BEE3: Revitalizing Computer Architecture Research

John D. Davis†, Charles P. Thacker†, Chen Chang*
*Microsoft Research, Silicon Valley Campus†, BEECube**
*[joda,cthacker]@microsoft.com†, chen@beecube.com**

MSR Technical Report MSR-TR-2009-45

Abstract

In recent years, advances in computer architecture have slowed dramatically with most simulation results demonstrating only incremental architectural innovation. This is further exacerbated by increased processor and system complexity spurred by a seemingly unlimited number of transistors at computer architect's disposal. Computer architects produce a myopic view of their systems through the lens of slow, highly-detailed software simulation or fast, coarse-grained software simulation, with fidelity always in question.

By leveraging silicon technology scaling in Field Programmable Gate Arrays (FPGAs), hardware can be used to accelerate simulation, emulation, or prototyping of systems. Furthermore, because the base components are reconfigurable, the same system can be used for a variety of research projects, amortizing the cost, both in dollars and in learning time. In this paper, we present the third generation of the Berkeley Emulation Engine or BEE3 system. We demonstrate a new collaboration methodology between academia and industry and compare the industrial and academic system design process. The BEE3 is a production multi-FPGA system with up to 64 GB of DRAM and several I/O subsystems that can be used to enable faster, larger and higher fidelity computer architecture or other systems research. Using a widely available hardware platform also facilitates a software community that can generate and share software modules, thereby enabling rapid system development for computer architecture research.

1 Introduction

Historically, software simulation has been the vehicle of choice for studying computer architecture because of its flexibility and low cost. Regrettably, users of software simulators must choose between high performance or high fidelity emulation. In contrast, building hardware in Application Specific Integrated Circuits (ASICs) provides high performance and accurate results, but lacks the flexibility to explore multiple designs. It is also very expensive. These tradeoffs have impeded our ability to thoroughly explore and evaluate new computer architectures. This lack of simulation fidelity and speed is further aggravated by the increase in multithreaded and/or multicore microprocessor architectures.

Traditionally, computer architects have leveraged increasing transistor density to implement a single large processor that exploits instruction level parallelism (ILP). However, continued performance gains from ILP are becoming increasingly difficult to achieve due to limited parallelism among instructions in typical applications [1]. Likewise, the problems associated with designing ever-larger and more complex monolithic processor cores are becoming increasingly significant. These problems include higher bug rates, longer design and verification times caused by the design complexity, and the need to design for increasing wire delay [2]. This fact has spurred great interest in exploiting thread-level parallelism (TLP) among independent threads to continue historical microprocessor performance trends. These multithreaded architectures effectively integrate multiple homogeneous or heterogeneous processors onto a single chip [3][4].

Software-based simulators allow researchers to evaluate small benchmarks or fragments of larger benchmarks using instruction-level simulation, but are too slow to simulate entire applications in a reasonable time. Complicated chip multiprocessor designs worsen this problem by requiring that many

processors be simulated simultaneously [3][5]-[11]. The complexity of even the most basic multithreaded architectures limits instruction-level simulation to an effective "clock rate" of less than 1 MHz; most simulators, especially RTL ones, achieve much less [12]-[19]. Simulation speed therefore limits the scope and effectiveness of research that can be performed in reasonable amounts of time, curtailing significant innovation and fostering future *incremental* research.

To address these problems, we have built the third generation of the Berkeley Emulation Engine or BEE3. This platform is a result of a unique industrial and academic collaboration targeting computer architecture research. We designed it both to support our research and to support the efforts of the RAMP (Research Accelerator for Multiple Processors) community [20]. This consortium of six universities had used the earlier BEE2 system as an emulation platform [21], but this system used FPGAs that were two generations older, and had proved to be expensive and difficult to build reliably.

BEE3 uses state-of-the-art Xilinx Virtex 5 Field Programmable Gate Arrays (FPGAs) combined with copious amounts of DRAM on a single printed circuit board (PCB) to create a flexible computation fabric that can execute billions of operations per second. The resulting platform executes code at speeds at least two orders of magnitude faster than execution-driven software simulation and an order of magnitude faster than previous hardware emulation using generic arrays of FPGAs [17]-[19]. Using BEE3, researchers can rapidly prototype a variety of architectures in a relatively short amount of time by using a repository of low-level component designs [24]-[26]. BEE3 allows detailed investigation of many topics related to processor and system design, including memory hierarchies, TLP extraction methods, TLP oriented software design for operating systems and high-level applications, reconfigurable architectures, embedded systems, and ISA extensions. This platform can also be used as an application accelerator for applications that map well to the FPGA architecture.

This paper presents the initial implementation of our configurable hardware system. Section 2 presents an overview of the system. Section 3 provides an overview of RAMP projects that could be mapped to the BEE3, other research domains that may benefit from the BEE3, and a discussion of BEE3 usage in computer architecture research. Related work is presented in Section 4. The PCB design comparison to the Stanford FAST PCB is presented in Section 5. Section 6 provides a short discussion of the collaboration method used in the project. Our conclusions and future work are presented in Section 7.

2 BEE3 Overview

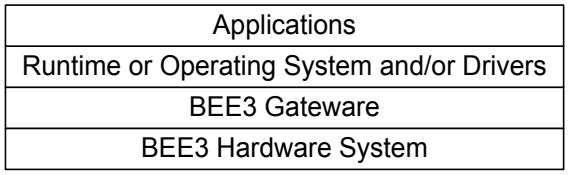

Figure 1. BEE3 system stack.

The BEE3 system is composed of a hardware platform, associated gateware (Verilog or VHDL code to program the FPGA), and the software stack as shown in Figure 1. The BEE3 printed circuit board (PCB) can accommodate three different variants of the Xilinx FPGA Virtex 5 family: General logic (LXT), signal processing (SXT), and embedded systems (FXT) [27]. The BEE3 system is packaged in a 2U rack-mountable enclosure that has several I/O subsystems that can be used to connect several BEE3 systems together or couple the BEE3 to servers or other devices.

