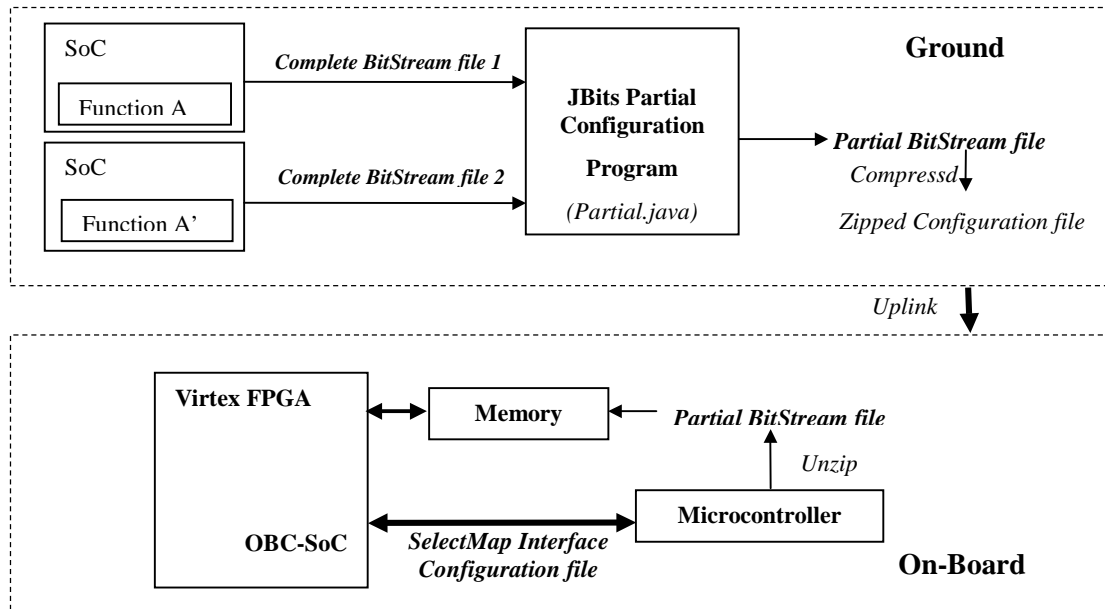


Figure 3 On-Board SoC Run-Time Reconfiguration – Basic Scheme

On Ground:

It is supposed that function A is required to change. Two different bitstream files are the input files. Using JBits Partial Configuration API, the different configuration frames are marked. Then, the partial configuration bitstream file is generated. The *compressed* bitstream file is uploaded through the uplink channel (Figure 3). The partial configuration bitstream is generated by JBits partial configuration program (*Partial.java*).

On Board:

After receiving the compressed .RBT file, the microcontroller will decompress it. The original partial configuration file is recovered and saved into the memory. The microcontroller also handles the RTR of the FPGA. The microcontroller should be some kind of smart CAN-controller or embedded controller, like the 8051.

The XILINX SelectMap is used as the interface between the Virtex FPGA and the microcontroller.

The Microcontroller has three functions: initial configuration of Virtex FPGA, scrubbing and decompressing of the uploaded compressed partial reconfiguration bitstream file.

This scheme allows to implement the local partial RTR of the RSC-OBC for the following situations:

- Updating of an on-board design.
- Fixing a design bug for example fixing a design error in an IP core.
- Adding new functions or updating some function for a new version: For example, modification of some parts of the peripheral circuit in the SoC to increase the speed of data processing (e.g. may be CORDIC or Ethernet part).
- Fault masking: Bypassing a fault to allow the system to continue to operate. For example, bypassing a failed memory block.
- SEU mitigation: Use of a scrubbing method to do SEU mitigation.

Below are given more details about the use of partial RTR of XILINX Virtex FPGAs for SEU mitigation in view of the importance of this aspect for space electronics. The Virtex series are SRAM-based FPGAs. In space, SEUs can harm Virtex FPGAs by altering the logic. Investigations by XILINX have shown that partial RTR in Virtex can be used for the purpose of correcting Single Event Upsets induced by cosmic rays to the configuration memory array.⁹ Now, the time to correct one frame that contains the SEU effected cell has fallen to a mere 3 μ s.¹⁰

An efficient method of SEU correction is **Scrubbing**. Scrubbing simply rewrites the device bitstream, so the time to repair an error is the scrub cycle time. Cycle time can be of the order of a few milliseconds and varies with device density. Continuous readback in conjunction with a detection algorithm (bit compare, CRC, etc) provides data on upsets encountered, time and frequency.

