BEE2: A High-End Reconfigurable Computing System

Chen Chang, John Wawrzynek, and Robert W. Brodersen
University of California, Berkeley

The BEE2 project is developing a reusable, modular, and scalable framework for designing high-end reconfigurable computers, including a processing-module building block and several programming models. Using these elements, BEE2 can provide over 10 times more computing throughput than a DSP-based system with similar power consumption and cost, and over 100 times that of a microprocessor-based system.

■ A HIGH-END RECONFIGURABLE COMPUTER (HERC) is a machine with supercomputer-level performance configured on a per-problem basis to match the structure of the algorithm and data flow of a computing task. In such a machine, all data and control paths; memory ports and controllers, and communication channels and controllers, are customizable to match a particular application's needs. In the Berkeley Emulation Engine 2 (BEE2) project, we seek to design, construct, and program a HERC for a set of application areas. At the Berkeley Wireless Research Center (BWRC), we have assembled the first two-module BEE2 system prototype, using entirely commercial off-the-shelf components. We will expand the prototype system to include 12 modules and serve as the computing platform for various research projects at BWRC.

Several computationally intensive problems are central to BWRC's research objectives. These problems act as an application benchmark set and design drivers for specifying the machine architecture and its associated software mapping tools. The applications fall into four broad categories:

- emulation and design of novel wireless communications systems,
- high-performance real-time digital signal processing,
- real-time scientific computation and simulation, and
- the acceleration of CAD tools.

The analysis in this article concentrates on high-performance DSP applications. Some DSP applications would typically not fit on a single processor; cell phone base station processing (as opposed to processing within individual cells phones) is one such application. Computational efficiency, measured in throughput per unit cost or power consumption, is crucial to the success of these systems.

When evaluating the performance of general-purpose software on microprocessors, it is possible to use platform-independent benchmarks written in high-level languages such as C or C++. DSP applications, however, often employ other types of implementations, such as DSPs, FPGAs, and ASICs. For these implementations, C or C++ compilation is immature and often produces final hardware implementations far less optimal than using alternative, low-level design methods. This performance degradation is mainly due to the mismatch between the inherently sequential execution model of the C description and the spatially parallel execution nature of the underlying hardware implementation technology.

Therefore, to maximize computational efficiency, engineers typically use low-level design input methods particular to the underlying hardware. For microprocessors and DSPs, current best-practice programming methodology uses assembly languages. For FPGAs and ASICs, hardware description languages (such as VHDL and Verilog) are the common choice. So far, no single DSP benchmark set is available on all four hardware implementation technologies.

Even within a single hardware technology, such as DSPs, the proprietary architectures of individual DSP chips have largely forced programmers to focus on the performance of the specific application at hand, rather than using established benchmarks. The situation is even worse in the case of FPGAs or ASICs, where all aspects of the processing architecture can be user defined. We have focused our study on FPGAs versus DSPs for a set of high-performance digital-signal-processing applications.

Target application domains

Ongoing research projects on advanced wireless communication systems at BWRC have used advanced design techniques. These techniques include sophisticated encoding-decoding techniques (such as turbo and low-density parity check codes), software-defined-radio design, cognitive spectral reuse, multimode operation (using the Global System for Mobile Communications, and the code division multiple access and time division multiple access protocols), and multiple antenna MIMO (multiple-input, multiple-output) systems, all of which require ever-increasing digital-processing capability and flexibility at minimal power consumption. The validation of these complex systems requires in-system emulation of the physical-layer processing over hundreds or thousands of packets or frames. This emulation would take weeks on conventional microprocessor-based computers, far exceeding the real-time requirement of the analog subsystems.

To address these issues, we designed the BEE2 system's predecessor, the Berkeley Emulation Engine (BEE),[1] in 2001. This system has been in active service for two years, serving as a real-time emulator of novel wireless communication ASICs. BEE2 will not only surpass this system's emulation speed, but also provide a scalable solution for ever-increasing emulation capacity requirements. BEE2 will handle a single ASIC or SoC, as well as a complete ad hoc sensor network with thousands of individual nodes.

The targeted high-performance real-time DSP applications use the BEE2 system as the final implementation, unlike the emulation applications, where the BEE2 system is part of the design automation tools. A key area in this domain is radio astronomy beamforming, spatial correlation, and wide-band fine-resolution spectroscopy for large radio telescope antenna arrays, such as the Allen telescope array (http://seti.berkeley.edu). These applications have low numeric precision requirements (typically 4- to 32-bit fixed-point) but must meet the processing requirements of several gigahertz of continuous RF bandwidth over hundreds of physical antennas. So they can easily require speeds of a teraoperations per second (TOPS) to more than a petaoperation per second (POPS).

The hard real-time throughput and the high-performance requirements make it next to impossible to implement these applications on conventional microprocessors with a nondeterministic execution model. Rather, the dataflow processing nature of these algorithms matches the stream-based computation model commonly used on reconfigurable devices, with throughput directly locked to the system clock rate. So the radio astronomy community has been using FPGAs or ASICs for these applications. The diverse scientific requirements from one telescope to another have, to date, meant that the scientific research community has engineered and developed most existing radio telescopes. So almost all existing implementations in this area have been custom designs with little reusability from one implementation to another.

The BEE2 system is one of the first attempts at providing a scalable, modular, economic solution for a range of high-performance radio telescope DSP applications. Furthermore, BEE2 provides a unique multiuser environment, much like that in a conventional PC cluster, where many users can share a common pool of computing resources with guaranteed computational throughput. Other applications in this domain include advanced video compression and transcoding, and real-time hyperspectral imaging.

Real-time scientific computing involves a set of computing tasks traditionally solved using supercomputers or clusters. One example of interest to BWRC is 3D electromagnetic field simulation for antenna design. This problem, like many other large-scale simulations of physical systems, is characterizable by large systems of partial differential equations, often involving large regular or adaptive grid structures. The conventional methods require fast-Fourier transform (FFT) and large-matrices operations; they typically employ double-precision floating-point computations. Traditionally, such problems are not solved in real time; however, real-time processing of this class of problem would have significant new application. Furthermore, many opportunities exist for innovation on the algorithms and mapping of these problems to reconfigurable machines. Recently, various academic research projects have shown that FPGAs can provide up to three

