# Heterogeneous High Performance Computer Emulation of a Space Based Radar On-board Processor

Assem Salama
*ITT*
assem@twcny.rr.com

Richard Linderman, John Rooks, and Allison Leider
*Air Force Research Laboratory, Information Directorate (AFRL/IF), Rome, NY*
{Richard.Linderman, john.rooks, allison.leider}@rl.af.mil

## Abstract

*This paper presents the successful emulation of an on-board processor (OBP) to support Space Based Radar (SBR). The emulation is demonstrated on the forty-eight node dual Xeon Heterogeneous High Performance Computer (HHPC) operated by the Air Force Research Laboratory (AFRL) located in Rome, New York. Each node in the HHPC supports one Annapolis Wildstar II board, composed of 2 Xilinx Virtex II 6 Million gate Field Programmable Gate Arrays (FPGAs).*

*As system complexity increases, debugging the software of tera-scale systems with hundreds to thousands of processors is poorly supported by time consuming simulations. However, the advent of large FPGAs allows a powerful new tool to assist in the architecture development effort—emulation. For the case at hand, the 96 FPGAs of the HHPC are capable of emulating at 8% of the actual system clock speed (20 MHz of 250 MHz) and close to 15% of the 2560 individual processors of the proposed SBR system. Even at this reduced scale, this emulation provides a testing environment roughly a million times more capable than HPC-based simulation for early software bug detection and correction. Further, this framework allows for experimenting with architecture enhancements and changes, and ultimately will ensure a low cost, reliable, fully reprogrammable product produced without re-spins.*

*The embedded system architecture of this SBR OBP is based on AFRL's dual processor, power efficient, programmable, Wafer Scale Signal Processor (WSSP). Target tracking and discrimination algorithms were developed and demonstrated on an earlier 96 processor embodiment of this architecture. For SBR, the algorithm set is being extended to include Synthetic Aperture Radar (SAR) image formation and Moving Target Indication (MTI) algorithms.*

## 1. Introduction

As the system complexity of major defense acquisition programs grows from tens to thousands of processors and thousands to millions of lines of software source code, new tools are required to successfully engineer the hardware and software of these systems under cost and schedule constraints. Reviews of several cases where DoD programs are struggling to deal with this complexity surface some common findings. These systems are often a complex web of intercommunicating processors and processes, separately developed by a host of subcontractors. Race conditions and state machine lockups among the components are a common malady. Often the subtle bugs in these interface designs are not exposed by the limited test sets focusing on the individual "boxes" of functionality. Simulation, even on HPCs, is too slow to run the software long enough to surface the problems. Since no development tool caught these flaws, they are passed through to the implemented design causing havoc at system integration and test. At this point, the ability to properly instrument the design to catch the problems is greatly reduced, leading to long timelines for debug and diagnosis.

High Performance Computing coupled with parallel software developments has long been viewed as a tool to speed up simulation. Conversely, the reconfigurable logic of FPGAs is known to accelerate a host of integer "number crunching" applications, primarily for signal and image processing. However, when merged in the context of a heterogeneous HPC, powerful new non-numerical capabilities arise, in this case the ability to emulate and instrument complex systems under development early in the design process to allow concurrent hardware and software development and facilitate system debugging. The FPGAs contribute the malleable logic to synthesize the proposed hardware design; the HPC provides the memory and I/O infrastructure to support the interaction of these computational elements. This emulation infrastructure provides a testing environment not only for early bug detection and correction, but also a framework for experimenting with architecture enhancements and extra instrumentation, and ultimately ensures a reliable product is fabricated without costly and time consuming re-spins.

The embedded system architecture of this SBR OBP is based on AFRL's dual processor, power efficient, programmable, Wafer Scale Signal Processor (WSSP). In an earlier 0.5 micron CMOS technology, this processing element achieved a peak power efficiency of over 900 MFLOPS/Watt[1,2]. This attribute allowed it to be packaged in dense Multi-Chip Modules (MCMs)[3] without overheating. The power efficiency is to be further improved to over 10 GFLOPS/Watt by scaling to 0.13um CMOS which allows eight dual processors to occupy the same chip. This power efficiency is critical to enabling teraflop scale systems for challenging space system applications like SBR where only a few hundred watts of power are available for the HPC.

