System-level Prototyping with HyperTransport

Myles Watson and Kelly Flanagan
Computer Science Department
Brigham Young University
Provo, Utah, USA
myles@byu.edu kelly@cs.byu.edu

***Abstract*— The complexity of computer systems continues to increase. Emulation of proposed subsystems is one way to manage this growing complexity when evaluating the performance of proposed architectures. HyperTransport allows researchers to connect directly to microprocessors with FPGAs. This enables the emulation of novel memory hierarchies, non-volatile memory designs, coprocessors, and other architectural changes, combined with an existing system.**

Keywords-HyperTransport; FPGA; prototype; emulation;

I. INTRODUCTION

In accordance with Moore's Law, the number of transistors available to chip designers has continued to double every 18 months. For many years, this transistor scaling also enabled increasing central processing unit (CPU) frequencies. Although CPU frequencies and performance increased rapidly, memory and I/O performance increased much more slowly. This disparity increased the importance of I/O and memory performance in computer systems design [1].

In the last few years, power consumption and cooling have caused CPU manufacturers to shift the focus from frequency scaling to scaling the number of processor cores per die [2]. This has exacerbated the pressure on, and the importance of, the memory and I/O subsystems [3].

The increase in importance of memory and I/O subsystems increases the need for understanding system-level design changes, and their impact on performance. Unfortunately, system-level simulation is error prone and costly. One alternative is to emulate part of the system to be studied using field-programmable gate arrays (FPGAs). Connecting the FPGAs to commercial CPUs enables the study of a portion of the I/O subsystem or memory hierarchy, while eliminating the need to faithfully model the CPUs and their internal components.

Designing and implementing an emulation system from scratch would be a costly endeavor, however in-socket accelerators are commercially available at a much lower cost [4]. In-socket accelerators are FPGA boards designed to fit into a CPU socket, and are marketed as flexible application accelerators. They provide low-latency and low-power computational resources for applications such as bioinformatics, data-mining, real-time financial analysis, and oil and gas exploration.

This work describes how an XtremeData XD1000 FPGA board in an AMD Opteron socket can serve as part of a flexible emulation platform. Since the XD1000 tightly couples an Altera Stratix II FPGA with the CPU and other system resources, such as the DRAM sockets on the motherboard, this platform is useful for exploring the design of I/O subsystems and memory hierarchies. Two emulation platforms incorporating the XD1000 are described, each of which is useful for emulating different system designs. Both of these platforms have been implemented, and preliminary performance results in terms of latency and bandwidth for reads and writes are presented for one of the systems.

The remainder of the paper is divided into sections. Section II presents the design of two emulation platforms using the XD1000, along with some of the implementation concerns. Section III describes three target application areas. Section IV presents preliminary performance measurements and discusses the importance of relative performance as an analysis tool. Section V discusses related work. Section VI is the conclusion.

II. SYSTEM DESIGN

An important characteristic of an emulation system is the connection point to the system, which determines the latency and bandwidth of accesses to the emulated device. Two possible locations are a peripheral bus (e.g., PCIe) and the system bus (e.g., HyperTransport or QuickPath Interconnect).

Connecting the emulation platform to a peripheral bus is a flexible and relatively low-cost way to emulate I/O devices and interfaces. Often, an application-specific integrated circuit (ASIC) can be used to connect to the bus, allowing the designer to use the FPGA entirely for the emulated device.

Using an FPGA to connect directly to the processor via the system bus allows lower-latency access to the device. In general, each bus or device through which memory accesses must pass increases the access latency. The option of using coherent (cache-coherent) memory is another benefit of connecting to the system bus.

Coherent memory provides more flexibility in the memory organizations that can be studied, since it can be cached and paged by the microprocessor. From the perspective of the operating system (OS) and applications, this makes it indistinguishable from DRAM connected to a remote processor. Coherent memory allows the study of caching and buffering schemes.