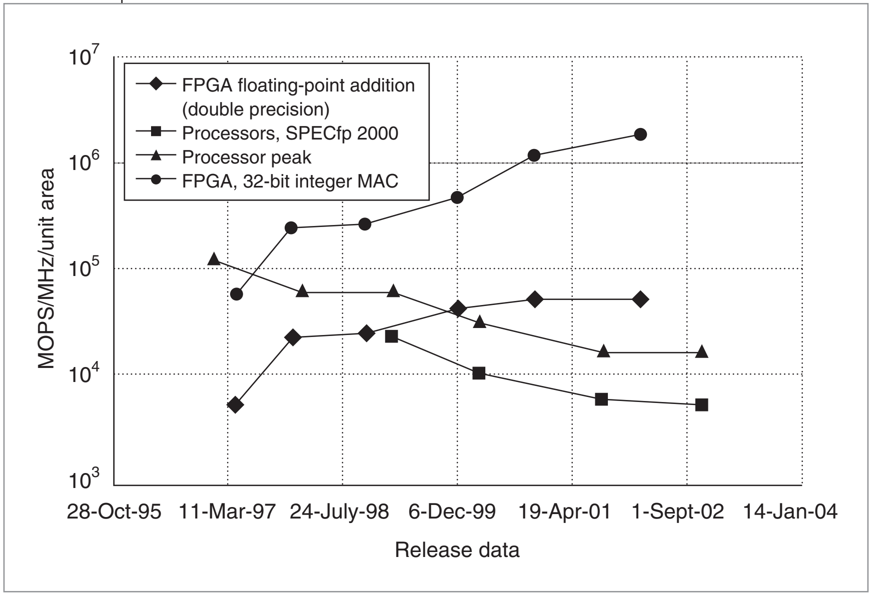


Figure 1. Computational density of FPGAs and Intel processors.

orders of magnitude higher performance than conventional microprocessor cluster solutions for selected 2D and 3D finite-difference time domain (FDTD) problems.[2,3]

We are also studying how to use BEE2 to develop techniques that speed up the tool flows for ASIC and FPGA design. Of primary concern is acceleration of Spice circuit simulation. Also, existing low-level FPGA tool flows (for placement and routing) are currently too slow to be practical for HERC systems. Current placement-and-routing cycle times of hours are unacceptable for reconfigurable systems with hundreds of FPGAs, potentially each with a unique configuration. We believe that BEE2-like HERC machines can eventually help accelerate their own tools. Some early work on hardware-assisted fast routing[4] and simulated annealing for FPGA placement[5,6] shows promise along these lines and will guide our work.

Traditional supercomputer approaches

Modern supercomputers are typically a collection of commodity microprocessors. These systems typically demonstrate peak system performance (on artificial benchmarks) of about 100 Gflops to 10 Tflops. The key idea behind the success of this class of machines is the adoption of commodity components, namely off-the-shelf microprocessors and memory modules. In most cases, the low-volume production of supercomputers makes it difficult to justify the cost and time for custom processor development. Also, using commodity components enables rapid technology refresh and performance scaling as new components come to market.

Figure 1 illustrates a trend in the computational densities of microprocessors and FPGAs. The graph shows the evolution of peak processor computation density over time through the previous six generations of Intel desktop processors. This data is specific to Intel microprocessors but exemplifies all superscalar microprocessor architectures. It clearly shows the inability of microprocessors to efficiently turn increasing die area and speed into useful computation.

Over the same period, the peak computational density of FPGAs has surpassed Moore's law. Because of their simple hardware structure, each successive generation of FPGA scales naturally with process technology. The use of more metal layers for on-chip interconnects and the inclusion of dedicated functional blocks have also aided FPGAs in scaling peak performance.

Of course, comparing processor and FPGA computational density is not entirely fair, because conventional processors come with a programming model and mature compilation tools, whereas FPGAs often require the laborious hand mapping of applications. However, the goal of our research is to make computing with reconfigurable devices similar in convenience to that of a microprocessor. Our experience with the BEE has demonstrated that, at least within the DSP domain, high user productivity and efficient application mapping is possible.

Another troubling characteristic of high-end microprocessor-based systems is the widening gap between memory and processor speeds. This gap has led to several layers of caches in these systems. The unpredictable latency through this cached-memory hierarchy makes it difficult to meet hard real-time requirements and program large multiprocessor systems.

A high-end computer based on FPGAs allows a high degree of spatial parallelism and circuit specialization within nodes, resulting in increased performance density even at significantly lower clock rates than that available in microprocessors. The lower clock rate results in reduced power and a better match to the speed of synchronous DRAM memory systems. Furthermore, FPGA designs can provide multiple independently addressed memory banks per processing node, resulting in significantly higher memory bandwidth per node than conventional microprocessor systems. This also provides deterministic memory latency as a result of removing caches. By interfacing FPGAs directly to the communication network, the low-level configurability of FPGAs permits a high degree of flexibility in the network—allowing, for instance, circuit-switched routing for some applications and packet-switched dynamically routed messages in others. Predictable memory and network latency enables the static scheduling of memory accesses and data transfers in real-time applications.

Commercial reconfigurable computer developments

Recently, the computational capabilities of commercial FPGAs from Altera and Xilinx have increased tremendously in terms of silicon gate count and circuit speed. By directly embedding dedicated arithmetic units (multipliers and accumulators) and general-purpose processor cores (such as the PowerPC 405 in the Xilinx Virtex-2 Pro) into the reconfigurable fabric, high-end FPGAs have evolved into a SoC solution for applications that require high throughput integer or floating-point computations. For example, the Xilinx Virtex-4 FPGAs,[7] built with 90-nm CMOS technology, have up to 512 multiply-accumulate (MAC) units, each with a dedicated 18-bit multiplier followed by a 48-bit accumulator, operating at greater than 500 MHz. Such an FPGA can achieve a peak performance of 256 billion MAC operations per second or 512 GOPS, using only the dedicated cores. If an implementation also uses a reconfigurable fabric, peak performance can reach as high as 1 TOPS (16-bit operations)—a 100 to 1,000 times higher throughput than that of any existing commercial microprocessor or DSP.

Currently, several commercial and academic projects are developing hardware and software systems to employ the raw computational power of these new FPGAs. Most of the available commercial HERCs are augmented computer clusters or supercomputers with FPGAs attached to the system bus or I/O interfaces as computation kernel accelerators. The Annapolis Wildstar FPGA systems (http://www.annapmicro.com) connect processing boards with one to four FPGAs to the Peripheral Component Interconnect (PCI) bus of a conventional computer node. Similarly, Starbridge Systems' Hypercomputer (http://www.starbridgesystems.com) packs up to 11 FPGAs on a large PCI-X (extended PCI) board. One major performance bottleneck in these systems is the communication bandwidth between the microprocessor and the FPGA accelerator board; at peak throughput, this is 132 Mbps on a PCI 33-MHz bus or 1,064 Mbps on a PCI-X 133-MHz bus. To remove this bottleneck, SRC Computers attaches their FPGA accelerator module, called the Multi-Adaptive Processor,[8] directly to the memory interface, which provides a 2,800-Mbps throughput to their proprietary Hi-Bar switch. This arrangement connects all processors and main memories in a single compute node. However, this interface's proprietary nature limits its adaptation to only selected high-end supercomputers.

The latest solution to this interface problem originally came from Octigabay, now acquired by Cray for its XD1 system (http://www.cray.com/products/xd1/index.html). This solution uses the HyperTransport interface technology[9] from AMD to connect up to six FPGAs to the AMD Optron microprocessor's north bridge. Because HyperTransport is part of all AMD Optron and Athlon 64 computers, it provides a more economical solution with a data throughput of 3,200 Mbps.