For Virtex XCV800, the word number of CLB is **142902**. Thus, it requires a 20-bit counter. The host system controls the scrub rate. The scrub rate as a percentage is calculated by the following equation:

$$Scrub_Rate = \frac{Scrub_cycle_time}{Time_interval}$$

The scrub rate should be determined by the expected upset rate of the device for the given application. The system should scrub, on average, ten times between upsets. It is difficult to determine the bit upset rate for which there is a consequence in terms of a device upset.¹¹ For LEO satellite, in 1991, the upset rate of XCV300 was 20.9/day.¹² If we were to assume a bit upset rate of once per hour and a configuration clock frequency of 30MHz, then the scrub rate should be one once every six minutes. Thus, the scrub time, for a XCV800 is 20 ms. Therefore, the scrub rate as a percentage would be 0.005%.

LEO satellites are visible for only 10 minutes at a time, six times a day presenting typical LEO-spacecraft-communication-problems. Normally, the upload speed of the radio link is slow (e.g. SSTL -- 9.6Kbit/s). But the configuration file (.BIT) for Virtex is over 1Mbit (1.7 Mbit for a V300, 4.6 Mbit for a V800). It is difficult to upload such a big file, although the partial configuration file is much less than the complete one. One solution is to upload the compressed .BIT file. Even using the normal compressed algorithm, a .BIT file can be compressed up to

75%. The worst situation for the complete configuration file to be uploaded was supported. For a V800, the complete configuration file can be compressed from 4.6 Mbit to 1.03 Mbit. It only takes about 107 seconds to upload this compressed BIT file. The on-board microcontroller can decompress this file to recover the original one.

On-Board Run-Time Reconfiguration – Client-Server Scheme

Theoretically, the RSC-OBC can be partially run-time reconfigured in a local or remote way. It runs the Java Runtime Environment and communicates with the ground station via Internet protocols (TCP/IP). In this scheme, RSC-OBC acts as an equivalent to an application server (hardware configuration), allowing client Java programs to run on the server and use its resource, including local (on-board) and remote access (from the ground). This design is based on JBits. The partial RTR of the Virtex based RSC-OBC is achieved by integrating JBits and the XILINX hardware interface (XHWIF) API. The RSC-OBC can be accessed via any TCP/IP network using the XHWIF interface remote access capability. A debugging tool, such as BoardScope, is used to communicate with the RSC-OBC through an XHWIF server, which is an application that implements the XHWIF interface. This capability enables the RSC-OBC to be debugged through remote access, even in space. When porting an RSC-OBC board to the XHWIF, a Java native interface (JNI) is used to accomplish Java and C programs.

The RSC-OBC can be partially run-time reconfigured in a remote way. It runs the Java Runtime Environment and communicates with the ground station via Internet protocols (TCP/IP). Here, the RSC-OBC acts as an equivalent to an application server (hardware configuration), allowing client Java programs to run on the server and use its resource, including local (on-board) and remote access (from the ground) (Figure 4). In the most common World Wide Web client-server model, the “client” is an Internet user’s Web browser; the “server” is the Web service from which the user wants information. When the user clicks on a menu item in the browser, the client (browser) sends a request to the server, which responds by sending back a Java applet to be run on the client machine. In this scenario, ground control on Earth wishes to upload a Java program to the satellite (the server) for executing

the hardware reconfiguration. This scheme can be illustrated in Figure 5.

