SpaceCubeX: A Framework for Evaluating Hybrid Multi-Core CPU/FPGA/DSP Architectures

Andrew G. Schmidt[1], Gabriel Weisz[1], Matthew French[1], Thomas Flatley[2], and Carlos Y. Villalpando[3]
[1]Information Sciences Institute, University of Southern California
[2]Science Data Processing Branch, NASA Goddard Space Flight Center
[3]Jet Propulsion Laboratory, California Institute of Technology
{*aschmidt, gweisz, mfrench*}*@isi.edu, thomas.p.flatley@nasa.gov, carlos.y.villalpando@jpl.nasa.gov*

***Abstract*—The SpaceCubeX project is motivated by the need for high performance, modular, and scalable on-board processing to help scientists answer critical 21st century questions about global climate change, air quality, ocean health, and ecosystem dynamics, while adding new capabilities such as low-latency data products for extreme event warnings. These goals translate into on-board processing throughput requirements that are on the order of 100-1,000× more than those of previous Earth Science missions for standard processing, compression, storage, and downlink operations. To study possible future architectures to achieve these performance requirements, the SpaceCubeX project provides an evolvable testbed and framework that enables a focused design space exploration of candidate hybrid CPU/FPGA/DSP processing architectures. The framework includes ArchGen, an architecture generator tool populated with candidate architecture components, performance models, and IP cores, that allows an end user to specify the type, number, and connectivity of a hybrid architecture. The framework requires minimal extensions to integrate new processors, such as the anticipated High Performance Spaceflight Computer (HPSC), reducing time to initiate benchmarking by months. To evaluate the framework, we leverage a wide suite of high performance embedded computing benchmarks and Earth science scenarios to ensure robust architecture characterization. We report on our projects Year 1 efforts and demonstrate the capabilities across four simulation testbed models, a baseline SpaceCube 2.0 system, a dual ARM A9 processor system, a hybrid quad ARM A53 and FPGA system, and a hybrid quad ARM A53 and DSP system.**

TABLE OF CONTENTS

1. INTRODUCTION

The goal of the SpaceCubeX project is to develop a high performance on-board computing architecture which enables next generation Earth science missions by effectively leveraging heterogeneous COTS processors. Next generation missions are invoking sensors with ever increasing data rates and higher fidelities while at the same time target global and persistent data collections. These goals translate into on-board processing throughput requirements that are on the order of 100-1,000× more than previous Earth Science missions for standard processing, compression, storage, and downlink operations. Low-latency data products and autonomous operations drive computational load even further. Recent studies have shown that in order to realize mission size, weight, area, and power (SWAP) constraints while meeting inter-mission reusability goals, compact heterogeneous processing architectures will be needed [1,2].

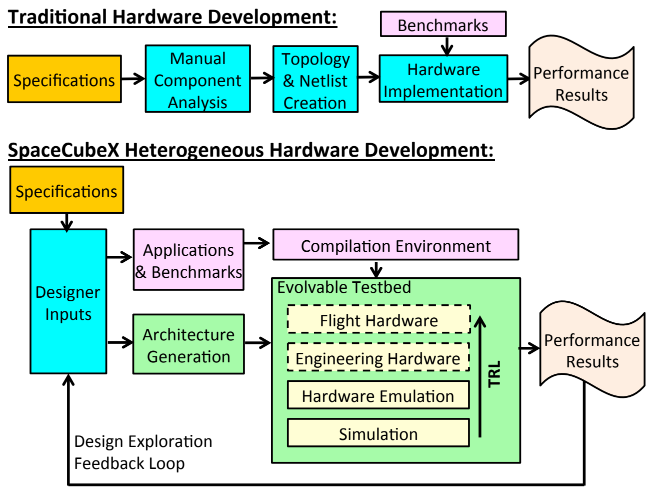


Figure 1. Process flow for traditional and heterogeneous hardware development

In a heterogeneous architecture, general OS support, high level functions, and coarse grained application parallelism are efficiently implemented on multi-core processors, while a co-processor provides mass acceleration of high throughput, fine-grained data parallelism operations. The SpaceCubeX project will find the most efficient and robust selection of multi-core CPU and co-processor accelerator resources for an on-board computing architecture specific to Earth science missions. Heterogeneous architecture development represents a significant departure from traditional homogeneous avionics development as it presents several new challenges. First, the architecture search space is no longer humanly tractable as the complexity increases from O(N), driven by the number of processor types in the homogeneous case, to O(N^2) in the heterogeneous case, driven by the number of processor and interconnect combinations. To maximize reuse, the hybrid architecture must perform at a high level across a robust suite of applications representative of a wide array of Earth science missions. The inter-processor physical connections, board topology, and programming model must work in concert to realize a low-overhead, scalable system that is transparently programmed. Finally, the architecture, including firmware and programming interfaces, must be portable across maturation from breadboard to engineering units to flight units.

The SpaceCubeX project addresses these challenges by creating a tool set and evolvable testbed which enable developers to perform focused design space exploration of candidate processing architectures. This technology fundamentally changes avionics processing hardware development from a waterfall approach to an iterative feedback model, facilitating a repeatable, extensible scientific exploration of the architecture space, seen in Figure 1. We report on the first year developments as part of this effort and specifically highlight the technical accomplishments and analysis of the Simulation Testbed, tools, and benchmark applications used to evaluate the preliminary system architectures. These architectures include a baseline SpaceCube 2.0 architecture, an ARM A9 Xilinx Zynq based single chip architecture, a Hybrid ARM A53 and FPGA architecture, and a Hybrid ARM A53 and DSP architecture. Evaluation and analysis cover diagnostic, micro-benchmarks, and full hybrid applications covering single, multi-core and co-processor accelerator experimental results.

