# Implementation of BEE: a Real-time Large-scale Hardware Emulation Engine

*Chen Chang[1], Kimmo Kuusilinna[1,2], Brian Richards[1], Robert W. Brodersen[1]*

[1]University of California, Berkeley
Berkeley Wireless Research Center
2108 Allston Way, Berkeley, CA 94704
{chenzh, kimmo, richards, rb}@eecs.berkeley.edu

[2]Tampere University of Technology
Institute of Digital and Computer Systems
P.O.BOX 553, FIN-33101 Tampere, Finland

## ABSTRACT

This paper describes the hardware implementation of a real-time, large-scale, multi-chip FPGA (Field Programmable Gate Array) based emulation engine with a capacity of 10 million ASIC (Application Specific Integrated Circuits) equivalent gates. Attainable system operation frequency can exceed 60 MHz, and the system throughput has been empirically verified to achieve 600 billion 16-bit additions per second. The emulator is custom designed to maximize the performance and resource utilization for a range of telecommunication and digital signal processing applications. With its high-speed interconnect architecture and large external I/O bandwidth, the emulator excels in prototyping real-time systems that have strict timing, logic capacity, and data rate requirements. Our development efforts are guided by such ongoing projects as ultra-wide band (UWB) and multi-channel-multi-antenna (MCMA) radio systems research.

## Categories:

I. Computing Methodologies
I.6 Simulation and Modeling
I.6.7 Simulation Support Systems

## Subject Descriptors:

Hardware Emulation Engine

## General Terms:

Algorithms, Performance, Design, Experimentation, Verification

## Keywords:

FPGA, Hardware Emulation, Rapid-Prototyping



FPGA'03, February 23-25, 2003, Monterey, California, USA.


## 1. INTRODUCTION

With the increasing complexity and integration of digital and analog systems, the computing power required for detailed cycle accurate and bit-true software simulation of even a single subsystem can easily become prohibitive. In addition, the excessively long and non-deterministic execution time of the simulation makes the accurate verification of integrated systems with heterogeneous components very difficult. In communication systems, digital base-band processing needs to be verified at the same data rate as the analog radio front-end in order to test the performance of the system in a real-world environment. This typically requires the whole simulated system to be run at least at tens of megahertz speed, with cycle accuracy and bit-level detailed modeling of the final target system. This is commonly 100,000 to 1 million times faster than what the best simulation software can do at the same abstraction level [1].

One alternative to simulation is hardware emulation. A typical hardware emulator utilizes an array of FPGAs to directly emulate the digital portion of the system using reconfigurable hardware instead of software running on general-purpose processor. Using various schemes of inter-FPGA connection topologies, these emulators can achieve overall system performance up to a few megahertz. A typical use for such an emulator would be in-circuit verification of a gate-level netlist with exhaustive test vectors. Another alternative is rapid prototyping based on FPIC (Field Programmable Interconnect Component), FPCB (Field Programmable Circuit Board), and FPGA technologies. These systems enable full functional verification at system operation frequency around a few tens of MHz, and offer the flexibility of integrating heterogeneous components, such as FPGA emulated ASIC designs, DSP chips, and general-purpose processors.

Ideally, a real-time reconfigurable hardware emulator should have multi-million ASIC gate logic capacity, identical system operation frequency to the final target system, and be able to seamlessly integrate multiple heterogeneous components. This presents three major challenges. First, in ASIC emulation, the gate-level ASIC structural netlist is retargeted to the FPGA technology, which can reduce the on-chip performance of a single FPGA emulated system to a fraction of the ASIC performance. Second, ASIC designs can be much bigger than the capacity available on even the state-of-the-art FPGA chips. Therefore, the original ASIC netlist needs to be partitioned into multiple pieces, each implemented on a different FPGA, which can lead to inter-FPGA routing congestion. Sometimes high pin-

count crossbar chips can be used to maximize local inter-FPGA connection while maintaining global routability at the expense of introducing additional delays between FPGAs. Due to insufficient routing resources, time-multiplexed virtual-wiring [2] may be necessary for emulation systems, at the expense of further reducing the overall system speed. Third, most commercial emulators either have limited external parallel I/O bandwidth or complex high-speed serial links, which makes it difficult to connect to analog front-ends. Furthermore, a typical way to implement external I/O connections is through buffered memory to synchronize the interface between digital and analog, which increases the interface design complexity.

Although real-time emulation of arbitrarily fast ASIC designs is impractical, designs with a clock frequency below 60 MHz can be prototyped using the current FPGA technology. For a range of low-power communication system designs, a low system clock frequency is desirable to save power. Direct-mapped parallel architectures are a straightforward way to achieve the necessary performance at the reduced system operation frequency [3], therefore allowing real-time emulation with FPGAs, a preferred solution. The *Berkeley Emulation Engine* (BEE) was built especially for this range of applications. The BEE system has the large design capacity of a hardware emulator—10 million ASIC equivalent gates per module, the extensibility of a rapid-prototyping system, a large amount of simple and flexible external interconnects, and the real-time system operation speed exceeding 60 MHz. It is primarily used to emulate real-time communication and DSP algorithms. In addition, a parallel path to the prototyping flow leads automatically from the top-level description to an ASIC implementation. The targeted communication systems include UWB (Ultra-Wide Band) radio and MCMA (Multi-Channel-Multi-Antenna) applications. Both the UWB radio with 1.2 GHz sample rate and the MCMA radio with up to 16 parallel radio front-ends have the worst-case external I/O bandwidth requirement of up to 90 gigabits per second, while the low-power criteria forces the digital portion of the system to have multi-million-gate parallel processing components running at speeds up to 60 MHz.