2.1 BEE3 Hardware Architecture

The BEE3 system is composed of two separate PCBs. If the reader is familiar with the BEE2 system [21], the control FPGA has been removed, and a smaller PCB that provides FPGA programming, persistent storage and console I/O functionality has been added. The main BEE3 PCB has four Virtex 5 FPGAs, up to 64 GB of DRAM, and several I/O ports.

By moving the control functionality off the main PCB, the overall design complexity is reduced and design flexibility is increased. This reduces PCB routing congestion and allows for little or no control logic or a more complex control and I/O interface tailored for a particular application. The control and I/O PCB fits in a modular sub-chassis. Figure 2 provides the block diagram illustrating the I/O PCB

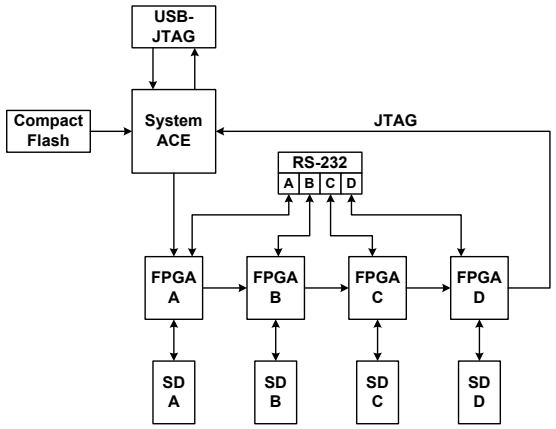

Figure 2. BEE3 control PCB and JTAG programming chain.

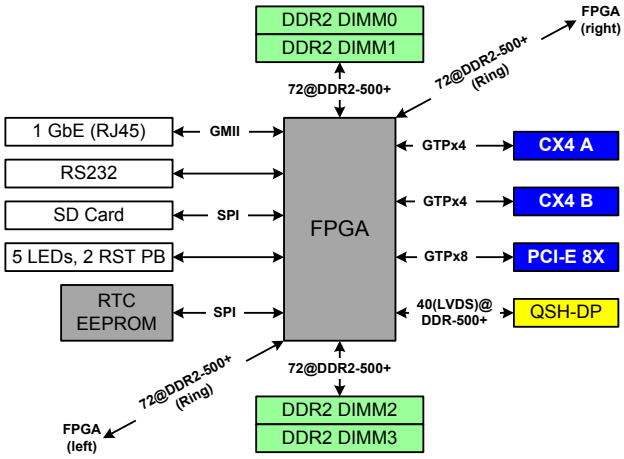

Figure 3. BEE3 single FPGA connectivity.

functionality for FPGA programming, low bandwidth I/O, and bit file (Compact Flash) and data (SD Flash) storage.

The control PCB uses a Xilinx System ACE chip to manage a Compact Flash card and program the FPGAs. Multiple bit file versions can be stored in the Compact Flash card and the System ACE can select which version to use to program the FPGAs. The Xilinx JTAG pod is contained in the sub-chassis as well. As a result, only a USB cable coupled with the programming software on a PC host is required. There are four SD Flash slots (one per FPGA) that provide persistent storage for each FPGA using an SPI interface. Likewise, one RS-232 physical connection per FPGA is available for a console interface. The main BEE3 PCB has two 50-pin headers that provide these interfaces: One header for the JTAG and system control and the other header for the SD SPI and RS-232 interfaces. Thus, other control and I/O PCBs can be created and customized for a particular application domain.

The main BEE3 PCB is footprint compatible with the following Virtex 5 parts: XC5VLX110T, XC5VLX155T, XC5VSX95T, XC5VFX70T, and XC5VFX100T [27]. This provides logic-, signal processing-, and embedded system-focused FPGAs, enabling application targeting of the system. The Virtex-5 FPGAs are a considerable improvement over the earlier Virtex-2 Pro devices used in BEE2 and the previous generation Virtex 4 devices. They provide six-input lookup tables, which improves logic density over the earlier four-input LUTs. They also have larger Block RAMs, and better I/O pin design, which improves signal integrity. The LUTs may also be employed as 64x1 memories. The LX155T parts used in BEE3 contain 97,280 LUT-flipflop pairs, 212 36K-bit Block RAMs, and 128 DSP slices. A DDR2 controller for two channels of DDR2 DIMM memory occupies about 3 percent of the resources of this chip [26].

A BEE3 system could be built as a heterogeneous or homogeneous FPGA platform, leveraging FPGA-specific features at the granularity required by the system. Figure 3 is a block diagram showing the connectivity of an individual FPGA in the BEE3 system.

Each FPGA has two DDR2 DRAM channels with two DIMMs per channel. The DIMM can be single or dual rank and as large as 4 GB, resulting in up to 16 GB supported by each FPGA and up to 64 GB for the system. The FPGA speed grade determines the maximum operating frequency of the DRAM interface. Currently, an unoptimized DDR2 interface using a speed grade -2 FPGA is running at DDR2-500. We use a similar DDR interface for communicating between FPGAs (to the right or left) using the ring wiring with a self synchronizer. The initial implementation is unidirectional, but can be easily modified to be bidirectional. There are also 40 differential pins wired to a Samtec QSH connector. This connector can be used to add interfaces to other devices using daughter cards or provide the fully connected FPGA interconnect to augment the ring wiring. The FPGA multi-gigabit transceivers provide two CX4 interfaces and one PCI-Express (PCI-E) interface. The CX4 interfaces can be used as two 10 Gb Ethernet ports with a XAUI interface and the eight-lane PCI-Express interface supplies endpoint

