12/8/2008

Technical Report

A Collaborative Research Proposal to the NSF:

Research Accelerator for Multiple Processors (RAMP) - A Shared Experimental Parallel HW/SW Platform

Arvind (MIT), Krste Asanović (UC Berkeley), Derek Chiou (UT Austin), Joel Emer (MIT/Intel),
James C. Hoe (CMU), Christoforos Kozyrakis (Stanford), Shih-Lien Lu (Intel),
David Patterson (UC Berkeley), Jose Renau (UC Santa Cruz), and John Wawrzynek (UC Berkeley)

1 Overview

Motivations and Challenges. In 2005 there was a historic change of direction in the computer industry: all microprocessor companies announced that their future products would be chip-scale multiprocessors (CMPs), and that future performance improvements would rely on software-specified core-level parallelism rather than depending entirely on software-transparent instruction-level parallelism. In doing so, the semiconductor industry has made a bold bet that a satisfactory solution to the general-purpose parallel programming problem will emerge, despite slow progress in previous decades of research.

This disruptive sea change has both the software and hardware industries retooling for a future of commodity parallel computing. However, at this point there is no consensus on the correct direction for moving forward in parallel computing. All components of the computer system, from the programming models, compilers, runtime environments, operating systems, and hardware architecture are being examined in the quest to make parallel execution more generally applicable. Being able to experiment with co-developed hardware and software concepts—rather than living within the confines of traditional hardware-software separations—is much more likely to produce the necessary breakthroughs.

Unfortunately, the prevailing simulation-driven architecture and microarchitecture exploration methodology makes hardware-software co-design virtually impossible. Accurate software simulators capable of comparing architectural mechanisms are already slow and are getting slower as the simulated systems become ever more complex. Simulation speeds are already far too slow for software development.

Alternatively, one could build a prototype of the target[1] using Field-Programmable Gate Arrays (FPGAs) to create a fast execution platform for software development before the final hardware becomes available. However, there are several major obstacles in this approach. FPGAs are often different than the final implementation technology, requiring changes to the design in order to fit and/or run well in the FPGA. Such changes potentially change the behavior of the FPGA implementation, thus reducing its accuracy. Since the design is generally written in a register-transfer-level language such as Verilog, changes for any reason, including those to better map to an FPGA, those to modify the design and those to add observability that would not otherwise be in the design are difficult and error-prone. For these reasons FPGA prototyping is used mostly for final validation once the RTL is close to completion and rarely used for architectural exploration and early software development.

FPGA-Accelerated Simulation. A promising new approach to the simulation speed problem is building *simulators using FPGAs*. Such FPGA-accelerated simulators internally may look different than the target system they simulate, but nevertheless can emulate the desired target system behaviors, including cycle-accurate timing, quickly and accurately. The promise of this approach is based on four concurrent developments:

1. the ever increasing size of FPGAs that can now accommodate small multicore systems on a single chip;
2. the tremendous progress in synthesis tools that can generate good quality hardware from high-level hardware descriptions;
3. much deeper understanding of techniques to model synchronous systems in a virtualized manner where each model clock cycle can take variable number of physical host clock cycles; and

[1] We use the term *target* to mean the system being simulated and the term *host* to mean the platform on which the simulator runs.

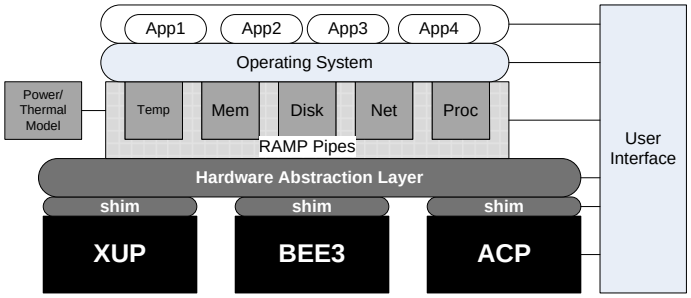

Figure 1: RAMP Implementation Stack. Components in shades of gray are the proposed infrastructure.

4. new simulator partitioning techniques that reduce implementation effort, reduce FPGA area, and provide full-system capabilities to FPGA-based simulators.

FPGA-based simulators can provide a hardware-speed environment for simultaneous software development and architectural evaluation. Moreover, they can be rapidly reconfigured to speed up architectural exploration, including exploration driven by feedback from synergistic software research and development. FPGA-based simulators can also incorporate observability and reproducibility mechanisms, implemented in hardware with minimal or no performance impact. FPGA-based simulators and their performance can scale with the target for studying many problems. Using such platforms, the hardware-software development feedback cycle can potentially be reduced from years to days.

Of course, being able to *rapidly* construct, debug, and modify such simulators requires better tools and infrastructure than what exists today. Though FPGAs eliminate many of the Non-Recoverable Engineering (NRE) costs often associated with hardware development, such as creating mask sets and actual fabrication, using FPGAs still requires a considerable hardware development effort. Implementing an efficient simulator in FPGAs using current FPGA tools requires a detailed design specification at the register transfer level (RTL) with a strict adherence to the physical constraints of the FPGA devices and the development board. Furthermore, the wide diversity in FPGA platforms creates an additional challenge for FPGA-based simulators, since different FPGA boards have different physical resources and thus require very different mappings.

We are proposing to build an infrastructure to exist on top of FPGA platforms to hide the individual details of the FPGA devices and boards from the user. In addition, a robust ecosystem of simulator models, including architecture models, real operating systems, run-time systems, compilers, and application software built on top of that infrastructure would be available as a research springboard and baseline (one instance is shown in Figure 1). Without such an infrastructure and ecosystem, the barrier to entry is gigantic; only the very capable or the very foolish would attempt to scale that barrier with a single research group.

We believe the upside potential to this proposed infrastructure is so compelling that it will create a "watering hole" effect in academic departments, as people from many disciplines would use it in their research. This infrastructure is obviously attractive to hardware and software researchers studying parallel computing, but should also be appealing to researchers studying most other areas of computing as well. Some representative research projects that we believe could benefit from RAMP are: testing the robustness of multiprocessor HW/SW under fault insertion; developing thread scheduling, data allocation/migration, and other operating systems mechanisms; developing and evaluating new application-specific instruction set architectures; creating an environment to emulate a geographically distributed computer, with realistic delays, packet loss, etc. (Internet in-a-box); developing and testing of hardware and software schemes for improved security; recording traces of complex programs; evaluating the design of multiprocessor switches; developing parallel file systems; designing and evaluating low power embedded processors, and testing dedicated enhancements to standard processors.

Such a watering hole would lead to conversations between disciplines that rarely talk to each other, helping us to more quickly develop multiprocessor systems that are easy to program efficiently. Indeed, to help industry win its bet on parallelism, we will need the help of many people, for the parallel future is not just an architecture change, but likely a change to the entire software ecosystem.

The RAMP Team. RAMP is a multi-university and industry collaboration, formed three years ago, to develop that FPGA-based simulation infrastructure to support computer system hardware and software research, including large-scale multiprocessor targets. It is the first and still the only group of its kind. Each of us had independently reached

the conclusion that we needed fast simulators and that FPGAs were the only viable way to get that speed but were resigned to build the necessary infrastructure ourselves.

When we met at ISCA 2005 and we discovered we had common needs, we immediately decided to share the considerable effort of building and distributing that infrastructure, and use our own models to seed the ecosystem built on top of that infrastructure. We proposed and were awarded a medium-size NSF CRI award that provided for two staff at Berkeley to help with RAMP coordination and integration of infrastructure and models developed in our own research projects, but no funding for model development.

Though we have made significant progress that we outline in Section 2, it became clear that achieving widespread use of RAMP would require significantly more effort than was possible within the time and funding of the original RAMP award. Issues include the following: (i) research quality often is not release quality nor based on modular infrastructure making it more difficult than anticipated to use the work of other groups, (ii) FPGA tools and systems are more expensive than originally anticipated, and (iii) FPGA-based processors are less dense than originally anticipated causing us abandon some of the original approaches to building RAMP systems. In addition, our target user base became both larger and less FPGA-sophisticated than originally anticipated. This proposal addresses these issues using our experiences from our initial RAMP effort and the applicable advances that we have made in our research.

The Continuing RAMP Vision. The progress we have made to date have been accomplished mostly through volunteer or otherwise leveraged efforts by the RAMP PIs and their students. In order to address the issues and additional constraints to realize the RAMP vision, we are requesting NSF funding for six universities to engage in a coordinated effort to enhance the RAMP infrastructure.

The ***intellectual merit*** of this proposal is embodied in the following contributions. First, we intend to create a set of design standards that facilitate easy adaptability and integration of hardware and software components into FPGA-based simulators that model interesting targets. Second, we intend to use these design standards to create multiple, easy-to-use-and-extend baseline simulators, some of which will scale to 1024+ target cores. Third, we will define and create infrastructure to observe the behavior of simulated large systems without disturbing the simulated computation.

The ***broader impact*** goal of this proposal is nothing less than the transformation of the computer research community from a software-simulation driven discipline to a hardware enabled discipline. The impacts of this transformation will be the acceleration of multi-disciplinary research in the areas of parallel computing and computing in general, removing the impediments of large-scale computer systems research in the 21st century.