The rest of this paper describes the hardware architecture of the BEE system in Section 2 and signal integrity issues along with the solutions found in Section 3. Hardware performance results are presented in Section 4. Due to the scope of this paper, the integrated design flow targeting both the BEE and the ASIC implementation can only be briefly introduced in Section 5. Finally Section 6 concludes the paper.

# 2. SYSTEM ARCHITECTURE

## 2.1 Overview

Figure 1 depicts the overall structure of the BEE system, which consists of three major components: BEE Processing Units (BPUs), analog front-ends, and host servers. Users first create their target design under Matlab Simulink [4], then synthesize on the host server into FPGA bit-streams, which are downloaded later through Ethernet to BPUs for emulation. Each BPU can directly connect to multiple analog front-ends through either Low-Voltage TTL (LVTTL) or Low-Voltage Differential Signaling (LVDS) signals. Multiple BPUs can also be directly linked to form aggregated systems to increase emulation capacity, or networked through the analog front-end, such as a RF communication front-end, to form wired or wireless asynchronous networks for complete multi-node communication network testing in realistic channel environments.

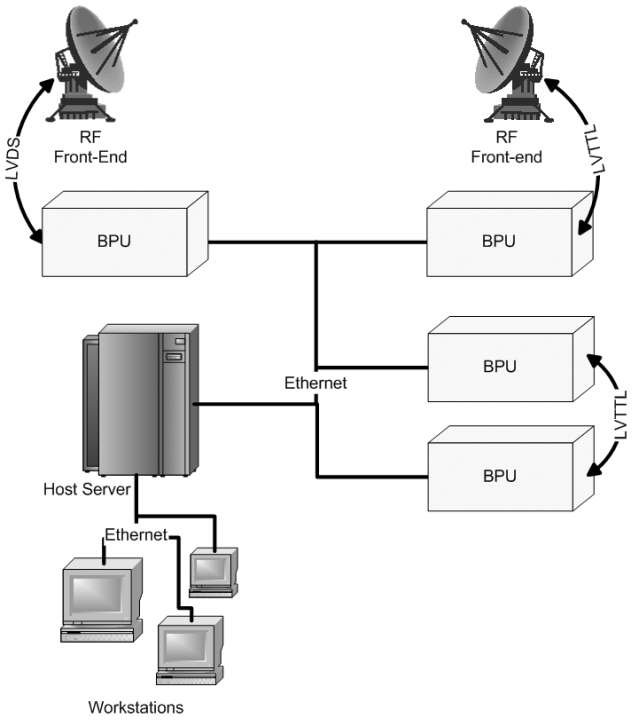


**Figure 1: BEE system overview**

## 2.2 BEE Processing Unit

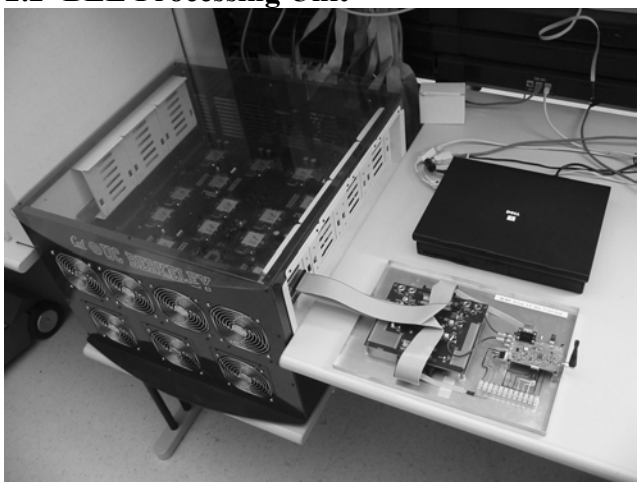

**Figure 2: Complete BPU with a radio front-end**

As shown in Figure 2, the core of a BPU is the Main Processing Board (MPB), which provides the computation power for the system. Eight riser I/O cards, vertically mounted to the MPB, provide a total of 2400 external connections off the BPU. A StrongARM-based single board computer (SBC) establishes the

connection between the BPU and the host server through a 10Base-T Ethernet link. A separate power board (not shown in Figure), along with two modular AC/DC converters, are capable of supplying the system up to 800 W.

Each MPB has 20 Xilinx Virtex-E 2000 chips, 16 ZBT (Zero-Bus Turn-around) 133 MHz synchronous SRAMs, and 8 VHDM-HSD (Very High Density Modular-High Speed Differential) I/O connectors for front-end connection and system expansion.

## 2.3 On-board Inter-FPGA Connections

As for any hardware emulator, the effectiveness of the inter-FPGA connection topology directly affects the performance, the algorithm mapping, and routing capability. The basic architectures analyzed in the literature are mesh (Figure 3a) [5], and partial crossbar (Figure 3b) [6], [7]. Previous research has shown that the partial cross bar is one of the most effective architectures [8], [9], and hybrid architectures, such as the hybrid complete-graph and partial-crossbar (HCGP, Figure 3c), have also been proposed to out-perform pure partial-crossbar structures [10].

Nevertheless, interconnect structure is not the only factor that determines the overall emulator performance. The logic capacity of each individual FPGA, that is, the granularity of the emulation system, also plays an important role in achieving desired performance. On one hand, high-density FPGAs can sometimes relax the inter-chip connection requirements by absorbing more tightly integrated design modules into each FPGA. On the other hand, FPGA utilization is directly affected by the interconnect topology. In an interconnect-constrained multi-FPGA system, the FPGA utilization can easily be driven below 50%. Simply increasing the number of I/Os on the FPGAs often helps little in reducing the routing congestion while it simultaneously increases the printed circuit board (PCB) design complexity.