The paper is organized as follows. In Section 2 the background and related works are presented. Section 3 covers the Design and Implementation details of the SpaceCubeX developed evolvable testbed technology and the initial architectures studied as part of the year 1 efforts. The evaluation and analysis of the results are presented in Section 4, followed by the conclusion and future works in Section 5.

2. Background and Related Work

This research addresses NASA's Earth Science missions and climate architecture plan and its underlying needs for high performance, modular, and scalable on-board processing. The decadal survey era missions are oriented not only to provide consistent observations with the previous generation of missions, but also to provide data to help scientists answer critical 21st century questions about global climate change, air quality, ocean health, and ecosystem dynamics, while adding new capabilities such as low-latency data products for extreme event warnings. Missions such as (P)ACE, HyspIRI, GEO-CAPE, and ASCENDS are specifying instruments with significantly increased temporal, spatial, and frequency resolutions and moving to global, continuous observations [3]. These goals translate into on-board processing throughput requirements that are on the order of 100-1,000$\times$ more than previous Earth Science missions for standard processing, compression, storage, and downlink operations. We have developed SpaceCubeX: a Hybrid Multi-core CPU/FPGA/DSP Flight Architecture for Next Generation Earth Science Missions to address these needs and enable the next generation of NASA Earth Science missions to effectively meet their goals.

NASA has long used compact, heterogeneous processing architectures in concert with reconfigurable computing technologies in order to realize mission size, weight, area, and power (SWAP) constraints while meeting inter-mission reusability goals [4]. In a heterogeneous architecture, general OS support, high level functions, and coarse grained application parallelism are efficiently implemented on multi-core processors, while a co-processor provides mass acceleration of high throughput, fine-grained data parallelism operations, to achieve high performance robustly across many application types. Hybrid architecture development represents a significant departure from traditional homogeneous avionics development and SpaceCubeX provides a structured approach to fundamentally change the avionics processing architecture development process to yield the following benefits:

- Enables selection of the most SWAP efficient processing architecture, with direct impact on mission capabilities, cost, and risk.
- Minimal extensions to integrate new processors, such as the anticipated NASA High Performance Spaceflight Computer (HPSC), reducing time to initiate benchmarking by months.
- Reduces risk due to supply chain disruptions by allowing a user to rapidly compare alternative component selections, quantify the impact, and update the processing architecture.
- Leverages a wide suite of high performance embedded computing benchmarks and Earth science scenarios to ensure robust architecture characterization.
- Utilizes a proven inter-task programming model to facilitate benchmark compilation and experimentation, while being fully interoperable with commercial compilers.

Related Work

The commercial industry has understood the value of heterogeneous processing for some time now. High performance computing and server centers are demonstrating significant performance benefits by placing multi-core processors and FPGAs on adjacent sockets in the same server blade. Convey, Mitronics, and Dini Group are established vendors who supply these hybrid processing solutions for servers. Microsoft recently announced achieving 2x acceleration on web server applications using heterogeneous server blade hardware [5], which is a significant achievement given the substantial commercial investment to date in accelerating this application type. Intel recently announced plans to integrate the two processor types onto the same device using 3-D technology to further increase performance gains [6]. While SpaceCubeX can leverage the lessons learned from the HPC industry, these ground based solutions typically use hundreds of watts of power and do not address radiation mitigation concerns. The two primary FPGA vendors sell chips that integrate ARM processor cores and a reconfigurable fabric, but these chips were not designed with power reduction and radiation mitigation in mind [7, 8].

In the mobile computing domain the extremely small form factor and power requirements coupled with the high volumes has justified fabrication of heterogeneous system on a chip devices, such as the Qualcomm SnapDragon or the TI OMAP processors, which couple multi-core ARM processors with VLSI tuned to specific wireless protocols, codecs, and image processing used by the wireless industry. These solutions meet avionics power requirements, but the accelerated VLSI functions are too specific to the wireless industry and would not provide significant processing performance benefit to a wide range of Earth observing missions.

In the avionics area, work has been performed to try to normalize and compare avionics processor performance [9] however this work is focused on homogeneous, not heterogeneous computing. The closest realization of avionics heterogeneous processing architectures is in the NASA SpaceCube family of Field Programmable Gate Array (FPGA) based on-board science data processing systems developed at NASA GSFC [10]. SpaceCube is based on the Xilinx Virtex family of FPGA processors, which include embedded CPU, FPGA and DSP resources. These processing elements are leveraged to produce a hybrid science data processing platform that accelerates the execution of science algorithms by distributing computational functions among the elements, allowing each type of processor to do "what its good at." The SpaceCube 1.0 system has flown as part of the Relative Navigation Sensors (RNS) experiment during Hubble Servicing Mission 4 and the current Naval Research Lab (NRL) Materials on the

International Space Station Experiment 7 (MISSE7) payload. The original SpaceCube 1.0 system was based on Xilinx Virtex 4 FPGA technology, and new systems are currently being developed (SpaceCube 2.0 and SpaceCube Mini) using Xilinx Virtex 5 FPGA technology.

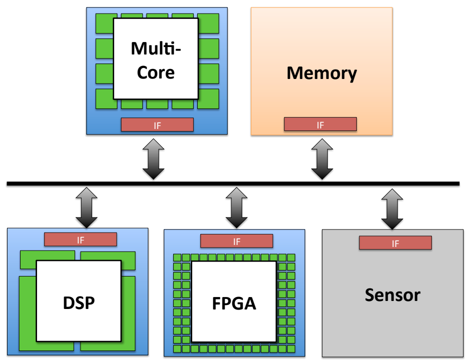
3. Design and Implementation

Heterogeneous on-board processing can realize drastic increases in processing performance while addressing processing avionics size, weight, power, and cost. The number of processors available to the aerospace industry is tractable, however the number of combinations of these elements and the number of different interconnect topology permutations, starts to make this process difficult and error prone for a human to do. The top-half of Figure 1 depicts the traditional waterfall hardware development process that has been used to date for homogeneous processing architectures. Here, the mission requirements are translated into processing specifications, which then guide the trade study and selection of the critical components, such as processor and memory types, and off-board I/O protocol. Using the data sheets from those components the additional details, such as board level topology, interconnects, boot up process are manually derived into a netlist. The netlist is then used to develop different maturation levels of hardware from emulation to engineering units to flight units. Processor level simulations can be used to predict the hardware performance with a high degree of confidence, and final performance results are verified with benchmarks executing on the final hardware.

