Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 546217, 12 pages
doi:10.1155/2010/546217

Research Article

Space-Based FPGA Radio Receiver Design, Debug, and Development of a Radiation-Tolerant Computing System

Zachary K. Baker, Mark E. Dunham, Keith Morgan, Michael Pigue, Matthew Stettler, Paul Graham, Eric N. Schmierer, and John Power

Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Correspondence should be addressed to Zachary K. Baker, zbaker@lanl.gov

Received 5 March 2010; Revised 28 July 2010; Accepted 14 September 2010

Academic Editor: Lionel Torres

Los Alamos has recently completed the latest in a series of Reconfigurable Software Radios, which incorporates several key innovations in both hardware design and algorithms. Due to our focus on satellite applications, each design must extract the best size, weight, and power performance possible from the ensemble of Commodity Off-the-Shelf (COTS) parts available at the time of design. A large component of our work lies in determining if a given part will survive in space and how it will fail under various space radiation conditions. Using two Xilinx Virtex 4 FPGAs, we have achieved 1 TeraOps/second signal processing on a 1920 Megabit/second datastream. This processing capability enables very advanced algorithms such as our wideband RF compression scheme to operate at the source, allowing bandwidth-constrained applications to deliver previously unattainable performance. This paper will discuss the design of the payload, making electronics survivable in the radiation of space, and techniques for debug.

1. Introduction and Background

Since the original Adaptive Computing Systems Program at DARPA in 1994, Los Alamos (LANL) has been developing RF processing systems with FPGA devices, now known in the literature as Software Defined Radios (SDR). Originally designed with Altera and Xilinx first generation logic arrays, these receivers have grown with the chip families underlying them [1].

The other components of an SDR have gained in performance as well. In particular, modern ADC technology now enables Direct Down-Conversion analog architectures due to the GHz input bandwidths available in 12 to 16 bit converters. Meanwhile point-of-load power conversion, high speed SRAM, and very large NonVolatile RAM technology all support extreme performance from the processing side. The net result is SDRs which are truly special-purpose supercomputers, and which can execute complex algorithms in real-time, over bandwidths meeting or exceeding those required for modern communications systems.

We focus on satellite-capable SDR designs, since the need for reconfiguration is greatest in these systems which are not accessible after launch. Space deployment requires a robust research effort in COTS Radiation Tolerance (COTS-RT). Identifying promising COTS-RT parts is key to our success, since, for example, the Xilinx Virtex 4 is 1000 times more capable than the best RadHard microprocessors available [2].

In the last decade radiation tolerance has advanced to the point that 400 kRad Total Ionizing Dose (TID) is routinely seen in commercial parts, and destructive latchup is disappearing in many families. These improvements are the result of fabrication process changes intended to improve electrical performance, but the radiation tolerance is a direct result as well. Remaining to be dealt with beyond TID and destructive latchup is Single Event Effects (SEEs) performance which in general is degrading as device feature size shrinks [3–5]. The LANL Rad Tolerance program therefore devotes much of its R&D to methods for mitigating SEE in modern electronics.

No RF receiver is complete without algorithms, which are often point designs to satisfy a specific communications

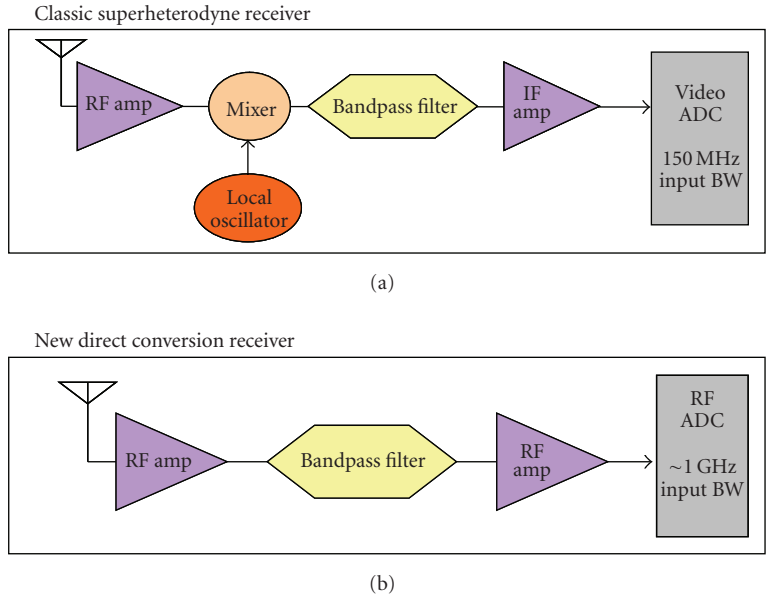

Figure 1: Comparing direct conversion to super-heterodyne architectures. The direct conversion approach does not require an external local oscillator. Rather, the FPGA synthesizes a local oscillator and computationally tunes the desired RF channel.

need. LANL is more interested in general scientific applications that are flexible for a wide range of signal classes. Therefore we describe two such applications in some detail here, an incoherent wideband signal detector, and a fully coherent wideband transform and compression engine. Such algorithms form the building blocks of subsequent user-specific applications, or perform as general purpose science data gatherers. LANL has a well-known VHF lightning research program, based around global monitoring of severe storms via the GPS spacecraft. Radio astronomy also forms another area of interest, especially radio bursts such as pulsars. Our radio is solely used as a receiver.

It is both a challenge and a blessing to the space electronics community that COTS electronics evolves at a rate of at least 2.5 generations/launch. In other words, by the time a new satellite design is delivered to the launchpad, the COTS technology will have advanced by at least 2.5 more generations. This paper describes a delivered receiver that is now two generations behind FPGA state-of-the-art, but we also discuss work with current generation parts in the laboratory.

2. Direct Conversion Principle of Operation

A key innovation of the LANL TeraOps SDR (T-SDR) is the use of direct conversion radio architecture. This feature is diagrammed in Figure 1, where it is seen that higher dynamic range is achieved concurrently with lower power, as long as analog tunability is sacrificed. In direct conversion, tuning is moved into the digital domain, while the analog section becomes a wideband band-selection filter. This simplifies design and manufacture, compared to traditional mixer-IF designs. Most importantly, significant flexibility is acheived by replacing fixed analog components with reconfigurable computation. Our receiver architecture utilizes a 60 MHz digital bandwidth (120 MegaSamples/Second) 16 bit ADC.

As Figure 1 shows, Nyquist's original theory specifies the bandwidth supported by a particular sample rate, but not the actual band, which can exist between any integer multiple of the Nyquist rate which is within the input bandwidth of the analog system (Other terms for this process are used in the literature include Nyquist conversion and aliasing [6]). The ADC does digitally downconvert the input RF signal to baseband, since the output can be interpreted as any of the bands shown in the inset of Figure 1, entitled Nyquist Downconversion (Aliasing). In essence, the digital output stream always occupies the lowest Band 1. The input may be positioned in any of the higher bands, aliasing to Band 1 (a 130 MHz signal would alias through the 120 MHz sample rate to 10 MHz). In the case of the Linear Technology LT2208 ADC, there are at least 18 possible Nyquist bands one can use, albeit with increasing degradation of effective bits.