Proposal Outline. The rest of the proposal is organized as follows. The next section summarizes the accomplishments of the RAMP collaborative to date. Section 3 describes the lessons we have learned and the research work that we we draw upon to enhance the RAMP infrastructure. Finally, Section 4 details what we are proposing.

2 RAMP Accomplishments

The genesis of RAMP was a 2005 medium sized NSF-CRI award in which we proposed a RAMP design framework that would allow flexible, modular, cycle-accurate emulation of target machines. The framework was to support both cycle-accurate emulation of detailed parameterized machine models and rapid functional-only emulations. It would provide an abstract hardware layer to make all hardware platforms appear the same to a RAMP design and to allow RAMP modules to be reused in subsequent hardware revisions. A full hardware system would be constructed within the framework from composable *units*, where a unit is a relatively large number of gates and has a single clock domain. Example units include CPUs, coherence engines, DRAM controllers, I/O device interfaces, and network router stages [1].

Over the life of the project, we have made significant progress toward our technical goals. We have designed and built much of the basic FPGA infrastructure as well as several example parallel processing systems to demonstrate the feasibility of the RAMP approach. Specifically, we have:

- Built several FPGA-based processor core and memory models, including models for novel architectural features such as transactional memory;
- Used the Berkeley Emulation Engine 2 (BEE2), a preexisting FPGA platform, to demonstrate a 1008 core parallel processor system;
- Used our experience with BEE2 to design the next generation FPGA emulation platform (BEE3) and transfered its design and responsibility for manufacturing to a third party (BEEcube);
- Implemented several approaches to standards interfaces for module level interchangeability and sharing;

- Demonstrated an approach to simulation "time dilation" for cycle-accurate simulation;
- Built several prototype hybrid host systems combining FPGA based emulation with conventional processors and software; and
- Investigated several novel techniques to processor modeling for improved simulation and efficiency and scaling.

The sections below highlight our major accomplishments resulting from the original RAMP NSF-CRI award.

2.1 Berkeley Emulation Engines

The catalyst that started RAMP was a presentation on the Berkeley Emulation Engine 2 (BEE2) [2] board at the first Workshop on Architectural Research using FPGA Platforms (WARFP) that most of the PIs attended. When we all approached the Berkeley group after the talk to ask how to get BEE2s for our own research, we discovered our mutual interest in hardware and software infrastructure. Thus, the RAMP collaboration formed around using the BEE2 as the first RAMP FPGA platform, which we called RAMP-1. In that capacity, the BEE2 has served well as the platform of the early RAMP machine prototypes and greatly influenced the feature list for BEE3. While fully capable for early RAMP work, BEE2s use older FPGAs circa 2002 and were not designed for mass production, thus calling for a new platform.

Based on our experience using the BEE2 FPGA platform and a survey of RAMP developers and users, we arrived at a high-level design specification for the next Berkeley Emulation Engine (BEE3). Under the direction of Chuck Thacker (Microsoft Research) and Berkeley designers, an arrangement was reached wherein a contract design house would professionally produce the electrical and mechanical design to specification. This work was undertaken as a donated contribution and funded by Microsoft in support of the project.

The BEE3, specifically designed for FPGA-based computation and simulation, is based upon devices from the state-of-the-art Xilinx 65nm FPGA family. Each 2U rack mount BEE3 module consists of four large Virtex-5 LXT/SXT/FXT FPGA chips, along with up to 64GB DDR2 DRAM and eight 10Gigabit Ethernet interfaces for inter-module communication. In addition, up to 4 PCI Express x8 connections allow a maximum of 16GB per second full-duplex data communication between each BEE3 module and a front-end computer. At a power consumption of less than 400 watts, each BEE3 module can provide over 4 trillion integer operations per second, or emulate over 64 RISC processor cores concurrently at a several hundred MHz rate.

The new hardware is entering full-scale production in 2H 2008, and contains numerous features and capabilities specific to the RAMP project. To eliminate the issue of academic-supervised manufacturing and distribution, which was problematic with the BEE2, BEEcube, Inc. was founded to commercialize the product. Projected demand for the BEE3 has already exceeded all preceding BEE generations, demonstrating industry and academic recognition of the validity of the approach.

2.2 RAMP Simulators

In the original proposal, we outlined a plan to create four basic gateware[2] models, fulfilling six project steps. That tentative plan was based on conservative assumptions as to the ability to implement soft processors cores and operating systems support, at that point a significant unknown. In the second year of the project, accumulation of experience and confidence advocated a more aggressive approach. We transitioned directly to developing three reference prototype systems: *RAMP Red*, *RAMP Blue*, and *RAMP White*. Each system was defined to exercise some particular aspect for which we had significant familiarity, and to rapidly fulfill two goals of RAMP: provide models to serve as the starting point for an entire community of architecture researchers, and to fully support and distribute those models.

Each reference design employs a complete gateware/software configuration of a scalable multiprocessor consisting of hardware components (such as processor cores, memories, and switches) and software components (such as operating systems). Because the designs are modular, the hope was to be able to interchange modules to quickly modify reference designs or create new ones. The resulting systems would be capable of running and monitoring complex system benchmarks. The expectation was that users would release back to the community new and enhanced gateware/software modules.

Below, we present a summary of the original RAMP Red/White/Blue systems as well as several associated systems and infrastructural technologies that rose out of our experience with the reference systems.

• **RAMP Red (Stanford)** has the distinction of being the first-ever multiprocessor system with hardware support for

[2]We use the word "gateware" as the code that instantiates hardware circuits within an FPGA.

Transactional Memory (TM), that shifts the onus for concurrency management from the programmer to the hardware system. By introducing access conflict detection and rollback capabilities to the shared memory in a parallel system, software tasks (transactions) can execute in isolation and atomically without the use of locks by simply detecting when atomic regions do not execute atomically and rolling back to recover. Given such support, software simply needs to correctly delimit atomic regions to get correct behavior. The current RAMP Red design includes 9 processors and boots the Linux operating system [3, 4]. It has served as the development platform for programming models based on TM, as well as productivity tools such as performance debuggers and deterministic replay mechanisms.

• **RAMP Blue (Berkeley)** is a family of emulated message-passing machines [5] that run parallel applications written for either the Message-Passing Interface (MPI) standard or for partitioned global address space languages such as Unified Parallel C (UPC). RAMP Blue can also be used to model a networked server cluster, and is an example of a mapping processor RTL directly to an FPGA. Currently at version 4, Blue has proved to be a rapidly evolving, highly scalable, very successful demonstrating vehicle. In rapid succession, the UCB team produced a 256, 768, and ultimately 1008 core system [6], followed by a production 256 core platform, thereby demonstration the prototyping flexibility of the RAMP approach.

• **RAMP White (UT Austin)** is a cache-coherent shared memory version of RAMP that incorporates some elements and lessons from both Blue and Red. Though it is useful as a RAMP platform, it is being developed as a parallel host platform for FAST (see next paragraph) to model parallel targets in a scalable fashion. Thus, it is the first of the RAMP projects to use the split functional/timing approach. RAMP White is currently capable of booting an SMP operating system on top of an un-cached multiprocessor based on the SPARC Leon3 [7] processor and will soon support coherent caches.

• **FAST**[3] **(UT Austin)** is a current research project developing a speculative split functional/timing approach for cycle-accurate simulation [8, 9, 10] and a non-speculative approach for even faster approximately cycle-accurate simulation. In a FAST system, the functional model speculatively executes the program, sending an instruction trace of that execution to the timing model. In the case where the functional trace instructions are not what the timing model expects, the timing model can force the functional model to rollback and execute down a different control path to get the correct target instruction order (the non-speculative approach doesn't support rollback.) The speculative functional model permits the functional model to run independently of the timing model, thus enabling efficient parallelization. Implementing the timing model in FPGAs further parallelizes the system, getting even more performance. The FAST prototype executes the complete x86 and PowerPC instruction sets, boots unmodified Windows XP (x86) and Linux (x86 and PowerPC) and runs unmodified applications while modeling an aggressive out-of-order superscalar core at an average simulation speed in excess of 1.2MIPS using a software-based functional model and an FPGA-based timing model. Very early experiments with the non-speculative FAST-MPLite indicate that 20MIPS-50MIPS per host core is possible.

• **ProtoFlex (CMU)** is developing a simulation architecture that uses FPGAs to accelerate architectural-level full-system simulation of multiprocessors [11]. The ProtoFlex simulation architecture reduces the hardware complexity of simulating large multiprocessor systems by mapping many logical processors onto an interleaved multi-context execution engine. Furthermore, it employs a hybrid simulation technique that implements only the most frequently encountered behaviors in the FPGA. Rare behaviors are simulated in software to more easily provide the complete set of behaviors needed in a full-system simulation. These two virtualization techniques in combination greatly reduce the effort of constructing an FPGA-accelerated simulator. We have successfully applied the ProtoFlex simulation architecture in a full-system functional simulator (BlueSPARC [12]) that models a 16-way UltraSPARC III symmetric multiprocessing server target. Using a single Xilinx Virtex-II XCV2P70 FPGA for acceleration, this ProtoFlex simulator achieves an average 39x speedup (and as high as 49x) over state-of-the-art software simulations for a range of benchmarking applications. Continuing development of ProtoFlex simulation architecture (supported by NSF CPA-0811702) will focused on the scalability and generality of this complexity-effective simulation architecture.

