RAPTOR – A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing

Mario PORRMANN, Jens HAGEMEYER, Johannes ROMOTH, Manuel STRUGHOLTZ, and Christopher POHL [1]
Heinz Nixdorf Institute, University of Paderborn, Germany

Abstract. A number of FPGA-based rapid prototyping systems for ASIC emulation and hardware acceleration have been developed in recent years. In this paper we present a prototyping system with distinct flexibility and scalability. The designs will be described from an architectural view and measurements of the communication infrastructure will be presented. Additionally, the properties of the system will be shown using examples, that can be scaled from a single-FPGA-implementation to a multi-FPGA, cluster based implementation.

Introduction

In the process of developing microelectronic systems, a fast and reliable methodology for the realization of new architectural concepts is of vital importance. Prototypical implementations help to convert new ideas into products quickly and efficiently. Furthermore, they allow for the development of hardware and software for a given application in parallel, thus shortening time to market. FPGA-based hardware emulation can be used for functional verification of new MPSoC architectures as well as for HW/SW co-verification and for design-space exploration [1,2,3]. The rapid prototyping systems of the RAPTOR family that have been developed in the System and Circuit Technology group in Paderborn during the last ten years, provide the user with a complete hardware and software infrastructure for ASIC and MPSoC prototyping. A distinctive feature of the RAPTOR systems is that the platform can be easily scaled from the emulation of small embedded systems to the emulation of large MPSoCs with hundreds of processors.

1. RAPTOR-X64 – A Platform for Rapid Prototyping of Embedded Systems

The rapid prototyping system RAPTOR-X64, successor of RAPTOR2000 [4], integrates all key components to realize circuit and system designs with a complexity of up to 200 million transistors. Along with rapid prototyping, the system can be used to accelerate

[1] This work was partly supported by the Collaborative Research Center 614 – Self-Optimizing Concepts and Structures in Mechanical Engineering – University of Paderborn.

computationally intensive applications and to perform partial dynamic reconfiguration of Xilinx FPGAs.

RAPTOR-X64 is designed as a modular rapid-prototyping system: the base system offers communication and management facilities, which are used by a variety of extension modules, realizing application-specific functionality. For hardware emulation, FPGA modules equipped with the latest Xilinx FPGAs and dedicated memory are used. Prototyping of complete SoCs is enabled by various additional modules providing, e.g., communication interfaces (Ethernet, USB, FireWire, etc.) as well as analog and digital I/Os. The local bus and the broadcast bus, both embedded in the baseboard architecture, add up to a powerful communication infrastructure that guarantees high speed communication with the host system and between individual modules, as depicted in figure 1. Furthermore, direct links between neighboring modules can be used to exchange data with a bandwidth of more than 20 GBit/s.

For communication with the host system, either a PCI-X interface or an integrated USB-2.0 interface can be used. Both interfaces are directly connected to the local bus, thus creating both a closely coupled, high speed, PCI-X based communication, or a loosely coupled, USB based communication. As configuration and application data can either be supplied directly from the host system or stored on a compact flash card, standalone operation is also supported. Therefore, the system is especially suitable for infield evaluation and test of embedded applications. In addition to these features, RAPTOR-X64 offers several diagnostic functions: besides monitoring of the digital system environment (e.g., status of the communication system), relevant environmental information like voltages and temperatures are recorded. All system clocks are fine-grain adjustable over the whole working range, allowing for running hardware applications at ideal speed.

The latest FPGA module that is currently available for RAPTOR-X64 (called DB-V4) hosts a Xilinx Virtex-4 FX100 FPGA and 4 GByte DDR2 RAM (see figure 1). The FPGAs include two embedded PowerPC processors and 20 serial high-speed transceivers, each capable of transceiving 6.5 GBit/s in full duplex. Utilizing these transceivers, four copper-based data links with a throughput of up to 32.5 GBit/s each are realized on the DB-V4 module. By adapting the cabling between the modules, the communication topology can be changed without affecting the communication via the RAPTOR base system. Serial data transmission at data rates of up to 6.5 GBit/s necessitates techniques to maintain signal integrity between the FPGAs. Utilizing all integrated signal integrity features of the FPGA and providing a sophisticated PCB environment

