Building a Multi-FPGA-based Emulation Framework to Support NoC Design and Verification

Yangfan Liu[a], Peng Liu[a*], Yingtao Jiang[b], Mei Yang[b], Kejun Wu[a], Weidong Wang[a] and Qingdong Yao[a]

[a]*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China*
[b]*Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA*
Corresponding author email: liupeng@zju.edu.cn

Abstract: In this paper, we present a highly scalable, flexible hardware-based NoC emulation framework, through which NoCs built upon various types of network topologies, routing algorithms, switching protocols, and flow control schemes can be explored, compared, and validated with injected or self-generated traffic from both real-life and synthetic applications. This high degree of scalability and flexibility is achieved due to the FPGA design choices made at both functional and physical levels. At the functional level, an NoC system to be emulated can be partitioned into two parts: (i) the processing cores and (ii) the network. Each part is mapped onto a different FPGA so that when there is any change to be made on any one of these two parts, only the corresponding FPGA needs to be reconfigured and the rest FPGAs will be left untouched. At the physical level, two levels of interconnects are adopted to mimic NoC on-chip communications: high bandwidth and low latency parallel on-board wires, and high speed serial multigigabit transceivers available in FPGAs. The latter is particularly important as it helps the proposed NoC emulation platform scale well with the size increase of the NoCs.

Keywords: networks-on-chip; emulation; FPGA; verification; on-chip interconnection networks.

1. Introduction

In future many-core system on chip (SoC) designs, on-chip interconnect links between cores have significant effects on the overall system performance and power consumption. In these SoCs with tens or even hundreds of cores integrated either onto a single integrated circuit die, or onto multiple dies in a single chip package, traditional interconnection techniques using a bus topology or dedicated wires are no longer feasible due to insufficient data bandwidth and poor scalability. As a viable alternative, the networks-on-chip (NoC) paradigm has been proposed as an enabling replacement to overcome the communication bottlenecks in future many-core SoCs (Benini and Micheli 2002).

In an NoC system, processing elements, such as processor cores, memories and specialized intellectual property (IP) blocks, exchange data using the network constructed from multiple point-to-point data links interconnected by switches/routers. There are a number of critical issues that need to be considered in designing a power-efficient and high-performance NoC, such as the topology selection, design of network interface and communication protocols, and fault tolerance. All these wide range of design choices require a very time-consuming and error-prone tuning and verification process, which involves extensive use of the software- and/or hardware-based simulation/emulation techniques and tools.

- Software-based simulation approaches (Bertozzi et al. 2005, Mahadevan et al. 2007, Chan and Parameshwaran 2004, Coppola et al. 2004) are suitable for system design at early stages as they allow rapid design space exploration due to their flexibility and low cost. However, these approaches often suffer from relatively slow speed (Genko et al. 2007), and thus their applicability to extensively evaluate a complete NoC system with real world data traces is questionable.
- Hardware-based emulation solutions (Genko et al. 2007, Moraes et al, 2004, Ogras et al. 2007, Krasteva et al. 2008, Kumar et al. 2007, Abdellah-Medjadji et al. 2008), for instance, the ones using

FPGA, on the other hand, can help drastically reduce the system evaluation time without compromising accuracy. One big problem of this solution is that it does not scale well with the large systems due to the resource limits of FPGA devices. In addition, FPGA-based emulation design typically requires the whole system to be re-synthesized and re-implemented on FPGA when there are any architectural and/or logic changes ever to be made on the system to be emulated.

In this paper, we propose a scalable NoC emulation platform implemented with multiple FPGA devices. Using this proposed emulator, designers shall be able to explore, compare, and verify every aspect of a complete NoC design for complex many-core SoCs, including network topologies, routing algorithms, switching protocols, and flow control schemes, with injected or self-generated traffic from both real-life and synthetic applications.

To make the emulation platform highly scalable and also minimize the re-synthesis needs when evaluating different NoC designs, we propose a number of solutions in both software and hardware layers.

- At the hardware layer, an emulation platform is built upon one or multiple emulation module boards. Each single module board consists of five FPGA chips, and it can emulate a complete 4x4 NoC architecture using a 32-bit RISC processor core. Functionally, these five FPGAs are partitioned into two parts: the resource part (4 FPGAs) and the network part (1 FPGA). One big benefit by doing this is that when certain functionality change needs to be made, only the impacted FPGA needs to re-synthesized. In addition, not only can this platform employ high bandwidth and low latency parallel on-board wires to mimic NoC on-chip communications, it also utilizes the high speed serial multigigabit transceivers available on FPGA to expand effortlessly along with the size increase of the NoC to be emulated, which allows multiple emulation module boards can be readily connected and configured to emulate NoCs with much larger sizes (i. e., 8x8, 16x16, or even larger).
- At the software layer, a real-life application program can be compiled using a customized parallel compiler. Basically, a program is partitioned into smaller pieces of run time units, tasks and/or thread constructs, and each unit is bounded to a processing core and executed synergistically in the context of an NoC. Moreover, software tools running at the host computer can set up the emulation parameters, control the emulation process, and collect the emulation results for display and further analysis.

The remainder of the paper is organized as follows. In Section 2, we briefly discuss the related work, and the functionalities of the proposed emulation platform are given in Section 3. In Section 4, we specify the NoC emulation framework and emulation module board design with wire modeling. In Section 5, we present a detailed explanation of the evaluation/verification flow, two types of partition strategies and the scale-up methodology. Implementation and experiments using the proposed emulation platform have been reported in Section 6. Finally the conclusions are drawn in Section 7 along with the suggested future work.

2. Related work

In recent years, significant research efforts have been made in NoC simulation and emulation to evaluate NoC designs at different abstraction levels. Several simulation environments in SystemC (Bertozzi et al. 2005, Mahadevan et al. 2007) have been proposed in order to perform NoC design without going down to extensive process of physical synthesis. A VHDL-based parameterized model to evaluate the performance in mesh NoC topology is presented (Chan and Parameshwaran 2004). An efficient framework based on object-oriented C++ library built on top of SystemC is provided for NoC simulation and design exploration (Coppola et al. 2004). However, the disadvantage of these software solutions is their relatively slow speeds.

FPGA-based emulation systems are proposed to emulate NoC architectures for reducing validation time. A switch employed an XY routing algorithm has been realized in an FPGA to validate the interconnection network (Moraes et al, 2004). In (Ogras et al. 2007), four NoC prototypes targeting specific applications, especially

multimedia applications are discussed and a 4x4 mesh network without an actual processing core is implemented. An FPGA emulation-based NoC prototyping framework is introduced (Krasteva et al. 2008), which utilizes partial reconfiguration capabilities of FPGA to reduce re-synthesis time for accelerating emulation process, and a 2x2 mesh is mapped onto an FPGA as a demonstration. A hardware-software emulation framework implemented on an FPGA is presented (Genko et al. 2007). It is shown that the FPGA-based emulation framework achieves four orders of magnitude of speedup over a software simulator. An integrated flow to automatically generate configurable NoC-based multi-processor SoC is proposed (Kumar et al. 2007). The complete NoC architecture with three cores, a single router and two network interfaces is emulated on an FPGA.