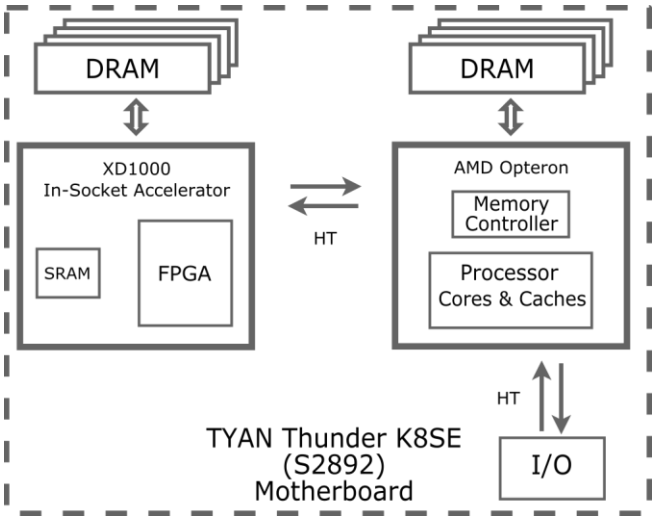

Figure 1. XD1000 in a cave configuration.

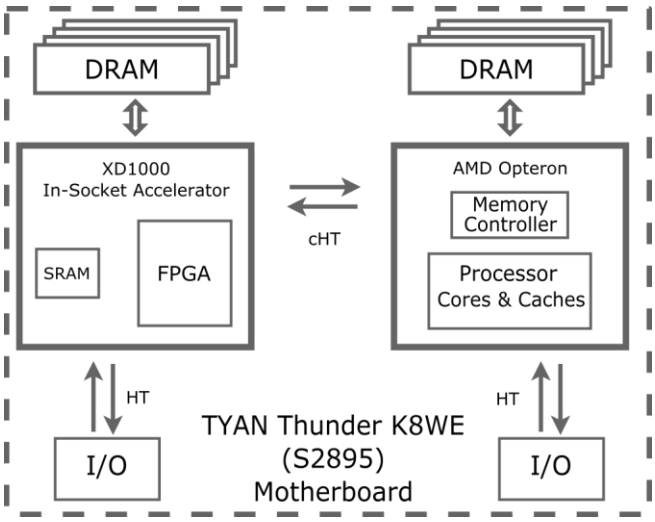

Figure 2. XD1000 in an I/O host configuration.

A. *Coherent HyperTransport*

The coherent HyperTransport (cHT) specification is a superset of the HyperTransport (HT) specification. The HT specification is open, but the cHT specification is only available under NDA with AMD [5,6]. The University of Heidelberg's Center Of Excellence for HyperTransport (CoEHT) has developed HT and cHT cores which can be deployed in FPGAs to connect to AMD Opteron processors through processor-socket interposers (e.g, the XtremeData XD1000) or HyperTransport Extension (HTX) boards (e.g., the CoEHT HTX board [7]).

B. *Architectural Variations*

Opterons and XD1000 modules have three HT links, allowing some flexibility in the configuration of a system. The cHT core adds another option to each configuration. Figures 1 and 2 show two of the configurations available using one or two links. In each case, the link between the Opteron and the XD1000 can be HT or cHT, yielding two additional configurations.

In this work, the XD1000 module is deployed in two Tyan motherboards, the Thunder K8WE (S2895) and Thunder K8SE (S2892). These motherboards were chosen because they are very similar and are supported by coreboot (open-source firmware) [8]. Using coreboot with BIOS emulation routines allows unmodified OSs to be booted, which eases application and driver development [9]. The S2895 has two chipsets, which allows the XD1000 to function as a coherent I/O host. Both configurations have four 1GB DDR DIMMs directly attached to the XD1000.

If main memory is part of the emulated system, cHT is chosen as the connection between the XD1000 and the Opteron. The DRAM connected to the Opteron can then be removed from the system, requiring all memory accesses to be serviced by the XD1000 and the DRAM connected to it. If more than 4 GB of emulated storage is required, the I/O host can connect to I/O devices (PCIe) on the motherboard through a second HT link.

In the configurations shown in Figure 2, where there are multiple HT links, care must be taken to avoid deadlock. HT specifies that no transactions should depend on the completion of other transactions, and transactions should not create new transactions. These guarantees are easily broken by a system which changes the integration level of components, so any new packets must be isolated from the rest of the system. The method of choice is to separate the traffic controlling the I/O devices from the read and write requests from the Opteron. The HT specification requires all packets from devices to traverse the complete chain to the host. This allows the packets to be routed based on their address by the I/O host. In this work, packets are filtered based on their source and destination to make sure that traffic that is part of the emulated system does not reach the CPU.

The XD1000 HT links can run at 200 or 400 MHz using the serializer/deserializer (SERDES) hardware in the FPGA, or at 200 MHz when implemented with DDR registers. When the XD1000 is used as an I/O host on the S2895, at least one of the links is limited to 200 MHz. This is due to a combination of the HT link connecting the XD1000 to the chipset, and limited FPGA resources. Since the links are 16 bits wide and HT is DDR, this provides 800 MB/s of theoretical peak bandwidth in each direction.