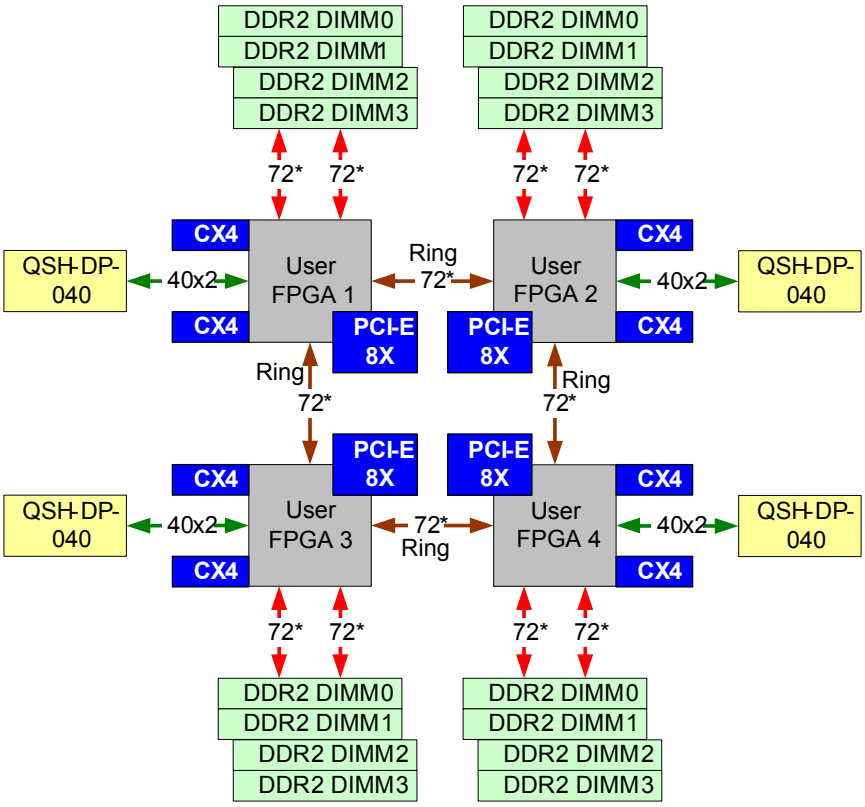

Figure 4. BEE3 main PCB subsystems.

functionality. Each FPGA also has an embedded 1 Gb Ethernet (GbE) MAC hard macro that is coupled to a Broadcom PHY chip. The other major components on the BEE3 PCB are the real-time clock (RTC) and EEPROM that can be used by an OS and to store MAC addresses and other FPGA-specific information. In addition, there are several user-controlled LEDs and a global and an FPGA-specific reset button located on the BEE3 PCB and on the enclosure front panel. Figure 4 provides the overall high-level BEE3 system block diagram and interfaces for the four FPGAs on the main PCB BEE3.

The CX4, PCI-E, or 1 GbE interfaces provide a variety of ways to interconnect multiple BEE3 systems or complement servers with specialized hardware for application acceleration. Using conservative performance estimates, Table 1 provides minimum bandwidth estimates for the I/O interfaces. The DDR2, ring, and QSH interfaces operate at 250 MHz, DDR. Both the DRAM and ring interfaces have two channels and transfer 8 bytes per clock. In contrast, the QSH is a single channel that can transfer only 4 bytes at a time.

2.2 Software Infrastructure

There are several layers that make up the software infrastructure as depicted in Figure 1. The gateware describes the software required to program the FPGAs to configure them for a particular task. Some of the gateware is provided by Xilinx, while other gateware components are provided by the community [24][25]. The BEE3 system includes a reference design that exercises most of the systems functionality. Furthermore, the power-on test suite tests all the subcomponents to guarantee functionality. The DDR2 interface is also available online [26].

The Xilinx gateware is associated with many of the hard macros and provides simple examples that need to be customized to truly be useful. Xilinx provides a variety of gateware, some of which like the PCI-Express, 10 Gb XAUI interface, and 1 Gb Ethernet MAC macros can be modified for use with the BEE3. The user or a third party is required to customize these initial designs for any application requiring that functionality. Xilinx also provides other useful gateware components such as the 32-bit Microblaze

processor. The latest version of the Microblaze incorporates an MMU, allowing full versions of operating systems to be run [28]. Unfortunately, the Microblaze is distributed as a "black box", without source code, so it cannot be customized.

3 Design Mapping

The BEE3 is a reconfigurable system that can be reprogrammed for use across many applications and research fields. Here we present a few applications from the RAMP community using the previous BEE2 system (in most cases, mapping these systems to BEE3 is trivial). Next, we present some examples of using BEE3 in other domains. Finally, we provide context for using BEE3 in computer architecture research.

Table 1. Maximum aggregate bandwidth for BEE3 I/O interfaces.

Interface	Rate	Total Bandwidth
DDR2 DRAM	500 MT/s	8.0 GB/s
Ring	500 MT/s	8.0 GB/s
QSH	500 MT/s	2.0 GB/s
1 Gb Ethernet	-	125 MB/s
CX4 (2 Channels)*	-	5 GB/s
PCI-E (8X)*	-	4 GB/s

* Assuming full duplex communication.

3.1 The colors of RAMP

The RAMP consortium has developed several working prototypes focusing on multiprocessor systems. "RAMP Blue", developed at UC Berkeley, has built a 1000+ core system using 21 BEE2 systems running MicroBlaze processors at 90 MHz [29]. This large scale system is very promising because it demonstrates that large scale FPGA-based systems can be developed that run at reasonable hardware speeds. "RAMP Red", developed at Stanford University, is an eight core transactional memory system on a single BEE2 board. It provides a software development environment for transactional memory programming. The hardware is not cycle accurate, but does run at 100 MHz and provides software performance results two orders of magnitude faster than the software-only simulation [30]. "RAMP White", developed at UT Austin, splits the software simulator into two components, one running in hardware and the other running in software [31]. Likewise, a group at CMU uses a similar novel simulation approach, demonstrating simulation speed-up of one to two orders of magnitude for a simulation system composed of 16 processors [32].

3.2 Using BEE3 in a different research domain.

Boolean Satisfiability (SAT) solvers are widely used as the underlying reasoning engine for electronic design automation, as well as in many other fields such as artificial intelligence, theorem proving, and program verification. As a result, if significant advances occur in this one area, many other fields benefit from this by enabling larger problem sizes and/or faster computation. In [33], Davis et al. have mapped the SAT problem onto a BEE3 system coupled to a host PC, using the FPGA as a co-processor. The key to this co-processor architecture is the novel approach to transform Boolean logic into a compact programmable structure to take advantage of the FPGA's high bandwidth and low latency on-chip block RAM. By splitting the SAT problem into a hardware-assisted software solution, the authors were able to provide software flexibility and hardware acceleration simultaneously. Other domains, such as DSP-based oil and gas exploration data processing and financial calculations are also amenable to FGPA acceleration [34].