However, none of the above works is capable of emulating a complete, large scale NoC, as their focus is on either limited aspects of NoC systems or small scale NoC architectures caused by the resource limitation of a single FPGA. To address this scalability problem, in (Abdellah-Medjadji et al. 2008), a multi-FPGA-based platform which can evaluate large scale NoCs using multiple XUP VirtexII Pro boards is designed. To connect multiple FPGA boards, high speed serial links are employed which simulate physical links between routers. Wire-multiplexing technique is applied to overcome resources and pin limitations. An NoC prototype with only the traffic generators and receptors is implemented on two interconnected boards as illustration. A similar platform built with multiple built on multiple Altera FPGA boards is reported in (Kouadri-Mostefaoui et al. 2008). However, the scalability of these designs is limited with single FPGA per board.

In the following, we will present a flexible and scalable multi-FPGA emulation module board for complete prototyping of large scale NoCs, which consists of processing cores together with an on-chip network. This emulation module uses parallel on-board wires with a high bandwidth and a low latency to implement the physical links. This way, the performance statistics for large scale NoCs to be emulated can be readily obtained.

3. Functions and features of the proposed NoC emulation platform

The objective of this research is to build a multi-purpose, scalable emulation platform which can evaluate, compare, and verify various NoC systems for different applications. In specific, this proposed emulation platform will be able to emulate every aspect of an NoC communication process which encompasses the functions of traffic pattern generation, routing/switching, flow control, process monitoring and evaluation, result data collection and analysis. As shown in Fig. 1, these functionalities can be abstracted and conveniently clustered into a three-layer structure which consists of the application, the link/network, and the physical layers (Liu et al, 2009a).

The application layer handles end-user applications under different traffic patterns. Most of the network implementation details are hidden from this level. Detailed functionalities that will be supported at this application layer are summarized below.

- Emulate different types of latency critical or data streaming applications' behaviors under multiple traffic patterns, including stochastic traffic in uniform or non-uniform distributions, and real-life traffic.
- Collect transient and statistic results for performance analysis in terms of latency, throughput, degree of congestion, power consumption, and so on.

The link/network layer concerns the network and router aspects of an NoC design, including the network topology, routing algorithm, switching strategy, flow control scheme, and communication mode. Detailed functionalities that fall into this link/network layer are summarized below.

- Evaluate both regular network topologies, such as mesh, torus, binary tree, PRDT (Yang et al. 2007), etc, and irregular topologies.
- Validate different deterministic and adaptive routing schemes, including deterministic source routing, X-Y routing, and various adaptive routing schemes that the routing path is decided on a per-hop basis

by dynamic arbitration mechanisms. There are a number of routing related features, and thus the emulation platform shall be able to model and emulate: (i) various switching protocols including the circuit switching, the packet switching, the virtual cut-through, the wormhole switching, and the hybrid switching schemes; (ii) various flow control schemes that involve different buffering schemes and virtual channels; and (iii) three communication modes: unicast, multicast, and broadcast.

At the physical layer, in addition to modeling and emulating the behaviors of the physical wires, the processor type, the memory, and the core clock shall be specified for a processing core. Synchronization at the physical layer is an utmost concern to realize reliable data transfer in an NoC system running with multi-clock domains. Also at the physical layer, the wire delay and the power consumption has to be estimated while studying the interconnect wire effects. In what follows, the functionalities that are supported at the physical layer are summarized.

- Realize synchronization for reliable data transfer in a multi-clock NoC system.
- Estimate wire delay and power consumption of various types of interconnection wires.

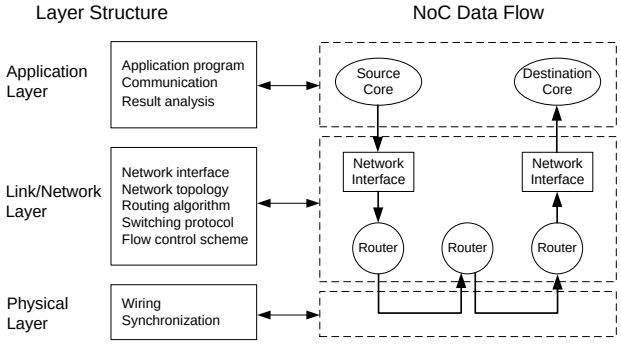

Figure 1. A Layered Structure for NoC Emulation.

Besides all the aforementioned functionalities that are supported by the proposed NoC emulation platform, there are a number of implementations and hardware features that can help effectively accomplish these functionalities.

- High speed: the whole system shall be able to take advantage of modern FPGAs running at high frequencies, 100MHz or higher. High speed FPGAs can help reduce emulation/simulation time considerably, as compared with software simulators. The high evaluation speed also enables an integral functional verification with a large amount of data and more scenarios.
- High accuracy: the NoC architecture is modeled in hardware description language (HDL) code, and cycle level accuracy is achievable.
- Flexibility: it is convenient to emulate several network topologies, explore various router structures, and implement different applications by reconfiguring corresponding FPGAs without re-synthesizing the whole architecture.
- Scalability: the platform provides the capability to extend the emulation system for a much larger scale NoC architecture involving a much larger number of processing cores (e.g., 8x8, 16x16 or even larger) by utilizing the serial multigigabit transceivers on FPGA to connect multiple emulation module boards.
- Great design space exploration: the emulator can validate various NoC characteristics for real-life applications, including network topology, routing algorithm, flow control scheme, and link synchronization implementation, etc. Hence, the optimum NoC communication architecture can be explored to meet different design and performance study needs.

4. NoC emulation design

4.1 *Emulation framework*

To achieve the functions and features described in Section 3, a multi-FPGA-based emulation framework is designed. The proposed NoC emulation framework includes both a hardware part and a software part as shown in Fig. 2.

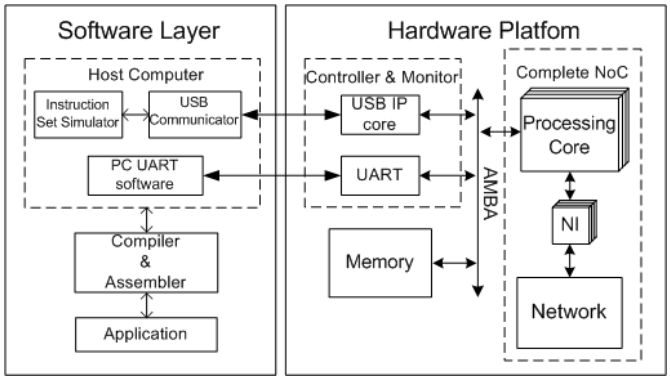

Figure 2. NoC emulation framework.