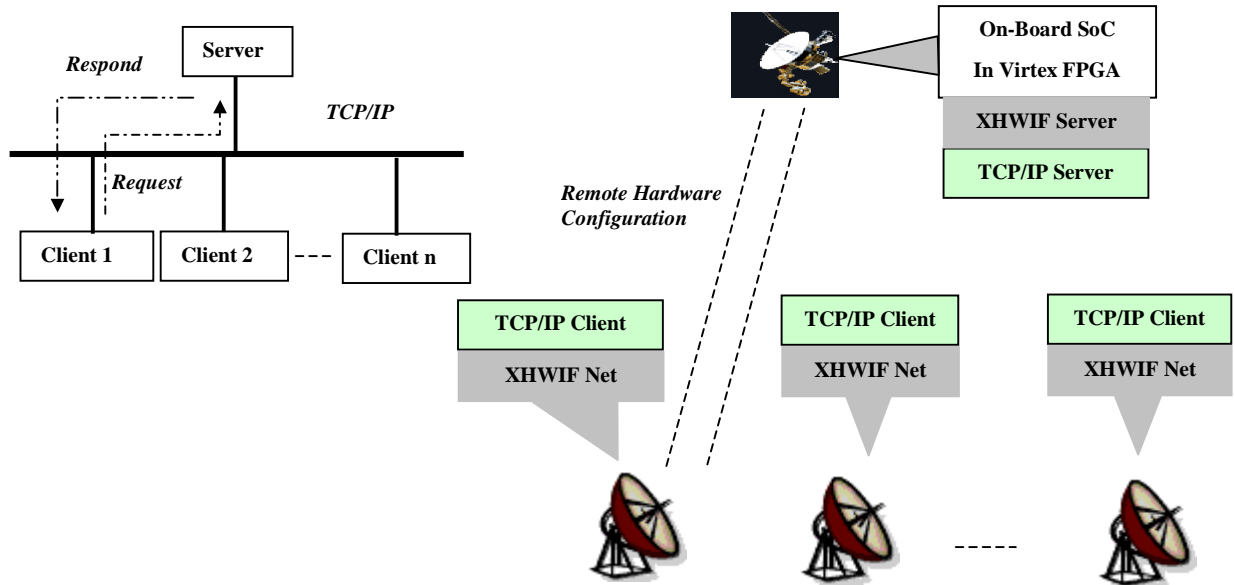


Figure 4. On-Board SoC Run-Time Reconfiguration – Client-Server Scheme

On-Board Java Implementation Options: JBits is based on Sun JDK run-time environments. The full Java platform includes all of the Java run-time libraries, and is intended for desktop implementations. Personal Java is a stripped-down version; the designer can choose which components to include. Embedded Java is even smaller. Normally, the Java Virtual Machine runs on top of an operating system. Sun Microsystems provides the Java OS, an operating system dedicated to Java, with the possibility of reducing the overhead and speeding up operation by eliminating several translation steps between the running program and the OS. The Java OS alone, with no application support, is reported to fit in 128 Kbytes of RAM and 512 Kbytes of ROM. With full application support but no windowing interface (unnecessary on the satellite), about 4 Mbytes each of RAM and ROM are required. It is obvious that this system requirement is difficult for Small Satellite on-board resources. As an alternative to the software-based Java Virtual Machine, several chips are available that

implement Java bytecodes directly. Coupled with the Linux OS, a Java chip offers the ultimate solution for high performance in a small size. Some of these chips can run Java programs at speeds rivaling that of traditional CPUs.

Table 2. Simulation of the Client-Server Scheme for On-Board SoC RTR

Client-Server Scheme	Simulation
Ground Station (Client) <ul style="list-style-type: none"> • Uplink and downlink • Java Applications • JBits Environment (BoardScope) • HWIFnet 	PC (Client) <ul style="list-style-type: none"> • Long distance network (TCP/IP) • Java Applications • JBits Environment (BoardScope) • HWIFnet
Small Satellite (Server) <ul style="list-style-type: none"> • Java Chip • XHWIFServer and XHWIF 	PC (Server) <ul style="list-style-type: none"> • JVM • XHWIFServer and XHWIF (Java, C program and DLL)
Reconfigurable Target <ul style="list-style-type: none"> • On-Board RTR SoC 	Reconfigurable Target <ul style="list-style-type: none"> • XESS prototyping board