Our experimental receiver (Figure 2) explores four separate Nyquist bands of the 18 possible, each exploiting a different portion of the ADC performance. Even at the highest bands (>1 GHz), we still obtain greater than 10 effective bits (Section 6). One common theme found while working with our filter suppliers was that we needed 10 MHz transition zones for 50 MHz bandwidth and reasonable size, so we relaxed attenuation at the Nyquist boundaries to 40 dB, as long as we were 80 dB down within another 5 MHz.

3. Detailed Architecture

In order to clearly outline the design aspects enabled by modern chip capabilities, we now describe each system component shown on Figure 3 in some detail.

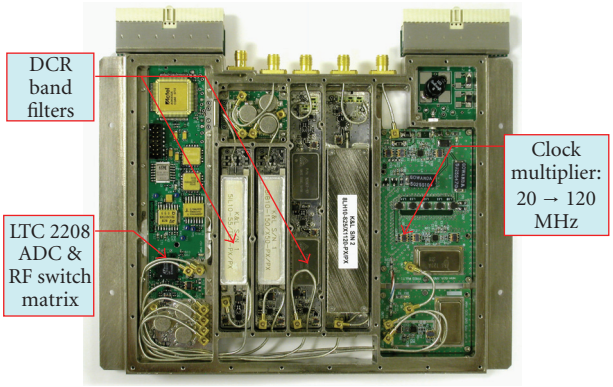

Figure 2: Direct conversion Receiver (DCR) System including analog bandpass filters and analog-to-digital conversion.

3.1. Input Antenna and Direct Conversion Receiver. A 16 bit, 120 Megasample/second ADC converts selected 50 MHz RF bands into Digital Intermediate Frequency (DIF) data streams, downconverting them at the same time it digitizes, in a process known as Direct Conversion. Direct Conversion is a major advancement suitable for HF, VHF, and UHF receivers, and offers gains in both performance and power consumption. The output stream from our Direct Conversion Tuner-Digitizer (DCR) is a 16 bit-wide parallel stream of samples at the 120 MHz clock rate, leading to an aggregate data rate of 1.92 Gigabits/second. Our sample clock remains at one frequency, but there is no reason not to change sample clock rate when input bands change, if desired.

One factor confounding good signal reception at HF and VHF frequencies is that well-matched antennas must subtend one-half wavelength in some dimension. However, at 50 MHz, the wavelength is already 3 meters, and keeps growing inversely with frequency. Hence, T-SDR uses active antennas (Figure 4) which obtain the same pattern as full sized elements, but are less than 1/20 wavelength at the lower end of range. Use of such a short antenna requires nonconventional matching amplifiers to deal with the very high impedances present. But in the HF and low VHF the spectrum is totally dominated by excess ambient noise, for instance at 10 MHz the total noise is ~900,000 K, or 3000 times more than at frequencies higher than 200 MHz [7]. Thus, the system noise figure will be dominated by this 34 dB ambient value, as long as coupling efficiency is adequate and LNA temperature is much lower than ambient.

3.2. The Real-Time Signal Processor (RTSP). The digitized IF output of the DCR is fed to a RTSP, containing two Xilinx Virtex 4 Platform FPGAs and their associated Quad-Data Rate SRAM memories.

The RTSP board and heatsink is shown in Figure 5. Benchmarks of these devices have already shown the ability to process 10 Gigabits/second of data, far higher than our 2 Gigabit/second stream. For a discussion of the partial reconfiguration scrubbing used to mitigate SEE in these devices, see [1, 3].

In the space allocated to the RTSP on cPCI, we have been able to accommodate seven synchronous Quad Data-rate Static RAM (QDR-SRAM) memories, each containing 32 Megabits of data organized 36 bits wide by 1 Megaword deep. ECC through Hamming codes is implemented in the FPGA. Processed data exits the RTSP via a PCI bus interface managed by an ACTEL RTAX Radiation-Hardened PLA. Our requirement is that ~10 Mbytes/second be transferred over the PCI bus. The RTAX is a key part because it is one of the few devices built for use in space, as opposed to COTS hardware repurposed for space. The space-qualified capability is important because the satellite has stringent Do-No-Harm requirements, and the RTAX isolates the satellite from the nonspace-qualified COTS hardware.

A final element in the design of the RTSP are new Point-of-Load power converters from Enpirion, which have been tested by Yale University and LANL to be extremely Rad Tolerant [8]. These are used as the primary supplies for both FPGAs, and for the QDR-SRAM. They operate at >75% efficiency.

3.3. SVIM. Following the RTSP is a Satellite Vehicle Interface & Memory module (SVIM), based on a SPARC 32 bit integer processor. The SVIM card (Figure 6) receives the PCI data, and temporarily stores it in the main 1 Gb memory via DMA. The SPARC is a European Space Agency standard design, and is radiation hardened for LEO and Geosynchronous orbit conditions.

The SVIM contains the Atmel AT697 SPARC controller, spacecraft bus interfaces, and also provides the radiation hardened and nonvolatile storage of all software for the T-SDR payload. The new space processor is rated best-in-class at 100 Mips/Watt, operating at a 80 MHz core clock. Memory supporting the microprocessor is organized as follows.

(i) 2048 Mbit of Start-Up ROM (SUROM) are provided to the SPARC alone, so that critical OS level functions are stored in RadHard memory. All elements of software required to decompress and load FPGA bitstreams from telemetry are contained in this SUROM.

(ii) A bank of blank Xilinx PROM 48 Megabits deep, with hard ECC. These are one-time programmable devices, but programmable on-orbit. We also store the initial applications in this bank of PROM. Compression is used in order to reduce bitstream image size by 4X–16X, which is bitstream dependent. Typical FPGA bitstreams are ~40 Mbits in size.

(iii) A 64 Mb bank of new BAE Chalcogenide-RAM, with hard ECC, which can be reprogrammed at will, including on-orbit. This is the target memory for most reprogramming after launch. It also must contain the SPARC application code associated with the FPGA application.

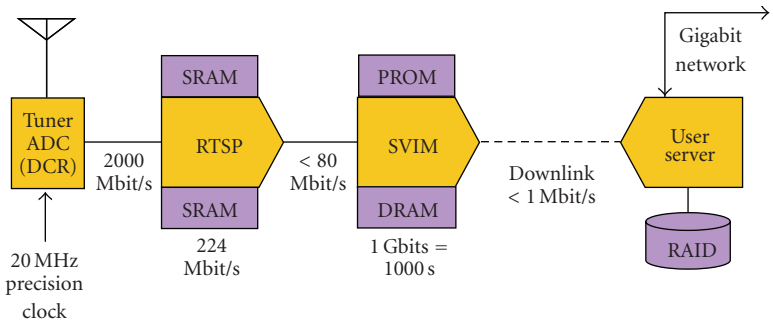

Figure 3: System architecture for satellite receiver system. The ADC accepts a 20 MHz clock and produces a 120 Msps 16 bit data stream. The FPGA-based RTSP processes the RF stream into full-bandwidth snapshots and data products. The radiation-hardened flight computer (SVIM) provides command and control as well as packaging data for transmission to the ground.