3.3 Using the BEE3 for Computer Architecture Research

Now that the BEE3 is available and in production, we have started to actually use it in our research. Although it is premature to report results, since we are still heavily involved in gateware infrastructure-building, this section describes a few of the projects we have begun. We also try to highlight the strengths and weakness of the BEE3 system.

The primary advantage of an emulation platform like the BEE3 for architecture research is that it is possible to run real workloads at reasonable speed in a well-instrumented environment. The designs produced in an FPGA are not competitive in clock speed with "real" implementations, being 10-20X

slower. However, this is still fast enough to run real applications to completion, rather than microbenchmarks or application kernels. Designing a processor core with a clock rate of 100 MHz is straightforward, and several such cores fit on a single FPGA. This operating frequency is only a few times slower than recent academic ASIC processor designs [35][36]. Likewise, additional logic may be added to a design to gather and log any information needed in an experiment. Additional logic may also be added to "warp" time to produce cycle-accurate emulations when needed. We are starting to experiment with the design of a message passing system that is exposed to the ISA of a conventional processor, as an alternative to the use of distributed shared memory in a many-core setting. In general, BEE3 provides a platform for many and multicore research that focuses on using simple base processors. This also enables studies that move up the abstraction layers and focus on various software layers.

Not all designs display uncompetitive performance. We are using the BEE3 to experiment with different ways of employing flash memory in a system, using a 32 GB flash memory DIMM module that plugs into one or more of the DIMM slots. This prototype is not cost-competitive with a real implementation, but it allows us to explore a number of competing approaches that operate at the speed of a real design. This platform provides the ability to investigate architecture, software, and algorithms as well as answer questions about the Flash devices that the Flash manufacturers do not answer.

Similarly, the BEE3 can be used to simulate and prototype advanced networking designs, similar to, but much more powerful than the NetFPGA systems developed at Stanford [37]. Using the system in this way allows designers to make much more informed choices before embarking on a custom design. This is particularly interesting in the context of large data centers, since these installations are quite different from the internet, and may benefit from a number of optimizations enabled by their known topology and limited (although large) size. The BEE3 provides both a 1 Gb and two 10 Gb interfaces per FPGA for network experimentation.

Emulation using FPGAs is not a panacea. Software simulators offer faster time-to-results for initial design studies where microbenchmarks are sufficient. It is still a major endeavor to implement the necessary Verilog or HDL to generate a large design, so this is appropriate primarily when the higher execution speed justifies it and it can build on preexisting infrastructure. As a result, initial case studies or limit studies are not good candidates for BEE3 and its immature software infrastructure. Likewise, there are still FPGA resource limitations that must be observed in order to garner good system performance. Overall, FPGAs are still limited to about 1-2 MB of on-chip memory. Designs requiring more than what is available on-chip may face some problems. FPGAs also suffer from limited off-chip pin bandwidth. Designs that span multiple FPGAs and/or do not exploit locality are not well suited for high-speed operation on FPGAs. Structures that require many read and/or write ports or which have high associativity require and rapidly exhaust FPGA resources. Examples of these structures are multi-way caches, register files with many ports, CAMs, and TLBs. Also, operations that can be implemented in a single ASIC cycle, such as gang clearing cache tag bits, are not possible in the FPGA and may require many hundreds of cycles. While FPGAs are hardware, they require using a restricted subset of hardware primitives to be efficient and these primitives do not necessary map to normal ASIC hardware primitives and techniques. Operating outside of the FPGA-provided building blocks is painful at best, and normally impossible for large designs.

We are currently developing various gateware infrastructure and providing it to the broader community. The first major gateware component is a DDR2 controller for the two channels in each FPGA [26]. These are independent, non-coherent DDR2 controllers that support two 4 GB dual rank, ECC RDIMMs at DDR2-500. With further optimization and/or -3 speed grade parts, the DRAM could operate faster.

The second major piece of gateware infrastructure we are developing is the Beehive [38]. The Beehive is a multiprocessor system composed of a number of RISC cores, a memory controller [26], Ethernet controller, local I/O subsystem and system interconnect in the form of a token ring. Although the core has a non-standard instruction set architecture, it is supported by a C- and C# compiler for software development.

The RISC core has 32 registers and uses word addressing to address up to 8 GB of DRAM. The processor contains a 4KB direct-mapped, allocate-on-write, blocking data cache and a 4KB direct-

mapped instruction cache. The caches are *not* coherent – coherence must be provided where needed through software. It is possible to provide a larger cache with increased associativity, but this comes at the expense of more Block RAMs, which are the scarcest FPGA resource. The data cache is also treated as a local I/O device, since it is possible to do an I/O operation to *evict* a line from the cache. Each core has a local I/O subsystem that also includes an RS232 controller, a multiply unit (implemented with a DSP) capable of doing a 32 x 32 2's complement multiply in ten cycles, an inter-processor messaging facility for exchanging short messages between cores, and a locking unit to provide atomic operations without involving the DRAM.

The processors are connected to each other, to the memory, and to the Ethernet controller using a token ring interconnect. The interconnect carries a stream of 32-bit data items plus a 4-bit *SlotType* field and a *SrcDest* field indicating the source or destination of an operation. The ring addresses one of the most serious limitations of an FPGA many-core design: Routing congestion. Had we used a crossbar or hierarchical bus, routing a design with more than a few cores would be extremely problematic. With a ring, all inter-core wiring is local and relatively short, so it is possible to instantiate a large number of cores.

The Beehive cores can operate at 125 MHz and about 20 cores can fit in a single LX155T FPGA. This enables approximately 80 cores in a homogeneous system on a BEE3 platform using the PCB ring wiring for communication between the FPGAs. Heterogeneous multiprocessor systems can also be derived from the base Beehive components.