This paper is organized as follows. First, summaries of the HHPC, FPGAs, and WSSP are offered. Next the different levels of processor modeling and simulation are discussed and contrasted. Then the emulation of the WSSP on the FPGAs is discussed. Finally, there is a discussion of future work and conclusions.

## 2. Heterogeneous High Performance Computer

The AFRL Information Directorate High Performance Computing Facility, funded by the Department of Defense High Performance Computer Modernization Program (HPCMP), supports the 48-node Heterogeneous High Performance Computer (HHPC) at the Rome Research Site (RRS), located in Rome, NY. This one million dollar computing asset was made available February 2003 to support the entire DoD community and enhance warfighter capabilities especially in the areas of Command, Control, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR).

The 2-chassis, 48-node HHPC combines the benefits of cluster computing with the reconfigurability of FPGAs. The dual 2.2 GHz Intel Xeon processors on each node of the HHPC combine for a peak system throughput of 845 GFLOPS IEEE single precision floating point and 422 GFLOPS double precision. Each node has 4 GB of SDRAM, 80 GB of disk, a fiber optic MYRINET interface complemented with a gigabit Ethernet interface. The unique aspect of the cluster is the Annapolis Microsystems Wildstar II FPGA board. Each board supports two Xilinx Virtex II 6 million gate FPGAs (see Figure 1).

At the time the system was proposed, it was widely known that FPGAs could offer tremendous acceleration of integer number crunching to HPCs, hence the "heterogeneous" mix of the proposed architecture. As a reflection of this, the acceptance tests for the machine's delivery included a demonstration of sustained performance over 34 Tera-OPs per second while performing signal processing filtering operations[6]. However, it was conjectured that the HHPC could offer similar acceleration to non-numerical challenges. In this case, the HHPC has been proven to greatly accelerate system emulation.

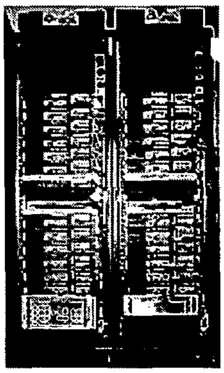


**Figure 1. Heterogeneous high performance computer**

## 3. Field Programmable Gate Arrays

Rapid prototyping in FPGAs can significantly reduce the time and cost of a system by ensuring a thoroughly tested ASIC will be produced from the first fabrication. The processor emulation runs approximately one million times faster than the VHDL based simulation, enabling a greater probability that rarely occurring bugs, often hard or impossible to find using VHDL simulation, will be detected. Additionally, reconfiguring an FPGA with a new design takes hours, rather than months for a re-spin of an ASIC. By emulating a significant portion of the SBR OBP on the FPGA-centric HHPC, different architecture tradeoffs can be explored, bugs in software and hardware can be identified early in the design cycle, and ultimately a fully reprogrammable, reliable product can be produced in a comparatively short time frame.

An FPGA is a reconfigurable hardware architecture of millions of gates and dedicated on-chip resources interconnected as needed to optimally configure for a given application. For this emulation, the Very High Speed Integrated Circuit (VHSIC) Hardware Definition Language (VHDL) was used to describe the structure and behavior of the processor under design. For simulation and synthesis, the ModelSim and Synplify Pro tools were used, respectively. Xilinx place and route tools were used to automatically route the design in the target hardware. The design flow is noted in Figure 2.