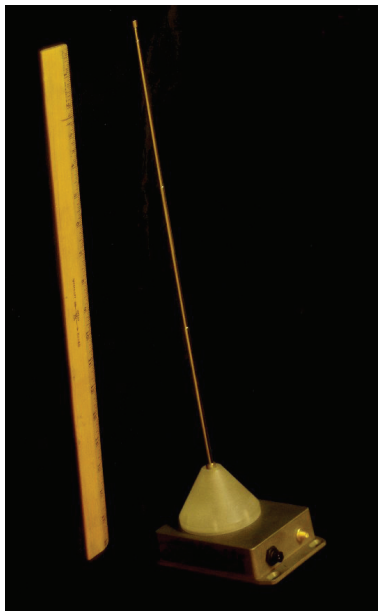

Figure 4: Active antenna assembly with 18″ ruler for size comparison.

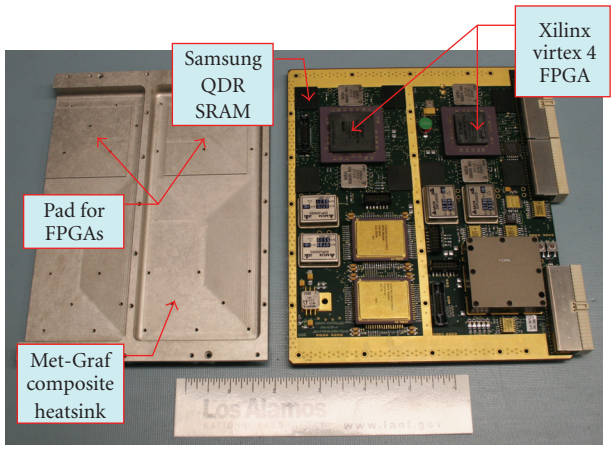

Figure 5: Real-Time Signal Processor (RTSP) board with Xilinx Virtex 4 LX200 and SX55, seven banks of Quad Data-Rate (QDR) memory, and radiation-hardened scrubbing and interface circuitry.

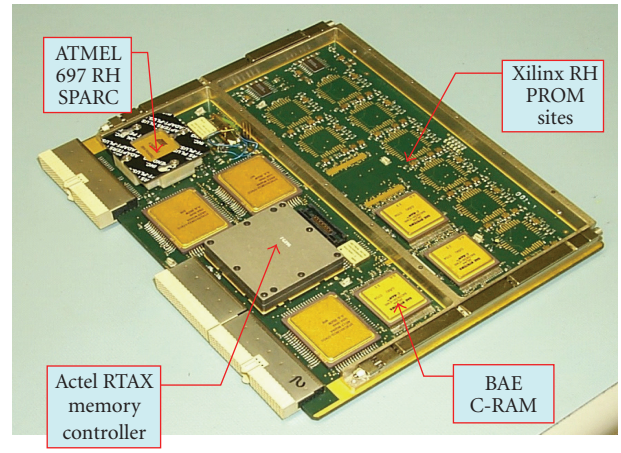

Figure 6: Flight computer board with radiation-hardened processor, SRAM, and Xilinx PROM programming circuits.

The SVIM is tasked with 3 major functions: responding to and transmitting telemetry, retrieving configuration data dynamically from C-RAM memory to support SEU mitigation, and operating the downlink with any associated data compression or thinning operations. The SPARC is capable, but will have to assign interrupt priority to these tasks in normal operations, so user applications reside primarily in the RTSP, with the SVIM supplying custom I/O functions.

4. FPGA Fault Injection Testing

As the FPGAs in our system are only radiation tolerant (and not radiation *hardened*) it is necessary to determine the impact of single-event functional interrupts (SEFIs) and single-event upsets (SEUs) on application availability. An FPGA application's availability is a function of its application error rate and the "down time" caused by application errors.

An application error is very simply an event in which the FPGA application's data output is corrupted. By definition all SEFIs cause application errors. All SEUs, however, do not cause an application error in an FPGA since a given application only uses a subset of the FPGA's available

configuration bits. As a result, the impact of SEFIs is independent of application and can be measured once for a given FPGA device. The impact of SEUs, however, is application dependent and must be measured for each application.

The duration of the data corruption caused by the application error is the "down time". The "down time" caused by a SEFI is system-dependent and can be measured once for a system. The "down time" caused by an SEU-induced application error is application-specific and therefore must be measured.

It is possible to measure an application's error rate with ground-based accelerators, but it is expensive and the radiation destroys hardware. A much less expensive nondestructive method with high fidelity is SEU emulation via fault injection.

4.1. Fault Injection Mechanism. More information about the mechanics of FPGA fault-injection can be found in [9, 10]. By emulating all possible SEUs in the FPGA's configuration memory, it is possible to determine the percentage of SEUs that cause an application error. Also, for each bit in the configuration bitstream, the following important information can be gathered during fault injection regarding an SEU of that bit.

(i) *Criticality*: If an SEU injected into a configuration bit causes an output error, the application error for the bit is classified as "critical" otherwise it is classified as "noncritical" (These are sometimes referred to in the literature as "sensitive" and "nonsensitive.").

(ii) *Recovery Type*: Each critical application error is subclassified as "self-recoverable" or "unrecoverable (These are sometimes referred to in the literature as "persistent" and "nonpersistent" [11])." An application error is "self-recoverable" if the application automatically recovers to an error-free state once the SEU was repaired. An application error is "unrecoverable" if the application requires a reset to return to an error-free state.

(iii) *Recovery Time*: For critical, self-recoverable application errors, the number of clock cycles with output errors is recorded. For critical, unrecoverable application errors, the number of clock cycles with output errors is system dependent.