Nevertheless, none of these solutions address the issue of nondeterministic access latency between multigigahertz microprocessors and multimegahertz FPGAs. With the overhead of the memory cache, bus negotiation, and synchronization, the perceived access latency can range from hundreds to thousands of processor cycles. This severely reduces the performance of applications that require a tight feedback loop between the microprocessor and the FPGAs, reaching the point where the FPGA accelerator provides little speedup. Hence, most applications running on these systems do not have a strict real-time requirement, and the portion of the algorithms accelerated on the FPGAs tends to be compute intensive with relatively few data transfers between the microprocessor and FPGAs.

BEE2 system

The BEE2 system uses Xilinx Virtex-2 Pro FPGAs[10] as the primary and only processing elements. This FPGA design directly embeds the PowerPC 405 processor cores into the reconfigurable fabric, minimizing the latency between the microprocessor and the reconfigurable fabric while maximizing the data throughput. Furthermore, with FPGAs running at clock rates similar to that of the processor cores, system memory, and communication subsystems, BEE2 does not need hardware-managed caches, hence all data transfers within the system have tightly bounded latency. BEE2 is therefore well suited for real-time applications, especially those that require high

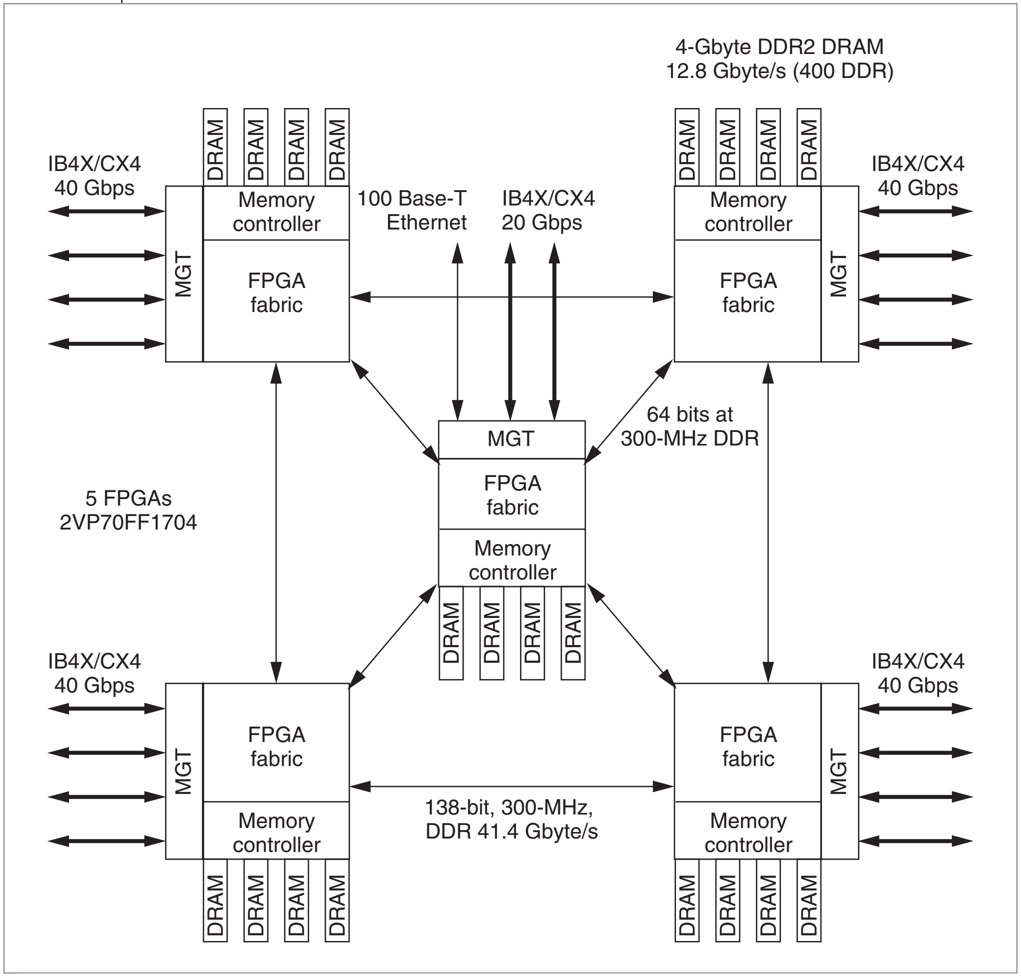


Figure 2. Compute module block diagram.

to generate the native execution binaries for each side—machine code binaries for microprocessors, and bit files for FPGAs.

Hardware architecture

The major components of the hardware architecture are the compute module and the global communication networks. We also briefly describe the mechanical design of a BEE2 module.

Compute module. Each compute module in BEE2 consists of five Xilinx Virtex 2 Pro 70 FPGA chips directly connected to four dual-data-rate 2, 240-pin DRAM DIMMs, with a maximum capacity of 4 Gbytes per FPGA. Figure 2 shows a block diagram of the compute module. The design organizes the four DIMMs into four independent DRAM channels, each running at 200 MHz (400 DDR) with a 72-bit data interface (for a 64-bit data width without error-correcting code). Therefore, the peak aggregate memory bandwidth is 12.8 Gbps per FPGA.

Each module uses four FPGAs for computation and one for control. The control FPGA has additional global interconnect interfaces and controls signals to the secondary system components, including those for temperature and voltage monitoring, Digital Video Interface display, the universal serial bus, and so on. We classify the connectivity on a single compute module into two types of connections: on-board low-voltage CMOS (LVCMOS) parallel and off-board multigigabit transceiver (MGT) serial.

The local mesh connects the four compute FPGAs on a 2D grid. Each link between the adjacent FPGAs on the grid provides over 40 Gbps of data throughput per link. The four down links from the control FPGA to each of the computing FPGAs provide up to 20 Gbps per link. These direct FPGA-to-FPGA mesh links form a high-bandwidth, low-

integer or fixed-point computational throughput.

Another key differentiator of BEE2 is its programming environment. Most commercial reconfigurable computers separate the microprocessor and FPGA programming. They use traditional, sequential high-level languages (such as C or C++) for the microprocessor and low-level hardware description languages (such as Verilog or VHDL) for the FPGAs. The discrepancy in design descriptions as well as computation models leads to an awkward interface between the microprocessor and reconfigurable fabric; thus, the interface is typically ad hoc and application specific. Instead, the BEE2 system uses a high-level block diagram design environment based on Mathworks Simulink and the Xilinx System Generator library. BEE2 uses one unified computation model—synchronous dataflow—for both the microprocessor and the reconfigurable fabric, enabling more flexible user-facilitated partitioning between the hardware and software. We use automatic compilation tools latency mesh network for the FPGAs on the same compute module; so it is possible to aggregate all five FPGAs to form a virtual FPGA with five times the capacity.

All off-module connections use the MGTs on the FPGA. Each individual MGT channel is software-configurable to run at 2.5 or 3.125 Gbps with 8B or 10B encoding and channel-bonded every 4 MGTs into a physical InfiniBand 4X (IB4X) electrical connector, to form a 10-Gbps full-duplex (20-Gbps total) interface. The IB4X connections are AC coupled on the receiving end to comply with the InfiniBand (http://www.infinibandta.org) and the 10GBase-CX4 specification (http://www.ieee802.org/3/10GBCX4/). There are a total of 18 IB4X connectors on each BEE2 module—two from the control FPGA, four from each of the four compute FPGAs—for a total 360-Gbps off-module communication bandwidth. These MGT interfaces can serve as direct intermodule communication, an InfiniBand or 10-Gbps Ethernet packet-switched network, or direct I/O connection to external high-bandwidth real-time devices, such as gigahertz ADCs or DACs.