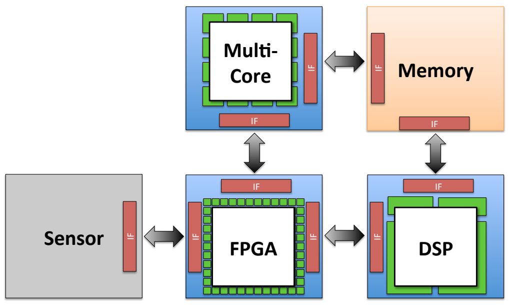
SpaceCubeX recognizes that heterogeneous architecture development adds significant complexity, such that the traditional hardware development process is no longer feasible. In SpaceCubeX the development process is updated, as shown in the bottom-half of Figure: 1, to enable an engineer to perform focused design space exploration to find the mixture of processors, topologies, interconnects, etc which best fit the mission needs by utilizing tools which eliminate manual fit analysis stages while also providing a common infrastructure which facilitates code reuse as an architecture matures from simulation to flight hardware. In this manner the designer can explore and compare several permutations. Here, the designer uses our developed Architecture Generation (ArchGen) tool to select the desired components and topology. The Architecture Generation tool configures an Evolvable Testbed, which uses a common infrastructure to control and monitor simulations (year 1) or hardware emulations (to be developed as part of year 2). The Evolvable Testbed is also extensible to support engineering or flight hardware developed in the future. To ensure the architecture generated supports Earth Science missions, the applications and benchmarks selected represent key processing challenges in the decadal survey missions. In a heterogeneous processing system, the application software must act in concert and leverage the benefits of the heterogeneous resources. To this end SpaceCubeX supports a Compilation Environment utilizing prior IP which provides a common API and infrastructure which allows a software developer to easily define communication interfaces between subtasks and to rapidly move subtasks across processor types. The performance results that will be collected with each architecture are discussed in Section 4.

Architecture Generation

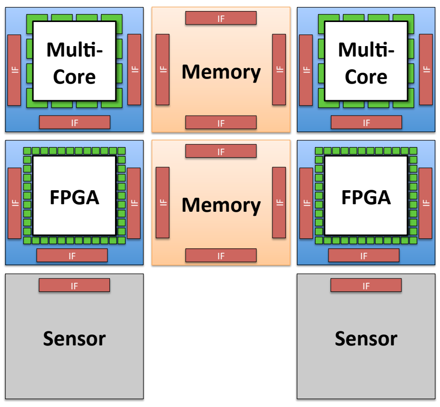
The Architecture Generation, ArchGen, tool allows a user to rapidly create a variety of boardlevel architectures by selecting component types and describing their on-chip and



(A) Bus Topology



(B) Ring Topology



(C) Mesh Topology

Figure 2. Conceptual diagrams of ArchGen systems

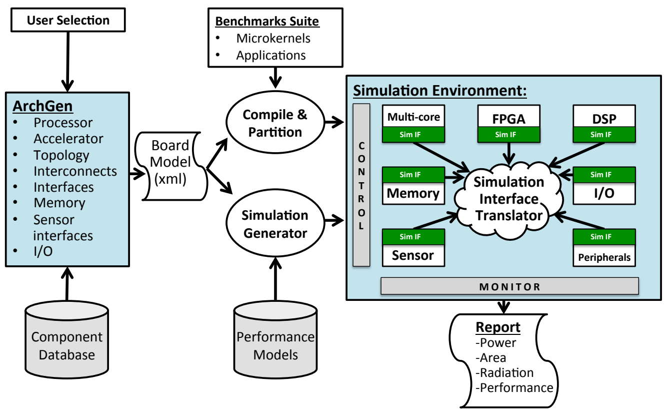


Figure 3. SpaceCubeX of Simulation Testbed

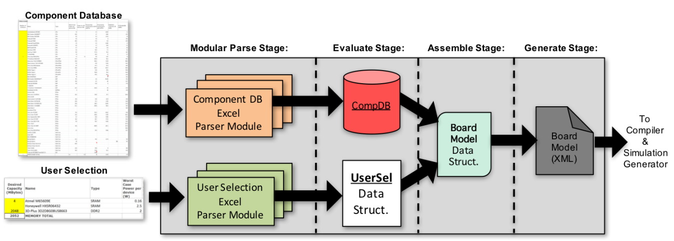


Figure 4. SpaceCubeX Architecture Generator (ArchGen) tool-flow

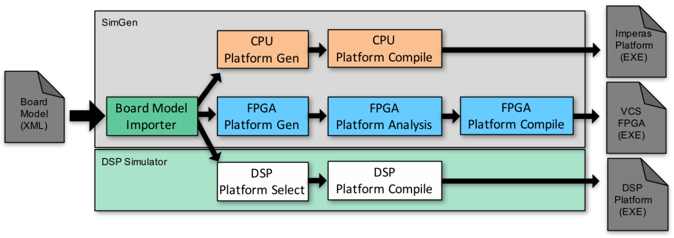


Figure 5. SpaceCubeX of Simulation Generator tool-flow

Table 1. Hybrid architecture component candidates

Category	Candidates
Multi-core	Tilera Tile64, Boeing Maestro, Aeroflex LEON4, ...
FPGA	Xilinx Virtex7, Altera StratixV, Achronix S22i, ...
DSP	BAE RADSPEED, ClearSpeed CSX700, ...
SRAM	Atmel M65609E, Honeywell HXSR06432, ...
Interconncets	PCIe (Gen2/3), XAUI, RapidIO, SpaceWire ...
Topologies	Bus, Mesh, Ring, Crossbar, Butterfly ...
Sensors I/O	RS-422, Ethernet, SpaceWire Direct I/O, I^2C ...