Once fault-injection has been performed and the preceding information collected it is possible to predict the application's availability. System availability is calculated with (1) which is simply the ratio of "up time" to "total time," where "up time" is the "total time" minus the "down time" as follows:

$$\text{Availability} = \frac{\text{uptime}}{\text{totaltime}}. \tag{1}$$

To use (1) to measure an application's availability it is first necessary to calculate the "down time" caused by each type of application error. The "down time" caused by self-recoverable errors varies with each SEU. It is recorded during fault-injection for each application error as the recovery time, or number of clock cycles with output errors. The expected value of the "down time" due to a self-recoverable error for a given time period is simply the product of the average number of cycles in error, the clock period and the number of self-recoverable errors expected (dependent on the orbit SEU rate) during the time period. Mathematically this can be written as follows:

$$\begin{aligned} &E(\text{downtime}_{\text{self-recoverable}}) \\ &\quad = E(\text{cyclecount}_{\text{self-recoverable}}) \\ &\qquad \times t_{\text{clockperiod}} \times E(\text{events}_{\text{self-recoverable}}). \end{aligned} \tag{2}$$

The "down time" caused by an unrecoverable error varies with the application duty cycle and when the error begins within the duty cycle. The distribution of "down time" caused by unrecoverable errors is assumed to be uniform. As a result, the expected value for the down time for a given period of time due to an unrecoverable error is simply the product of one-half of the duty cycle and the number of unrecoverable errors expected (dependent on the orbit SEU rate) for the given time period. Mathematically, this can be written as follows:

$$E(\text{downtime}_{\text{unrecoverable}}) = \frac{t_{\text{dutycycle}}}{2} \times E(\text{events}_{\text{unrecoverable}}). \tag{3}$$

The "down time" caused by each SEFI is not measured by fault-injection but rather is a measure of the system's average response time for detecting and recovering from a SEFI. In our system, for example, the time to detect and recover from a SEFI was measured at approximately 5 seconds. In other words, the contribution of SEFIs to down time for a given period is the product of the SEFI response time and the number of SEFIs expected for the given time period. Mathematically, this can be written as follows:

$$E(\text{downtime}_{\text{SEFI}}) = t_{\text{SEFI-response}} \times E(\text{events}_{\text{SEFI}}). \tag{4}$$

The overall estimated "down time" is the sum of the "down time" for self-recoverable errors, unrecoverable errors, and SEFIs for a given time period. Once this is known, (1) can be applied as in (5):

$$\text{Availability} = \frac{\text{totaltime} - (E(\text{downtime}_{\text{self-recoverable}}) + E(\text{downtime}_{\text{unrecoverable}}) + E(\text{downtime}_{\text{SEFI}}))}{\text{totaltime}}. \tag{5}$$

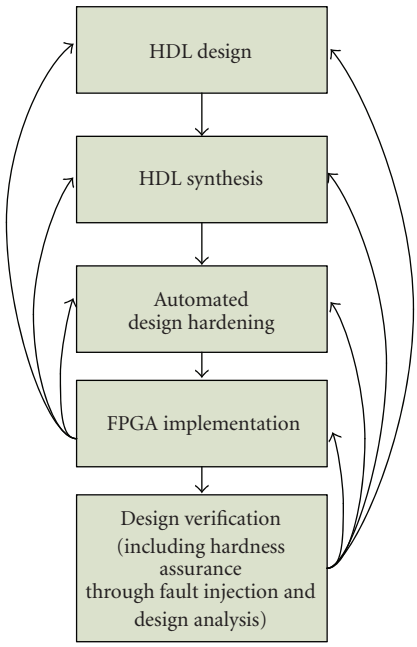

Figure 7: Recommended FPGA design flow for space (only some feedback paths shown).

4.2. Codesign for Robustness. In addition to its utility for developing a dynamic SEU cross-section and availability model for a particular device and application, fault injection provides some additional insight into the robustness of Xilinx Virtex-4 FPGAs and SEU-mitigated designs in space. A few of those beneficial insights will be described in the next several paragraphs.

First, fault injection can provide feedback to designers regarding the actual robustness of the design techniques applied to a particular design, providing an opportunity to gain further insight into design practices for space. For instance, it can provide some feedback on how to effectively apply triple-modular redundancy (TMR) to a FPGA design intended for space. Though the concept is quite simple, the effective execution of applying TMR to a particular design and the resources it uses on a particular FPGA has proven to be quite challenging due to the complexity of the FPGAs themselves and the fact that design software for FPGAs is completely unaware of reliability- and SEU-related issues. In fact, design tools frequently try to optimize designs in ways that actually unintentionally reduce application reliability. Additional understanding about the robustness of TMR and other reliability techniques used will be developed through fault injection.

As an example of this first benefit, for the Radiation Tolerance Test application (described later), we performed fault injection on our final design and noticed several anomalies in its operation. In cases where TMR should have masked errors, application output errors were being observed. By looking at the design and the design build reports, we observed that we were incorrectly using the automated design hardening tool (BLTMR) for handling our QDR memory interfaces, which could not be triplicated. After identifying the problem, we had to return to the automated design hardening stage of the design flow and rerun the BLTMR tool on our design with different mitigation options to resolve the problem. After running the design through the complete design implementation flow, we were able to perform fault injection again and verify that TMR was working properly before burning the final design into PROMs on the SVIM.

Second, fault injection can provide designers with resource-specific vulnerability information. Due to LANLs intimate understanding of Virtex-4 FPGAs and their programming data, we can effectively quantify not only the number of upsets that can affect the operation of a FPGA design but also what types of resources are affected by SEUs, providing insight into the cause of particular errors in an FPGA design. Another key aspect of fault injection is its function in a "best practices" FPGA design flow for space. As illustrated in Figure 7, the LANL approach to FPGA design for space includes a step for automated hardening of the design for space as well as hardness assurance efforts in the design verification step. By developing a fault injection test setup for MRM-LANL, we can provide better design verification tools for designers to verify that their designs meet mission hardness requirements, a key part of the design verification step of the FPGA design flow.

Third, fault injection can provide system developers a validation that their SEU scrubbing software and hardware are working properly. Both in this system and other FPGA-based space systems with which we are familiar, fault injection has been a key test for ensuring that the SEU scrubbing and error reporting subsystems are working properly. In the case of the T-SDR, we were able to use fault injection to identify a data alignment issue in the SEU scrubbing hardware and modify the associated software to compensate for the issue before final delivery of the T-SDR system. We are aware of at least one other FPGA-based system that would have benefited from using fault injection for validating their SEU scrubbing hardware and software. Since actual fault injection on the FPGA hardware was not used during system test, an error in the SEU scrubbing subsystem was not caught until the hardware was on orbit. This was later fixed, but at significant expense relative to if the developers had caught the issue during development.

For the SoftwareDefined Radio application (described later), we applied partial TMR quite judiciously. The clock rate was quite aggressive for the Virtex 4 at 120 MHz. It was clear early on that applying replication and voters would push the maximum clock rate significantly below the target. We reduced the TMR to minimal replication on the LX200. None of the seven QDR interfaces were replicated, as the most trying timing error was in the QDR-SRAM timing. The QDR was inconsistent before TMR was applied. Even though the FPGA nominally met timing, the QDR was plainly mistimed. We utilized the QDR placements from a working nonTMRed build, and applied them as constraints to the TMRed version, resulting in a functioning system.