4 Related Work

Historically, hardware emulation platforms using arrays of FPGAs have been used to generate rapid prototyping systems that can simulate entire applications at an RTL level [17]-[19]. Unfortunately, efforts to compile multiprocessor designs to these systems have been limited by poor FPGA logic utilization, limited interconnectivity in the FPGA arrays, and poor word-size data manipulation by bit-width FPGA logic units [41]. Likewise, crossing chip boundaries has been a fundamental limitation because off-chip bandwidth is much less than on-chip bandwidth. However, as silicon technology has improved, the capacity of reconfigurable devices and per-pin bandwidth has dramatically improved. Today, multiple simple processors can be mapped to a single FPGA [29] and overall, FPGA capacity and increased per-pin bandwidth enable much greater prototyping flexibility.

There have been several previous systems that have incorporated reconfigurable hardware to enable limited design exploration around a single design point. These hardware prototypes were one-off systems that validated a particular idea [41]-[44]. Other systems were used to explore a narrow research domain [39][40][23]. With advances in FPGA technology derived from Moore's Law, this research domain continues to be broadened by the greater capacity and speed of these system building blocks.

One of the first notable FPGA-based hardware emulators was the Rapid Prototyping engine for Multiprocessors (RPM). This initial hardware emulator for multiprocessor architectures moved beyond simply observing system behavior like bus traffic [39]. RPM was built to prototype Multiple Instruction Multiple Data (MIMDs) multiprocessor machines. The configurable technology available for this project enabled reconfigurable memory system implementations with a fixed SPARC processor. Each processor module was implemented on a single PCB with up to eight boards connected to a backplane and host machine. RPM focused on the memory system of large multiprocessor systems. Even though RPM combined SPARC processors with FPGAs, RPM preserved the memory system transparency all the way up to the processor. Unfortunately, RPM lacked the software infrastructure to make it easy to use and as a result, it failed to achieve widespread adoption like software simulators [12][15][16].

A decade later, The Flexible Architecture for Simulation and Testing (FAST) was developed to investigate TLP architectures. Like RPM, this system had full memory system transparency up to the processor. FAST used simple processors (MIPS R3000 and R3010) combined with state-of-the-art FPGAs and memory chips on a single PCB to create a flexible compute fabric that can execute millions of instructions per second. The resulting environment executes code at speeds at least two orders of magnitude faster than execution-driven software simulation and an order of magnitude faster than

Table 2. FAST, BEE2, BEE3 PCB comparison.

Platform	Dimensions	Layers	FPGAs	Components	Pins	Vias	Connections	Nets	BRAM	SRAM	DRAM
FAST	16" x 16"	20	10	4300	22000	32000	17000	4200	640 KB	68 MB	N/A
BEE2	13.8" x 17.3"	22	5	4000	25000	18000	18000	5200	3 MB	N/A	20 GB
BEE3	12" x 16.5"	18	4	2500	16000	9900	12000	3300	3.7 MB	N/A	64 GB

previous hardware emulation using generic arrays of FPGAs [23]. FAST was designed explicitly to enable memory system and additional architectural research such as speculative coprocessors for Thread Level Speculation or other off-load engines, in large multiprocessor (MP) or small, fast CMP systems. FAST was able to replicate the memory latency for both MP and CMP systems, as well as anywhere in between by artificially throttling the high-speed SRAM [23]. The FAST system was scalable up to 16 PCBs, enabling up to 64 processor systems, although only one system was ever built. Like RPM, FAST also suffered because of a lack of resources and resulting software infrastructure.

The RAW Architecture Workstation (RAW) [41] enabled in depth research on multicore architectures, but the RAW fabric lacked the capacity and the ease of use for mapping other designs. Even though RAW used a commercially available emulation platform, the difficulty experienced when mapping other designs to this resource-constricted FPGA array prevented the RAW FPGA array from being used for other research projects. Partitioning designs across 64 very small FPGAs also presented a significant challenge and considerable communication delays because of pin pressure.

The RAMP consortium faces the same financial hurdles that face any project trying to distribute a system that is not free, but RAMP is also trying to attack the software infrastructure problem by collaborating and creating an open-source-like community for the hardware infrastructure [25]. The combination of hardware and software continues to be essential for the next generation prototyping platform. The hardware design should be flexible, but the software infrastructure is crucial for wide adoption and rapid system building. Constructing research systems with software simulators will always be easier, but leveraging a gateware repository will reduce the hardware burden.

RAMP repurposed the BEE2 platform to use an architecture research platform. The BEE2 is an evolution of a platform focused on DSP development for wireless radio applications such as cellular phone base stations. The BEE2 has 5 Virtex-II Pro 70 FPGAs. One FPGA is designated as the master or control FPGA and the other four FPGAs are user FPGAs. Each FPGA has 4 DDR2-400 channels. The four FPGAs communicate using a ring topology. The control FPGA has point-to-point links to each user FPGA. Each user FPGA has 4 independent high-speed serial links and connections to common peripherals such as 10/100 Ethernet, USB 1.1, RS232 serial, DVI, and GPIOs [21].

Table 2 provides a comparison of the FAST, BEE2, and BEE3 systems. Even though the BEE3 is more complex than the older systems in the comparison, it is smaller, thinner (less layers), and has fewer components, nets, and most importantly, vias. This leads to a system that is easier and cheaper to manufacture and more reliable as a result. While BEE3 uses the latest generation FPGAs, it is interesting to compare the recent FPGA computer architecture systems. The Berkeley Emulation Engines have evolved from their DSP and wireless radio research genesis. Design elements like the FPGA ring are evidence of its history. The QSH interconnect is an addition to the BEE3 platform tailored for all-to-all communication with computer architecture research in mind. Likewise, few DRAM channels with higher capacity were selected instead of more DRAM bandwidth. This significantly reduced the system cost by utilizing an FPGA with a smaller package. Finally, FAST utilized SRAM to model low-level cache and memory behavior, while BEE3 provides copious amounts of DRAM that enables investigation at a higher system level using simple cores with simple cache hierarchies.

A subset of computer architecture research will continue to incorporate hardware prototypes given the increased capacity of FPGAs, but the ability to leverage both hardware and software infrastructure will make the use of FPGA-based systems like BEE3 much more widespread. This amortizes the hardware

and software development and cost across multiple projects and leverages both hardware and software infrastructure. By developing the hardware and software as a community, computer architecture research can once again validate ideas with working hardware that is easier and cheaper to build. Furthermore, the hardware enables more thorough system and software investigation and software development and tuning, completing the once-broken research cycle.