Given the BEE system target design capacity of 10 million ASIC equivalent gates, 20 high-density Xilinx XC2000E FPGA chips are needed, each with a capacity to emulate half a million logic gates. Then, to maximize the interconnect bandwidth between the chips while achieving above 60 MHz link speed, all 20 FPGA chips need to be placed on the same PCB. Finally, from available Xilinx FPGA packages, the 680-pin chip with 512 user I/Os was chosen to keep PCB design complexity under bound.

To match the direct-mapped architectures, which typically have more local connections than global routes, 16 out of the 20 FPGAs are used to the form a 4-by-4 array, and the 8-way local mesh structure is used to provide local interconnect, thus creating a uniform and tightly integrated reconfigurable fabric among the FPGAs. Each mesh link is 48 bits wide and can be configured at run-time for bi-directional signaling. The FPGAs on the periphery of the board have either 3 or 5 neighboring FPGAs, thus the remaining links are routed to the nearby off-board connectors.

Although the local 8-way mesh is an effective interconnect strategy for direct-mapped architectures, it has two major weaknesses. First, long connections between large design modules, such as in a feedback loop, compete not only with local connection for the available channels, but also FPGAs along the path. For example, if a global 8-bit bus needs to be connected between the top-left FPGA and the bottom-right FPGA, then using only the local-mesh, the connection needs to pass through the two FPGAs in the middle of the board. Each of the FPGAs has to dedicate 16 I/O pins to pass-on the signal. Therefore, the route-through FPGA I/O pin resources are reduced by two times the width of the global connection. Second, control logic, which may be a single large module that connects many data-path modules, competes with data-path components for FPGA capacity and routing channels.

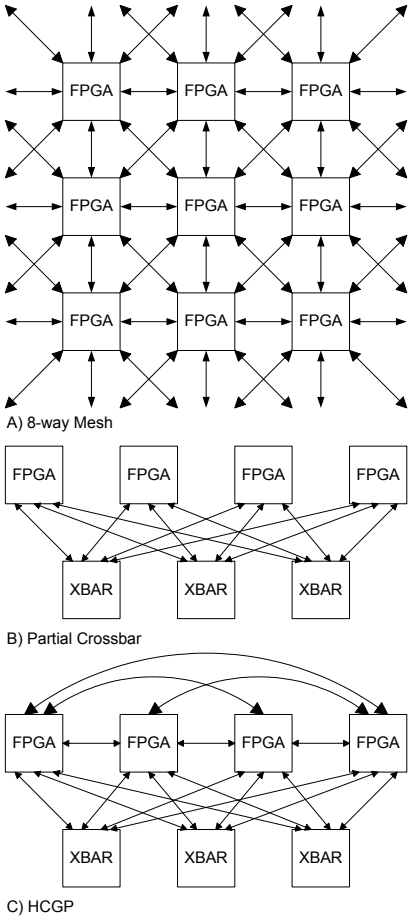


**Figure 3: Various multi-FPGA interconnect topologies**

In light of the above two weaknesses, a global mesh was designed on top of the local mesh. Four FPGAs are used, one in each quadrant of the board. Since these FPGAs can be used as a crossbar for global routing, they are dubbed XBAR for distinction. Each XBAR connects to the neighboring four FPGAs in the same quadrant via a 36-bit wide link and to other XBARs through a 96-bit link in a 3-way second layer mesh. The resulting routing architecture is depicted in Figure 4 and called a *Two-layer Mesh*. Under this structure, global connections can be routed though the XBARs independent of the local mesh. In addition, since each XBAR has a total of 288 inter-XBAR

connections and 144 XBAR-to-FPGA connections, the 2-to-1 ratio ensures efficient utilization of the global routing channels between non-neighboring FPGAs. Furthermore, XBARs can be effectively used for central control logic in a quadrant, while the data-path is distributed among the four FPGAs.

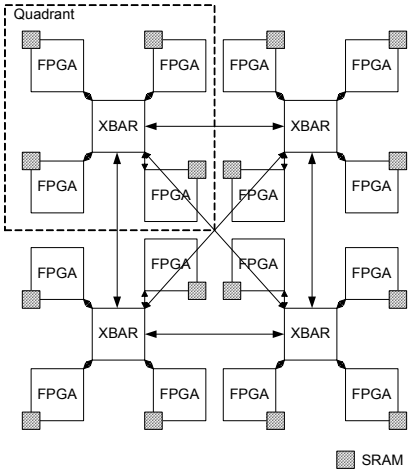


**Figure 4: Second layer XBAR mesh**

When compared to other interconnect topologies, such as simple 8-way mesh or HCGP, with the same four-by-four array of FPGAs used in BEE, the Two-layer Mesh has on average a lower number of hops between FPGAs, which in general leads to lower inter-chip latency. However, this design limits the scalability of the system. Although up to four BPUs can be directly connected using external connections, the increased latency and the reduced connection bandwidth between BPUs drastically reduces the usability of large systems scaled in this fashion.

## 2.4 External I/O Connections

The simplest way to link the BPU to analog front-ends or other BPUs is through external connectors using LVTTL signals. However, when external cables are used, the LVTTL signal strength degrades rapidly with the increase of cable length, limiting the maximum link speed to less than 40 MHz over a 1 meter long ribbon cable. Therefore, when higher than 60 MHz links are desired, Low-Voltage Differential Signaling is used instead. LVDS links can achieve up to 200 Mbps speed over a 2 meter twisted-pair cable.

Using LVDS solves some of the speed and cable length problems; however, it also creates two new problems. First, LVDS signal termination is asymmetrical. Second, the LVDS drivers used on the Xilinx Virtex-E series FPGAs require external source resistor networks to achieve the standard LVDS signal voltage levels [11]. Although both of these problems have been solved using on-chip active termination as provided in the Xilinx Virtex II chips, due to the unavailability of the newer generation FPGA chips at the time of BEE construction, external riser card solution had to be used to maximize the reusability of BEE hardware for many applications.