In essence, the hardware part harbors the complete NoC architecture to be emulated. The memory supplies data storage space for the processing cores. The controller and monitor unit initializes the NoC communication and collects the emulation result data. Besides, the hardware platform features a universal serial bus (USB) IP core and a universal asynchronous receiver transmitter (UART) interface between the host computer and the NoC. To allow the interaction between the hardware and the software, interface controllers that are accessible through a high performance advanced microcontroller bus architecture (AMBA) have been designed.

Software layer is responsible for (i) compiling and partitioning an application program into tasks and then assigning them to the processor cores, (ii) setting up the emulation parameters, (iii) controlling the emulation process of NoC architecture, and (iv) displaying result statistics. In general, designer can utilize software tools residing and running at the host computer to configure the hardware emulation platform for different NoC emulation purposes. The software tools contain USB communicator, instruction set simulator, and personal computer (PC) UART monitor program.

1) USB communicator is used to access USB hardware based on USB 1.1 protocol. All address space of the AMBA bus can be accessed by the USB communicator. It can be used to download the compiled program into the SRAM and SDRAM for processor core and collect the result that is stored in memory. This program is loaded during the initialization and controls the emulation process that has several driver functions.

2) Instruction set simulator (ISS) is an instruction-accurate simulator whose instructions are compatible with MIPS 32 instruction set (MIPS Corp 2005). The ISS can connect with USB interface so as to access the AMBA bus. The ISS can replace the processor IP soft core for simulating the application program.

3) PC UART monitor completes PC UART functions. Any commercially available PC UART software tools can be used to communicate with hardware platform. The program can initialize UART, print initial information, and wait for user's input to configure parameters for NoC emulation.

4.2 *Emulation module board*

The NoC emulation module board consists of five Xilinx Virtex-5 LX110T FPGA chips shown in Fig. 3. The physical wires on the emulation module are organized as low-voltage CMOS (LVCMOS) parallel links and multigigabit transceiver serial (MGT) lines. The four surrounding FPGAs are connected through a 2D mesh grid. Each link between the adjacent FPGAs on the grid provides 90 single-bit lines running at 100MHz with a total data throughput of 9Gb/s. The 152-bit parallel LVCMOS interconnection wires are provided between the middle FPGA and the surrounding FPGAs to achieve a total of 15.2Gb/s data bandwidth. All these FPGA-to-FPGA parallel links, which can faithfully emulate the interconnection links in an NoC architecture, form one virtual FPGA but with much larger resources capacity than any one actual FPGA can provide.

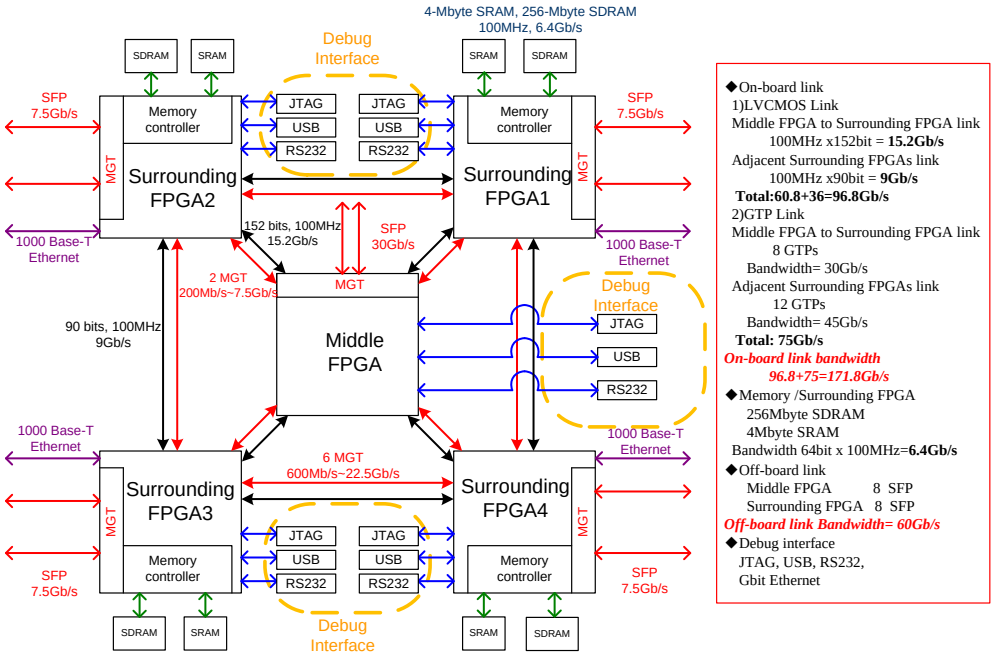

Figure 3. NoC emulation module block diagram.

MGT is a power-efficient transceiver for Xilinx Virtex-5 FPGAs. It is highly configurable and tightly integrated with the programmable logic resources of the FPGA. The full-duplex serial channel can support 100Mb/s to 3.75Gb/s bandwidth using 8B/10B encoding (Xilinx 2008). There are a total of 16 MGT transceivers on the emulation model for off-board extension to provide up to 60Gb/s communication bandwidth. The MGT serial interfaces are thus used to connect on-board and off-board FPGAs for scalability purposes. That is,

- For the middle FPGA, 2 MGT transceivers are used to connect it with each of the four surrounding FPGAs and the remaining 8 MGT transceivers are reserved for connecting to other FPGAs in another module board (off-board extensions).
- For each surrounding FPGA, 2 MGT transceivers are needed to connect with the middle FPGA, 6 MGT transceivers to connect with every adjacent surrounding FPGA, and the rest 2 MGT transceivers reserved for off-board connections.

Each FPGA on a module board has reserved some MGT channels for off-board connections. Each of these off-board MGT channels is connected to small form-factor pluggable (SFP) connector. The SFP transceiver is designed to support synchronous high speed data communications protocols, such as the optical networking (SONET), Gigabit Ethernet, Fibre Channel, and other communications standards, with data rates up to 4.25 Gb/s. Using these SFP transceivers, the emulation system can be extended to include multiple emulation module boards that support a larger NoC architecture.

Each surrounding FPGA communicates with SRAM (4Mbytes) and SDRAM (256Mbytes) for data storage. Each memory channel is running at 100MHz with a 32-bit or up to 64-bit data interface. As such, the peak aggregate memory bandwidth for one FPGA can reach 6.4Gb/s.