• **RAMP Gold (UC Berkeley)** is a new reference design inspired by these new directions. RAMP Gold incorporates many of the new simulator technologies developed in allied efforts. RAMP Gold is being developed with funding from the Intel/Microsoft UPCRC grant to the Berkeley Parallel Computing Laboratory (ParLab). RAMP Gold provides a parameterized model of a manycore processor with coherent caches, and is intended to support ParLab research on parallel architectures and new parallel software stacks, including hypervisor and operating system layers. The RAMP Gold model employs split functional and timing models both of which are implemented in hardware, and both of which use host multithreading for increased FPGA simulation efficiency, scalability, and flexibility. The timing model

[3] FPGA-Accelerated Simulation Technologies

accepts parameters at model boot time to support a wide range of possible machine models, including varying numbers of cores, cache configurations, and DRAM bandwidths and latencies.

We anticipate a single FPGA on the BEE3 board can model 128 target processor cores including timing models, while a design scaled out to 16 BEE3 boards can emulate target systems with 8192 cores. We predict target application performance of around 1–20 aggregate GIPS while maintaining full cycle-level timing information on the largest configuration. In addition, a smaller version of the RAMP Gold design can run on the single FPGA of a Xilinx XUP board, providing emulation of a 64-processor systems at around 130 aggregate MIPS with full cycle-level timing models.

• **HAsim (MIT and Intel)** focuses on using FPGAs to produce cycle-accurate simulators of microprocessors and multicore systems. By implementing the simulators in FPGAs, HAsim intends to greatly accelerate simulation speeds while retaining the accuracy of software simulators. Contributions include developing a lightweight, distributed scheme for cycle-accurate simulations on highly-parallel environments [13], and a scheme for implementing a closely-coupled partitioned simulator [14]. The HAsim project has demonstrated a cycle-accurate simulator of a MIPS R10K-like 4-way out-of-order superscalar core running on a Virtex 5 FPGA achieving a simulation speed of 10 MHz, a significant improvement over software-only simulators [15, 16, 17].

One observed outcome is that gains in simulation speed may be rendered unimportant if it takes developers a significant time to implement the simulator. To this end the HAsim project has prioritized the development of a common, reusable infrastructure of abstractions for FPGA programming inspired by best-practices in software engineering and high level hardware synthesis. One of the goals of this work is to allow the programmer to develop plug-and-play modules that can be interchanged. For example, a set of virtual devices abstract away the physical details of a particular system, thus easing porting between different FPGA platforms. Capabilities of this system also include sharing common data types between the FPGA and software, and automatically generating circuits and code to abstract the communication, similar to the successful Remote Procedure Calls paradigm in software.

• **RAMP Design Language (RDL) (UC Berkeley)** is an effort to present a system architect a standard interface for interoperability, and simulation time management, with a level of abstraction which accurately captures the machine behavior in a form amenable to rapid design evolution [18]. RDL is a system level language that relies on other frameworks, languages and tools for the implementation of individual components. RDL provides a distributed event simulation and message passing framework, which supports the overall goals of RAMP, including timing accurate simulation through the virtualization of clock cycles to support accurate simulation, and the independent development of structural model simulators, to support collaboration.

2.3 Outreach Experiences

The RAMP project actively sought interactions on three fronts: (i) student recruitment to engage and train young researchers in parallel computing, (ii) collaboration with non-RAMP PIs to foster discussions, creativity, and fresh approaches, and (iii) off-site retreats where industrial partners and other academics are invited to maintain the relevance and reaffirm the direction of the RAMP project. As the project matured, additional benefit was perceived in a strong online presence, specifically a descriptive website and publicly available design repository, that served to broaden and diversify the participating community. It should be noted the RAMP effort has strong collaboration and participation crossing gender, ethnic, and disability boundaries.

2.3.1 Design Repository and Website

The mechanism for design sharing is a web-based model, with two general components: an archive of publicly available designs, and a website with detailed descriptions of various projects, publications, and other materials. The public repository, now in its third generation, has evolved in direct response to user feedback and a desire to make it as useful a resource as possible. The repository has had a positive effect on collaboration and dissemination; web statistics record consistent external accesses in excess of 5000 unique visits per month for the descriptive RAMP website, indicating strong and continued interest in the project. Furthermore, the design archive typically logs over 1500 monthly repository downloads, the majority from unique sites, pointing to ongoing and widespread community interest in the effort. This public success underscores not only the validity of the direction and relevance of the research effort, but also the inherent value of the dissemination approach. The next phase of the proposed research seeks to dramatically extend the impact and utility of this shared resource.

2.3.2 Academic Incorporation

Student involvement was facilitated through incorporation of RAMP infrastructure in lectures and presentations, as well as classwork. Additionally, at each of the member institutions demonstrations were performed during open house events and exhibits. Classwork also incorporated and contributed back to the RAMP effort at each university, which mirrored the RAMP vision of seeding next generation architectural concepts. For example, Prof. David Patterson's graduate computer architecture course at Berkeley helped create hardware components which were utilized in the design of the RAMP Blue system. At Carnegie Mellon, Prof. Todd Mowry's graduate architecture class downloaded the RAMP Blue system and explored the benchmark profiling of parallelization on the reference manycore systems. In another Berkeley effort, the RAMP-1 hardware was as used as a verification platform for classwork that developed the novel highly concurrent FLEET computer architecture (led by Ivan Sutherland of Sun Research). These interactions, where the design elements of one institution or group provides the basis for exploratory work at another, is exactly the behavior we had hoped would manifest.

Additional venues such as campus-wide lab visits and demo days were leveraged as well, and specialized closed events, for instance the Berkeley ParLab visit by Intel/Microsoft researchers where RAMP Blue was prominently showcased. These early discussions and explorations with colleagues have resulted in many of the new directions proposed for future work, and emphasized the need for broad external access to project resources.

2.3.3 Workshops and Tutorials

During the second project year, dissemination of RAMP concepts to external academic and industry researchers was accomplished in two highly successful tutorial/workshops held in conjunction with leading computer architecture conferences, and served to engage external colleagues. Significant interest was expressed by the architecture community at both venues, resulting with in-depth discussions with attending PIs.

• **ISCA 2007.** We held an all-day, hands-on tutorial at the International Symposium on Computer Architecture (ISCA) at the 2007 Federated Computer Research Conference (FCRC) in San Diego that was attended by over 50 researchers. RAMP Blue v3 was demonstrated. The participants also developed, debugged, and optimized parallel programs with transactional memory using the RAMP Red model. A separate demonstration of the RAMP Blue hardware was presented at a joint poster session, which was functionally a 768-core system performing the NASA parallel benchmark and was demonstrated in real-time. The ISCA demonstration and the heavily-attended workshop were well received. Due to broad interest expressed by the computer architecture community, an invitation was issued to present a similar program at ASPLOS 2008.

• **ASPLOS 2008.** At the Architectural Support for Programming and Operating Systems (ASPLOS) 2008 conference in Seattle, we held another all-day tutorial attended by over 20 registered attendees. RAMP Blue v4 and RAMP White were demonstrated. At that time, a number of RAMP efforts and allied projects were in a functional stage, and provided a basis for several hands-on sessions, one to a remote RAMP-1 hardware stack. Attendees operated various RAMP hardware projects and received a CD containing ProtoFlex and RAMP Blue download instructions. It should be noted that at both events, students presented the material and were instrumental in instructing and operating the demonstrations in keeping with RAMP's educational mission.

2.3.4 Industry Adopters

We have also had considerable interactions and collaborative work with a number of companies and with other universities—not part of the official RAMP collaboration. These interactions are summarized below.

• **Sun Microsystems-OpenSPARC.** OpenSPARC T1 is the open-sourced version of the commercial UltraSPARC T1, released by Sun in an effort to stimulate research in the field of multi-core processors. The 64-bit processor, originally designed for high performance using a custom design methodology, required selected portions of the processor source code to be re-written for optimal programmable logic synthesis. At Sun, a complete FPGA system was built around a single OpenSPARC T1 core and implemented on a number FPGA boards, including two generations of BEE platforms.

The configured hardware system is capable of booting a version of a commercial operating system, OpenSolaris—modified to use the FPGA board I/O and to fit in the limited amount of supported memory. This version of OpenSolaris has been made available to other RAMP efforts as well as external adopters.

• **Xilinx, Inc.** Xilinx Research Labs worked closely with Sun Microsystems in the alliance to implement OpenSPARC T1 on Xilinx FPGAs. By releasing the FPGA optimized version of the OpenSPARC T1 eight-core, four-threads per core chip multi-threaded (CMT) 64-bit processor with hypervisor virtualization support, the two partners hope to

enhance and extend the RAMP vision beyond the initial collaborators and to the community already supported by the University Program at Xilinx. Currently available in the OpenSPARC T1 1.6 release are designs targeting the broadly available and subsidized ml505_v5lx110t reference board. At the August 2008 RAMP Workshop, a four-thread OpenSPARC T1 core booting OpenSolaris on the BEE3 RAMP platform was demonstrated; the OpenSPARC T1 1.7 release is anticipated in the second half of 2008 will include that BEE3 OpenSPARC design.

Xilinx has also made a significant commitment to the RAMP effort by donating FPGA devices, tools, and expertise, along with funds to acquire FPGA boards (see supplementary material for their support letter).