Riser cards are designed to bridge this link between the MPB and the physical cables to analog front-ends or other BPUs. The 2400 external I/Os from the MPB are broken into 8 groups of 300 signals each. Each group connects to one riser card though a 400-pin connector. Each signal pair of the connector is individually shielded from the other signals, thus lessening cross talk and allowing high-speed differential signals. Immediately after the connector, signals are terminated with appropriate resistor networks on the riser card, and then routed to six 68-pin SCSI (Small Computer Systems Interface) connectors. Using external cables, connections can be established between any two SCSI connectors, therefore also between any two BPUs, a BPU and a front-end, or to form a BPU self loop back. The choice of SCSI connector and cable is mainly due to the high-speed differential signal standard used in SCSI, and the availability of high quality cables and connectors.

Both LVDS and LVTTL signaling standards can be used on the riser cards by simply populating the resistor termination network with footprint compatible resistor chip array packages with different termination structure. In addition to I/O functions, custom riser cards can also be used as expansion modules to integrate heterogeneous components into the BEE system, such as DSP or general-purpose processors, or high-density memory components.

## 2.5 System Controls

Another key feature in the BEE system design was the ease of information propagation between the user and the emulator. The use of the integrated single board computer not only removes the requirement of an external service computer, but also provides the Ethernet interface as a convenient link between the user and the BEE system. With a 206 MHz Intel StrongARM Processor, 32 MByte of SDRAM, 16 MByte Flash ROM, 10 Base-T Ethernet controller and connector, and a compact flash slot for expansion, the SBC can support complex software, such as Linux operating system, Apache web server, and other BEE servicing programs. Therefore, the users of the BEE system can easily log into each BPU and perform the necessary tasks. With the compact flash slot, additional storage, such as a micro drive or a compact flash memory card, can further expand the storage capacity of the SBC.

Through the SBC Linux network login interface, the users can remotely control all functions of the system, such as uploading design files and the read-back of emulation results. The SBC connects to the 20 FPGAs on the MPB through a configuration FPGA, which mainly serves as a bi-directional signal multiplexer between the 16 general-purpose I/O lines from the SBC to over 100 control signals on the MPB, some of which are further connected to the power supply system and chassis control switches. Shown in Figure 5, the control functions of the entire BPU can be classified into the following five categories:

1) programming the FPGAs
2) data read-back from the FPGAs
3) clock domain control
4) power management
5) thermal management

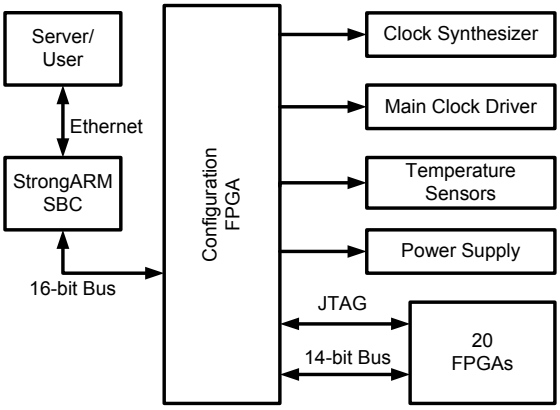


**Figure 5: BEE configuration subsystem**

Processing FPGAs can be programmed by the SBC using either the Xilinx SelectMAP mode [12] or JTAG [13]. The JTAG daisy chain originates from the configuration FPGA, loops through all 20 processing FPGAs, and back to the configuration FPGA. It can be driven directly by the SBC or through an external header using a JTAG cable from a PC. Due to its faster programming speed, the SelectMAP is used as the primary programming mode. The 14-bit configuration bus originates from the configuration FPGA and routes through all 20 FPGAs.

Read-back of the FPGA signal states can be achieved using the JTAG or through the user bus interface. Read-back directly using the JTAG interface is not only too slow, but also too complex for convenient usage in some applications. Using Xilinx ChipScope ILA (Integrated Logic Analyzer), synchronous data can be selectively recorded in on-chip RAM at the same rate as the design, then read back through the JTAG interface and displayed intuitively as waveforms. However, the JTAG speed limitation still makes it hard to achieve read back at high data rate. Therefore, a user bus interface was designed to provide both run-time upload and read-back capability. After initial programming, the 14-bit configuration bus can be used for communication between the FPGAs and the SBC. Between the FPGAs on the user bus, a throughput of 10 Mbytes per second can be achieved, however, between the SBC and FPGA, the throughput is limited by the SBC to 2 Mbytes per second.

Each FPGA on the MPB can be simultaneously driven by four different clock sources. The primary clock provides a synchronous clock domain throughout the entire board. The source of the primary clock is a configurable PLL clock driver, whose output frequency can be digitally programmed between 1 MHz and 200 MHz by the SBC with a resolution of under 5 ppm. The secondary clock consists of four independent clock domains throughout the MPB, one for each quadrant. Each quadrant clock can be independently driven from an external SMA connector. The tertiary clocks independently provide each of the 16 peripheral FPGAs and the four XBARs with two clock sources. Both clocks originate from a SCSI connector on one of the riser cards; one uses LVTTL signaling, the other uses LVDS.

By implementing a simple resource sharing program on the control Linux module, multiple users can access and run different designs on a single BPU simultaneously. Spatially different users can be allocated to different FPGA chips or different quadrants of the board. Temporally different users are served in a first come first serve fashion by specifying the duration of the FPGA chip reservation. When the reservation expires, other users can reclaim the FPGA. In practice, each BPU can support between 4 to 8 users depending on contention for shared resources such as system main clock, configuration bus, and external I/Os.