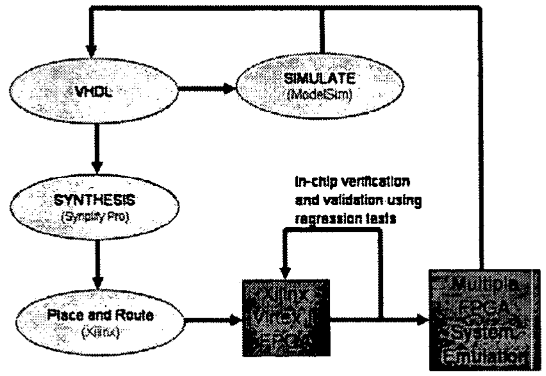


**Figure 2. Architecture design flow**

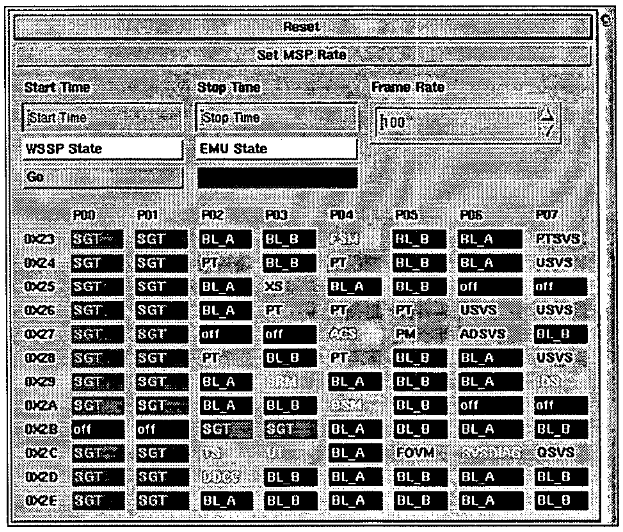
## 4. Wafer Scale Signal Processor

The 0.5um CMOS WSSP design was completed by AFRL's Information Directorate in 1999. The primary goal was extreme power efficiency on floating point intensive operations and ease of programming for real-time embedded systems. Ultimately, this processor achieved a power efficiency over 900 MFLOPS/Watt. A 96 processor embedded HPC was demonstrated running 250K lines of real-time signal/image processing code.

The well-known GNU toolset provides the compilers (C, C++, Fortran), assembler and debugger for the processor[4], permitting cross-platform development on traditional PCs. The operating system is the US Army developed Real-Time Executive for Military Systems (RTEMS). RTEMS provides a real-time multi-tasking operating system with open source code[5]. A simulator/communicator is available for PCs running LINUX and Sun Workstations running Solaris. This tool provides visibility into the processor operations and assists in communications with the processors for code debugging. The simulator/communicator allows a mixture of actual nodes and simulated nodes, which is helpful when debugging suspect processes where some can be "placed under the microscope" of enhanced observability that the simulator provides. The tool also boots the processor elements and handles standard input/output functions on behalf of the processing nodes.

Figure 3 illustrates a typical operator's console display. From the operator's console, the status of the 96 floating-point processors is graphically displayed. The screen capture in Figure 3 shows the existing 96 processor system as it is booting. The elements start out with "off" written in their block, next they get an RTEMS base load, followed by one or more application programs, finally when they get connected to their real time distributed data base the block turns green. By clicking on any processor's block its text messages can be displayed. If a warning or error occurs the block will turn yellow or red.

A strong point of the WSSP effort was the open source for all hardware and software aspects. This greatly reduced our system integration time by not having the hurdles of hidden source code.



| Start Time | Stop Time | Frame Rate |
|---|---|---|
| Start Time | Stop Time | 100 |
| WSSP State | EMU State | |