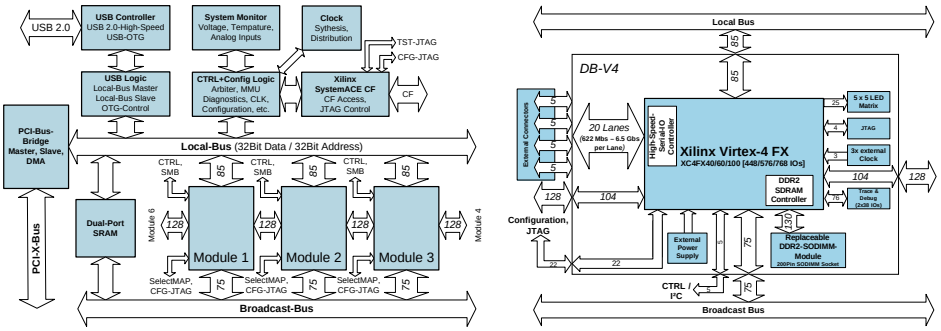

Figure 1. Architecture of the RAPTOR-X64 prototyping system and the FPGA based module DB-V4

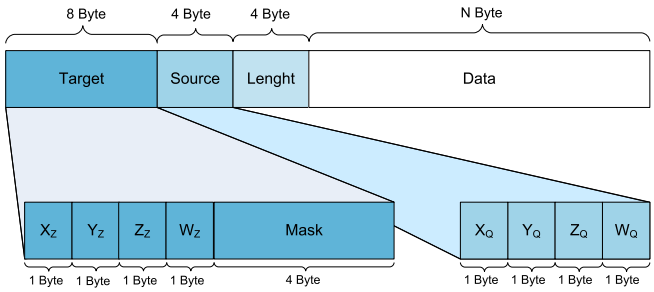

Figure 2. Data-packet format used for communication between the FPGAs.

together with high-end cables and connectors, wire-based communication between the RAPTOR boards is possible up to a distance of two meters at full data-rate.

For easy usage of the high-speed, low-latency communication between the FPGAs, a five-port Rocket-IO switch has been developed and is available as an IP-core. The switch supports non-blocking cut-through operation between Rocket-IO ports, and handles all details of the Rocket-IO based communication such as initialization, channel-bonding, flow-control, error detection, etc. This functionality is built on top of the Xilinx Aurora-protocol. Aurora inherits low-level tasks like bonding multiple Rocket-IOs to a single lane, clock and data recovery, and error detection via 8B/10B encoding mechanisms. It implements an easy to use data and control interface that can be interfaced by the higher layers of the user-protocol.

A generic on-chip interface allows for easy integration of the switch into every design. Futhermore, for integration in a processor-centric design, the switch utilizes a scatter-gather-DMA (SG-DMA) based interface. This implementation enables sending/receiving of packets directly from the processor's main memory (e.g., DDR2-SDRAM), where the processor is only involved in the initialization of the process. The switch implementation is available as an IP-core integrated into the Xilinx EDK. The software libraries allow to access the serial high-speed links similar to other packet-based protocols, e.g., Ethernet, and can easily be integrated into embedded operating systems like Linux.

A custom communication protocol has been developed, which is especially suitable for MPSoC emulation. The protocol is implemented in hardware in the switching fabric, and allows routing of data in up to four dimensions. Switching together with data transmission or reception is executed in parallel to achieve maximum throughput and minimal latency. Besides efficient routing algorithms, a custom data-packet format has been defined for the new protocol, including information about target, source and length of the data packets (see figure 2). Target and source fields in the packet header represent the coordinates of the FPGA-module in the global structure, thus enabling fast routing mechanisms for the desired infrastructure. A mask-field, which extends the target information can be used for addressing multiple FPGA-boards, therefore enabling the use of multicasts or broadcasts.