## 2.6 PCB Design & Testing

Figure 6 shows the photo of the main processing board—a 26 layer PCB, with a width of 58 cm and depth of 53 cm, which is the largest and the most complex board in the BEE system. Table 1 shows statistics of the PCB.

**Table 1: Main processing board PCB**

| | |
|---|---|
| **Component count** | 3400 |
| **Pin count** | 28611 |
| **Layout area (sq cm)** | 2754.8 |
| **Number of nets** | 8493 |
| **Number of connections** | 19877 |
| **Manhattan distance (km)** | 1.167 |
| **Etch length (km)** | 1.316 |
| **Via count** | 32334 |

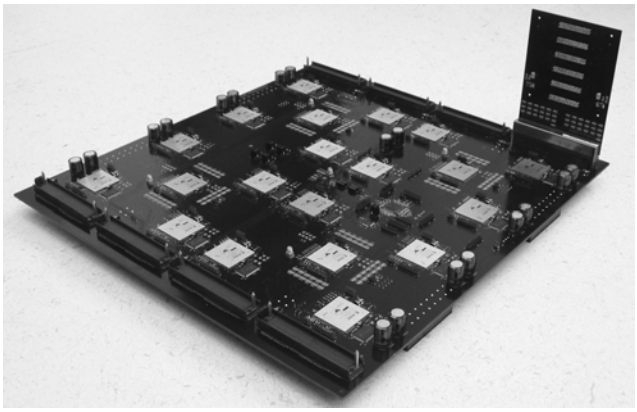

**Figure 6: A main processing board and a riser card**

The complexity of the MPB leads to numerous testing challenges. A rapid diagnostic method is needed to verify connections between the FPGAs and the SCSI connectors through the riser I/O boards. The connectivity diagnostic is separated into two stages. The first stage is to verify connections between the peripheral FPGAs and the external SCSI connectors. The FPGAs are configured to output a distinct sequence of patterns to each of the SCSI connectors through the riser I/O boards; then an external SCSI cable tester is used to display the pattern on LEDs for each of the SCSI connectors. Therefore, all external connections can be verified visually. Second stage is to verify all internal on-board connections between the FPGAs.

## 2.7 Mechanical Issues

Given the large size of the MPB and the worse case power consumption of 400 W, the BPU mechanical chassis has to not only accommodate the MPB and all the power supplies, but also provide active ventilation for the heat dissipation.

Figure 7 is a cross section photo of the BEE chassis. The MPB slides into a slot in the middle of the chassis dividing it into the upper and lower chambers. The lower chamber contains all the power supplies, SBC, and all internal wires. On the front panel, seven high flow fans are used to cool both chambers. Cold air is pulled into the chassis from the fan panel side, and then exhausted from the opposite panel, thus increasing the airflow speed. Detailed thermal modeling of the chassis has been used to determine the maximum allowed power consumption for each FPGA. When only using the chassis fans, the FPGAs can operate safely up to 12 W each; if passive heat sinks are used, then over 20 W power consumption can be tolerated [14]. All FPGAs are continuously monitored by the configuration FPGA for thermal runaway. If any of the FPGA junction temperatures exceeds 80°C, the configuration FPGA automatically shuts off the main power and alerts the user through the separately powered SBC.

**Figure 7: BEE chassis cross section**

# 3. SIGNAL INTEGRITY

As for any large-scale high-speed PCB design, signal integrity is a challenging obstacle in achieving the desired performance within the BEE system. Signal integrity solutions taken in BEE can be classified into the following categories: crosstalk reduction, impedance control, signal reflection reduction, delay matching, and supply voltage bypassing.

The root of the crosstalk problem is the large mount of signal routes on the MPB board. With a given PCB board size, high routing density can cause severe cross talk issues that drastically slow down the overall performance. Thus, the primary method for improving signal integrity is to reduce the routing density, which can be achieved by increasing the PCB area and the total number of PCB layers at the expense of increased manufacturing cost. However, this solution is limited by the PCB manufacturing technology.

The MPB PCB utilizes the largest form factor that was available from the PCB fabrication vendor at the build time, and the number of layers was determined by the routing necessity. Routing density is measured in terms of the space between any adjacent traces, which is kept above 8 Mils between traces on the same layer to reduce crosstalk. In addition, all signals designed to run above 10 MHz are routed on the internal PCB layers as striplines to further reduce inter-layer crosstalk by sandwiching the traces by a power or ground plane. Nevertheless, due the large size of the PCB, long traces, such as between XBARs, requires increased inter-trace spacing to reduce the capacitive coupling.

On the MPB, an impedance of 50 Ohm for single ended and 100 Ohm for differential signals was chosen for approximate matching with signal drivers and other PCBs. Since the external connections use LVDS signaling standard, each differential signal trace pair was routed together with fixed spacing. PCB layer thicknesses and trace widths were chosen to ensure uniform impedance value throughout the board. Similar rules were also used for the riser I/O card PCB design to control impedance matching.

Due to the large PCB size, connections between distant components on the board, such as between XBARs, typically have trace lengths exceeding 30 cm. The 8 mA slow I/O buffer does not exactly match to 50 Ohm impedance, thus signal reflections can degrade signal quality. Therefore, series resistor terminations are used on these long traces to attenuate the reflections. External LVDS connections typically require the termination resistors to be placed within a couple of centimeters from the FPGA chip. As discussed in section 2.4, to increase the reusability of the MPB, external terminations are placed on the riser I/O cards, increasing the distance to approximately 6 cm. Although this approach reduces the maximum LVDS link speed, the external differential links can still operate at 160 MHz speed.