| | P00 | P01 | P02 | P03 | P04 | P05 | P06 | P07 |
|---|---|---|---|---|---|---|---|---|
| 0X23 | SGT | SGT | BL_A | BL_B | FSM | BL_B | BL_A | PTSVS |
| 0X24 | SGT | SGT | PT | BL_B | PT | BL_B | BL_A | USVS |
| 0X25 | SGT | SGT | BL_A | XS | BL_A | BL_B | off | off |
| 0X26 | SGT | SGT | BL_A | PT | PT | PT | USVS | USVS |
| 0X27 | SGT | SGT | off | off | ACS | PM | ADSVS | BL_B |
| 0X28 | SGT | SGT | PT | BL_B | PT | BL_B | BL_A | USVS |
| 0X29 | SGT | SGT | BL_A | [illegible] | BL_A | BL_B | BL_A | IDS |
| 0X2A | SGT | SGT | BL_A | BL_B | DSM | BL_B | off | off |
| 0X2B | off | off | SGT | SGT | BL_A | BL_B | BL_A | BL_B |
| 0X2C | SGT | SGT | [illegible] | [illegible] | BL_A | FOVM | [illegible] | QSVS |
| 0X2D | SGT | SGT | DDCS | BL_B | BL_A | BL_B | BL_A | BL_B |
| 0X2E | SGT | SGT | BL_A | BL_B | BL_A | BL_B | BL_A | BL_B |

**Figure 3. Operators console for 96 processor configuration**

This design is undergoing modification for several new implementations. This paper will focus on an implementation by Boeing Satellite Systems (BSS) targeting Space Based Radar. Though the processor design is by no means limited to the SBR application, it does serve as an excellent challenge problem. SBR requires on the order of 1 trillion Floating Point Operations per Second (TFLOPS) and ease of programming.

The original 0.5 micron CMOS design was based on a custom layout containing two processors that could operate independently or as a self-checking pair[1]. In two successive Multi-Chip Module (MCM) implementations the processors were given 0.5 Mbytes of external SRAM each (MCM2) or 2 Mbytes of external SRAM each (MCM3). Each processor can perform two single precision floating point ALU operations per clock cycle or one double precision floating point ALU operation. In addition, two single precision/one double precision floating point multiplication can be performed each clock cycle. The instruction set is a standard RISC load/store style with 32 bit addressing similar to the MIPS architecture but augmented with vector, matrix, and DSP instructions such as FFTs and matrix multiplies. The floating point units achieve a high utilization rate for these optimized instructions. Another important feature of the WSSP is writeable VLIW instruction store, which allows new optimized instructions to be developed as needed and loaded from the program's code.

The BSS WSSP implementation will implement 16 processors (as 8 pairs) on a single IBM 0.13 um technology die. Each will have 1 Mbyte of embedded DRAM. The original processor had single cycle access SRAM. The new, enhanced WSSP will now have a multi-cycle access DRAM. However the data width of the DRAM is 256 bits approximately doubling the overall bandwidth to the WSSP local memory. To effectively use the DRAM a small cache is being added to the new design. This cache is expected to be small enough to allow predictable performance and minimize power consumption. Presently, cache sizes around 1 KB are planned, but this is an area where running software on top of emulated hardware will greatly assist the quantification of cache size tradeoffs.

With the 16 processors and memory there is an 11 port crossbar switch as depicted in Figure 4. Off-chip interfaces include two local ports for onboard communication to other similar chips with a ring bus configuration and one 10 Gbit Ethernet XAUI port which can be viewed as an uplink port. Each ASIC will also have a port to external DRAM. For the SBR case where component selection is more limited due to radiation tolerance requirements, this is set at 256 MB. This could be multiple gigabyte DIMMs in a terrestrial HPC application.