2. RAPTOR-XPress – A Highly Scalable Rapid Prototyping Platform

RAPTOR-X64 together with the proposed Virtex-4 modules can be used to set up FPGA-based systems with dozens of high-end FPGAs and a high-speed communication infrastructure between the FPGAs. With an increasing number of FPGAs however, the requirements on monitoring and debugging steadily increase, requiring high-bandwidth communication between each individual FPGA and the host computers. Furthermore, in the RAPTOR-X64-based environment, the topology between the FPGAs can only be changed by adapting the cabling. Therefore, the next generation of RAPTOR systems has been developed – RAPTOR-XPress, which will facilitate FPGA-based cluster computing with hundreds of FPGAs connected by a very flexible communication infrastructure. The RAPTOR-XPress base board (see figure 3) can be equipped with up to four daughterboards and provides extensive capabilities for system management and communication. The host connection is realized using eight PCI Express 2.0 channels. Using a dedicated PCI Express switch on the base board, each daughterboard can access the full bandwidth of 32 GBit/s to the host, allowing for a fast, low-latency access of the host CPU to each FPGA in the system. A PCIe to local-bus bridge allows for simple bus access to ease porting legacy FPGA designs and to reuse older modules of the RAPTOR family that do not offer PCIe links. Furthermore, interfaces for USB 2.0 high-speed and Gigabit Ethernet are available.

For communication between FPGA modules, the RAPTOR-XPress base board offers direct connections between adjacent modules. This facilitates a ring topology between all modules on one base board with a bandwidth of 80 GBit/s and a latency of less

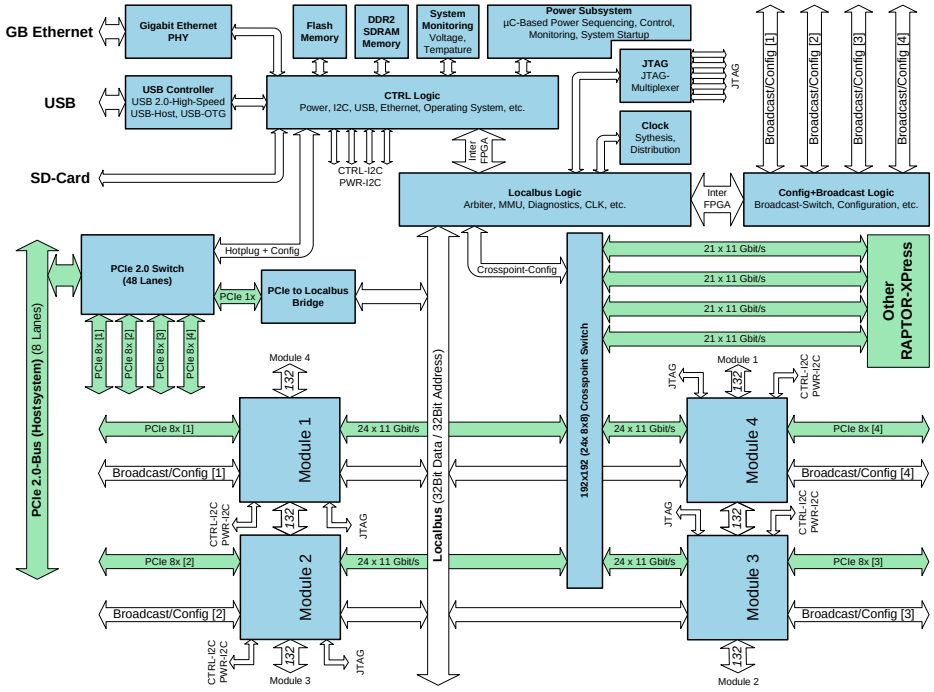

Figure 3. Architecture of the RAPTOR-XPress Baseboard.

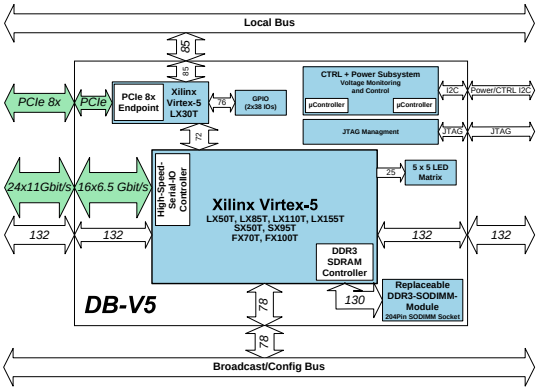

Figure 4. Architecture of the Xilinx Virtex-5-FX-based module DB-V5.

than 10 ns. Furthermore, a central switch, implemented in a dedicated FPGA on the base board, provides the modules with an additional bandwidth of 10 GBit/s in any topology.