FPGAs have reached the point where they can implement multiple designs given an intelligent framework. RPM, FAST, and BEE2 have demonstrated the ability to build working, configurable prototyping platforms. The RAMP consortium may provide the most important thing missing from previous reconfigurable hardware systems: Community. While RAMP has demonstrated that multiple designs can be mapped to the same hardware substrate, it is still to be seen if they can foster a community that develops and shares infrastructure [20][29]-[32].

5 Building a system in academia versus industry

There have been several academic PCB designs in addition to microprocessor and other ASICs. The BEE3 provides a unique opportunity to compare and contrast the PCB development process in an academic and industrial environment. We compare the timeline and design process for the BEE3 and the Stanford FAST PCB. FAST's major differences compared to the BEE3 are the memory and I/O systems. FAST integrated two levels of SRAM and had no DRAM interface verses the DRAM-only memory interface of the BEE3. Furthermore, BEE3 also has many more I/O subsystems than FAST. Unlike the BEE3, FAST provided both dedicated integer and floating point pipelines by tightly coupling MIPS CPUs and FPUs with FPGAs. However, high performance floating point can be provided on BEE3 using the DSP hardware macros provided by the FPGA.

The FAST project at Stanford initially had a false start. The first graduate student experienced significant design hurdles for two years, which finally resulted in him leaving the PhD program due to burnout. The FAST PCB design was restarted from scratch and the layout and routing took about 6 months and was completed by a single graduate student. Again, this time was doubled by CAD tool problems, which halted progress until the problems were corrected. Once the FAST PCB design was complete, an additional 2 months were required to adjust the routed design to improve its manufacturing yield, since design for manufacturability (DFM) was an afterthought for this graduate student's first major PCB design. The overall time to produce a single untested system was eighteen months. There was an additional 8 months of system bring up, testing and development that demonstrated FAST's capabilities.

In contrast, no graduate students were lost or harmed in the design and prototype build of the BEE3. By using a professional PCB design house (Celestica Corporation), the resulting BEE3 system (a more complex design) was simpler (fewer PCB layers) and as a result cheaper to build. By using a PCB design house, we were able to develop the PCB, the bring-up gateware, and system gateware like the DRAM controller [26], in parallel. Much of the gateware from BEE2 was ported to BEE3 to quickly produce a bring-up test. Like FAST, the BEE3 required no rework based on the initial specification. Unlike FAST, the initial BEE3 specification had to be revised, which required a single revision of the PCB before it could be released for production. This was the only unscheduled delay in the BEE3 development. From inception to initial prototype, the BEE3 took about 9 months, half the time required to produce FAST. Even when considering the BEE3 revision, the fully functional BEE3 design was completed in 17 months compared to 28 months required to design, build, and test FAST.

Unlike FAST, the BEE3 PCB prototype was a global project. The PCB layout and routing was done in Shanghai. The simulation (static timing analysis and signal integrity analysis) and PCB fabrication and assembly were done in Toronto. The design specification, design reviews, and related logistics were done in California. In aggregate, the BEE3 prototype had 30 individuals contributing to the project and many of the contributors were specialists with extensive knowledge of the CAD suite. In the case of FAST and BEE2, these specialists were replaced by one or a few graduate students. As a result, the academic projects produced more complex and less reliable PCBs (as demonstrated by the PCB layer count), longer project time horizon, and graduate student casualties.

6 A New Model for Industry-Academic Collaboration

The BEE3 is an example of a new type of tightly coupled industrial and academic collaboration. In this case, we have developed a system based on academic specifications and that is available to both academia and industry. This is a dramatic change from providing research funding for a particular project or providing supplement project support, e.g., silicon chip back-end process and/or fabrication support, and has many benefits over the normal government or industrial funding process. First and foremost, the industrial partner can leverage expertise that does not exist in or cannot be accessed from academia. Second, in the case of BEE3, PCB design professionals produced a system in less time and with a lower manufacturing cost than could be done by an academic institution using graduate students to design the system. Graduate student designers must spend a great deal of time learning the complex CAD tools, an activity with limited pedagogical value. Third, the BEE3 is available to both academics and industry alike, in contrast to the FAST or BEE2 systems. The BEE3 project is unique in that it provides a research platform for a wide variety of research domains that exist both in academia and industry. It is an example of industry providing research support that is outside the normal support channels, but is a more efficient use of resources with potentially much higher impact than an academic-only system. Finally, the BEE3 design was licensed, *royalty free*, to BEEcube Corporation [45], a company that produces, sells, services, and enhances the BEE3 system.

This collaboration model also has it challenges. Not all projects are amenable to industry development. There may be issues related to intellectual property and licensing. Feature creep and specification deviation may also be a problem for projects that have long time horizons resulting in something that neither industry nor academia want. In this case, we were able to work closely and effectively with academia and other industry partners and this helped to guarantee a very favorable outcome. The BEE3 also benefits from a third party distribution mechanism that may be difficult to reproduce. Finally, it may be difficult to define a project well enough from its onset that industry can provide an alternative mechanism.

7 Conclusions & Future Work

The initial bring up and testing of the BEE3 system has been very successful. We have tested all of the subsystems at or above target operating frequency and have found no problems. The complete production BEE3 system in its 2U chassis is shown in Figure 5.

The delivery of the first production run of licensed BEE3 system took place in August 2008. The BEE3 system was completed faster and better than previous academic designed multi-FPGA systems. The result is a system with better signal integrity and cheaper PCB manufacturing costs. Like all previous multi-FPGA PCBs, the BEE3 system is nowhere near the initial $5,000 target PCB price, even if all the PCB parts were free. It is also the case that the BEE3 will not replace software simulation in computer architecture, but augment the research cycle with hardware either as a simulation accelerator, software development platform, or a prototyping platform. This is a viable method to reintroduce hardware back into the research cycle for a broader class of architectures and systems research. Finally, we have presented a short survey of RAMP projects that demonstrate the fidelity spectrum and variety of usage models for systems like BEE3. Depending on the target, BEE3 can provide a wide range of fidelity without the performance impact that software simulation exhibits. BEE3 will not supplant software simulators, but, with some software infrastructure building, BEE3 does provide a solution as a computer architecture research platform for multicores and many cores. We have demonstrated this by building the Beehive many-core system.