The emulation module utilizes various available FPGA interconnect interfaces, including JTAG, USB, and RS232 serial port to (i) connect the host computer with the emulation module to set up the emulation parameters, (ii) control emulation process, and (iii) collect result data. Moreover, each surrounding FPGA provides one 1000Base-T Ethernet interface for other types of data communications.

4.3 *Modeling of wires*

With the modern deep submicron and nano fabrication technologies, the interconnect wires, as opposed to

logic cells, dominate the system performance in terms of timing delay and power consumption. It is significant to study the NoC architecture on wire modeling at the physical layer (Fig. 1). The on-board wires and internal wiring inside an FPGA chip will be modeled to provide emulation for on-chip wires of an NoC system so that physical design issues, such as wire delay and power consumption can be estimated.

The wiring delay of a distributed RC line can be modeled as follows (Liu et al. 2003).

$$T_{wire} = 0.39rcl^2 \quad (1)$$

where T_{wire} is the wiring delay, l is the wiring length, r is the resistance per unit length and c is capacitance per unit length. The delay can be reduced by employing various circuit structures, like inserting repeaters (Liu et al. 2003).

Based on Eq. (1), the wire delay on different links between routers in an NoC architecture can be determined, and thus the wires in PCB board are deliberately designed so that they exhibit the same delay as appeared in an NoC chip.

Power consumption is another design issue to be studied. The average packet traversal energy can be utilized as a network energy efficiency metric (Lee et al. 2006). The energy consumed by one packet from the sender to the receiver can be estimated by the following equation (Dally et al. 2001, Lee et al. 2006).

$$E_{pkt} = H \cdot (E_{queue} + E_{SF} + E_{ARB}) + L \cdot E_{Link} + E_{Queue} \quad (2)$$

where H and L are hop counts and link distance, between a sender and a receiver respectively. Energy consumption on a switching hop is composed of energy consumption in an input queuing buffer or latch E_{queue}, switching fabric E_{SF}, and arbitration logic E_{ARB}. E_{Link} stands for transmission energy on a unit length link.

In our design, Eq. (2) is used to estimate the energy consumed by one flit traveling from the sender to the receiver. Estimation of these parameters can be obtained by synthesis on Synopsys Power Compiler tool. For example, the power consumption on wires consists of dynamic power and leakage power. The leakage power can be estimated from Synopsys tool. The dynamic power can be estimated as $0.5cV^2fa$, where c is the capacitance of the wire, V is the voltage to be charged to, f is the clock frequency, a is the number of switching activities per unit time. The switching activities per unit time can be acquired by simulation tool, such as Modelsim. Thus, the dynamic power can be reported by Power Compiler.

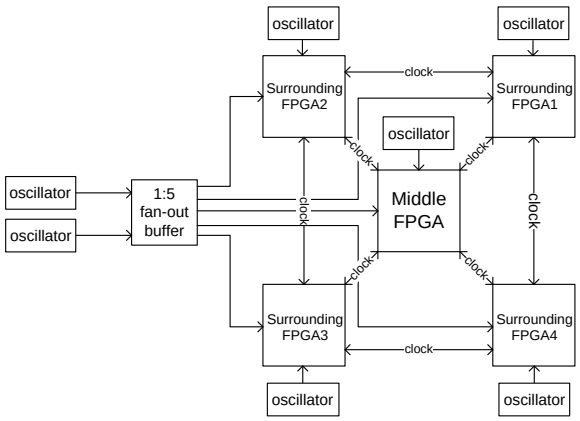

Figure 4. Synchronization on emulation platform.

Synchronization on the wires at the physical layer is important to implement reliable data transfer in the multi-FPGA NoC emulation system. In digital system communication, when a transmitter chip sends data to receiver chip, the receiver must sample the data with some clock source. One option is to generate a clock on the board and distribute it to both chips. This implementation requires slower operating speed so that the clocks can be in phase with each other. The other is that transmitter sends both the clock and data to the receiver, which can run at higher operating speed. The receiver chip uses asynchronous FIFO to accomplish transformation between

two clock domains. The FIFO in the router input/output channel can be employed justifiably. In the proposed emulation platform, both synchronous schemes, shown in Fig. 4, are supported.

5. Work flow using the proposed NoC evaluation on emulation platform

5.1 *Synthesis flow*

To use the proposed emulation platform described in Sections 3 and 4, the NoC to be emulated shall be modeled at the RTL level using HDL, preferably Verilog. Fig. 5 shows the synthesis flow to validate an NoC design. The flow begins with a complete NoC architecture and an underlined application. As there are five FPGAs in the emulation platform, the whole NoC architecture needs to be partitioned into no more than five subsets, and afterwards, each subset is synthesized and mapped onto one FPGA.

The software tools running at the host computer are in charge of downloading the application programs to the FPGAs, setting up all the configuration parameters, controlling the emulation process, and displaying the results collected from the emulation hardware.

The FPGAs on emulation platform can be configured via the JTAG interface (Xilinx Corp 2008). All five FPGA chips are connected in a serial daisy chain, as shown in Fig. 6. The devices connected in the JTAG chain are configured one by one according to NoC partition subsets shown in Fig. 5.

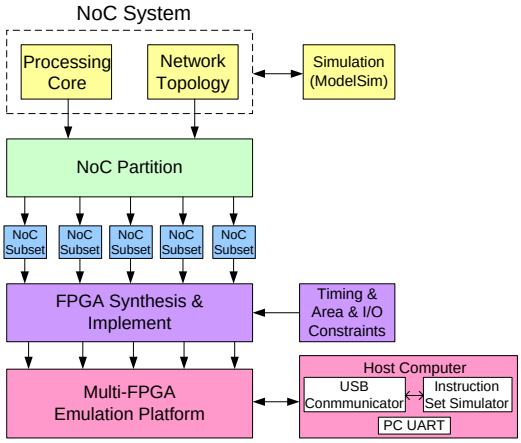

Figure 5. NoC emulation synthesis flow.

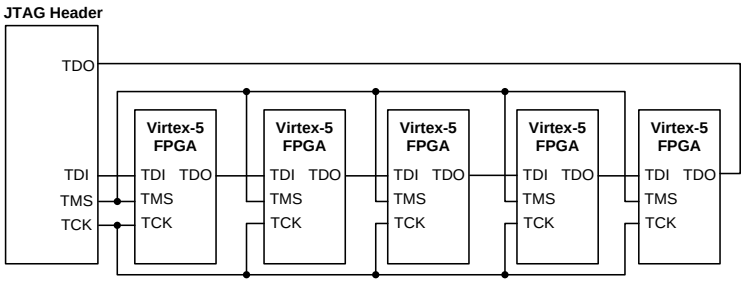

Figure 6. JTAG configuration chain for FPGAs.