• **Microsoft Research.** Microsoft's contribution to the RAMP consortium has consisted of collaborating in the top-level design of the BEE3 system and funding the printed circuit and mechanical design using professional contract engineering firms. This produced a system with reduced cost and greater reliability than is generally possible in academia, and is scalable to production levels. It is Microsoft's conviction is that this model of cooperation between industry and academia is more effective than directly funding graduate students, since performing designs of this scale are not an effective educational tool.

Microsoft awarded BEEcube, Inc. a royalty-free license to manufacture and sell the design to any interested party, beyond just NSF-supported entities, and through this generous effort, makes the results produced by the RAMP consortium much more relevant to commercial or government entities. Microsoft is also contributing Verilog designs for some of the BEE3 infrastructure (e.g, the design for a controller for the BEE3's DDR2 RAMs) to the research community. Microsoft will also use the BEE3 systems internally for its own research in computer architecture and acceleration of algorithms, and exploration of future operating systems.

• **Intel-OML.** Intel's Oregon Microarchitecture Research Lab (OML) is actively engaged in microarchitecture research to exploit the continuation of Moore's Law. Currently OML is pursuing two main research thrusts: 1) *resilient microarchitecture* which requires investigation of techniques for large Vcc range in anticipation of future process scaling challenges and the implication of resilient microarchitecture on platforms, and 2) *memory hierarchy architecture*, which involves developing advanced microarchitecture techniques for achieving lower power memory sub-systems, as well as research into memory architectures suitable for potential future memory technologies such as MRAM, 3D-DRAM stacking and other flash memory replacement technology. OML employs FPGA platforms, such as the RAMP infrastructure, to achieve its research agenda.

• **Lawrence Berkeley National Lab (LBNL)-Climate Supercomputer.** LBNL is doing a speculative extrapolation of the performance aspects of an atmospheric general circulation model at ultra-high resolution and is exploring alternative technological paths to realize such a model in the relatively near future. The unparalleled requirements and scale of the project provide a rigorous crucible for RAMP concepts in a critical field on the forefront of HPC research. Due to a super linear scaling of the computing burden dictated by stability criterion, solving the equations of motion dominate the calculation at ultra-high resolutions. From this extrapolation, it is estimated that a credible kilometer scale atmospheric model would require at least a sustained ten petaflop computer to provide scientifically useful climate simulations. The LBNL design study portends an alternate strategy for practical power-efficient implementations of petaflop scale systems.

It is hoped that a customized embedded processor can be used, at acceptable cost and power considerations, in a machine specifically designed for ultra-high performance climate modeling. The major conceptual changes required by a kilometer-scale climate model are certain to be difficult to implement. Although the hardware, software, and algorithms are all equally critical in conducting ultra-high climate resolution studies, it is likely that the necessary petaflop computing technology will be available in advance of a credible kilometer scale climate model. RAMP is the only technology capable of rapidly evaluating hardware possibilities concurrently with algorithmic development.

2.3.5 University Adopters

• **University of Toronto-Paul Chow.** Prof. Paul Chow of the University of Toronto is investigating the use of multi-FPGA systems for large-scale computing applications. Under development is a programming model for multi-FPGA systems based on MPI, a common API used in supercomputing applications. 10 BEE2 systems that form a large FPGA cluster are currently used for application demonstrations. Molecular Dynamics (MD) was chosen as the initial driving application. Development originally started on a cluster of five FPGA development boards. The BEE2 cluster enabled scalability demonstrations of the MD application while a next generation BEE3 cluster should provide an additional factor of five to ten speedup.

• **Stanford University-Theresa Meng.** Prof. Theresa Meng of Stanford University is implementing a software/hardware co-design approach to learning Bayesian networks from experimental data that is scalable to very large networks. The

research work has demonstrated performance improvements over software in excess of four orders of magnitude. Through parallel implementation and exploitation of the reconfigurability of FPGA systems, scientists are able to apply Bayesian learning techniques to large problems, such as studies in molecular biology, where the number of variables in the system would overwhelm any state-of-the-art software implementation.

This approach has been adopted by biologists to investigate real-life problems in the context of systems biology, a fact that indicates the significant cross-disciplinary potential of easy-to-use infrastructures for the development of novel systems.

• **GSRC.** With support from DARPA, in late 2006 an initiative to supply and support BEE2 FPGA computing platforms for the Focus Center Research Program (FCRP), Gigascale System Research Center researchers was established. The objective of the initiative was to accelerate the research objectives of the various groups, to help develop a shared research platform, and to encourage collaboration. Cross-disciplinary usage of the RAMP-1 platform is expected to provide new perspectives of the research, broaden the hardware base, and expand the pool of future download-capable sites. A total of 14 BEE2 systems have been distributed for the following purposes: 3 for addressing architectural development in multicore and concurrent architectures, 2 for networking issues, 2 for reliability and fault emulation, 3 for purpose-specific processing, and 2 for SOC and chip design issues.

3 Technology Basis for this Infrastructure Proposal

The original RAMP award was a CRI award and, as such, did not include research. Within our research projects separate of RAMP, however, we have pioneered a series of techniques that show tremendous promise in the area of FPGA simulation. Before detailing the proposed work in Section 4, in this section we first describe the technology basis that the new RAMP infrastructure will be built on. They represent the key research outcomes and lessons learned by the PIs over the last three years.

3.1 User Interfaces

Modern full-system software simulators, such as Simics, provide a user experience identical (except in speed) to the target system. For example, when running Microsoft Windows on Simics, there is a window that displays a Windows desktop and allows the user to interact with the target desktop. There is another window that allows the user to interact with the simulator itself to print statistics, set different run modes and so on. In addition, such simulators provide interfaces that permit the simultaneous monitoring and debugging of both the target system as well as the simulator itself. Full-system software simulators are integral components of both ProtoFlex and FAST, providing them with a software simulator-like user interface. We have found that such interfaces make ProtoFlex and FAST especially attractive to users and especially easy to use. We want to design all future RAMP systems around this interface paradigm.

An key aspect of user-interface that has been blatantly missing in our earlier work is a convenient and intuitive mechanism for *debugging and state viewing*. The presence of FPGAs in RAMP platforms introduces a challenge that software simulators do not have. Debugging software is relatively easy because most software can be made completely transparent. Standard software debuggers permit single stepping the code, examining and setting the value of every variable and setting watchpoints and breakpoints. Unfortunately, the same is not true for hardware since there often isn't even a way to observe a desired value or to single step the hardware. Extra hardware can be added to the FPGAs to provide target observability and leverage the time control to make the RAMP user experience closer to a software simulator. However, there is still the issue of observing and debugging the FPGA-based simulator structures which is not addressed by target observability logic. Since bugs can occur anywhere, including in the observability logic, we need a strategy to debug the hardware itself.

The FAST group has been developing the FAST viewer that uses the inherent ability for a Xilinx device to dump its entire state and mapping that state to the original user signal names using information automatically generated by Xilinx place-and-route tools. Such a tool enables cycle-by-cycle stepping, viewing and setting of the entire FPGA state through either a waveform viewer or gdb, with minimal or no additional hardware cost for the observational ability. The FAST FPGA viewer can also provide a general interface to export statistics that do not require run-speed performance, such as counters that do not need to be sampled every cycle. Thus, by coupling the FAST viewer into the simulator front end, we can provide a seamless view of the simulator where the user might not even know if a particular component is running in software or on an FPGA.

It is even possible to process signals at run-speeds. This ability is an attractive dimension to FPGA simulation since filtering/compressing/trigger detection of signals and the subsequent decompression often dominates software simulation performance. For example, hardware to detect when a branch misspeculation occurs at the same time an L1 cache miss is outstanding can be introduced into the FPGA to continuously monitor for that situation. Traces can be compressed on the FPGA before export to minimize I/O and disk usage. Carefully specified APIs within modules and RAMP pipes that permit the export of statistics into processing logic and/or a statistics network can make the use of such capabilities a matter of specifying the appropriate connections during instantiation.

3.2 Modularity and Composability

Both FAST and HAsim have been designed as simulators since their conception and, as such, assume that timing models that are partitioned into *modules* with well specified (but flexible) interfaces that are connected together by *pipes* that provide communication, time tracking, and statistics gathering capabilities. Ideally, a specific pipe implementation can be used to connect together any module, requiring pipes to be able to be passed a data type or a union of data types that will indicate what sort of data will be communicated.

There are complex issues surrounding such a partitioning, including the basic semantics of the pipe/module interfaces, what simulator and target functionality sits in a module versus a pipe, how to ensure freedom from deadlock, and how to name instances of each module in statistics traces. For example, there are significant ramifications of (mostly) modeling time in pipes rather than in modules. What that implies is that modules behave the same with the same sequence of inputs, regardless of how time proceeds around it. Doing so, however, ensures that modules can be used in a wide range of timing situations without modification. These module interface and pipe issues have been extensively discussed and developed within the RAMP Description Language (RDL) effort as well as within the FAST group (BaseTypes and FAST Connectors) and the HAsim group (A-Ports).

HAsim and FAST both use Bluespec [19, 20], a high-level hardware description language (HDL), to specify hardware. Bluespec has many advantages over standard HDL including the ability to pass types, including union types, as parameters to a module. Both HAsim and FAST use that capability to create truly generic pipes.