The BEE3 systems have several future directions inside and outside our research group. Focusing on computer architecture, we plan on developing infrastructure to investigate transactional memory running on real hardware, multiprocessor and memory research, as well as, operating system research enabled by special purpose hardware. Using software defined processors, we are able to move up the stack into software and implement interesting systems that were infeasible in a software simulator. As one can

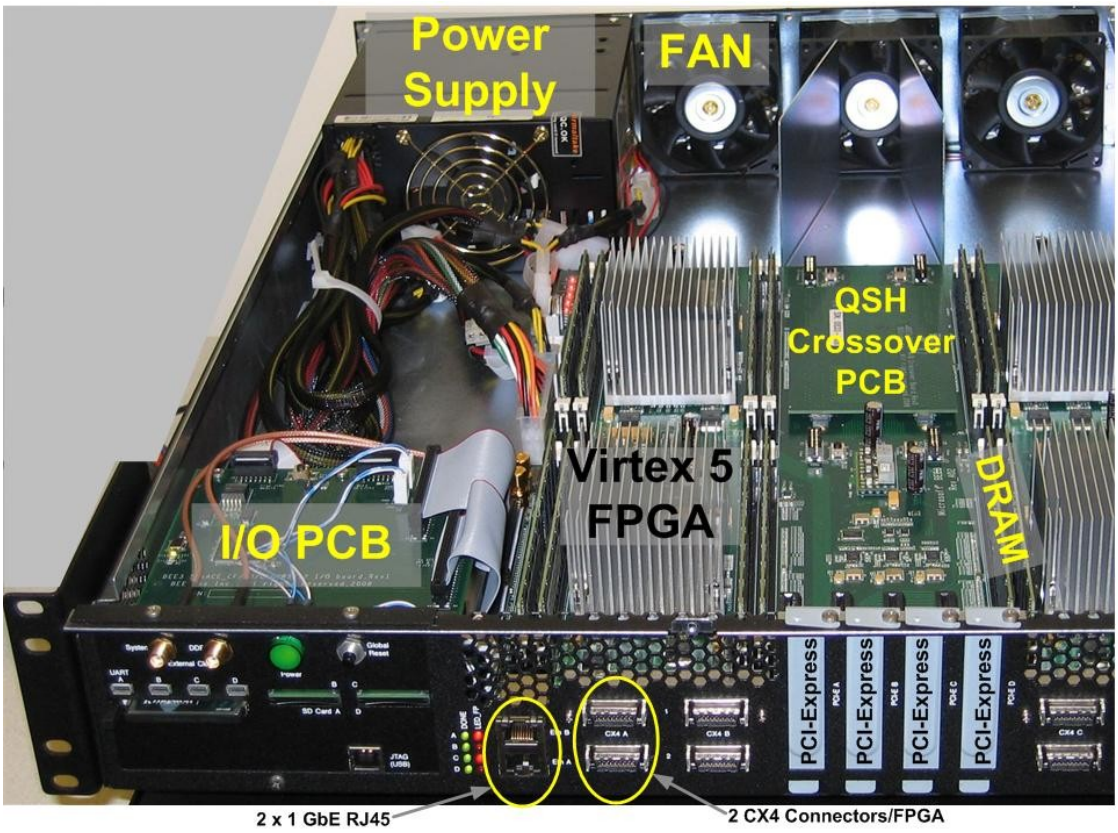

Figure 5. Production BEE3 system.

imagine, there are several systems research areas to pursue as well. In those areas, we plan on mapping the initial application accelerators and prototype research platforms to the BEE3 for evaluation.

Outside of our research group, the main initial target for the BEE3 is the RAMP consortium. This collection of universities has many ongoing projects and our hope is that a vibrant community continues to develop and share infrastructure to enable research and collaboration across university boundaries. These research areas will be accelerated by a community sharing and supporting a common platform.

8 Acknowledgements

There were many people involved in this project that made it successful. On the academic side, the RAMP consortium and in particular, the people at BWRC were very helpful. Celestica made the hardware development easy with a distributed team in China and Canada. In total, over 30 people from Microsoft, Celestica, and BEEcube were intimately involved in the design, manufacturing, and licensing of the BEE3 system.

9 References

[1] D. W. Wall, "Limits of Instruction-Level Parallelism," WRL Research Report 93/6, Digital Western Research Laboratory, Palo Alto, CA, 1993

[2] D. Matzke, "Will Physical Scalibility Sabotage Performance Gain?," IEEE Computer Magazine, September 1997, page(s) 84-88

[3] J. D. Davis, J. Laudon, K. Olukotun, "Maximizing CMT Througput with Mediocre Cores," *In Proceedings of the* 14th International Conference on Parallel Architectures and Compilation Techniques (PACT), Sept. 2005

[4] R. Kumar, D. Tullsen, et al., "Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance," The 31st International Symposium on Computer Architecture (ISCA-31), June 2004.

[5] L. Hammond, B. Hubbert, et al., "The Stanford Hydra CMP," *IEEE MICRO Magazine*, March-April 2000.