5.2 *NoC partition*

With five FPGAs available on the emulation module board, the NoC architecture being emulated needs generally to be partitioned into five subsets. The partition strategy should lead to minimum inter-FPGA connections, meanwhile, it must ensure that each subset fits on one FPGA chip in terms of logic resources. In the following, two types of partition strategies are described.

A. Function-based partition

In the function-based partition strategy, the five FPGA chips on the module board are logically partitioned into two sets based on their functions: the resource chip set which emulates processing cores and the network chip set which emulates various interconnection networks. The resource chip set is composed of the four

surrounding FPGAs, and the network FPGA is composed of the middle FPGA.

Each FPGA chip in the resource chip set can be configured to emulate single or multiple processing cores for different emulation requirements. The basic function units of a processing core include a packet generator (PG) and a packet receptor (PR). The PG generates packets under various traffic patterns and different communication modes (i.e., unicast, multicast, and broadcast). Traffic patterns include (i) stochastic traffic with uniform and non-uniform distributions and (ii) traffic generated from real-life applications. The PR, on the other end, not only handles received packets and conducts error checking, and also performs statistic analysis on traffic results and assesses each and every individual independent packet trace.

The network FPGA chip can be reconfigured to emulate different on-chip interconnection topologies, such as mesh, torus, and PRDT. In addition, various types of switching and flow control functions are supported by configuring this FPGA chip.

The function-based partition strategy utilizes the limited user I/O resource and reconfiguration feature of FPGA sufficiently. The parallel on-board wires implement interconnections between the processing cores and their routers. This partition strategy also simplifies the re-synthesis of FPGAs in the way that if there is any change to be made on one of the two parts, only the corresponding FPGA needs to be reconfigured and resynthesized and the rest FPGAs will be left untouched. For example, we can only reconfigure the network FPGA to alter the network topology or routing algorithm while the network interface between processing core and router is unchanged.

Under this partition strategy, the size of the NoC system to be emulated will be limited by the number of I/O pins available on the network FPGA. For the Virtex-5 LX110T FPGA (with 640 I/O pins), the emulation module can emulate up to 4x4 complete NoC systems with 16-bit data bandwidth of each channel between each processing core and its router. Fig. 7(a) illustrates the function-based partition strategy using the example of a 4x4 mesh-based NoC system. The network FPGA is mapped with a 4x4 mesh and each resource FPGA is mapped with four processing cores. Every core connects with its own router by parallel on-board wires. The specific signals between one processing core and its router are listed in Table 1. The communication channel (on one direction) between a processing core and its router consists of 19-bit, including 16 data bits, 2 control bits and 1 clock bit.

Table 1. Signals between processing core and router.

Name	Bit Width	Function
Packet_in	16	Flit data for input channel
Req_in	1	Request for input channel
Ack_out	1	Acknowledgement for input channel
Rx_clk	1	Clock for input channel
Packet_out	16	Flit data for output channel
Req_out	1	Request for output channel
Ack_in	1	Acknowledgement for output channel
Tx_clk	1	Clock for output channel

B. Region-based partition

In the region-based partition strategy, each FPGA chip implements a region of the NoC system which is composed of a set of processing cores and their routers. The partition of regions may be various to achieve different objectives, such as better utilization of resources on each FPGA or better utilization of the parallel wires on the board. Fig. 7(b) shows one type of the region partition for 4x4 mesh-based NoCs, which has better utilization of the parallel wires between the middle FPGA and the surrounding FPGAs.

Under the region-based partition strategy, the number of wires needed between two FPGAs is determined by the partition of regions. Table 2 lists the inter-FPGA wires, using 19-bit channel (16 data bits, 2 control bits, 1 clock bit) between routers, for different 4x4 NoC topologies based on the partition in Fig. 7(b). As the number of parallel wires available between two surrounding FPGAs is 90-bit, which is less than the needed amount for

torus and PRDT, the MGT serial links will be used for inter-FPGA communications between the surrounding FPGAs. Thus, wire multiplexing technique must be used.

Though extra logic for wire multiplexing is needed, compared with the function-based partition strategy, the benefit of the region-based partition strategy lies in the possibility of supporting larger size NoCs as the resources in the middle FPGA are better utilized under this partition strategy. For instance, by implementing four processing cores and their routers on one FPGA, the emulation module can support up to 4x5 network. In addition, the region-based strategy is convenient for emulating smaller size NoCs, e.g., only the middle FPGA need be configured to emulate 2x2 NoCs.

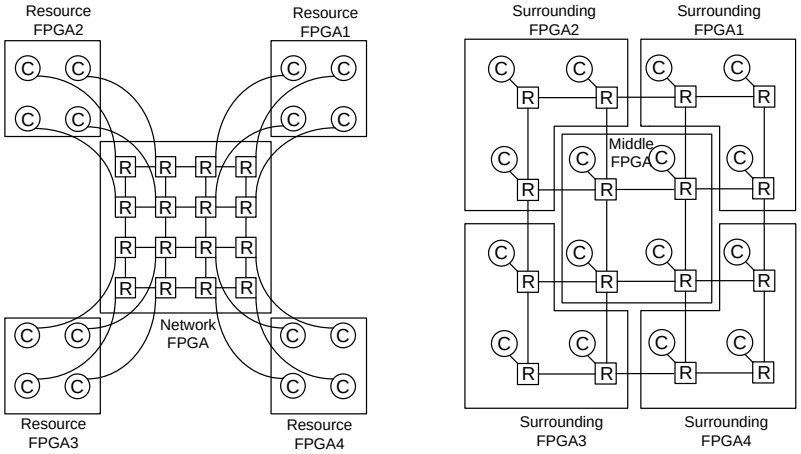

Figure 7. (a) Function-based partition. (b) Region-based partition.

Table 2. Inter-FPGA connections of 4x4 NoC based on the partition shown in Fig. 7(b).

Item	Requested Number of Wires		
	Mesh	Torus	PRDT
Middle FPGA—Surrounding FPGA	76	76	114
Adjacent surrounding FPGAs	38	114	152

5.3 *Scalability*

For an NoC architecture with many more processing nodes, multiple emulation modules can be built to construct even larger scale emulation system using MGT transceivers. We give an example of the possible interconnection of multiple emulation modules for mesh NoC topology under the function-based partition strategy. Fig. 8 shows the connection of four 4x4 meshes to form an 8x8 mesh NoC architecture.