Conceptually, one can easily assemble a system from a sufficient set of modules that support a compatible set of interfaces and a set of pipe implementations. In reality, selecting and instantiating the various modules, either for a hardware or a software implementation, can be a tedious and error-prone task that is best not done by hand. The Architect's WorkBench (AWB), released as GPL code by Intel to support RAMP, is a mature solution to this problem (having been used at DEC/Compaq/Intel). AWB provides the ability to register an arbitrary number of entities residing in an arbitrary number of separate repositories, and allows the user to select and configure each of the components needed to assemble a complete system. AWB then automatically checks out the appropriate versions of each component, generates glue files that instantiate and connect those components and builds the simulator.

3.3 Host-level Techniques : Architecture-Level FPGA Emulation Approaches

Over the last few years, the participants in the RAMP project have explored a range of approaches to architecture-level FPGA emulation. Simulators of each component can be built using several different strategies, which differ in how functionality and timing are modeled. It is also possible to connect models that use different emulation styles when building a full system model.

• **Direct RTL Mapping.** In this approach, the RTL design of a block given in a standard hardware description language is synthesized directly into FPGA gates. In effect, the RTL model provides both functionality and timing information and requires little rework of an existing design. However, RTL designs intended for conventional ASIC fabrication often map poorly to FPGAs, requiring large area and with slow cycle times. To improve emulator capacity, the RTL design usually requires rework to use less FPGA resources. We have observed a factor of 3–5 reduction in area and factor of 2–3 improvement in cycle time with an FPGA-conscious mapping versus the initial ASIC RTL. A complete RTL model also requires a large design effort to change any target system details or to add new tracing or debugging functionality. RAMP Blue was the first RAMP example of direct RTL mapping.

• **Host Multithreading.** Host multithreading is a technique to improve utilization of FPGA resources. When the target system contains multiple instances of the same component (e.g., multiple processors in a manycore design), the host model can be designed so that one physical FPGA pipeline can model multiple target components by switching processing between each target. For example, one could interleave targets on a cycle-by-cycle basis, thus trading time (more cycles between the execution of a specific core) for space (just one pipeline.) One advantage of this approach is

that it can hide emulation system communication latencies that may otherwise slow the emulation. For example, while one processor target model is making a request to a memory module, the activity of 63 other target processor models can be interleaved. Provided modeling of the memory access takes fewer than 64 FPGA clock cycles, the emulation will not see a stall. Multithreaded emulation adds additional design complexity versus a pure direct RTL approach, but can provide a significant improvement in emulator throughput. It is possible that a tool could be developed to automatically generate host-multithreaded versions of an RTL design. ProtoFlex was the first RAMP example of a multithreaded host system.

• **Split Functional/Timing Models.** Partitioning a simulator into a functional model and a timing model was first done in software architectural simulators. A functional model calculates the results of each operation (i.e., an IEEE floating-point multiply), but does not predict timing. A separate timing model is constructed to model the number of clock cycles required for a flow of operations to propagate through the target machine pipelines. The advantage of this approach is that most of the complexity in a design is the functional model, and thus can be reused to model many different target systems with different timing behaviors by only changing the much simpler timing model. A further advantage of this split timing/functional approach is that it can flexibly trade area versus emulation speed. For example, whenever a target machine floating-point unit (FPU) is stalled, the corresponding FPU timing model does not need to activate an FPU functional model, thus improving FPGA utilization by sharing the functional models across multiple timing models. A direct RTL mapping would require a complete FPU for each FPU in the target design, and these would simply sit idle if the corresponding target FPU pipeline was stalled. FAST was the first RAMP example of a partitioned functional/timing simulator.

• **Hybrid Techniques.** The three general techniques can be combined in various ways depending on the emulator goals. For example, a processor model might use both split timing/functional models and host multithreading to gain advantages in both simulator throughput and flexibility. Alternatively, different components might be modeled in different ways. For example, multiple direct RTL implementations of a processor might be connected to a highly-multithreaded memory system model. In another example, ProtoFlex implements the processor model in FPGA for accelerated simulation but still relies on software simulation to provide the full system behavior (e.g., I/O devices) [12]. ProtoFlex further utilizes the *transplant* technology to allow the CPU model to execute in FPGA for the common-case instructions but fall back to a software simulated execution for complicated rare instructions.

3.4 Power and Thermal Modeling

Power and thermal modeling is an important new focus introduced into RAMP since the previous NSF-CRI proposal. The recent NSF-funded work by Jose Renau (a PI) has built the basic infrastructure required to measure temperature and estimate power consumption from AMD chips. The infrastructure addresses some of the difficulties associated with the validation of power and thermal models by generating a detailed power breakdown (leakage and dynamic) for each processor floorplan unit. These measurement capabilities have resulted in the demonstration that some commonly used commercial and academic thermal models have significant errors and the subsequent implementation and validation of new thermal models. We will expand on the integration of power and thermal model in the proposed infrastructure in Section 4.1.5.

4 Proposed Work

We propose to (i) enhance RAMP to develop a composable, uniform and accessible common infrastructure — including standards, tools, base modules, and remote access capabilities — that will form the foundation of an easy-to-use RAMP, (ii) adapt simulators from our other research work to the common RAMP infrastructure and make those adapted simulators release-ready through extensive testing, using them for class labs within our own institutions and sharing amongst other RAMP PIs and (iii) outreach to a wide community to ensure those that might benefit from RAMP have the necessary knowledge and support to do so. The work will be done in a distributed fashion at the various PI institutions, ensuring the systems and infrastructure work beyond the developing institution.

The overarching goal of RAMP is to make fast, easy-to-use, easy-to-modify, easy-to-instrument, scalable computer system simulators available to as many users as possible. Originally, we had incorrectly assumed that much of our target user base would be have backgrounds and experiences roughly similar to ours. Over the past three years, we have discovered that there are at least four different classes of users that are enumerated below.

- Users who are not interested in modifying hardware but want a platform that models tomorrow's machines for

software development.
- Users that would like to start with a predefined target machine model and tweak hardware parameters, but retain the basic machine structure.
- Users who would like to start with an existing machine model and make large structural changes to the architecture or to key components. For instance, the user may want to start with a target model for a parallel processor with cache-coherency and change the memory hierarchy to support transactional memory.
- Users who require an FPGA platform with a base set of firmware on top of which to build new components or special-purpose processing elements. (The RAMP PIs are in this class of users.)

Of course, users may fall into different categories at different times or even at the same time. Also, even within a particular category, there are different levels of experience. For example, any of these users may be students working on such platforms for the first time. Each category of users have different requirements in terms of host platforms, target functionality, control, monitoring, documentation, and training. We are proposing to develop the infrastructure necessary to support all users, from the most hardware savvy to the least.

In summary, we propose to do the following.

- We propose to create a baseline infrastructure, adapted from our research and RAMP work, consisting of the specifications and reference implementations of module interfaces, communication pipes, the user interface, monitoring/debugging, power/thermal modeling, and base modules.
- We propose to use those specifications and reference implementations to build several full simulators, including ones that model 1024+ nodes.
- We propose to aggressively outreach to several communities of users, ranging from hardware architects, through implementers, all the way to software developers through classes, on-line and live tutorials, and by providing all of the necessary resources as a freely available, full-time staff supported service on the Internet.

In the rest of this section, we provide more details of these basic components of our proposal: infrastructure, models, and outreach.

4.1 Proposed Infrastructure

The proposed RAMP infrastructure will provide the foundation on top of which RAMP models will be built.

4.1.1 Host Platforms

In addition to the BEE3, we plan to use Intel Front Side Bus (FSB) ACP FPGA systems and the Xilinx XUP board. The ACP system is a four processor socket system where one or more sockets can be populated with up to four Virtex 5 330 FPGAs, offering approximately three times the FPGA capacity of a BEE3 board per processor socket. Processor-FPGA communication takes approximately 100ns and is capable of 8GB/sec. The FPGA can also access all system memory (128GB of capacity.) Such systems are especially suited for ProtoFlex, FAST, and HAsim systems that rely on a general-purpose processors. The Xilinx XUP, on the other hand, is a low cost board ($600) that is found at most universities. **We propose to define a RAMP hardware platform interface specification that will permit any reasonable FPGA platform to act as a RAMP FPGA host and provide the appropriate shims for BEE3, ACP and XUP so that they adhere to that specification.**

4.1.2 User Interface

Providing a familiar, easy-to-use interface to RAMP simulators would make them far more accessible to far more users, especially those that are not accustomed to traditional architectural level simulators. Since both ProtoFlex and FAST incorporate full-system software simulators we have considerable experience providing such user interfaces. **We propose to use our experience to define a standard RAMP user interface (RAMP UI) for simulation control, debugging, and monitoring that can be used to provide a consistent user experience across all RAMP platforms.**

4.1.3 Modularity, Communication, and Composability

We are relying on aggressive modularity to realize the RAMP vision of rapid development of FPGA-based simulators and prototypes. In order to support modularity, modules of a particular type are required to support a common interface. Interfaces are point-to-point through which RAMP modules send messages. The module can access all fields

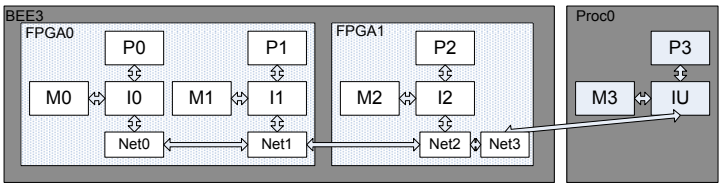

Figure 2: Processor (P), Chipset (I), Memory (M) and Network (Net) modules connected with RAMP pipes. Notice that the implementation technology of each component might be different (P3/I3/M3 are implemented in software on a processor) and the partitioning might be different (0 and 1 are on a single FPGA while 2 is on a different FPGA on the same BEE3 board), but the design remains the same.

of the interface structure both as it creates and populates a message as well as when it receives a message. Thus, the interface hides implementation realities, such as a communication channel that is narrower than a full message. The interface will also be designed to avoid requiring the module to model target time.