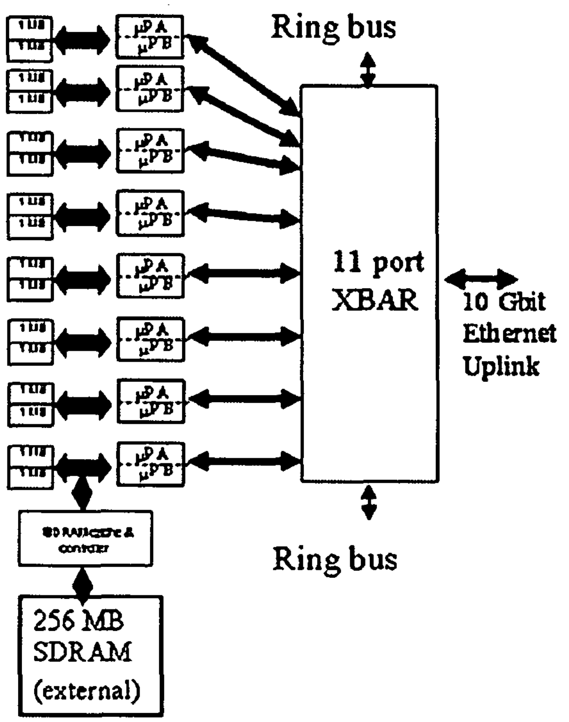
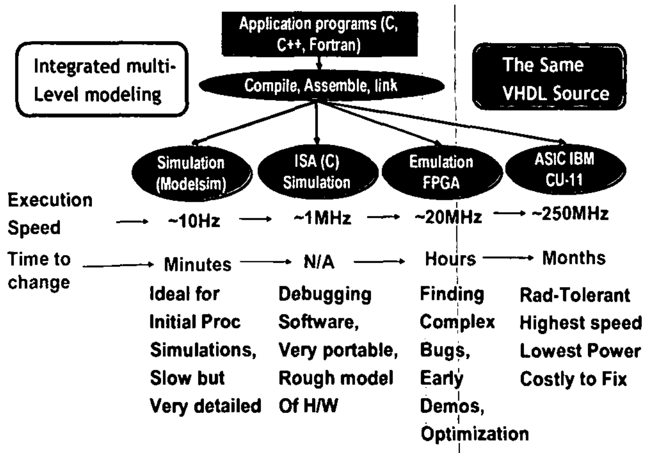


Figure 4. Multiprocessor chip with 11 port crossbar

## 5. Processor Models

The design process of the WSSP includes a complete set of interoperable models ranging from VHDL through the final hardware. Figure 5 summarizes the key features of each stage in the modeling process.



Application programs (C, C++, Fortran)
Integrated multi-Level modeling
Compile, Assemble, link
The Same VHDL Source
Simulation (Modelsim)
ISA (C) Simulation
Emulation FPGA
ASIC IBM CU-11
Execution Speed
~10Hz
~1MHz
~20MHz
~250MHz
Time to change
Minutes
N/A
Hours
Months
Ideal for Initial Proc Simulations, Slow but Very detailed
Debugging Software, Very portable, Rough model Of H/W
Finding Complex Bugs, Early Demos, Optimization
Rad-Tolerant Highest speed Lowest Power Costly to Fix

Figure 5. Design process

VHDL models exist for the processor, board, and system. The VHDL models are best suited for low level processor design and optimization. Typically the VHDL runs at about 10 clock cycles per second. The slow speed prevents running an operating system and applications. However, this level of modeling and simulation is very useful for testing the processor instruction set, including the development of new application specific instructions. At this level a complete set of regression tests are developed to check for architecture consistency across the numerous revisions and later use in ASIC chip testing. Each regression test is designed to verify a particular function and check the answer. Incorrect answers result in the processor taking a software trap. An automated post run analysis identifies any regression test that failed.

For software debugging an Instruction Set Architecture (ISA) simulator is used. This program gives insight into the processor's status. Although, the ISA simulator is clock cycle accurate, the simulation does not verify the accuracy of the VHDL that will ultimately be used to synthesize the final ASIC. The ISA simulator runs at about one million clock cycles per second. This speed is sufficiently fast to test and debug most software and is often the preferred debugging tool for programmers due to the great observability into the internal state of the processing elements that the simulator provides.

Emulation of the processors permits the verification of the hardware, system software, and applications programs in an overall system context. The final emulation is expected to run at about 20 million clock cycles per second. Presently, the emulator is running at 6 MHz. At 8% of the final hardware speed, the emulation will be fast enough to find complex state machine interactions that are otherwise extremely difficult to find using the slower VHDL or ISA simulations. In addition, since the emulation is VHDL-based, it will catch some bugs which elude the ISA simulator.