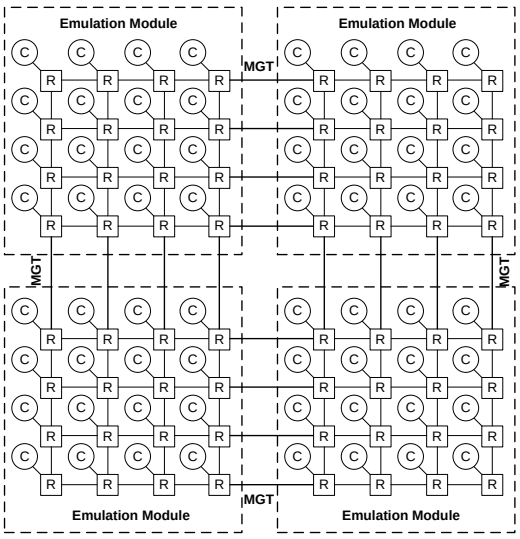

Figure 8. Interconnection of 8x8 mesh NoC using four emulation module boards.

To interconnect two emulation module boards, four MGT transceivers on the network FPGAs of different module boards will be needed, respectively. All eight MGT off-board links of network FPGA will be used for the connection of adjacent modules to form 8x8 mesh NoC. The connection channels between every two routers on the two adjacent emulation modules must be multiplexed onto one MGT transceiver. For other network topologies, such as torus, PRDT, more interconnections will be multiplexed on one MGT channel.

6. Implementation

6.1 *Board implementation and emulating configuration*

The emulation module is implemented on a 22-layer PCB board (14.3 inch x 12.4 inch). It consists of five Virtex-5 LX110T FPGAs with total 16MB SRAM and 1GB SDRAM for data storage. Besides, there are five USB, JTAG, and RS232 serial ports for debugging, four 1000Base-T Ethernet interfaces for communication, and 16 SFP connectors for off-board extension. The picture of the NoC emulation module board is shown in Fig. 9.

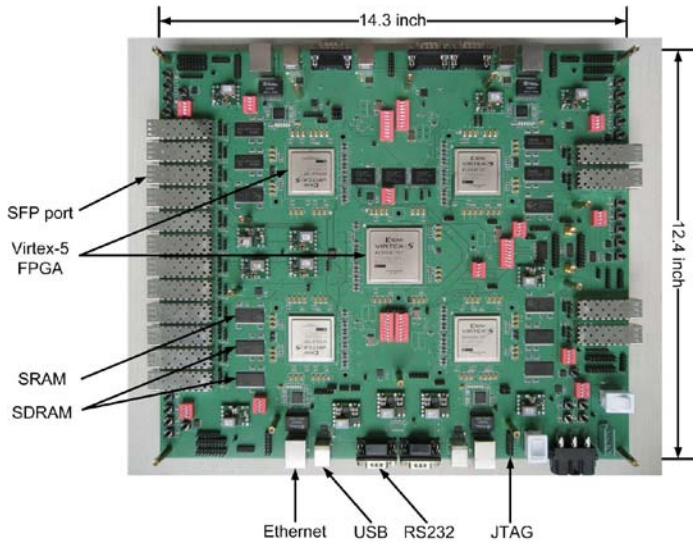

Figure 9. Picture of the NoC emulation module board.

The proposed emulation module board can be used for various design and verification purposes. In this section, one reference design is suggested with all possible configuration setups that are needed to run an NoC on the proposed emulation platform are given in Table 3. In specific,

- A processing core is configured with either a RISC core or a basic processing module with PG and PR, and the cache and translation lookaside buffer (TLB) sizes of the RISC core are configurable.
- The network interface (NI) (Xia et al, 2010), which enables a processing core to communicate with the network, can be partitioned into two parts: the resource-dependent part and the resource-independent part. There are three types of NIs with different resource dependence: (1) AMBA bus-based NI which connects the processor to the network through the AMBA bus; (2) direct memory access (DMA)-based NI which accesses processor's on-chip memory through a DMA channel; (3) memory-based NI which connects memory element with network that could respond requests.
- The architecture of a router can be configured in different aspects. The number of input/output ports can be modified according to the network topology. The buffer depth in each input/output channel of router can be changed according to the flow control strategy. The routing algorithm can be either a deterministic or an adaptive one. The switching scheme is fixed as wormhole switching.

The applications running on the emulated NoC architecture can be various, including synthetic traffics, real-life scientific computation kernels, such as FFT, Cannon and Gauss Jordan algorithm, and multimedia programs, for instance H.264, MPEG-1 and MPEG-4.

Table 3. Configuration setups of various building blocks in an NoC of interest.

Building Blocks	Type	Configurable Parameters
Processing core	RISC processor	ICache and DCache: 8, 16, 32, 64KB, 4-way, 16-bytes line size 2-level TLB: 3-entry ITLB and DTLB, 16-entry JTLB ScratchPad memory: 4, 8, 16KB
	PG & PR	—
NI	Resource dependent	AMBA bus-based, DMA-based, Memory-based
Router	Input/Output ports	Mesh with 4 ports, Torus with 5 ports, PRDT with 9 ports
	Buffer depth	4, 8, 16-entry FIFO
	Routing algorithm	Deterministic, Adaptive

6.2 *Emulation example*

To verify the implemented emulation platform, one emulation module board is used for evaluating the H.264 decoding program on a 2x2 mesh-based NoC configured with four RISC processor cores under function-based partition strategy. Each core is a high-performance 32-bit RISC processor compatible with MIPS4Kc (Liu et al. 2005). Fig. 10 shows the specific mapping of the 2x2 mesh-based NoC architecture. Each resource FPGA is configured with one RISC processor core which is attached to AMBA bus in order to connect with peripheral memory, communication, and debugging interfaces. One of these four cores, named as control core, is responsible for initializing the NoC communication, starting other cores, collecting the result data and calculating statistics, while the other three cores (called as synergic cores) are only used for computation. The network FPGA is mapped with 2x2 mesh of routers implemented with deterministic routing algorithm. Each router consists of five input/output ports, 16-depth first-in first-out (FIFO) buffers and two virtual channels for each port (Liu et al. 2009b).