Of course, each individual RAMP system might require a set of specialized interfaces to achieve specific research goals in which case a specialized interface may need to be written. However, all interfaces will use a base type that provides the minimum signals necessary to support the underlying infrastructure.

Each pair of RAMP interfaces is connected by a RAMP pipe that implements the actual communication channel, models time, and provides statistic, monitoring, and debug capabilities. RAMP pipes will be implemented in a generic way, providing the ability to connect any pair of compatible RAMP interfaces. Various RAMP pipe implementations will be provided to enable communication within an FPGA, across FPGAs, within software running on a single host core, between software running on different host cores and between software and an FPGA. RAMP pipes hide implementation constraints, such as a physical communication channel that is narrower than an interface message. Thus, the module writer only needs to communicate through a pipe to ensure the module can be run on any host platform. See Figure 2 for an example.

Thus, we propose to coordinate, document, and release our improved infrastructure through specifications and reference designs for (i) a set of canonical module semantics and interfaces, (ii) pipes for communication between modules, and (iii) debugging/monitoring mechanisms. All of these specifications and reference designs will draw heavily from our experience and code developed in our previous and current research projects, including FAST, RDL, HAsim and RAMP White.

4.1.4 Base Module Library

Given a reasonable set of composable and configurable base modules such as CAMs, caches, FIFOs, arbiters, forks, joins, switch fabrics, and memories, as well as common structures built on top of those base structures such as branch predictors, one can very quickly assemble realistic systems. The only modules that would need to be written are those that differentiate the target architecture.

Many of these modules exist, even in composable form, within various PI research projects such as FAST and HAsim. **We propose to adapt and develop a base module library that will adhere to the RAMP module interface specification and use RAMP pipes.**

4.1.5 Power/Thermal Modeling

The original RAMP plan did not adequately address target power prediction. For that reason, we have added Jose Renau of UC Santa Cruz, a pioneer in power and thermal measurement and modeling, to the RAMP team and as a co-PI on this proposal. We propose to build a power/thermal infrastructure for RAMP. The proposed infrastructure has two major components: an infrastructure to export data necessary for power/thermal modeling from FPGAs and a GPU-accelerated thermal simulation. In addition, we propose to make available the infrared infrastructure developed in previous projects to build shared thermal and power models calibrated with commercially available processors.

FPGA Power/Thermal interface. Digital systems dissipate power through leakage and through dynamic switch power. Leakage power is temperature dependent and dynamic power is activity dependent. **We propose to define and build an interface that will export target activity from the FPGA to the monitoring computer to compute**

the dynamic and leakage power. The monitoring computer is a desktop with a GPU that computes the power consumption and thermal statistics of the target simulated system.

Current architectural simulators like Wattch [21] and SESC [22] count the activity in each of the major blocks of the modeled system. For each block, a given amount of dynamic energy is charged each time that it is utilized. If the block is unused, different clock gating policies could be used to estimate the dynamic power. The key difference of the proposed approach is that the activity rate is computed by the FPGAs and transfered to the power/thermal monitoring computer which computes the dynamic and leakage power.

As is done in current architectural simulators, the data only needs to be transfered to the power/thermal simulator every few thousand cycles. To maximize the infrastructure applicability across different projects, we plan to define and build a flexible gateware where each block reports the activity rate for a programmable time interval. At every interval, the per block activity would be transferred to the monitoring computer that then computes the dynamic and leakage power.

There is also a feedback loop between temperature and operating system that needs to be modeled. If the processor is too hot, different scheduling policies or even emergency thermal shutdown may be triggered, affecting performance and power. To provide this feedback, we propose to build a thermal sensor model. The power/thermal monitoring computer knows the temperature at any die position. The thermal sensor model running in the timing model will read the temperature from the monitoring computer. The resulting system can correctly model different architectures performance, power consumption, and transient thermal and their interactions.

GPU Accelerated Thermal Simulation. Obtaining a dynamic thermal transient for given a power trace is a compute intensive process where over 90% of the thermal simulation time is spent solving a single precision system of equations. Modern processors running power models are fast enough for traditional architectural simulators, but not fast enough for a RAMP system. We propose to adapt the thermal solver algorithm to run on graphics processing units (GPUs). Modern GPUs such as the nVidia GTX280 have over one hundred single precision floating point units. By converting our existing thermal simulation to CUDA, we expect to improve performance by an order of magnitude.

4.1.6 RAMP-as-a-Service

A goal of RAMP is to minimize both the monetary and time cost-of-entry to FPGA-based simulators. Unfortunately, FPGA platforms and their tool chains are currently expensive and difficult to installing and maintain. Ideally, their cost, setup, and maintenance are shared. **Thus, we propose to build and support the infrastructure necessary to provide RAMP, including the tool chain and FPGA platforms, as an Internet-based service.** This new paradigm will consist of two major components: a virtualized build environment that can be run either locally or as a service, and remotely accessible FPGA platforms. Rather than having to obtain and install all of the various components from tools to software to design sources, a user would instead download a single preconfigured virtual machine that contains not only the design bitfiles, but also all of the source code, tools, configuration files, tests, validation suites, and benchmark software. Users will also be provided with remote access to a set of commercial tools and machines on which to run those tools, avoiding the need to acquire, install and maintain those tools on the user's local machines.

In addition, BEE3 platforms will be made available as a resource where jobs can be run, eliminating the necessity of acquiring and maintaining these expensive boxes. We will build and put online a set of FPGA platform clusters, to be used initially within the RAMP collaboration and later to be supported for universal internet access. The largest cluster, procured by a major donation from Xilinx, will be sited at UC Berkeley. It will comprise 16 BEE3 modules, with the capacity to simulate in excess of 8K processors cores with our planned simulation approach. Other smaller clusters, acquired with funds from this proposal and FPGA devices donated by Xilinx, will be distributed geographically, but made available online through a common interface.

4.2 Reference Designs

The original RAMP vision relied heavily on the idea of direct mapping existing RTL models to FPGAs. While this approach has allowed us to quickly validate the approach through RAMP Blue, in practice the approach has the following problems: (i) synthesizable RTL models for future systems do not exist yet (ii) the available synthesizable RTL models that do exist consume significant FPGA resources since they are not optimized for FPGA implementation (iii) the existing models are inflexible and poorly instrumented, and (iv) except for the case of very simple processor cores, scaling to a large number of cores is expensive.

For these reasons, we have moved toward processor virtualization, host multithreading, and split functional/timing

that providing a flexible means to easily trade-off target core density for simulation performance. Such space saving techniques coupled with appropriate shimming to provide an abstract hardware interface ensure that RAMP is FPGA platform independent. In fact, we currently use the XUP, an inexpensive Xilinx development board intended for teaching purposes, as a RAMP platform.

As part of their other research projects, the RAMP PIs have developed several simulators that encapsulate many of the RAMP principles. In their current state, however, these simulators are still research vehicles and thus are not yet ready for general distribution. **Thus, we propose to "harden" Red (Stanford), ProtoFlex (CMU), FAST-MPLite (Texas), and Gold (Berkeley) into release-ready versions.** In addition to enhancing their functionality and hardening them through code cleanup, extensive testing and sharing amongst ourselves, we will also ensure that they conform to our new module interface and RAMP pipes, integrate them with the RAMP UI, and then to make these available in the repository and/or as runnable models on the service.

Specific efforts toward preparing the reference simulators are detailed below.

• **RAMP Red.** RAMP Red2, to be developed at Stanford University, will provide a shared-memory multiprocessor with an enhanced memory system. The focus of the original Red design has been on hardware support for transactional memory (TM) and the development of TM-based tools (programming models, debuggers, optimization frameworks). This proposed will build upon our experience using the original model as a research vehicle for over two years and will focus on the develop of a model with *scalability* and *flexible interfaces* in mind. Building upon the techniques discussed earlier (host multithreading and split functional/timing models), we will scale Red2 from eight to hundreds of processors. The model components will follow the plug-and-play interfaces to maximize reuse and flexibility. To the extend possible, the components will be instruction set agnostic in order to support both the SPARC and PowerPC ISAs.

As part of the modeling process, we will integrate a recently published, scalable implementation for TM. We will also decompose TM functionality into its basic components (atomicity, isolation, and ordering) to enable the use of the hardware support for research activities beyond concurrency control (security, reliability, productivity, etc). Finally, we will scale up the Linux for Red2, to support hundreds of cores and provide hardware-assist for adaptive thread management. We will package the Linux environment separately to allow reuse with other modeling efforts by RAMP users. Overall, the new model will provide a flexible framework for software and hardware research on scalable, shared-memory systems. Users will be able to easily set the scale of the system to match the capabilities of the FPGA platform and configure the supported hardware features according to their research needs.