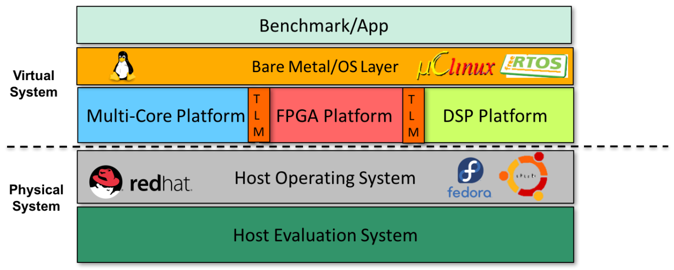


Figure 6. Run-time architecture of the SpaceCubeX Simulation Environment on a host X86 system

off-chip interconnect topologies and interfaces, including sensors and memories. Figure 2 depicts some of the different board configurations that ArchGen enables a user to specify. Internally, ArchGen and its Components Database, shown in Figure 3, leverages conventional object-relational mapping to describe each component, its properties, and how they are interconnected. USC/ISIs open-source tool set Torc [11] is be leveraged along with a Python front-end to initially populate ArchGens framework in a style similar to the popular EDIF netlist format, to greatly reduce development efforts and allow standard object-oriented languages to be utilized. A high-level diagram of this flow is shown in Figure 4.

Leveraging domain expertise from GSFC, JPL and USC/ISI the Components Database has been populated with avionics components of interest for experimentation. A component entry consists of computational, programmatic, and physical metadata. Computational metadata are details about how the component performs computation: number of cores, cache size, operating frequency, etc. Programmatic metdata describes how to use the component for computation: the compiler tool chain, debuggers, and run-time environments for operating systems. Physical metadata lists the components material characteristics: size, weight, I/O, power, heat, radiation tolerance, etc. A representative collection of the candidate components and features available during the selection stage is shown in Table 1. An end user describes a board architecture using our developed command line *ArchGen* tool to select components and describe their connections. ArchGen output is a Board Model XML file used to configure the Evolvable Testbed and is in an easily parsed format that could directly imported into common PCB development tools to start hardware development.

Evolvable Testbed

The Evolvable Testbed enables a developer to rapidly evaluate design space trade-offs based on configurations specified using ArchGen, while reusing the same testbed infrastructure to migrate from simulation to emulation level models, greatly reducing hardware development time.

Simulation represents the preliminary stage of testing and when performed properly can result in better designs, saving significant overall development time. The SpaceCubeX project extends the conventional simulation concept beyond simple software and hardware development to incorporate the entire hybrid compute architecture, shown in Figure 3.

The Simulation Generator tool, shown in Figure 5 utilizes information from ArchGens Board Model to combine the processor simulation models, incorporating industry developed tools and testing environments, such as Tileras Multi-core Development Environment and Xilinx UNISIM/SIMPRIM libraries, for accurate multi-core and FPGA simulation. The Performance Models database further enhances the simulation by providing parameter adjustments to more accurately model avionics components, such as adjusting speed grades or the number of cores available to account for radiation hardened devices more accurately. The final Simulation Environment runs on an x86 host PC and allows the use of existing simulation debugging tools and waveform viewers, similar to how virtual machines operate on host platforms, illustrated in Figure 6.

Applications or benchmarks can be compiled based on the configured architecture and are loaded into the Simulation Environment. The output of the simulation generator and compiler is fed into the Simulation Environments control and monitoring logic which interacts with each of the components to synchronize run-time execution, loads benchmark executables, and evaluates the performance of the system. The control and monitor are a standalone applications running on the host PC and interact with components via the Simulation Interface (Sim IF) wrappers. A Simulation Interface Translator provides a layer of abstraction between the components to assemble the full hybrid system, since traditional simulators are limited to single components, and eliminates the need to manually interface with each component individually. The Simulation Environments monitoring logic collects run-time statistics (compute time, memory utilization, latencies, power etc) for each benchmark. A user can then adjust the architecture based on this feedback, such as increase the number of cores for computation or changing interface types, and re-run

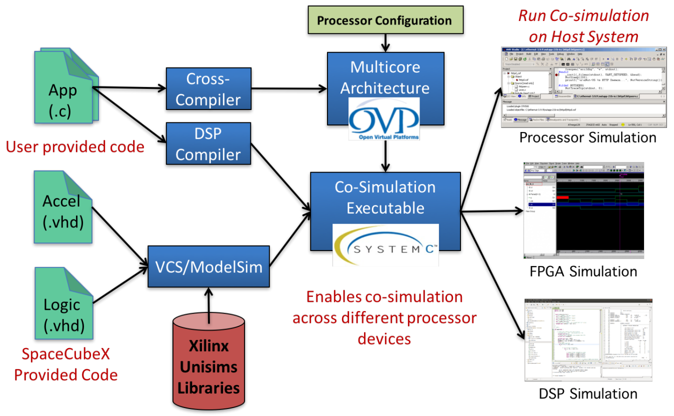


Figure 7. SpaceCubeX co-simulation flow where an end user provides application source code and the SpaceCubeX framework compiles for the specific architecture simulation

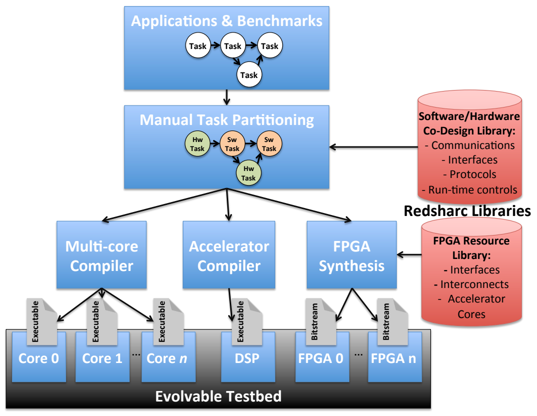


Figure 8. SpaceCubeX Compilation Flow

the experiments to compare the resulting effects. This allows the end user to quickly identify configurations suitable for further exploration as an Emulation Testbed.