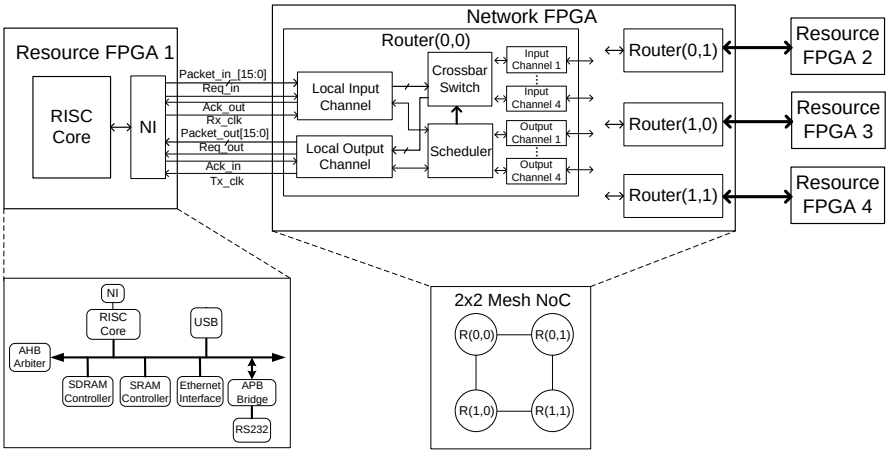

Figure 10. Specific mapping for 2x2 NoC architecture based on the partition shown in Fig. 7(a).

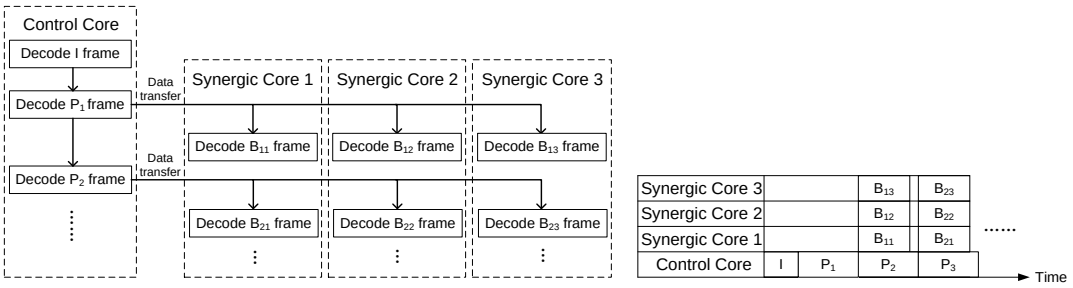

Figure 11. H.264 decoding program parallel partition and execution on a complete 2x2 mesh-based NoC system.

The H.264 decoding program is partitioned into four cores following a coarse-grained program partition strategy at frame-level (Fig.11). The program is first downloaded in the peripheral memory of the control core. The control core is started through external manual reset signal. During the initialization phase, the partitioned programs for the synergic cores are loaded to their corresponding cores by the control core though the 2x2 mesh network. After the initial program transfer, the control core starts the synergic cores running by configuring the

dedicated register connected with the reset signals of the synergic cores. During the program execution, all RISC cores also communicate through the network.

In this experiment, four typical video sequences with 58-frame in Quarter Common Intermediate Format (QCIF) format are tested. The application program is partitioned statically in frame-level shown in Fig. 11. The control core is responsible for decoding I-frames and P-frames in the sequence, and the synergic cores are in charge of decoding B-frames. To decode one B-frame, the decoded data of reference I-frame or P-frame need to be transferred for the synergic cores. Table 4 reports the execution time in clock cycles. It can be seen that the payload on the control core is heavy and the utilization of the synergic cores is about 75%. Moreover, the communication time accounts for less than 1% of the program running time because of the small amount of communication under the frame-level program partition strategy. The total number of data transferred between RISC cores is 90000 words (32-bit). The average latency to transfer one packet (32-bit data) on the 2x2 network is 2.6 clock cycles. The program execution time can be reduced by partitioning the program at finer grains (e.g., macroblock-level), which will better exploit the parallelism of program and balance the payload on each processor core.

Table 4. Execution time of H.264 decoding program on 2x2 mesh NoC.

Sequence	Total cycles (10^6)	Communication cycle (10^3)	Execution cycles on control core (10^6)	Execution cycles on one synergic core (10^6)
Foreman	621.89	235.55	621.89	458.99
Salesman	622.24	236.29	622.24	464.01
Container	610.26	237.87	610.26	432.96
Akiyo	605.87	234.43	605.87	442.71

As shown in Fig. 12, compared with software simulation in RTL level using Modelsim tool, our FPGA-based emulation platform achieves more than 10^4 magnitude speedup for the same experiment using foreman sequence. The speedup result is obtained in the following way. The execution time needed to decode the 58-frame sequence on our FPGA-based emulation module operating at 54MHz is 13 seconds. It takes 247 minutes to decode one frame on the software simulator, therefore, the simulation time needed to decode the whole sequence (58-frame) is approximately 239 hours. The environment of the software approach is AMD Opteron 2387 processor running at 2.80GHz configured with 4GB memory.

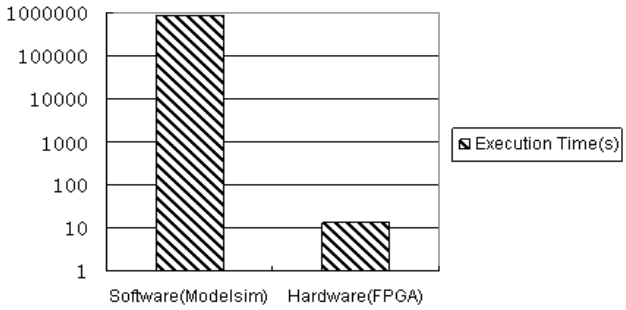

Figure 12. Comparison between hardware emulation and software simulation.

6.3 *Demonstrative evaluation*

Under the function-based partition strategy, as discussed in Section 5.2, a complete 4x4 mesh-based NoC system can be supported on one emulation module board by configuring each resource FPGA with four processing cores, and the network FPGA with 4x4 mesh of routers. The utilization of the resources on one FPGA is shown in Table 5, which shows that the network requires less logic resources than the RISC processing cores.

On the other side, under the region-based partition strategy, each FPGA chip is configured with four processing cores and their own routers. Based on the resource usage reported in Table 5, it is clear that the emulation module is capable of emulating a complete 4x5 mesh-based NoC system.

In addition, for evaluating interconnection network behaviors only, another configuration is used where each core is configured with the basic processing modules, PG and PR. In this simple configuration, much less logic resource is needed than that by the RISC core (Table 5). Note that if the FPGA I/O pins become the constraining factor for scale up, wire multiplexing has to be used as suggested in Section 5.

Table 5. Resource usage of NoC elements.