• **ProtoFlex.** While the ProtoFlex technology continues to be under active development, the existing BlueSPARC FPGA-accelerated simulator can already offer many utilities to the multicore architecture research community. As a part of the proposed RAMP collaboration, we will deploy BlueSPARC as one of the target systems available on the public RAMP simulation cluster. The effort to deploy BlueSPARC will begin with porting the current BEE2-based BlueSPARC to the BEE3 platform. This initial effort will make available to the public a fast multiprocessor simulation alternative equivalent in function to the Simics/Sunfire software simulator. The deployment effort continues with integrating BlueSPARC as a system-state checkpoint generator for use with SimFlex, a publicly available multiprocessor simulation system (developed with support from CCR-0509356) [23][4]. Lastly, we will extend BlueSPARC with instrumentation interfaces to export important execution traces (instruction and memory addresses, branch history, etc) to drive complementary trace-driven simulation environments (whether hardware or software-based). Documentations and tutorials will be prepared to facilitate public adoption of BlueSPARC as a research vehicle.

• **FAST-MPLite.** FAST-MPLite is a version of FAST that sacrifices some accuracy for a significant boost in performance by eliminating checkpoint and rollback, while maintaining its full-system and multiple current ISA capabilities. FAST-MPLite will provide users with the ability to perform MP simulations on the x86/PowerPC ISA running unmodified applications running Windows (x86) or Linux in the 10s to 100s of MIPS aggregate performance.

As part of the proposed work, we will harden FAST-MPLite, port it onto the Xilinx ACP platform, incorporate the scalable network infrastructure from RAMP White into FAST-MPLite as its memory hierarchy and network timing model as well as incorporate key components from the RAMP White port of MP Linux. As part of the hardening process, we will use it for labs in a graduate class under close supervision. We will make it available on RAMP-as-a-Service.

• **RAMP Gold.** The funds for the current infrastructure proposal will be used to extend RAMP Gold and make a supported version available, both as a BSD-licensed source code distribution and as a supported model on the externally

[4]SimFlex employs simulation sampling methodology to reduce microarchitecture simulation time by many orders of magnitude. An accurate performance estimate is constructed by performing many short microarchitecture measurements starting from randomly selected checkpointed system states.

accessible BEE3 cluster. In addition, the infrastructure grant will fund the packaging of RAMP Gold components for use within other RAMP target models. Since RAMP Gold is currently in the process of being developed, we will ensure that it adheres to the RAMP module interface and pipe specifications. We will also incorporate modules from other RAMP projects.

• **IBM/MIT OpenPowerPC.** IBM Research is co-developing a flexible PowerPC processor in Bluespec and plan to donate the design to the RAMP project (see IBM letter of support.) The processor was designed for maximum configurability of almost any parameter including issue width, pipeline delays, branch predictors and so on. The processor supports the entire PowerPC ISA and will boot Linux. We propose to harden the processor and integrate it into the rest of the RAMP infrastructure, including ensuring conformance with the RAMP module interface and integration with RAMP pipes.

4.3 Proposed Educational and Research Outreach

We expect that a fully developed RAMP infrastructure will significantly enhance the educational environment at universities. Education occurs both within the classroom and, particularly for graduate students, outside in less formal settings. In this section we will elaborate on how we expect RAMP to be used in both of these environments.

Student training. Easy access to appropriate tools do more than just enable research, they serve as the genesis of it. Witness SimpleScalar [24]. CiteSeer reports over 351 scholarly publications referencing its use. SimpleScalar spawned an entire generation of graduate students trained on superscalar processor research.

With the coming age of chip multiprocessors a new generation of graduate students must be trained. This new generation of fresh PhD's are needed by the computing industry to solve the programming, architecture, and systems challenges desktop MPP systems will have. Studying large scale MP systems purely in simulation is intractable. By being both fast and accurate, RAMP is a platform that will enable this training and spawn new research.

In addition to graduate student research, if we are funded we will apply for REU grants so that undergraduates can participate in the research to realize the RAMP vision.

Classes. While student training and spawning research is an important goal of the RAMP project, facilitating classroom education is equally important. Classes serve to educate beyond the specialists in the area and create a broad group of professionals experienced with MPP systems. Here in this section we'll limit ourselves to discussing the impact on only the three most related classes taught at undergraduate and graduate levels. These are architecture, parallel systems, and digital systems design.

- Architecture: Current undergraduate and graduate architecture classes are limited by the available infrastructure. Those that make use of FPGAs lack the ability to teach about the "whole system" including architecture / operating system interaction. Those taught using only software based simulation tools lack the ability to impart to students the true complexity, design challenges, and hardware opportunities available.

  RAMP would change this. Starting from the base RAMP system the PIs included on this proposal will develop classroom projects designed to teach about the interaction of the operating system and architecture as well as the design and tuning of real microprocessors. Since a community of educators will be built that utilize the same base hardware and software platform this community can share classroom exercises and thereby gain leverage.

- Parallel systems: If parallel computing classes are taught at all in universities, they are not using the systems of the future, but make do with the computers of the past. Parallel architecture is often taught as a "history lesson", focusing on reading papers about significant systems. Parallel programming is forced to make do with whatever make shift parallel system the educators have access to—a cluster of workstations and/or a small SMP. The scope of class examples, assignments, and projects is limited to small-scale uniprocessor systems.

  RAMP can make a significant impact on the teaching of parallel systems. First, for parallel architecture classes it will enable advanced class projects that explore the processor, bus, coherence protocol, and I/O storage system of MPP systems. For parallel programming classes the availability of canned "model" RAMP systems with complete infrastructure: Linux operating system, GNU tools, etc., facilitates class room projects built around software development for future instead of historic parallel computers. Moreover, RAMP will allow both architecture and programming courses to focus on large-scale systems and provide students with hands-on experience on the real challenges with tens, hundreds, and thousands of nodes in a parallel system.

Being educators as well as researchers, the PIs are keenly interested in facilitating classroom experience with MPP systems. To this end, we intend to make the base RAMP system components as parameterizable as possible. This will enable students to explore variations on the base system design through easy change of the parameters. Furthermore, the core design standard facilitates a compartmentalized design approach. This will enable student projects that alter and extend the base RAMP system with minimal impact on the entire system fragility.

- Digital systems design: Advanced undergraduate and graduate level digital systems design is taught with FPGAs. RAMP has the potential to provide a canned system with tested infrastructure to serve as a starting point for these classes. The PIs on this proposal teach digital systems design and will develop a number of integrated systems projects that can be shared with the entire RAMP community.

- Broader educational impact: Once RAMP is fully adopted in these "core" related areas, we see it achieving a broader impact throughout a department's curriculum. In particular, applications classes such as databases, graphics, and AI should be training undergraduates on how these areas will be impacted by future desktop MPP systems. With RAMP available as an easily accessible service on the web, these classes can use it to explore how their respective algorithms and software artifacts execute on models of future processing systems.

Research Outreach. Although the universities involved in the development of RAMP and the initial group of external supporters come from the usual leading research institutions and industrial research labs, we believe online access to large RAMP systems will allow a much wider range of departments and companies to have easy access to massively parallel processor for use in courses and research. The RAMP systems we plan to support online will have the capacity to simulate processor architectures in excess of thousands of CPU cores. In addition to online access, departments and labs willing to make the investment in their own FPGA platforms, will be able to take our hardened target MPP models and install them locally.

We are also developing the RAMP infrastructure for those with more modest needs. We are building the RAMP infrastructure to be FPGA platform independent. Inexpensive FPGA development boards such as the XUP board from Xilinx (around $600) provide an inexpensive platform for simple experiments.

In addition to broadening the types of departments that have access to the large scale parallel processing, the RAMP project will also engage underrepresented groups. Many students developing RAMP systems are from underrepresented groups.

Tutorials. We already have experience running tutorials and hands-on workshops at the International Symposium on Computer Architecture (ISCA) and Architectural Support for Programming Languages and Operating Systems (ASPLOS) conferences. We propose to offer RAMP tutorials over the next three years at ASPLOS conferences and ISCA conferences. We see ISCA as an opportunity to enlist the hardware savvy to help use extend the basic infrastructure, extend our component library, and reference target models. ASPLOS is an opportunity to attract adopters from a wider community of computer systems researchers and educators. To attract an even wider audience of users we plan to run introductory tutorials at other computer systems conferences which focus on operating systems, networking, and embedded systems. We are also plan to offer a series of web cast tutorials and workshops. Our planned RAMP-as-a-service infrastructure is an ideal way to enable distance learning with a hands-on experience. Under our guidance, attendees will log onto our machines and experience the RAMP infrastructure without the hassle of acquiring FPGA platforms and obtaining and installing software on their machines. For all our workshops and tutorials we will be collecting surveys from the attendees to help us evaluate user satisfaction and feedback. A preliminary tutorial schedule is shown in Table 1.

openRAMP.org. Our intent is to model our project after an open source software development where a vibrant community of developers and users come together to develop and use a set of software tools. These collaborattions are usually organized around a website that serves as a point of information for new developers and users, and a repository of the current state of development. We will enable our community similarly with a website we will call openRAMP.org. With this web presence we will provide a well maintained and documented repository for RAMP tools, component libraries, and reference designs, along with a monitored news group, and links to RAMP related activities. For the first few years, the co-PIs of this proposal will have a strong presence on the website, and it will be maintained by RAMP staff. We expect that within a few years, other members of the community will step up as leaders. openRAMP.org is particularly important while we are still in the early phases of development, as it is important that we enlist people with FPGA development experience in help build up the component library. openRAMP.org, along with

our planned workshops and tutorials, will help us nurture this community of developers. Furthermore, the website will provide a convenient means for us to collect feedback from users and developers to help us evaluate their satisfaction.