C. *Firmware Modifications*

In order to use the XD1000 to emulate multiple system configurations, the firmware which initializes the system must be modified. The modifications can be grouped into three types: XD1000 initialization, address space allocation, and resource reporting. The modifications are more extensive for the I/O host than for the cave.

When used as a cave, the XD1000 initialization is minimal. It consists of an extra hard reset if the HT link is not active. This is necessary to allow the clock generation circuitry of the FPGA sufficient time to stabilize. The resource allocation process must be circumvented for the 4 GB of DRAM, which is allocated above main memory. The Advanced Configuration and Power Interface (ACPI) tables must then be modified so that the XD1000's bus is visible to the OS.

When the XD1000 is an I/O host, it appears to software to be an Opteron processor. It must be programmed with the correct routing values and included in the routing table so that memory accesses reach it correctly. Since the DRAM controller is implemented in the FPGA fabric, the DRAM initialization code needs to be skipped as well. The size of the address space occupied by the emulated storage must be specified, and some ACPI tables must be modified in order for the memory to appear to be attached to node 0. Since there are no processor cores, the code which initializes the Opteron processor cores must be skipped so that the cores appear to be disabled. As a final step, the devices connected to the HT link of the I/O controller, which will be part of the emulated system, must be initialized and hidden from the OS.

D. *Bandwidth and Write Buffering*

The basic unit of transfer in the HyperTransport protocol is the thirty-two bit (four-byte) word. The most efficient transfers (with the lowest overhead) are transfers of 64 bytes. Transfer sizes depend on the Opteron's memory type and page attributes. When the address space is write-back, reads transfer 64 bytes at a time, but writes are performed according to the data size of the store instruction. When the address space is write-combining, the opposite is true.

In order to maximize bandwidth in both directions, the XD1000 example application makes use of DMA engines in the FPGA to transfer data to and from the host memory. This works well when the emulated device is accessed only through a driver, which can set up the transfers. When any size of transfer may be used, this asymmetric performance must be taken into account.

Even with 64-byte transfers, write buffering must be used, since the DRAM controller has a width of 128 bytes. This means that 128 bytes must be read from DRAM before 64 bytes can be written. Much of the complexity involved in creating an application with HyperTransport is a product of the different widths. The 32-bit HT bus protocol is converted by the core to 64-bit data for processing on the FPGA, since FPGAs make better use of wide widths than high clock rates. These data words must be assembled for the DRAM controller. In order to manage this complexity, all writes to RAM are handled by the write buffer, as are any reads that are smaller than 64-bytes.

III. APPLICATIONS

Many areas of system design can be explored using emulation. Three of the areas that seem most promising are: adding non-volatile memory (NVRAM), adding an application-specific coprocessor (or changing the way one is integrated with the system), and changing the memory hierarchy.

A. *Non-volatile Memories*

Nonvolatile memory technology is advancing. Flash memory is being used as a disk replacement in performance-critical applications. Other technologies, such as phase-change memory (PCM) and spin-torque transfer memory (STTM), are also being developed. Their densities are increasing, and they may be included in future computing systems.

These technologies differ from the DRAM in several important ways, which will influence their integration into computer systems. The two most obvious differences are asymmetric access times for writes and reads, and the need for wear leveling. Both of these factors will influence the design of memory controllers and the resulting performance of applications.

Building prototype systems is prohibitively expensive for exploring the design space, and cannot be done before devices are produced. In order to explore the design space, tools must be developed that will allow accurate performance comparisons for different organizations, block sizes, and wear-leveling and buffering algorithms.

The emulation system of Figure 1 can be used to explore design choices and the interactions of applications with up to 4 GB of NVRAM connected to the system. Programmable delays can be added to the DRAM controller [10] and/or the write buffer in order to more accurately model the access latencies of each technology.

B. *Coprocessors*

One way to increase the time and power efficiency of computation is to use application-specific processors. Many applications have abundant available parallelism. This parallelism can be efficiently exploited by architectures combining many simple, low-power processing elements. General-purpose computing on graphics processing units (GPGPU) is an example of this. The connections between the GPU, the CPU, and memory affect the performance of the application. This could affect how the work is divided among processing units.