Careful routing of the clock traces is critical for reducing clock skew on a large PCB. Since the main clock originates from the center of the board, if routing to each FPGA followed the shortest route, the trace length would vary over 15 cm between the near and far chips. Therefore, all clock traces are routed manually, matching lengths to within one tenth of a millimeter, and are isolated from other signal traces by using dedicated PCB layers.

Ground bounce is another problem due to high switching activity on multiple high pin-count FPGAs. In extreme cases, the voltage drop could be so significant that the chip can momentarily malfunction and produce errors. Therefore, sufficient bypassing capacitors of various values are necessary to ensure the proper operation of the system. There are over 2400 bypassing capacitors on the MPB, which are divided into four tiers of values: 47 nF, 100 nF, 100 µF and 2200 µF.

# 4. TEST AND MEASUREMENT

After the assembly of all four BPUs, a series of tests were performed to verify the functionality, as well as to determine the performance and capacity of each system. Functionality tests

verify the basic operation of the system, such as system power up with proper voltage levels, programming all FPGAs though JTAG and through SBC Ethernet link, and so on. Performance tests measure the maximum clock speed, internal inter-chip speed, and external link speed. Capacity tests gauge the maximum design size and the throughput of the system. All performance tests are done using designs implemented directly on the FPGAs, rather than through logic analyzer measurements. Therefore, tests can be easily repeated on different BPUs for consistency and convenience.

## 4.1 System Clock Rate

The maximum clock rate of the system is determined through the clock speed test. The programmable system main clock is slowly ramped from 1 MHz, until the test structure indicates an error. This clock rate has been measured to be above 160 MHz for all four BPUs. Since the internal clock rate of the clock speed test design on the FPGA is estimated to be above 180 MHz with Xilinx post placement and routing timing analysis, the failure at 160 MHz is most likely due to the distribution of the main clock on the PCB. Therefore, 160 MHz is determined to be the upper bound of the distributed system clock. However, in practice, the clock rate usually does not exceed 100 MHz because of the design style and goals, and if higher clock rates are desired, on-chip delay-locked loops can be used to double or quadruple the clock frequency on the FPGA.

## 4.2 On-board Internal Connections

The inter-FPGA link speeds are one of the determining factors of the overall system performance. Nevertheless, due to the vast amount of links, the precision measurement of the link speed is usually tedious and time-consuming. Instead of measuring the links speed for every single connection individually, the speed is measured on the whole group of signals between two chips, that is, 48 bits between FPGAs, 96 bits between XBARs, and 36 bits between FPGAs and XBARs. The source FPGA implements a pseudo-random number generator to create for every clock cycle a unique word as well as the checksum bits. The data word and its checksum are transmitted through the link under test to the destination FPGA, where the data word is extracted and its checksum recomputed and compared with the transmitted checksum. If the two checksums are not matched, an external LED indicator is lighted. The inter-FPGA links are registered on both the source and destination FPGA, and the registers are packed into the I/O block to minimize the effect of on-chip routing on the inter-chip link speed. The designs on both the transmitter and receiver side are capable of running at over 158 MHz on-chip as reported by the Xilinx post placement and routing timing analysis. Since each FPGA only directly connects to one LED, several sets of tests are needed to cover all the links on the MPB, each exclusively covering a portion of the links on the board.

As summarized in Table 2, using the 12 mA slow LVTLL I/O buffers, the internal link speed on all BPUs can achieve at least 60 MHz. The relatively large speed variations on the same BPU is largely due to the link trace length differences. In general, the adjacent FPGAs have much higher link speeds than the longer range links, such as between XBARs. Illustrated in Figure 8, the link speed profiles for each BPU look very different from each other; however, the mean and standard deviation only varies less than 5% among different BPUs.

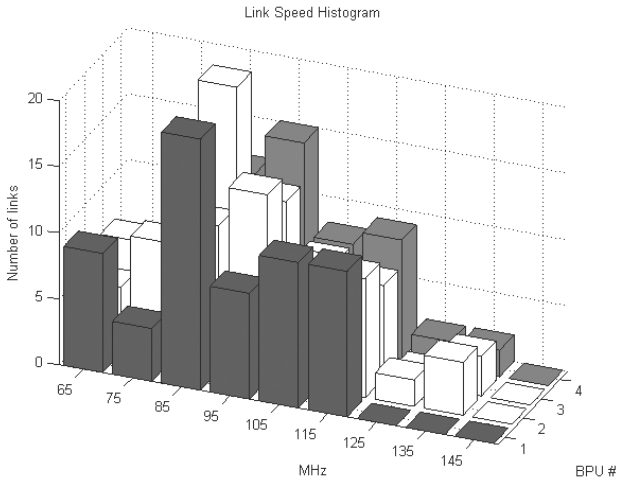


**Figure 8: Internal link speed histogram**

**Table 2: Internal link speed statistics**

| | BPU1 | BPU2 | BPU3 | BPU4 |
|---|---|---|---|---|
| **Mean (MHz)** | 90.2 | 95.1 | 92.7 | 98.1 |
| **STD (MHz)** | 16.3 | 19.1 | 17.2 | 19.1 |
| **Min (MHz)** | 60.3 | 62.5 | 63.5 | 65.5 |
| **Max (MHz)** | 116 | 136 | 137.5 | 145.8 |

The connection speed between a FPGA and a SRAM can be determined by first writing a sequence of random words to the SRAM and then reading them back for comparison. Since the connection is always less then 5 cm and routed similarly throughout the board, the maximum speed of above 110 MHz is achieved consistently throughout all FPGAs on all BPUs.