4. Evaluation and Analysis

To evaluate the tools and techniques used in year 1 of the SpaceCubeX project four architectures have been generated with the ArchGen and SimGen tools. The architectures were selected based on criteria specified by space-based platform architecture experts from GSFC and JPL. A baseline SpaceCube 2.0 PowerPC system was first developed to provide an understanding of the performance differences between the generated simulator and existing hardware. While we expect performance differences based on the granularity of operating in a simulation environment, quantifying some of these differences and tuning our performance models will reduce the differences in future architectures.

Experimental Setup

Using the ArchGen tools the four architectures are specified and a board model is generated. The simulator leverages the Imperas Open Virtual Platform (OVP) simulator [12] as a core component for simulating and testing the PowerPC and ARM processors that are part of the evaluation architectures. OVP provides a suitable interface for specifying the configuration parameters and generating an executable to run our software benchmarks and applications. However, to include support for the FPGA and DSP co-accelerators a custom accelerator peripheral was developed to provide control and data transactions across the different simulator platforms. This results in separately executing simulations for the different computing platforms, OVP for the multi-core processors, Synopsys VCS for the FPGA simulation, and TI's Code Composer Studio for the DSP. The simulation environment compilation and execution is illustrated in Figure 7. This allows an end-user to follow conventional hardware/software development methodologies, but execute on co-simulators without having to manually launch individual simulators and manage data transfers between each simulator. The next section covers the benchmarking and applications that are then run on each simulated architecture to analyze performance and capabilities of SpaceCubeX's Simulation Environment.

Benchmarking and Applications

Each architecture is evaluated using a robust benchmark suite consisting of micro benchmarks, existing applications, and future mission scenarios, listed in Table 2. The micro-benchmarks provide diagnostic metrics on architectural specifics (achievable memory bandwidth, processor to co-processor bandwidth, etc). JPL provided benchmarks developed under the High-Performance and Fault-Tolerant Embedded Computing (HPFEC) task which are relevant to high computational on-board processing needs and are close approximations to several Earth observing applications. The GSFC team contributed benchmarks ranging from processing kernels to representative Earth science mission scenarios. The benchmarks outline additional benefits of this technology for Earth Science missions, including on-board data reduction, on-board event detection, adaptive sensing, real-time reconfiguration and on-board product generation (Level 0/1/2/3). The focus of this work is not currently on achieving high performance implementations of these benchmarks on the different architecture's Simulation Environments. Instead, the focus is on the capabilities of the SpaceCubeX framework to compile and deploy these applications across the architectures to better understand how the performance scales moving across the different architectures. Work is currently underway now that the Simulation Environments are developed to port more of the applications to the co-processors and determine effective speedups as compared to pure software implementations. In fact, the benefit of the SpaceCubeX project is that with the Simulation Environment, developers can quickly begin the hardware/software code-design process before any real hardware is even available, allowing the complex and often time-consuming development timelines to run concurrently rather than sequentially.

Application Mapping Process

Targeting application software for a hybrid compute architecture can be a daunting task as an application needs to be partitioned into subtasks, then mapped to specific processor types and compiled for the different processor types. Bridging from CPU (software) resources to FPGA (resources) is particularly difficult. The SpaceCubeX approach is to leverages prior IP developed under DARPA funding to help streamline the software compilation flow, reducing the effort on the benchmark developer. The compilation flow for an application targeting SpaceCubeX is shown in Figure 8. This flow still requires

Table 2. SpaceCubeX Application Benchmark Suite

Benchmark	Description
Micro Benchmarks	Kernels to benchmark architecture subcomponents and measure system viability
NAS Parallel Benchmarks	NASA generated set of programs designed to help evaluate the performance of parallel supercomputers, derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications
Packet Routing	2 kernels: Packet generation and transmission & Packet reception and verification
ATCORR	Atmospheric correction algorithm commonly used in Hyperspectral and other sensing applications
Hyperspectral Classifiers	Two classification kernels: Sulfur, Thermal
Hyperspectral Compression	Lossless Compression algorithm tuned for hyperspectral data
Image segmentation and segment analysis	Autonomous spacecraft tasking, geological feature identification, analysis, and data handling. (HPFEC-3)
Image Classification	Common image processing kernels including feature extraction, shape analysis, and surface analysis. (HPFEC-4)

Table 3. SpaceCubeX diagnostic test results for memory access

Platform	Instructions	Time (s)	Speedup
SpaceCube 2.0 (PPC)	67944358	2.059	1.000
Zynq (A9)	31208760	0.624	3.299
Hybrid (A53)	29536909	0.591	3.485

Table 4. SpaceCubeX diagnostic test results for interfaces access

Platform	Instructions	Time (s)	Speedup
SpaceCube 2.0 (PPC)	7340	0.00022	1.000
Zynq (A9)	6248	0.00012	1.780
Hybrid (A53)	5790	0.00012	1.921

manual task partitioning to map computational tasks onto each of the compute platform resources, but makes use of USC/ISIs software / hardware co-design library. This library defines common inter-task communications for dissimilar devices, providing a common API [13] implementations on several different processor types. The most difficult piece of this is bridging to FPGA hardware-like resources which is accomplished through the use of new COTS High Level Synthesis (C to gates compilers) and USC/ISIs Redsharc [14] library which provides the specific VHDL level implementations of the software / hardware co-design communication API. The user does not have to convert C code to HDL when migrating a task to an FPGA device, but still can use pre-existing highly optimized HDL if desired. This approach provides a high level of abstraction between the compute platform and application, enabling developers to seamlessly cross the software / hardware heterogeneous boundary to exploit orders of magnitude performance gains from FPGA implementations through a single API call.