Partition	Block	Cell	Usage	Percentage
Function-based	Four RISC processor cores	Registers	34079	49%
		LUTs	54648	79%
	4x4 routers network	Registers	13442	19%
		LUTs	26635	38%
Region-based	Four processing cores with their own routers	Registers	37526	54%
		LUTs	57481	83%
—	PG & PR	Registers	1361	1%
		LUTs	1624	2%

7. Conclusion

A flexible and scalable multi-FPGA emulation platform that can be utilized to validate and test a complete large scale NoC system has been presented. The platform not only provides abundant logic resources for emulating real processing core behaviors, but also employs high bandwidth, low latency parallel links between FPGAs to directly emulate interconnections in NoCs. The multiple emulation module boards can be interconnected through FPGAs' MGT serial links that would allow the system to be scaled up to a much larger size. A work flow, based on multiple FPGA configurations, with two NoC architecture partition strategies, has also been proposed. As a demonstration, the H.264 decoding application program using a coarse-grained partition scheme has been executed on processor cores connected through a 2x2 mesh-based NoC. The run time speedup for this application is shown to be four orders of magnitude of the software-based simulator. In the next step, we plan to use the proposed emulation platform to carry out extensive performance evaluation of large size NoC architectures in terms of program execution, network communication and power performance under various synthetic and real-life applications using a fine-grained program partition strategy.

Acknowledgements

This research was supported in part by NSF under grant ECCS-0702168, National Natural Science Foundation of China under grant 60873112, and the National High Technology Research and Development Program of China under Grant 2009AA01Z109. The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions.

References

Abdellah-Medjadji, K.-M., Senouci, B., and Petrot, F. (2008), 'Large scale on-chip networks: an accurate multi-FPGA emulation platform', *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, pp.3-9.

Benini, L., and De Micheli, G. (2002), 'Networks on chips: a new SoC paradigm', *Computer*, Vol.35, No.1, pp.70-78.

Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R., Stergiou, S., Benini, L., and De Micheli, G. (2005), 'NoC synthesis flow for customized domain specific multiprocessor systems-on-chip', *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, No.2, pp.113-129.

Chan, J., Parameshwaran, S. (2004), 'NoCGEN: a template based resuse methodology for networks on chip architecture', *17th International Conference on VLSI Design*, pp. 717–720.

Coppola, M., Curaba, S., Grammatikakis, M.D., Locatelli, R., Maruccia, G., and Papariello, F. (2004), 'OCCN: a NoC modeling framework for design exploration', *Journal of Systems Architecture*, Vol.50, No.2-3, pp.129-163.

Dally, W.J. and Towles, B. (2001), 'Route packets, not wires: On-chip interconnection networks', *Proceedings of the 38th Design Automation Conference*, pp.684-689.

Genko, N., Atienza, D., De Micheli, G., and Benini, L. (2007), 'Feature-NoC emulation: a tool and design flow for MPSoC', *IEEE Circuits and Systems Magazine*, Vol.7, No.4, pp.42-51.

Goossens, K., Dielissen, J., and Radulescu, A. (2005), 'Æthereal network on chip: concepts, architectures, and implementations', *IEEE Design & Test*, Vol.22, No.5, pp.414-421.

Jantsch, A., and Tenhunen, H. (2003), *Networks on chip*, Springer.

Kouadri-Mostefaoui, A.M., Senouci, B., and Petrot, F. (2008), 'Large scale on-chip networks: an accurate multi-FPGA emulation platform', *Proc. 11th EUROMICRO Conf. Digital System Design Architectures, Methods and Tools*, pp.3-9.

Krasteva, Y. E., Criado, F., Torre, E., and Riesgo, T. (2008), 'A fast emulation-based NoC prototyping framework', *International Conference on Reconfigurable Computing and FPGAs*, pp.211-216.

Kumar, A., Hansson, A, Huisken, J., and Corporaal, H. (2007), 'An FPGA design flow for reconfigurable network-based multi-processor systems on chip', *Design, Automation & Test in Europe Conference & Exhibition*, pp.1-6.

Lee, K., Lee, S.J., and Yoo, H.J. (2006), 'Low-power network-on-chip for high-performance SoC design', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.14, No.2, pp.148-160.

Liu, J., Shen, M., Zheng, L.R., and Tenhunen, H. (2003), 'System level interconnect design for network-on-chip using interconnect IPs', *Proceedings of the 2003 international workshop on System-level interconnect prediction*, pp.117-124.

Liu, P., Wang, W., Xiao, Z., Lai, L., Teng, Z., Yu, G., Chen, K., Jiang, Z., Zhang, Y., Zhou, J., Cai, W., Zhai, Z., Shi,C., and Yao, Q. (2005), 'MediaSOC: A system-on-chip architecture for multimedia application', *IEEE International Workshop on VLSI Design and Video Technology*, pp.203-206.

Liu, P., Xia, B.., Xiang, C., Wang, X., Wang, W., and Yao, Q. (2009a) 'A networks-on-chip architecture design space exploration – The LIB', Computers and Electrical Engineering, Vol.35, No.6, pp.817-836.

Liu, P., Xiang, C., Wang, X., Xia, B., Liu, Y., Wang, W., and Yao, Q. (2009b), 'A NoC emulation/verification framework', *Sixth International Conference on Information Technology: New Generations*, pp.859-864.

Mahadevan, S., Virk, K., and Madsen, J. (2007), 'ARTS: a systemC-based framework for multiprocessor systems-on-chip modelling', *Design Automation of Embedded Systems*, Vol.11, No.4, pp.285-311.

Marculescu, R., Ogras, U.Y., Peh, L.S., Jerger, N.E., and Hoskote, Y. (2009), 'Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.28, No.1, pp.3-21.

MIPS Corp. (2005). 'MIPS32 Architecture for Programmers Volume II: The MIPS32 Instruction Set', http://www.mips.com/products/architectures/mips32/index.cfm#resources.

Moraes, F., Calazans, N., Mello, A., Mller, L., and Ost, L. (2004), 'HERMES: an infrastructure for low area overhead packet-switching networks on chip', *Integration, the VLSI Journal*, Vol.38, No.1, pp.69-93.

Ogras, U.Y., Marculescu, R., Lee, H.G., Choudhary, P., Marculescu, D., Kaufman, M., and Nelson, P. (2007), 'Challenges and promising results in NoC prototyping using FPGAs', *IEEE Micro*, Vol.27, No.5, pp.86-95.

Xia, B., Wu, K., Xiang, C., Yang, M., Liu, P., Yao, Q. (2010) 'Network interface design based on mutual interface definition,' *International Journal of High Performance Systems Architecture*, (in press)

Xilinx. (2008). 'Virtex-5 FPGA RocketIO GTX Transceiver User Guide'. http://www.xilinx.com/support/documentation/virtex-5.htm.

Xilinx Corp. (2008). 'Virtex-5 FPGA Configuration User Guide'. http://www.xilinx.com/support/documentation/virtex-5.htm.

Yang, G., Yang, M., Yang, Y., and Jiang, Y. (2007), 'On the implementation physical layout of PRDT-based NoCs', *International Conference on Information Technology*, pp.729-733.