Global communication networks. The BEE2 design supports a variety of global connection schemes to accommodate various applications. Figure 3 illustrates the three basic types of global communication networks: a low-latency fourary global communication tree, a high-bandwidth nonblocking crossbar, and a 10 or 100 Base-T Ethernet.

Each BEE2 compute module can serve as a global communication tree node, connecting up to 16 other compute modules and up to two independent parent nodes. This type of tree communication network is useful for data aggregation or distribution. Using the IB4X physical connections, the compute modules can also form many other types of network topology, such as 3D mesh.

For applications that require high bisection bandwidth for random communication among many compute modules, we designed the BEE2 system to take advantage of the commercial network switch technologies, such as InfiniBand or 10-Gigabit Ethernet. The crossbar switch provides the highest throughput connection in the BEE2 system with commercial switches currently reaching 244, 4¥ ports in a single physical 9U chassis, aggregating to 4.88-Tbps bisection bandwidth.

The regular 10 or 100 Base-T Ethernet connection, available only on the control FPGA, provides an out-of-band communication network for the user interface, low-speed system control, monitoring, and data archiving. We designed the compute module to run Linux on the control FPGA with a full Internet Protocol network stack.

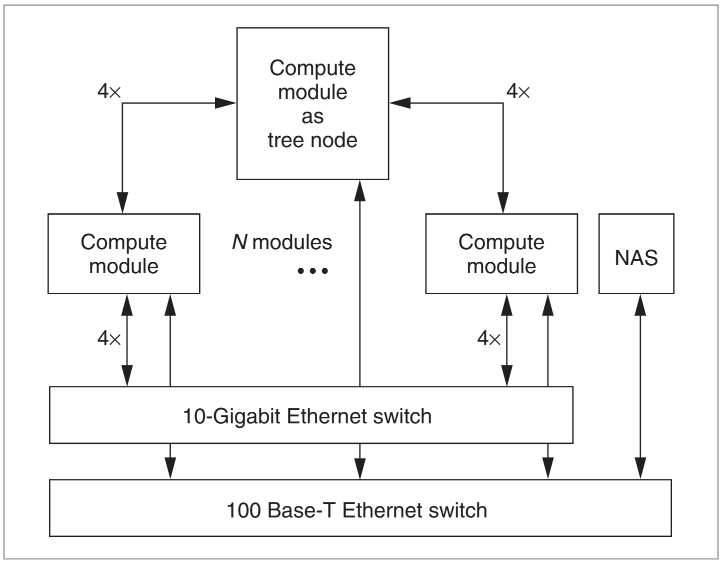

Compute module as tree node
4×
4×
Compute module
N modules
• • •
Compute module
NAS
4×
4×
10-Gigabit Ethernet switch
100 Base-T Ethernet switch

Figure 3. Global communication network.

Mechanical design. Each BEE2 module resides in one of the 10 blades in a custom 8U rack-mounted chassis. With two of the 10 blades reserved for AC or DC power supplies, each chassis packs eight BEE2 modules, hence a standard 42-to-48U rack can host up to 40 BEE2 modules. With each BEE2 module designed to dissipate a maximum 250 W (similar to a standard 1U dual-processor computer server), each rack has a power budget of 10 KW (12-KW AC input). Such a system with 40 BEE2 modules (200 FPGAs) can deliver up to 16 TOPS (32-bit integer) or 2 Tflops, with up to 800 Gbytes of DRAM, and over 7.2 Tbps of full-duplex I/O bandwidth.

Programming models

Because of the diverse application domains that BEE2 targets, any single programming model would not be optimal for all applications; hence the need for domain-specific programming models that can fully exploit BEE2's computing power.

Currently, the most mature programming model for BEE2 is the synchronous dataflow model for DSP and communication applications. Commercial tools (Mathworks Matlab/Simulink and Xilinx System Generator) and BWRC-developed automation tools provide automatic mapping from high-level block diagrams and state machine specifications to FPGA configurations. This programming model and tool flow have proven successful on a variety of projects at BWRC,[11-13] particularly in digital signal processing and other data-path-intensive streaming applications. To extend this model to support BEE2-specific hardware, we are currently developing stream-based design abstractions for external DRAMs and global communication networks.

The BEE2 DSP programming model uses synchronous dataflow diagrams specified in the Mathworks Simulink environment to spatially describe the application algorithm. Using Xilinx System Generator, we abstract the physical FPGA fabric into a set of parameterizable library blocks, including those for arithmetic operators, control operators (such as demultiplex and merger), memory interfaces (for SRAM or DRAM), and I/O interfaces (such as chip to chip). Analogous to an instruction set architecture for microprocessors and DSPs, this functional-level abstraction on FPGAs is largely invariant through generations of FPGAs, at least those from the same vendor. For example, the last four generations of Xilinx FPGAs (from Virtex, Virtex-II, Virtex-II Pro, and Virtex-4) exhibit the same functional-level abstraction.

In addition to the basic operators, the higher-level libraries also provide complex operations—such as those for FFT, finite impulse response (FIR), and Viterbi decoding—much in the same way as the dedicated accelerators implemented on DSPs. Nevertheless, these special functions are highly architecture dependent, and can vary from one FPGA generation to another.

The Simulink environment also provides a cycle-accurate, bit-true emulation of the FPGA hardware. This environment simulates applications with the correct behavior for the FPGA hardware and can accurately and quickly estimate hardware resource utilization. This estimation approach is important because it eliminates the need to run through the complete hardware synthesis tool flow down to the physical FPGA hardware to verify functionality or to obtain resource utilization statistics. Currently, besides generating the final hardware implementation, the only reason for running the full FPGA back-end flow is to accurately determine the design's clock rate.

Overall, the BEE2 DSP programming environment provides a similar level of usability as assembly language programming on DSPs. Despite the lack of efficient high-level compiler technology, the synchronous dataflow programming model for FPGAs extends naturally into multiple-FPGA systems. It is possible to tightly interconnect multiple FPGAs, as on a single BEE2 module, to form a virtual FPGA, therefore enabling the partition of the dataflow graph in the same fashion off chip as on chip. When combined with high-bandwidth serial links, the virtual FPGA concept can even extend, in a fashion, to include physical FPGAs located across many boards or modules.

BEE2 performance evaluation

In this article, we evaluate the computational efficiency of the BEE2 system on three representative applications from the radio astronomy signal-processing domain:

- a billion-channel spectrometer,
- a 1,024-channel polyphase filter bank (PFB), and
- a two-input 1,024-channel correlator.