Results

The year 1 results first analyze the SpaceCubeX's simulator capabilities against the existing SpaceCube 2.0 platform using GSFC developed diagnostic tests. These tests include evaluating functionality as well as performance of core operations currently performed by the SpaceCube 2.0 platforms. The simulator is evaluated to determine if these existing diagnostics can directly run, if any modifications are necessary, and how the performance differs based on the actual hardware in order to determine a performance baseline. The preliminary tests of interest for study are on memory and peripheral interface access to read/write to memory accessible by the processor and read/write to control/status/data registers for each of the peripherals in the system. Tables 3 and 4 list the results based on runtime and speedup of the tests over the different architecture simulation platforms. To further evaluate the performance differences between the SpaceCube 2.0 simulator and hardware platforms Tables 5 and 6 show common microbenchmark results for Dhrystone and Whetstone and the performance differences between the simulator and hardware platforms.

The results are highly encouraging as the performance differences between the simulator and actual SpaceCube 2.0 hardware were less than 10%. This is within tolerance given that

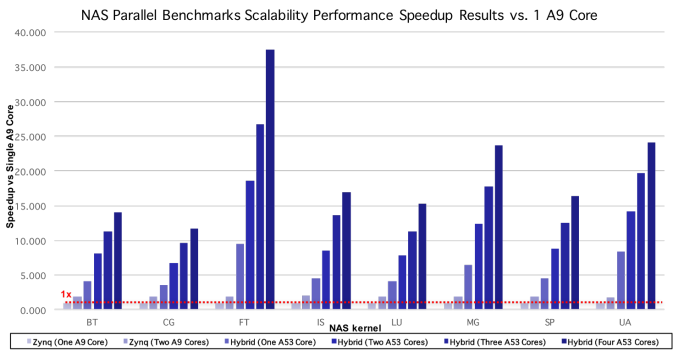


Figure 9. SpaceCubeX NAS Parallel Benchmark results on ARM A9 and A53 simulation architectures

most simulation environments only offer instruction accurate simulation as opposed to cycle accurate simulation due to overall simulation execution times. The SpaceCubeX team currently is satisfied with the slight differences because the overall simulation run-times allow an end user to quickly recompile and run applications at near real-time speeds rather than waiting for hours or even days. As is often the case when a hardware platform is being developed, the initial results may mean changes are necessary in the hardware platform in later development stages. Having a mechanism to quickly capture performances allows developers to permute different architectures and start making SWAP-based design space tradeoffs.

Additional experiments were performed to evaluate scalability of the benchmarks across the different simulation architectures. Figure 9 illustrates the speedup of the NAS Parallel Benchmarks across the ARM A9 and A53 processors. Furthermore, utilizing OpenMP the benchmark shows how the benchmark scales with the number of processor cores (1-4) for the two architecture platforms. The purposes of NAS in this work are to show the simulator properly handles the scaling between multiple cores correctly and the speedup moving to a faster and more capable processor core (ARM A9 to ARM A53) are accurately modeled.

A number of additional application experiments were run on

Table 5. SpaceCubeX simulator performance evaluation for Dhrystone benchmark

Platform	Runs	Clock Cycles/Instructions	Time (s)	DMIPS	Difference
SpaceCube 2.0 HW Platform	20000	23651950	0.2365	48.12727607	-
SpaceCubeX Simulator	20000	7980020	0.2418	47.07260082	2.22%
SpaceCube 2.0 HW Platform	200000	249200120	2.4920	45.67830574	-
SpaceCubeX Simulator	200000	80000020	2.4242	46.95502526	2.76%

Table 6. SpaceCubeX simulator performance evaluation for Whetstone benchmark

Platform	Clock Cycles/Instructions	Time (s)	Difference
SpaceCube 2.0 HW Platform	13334671	0.1333	-
SpaceCubeX Simulator	4780862	0.1449	8.29%

both standalone baremetal simulators as well as full Linux systems running on the simulator. A full Linux kernel and root file system along with libraries for OpenMP, GNU Scientific Library (GSL) and networking support allow for a VM-like experience once the simulator boots and starts running. The end user cross-compiles the application and copies the binary over to the simulator, runs the experiment, and collects the results. Figure 10 shows the full run-time comparisons between the eight applications on the simulator generated architectures. These results are on a log scale and lower is better. Not all benchmarks have been perfectly optimized for the hardware. As part of year 2 the modifications for each application based on the platform, especially the FPGA, are being made. Figure 11 provides an energy consumption comparison based on the applications to allow a designer to start making power and performance design considerations based on the application needs and system power capabilities.

Overall, these results are highly encouraging and motivating the need to migrate from lower performance processor-based systems to hybrid multi-core/FPGA/DSP architectures for next generation space-based systems used for earth science missions. The year 1 results include running over 200 benchmark application experiments on a full set of eight permuted architectures for multi-core processors, FPGAs and DSPs. In general it was observed the ARM A53 system significantly outperforms the ARM A9-based architectures, but at a slightly higher cost in power consumption. Multi-core architectures provide a fast, scalable approach to quickly accelerate an existing single threaded application using common libraries, such as OpenMP. Also, hybrid architectures provide the best performance and power efficiencies at a cost of additional development time and complexity. However, the benefit of the SpaceCubeX framework is that a developer can begin the hardware/software co-design process far earlier in the development cycle and even provide feedback on architecture decisions. Finally, the SpaceCubeX framework provides a convenient mechanism to quickly port applications between different architectures and perform design space exploration studies specific to their application or mission needs.