The same architectural questions can be explored for general graphics processing. AMD's Fusion architecture more tightly couples the GPU and the CPU in order to achieve higher performance, lower power consumption, or both. An emulated system can be used to explore the design space and performance benefits of such a system before it is built.

C. Memory Hierarchies

The increasing gap between main memory and CPU speed has increased the importance of the memory hierarchy in system performance. Much of the area on recent CPU dies is dedicated to caches. There is a large design space to be explored, and its complexity is increasing with the number of processor cores. Structures such as coherence directories are good candidates for emulation, since they can be implemented with the RAM resources of the FPGA.

One extension to the memory hierarchy which can be explored using emulation is a hardware single-level store, which moves control of swapping pages of memory from the OS into hardware. Swapping is a feature of virtual memory when the virtual memory space is larger than physical RAM. Memory pages are swapped when pages of data are transferred to and from the secondary store to maintain the illusion of large memory space. If a page is chosen for replacement that will be used again soon, its next access will cause another swap. Since secondary storage is much slower than RAM, minimizing swapping is essential to performance. Some related features, such as file caching, can also be controlled by the same hardware, since the files reside in the secondary store and get moved to RAM for faster access.

Hardware paging support is interesting because there is limited information available to the OS about page usage. Usage bits are only updated during page table walks, which occur on TLB misses. In order for an OS to collect more usage information, it must invalidate TLB entries to cause misses, which is expensive. With more information, paging algorithms make better replacement decisions, increasing performance [11]. A hardware paging implementation would be aware of all memory accesses that miss the last level of cache, and therefore have more information on which to base page replacement decisions.

Moving paging support out of the OS is not a new idea. The IBM AS/400 and its predecessor, the IBM System/38, implement paging in virtual machines. This simplifies software development, since from the perspective of the OS and applications, memory is flat and uniform [12]. A virtual machine implementation of paging suffers the same performance penalties as other software implementations, due to limited usage information,.

IV. PERFORMANCE

Performance measurements and comparisons are two of the most compelling reasons to emulate modifications to computer systems. Although the most straightforward way to measure system performance is by measuring wall clock time, it is not the most helpful metric for comparing emulated systems. Although the FPGAs used for emulation continue to improve in speed, they are not as fast as a final implementation.

TABLE I. READ AND WRITE BANDWIDTH MEASUREMENTS.

Transaction Type	Bandwidth (MB/s)
32-bit writes	60
64-bit writes	90
64-byte writes (write-combining)	120
32-bit reads	5.5
32-bit reads (two threads)	11
64-byte reads (cacheable 32-bit)	50
64-byte reads (two threads)	92

A. Preliminary Performance Measurements

In order to understand the performance characteristics of a system, simple latency and bandwidth measurements are taken. The system shown in Figure 1 is booted into Linux, and a modified device driver based on the example XD1000 driver is loaded. A simple application is then run, which calls mmap to obtain a pointer to the 4GB of memory on the XD1000. Once the program has a pointer, it is straightforward to write timing loops which measure the average latency and bandwidth of memory accesses. The measured latencies can be verified using Altera SignalTap to view the HT requests.

The latency for each read or write targeting the DRAM is around 850 ns, with the write buffer implemented, but no workload-specific optimizations. This yields varying bandwidths depending on the transaction types and sizes, as shown in Table 1. Because the write buffer is organized as a cache, each write to a new line causes a line fill from the DRAM, and possibly a write back for dirty data. An obvious performance optimization is to bypass the write buffer when multiple consecutive writes are received, and write a full 128 bytes directly to DRAM. Avoiding the write buffer in this way would substantially increase the write bandwidth. Note that read bandwidth is significantly lower than write bandwidth because each read must complete before software can issue another read; writes have no such restriction. Running two threads nearly doubles the read bandwidth because the two processor cores can issue reads in parallel, but it has no effect on write bandwidth.

B. Relative Performance Comparisons

Using absolute performance numbers with emulated architectures can be misleading. The solution is to use relative performance comparisons. Some of the factors that make relative performance comparisons more appropriate than using absolute performance include: the lower frequency of an FPGA implementation of HyperTransport, the fact that the emulated prototype may not be fully optimized, and even restrictions with the NDA in publishing performance numbers for the coherent core.

In order to compare the performance of multiple non-volatile memory technologies and their controllers, the path

for each access should be equivalent. This means that a comparison between the delayed RAM on the XD1000 and the RAM attached to the host Opteron would be much less informative than a comparison between two delay settings on the XD1000.