For the communication between multiple RAPTOR-XPress base boards four serial high-speed connections are used. The technology is basically the same as for the DB-V4, enabling reuse of the developed protocols and of the switch IPs. The RAPTOR-XPress base board realizes 24 serial high-speed (full duplex) lanes to each module. Each of the four connectors to other RAPTOR-XPress base boards is connected to 21 serial high-speed (full duplex) lanes. Each of these 180 full-duplex lanes offers a bandwidth of 11 GBit/s. The topology of these connections can be changed at runtime using a 180x180 crosspoint switch (1980 GBit/s aggregate bandwidth) on the base board. The first module that realizes the new concepts integrates a Xilinx Virtex-5 FX100T and up to 4 GByte DDR3 memory (see figure 4). The PCIe interface is realized by a dedicated Virtex-5 FX30T on this module.

In addition to the management features of RAPTOR-X64, RAPTOR-XPress integrates advanced temperature monitoring and power management. Supply voltages can be adapted at runtime and power-up sequences for the whole system are controlled by a microcontroller. Voltages are automatically limited to applicable values, to provide maximum compatibility with all existing modules.

Based on the new RAPTOR-XPress system we are currently setting up an FPGA cluster, which will be especially suited for the emulation of large MPSoCs and for high-performance computing since it offers a unique communication infrastructure between the FPGAs and between the FPGAs and the host computers. As depicted in figure 5, the cluster will combine RAPTOR-XPress systems and special FPGA modules with direct connection to the frontside bus (FSB) of the host processor, provided by Nallatech [5]. These FPGA modules can be attached to a socket 604 type CPU socket on an Intel Xeon-based mainboard. This allows to combine the FPGA-module with a normal 7300 (Tigerton) or 7400-series (Dunnington) Intel Xeon CPU on a multi-CPU mainboard. The direct connection to the FSB offers 8.5 GByte/s sustained peak write bandwidth, 5.6 GByte/s sustained peak read bandwidth, and 105 ns latency. Beside the direct connection to the FSB, the FPGAs can communicate with any other FPGA in the cluster using the same serial high-speed links that are used for the connections between multiple Raptor-XPress boards.

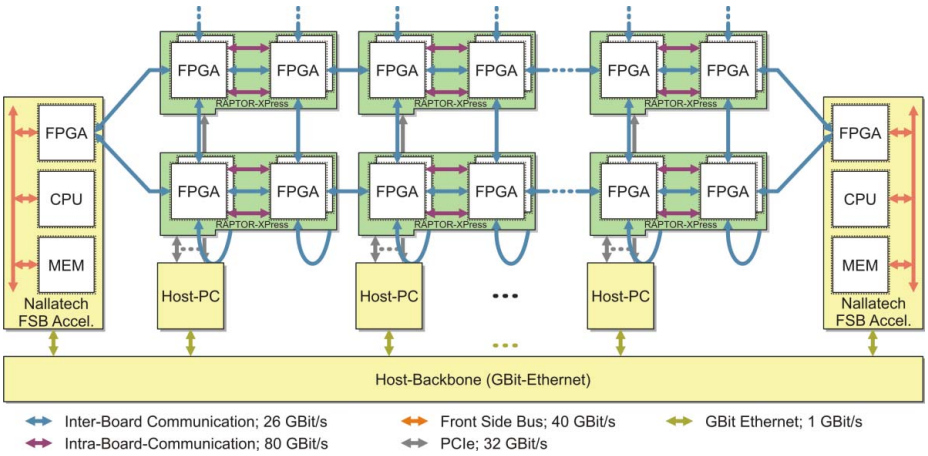

Figure 5. Architecture of the RAPTOR-Xpress-based FPGA cluster, comprising 72 Xilinx Virtex 5 FPGAs

The starting configuration of the cluster consists of 64 Virtex-5 FX100T-2 FPGAs on 16 RAPTOR-XPress systems and 8 Virtex-5 FX200T-2 with FSB-connection. All FPGAs are closely coupled using the proposed high speed serial interfaces, offering a point-to-point bandwidth of 26 GBit/s in a matrix configuration of the crosspoint switch. In total, about 1.3 Million Virtex-5 FPGA slices, 19500 DSP-blocks, 256 GByte DDR3 RAM, 80 MByte BRAM, 12 MByte distributed RAM, 144 integrated PowerPC processors, and 256 GByte of embedded memory are available to the user.