5. FUTURE WORK

Emulation follows simulation as the next stage of testing and is critical to enable medium to large scale tests of the system, in near real-time on representative data sets. As the project has concluded with year 1 simulation research, we are now beginning to design and implement the Emulation testbed. The work will extend the capabilities of the Simulation testbed presented here to support running on disparate development boards and devices. Emulation allows applications to run on representative hardware, modeling integration between the different physical components. In some cases the devices might be early release development boards, whereas, in other cases the hardware maybe close to the actual final hardware, but requires fine-tuning the system to better reflect the target architecture. Furthermore, with emulation the applications are running at closer to real-time speeds which decreases the time per experiment and allows for larger data sets to run. This enables tighter integration with memory, sensor interfaces, and other protocols between the target architecture and other components in the system.

Proceeding to emulation does present some challenges, such as the requirement of careful accounting of computation and data transfer times to avoid penalizing the experiment for emulator limitations. The interfaces between the different physical components, such as an ARM A53 processor and FPGA device, must be properly mapped and managed. Finally, all of this needs to be implemented in a seamless top-level interface such that the end user does not manually have to manage loading and running their application across all of these different devices.

We will leverage the board model generated by the ArchGen tool and develop a complementary Emulation Generator tool to build the necessary infrastructure to assemble breadboard hardware to create the Emulation Testbed. A key advantage of this approach is the considerable reuse of interfaces and code from the Simulation Testbed and the Benchmarks. This includes all of the software infrastructure to handle the run-time distribution, control, and monitoring of the results.

Overall, the results from the SpaceCubeX project have been highly encouraging and are directly impacting the selection and design of the future SpaceCube hardware. As a result, the diagnostics, benchmarks, and applications already developed can be immediately leveraged for the development of engineering and future flight hardware. Lastly, we have been in discussions with external groups to test their applications on these simulation and emulation platforms to help develop more user friendly interfaces, APIs, and models that can directly impact future missions and earth science research. We hope to continue to open our tools and models to the broader research community and look forward to continued advancements and research into heterogeneous multi-core/FPGA/DSP platforms for space-based applications.

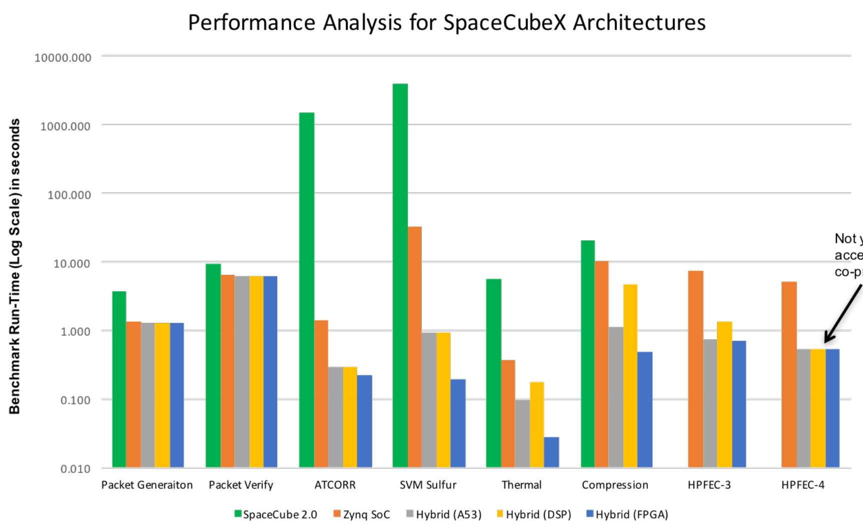


Figure 10. SpaceCubeX performance results for the architectures implemented in the Simulation Environment

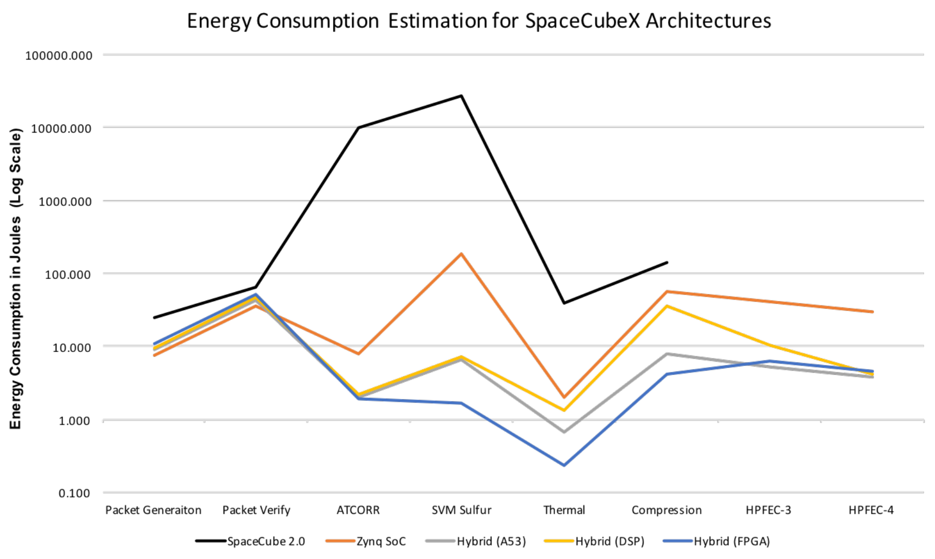


Figure 11. SpaceCubeX energy consumption results for the architectures implemented in the Simulation Environment

REFERENCES

[1] C. Villalpando and R. Some, "Reconfigurable machine vision systems using fpgas," in *Adaptive Hardware and Systems (AHS), 2010 NASA/ESA Conference on*, June 2010, pp. 31–35.