For the case of an emulated single-level store, the only DRAM in the system is attached to the XD1000, and all requests must traverse the same path. The difference being measured can then be attributed to the difference in the paging algorithm, and the information available to it. The latency of a memory access in this scenario is the sum of the latencies due to: the HT link, the write buffer access, the DRAM access, and in the case of a miss, a page transfer from the backing store to DRAM.

When making the baseline measurements, the Opteron is initialized to access 4 GB of RAM with the XD1000 as the only memory controller. Memory needs beyond 4 GB must be supplied by OS-controlled paging to the secondary storage. The baseline is then compared to the same configuration, but hardware paging is enabled and the XD1000 is initialized as a memory controller with up to 1 TB of storage addressable as RAM. The 1 TB limit is a hard limit dictated by the 40 physical address bits available to the processors. Newer Opterons have 48 physical address bits, expanding their addressing capabilities to 256 TB.

V. Related Work

There are many system-level simulators, but there are relatively few systems which add emulation to an existing system using FPGAs. In this section, a case is presented for using emulation in place of full-system simulation. This analysis is followed by a discussion of three related emulation systems, and two FPGA prototype systems that use HT to enable low-latency cluster interconnects.

A. Emulation vs. Simulation

Several factors make system-level simulation time consuming, expensive, and error-prone. These include the asynchronous interactions among multiple devices, the closed nature of many CPUs, the complexity of these CPUs and their interconnects, and the increasing sizes of caching structures and translation look-aside buffers (TLBs).

Since modern computer systems incorporate many diverse components, modeling their interactions faithfully can be difficult. Computer systems include devices ranging from PCI Express (PCIe) graphics cards to hard drives to serial ports, with widely varying performance characteristics and latencies. Modeling the system at a sufficient level of detail to accurately reflect system performance is a challenge.

Modern CPUs have complex performance characteristics, which can be difficult to model [13]. Although some high-level details of CPU architectures are available, many of the details needed for accurately simulating their performance are not. Even if all the design parameters are available, the complexity of faithful modeling slows simulations significantly, and it is difficult to assure the correctness of the final model. This also applies to the interconnections among CPU cores and the connections to other subsystems. Multi-core architectures exacerbate this problem.

As storage structures such as caches and TLBs increase in size, the amount of simulated run time needed in order to characterize their performance increases. Measuring the benefit of another level of cache, for example, will require the benchmark to generate many misses in the previous levels.

Emulation is a promising way to reduce the complexity involved in understanding the effects on performance of modifications to an existing system. FPGAs combine programmable logic and I/O interfaces, and some contain implementations of simple microprocessors. This makes them suited to implement a wide variety of functions for experimentation. Their performance is limited in terms of maximum clock frequency, but many times that can be mitigated by the high degree of fine-grained parallelism available in them.

Emulated subsystems implemented in an FPGA run fast enough to allow multiple benchmark runs. These multiple runs add statistical significance to performance measurements of the emulated systems and minimize the effect of performance variability of the other system components.

B. Emulation Systems

Three related FPGA emulation systems are Flexible Architecture Research Machine (FARM) [14], Research Accelerator for Multiple Processors (RAMP) [13, 15], and High-performance Advanced Storage Technology Emulator (HASTE) [10].

FARM is similar to this work, in that it modifies and repurposes an existing FPGA and Opteron system in order to explore system architecture. FARM differs from using an in-socket accelerator because the original system is much more expensive, and the FPGAs are not directly connected to the DDR or chipset on the motherboard.

RAMP is a collaborative effort by a number of researchers to enable comparable architectural research and bring down the costs associated with FPGA emulation, specifically for many simple cores and their interconnects. In order to achieve this goal, RAMP specifies FPGA boards, and encourages the sharing and reuse of design components for the FPGA designs. RAMP focuses on the challenges of multi-core architectures and the software which runs on them.

HASTE is a system constructed by UCSD to evaluate NVRAM technologies in supercomputing applications. HASTE connects DRAM with an FPGA controller on a PCIe card, and is compared with the system DRAM and solid-state disks to explore the performance of storage devices built from emerging NVRAM technologies.

C. *Low-Latency Cluster Interconnects*

Two systems which use FPGAs with HT to prototype low-latency cluster interconnects are the Virtualized Engine for Low Overhead (VELO) [16], and the Hyper Parallel Processing (HPP) architecture [17].

VELO is an implementation of a network engine using an HTX card. The resulting network exhibits latencies of just over 1 µs, including routing.