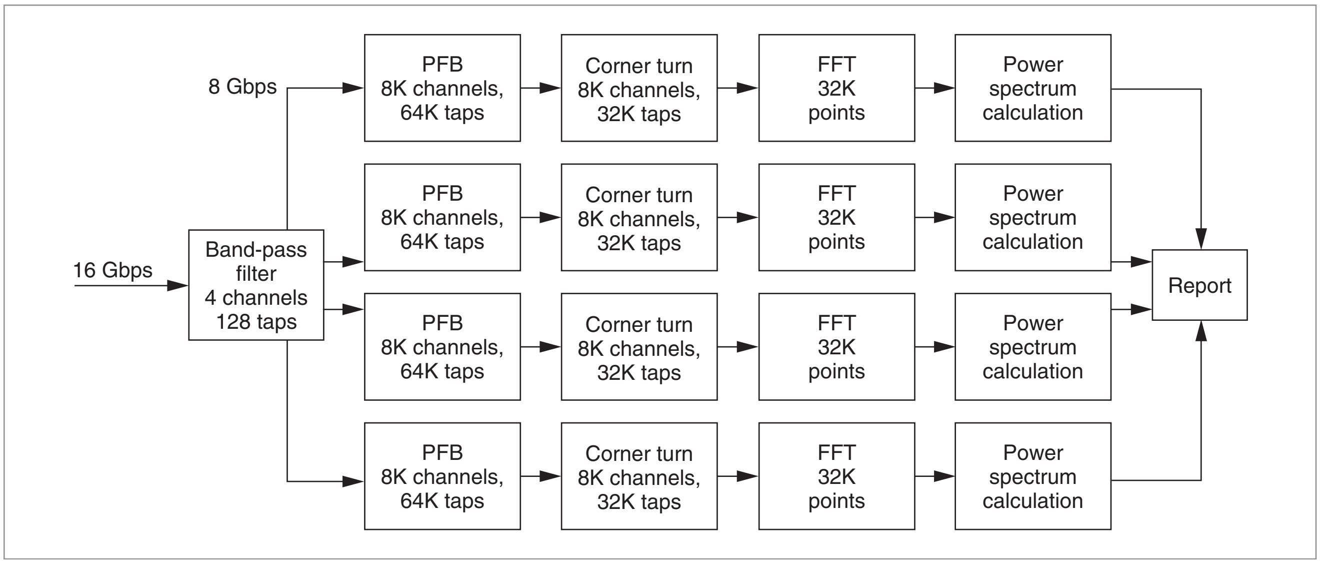
Spectrometer

The first application is a spectrometer, developed in collaboration with the Search for Extraterrestrial Intelligence (SETI, http://seti.berkeley.edu) project at the University of California, Berkeley. SETI's scientific requirements demand the spectrometer to have a spectral resolution of less than 1 Hz. The goal of this application is to produce a subhertz spectral resolution of over 800 MHz, hence delivering a billion-channel real-time spectrometer in a single BEE2 module.

As Figure 4 shows, the data processing of the spectrometer starts with 16-Gbps digital inputs from the radio telescope antenna ADC, which digitizes 4-bit in-phase and quadrature (4-bit I, Q) samples of two polarization signals and sends through two IB4X (or CX4) cables to the BEE2 module's control FPGA. Using a 128-tap 4-channel PFB implemented in the control FPGA, the 800-MHz complex, dual polarization input signal stream is split into four 200-MHz 8-bit complex, dual polarization streams; each stream carries one-fourth of the 800-MHz spectrum. Each of these streams then goes through the LVCMOS links on the BEE2 module to each of the four compute FPGAs, which implement a local spectrometer to split the 200-MHz input signal into 256 million, 0.745-Hz channels. The 256-million-channel spectrometer consists of three major steps: the 8K-channel PFB, followed by a corner turn, and finally a 32K-point FFT. The 8K-channel 64K-tap PFB splits the 200-MHz bandwidth input signal into coarse bands of 24.4 KHz each. Because the PFB outputs data in channel order, a corner turn stage is required to reorder the data in time sample order before it goes to the 32K-point FFT stage. The compute FPGAs then accumulate the spectral results at each frequency bin for 10 ms to increase the signal-to-noise ratio. Finally, they calculate the power spectrum and report bins with a power of 20 dB over the local average to the control FPGA, which relays the results to an external PC over the Ethernet interface.

Polyphase filter bank

The second application is a 1,024-channel PFB for



8 Gbps
16 Gbps
Band-pass filter 4 channels 128 taps
PFB 8K channels, 64K taps
Corner turn 8K channels, 32K taps
FFT 32K points
Power spectrum calculation
Report

Figure 4. Billion-channel spectrometer.

dual polarization inputs. This is a common step shared by many applications for radio telescope arrays. A PFB is essentially a decimated complex FIR filter followed by an FFT operation, as Figure 5 shows. It's mainly used as a more efficient implementation than a bank of adjacent band-pass filters, and it divides the input spectrum into a set of adjacent spectral channels. In this case, we use an 8,192-tap FIR filter decimated by 1,024, along with a 1,024-point FFT. The inputs to the PFB are two complex data streams (8 bits each, real and imaginary), one for each of the polarizations from the radio telescope. The FIR coefficients are 12-bit real numbers, and FFT coefficients are 24-bit complex numbers (12 bits each, real and imaginary). All multiplications use 16-bit multipliers and 32-bit additions.

In a typical telescope array environment, each antenna requires at least one PFB unit, so the PFB design's goal is to pack as many PFB units as possible into a signal physical FPGA while sustaining the input data rate. At a target input clock rate of 250 MHz (a 1-Gbps data rate per PFB), each XC2VP70 FPGA on a BEE2 module can accommodate four PFB units, using the four IB4X interfaces. So the ATA-350 system requires 88 BEE2 modules for all 350 antennas per each 250-MHz intermediate frequency (IF) band.

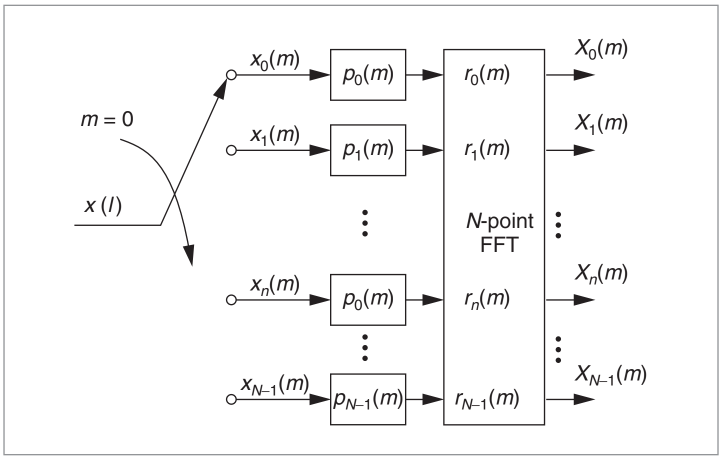


Figure 5. Polyphase filter banks.