[2] C. Villalpando, R. Werner, J. Carson, G. Khanoyan, R. Stern, and N. Trawny, "A hybrid fpga/tilera compute element for autonomous hazard detection and navigation," in *Aerospace Conference, 2013 IEEE*, March 2013, pp. 1–9.

[3] "NASA Missions A-Z," http://www.nasa.gov/missions, accessed 2016-10-21.

[4] P. Pingree, *Advancing NASA's On-Board Processing Capabilities with Reconfigurable FPGA Technologies*. INTECH Open Access Publisher, 2010. [Online]. Available: https://books.google.com/books?id=gdWgoAEACAAJ

[5] A. P. et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *41st Annual International Symposium on Computer Architecture (ISCA)*, June 2014.

[6] J. Clark, "http://www.theregister.co.uk/2014/06/18/intel_fpga_custom_chip."

[7] Xilinx, "Xilinx Zynq All-Programmable SoC Technical Reference Manual," September 2013, v1.6.1.

[8] Altera, Inc., "Arria 10 Device Overview," September 2014, aIB-1023.

[9] T. M. Lovelly, D. Bryan, K. Cheng, R. Kreynin, A. D. George, A. Gordon-Ross, and G. Mounce, "A framework to analyze processor architectures for next-generation on-board space computing," in *Aerospace Conference, 2014 IEEE*, March 2014, pp. 1–10.

[10] T. P. Flatley, "Keynote address i: Spacecube: A

family of reconfigurable hybrid on-board science data processors." in *AHS*, U. D. Patel, K. Benkrid, and D. Merodio, Eds. IEEE, 2012. [Online]. Available: http://dblp.uni-trier.de/db/conf/ahs/ahs2012.html#Flatley12

[11] N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French, "Torc: Towards an Open-Source Tool Flow," in *Proceedings of the 2011 ACM Nineteenth International Symposium on Field-Programmable Gate Arrays, FPGA 2011, (Monterey, California), February 27–March 1*, 2011, http://torc.isi.edu.

[12] Imperas, "http://www.imperas.com/ovpworld."

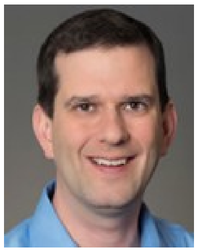
[13] F. Labonte, P. Mattson, W. Thies, I. Buck, C. Kozyrakis, and M. Horowitz, "The stream virtual machine," in *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 267–277. [Online]. Available: http://dx.doi.org/10.1109/PACT.2004.29

[14] W. V. Kritikos, A. G. Schmidt, R. Sass, E. K. Anderson, and M. French, "Redsharc: A programming model and on-chip network for multi-core systems on a programmable chip," *International Journal of Reconfigurable Computing*, December 2011.

BIOGRAPHY

Andrew G. Schmidt *is a Research Lead and Senior Computer Scientist at the University of Southern California's Information Sciences Institute. He is the technical lead of the SpaceCubeX project investigating heterogeneous computing platforms for next generation earth science missions. He received his Ph.D. in electrical engineering from the University of North Carolina at Charlotte in 2011 investigating efficient utilization of heterogeneous compute resources through static analysis and runtime performance monitoring. He received his M.S. and B.S. in computer engineering from the University of Kansas in 2007 and 2005. His research interests include heterogeneous and reconfigurable computing, fault tolerant and resilient computing, high performance computing, and embedded systems. Dr. Schmidt has over 30 publications in the areas of reconfigurable computing based technology and is a member of IEEE and ACM.*

Gabriel Weisz *is a computer scientist at the University of Southern California's Information Sciences Institute. His research are in computer architecture, with a focus on reconfigurable and high performance computing. He has published a several papers in these fields, and holds one patent. He has a B.S. in Computer Science and Electrical Engineering from Cornell University, and M.S. and Ph.D. in Computer Science from Carnegie Mellon University.*



Matthew French *is a Research Director in the Computational Systems and Technology group at USC/ISI where he facilitates large scale, cross-discipline research which spans the institute. In this capacity, Mr. French serves as the director of the SecUre and Robust Electronics (SURE) center at ISI, which conducts research and provides services in hardware trust, security, reliability, and resiliency. Mr. French also oversees the Reconfigurable Computing Group at ISI, which performs research in application mapping, hardware / software co-design, CAD tools, and front-end ASIC design. Mr. French has over 40 publications and 2 patents, and serves on the program committee for several conferences in his field. Mr. French holds M.E. and B.S. degrees in Electrical Engineering from Cornell University.*

Thomas Flatley *is a Computer Engineer at the NASA Goddard Space Flight Center, and is currently Branch Head of the Science Data Processing Branch. Mr. Flatley's current work includes the coordination of embedded science data processing technology development and hardware accelerated science data processing activities, serving as Principal Investigator on multiple flight processing experiments, with the primary goal of developing reconfigurable computing technology and hybrid systems for flight and ground science data processing applications. He is also a key member of the GSFC CubeSat/SmallSat technology working group, manages numerous collaborations with government, industry and academic partners, and serves as liaison between technology developers and end users in the science community. Mr. Flatley was named a Goddard Senior Fellow in 2016.*

Carlos Y. Villalpando *is a Senior Member of Technical staff in the Advanced Computer Systems and Technologies group at the Jet Propulsion Laboratory. He is currently a digital designer for advanced computing techniques for machine vision applications in FPGAs as well as system designer and programmer for machine vision tasks on multicore processors. He earned his Bachelor of Science degree in Electrical Engineering, Computer Block at the University of Texas at Austin in 1996 and a Master of Science in Electrical Engineering- Vlsi at the University of Southern California in 2003. He has been a member of the JPL community continuously since 1993 and has worked primarily on Technology development tasks. Some of his technology work is now being infused into the upcoming Mars2020 rover mission.*