5. Introduction to In Situ Debug

Visibility in a design is a priceless commodity during debug. Providing that visibility outside the normal channels of a register system and application behavior can be the difference between unexplained failures and mission success. At Los Alamos National Laboratory aggressive launch schedules force engineers to solve problems quickly, thus requiring unique tools. We have recently developed tools that have quickly been found to be invaluable. These JTAG (Joint Test Action Group) tools provide in situ debug, providing an increased amount of debug control and visibility than would otherwise be available. This is possible because we can read and write an arbitrary address space through the JTAG controller. This allows us to set the system to a particular state, set parameters, trigger actions, as well as provide programmed input/output (PIO) to allow for access to a secondary, indirect address space. This is particularly useful for debugging large data-driven applications. For instance, through PIO, we provide access to the entire seven banks of QDR-SRAM in our system from the Linux command line. Because it is meant for interactive debug, the slow JTAG debug connection is not a problem—the system runs more than fast enough for human use.

The JTAG standard is a common standard for running boundary scan tests, running debug tools, and programming memories and FPGAs. Innumerable commercial tools, as well as open-source efforts [12, 13], support the standard. A JTAG connection is one of the several methods to program Xilinx FPGAs [14]. The JTAG state machine is controlled by two pins; TMS (mode select), TCLK (the clock pin, which can be run to near-DC rates). The mode of the state machine determines what happens with the other two JTAG pins, TDI and TDO (data in and data out, resp.,). The TAP state machine is standardized across all platforms, but the registers that are connected to it and that extend into the fabric of the particular chip are determined entirely by the hardware designer. Because Xilinx has provided hooks into FPGA user fabric [14], they have provided the opportunity for a wide variety of powerful tools. For instance, the Picoblaze soft processor can use this approach to load user programs [15] and the Chipscope internal logic analyzer provides cross-hierarchy visibility.

In order to provide in situ access to the JTAG chain, we implement a JTAG host in an embedded microcontroller. In practice, we assume this is a radiation-tolerant processor providing reliable operation, although in our test system we use a Microchip PIC 18f2550 [16]. The controller implements software routines to run the JTAG I/O pins. The software is based on Xilinx Application Note #058 [17], which demonstrates how to implement a SVF file player (Serial Vector Format). This can be used to query ID codes or program the FPGA using SVF files produced through the Xilinx Impact program.

We have extended the functionality to support extended user-mode JTAG commands required to support the debug core. The original source code only plays back previously generated SVF files, whereas our modified version generates TAP controller sequences on the fly. Because the set of extended commands is built within the normal TAP controller architecture, it is a minimal extension to the original source (see Figure 8).

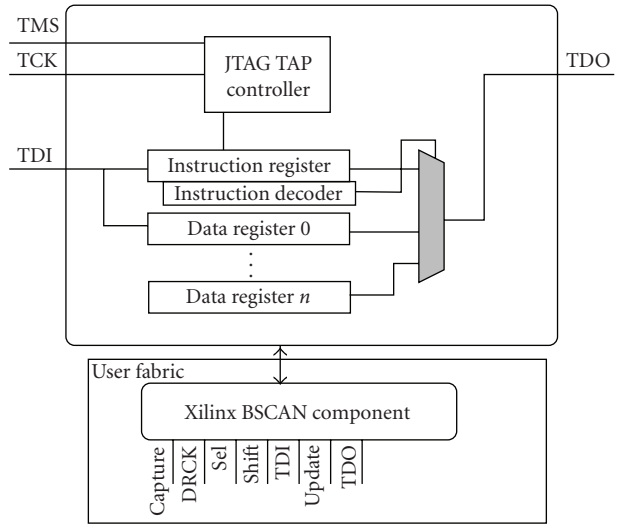

Figure 8: Xilinx internal boundary scan state machine for Virtex 4 [14].

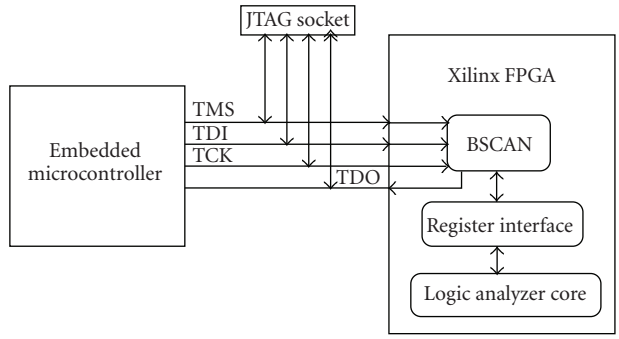

Figure 9: High-level FPGA connection to JTAG controller.

5.1. Implementation Details. The internal debug core resides in user logic. Thus, the FPGA must be programmed for the internal debug system to function, however, IDCODE, FPGA programming, and boundary scan can be accomplished through the JTAG controller without a loaded configuration bitstream. The register debug core is based on the Xilinx BSCAN primitive, which gives user logic access to the boundary scan. The initial code implementing this functionality was based on the S3 Gnat application note [18], but extended significantly to include read/write register access as well as the more elaborate triggering and snapshot system required for the internal digital logic analyzer. The Gnat application note includes firmware examples that we found to be somewhat unreliable on our system. This code was extended to improve its reliability using error correction among other techniques.

In all recent Xilinx Spartan and Virtex devices, there are at least two BSCAN components. We use the second component in the chain, such that the first is available for other debug tools. For instance, Xilinx Chipscope [19] uses

the first component, thus, we can simultaneously have both Chipscope and our tools in the same bitstream (While both components can be instantiated, only one host controller can own the JTAG chain at any time. Thus, the debug cores cannot be operated simultaneously.). It is sensible to instantiate both cores in a system to cover difficult debugging chores, as Chipscope is excellent for logic analyzer chores and our tools are excellent for user-interactive or script-driven data movement.

Our debug core is based on a simple triggering system, which includes a trigger pattern, a trigger mask, and an arm bit. The snapshot system is based on a Block-RAM along with a double-registered input.

5.2. Integrating Microcontroller with FPGA Device. In Figure 9, the connections of the JTAG controller to the JTAG pins of the FPGA slave is illustrated. TMS and TDI have internal pull-ups in Xilinx FPGAs [14]. However, TCLK does not and must be driven high. Thus, in either the open-collector or high-Z option is used, TCLK must be arranged so that it can be cut off to allow an external controller to drive the chain.

The logic analyzer core is based on the register functionality handled directly from the BSCAN primitive. This can be used for generalized register reads and writes in addition to the logic analyzer. We allocated a block of register addresses for replicating access to the control system normally handled through the rad-hard flight computer. The logic analyzer is configured, read, and controlled via the register interface, but its connection to user logic is purely through VHDL port assignment. This is less flexible that Xilinx's postsynthesis black box insertion approach, but it is sufficiently to demonstrate the capabilities of our custom JTAG interface.

This is highly useful in a debug scenario as it can be used independently of the normal communication control channel. This allows in situ system monitoring while normal activities proceed.