HPP connects multiple motherboards with an HT backplane and a switch implemented with an FPGA. The HPP prototype demonstrates low-latency, high-bandwidth connections between motherboards in a prototype high-performance, low-cost cluster.

Both VELO and HPP are specifically designed to prototype connections between systems, whereas systems using in-socket emulators are better suited for emulating and prototyping modifications to parts of a single system.

VI. CONCLUSION

This work demonstrated how HT and FPGAs can be used in commodity systems to emulate and evaluate the performance of proposed system modifications. The ability of the XD1000 to connect directly to the motherboard HT links was shown to allow the exploration of many system configurations. Two of these configurations were presented, along with preliminary performance results from one of them. These emulation systems were presented as a viable way to evaluate new technologies such as NVRAM, and the many ways that they can be incorporated into computer systems.

ACKNOWLEDGMENTS

Thanks to Heiner Litz, Maya Gokhale, and the anonymous reviewers for their comments and helpful suggestions.

REFERENCES

[1] W. A. Wulf and S. A. McKee. 1995. “Hitting the memory wall: implications of the obvious,” SIGARCH Comput. Archit. News 23, 1 (March 1995), 20-24.

[2] K. Asanović, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick. “A view of the parallel computing landscape,” Commun. ACM 52, 10 (October 2009), 56-67.

[3] P. Conway and B. Hughes. “The AMD Opteron northbridge architecture,” IEEE Micro 27, 2 (March 2007), 10-21.

[4] XtremeData web site, http://www.xtremedata.com/.

[5] HyperTransport Center of Excellence web site, http://ra.ziti.uni-heidelberg.de/coeht/

[6] HyperTransport Consortium web site, http://www.hypertransport.org/.

[7] H. Fröning, M. Nüssle, D. Slogsnat, H. Litz, U. Brüning, “The HTX-board: a rapid prototyping station,” Proc. Of 3rd annual FPGAworld Conference, Nov. 16, 2006, Stockholm, Sweden.

[8] Coreboot web site, http://www.coreboot.org/.

[9] A. Agnew, A. Sulmicki, R. Minnich, W. A. Arbaugh: “Flexibility in ROM: a stackable open source BIOS,” USENIX Annual Technical Conference, FREENIX Track 2003: 115-124.

[10] A. M. Caulfield, J. Coburn, T. I. Mollov, A. De, A. Akel, J. He, A. Jagatheesan, R. K. Gupta, A. Snavely, and S. Swanson, “Understanding the impact of emerging non-volatile memories on high-performance, IO-intensive computing,” SC'10: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, New Orleans, Louisiana, Nov. 2010.

[11] P. Zhou, V. Pandey, J. Sundaresan, A. Raghuraman, Y. Zhou, and S. Kumar. “Dynamic tracking of page miss ratio curve for memory management,” In Proceedings of the 11th international conference on Architectural support for programming languages and operating systems (ASPLOS-XI). ACM, New York, NY, USA, 177-188.

[12] F. G. Soltis, Inside the AS/400, second ed. Duke Communications, Loveland, CO, 1997.

[13] J. Wawrzynek, D. Patterson, M. Oskin, S. Lu, C. Kozyrakis, J. C. Hoe, D. Chiou, K. Asanović. “RAMP: Research Accelerator for Multiple Processors,” IEEE Micro, 27(2):46–57, 2007.

[14] T. Oguntebi, S. Hong, J. Casper, N. Bronson, C. Kozyrakis, K. Olukotun, “FARM: a prototyping environment for tightly-coupled, heterogeneous architectures,” FCCM '10: The 18th Annual International IEEE Symposium on Field-Programmable Custom Computing Machines, May 2010.

[15] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanović, “RAMP gold: an FPGA-based architecture simulator for multiprocessors,” In Proceedings of the 47th Design Automation Conference (DAC '10). ACM, New York, NY, USA, 463-468.

[16] M. Nüssle, B. Geib, H. Fröning, and U. Brüning, “An FPGA-based custom high performance interconnection network,” In Proceedings of the 2009 International Conference on Reconfigurable Computing and FPGAs (RECONFIG '09). IEEE Computer Society, Washington, DC, USA, 113-118.

[17] X. Yang, F. Chen, H. Cheng, N. Sun, “A HyperTransport-based personal parallel computer,” Cluster Computing, 2008 IEEE International Conference on, pp.126-132, Sept. 2008.