## 4.3 External Connections

External link speeds can also be determined using similar methods as internal links. In this case, an external cable is used to link between any two SCSI connectors on either the same or different riser cards. The data link loops through the riser I/O card and the external cable, then back through the riser I/O card and connector to either another FPGA or back to the same FPGA. In addition to the LVTTL standard, external links can use LVDS standard. Due to the addition of riser I/O cards and the external 1 meter long cable, when using LVTTL, the maximum external links speed is between 30 MHz to 50 MHz; when using LVDS, the maximum speed is between 160 MHz and 210 MHz.

## 4.4 System Configuration

Another important performance factor is the run-time configuration and read-back speed. As explained in Section 2.5, the configuration bus originates from the SBC, and is then multiplexed by the configuration FPGA to the rest of the 20 FPGAs. The same bus is used at run-time for communication

between the SBC and the FPGAs. Between the configuration FPGA and the rest of the chips, the bus has been verified to operate at 10 MHz. However, due to the Linux kernel overhead and the StrongARM GPIO (General Purpose Input and Output) speed limitations, the link between the SBC and the configuration FPGA can only achieve up to 2 MHz. Furthermore, due to the 10-base-T Ethernet link interface, the connection between the SBC and client workstation is limited to 10 megabits per second, which in practice is typically 7 Mbps. Given the above limitations, the entire process of downloading a bit file to the SBC and then programming it onto the FPGA, takes about 2 seconds, which is substantially faster than using the JTAG interface, which takes about 30 seconds. At run-time, the communication between the client PC and the FPGA on the MPB is limited to about 1 MByte per second.

## 4.5 System Design Capacity

Performance of the system can also be measured with its capacity in terms of the maximum number of operations per second. To verify the capacity, a benchmark design was constructed to fully utilize all resources available on the main processing board. The benchmark design is a 10,240-tap 16-bit fixed-coefficient FIR (Finite Impulse Response) filter. Depicted in Figure 9, each filter tap is implemented with one 16-bit adder, one 12-bit constant multiplier, and one tap-delay register. Multiple taps are cascaded along with additional registers to reduce input fan-out and pipeline inter-chip connections. Each FPGA implements 512 taps, and all 20 FPGAs aggregate to 10240 taps.

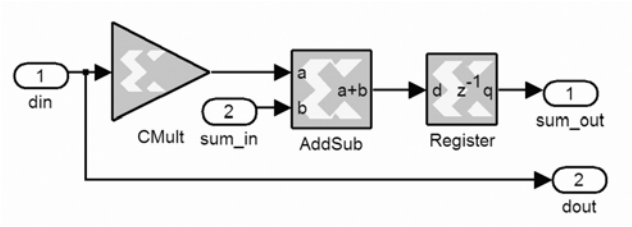


**Figure 9: A single FIR tap implementation**

As summarized in Table 3, this benchmark FIR design illustrates that each BPU is capable of operating at full capacity with a throughput of up to 600 billion operations per second, while emulating an 8 million ASIC gate equivalent design. Since the FIR design does not utilize any of the on-chip dedicated 640 Kbit RAM components, the total capacity upper bound including memory is at least 10 million gates per BPU.

**Table 3: 10,240-tap FIR filter design statistics**

| | |
|---|---|
| **FPGA utilization** | 99% of 19200 slices [In all 20 FPGAs] |
| **Max Clock Rate** | 28.5 MHz |
| **System Operations per Second (addition & multiplication)** | 583.68 billion |
| **ASIC equivalent gates (per FPGA)** | 401000 |
| **ASIC equivalent gates total** | 8 million |
| **Power consumption** | 2.5 W per FPGA |

# 5. DESIGN FLOW

In order to take full advantage of the large design capacity offered by BEE, an automated design flow using high abstraction level description is essential to the practical usefulness of such system. Figure 10 illustrates the current BEE design flow.

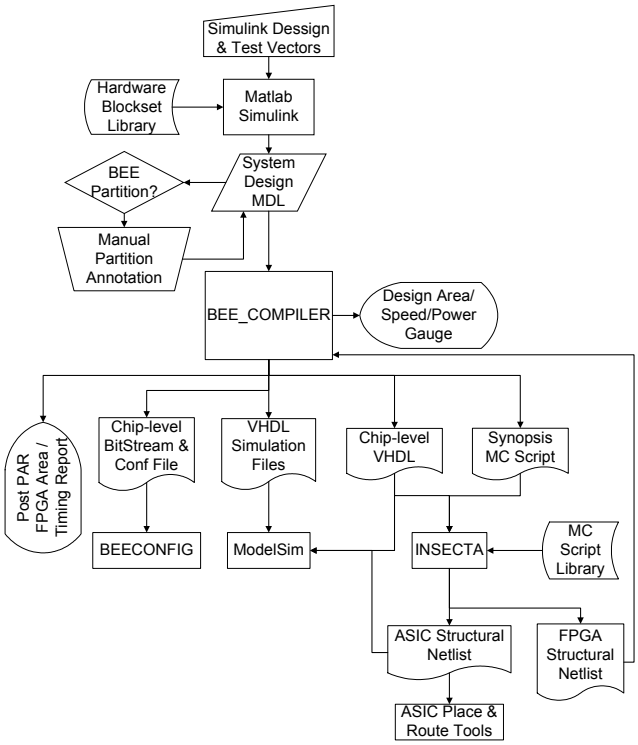


**Figure 10: BEE design flow**

The main goal of the BEE design flow is to enable communication algorithm designers to use system-level design tools to create hardware designs that can be both implemented rapidly on a BEE system and eventually taped out as an ASIC chip. The chosen system design tool is Simulink from Mathworks with the Xilinx System Generator toolbox [15]. On the system-level, the user is provided with a set of hardware-like components that model the cycle-to-cycle behavior of the hardware and simulate using fix-point numbers. The block library includes low-level components, such as an adder and a multiplier, and high-level components, such as FIR filter and FFT block. Hardware specific issues, such as word-length, quantization effects, and latency, can be controlled directly by the system designer. In addition, different system implementation architectures can be conveniently explored. High-level performance estimates, such as the speed, area, and power, allow different architectures to be compared. After constructing the design in Simulink, the BEE compiler is used to generate VHDL structural netlist that is functionally equivalent, bit-true, and cycle-accurate with the original Simulink design. The same VHDL netlist is used for both FPGA and ASIC implementations, but with each block implemented with either FPGA or ASIC optimized cores depending on the target technology. Therefore, the functionality of each library block can be maintained between the FPGA and ASIC implementation,