5.3. Case Studies. We have already have some success using the tools for development before launch. In particular, the simple register interface has proven to be an irreplaceable tool in solving difficult debug problems. Because the JTAG register system is a simple extension to the normal register command interface (only a few dozen slices are required), it can be included in all builds. We will present three case studies wherein the tools have proven useful.

Case 1. In our first debug challenge, we were experiencing intermittent register write failures. It was rare enough to disregard any obvious problem, and, at 30 MHz and easily meeting the clock constraints it was fairly unlikely the problem was in timing. We have seen some interesting problems where the language construct used to address the array of std_logic vectors has resulted in measureably different behavior in certain conditions (Specifically, when a *for loop* is used to produce a variable input to *conv_std_logic_vector*, the output behaves randomly in some situations. This block of legacy code was still using the nonstandard ieee.std_logic_unsigned library in lieu of the recommended ieee.numeric_std library.), and thus were not certain where the host or slave gate array was to blame.

Using the tools, we were able to connect to the system via JTAG, then write a set of test registers and read them back. This was working fine, so we moved to reading and writing in combination with the failing normal register system. This allows us to determine that the problem lay solely in the mechanical connection between the two chips. The problem was solved through the replacement of the grid array interposer. This is a pad matching the footprint of the part, with a small "spring" for each pin to ensure a stable connection over thermal cycling. They have a limited number of installation cycles before they begin to fail.

Case 2. In our second problematic debug challenge, we struggled to explain the behavior of an application that worked fine when the FPGA was programmed via JTAG but certain application functions failed when programmed via the select MAP interface. The select MAP programming was successful, but it was clear something was being corrupted in the programming sequence. At this point, PROMs for both the FPGA bitfiles as well as the software PROM had been burned, toward an early deadline. This was important because it forced us to try to minimize the changes to either PROM. The situation was made even more intriguing by the fact that any attempt to dump state for debugging by using a new software load would cause the failure to disappear. Thus, normal softwarecontrolled register reads for debug were useless.

However, by connecting to the system through the JTAG interface, we were able to dump the register space and determine that one of the registers was corrupted between the start of FPGA programming and the application being armed. We then forced an application break and dumped the memory location that was the source of the register write. This memory was corrupted as well. Eventually, we determined that the decompression routine was corrupting a single byte outside of its allocated space, which eventually was written to the FPGA. Because the JTAG interface provided access to FPGA application space, we were able to track down a software corruption on the other end of the PCI bus.

Case 3. In another project, an airborne persistent surveillance platform, we wished to develop multiple hardware components in parallel, namely, a high speed serial connection between a microprocessor and the FPGA, and the QDR-SRAM interfaces. Without the microprocessor connection, we had no way to communicate with the FPGA to test the SRAM interfaces. By connecting to the system via JTAG, we were able to move test images in and out of the SRAMs without use of the eventual flight communication path. While the JTAG connection is much slower, it allowed us to make progress where it otherwise would be impossible.

Because the tools operate in a Linux environment, an operator can connect to the server and operate an FPGA

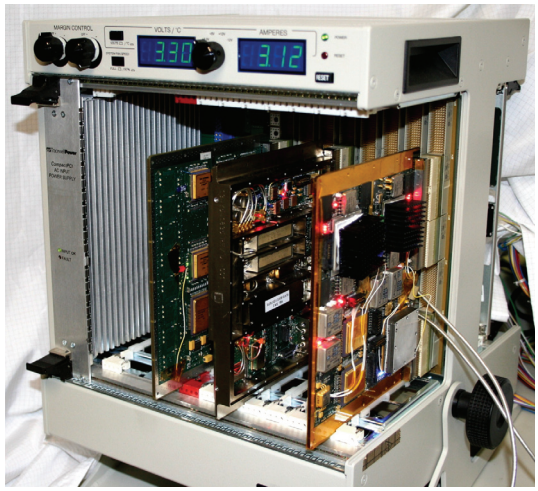

Figure 10: T-SDR modules loaded in a compact-PCI test chassis. The use of commodity compact-PCI chassis allow for cheap and efficient development cycles. Shown right to left are RTSP, DCR, and SVIM in COTS cPCI test chassis.

board remotely from the command line, without having a direct board interface. This is useful not just in a laboratory or commercial setting, but also in educational settings where communicating with a development board is nontrivial. For instance, common Spartan 3 boards for university classes often have no way to talk to the FPGA except buttons and LED readouts. These tools can allow sufficiently fast communication to open a wide variety of application development opportunities.

The functional prototype system operates through a USB connection. This makes it very similiar to a commercial JTAG cable. However, our initial system is just a demonstration for the final system, which will be implemented in a rad-hard microcontroller and connected via SpaceWire to the satellite ground interface. At that point, the interface to the JTAG controller will be via SpaceWire packets, which will be translated from the satellite's real-time command and telemetry interface.

6. Results and Performance Testing

Here we present a few of the applications we have developed for the T-SDR. Our system has been under test for two years as we prepare for flight delivery, and is seen in finished form in Figure 11. A COTS form factor was also chosen, Compact PCI, to to enhance testability and to allow use of low-cost test resources. Figure 10 shows a compact PCI test chassis housing the three system cards. We can replace the RadHard Controller module with commodity PowerPC control modules, enabling device development without requiring emulation of the satellite interfaces.

When inserted in the 3 liter chassis shown in Figure 11, the entire T-SDR weighs 5.5 kg, and has a maximum power consumption of 53 W. The heat must be dissipated through the case. Even with 60% efficient 25 VDC bulk power converters, this enables the 16 k FFT cores in the third

Figure 11: Completed, sealed T-SDR system with satellite interfaces.

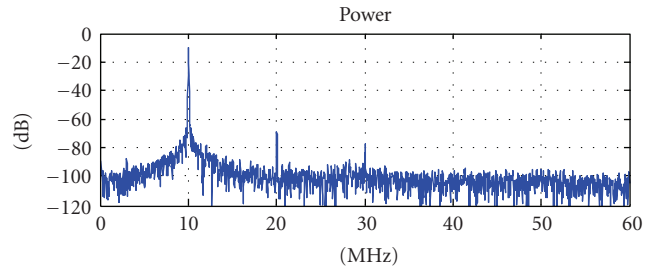

Figure 12: Spectrum output from 130 MHz sine wave input illustrating the low-noise floor of the DCR. Due to the fixed 120 MHz sample rate, the bandpassed signal is aliased to 10 MHz.

application to function at the 120 Msps input rate. The next system architecture (described in Section 7) will reduce the power conversion inefficiency.

The system provides the capability of loading new applications after delivery and launch. This is accomplished through the uploading of new FPGA bitstreams and support software. The system was delivered with two applications, a "parts test" application that will test various individual components (including FPGAs and memory systems), and a SoftwareDefined Radio (SDR) application that will be the main RF data collection system.