In addition to hardware assisted debugging, FPGA-based emulation offers several other advantages. Instrumentation hardware that may not be desirable in the final system can be added to extract or inject any desired information. For example, performance gathering hardware can monitor the cache performance. A critical item for space applications is how errors are handled. Space based electronics are susceptible to Single Event Upsets (SEUs) or bit flips. Gates can be added to the FPGA emulation allowing SEUs to be injected under control of the host processor. This allows more thorough testing of the hardware and software response to an SEU than using slower, less efficient VHDL simulations running application programs.

## 6. Processor Emulation

Emulation of the processor was performed on the HHPC which consists of 48 nodes each having a Wildstar II FPGA board. Each board has two 6 million gate Xilinx Virtex II FPGAs. In order to reduce the risk of having bugs in the final ASIC, the synthesized VHDL must match exactly what will be on the ASIC.

The environment that surrounds the FPGA, (i.e., memory, bus interfaces etc.) is not exactly same as the environment surrounding the ASIC. Wrappers are added to make the environments look the same. As an example, the memory on the Wildstar board is DDR II SRAM which is different than what the ASIC will have. So, a "converter" had to be written in VHDL to make DDRII SRAM function like the embedded DRAM available in the SBR implementation.

A critical entity in any HPC design is the message passing interface. Rare state machine interactions can make it difficult to identify all bugs. The processor itself can be tested quite thoroughly using ModelSim. However, the message passing interfaces are limited by simulation speed. One of the major obstacles to fully testing the message passing interfaces is running a broad diversity of codes over extended periods of time (hours to days), some representing the actual system context, some contrived, to surface problems. The commercial VHDL simulation environments are not parallelized on HPCs, so they performance degrades as processing elements are added to the simulated system. However, on the emulated system, the performance is not only a million fold faster, but parallelizable across the nodes of the HHPC.

Another technical challenge to emulation of the large scale system is hiding the difference in the interconnection fabric and the means of interfacing to the host processor. The cluster has gigabit Ethernet and MYRINET internodal connectivity. The emulators' communications use the gigabit Ethernet since it is roughly one tenth the targeted 10 gigabit Ethernet fabric and the emulated clock rate is also approximately one tenth the final system speed. The host interface of the Wildstar board is connected through a local bus on the board to the PCI bus then to the dual Xeons of the node. The headnode of the HHPC cluster runs the simulator/communicator which in turn controls slaved simulators running on a Xeon processor in each nodes of the HHPC. These in turn monitor and provide I/O services to the emulated processors on the Wildstar II boards.

On each FPGA, we presently can emulate two processors including their I/O and memory interfaces. Through optimization and refinement, the objective is to fit 4 emulated processors within each FPGA. Each processor has 2 MB of SRAM configured to mimic the embedded DRAM of the target design. Because each board has 2 FPGAs, we can actually emulate 4 processors on one board including their I/O and memories. Presently, the FPGA resources are 75% utilized when two processors are emulated. Most of the space is given to the double precision multiplier. Each processor has one double precision multiplier which takes up roughly 15% of the FPGA. So, the double precision multipliers take up about 30% of the FPGA. If the double precision multipliers were taken out, the utilization of the FPGA would be around 45%. This is a near term path to allow 4 processors to fit on one FPGA and 8 on one board, resulting in a total of 384 processors on the entire HHPC. For many applications and stress tests this would be the preferred configuration. However, for a full scale test of the system, with all its interacting software processes, more FPGA gates, likely in the form of more, larger FPGAs on more nodes, are required. Some percentage of bugs will only be discovered with such a full scale test.

## 7. ISA Simulator/Communicator

The ISA simulator can be executed in two different modes, simulator or communicator. In either mode, there are many common functions that help support the processor.