3. Software Environment

For easy and comfortable use of the RAPTOR systems, the software environment RaptorSuite has been developed, consisting of three layers. The bottom layer, called RaptorLIB, offers a direct interface to the hardware, which is nearly the same for all supported interfaces (USB, PCI, PCI-X, and PCIe, Ethernet), such that an application can easily switch between these protocols. By using the RaptorAPI, remote usage capabilities are added, enabling remote access to the RAPTOR systems using a client/server infrastructure via the local network or the Internet. The graphical user interface RaptorGUI implements comfortable management functionalities and allows basic tests of an application without the need to develop a specific software application based on RaptorLIB/API. Hardware-Software co-verification is further facilitated by a hardware-in-the-loop environment, enabling, e.g., a direct connection between RAPTOR systems and simulators or graphical environments, e.g., Modelsim, Simulink or System Generator [6].

The design-flow for this FPGA based cluster deviates from standard FPGA flows, as an additional partitioning step has to be performed. There are tools available (e.g., Synopsys Certify [7]), which allow for the partitioning of designs for multi-FPGA prototyping environments with a fixed communication infrastructure. For the FPGA cluster, however, this cannot be applied, especially as Certify currently does not support RocketIOs as communication channels. Apart from that, the partitioning algorithm is optimized for prototyping large irregular structures (e.g., singular processors) rather than for

regular structures as required by the typical application scenarios sketched in section 4. Therefore, our design-flow relies on a tool-supported rather than an automatic approach. Basically, the partitioning step is done manually, but the user is supported by a graphical IDE that allows for easy partitioning of FPGA code and, by utilizing standard synthesis tools, for easy assessment of the required FPGA resources. The IDE and all associated tools use vMAGIC, a library for reading, manipulating and writing VHDL code [8].

4. Application Examples

In this section we describe two applications mapped to our FPGA cluster. The emulation of large Multiprocessor Systems-on-Chip (MPSoCs) was one of the most important reasons to set up the FPGA cluster. As a first example of a scalable, network on-chip (NoC) based MPSoC architecture, the GigaNetIC architecture [1] has been implemented on the RAPTOR systems. An MPSoC consisting of four 32-bit N-Core RISC processors [1], embedded program and data memory, and the switch box of the NoC can be emulated on the DB-V4. Therefore, GigaNetIC systems with up to 24 embedded processors can be emulated on a single RAPTOR-X64 system. Using simpler NOCs and smaller processors (or processors that are optimized for FPGA implementation like the Xilinx MicroBlaze [2]) the number of processors can easily exceed a hundred.

The second application is an implementation of a neural network based algorithm for hardware accelerated data analysis: Kohonen's self-organizing map (SOM) [9]. SOMs use an unsupervised learning algorithm to form a nonlinear mapping from a given high-dimensional input space to a lower-dimensional (in most cases 2-D) map of neurons. Here, the neurons are emulated by processing elements (PEs), operating in a SIMD (single instruction, multiple data) manner most of the time. The algorithm works as follows:

1. Initialization: The weight vectors $\mathbf{m_i}$ of all neurons N_i need to be initialised, this is often done by assigning random values:

$$\forall_{i,j}\ m_{i,j} = rand[0...1], \tag{1}$$

   where j denotes the components of the vector $\mathbf{m_i}$.
2. Search bestmatch: A vector $\mathbf{x}(t)$ is randomly selected from X and the distance between $\mathbf{x}(t)$ and all $\mathbf{m}_i$ is calculated. The neuron with the smallest distance to the input is called bestmatch (BM).