6.1. Radiation Tolerance Test Application. As the demands for on-orbit processing increase, the need to insert improved computing technology into space systems also increases. The challenge, of course, is how to carefully adopt and insert new hardware technologies into operational systems without putting the system's missions at risk. One approach is to perform a considerable amount of ground-based testing and analysis. Another more conservative approach leverages ground-based testing but also demonstrates the hardware on orbit in noncritical systems (as experiments or otherwise) to provide flight heritage for the new hardware. Our approach is an example of the latter, combining both ground test data and the data from actually flying hardware to demonstrate the utility of new technology on orbit.

The main goal of the Technology Readiness Level (TRL) verification application is to demonstrate that the new

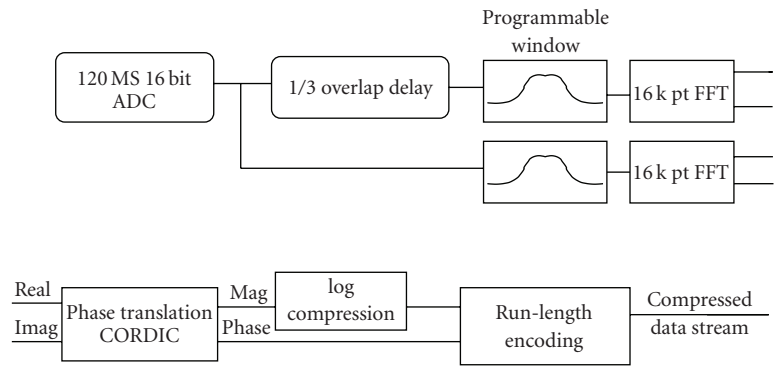

Figure 13: Application architecture for Gabor transform-based wideband compression system.

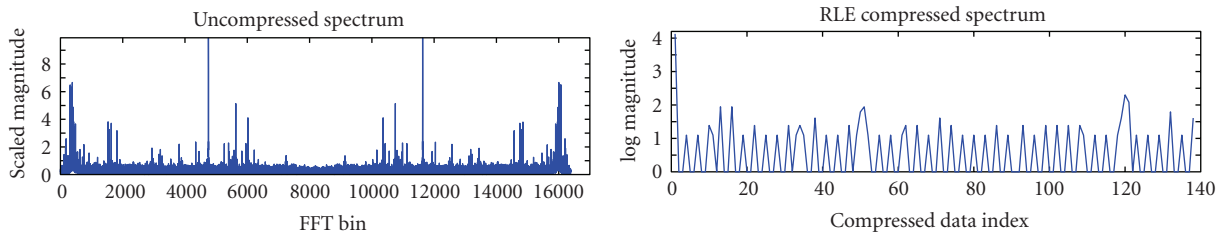

Figure 14: (a) Output from 16 k FFT of data captured from the FORTE satellite. (b) The same data compressed using the Gabor compression scheme with Run-length Encoding.

devices being flown in this system are useful for other government space missions. More specifically, we want to show that the Xilinx Virtex-4 FPGAs, the BAE C-RAM, the Atmel LEON-based microprocessor, QDR-SRAMs, and other components should be considered for future missions. As well, we will demonstrate how they can be used by mitigating risk. Since a significant amount of ground-based radiation testing has gone into most of the new parts used, probably the most challenging issue is to demonstrate that these components can operate reliably as components in a larger system and how to accomplish this. For the Xilinx FPGAs and most of the other new parts, this involves demonstrating that user applications can operate reliably in the presence of SEUs since the Total Ionizing Dose (TID) and single-event latchup characteristics of these devices have been demonstrated through ground-based radiation testing.

Our primary parts test routine is an FFT, with choice of on-board digital sine wave test sequence, or actual antenna input. Qualification test data showing actual dynamic range achieved by the T-SDR with 10 MHz input to the RF port is shown in Figure 12.

6.2. General-Purpose Recorder and Receiver. While the FPGA-based RTSP can implement any functionality, in the satellite module it is well suited to implementing a SoftwareDefined Radio and RF capture functionality. The system provides a variety of modes for capture of radio signals. Radio data can be captured in full-bandwidth snapshots directly off the Analog-to-Digital Converter (ADC), or from several taps, including subband tuned outputs and demodulated outputs.

While performing experiments scientists are often required to sift through large amounts of uninteresting data waiting for a brief event. The data stream is constantly monitored and a snapshot recorder is activated when the trigger conditions are met. For instance, a basic ADC level trigger checks for high RF signal levels at a very coarse level. A more sensitive trigger is implemented using a frequency trigger that detects energy in desired frequency bands. The frequency trigger is very effective as it is able to detect and activate on small signals far below the noise floor.

In addition to these data capture triggers based on simple energy presence, the T-SDR can also trigger based on communications parameters such as turbo decoder fram acquisition or demodulator lock.

6.3. Wideband Gabor Compression Application. For our third application, we looked for ways to pack more raw RF signal into a smaller digital bandwidth. The general term for this problem is data compression and expansion, or "companding." Application 3 relies on a "lossy" companding algorithm, which is suited to the transmission of RF signals as long as certain minimum fidelity requirements are met. LANL patented such a method several years ago, and now is actively exploring the technology [20].

Our scheme is built on the realization that most RF signals are modulated as the mathematical product of the information and an underlying carrier, where the strength of the received signal can vary dramatically. Thus, the great

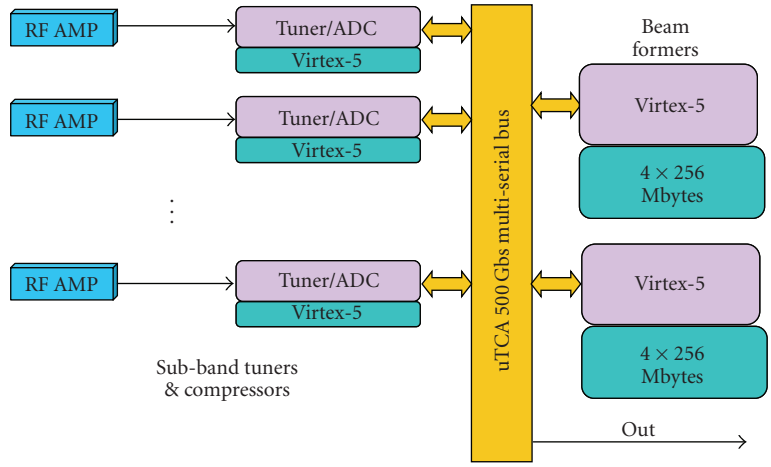

Figure 15: Multiple-Input Multiple-Output (MIMO) architecture for Virtex-5 FPGA-based beamformer.

majority of practical sensor applications are multiplicative modulation processes, which only require constant relative precision. This implies that logarithmic magnitude representations may be used without loss of resolution, as long as they are properly scaled. Few assumptions can be made about phase, requiring more information to be kept. The basic block diagram of Application 3 is illustrated in Figure 13.