[6] J. Huh, et al., "Exploring the Design Space of Future CMPs," *PACT*, pp. 199-210, Sept. 2001.
[7] V. Krishnan and J. Torrellas, "A Chip Multiprocessor Architecture with Speculative Multithreading," *IEEE* Transactions on Computers, Special Issue on Multithreaded Architecture, September 1999
[8] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. Dally, and M. Horowitz, "Smart Memories: A Modular Reconfigurable Architecture," *ISCA-27*, June 2000.
[9] K. Sankaralingam, R. Nagarajan, et al., "Exploiting ILP, TLP, and DLP Using Polymorphism in the TRIPS Architecture," *ISCA-30*, pp. 422-433, June 2003
[10] G. Sohi, S. Breach, and T. Vijaykumar, "Multiscalar processors," *ISCA-22*, pp. 414–425, June 1995.
[11] J. Steffan and T. Mowry, "The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization**,"** Proceedings of the Fourth International Symposium on High-Performance Computer Architecture *(HPCA-4)*, February 2-4, 1998.
[12] T. Austin, E. Larson, and D. Ernst "SimpleScalar: An Infrastructure for Computer System Modeling," *IEEE Computer Magazine*, Feb. 2002, Page(s): 59 -67
[13] J. Emer, et. al., "Asim: A Performance Model Framework," *IEEE Computer Magazine*, Feb. 2002, pp: 68-76
[14] C. J. Hughes, et al., "Rsim Simulating Shared-Memory Multiprocessors with ILP Processors," *IEEE Computer Magazine*, Feb. 2002, pp. 40-49
[15] P. Magnusson, M. Christensson, J. Eskilson, et al., "Simics: A Full System Simulation Platform," *IEEE Computer Magazine*, February 2002, pages 50-58
[16] M. Rosenblum, E. Bugnion, et al., "Using the SimOS Machine Simulator to Study Complex Computer Systems,"ACM Transactions on Modeling and Computer *Simulations*, vol. 7, no. 1, pages 78-103, Jan. 1997
[17] Mentor Emulation Products, http://www.mentor.com/emulation
[18] Dini Group Product Brochure, http://www.dinigroup.com/files/dini_brochure.pdf
[19] HiTech Global FPGA Prototyping and evaluation boards, http://www.hitechglobal.com/Boards/allboards.htm
[20] Arvind, et al.; "RAMP: Research Accelerator for Multiple Processors - A Community Vision for a Shared Experimental Parallel HW/SW Platform." 2005; Available from: http://ramp.eecs.berkeley.edu/index.php?publications.
[21] C. Chang, J. Wawrzynek, and R. W. Brodersen. "BEE2: A Hig-End Reconifgurable Computing System." IEEE Des. Test, 2005. 22(2): p. 114-125.
[22] J. D. Davis, C. Fu, and J. Laudon, "*The RASE (Rapid, Accurate Simulation Environment) for chip multiprocessors."* SIGARCH Computer Architecture News, 2005. 33(4):p. 14-23.
[23] J. D. Davis, "FAST: A Flexible Architecture for Simulation and Testing of Multiprocessor and CMP Systems," Ph.D. Thesis in Electrical Engineering, 2006, Stanford University: Stanford, CA.
[24] Open source for hardware modules, http://www.opencores.org/
[25] Berkeley FPGA Repository, http://repository.eecs.berkeley.edu/
[26] C. Thacker, "DDR2 Controller for the BEE3," http://research.microsoft.com/en-us/downloads/12e67e9a-f130-4fd3-9bbd-f9e448cd6775
[27] Xilinx Virtex5 Datasheets, http://www.xilinx.com/support/documentation/virtex-5_data_sheets.htm
[28] MicroBlaze Processor Reference Guide; Available from: http://www.xilinx.com/support/documentation/sw_manuals/edk92i_mb_ref_guide.pdf
[29] A. Krasnov, A. Schultz, J.Wawrzynek, G. Gibeling, and P. Droz, "RAMP Blue: A Message-Passing Manycore System In FPGAs," *Proceedings of International Conference on Field Programmable Logic and Applications*, Amsterdam, The Netherlands, August 2007.
[30] N. Njoroge, et al., "Building and Using the ATLAS Transactional Memory System," in Proceedings of the Workshop on Architecture Research using FPGA Platforms, held at HPCA-*12*. 2006.
[31] H. Angepat and D. Sunwoo and D. Chiou. "RAMP-White: An FPGA-Based Coherent Shared Memory Parallel Computer Emulator." 8th Annual Austin CAS Conference, March 2007.
[32] E. S. Chung, E. Nurvitadhi, J. C. Hoe, B. Falsafi, *and* K. Mai, "A complexity-effective architecture for accelerating full-system multiprocessor simulations using FPGAs," *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, Monterey, California, 2008
[33] J. D. Davis, Z. Tan, F. Yu, and L. Zhang, "Designing an Efficient Hardware Implication Accelerator for SAT Solving," SAT 2008 , H. K. Büning and X. Zhao (eds.),LNCS,vol. 4996, pp.48-62
[34] R. Dimond,M. J. Flynn, O. Mencer, and O. Pell, "MAXware: Acceleration in HPC," in Proceedings of HotChips 20, August, 2008
[35] J. H. Ahn, W. J. Dally, et al., "Evaluating the Imagine Stream Architecture," Proceedings of the 31st Annual International Symposium on Computer Architecture, Munich, Germany, June 2004

[36] K. Sankaralingam, R. Nagarajan, et al., “The Distributed Microarchitecture of the TRIPS Prototype Processor," *39th International Symposium on Microarchitecture* (MICRO), December, 2006
[37] J. W. Lockwood, N. McKeown, et. Al., “NetFPGA - An Open Platform for Gigabit-rate Network Switching and Routing,” *MSE 2007, San Diego, June 2007.*
[38] C. Thacker, “Beehive,”
[39] L. A. Barroso, S. Iman, et al., “RPM: a rapid prototyping engine for multiprocessor systems,” IEEE *Computer Magazine*, Feb. 1995, Page(s): 26 -34
[40] M. Dubois, J. Jeong Y. H. Song, A. Moga, “Rapid hardware prototyping on RPM-2,” *IEEE Design & Test of Computers*, July-Sept. 1998, Page(s): 112-118
[41] E. Waingold, et. al., “Baring It All to Software: Raw Machines.” *IEEE Compter*, 30(8), pages80-93, September 1997, MIT/LCS Technical Report TR-709, March 1997.
[42] Agarwal, A., et al., "The MIT Alewife machine: architecture and performance," in Proceedings of the ISCA-22. 1995, ACM Press:
[43] Gibson, J., et al., "FLASH vs. (Simulated) FLASH: closing the simulation loop," in Proceedings of the ninth international conference on Architectural support for programming *languages and operating systems*. 2000, ACM Press: Cambridge, Massachusetts.
[44] Lenoski, D., et al., "The DASH prototype: implementation and performance," in Proceedings of the ISCA-19. 1992, ACM Press: Queensland, Australia.
[45] BEEcube Corporation, http://www.beecube.com/