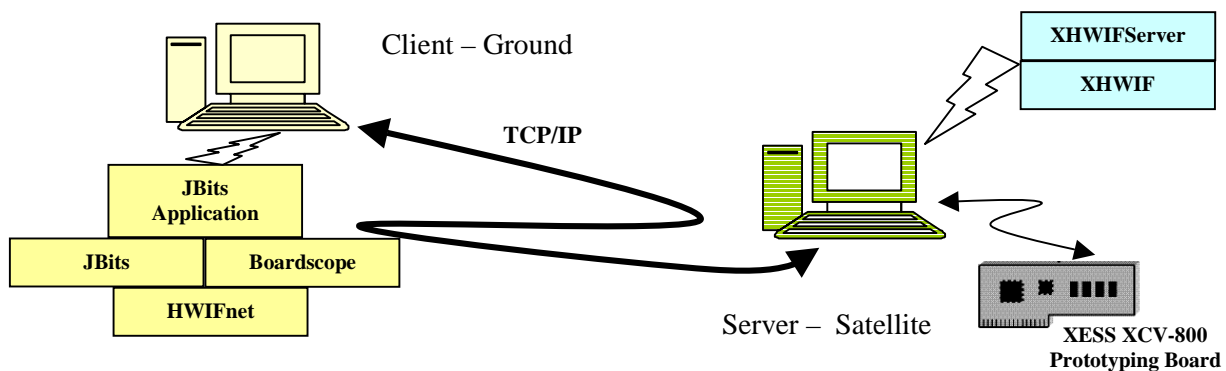


Figure 5. Simulation of Client-Server Scheme of On-Board SoC RTR

Conclusions

Because of the unusual operating conditions, and the need for reliability, satellite electronics have typically used a very conservative design approach¹³. RTR has not been utilized very often for on-board space applications, although reconfigurable FPGAs have been widely used in terrestrial equipment for over a decade. This paper has shown that the application of RC technology to space is technically possible and can lead to very significant results. A reconfigurable system-on-a-chip can be very advantageous to small satellite on-board data handling systems allowing to perform on-board hardware bug fixes, hardware updates, single-event upsets (SEU) mitigation and also it can be used for a generic circuit board design.

In this paper SRAM-Based XILINX Virtex FPGAs are chosen as media for Run-Time-Reconfiguration. However this kind of media is vulnerable to damage by SEUs in space environment. Scrubbing applied to Virtex FPGAs can provide a relatively simple solution to SEUs correction in space since it does not require any readback or data verification operations, nor does it require any data generation when reloading the data frame^[14].

Two schemes for Run-Time Reconfiguration on-board small satellites have been proposed - a basic scheme and a Client-Server scheme. Several Commercial-of-Shelf (COTS) and State-of-the-Art technologies (e.g SoC, RC,

Internet Communication) are combined in these schemes.

The application of Run-Time Reconfiguration to small satellites on-board circuit design marks the beginning of a new revolutionary stage in space electronics. It will allow to solve problems which have not been accomplished under the traditional electronic technology.

Reference

- [1] "On-board Data Handling: OBC 386", http://www.sstl.co.uk/datasheets/Subsys_obc386.pdf
- [2] H.Tiggler, T. Vladimirova, D. Zheng, J. Gaisler "Experiences Designing a System-on-a-chip for Small Satellite Data Processing and Control", *MAPLD'00, 2000 Military and Aerospace Applications of Programmable Devices and Technologies Conference*
- [3] J. Gaisler, "A Portable Fault-tolerant Microprocessor Based on the SPARC V8 Architecture," *PROCEEDINGS OF DASIA (DATA SYSTEMS IN AEROSPACE) 99, Portugal, May 1999*
- [4] IBM Microdrive, <http://www.storage.ibm.com/hardsoft/diskdrdl/micro>
- [5] P. Bellows, B. Hutchings, "JHDL - A HDL for Reconfigurable Systems", *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, p. 175, 1998.
- [6] Xilinx Inc, "JBits Xilinx Reconfigurable Computing Platform", *JBits 2.6 Tutorial*, 2001.

[7] S. A. Guccione and Delon Levi. "Run-Time Parameterizable Cores." *Proceedings of the 9th International Workshop on Field-Programmable Logic and Applications, FPL 1999*.

[8] K. Compton, S. Hauck, "Configurable Computing: A Survey of Systems and Software" (PDF), *ACM Computing Surveys*, 2000.

[9] C. Carmichael, "Correcting Single-Event Upsets Through Virtex Partial Configuration", <http://www.xilinx.com/xapp/xapp216.pdf>

[10] C.Carmichael, E. Fuller, P. Blain and M. Caffrey, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", *Proceedings of the Military and Aerospace Applications of Programmable Devices and Technologies Conference (MAPLD'99)*, B2A September 28-30, 1999, Maryland,

[11] E. Fuller, M. Caffrey, P. Blain, C. Carmichael, "Radiation Test Results of the Virtex FPGA and ZBT SRAM for Space Based Reconfigurable Computing", *1999 Military and Aerospace Applications of Programmable Devices and Technologies Conference, C_2, September 1999* <http://www.mrc.unm.edu/symposiums/symp99/s3/fuller/fuller.html>

[12] The Programmable Logic Data Book, Xilinx, Inc., San Jose, CA.

[¹³] C.D. Jilla, D.W. Miller, "Satellite Design: Past, Present and Future," *International Journal of Small Satellite Engineering*, Vol. 1, No. 1, (1995)

<http://www.ee.surrey.ac.uk/EE/CSER/UOSAT/IJSSE/issue1/cjilla/cjilla.html>

[¹⁴] C. Carmichael, "Correcting Single-Event Upsets Through Virtex Partial Configuration", XILINX Application Notes, XAPP216(V1.0), <http://support.xilinx.com/xapp/xapp216.pdf>