Given the above assumptions, our process consists of applying a logarithmic transform to the output of a Fourier transform. The purpose of the frequency transform is to convert convolution processes to multiplication in frequency space, and to separate various signals that overlap in the time series, but which are separate in the frequency space. Use of the log transform then allows the multiplicative signals still contained in each frequency sample to be treated as additive superpositions.

The 120 Mega-Sample/second input data is subjected to a 1/3 overlap delay (1/3 of the FFT size of 16 k samples) to produce two data streams. This allows the output to completely cover the input signal after the windowing function is applied. Without the overlap delay, signals occurring at the edge of the transform might be masked and inadvertently destroyed. The two signals are then fed through a window multiplier to reduce spectral leakage and to allow near-perfect reconstruction. The outputs of the windowing operator are suitably delayed such that both signals can be presented to the FFT simultaneously. This is necessary because the system uses a single FFT to compute the transform of both streams, reducing the area and power requirements for the large 16 k point FFT. Because the inputs to the FFT are real, the output of the FFT is separable into two unique frequency transforms.

The output of the FFT is then fed into a rectangular/polar CORDIC transform. This is a pipelined core that converts the complex FFT output into corresponding polar representation consisting of magnitude and phase, which is the natural domain to either compress or to interpret man-made signals. At this point the magnitude enters a lookup table, providing Mu Law scaling, which is linear at low amplitudes and logarithmic at larger amplitudes, providing a balance between range and precision. At present we do not process the phase, but methods of removing phase trend information and compressing it are very promising, and under study.

Finally the data enters a run length coder, with variable threshold, so that only magnitude values over a chosen level are accepted. For the discarded values below threshold, the coder discards both magnitude and phase, and substitutes a sequence length code instead. In this way, larger or smaller data compaction ratios are achieved, depending on the degree of fidelity required. Figure 14 illustrates the effectiveness of the scheme with data collected on FORTE, another LANL satellite. A sample size of 16 k real points is reduced to 140 points in log-polar representation (with magnitude and phase). In this scheme, low energy frequency bins are discarded and the length of runs of discarded bins are recorded.

7. Summary and Work in Progress

Even as the receiver was developed, new COTS hardware appeared that dramatically changed architectural possibilities again. The two most significant developments for LANL were the following.

(i) The advent of Multi-Gigabit Serial cores in FPGAs, DSPs, and microprocessors. These hard IP blocks enable all-serial modular systems.

(ii) The development of silicon-on-insulator ADC such as the TI5424, which offer very strong Rad Tolerance while providing input bandwidths exceeding a GHz and sample rates > 400 Msps. This enables continuous band coverage from Direct Conversion architectures.

Together, these key evolutions led us to consider multichannel SDR architectures for MIMO and other spatial diversity applications. One such architecture now under active pursuit at LANL is shown in Figure 15, which we call NextRF. A TI ADS5474 ADC was chosen, and operates at 400 Msps, followed by a Xilinx Virtex-5 SX95T processor, creating an SDR with almost seamless coverage from DC to 500 MHz.

Test results are expected from NextRF by 2010. Our simulations indicate that the V5SX95 with the 16 k FFT and CORDIC cores described above can operate at 400 MHz, and occupancy is about 18% of the device. In this partitioning scheme, the compressor runs within the receiver head, and delivers a serial stream of digital data to postprocessors which combine and further reduce the signal size into a final output datastream.

Acknowledgments

The LANL team gratefully acknowledges long-term support from DOE NA-22, DARPA, and other DOD agencies in development of all our SDR work. Only through such long-term support can a new capability be fully developed. The authors also recognize the contributions of our commercial partners, Xilinx, Lockheed-Martin, BAE Systems, Linear Technology, and Texas Instruments. This document is released under LA-UR 10-06455.

References

[1] P. S. Graham, M. Caffrey, M. J. Wirthlin, and D. E. J. N. H. Rollins, "Reconfigurable computing in space: from current technology to reconfigurable systems-on-a-chip," in *Proceedings of the IEEE Aerospace Conference*, pp. 2399–2410, 2003.

[2] Published specifications of the BAE Rad750 PPC (50 Mips/Watt) device, or the Atmel AT697E (100Mips/Watt, versus measured Virtex-4 performance (50 GOps/Watt) at several facilities).

[3] H. Quinn and P. Graham, "Terrestrial-based radiation upsets: a cautionary tale," in *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '05)*, pp. 193–202, April 2005.

[4] H. Quinn, P. Graham, and B. Pratt, "An automated approach to estimating hardness assurance issues in triple-modular redundancy circuits in xilinx FPGAs," *IEEE Transactions on Nuclear Science*, vol. 55, no. 6, pp. 3070–3076, 2008.

[5] H. Quinn, P. Graham, J. Krone, M. Caffrey, and S. Rezgui, "Radiation-induced multi-bit upsets in SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2455–2461, 2005.

[6] R. Crochiere and C. Rabiner, *Multi-Rate Digital Signal Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 1983.

[7] CCIR Reports 341-6, 258-5, & 670-1, International Telecommunication Union, Geneva, Switzerland.

[8] S. Dhawana, O. Bakera, R. Khannad, et al., "Radiation resistant dc-dc power conversion with voltage ratios > 10 capable of operating in high magnetic field for lhc upgrade detectors," in *Proceedings of the Topical Workshop on Electronics for Particle Physics*, 2008.

[9] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 2147–2157, 2003.

[10] P. S. Ostler, M. P. Caffrey, D. S. Gibelyou,, et al., "SRAM FPGA reliability analysis for harsh radiation environments," *IEEE Transactions on Nuclear Science*, vol. 56, no. 6, pp. 3519–3526, 2009.

[11] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-induced persistent error propagation in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2438–2445, 2005.

[12] "Open On-Chip Debugger: Free and Open On-Chip Debugging, In-System Programming and Boundary-Scan Testing," 2008, http://openocd.berlios.de/web/.

[13] "URJTAG: Universal JTAG library, server and tools," 2008, http://www.urjtag.org/book/index.html.

[14] "Xilinx UG071 Virtex-4 Configuration Guide," 2006, http://www.xilinx.com/support/documentation/user_guides/ug070.pdf.

[15] "PicoBlaze User Resources," 2008, http://www.xilinx.com/ipcenter/processor central/picoblaze/picoblaze user resources .htm.

[16] "PIC18F2455/2550/4455/4550 Product Family," 2007, http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=dDocName=en010280.

[17] "Xilinx In-System Programming Using an Embedded Microcontroller," 2007, http://www.xilinx.com/support/documentation/application_notes/xapp058.pdf.

[18] "Using the JTAG Interface as a General-Purpose Communication Port," 2005, http://www.xilinx.com/support/support/documentation/user_guides/ug070.pdf.

[19] "Xilinx Chipscope Pro Tool," 2009, http://www.xilinx.com/ise/optionalprod/cspro.htm.

[20] M. Dunham, "Logarithmic compression methods for spectral data," US Patent no. 6,529,927, 2003.