while the block-level performance is not sacrificed for either technology.

Using the BEE design flow, a design of 400,000 ASIC equivalent gates can be generated and implemented on the BEE hardware in less than an hour. Therefore, rapid design changes are practical and the impact of a change on the performance can be verified directly on the BEE hardware, which shortens the design cycle of a typical communication system. Furthermore, when connected to radio front-ends, the entire system can be tested at the target clock rate, and the system performance metrics, such as bit error rate, can be directly measured and verified.

# 6. CONCLUSION

Fully exploiting the design space available to the contemporary designer of digital signal processing systems is difficult with traditional design automation tools. Particularly, if design time is to be minimized, the use of algorithm level design capture is a necessity. Furthermore, the run-time through the design flow from top to bottom usually directly affects the amount of design space exploration that is practical. The BEE hardware emulator allows rapid prototyping for direct-mapped communication or DSP designs. Running low-power applications in real-time facilitates the verification of complete systems instead of just verifying the functionality of sub-systems, one at a time. This provides more ASIC implementation choices to be explored without sacrificing time-to-market.

A component library targeted for signal processing is used for BEE designs to accelerate the design process and capture application domain specific information. By constraining the application domain to low-power signal processing, reasonable assumptions can be made about the required optimizations. Moreover, the components guarantee that the design can be implemented on both FPGA and ASIC technologies with identical functionality.

The BEE hardware architecture is tailored to accommodate a class of wireless communication applications. The sub-designs are assumed to connect primarily locally, thus emphasizing the fast links between the FPGAs. Similarly, future radio systems are thought to require increasingly complex digital computing to replace some of the analog signal processing. This contributed to the emphasis on the I/O channel capacity currently implemented on BEE.

Currently, four BEE systems have been constructed, and several radio designs are under development. For example, a 2.4 GHz 1 Mbps narrow band radio has been demonstrated on BEE. In addition, an ultra-wide band radio front-end has been developed. The operation speed and capacity of BEE has exceeded the original design goal of 100 to 200 giga-operations per second at 50 MHz. Also, the system has been used for more general purpose computing applications, such as high-level communication network simulation, wireless channel modeling, and image compression. The real-time emulation capability along with the easy-to-use fully integrated design flow make the BEE system a indispensable tool in the next generation communication system design.

The future work in this project includes the construction of larger communication environments based on multiple BPUs and radio front-ends. These environments allow system-level issues to be examined on top of the configurable wireless communication network.

# 7. REFERENCES


[1] M. Courtoy, “Rapid system prototyping for real-time design validation,” Proc. Ninth International Workshop on Rapid System Prototyping, pp. 108-112, 1998.

[2] A. Agarwal, J. Babb, and R. Tessier, “Virtual wires: overcoming pin limitations in FPGA-based logic emulators,” Proc. IEEE Workshop on FPGAs for Custom Computing Machines, pp. 142-151, Apr. 1993.

[3] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, Low Power CMOS Digital Design, Kluwer, 1995.

[4] www.mathworks.com

[5] S. Hauch, G. Borriello, and C. Ebeling, “Mesh Routing Topologies For Multi-FPGA Systems,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, Vol. 6 No. 3, pp. 400-408, Sept. 1998.

[6] M. Butts, J. Batcheller, and J. Varghese, “An Efficient Logic Emulation System,” Proc. 1992 IEEE Int’l Conf. Computer Design: VLSI in Computers and Processors, pp. 138-141, Oct 1992.

[7] M.A.S. Khalid and J. Rose, “A Novel and Efficient Routing Architecture for Multi-FPGA Systems,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, Vol.8, No.1, pp. 30-39, Feb. 2000.

[8] M.A.S. Khalid and J. Rose, “Experimental Evaluation of Mesh and Partial Crossbar Routing Architectures for Multi-FPGA Systems,” Proc. 6th IFIP Int’l Workshop on Logic and Architecture Synthesis, pp. 45-54, Dec. 1997.

[9] C. Kim and H. Shin, “A Performance-Driven Logic Emulation System: FPGA Network Design and Performance-Driven Partitioning,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol.15, No.5, pp. 560-568, May 1996.

[10] M.A.S. Khalid and J. Rose, “A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems,” ACM/SIGDA Int’l Symp. Field Programmable Gate Arrays, pp. 45-54, Feb 1998.

[11] J. Brunetti and B. V. Herzen, “Virtex-E LVDS Drivers & Receivers: Interface Guidelines,” Xilinx Application Note 232 (v1.0), Oct. 1999.

[12] Mark Ng and Mike Peattie, “Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode,” Xilinx Application Note 502 (v1.2), June 2002.

[13] “Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary Scan,” Xilinx Application Note 139 (v1.4), Apr. 2002.

[14] “Packages and Thermal Characteristics,” Xilinx Application Note (v2.1), Feb. 2000.

[15] www.xilinx.com