Correlator

The third application is a 1,024-channel, two-input cross-correlator, the building block for the full N^2 antenna correlator. Each of the two inputs is a continuous stream of quantized spectral data from the outputs of the PFB unit on each antenna. The correlator accumulates the complex correlation results at 10 ms to 30 s, per the requirements of the image. The input spectral data is in an 8-bit complex number representation (4 bits each, real and imaginary), hence we perform the multiplication using 4-bit input multipliers and then accumulate it on 22-bit registers. At a target clock rate of 200 MHz, we can pack five cross-correlator units in a single XC2VP70 FPGA. Correlating all the antennas in the ATA-350 system would require over 3,000 BEE2 modules for each 200-MHz band.

Performance results

We mapped all three benchmark applications to FPGAs using the programming model and tool flow we described earlier. The FPGA performance numbers are

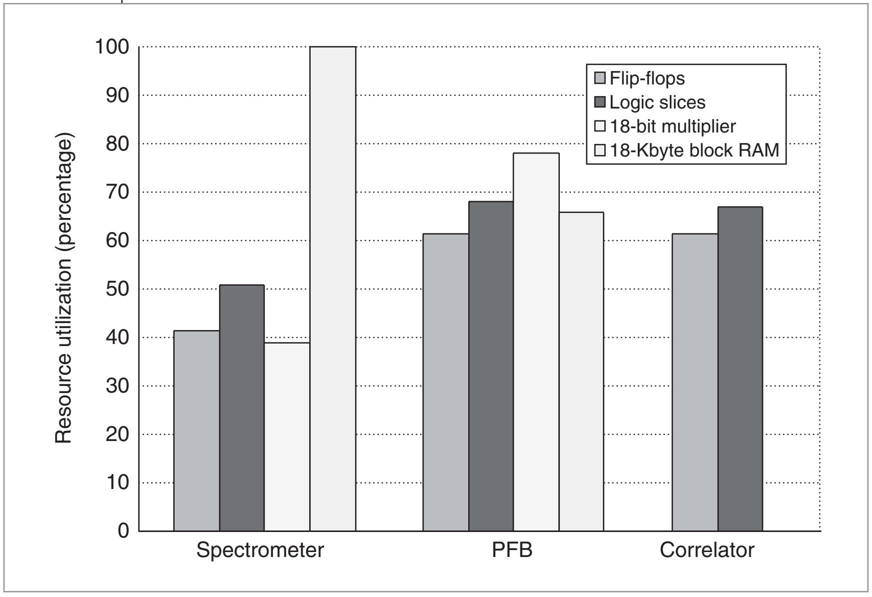


Figure 6. Resource utilization on the XC2VP70 FPGA.

from the post place-and-route timing analysis of the Xilinx ISE design tool suite; therefore, they are identical to that for the final clock rate on the actual hardware. We estimated the FPGA power consumption using the Xilinx XPower tool.

Resource utilization. Figure 6 shows the physical FPGA resource utilization for each benchmark. Typical utilization measurements on DSPs or microprocessors consider utilization as the temporal usage of the functional units. In contrast, FPGA resource utilization is the measure of both the spatial and temporal allocation of the functional units. In this particular set of benchmark applications, however, since all allocated functional units are running at 100% duty cycle, the spatial utilization of the FPGA is the overall utilization.

In the billion-channel spectrometer application, each of the four 256-million-channel local spectrometers uses the entire on-chip block RAM on a Virtex-II Pro 70 FPGA and about 50% of the other resources. This imbalance arises from the large on-chip memory utilization of the 32K-point FFT. On the other hand, the resource utilization is much more balanced in the PFB application. Although the leftover resources are enough to implement one more PFB unit on the FPGA (a 20% increase in PFB units), this would push the FPGA utilization near its limit, seriously and negatively affecting the clock rate. In this case, the mapping of the five PFB units can only run at half the target 250-MHz clock rate, not only reducing the overall throughput but also violating the input data rate requirement. This clock rate degradation for high FPGA utilization mainly comes from the congestion of on-chip routing resources.

Both the spectrometer and the PFB application use 16-bit multiplications and 32-bit additions, which have equivalent operations on the DSP processors. However, the correlator application performs multiplications on 4-bit inputs. Although DSPs can multiply 4-bit inputs, they have hardware multipliers optimized for 8-bit inputs, thus putting the DSPs at a disadvantage. On the other hand, FPGAs, with their bit-level configurability, can provide a much higher level of spatial parallelism. This application uses neither the dedicated on-chip 18-bit multiplier nor the block RAMs. Similar to the PFB mapping case, you could map six correlators (instead of five) on the FPGA, but at the cost of an unacceptably low clock rate.

Computational efficiency. Next, we compare the FPGA performance of the three benchmarks to that of the state-of-the-art Texas Instruments DSPs and Intel Pentium 4 microprocessor. The FPGA we used in this study is the same 130-nm CMOS technology XC2VP70-7FF1704C chip in the BEE2. Two DSP chips from Texas Instruments are the 130-nm CMOS C6415-7E chip running at 720 MHz and the 90-nm CMOS C6415T-1G chip running at 1 GHz.[14] The microprocessor we used was the latest Intel 90-nm Pentium 4 570 running at 3.8 GHz. The microprocessor and the DSP performances assume the peak performance for the corresponding arithmetic precision used in the applications. We report the DSP power consumption from the vendor power calculation spreadsheet[15] based on full utilization and maximum clock rate. The microprocessor power consumption assumes full utilization as well. Hence, in terms of performance, this comparison is intrinsically in favor of the DSPs and microprocessors. All chip costs are the vendors' price for 1,000-chip lots.

We measured performance in billions of MACs per second (GMACs/s), as Figure 7 shows. For example, in the spectrometer case, at the targeted 200-MHz clock rate, each of the four compute FPGAs performs 28.8 billion multiplies/s and 39.2 billion additions/s. The control FPGA performs 32 billion multiplies/s and 32 billion additions/s. Thus, on average, each FPGA performs 29.44 GMACs/s.

In terms of computational throughput per chip, the FPGAs in BEE2 can outperform the 720-MHz DSP by a factor of 10 to 34, 7 to 25 times faster than the 90-nm 1-GHz DSP, and 4 to 13 times faster than the latest Pentium 4. Figure 8 shows power efficiency results; the XC2VP70 FPGA delivers 72 to 106% more throughput on 16-bit operations compared to the DSPs, and on 4-bit operations, it delivers more than 11 times the DSP throughput. Compared to the microprocessor, the FPGAs provide over two orders of magnitude more power efficiency.

We also compared the various chips in terms of cost, as Figure 9 shows. The FPGA's compute throughput per unit chip cost is 20 to 307% more than that of the 1-GHz DSP, and 50 to 505% more than that of the 3.8-GHz Pentium 4.