$$\|\mathbf{x} - \mathbf{m}_{bm}\| = \min_{\forall_i} \|\mathbf{x} - \mathbf{m}_i\| \tag{2}$$

3. Adaptation: The weight vectors $\mathbf{m}_i$ are adjusted to the input $\mathbf{m_i}$ according to their distance to the bestmatch in the grid.

$$\forall_i\ \mathbf{m}_i(t+1) = \mathbf{m}_i(t) - |\mathbf{m}_i(t) - \mathbf{x}(t)| \cdot h_{ci}, \tag{3}$$

$$\text{where } h_{ci} = h_{ci}(t, \|N_i - N_{BM}\|) \tag{4}$$

   The so called neighborhood function h_{ci} is a function that decreases in space and time, often a Gauss-kernel is chosen.

For an optimal resource utilization on different systems, two different versions of the algorithm have been implemented on the RAPTOR systems. The algorithm itself reveals two inherent degrees of parallelism: all neurons perform the same operations on the same data vectors (neuron-parallel), and all vector components are processed in parallel (component parallel), respectively. For the neuron-parallel approach, up to 512 processing elements can be realized on RAPTOR-X64. The rather more complex and slightly less resource efficient neuron- *and* component-parallel approach is used on the RAPTOR-XPress cluster, mainly for one reason: utilizing all 64 compute FPGAs of the cluster, a total of $128 \times 64 = 8192$ PEs may be instantiated using the cluster. The largest maps currently in use at our department consist of 1600 neurons; therefore, approx. 80% of the cluster could not be utilized using this architecture. The neuron- and component-parallel approach, however, can utilize the complete processing power, because several processing elements can be assigned to the emulation of one neuron. Measurements in comparison with an Intel Core2 Duo processor running at 2.5 GHz show a speed-up of about 10 for RAPTOR-X64 and a speed-up of more than 80 for the RAPTOR-XPress cluster.

5. Conclusion

FPGA-based prototyping systems have been proposed, which offer emulation capacities ranging from typical embedded processor architectures to next-generation high-end MPSoCs. A tightly integrated cluster of FPGAs, together with a software environment for architecture mapping, monitoring, and debugging enables the fast analysis of MPSoCs with hundreds or even thousands of processors. Furthermore, the close coupling of the cluster to the host PCs via PCIe and especially via the frontside bus of the processors enables the development of powerful hardware accelerators, e.g., to speed-up large scientific simulations.

References

[1] J.-C. Niemann, C. Puttmann, M. Porrmann, and U. Rückert. Resource Efficiency of the GigaNetIC Chip Multiprocessor Architecture. *Journal of Systems Architecture (JSA)*, 53:285–299, 2007.

[2] A. Krasnov, A. Schultz, J. Wawrzynek, G. Gibeling, and P.-Y. Droz. RAMP Blue: A Message-Passing Manycore System in FPGAs. In *Proc. of FPL2007*, pages 54–61, 2007.

[3] Annapolis Micro Systems, Inc. Heterogeneous Processing Platform – FPGA Solutions for the IBM Blade Center, White Paper: 200905.01, 20 May 2009.

[4] Heiko Kalte, Mario Porrmann, and Ulrich Rückert. A Prototyping Platform for Dynamically Reconfigurable System on Chip Designs. In *Proc. of the IEEE Workshop Heterogeneous Reconfigurable Systems on Chip (SoC)*, 2002.

[5] C. Petrie. A review of the FPGA high performance computing industry and the future role of FPGAs within data-centric processing architectures. In *Proc. of the Many-core and Reconfigurable Supercomputing Conference*, 2008.

[6] C. Paiz, C. Pohl, and M. Porrmann. Hardware-in-the-Loop Simulations for FPGA-Based Digital Control Design. *Informatics in Control, Automation and Robotics*, 3:355–372, 2008.

[7] Synplicity. Certify ASIC prototyping solution. online.

[8] Christopher Pohl, Carlos Paiz, and Mario Porrmann. vMAGIC - Automatic Code Generation for VHDL. *International Journal of Reconfigurable Computing*, Special Issue ReCoSoC:7, 2009.

[9] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, second edition, 1997.