In order for a processor to execute code, it must be "booted up." This process involves writing to the processor's memory a known memory image that contains the executable. After this is done, the processor must be taken out of reset and its clock turned on. Communication in and out of the processor is presently achieved using an I/O bus. An external source, here the cluster's Xeon headnode, writes to memory and take a processor out of reset and turn on its clock. This functionality is used to boot up the processor.

In either mode, the ISA simulator/communicator must boot up an emulated processor. Depending on whether or not the processor exists, the simulator/communicator can boot up a simulated processor or an emulated processor. The simulator/communicator must know how to communicate to the other processors. In the older design, MYRINET was used. The simulator uses the PCI bus/LAD bus with gigabit Ethernet with the new emulation.

The ISA simulator writes to memory and achieves all of its goals by sending messages and waiting for messages, hence its "communicator" aspect. After the processor is booted, the simulator waits for messages from the processor and handles any acknowledges or resends as required.

In order to make the programming and debugging of the processor easier, the standard I/O library has been written to support the architecture. If a printf is called, a special call goes to the simulator/communicator for servicing the request form the embedded node. Files can be opened on the headnode that is running the simulator/communicator and the emulated and simulated processors can read and write to these files using messages.

Message passing between different simulators is also available allowing for booting of different processors and different nodes to have communication. When the simulator/communicator sees a message that needs to be forwarded, it will forward it to the appropriate simulator. Not all of the processors have to exist. A real processor can send to a simulated processor that is on another node using the same principle. This allows us to easily add additional processors to a system by simulating more.

## 8. Future Work

The HHPC presently allows a meaningful portion of the SBR architecture to be emulated at a fraction of the eventual system clock rate. However, FPGA technology has continued to advance rapidly, such that in a next generation HHPC with 10–20M gates per FPGA and with additional nodes, the entire SBR architecture can be emulated. In addition, we expect the clock rates to continue to improve into the 30–50 MHz regime.

Further work will also be performed to optimize the emulations footprint in the FPGA by analyzing the critical portions of the design consuming the majority of FPGA resources.

Further work will pursue flexible means of injecting additional instrumentation into the design to enhance observability and controllability.

The power efficiency of the architecture is a driving requirement for space systems, but also has benefits for terrestrial HPC centers as they struggle to deal with practical megawatt limits. These spin-off applications of power efficient architectures are still to be explored.

Finally, other non-numerical applications of FPGAs will be explored, such as pub-sub brokering of information objects, as discussed in a companion presentation at this HPC Users Group meeting [7].

## 9. Conclusions

When cluster HPCs are augmented with reconfigurable FPGA-based capabilities the resulting heterogeneous computing environment is capable of emulating architectures under development. This allows for concurrent hardware and software development and enhances instrumentation capabilities to assist in system debugging.

By focusing on power efficient, yet fully programmable architectures with IEEE standard floating point arithmetic support the advances of the HPC software community can be leveraged to meet the challenges of space based systems with flexibility and scalability.

## References

1. Linderman, R., et al., "A dependable high performance wafer scale architecture for embedded signal processing." *IEEE Transactions on Computers*, 47(1), January 1998.

2. Rooks, J., J. Lyke, and R. Linderman, "Wafer Scale Signal Processors and Reconfigurable Processors in a 3-Dimensional Package." *GOMAC Digest of Papers*, 2002.

3. Linderman, R., M. Linderman, R. Kohler, Maj. J. Comtois, and J. Sabatini, "HDI Design of a 100 MFLOPS/Watt Floating Point DSP." *GOMAC Digest of Papers*, 1998.

4. GNU Operating System – Free Software Foundation. Free Software Foundation, Copyright 1996, http://www.gnu.org..

5. RTEMS. OAR Corporation, http:// www.rtems.org.

6. AFRL/IF Distributed Center, Air Force Research Laboratory Information Directorate, http://www.rl.af.mil/tech/facilities/HPC/Hardware.html.

7. Linderman, R., C.S. Lin, and M. Linderman, "Heterogeneous HPC Acceleration of JBI Pub-Sub Brokering." 2004 DoD HPCMP Users Group Conference, 10 June, 2004.