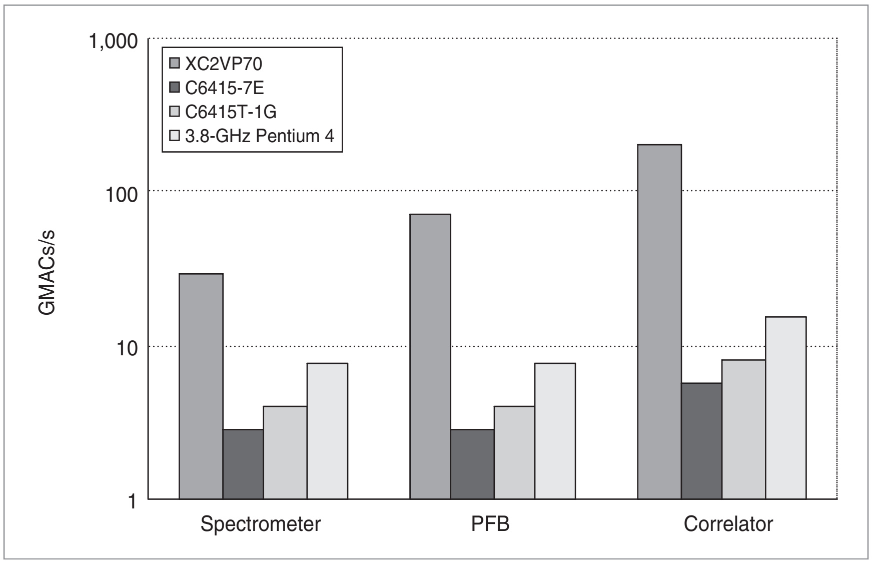


Figure 7. Absolute performance comparison.

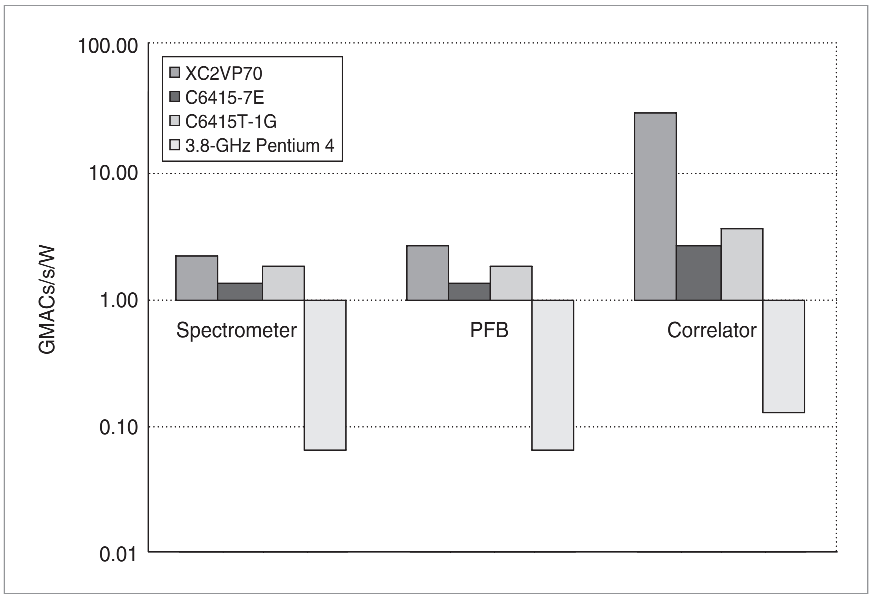


Figure 8. Power efficiency comparison.

System integration. In addition to the computational-efficiency advantage at the individual chip level, FPGAs also provide more energy- and cost-efficient solutions at the system level. The hardware's overall power consumption and cost must include system components, such as memory, printed circuit boards, and network interfaces. In this area, FPGAs offer a more integrated solution. In addition to the current practice of integrating general-purpose processor cores on the FPGA fabrics, the latest Virtex-4 FX FPGA now provides built-in hardware support for Ethernet and ADCs.

In applications where the algorithm is naturally partitionable to take advantage of cheaper and smaller-capacity FPGAs, we can further reduce the overall system cost. For example, in the 1,024 PFB application, a natural division of the algorithm would split the four PFB units into four physical FPGAs with smaller capacity but the same speed, such as the XCV2P20-7. Therefore, the smaller FPGA can maintain the same clock rate requirements, but it would cost much less than a quarter of the larger FPGA's price. A 1,000-unit lot of XC2VP20-7s costs only $367 and delivers only 49.1 (GMACs/s)/dollar—58.4% more compute throughput per dollar than the XC2VP70. At a power consumption of 3.83 W, the XC2VP20-7 is also 72.8% more power efficient.

Nevertheless, using smaller-capacity FPGAs will marginally increase the system cost, because of the additional external components and extra board resources required. Detailed quantitative analysis on the overall system cost would depend on the target application and many other factors that are beyond the scope of this article.

We now estimate the BEE2 module hardware cost at $20,000 when manufactured in quantities of several lots of 10, including full 20-Gbyte memories. Compared to the typical cost of $4,000 for an Intel single-processor server,

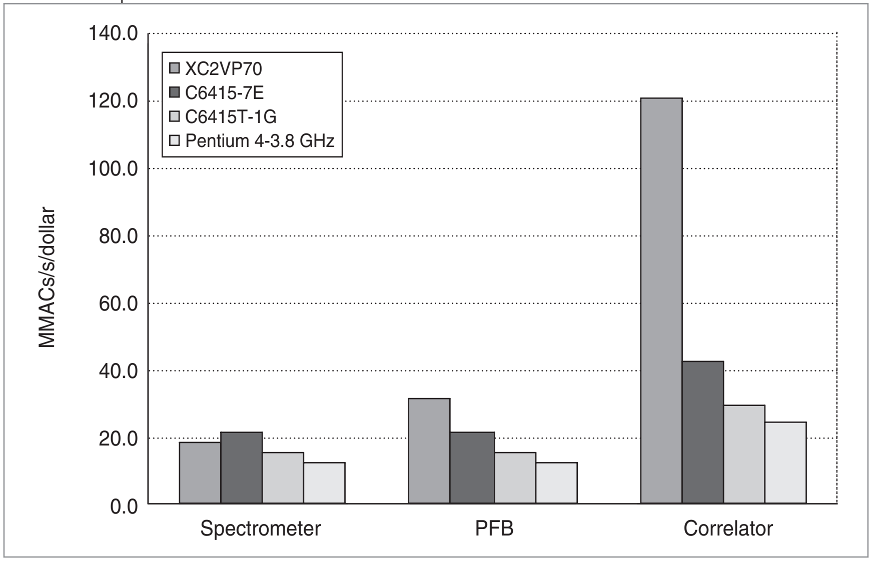


Figure 9. Cost efficiency comparison.

the BEE2 is five times more expensive, but it provides over 50 times more compute throughput; hence, it still retains 10 times more computing throughput per unit of hardware system cost.

BEE2 IS OUR FIRST ATTEMPT at creating a universal reconfigurable computing system that can target a wide range of application domains, starting from high-performance digital signal processing where FPGAs are a proven technology, to scientific computing, where FPGA use remains a relative novelty. The introduction of modular FPGA computing platforms such as BEE2, and improvements in domain-specific programming models, can perhaps overturn the negative image of FPGAs as a user-unfriendly technology.

As our work extends into other computational domains, we will explore additional programming methodologies. One of primary interest is the message-passing interface (MPI) standard (http://www.mpi-forum.org), which has been the predominant programming model for existing microprocessor-based supercomputers. We plan to implement the MPI library for our HERC as a means to ease the task of porting supercomputer applications. Many novel opportunities exist for specializing MPI functions in the reconfigurable fabric.