4.4 Milestones and Management Plan

This project is a collaborative effort with faculty from six universities along with two industrial researchers. All faculty participants, with the exception of Jose Renau, are members of the original RAMP collaboration. Dr. Shih-Lien Lu is a Senior Researcher at Intel's Microprocessor Technology Laboratory, and Dr. Joel Emer is an Intel Fellow and Director of Microarchitecture Research at Intel's Digital Enterprise Group. Both will participate as unfunded collaborators. Dan Burke from UC Berkeley will participate as Senior Personnel and will be the technical project manager. Dan has spent his career designing, building, and supporting scientific instrumentation. He has worked as RAMP project engineer for the past three years.

Prof. Arvind of MIT will be a subawardee to Prof. Derek Chiou at the University of Texas at Austin.

For most of us, we have had three years of practice working together. With very few breaks, the co-PIs have held regular bi-weekly teleconferences to coordinate our activities. We also meet in person four times a year, twice at our research retreats, and twice at computer architecture conferences. The entire RAMP community meets twice annually at our two day long research retreats. These retreats are a very effective mechanism for students and others working on the project to hear about progress at other universities and for all to get feedback and advice from industrial visitors and trusted advisers from other universities.

Our major project milestones and dates are summarized below in Table 1. The milestones are organized into the three major categories of work we are proposing: *services, infrastructure, and outreach*. The primary services and infrastructure release schedules will follow industry practice; start with a limited (or alpha) release first and incorporate feedback from early adopters into later releases. The final column in the table list the co-PI responsible for each milestone. Though a specific PI is in charge of each milestone, we expect significant contributions from all the members of the group.

Table 1: Major Project Milestones

Date	***Services***	***Infrastructure***	***Outreach***	*PI in Charge*
2Q09			openRAMP.org online	Wawrzynek
	SW service online for internal use			Wawrzynek
	Small BEE cluster online at Berkeley (6 BEE3s)			Wawrzynek
		Component Interfaces standard released		Emer, Arvind
		Front-end / user interface standard released		Hoe
		Debugging / monitoring standard released		Chiou
			Conf. Tutorial (ISCA)	
			Summer Retreat (UT)	
3Q09	SW service online for limited external use			Wawrzynek
		ProtoFlex alpha release		Hoe
		Power/Thermal modeling standard released		Renau
			Webcast Tutorial	
4Q09	SW service online for unlimited external use			Wawrzynek
	BEE cluster online at MIT			Arvind
	BEE cluster online at CMU			Hoe
	BEE cluster online at UT			Chiou
	BEE cluster online at Stanford			Kozyrakis
	Large BEE cluster online at Berkeley (16 BEE3s)			Wawrzynek
		FAST-MPLite alpha release		Chiou
		RAMP Red alpha release		Kozyrakis
		OpenPowerPC alpha release		Arvind
1Q10	Distributed cluster service for internal use			Wawrzynek
		Accelerated thermal release		Renau
		RAMP Gold alpha release		Asanovic
			Winter Retreat (SF Bay)	
2Q10	Distributed cluster service for external use			Wawrzynek
		Power modeling release		Renau
			Conf. Tutorial	
3Q10			Webcast Tutorial	
1Q11			Winter Retreat (SF Bay)	
			ASPLOS Tutorial	
2Q11		ProtoFlex 1.0 release		Hoe
		Power/Thermal 1.0 release		Renau
			Conf. Tutorial	
			Summer Retreat	
3Q11			Webcast Tutorial	
4Q11		FAST-MPLite 1.0 release		Chiou
		RAMP Red 1.0 release		Kozyrakis
		RAMP Gold 1.0 release		Asanovic
		OpenPowerPC 1.0 release		Arvind
			Winter Retreat (SF Bay)	
1Q12			ASPLOS Tutorial	

References

[1] John Wawrzynek, David Patterson, Mark Oskin, Shih-Lien Lu, Christoforos E. Kozyrakis, James C. Hoe, Derek Chiou, and Krste Asanovic. Ramp: Research accelerator for multiple processors. *IEEE Micro*, 27(2):46–57, 2007.

[2] Chen Chang, John Wawrzynek, and Robert W. Brodersen. BEE2: A High-End Reconfigurable Computing System. *IEEE Des. Test*, 22(2):114–125, 2005.

[3] N. Njoroge, J. Casper, S. Wee, Y. Teslyar, D. Ge, C. Kozyrakis, and K. Olukotun. ATLAS: A Chip-Multiprocessor with Transactional Memory Support. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, Nice, France, April 2007.

[4] S. Wee, J. Casper, N. Njoroge, Y. Tesylar, D. Ge, C. Kozyrakis, and K. Olukotun. A Practical FPGA-based Framework for Novel CMP Research. In *Proceedings of the 15th ACM/SIGDA Intl. Symposium on Field Programmable Gate Arrays (FPGA)*, Montery, CA, February 2007.

[5] Alex Krasnov, Andrew Schultz, John Wawrzynek, Greg Gibeling, and Pierre Droz. Ramp blue: A message-passing manycore system in fpgas. *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 54–61, 27-29 Aug. 2007.

[6] Daniel Burke, John Wawrzynek, Krste Asanovic, Alex Krasnov, Andrew Schultz, Greg Gibeling, and Pierre-Yves Droz. RAMP Blue: Implementation of a Manycore 1008 Processor System. In *Proceedings of the Fourth Annual Reconfigurable Systems Summer Institute*, Urbana, Illinois, July 2008.

[7] http://www.gaisler.com/.

[8] Derek Chiou. FAST: FPGA-based Acceleration of Simulator Timing Models. In *Proceedings of the Workshop on Architecture Research using FPGA Platforms, held at HPCA-11*, February 2005. http://cag.csail.mit.edu/warfp2005/submissions/25-chiou.pdf.

[9] Derek Chiou, Dam Sunwoo, Joonsoo Kim, Nikhil A. Patil, William H. Reinhart, D. Eric Johnson, and Zheng Xu. The FAST Methodology for High-Speed SoC/Computer Simulation. In *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, November 2007.

[10] Derek Chiou, Dam Sunwoo, Joonsoo Kim, Nikhil Patil, William Reinhart, Eric Johnson, Jebediah Keefe, and Hari Angepat. FPGA-Accelerated Simulation Technologies (FAST): Fast, Full-System, Cycle-Accurate Simulators. In *Proceedings of MICRO*, December 2007.

[11] http://www.ece.cmu.edu/~potoflex.

[12] Eric S. Chung, Eriko Nurvitadhi, James C. Hoe, Babak Falsafi, and Ken Mai. A complexity-effective architecture for accelerating full-system multiprocessor simulations using FPGAs. In *Proc. International Symposium on Field Programmable Gate Arrays*, February 2008.

[13] Michael Pellauer, Muralidaran Vijayaraghavan, Michael Adler, Arvind, and Joel Emer. A-ports: an efficient abstraction for cycle-accurate performance models on fpgas. In *FPGA '08: Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, pages 87–96, New York, NY, USA, 2008. ACM.

[14] Michael Pellauer, Murali Vijayaraghavan, Michael Adler, and Joel Emer. Quick performance models quickly: Timing-directed simulation on fpgas. In *ISPASS '08: Proceedings of the International Symposium on Performance Analysis of Systems and Software*. IEEE, April 2008.

[15] Simflex: Fast, accurate & flexible computer architecture simulation. http://www.ece.cmu.edu/~simflex/flexus.html.

[16] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *Computer Architecture News*, 2005.

[17] Nathan L. Binkert, Erik G. Hallnor, and Steven K. Reinhardt. Network-oriented full-system simulation using M5. In *Proceedings of Sixth Workshop on Computer Architecture Evaluation using Commercial Workloads*, February 2003.

[18] Greg Gibeling, Andrew Schultz, and Krste Asanović. The RAMP Architecture & Description Language. In *Proceedings of the second Workshop on Architecture Research using FPGA Platforms, held in conjunction with HPCA-12, Austin, TX*, February 2006.

[19] Lennart Augustsson, Jacob Schwarz, and Rishiyur S. Nikhil. Bluespec Language definition, 2001. Sandburst Corp.

[20] Inc. Bluespec. Bluespec(tm) SystemVerilog Reference Guide, 2004. Description of the Bluespec SystemVerilog language and libraries.

[21] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a Framework for Architectural-Level Power Analysis and Optimizations. In *International Symposium on Computer Architecture*, pages 83–94, Jun 2000.

[22] Jose Renau, Basilio Fraguela, James Tuck, Liu Wei, Milos Prvulovic, Luis Ceze, Smruti Sarangi, Paul Sack, Karin Strauss, and Pablo Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[23] Thomas F. Wenisch, Roland E. Wunderlich, Michael Ferdman, Anastassia Ailamaki, Babak Falsafi, , and James C. Hoe. SimFlex: Statistical sampling of computer system simulation. *IEEE Micro*, 26(4), July/August 2006.

[24] Doug Burger and Todd M. Austin. The simplescalar tool set, version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin, Madison, June 1997.