We could, for example, accelerate the computational kernels and inner loops of the programs using the reconfigurable fabric. For this, we will rely on the use of predefined library elements and on recent work in automatic C-to-hardware compilation. The PowerPC core on each FPGA provides a conventional RISC processor to run the program's nonaccelerated parts.

In a sense, MPI forms an abstract machine. It is the target for the application algorithm developer and, for us, the organizing structure for the communication circuits built in the reconfigurable fabric. We would like to explore other abstract machine architectures tailored for specific problem domains. For instance, these might include abstract architectures for regular grid problems and cellular automata; and irregular graph problems, such as transistor-level circuit simulation. These abstract machine models will be the target for domain-specific programming languages, and the organizing principle for automatic problem mapping and optimization. ■

■ References

1. C. Chang et al., "Implementation of BEE: A Real-Time Large-Scale Hardware Emulation Engine," *Proc. 2003 ACM/SIGDA 11th Int'l Symp. Field-Programmable Gate Arrays* (FPGA 03), ACM Press, 2003, pp. 91-99.
2. J.P. Durbano et al., "FPGA-Based Acceleration of the 3D Finite-Difference Time-Domain Method," *Proc. 12th Ann. IEEE Symp. Field-Programmable Custom Computing Machines* (FCCM 04), IEEE CS Press, 2004, pp. 156-163.
3. W. Chen et al., "An FPGA Implementation of the Two-Dimensional Finite-Difference Time-Domain (FDTD) Algorithm," *Proc. 2004 ACM/SIGDA 12th Int'l Symp. Field-Programmable Gate Arrays* (FPGA 04), ACM Press, 2004, pp. 213-222.
4. A. DeHon, R. Huang, and J. Wawrzynek, "Hardware-Assisted Fast Routing," *Proc. 10th Ann. IEEE Symp. Field-Programmable Custom Computing Machines* (FCCM 02), IEEE CS Press, 2002, p. 205.
5. M. Wrighton and A. DeHon, "Hardware-Assisted Simulated Annealing with Application for Fast FPGA Placement," *Proc. 2003 ACM/SIGDA 11th Int'l Symp. Field-Programmable Gate Arrays* (FPGA 03), ACM Press, 2003, pp. 33-42.
6. T.J. Callahan et al., "Fast Module Mapping and Placement for Datapaths in FPGAs," *Proc. 1998 ACM/SIGDA 6th Int'l Symp. Field Programmable Gate Arrays* (FPGA 98), ACM Press, 1998, pp. 123-132.

7. *Xilinx Virtex-4 Data Sheet*, DS112, version 1.2, Xilinx Corp., 18 Dec. 2004.
8. D. Caliga and D.P. Barker, "Delivering Acceleration: The Potential for Increased HPC Application Performance Using Reconfigurable Logic," *Proc. SC2001*, IEEE CS Press, 2001, p. 28.
9. C.N. Keltcher et al., "The AMD Opteron Processor for Multiprocessor Servers," *IEEE Micro*, vol. 23, no. 2, Mar. 2003, pp. 66-76.
10. *Xilinx Virtex-II Pro Data Sheet*, DS083, version 4.1, 11 Nov. 2004.
11. K. Kuusilinna et al., "Real-time System-on-Chip Emulation," *Winning the SoC Revolution: Experiences in Real Design*, G. Martin and H. Chang, eds., Springer, 2003, pp. 229-253.
12. C. Chang et al., "Rapid Design and Analysis of Communication Systems Using the BEE Hardware Emulation Environment," *Proc. 14th IEEE Int'l Workshop Rapid Systems Prototyping*, IEEE Press, 2003, pp. 148-154.
13. K. Kuusilinna et al., "Designing BEE: a Hardware Emulation Engine for Signal Processing in Low-Power Wireless Applications," *EURASIP J. Applied Signal Processing*, vol. 2003, no. 6, May 2003, pp. 502-513.
14. *Texas Instruments TMS320C6414T/15T/16T Fixed-Point Digital Signal Processors Datasheet* (Rev. D), 31 Oct. 2004; http://www-s.ti.com/sc/ds/tms320c6414t.pdf.
15. *Texas Instruments TMS320C6414T/15T/16T Power Consumption Summary,* 2 Aug 2004; http://www-s.ti.com/sc/psheets/spraa45/spraa45.pdf.

Acknowledgments

The BEE2 project's radio astronomy application development is in collaboration with the SETI@Home and Serendip (Search for Extraterrestrial Radio Emissions from Nearby Developed Intelligent Populations) projects at the UC Berkeley Space Science Laboratory (Dan Werthimer), as well as the UC Berkeley Radio Astronomy Laboratory (Melvyn Wright). We also thank Xilinx for its generous donation of FPGAs, software tools, and engineering assistance. Many thanks to all the hard work by the students and staff members of the BEE2 team: Pierre-Yves Droz, Greg Gibeling, Nan Zhou, Yury Markovskiy, Zohair Hyder, Adam Megacz, Alexander Krasnov, Hayden So, Kevin Camera, Brian Richards, and Susan Mellers. The BEE2 project is funded by the Microelectronics Advanced Research Corp. (Marco), C2S2, and the Multidisciplinary Research Program of the University Research Initiative (Muri) programs, as well as BWRC industrial-sponsor companies.

Chen Chang is a PhD student of electrical engineering and computer sciences at the University of California, Berkeley. His research interests include large-scale FPGA-based real-time computer systems, digital system design automation and hardware emulation, and wide-band antenna array signal processing systems. Chang has an MS in electrical engineering and computer sciences from the University of California, Berkeley. He is a member of the IEEE and the IEEE Computer Society.

John Wawrzynek is a professor of electrical engineering and computer sciences at the University of California, Berkeley, where he is the head of the Berkeley Reconfigurable Architectures, Software, and Systems (Brass) group. His research interests include the design and application of reconfigurable computing systems, parallel computing architectures, and VLSI design. Wawrzynek has an MS in electrical engineering from the University of Illinois, Urbana-Champaign, and an MS and a PhD in computer science from the California Institute of Technology. He is a member of the IEEE and the IEEE Computer Society.

Robert W. Brodersen is the John R. Whinnery Distinguished Professor in the Department of Electrical Engineering and Computer Science at the University of California, Berkeley. He is also the co-scientific director of the Berkeley Wireless Research Center (BWRC) where his research focuses on new applications of IC as applied to personal communications systems with an emphasis on wireless communications, low-power design, and the CAD tools necessary to support these activities, including system-level real-time prototyping, ultra-wideband (UWB) radio systems, multiple-carrier multiple-antenna algorithms, and microwave CMOS radio design. Brodersen has a BS in electrical engineering and mathematics from the California State Polytechnic University, Pomona, and an MS and a PhD in engineering from the Massachusetts Institute of Technology. He is a member of the National Academy of Engineering and a Fellow of the IEEE.

■ Direct questions and comments about this article to Chen Chang, 2108 Allston Way, Suite 200, Berkeley, CA 94704; chenzh@eecs.berkeley.edu.