International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599654

Building a multi-FPGA-based emulation framework to support networks-on-chip design and verification

Yangfan Liu[a]; Peng Liu[a]; Yingtao Jiang[b]; Mei Yang[b]; Kejun Wu[a]; Weidong Wang[a]; 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

Online publication date: 06 October 2010

To cite this Article Liu, Yangfan , Liu, Peng , Jiang, Yingtao , Yang, Mei , Wu, Kejun , Wang, Weidong and Yao, Qingdong(2010) 'Building a multi-FPGA-based emulation framework to support networks-on-chip design and verification', International Journal of Electronics, 97: 10, 1241 — 1262
To link to this Article: DOI: 10.1080/00207217.2010.512017
URL: http://dx.doi.org/10.1080/00207217.2010.512017

International Journal of Electronics
Vol. 97, No. 10, October 2010, 1241–1262

Building a multi-FPGA-based emulation framework to support networks-on-chip 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*

(*Received 19 July 2010; final version received 27 July 2010*)

In this article, we present a highly scalable, flexible hardware-based network-on-chip (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 field programmable gate array (FPGA) design choices made at both functional and physical levels. At the functional level, a 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 to any one of these parts, only the corresponding FPGA needs to be reconfigured and the rest of the 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 will 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 network-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; Jantsch and Tenhunen 2003).

In a NoC system processing elements, such as processor cores, memories and specialised intellectual property (IP) blocks, exchange data using the network

*Corresponding author. Email: liupeng@zju.edu.cn

ISSN 0020-7217 print/ISSN 1362-3060 online

DOI: 10.1080/00207217.2010.512017
http://www.informaworld.com

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 (Marculescu et al. 2009), such as the topology selection, the design of the network interface and communication protocols, and fault tolerance. This wide range of design choices requires 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 (Chan and Parameshwaran 2004; Coppola et al. 2004; Bertozzi et al. 2005; Mahadevan, Virk, and Madsen 2007) 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, Atienza, De Micheli, and Benini 2007), and thus their applicability for extensively evaluating a complete NoC system with real world data traces is questionable.
- Hardware-based emulation solutions (Moraes, Calazans, Mello, Mller and Ost 2004; Genko et al. 2007; Kumar, Hansson, Huisken, and Corporaal 2007; Ogras et al. 2007; Abdellah-Medjadji et al. 2008; Krasteva et al. 2008), such as ones using FPGAs, on the other hand, can help drastically reduce the system evaluation time without compromising accuracy. One big problem with 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-synthesised and re-implemented on FPGA when there are any architectural and/or logic changes to be made on the system to be emulated.

In this article, 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 minimise the re-synthesis needs when evaluating different NoC designs, we propose a number of solutions in both the 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 4×4 NoC architecture using a 32-bit RISC processor core. Functionally, these five FPGAs are partitioned into two parts: the resource part (four FPGAs) and the network part (one FPGA). One big benefit by doing this is that when certain functionality change needs to be made, only the impacted FPGA needs to re-synthesised. 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 utilises 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 to be readily connected and configured to emulate NoCs with much larger sizes (i.e. 8×8, 16×16, or even larger).

- At the software layer, a real-life application program can be compiled using a customised 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 article is organised 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 modelling. 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 for evaluating 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 the extensive process of physical synthesis. A very high speed integrated circuit hardware description language (VHDL)-based parameterised model for evaluating the performance in mesh NoC topology has been presented (Chan and Parameshwaran 2004). An efficient framework based on an object-oriented C++ library built on top of SystemC has been 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 for emulating NoC architectures for reducing validation time. A switch employed in an XY routing algorithm has been realised 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 4×4 mesh network without an actual processing core is implemented. An FPGA emulation-based NoC prototyping framework has been introduced (Krasteva et al. 2008), which utilised the partial reconfiguration capabilities of an FPGA to reduce re-synthesis time for accelerating emulation process, and a 2×2 mesh was mapped onto an FPGA as a demonstration. A hardware-software emulation framework implemented on an FPGA has been 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 a configurable NoC-based multi-processor SoC has been proposed (Kumar et al. 2007). The complete NoC architecture with three cores, a single router and two network interfaces was emulated on an FPGA.

However, none of the above works is capable of emulating a complete, large scale NoC, as their focus is either on 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 was designed. To connect multiple FPGA boards, high-speed serial links are employed which simulate physical links between routers. The wire-multiplexing technique is applied to overcome resources and pin limitations. A NoC prototype with only the traffic generators and receptors is implemented on two interconnected boards as an illustration. A similar platform built on multiple Altera FPGA boards is reported in Kouadri-Mostefaoui, Senouci, and Petrot (2008). However, the scalability of these designs is limited with a 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. Specifically, this proposed emulation platform will be able to emulate every aspect of a 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 Figure 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. The detailed functionalities that will be supported at this application layer are summarised below.

- Emulate different types of latency critical or data streaming applications' behaviours under multiple traffic patterns, including stochastic traffic in uniform or non-uniform distributions, and real-life traffic.

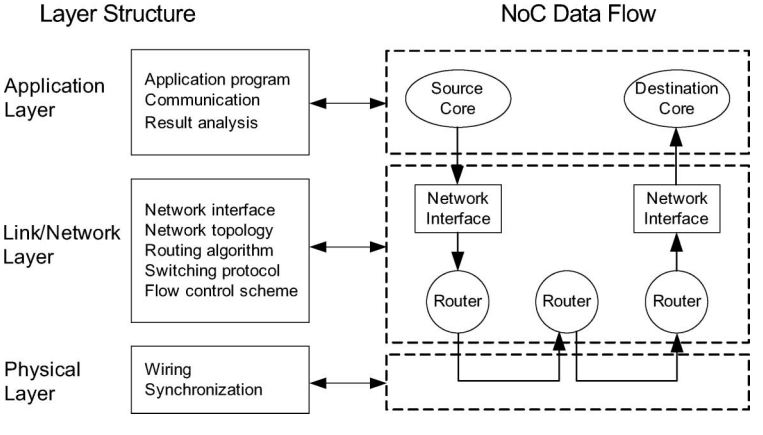

Figure 1. A layered structure for NoC emulation.

- 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 a 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 summarised below.

- Evaluate both regular network topologies, such as mesh, torus, binary tree, PRDT (Yang, Yang, Yang, and Jiang 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 modelling and emulating the behaviours of the physical wires, the processor type, the memory, and the core clock shall be specified for a processing core. Synchronisation at the physical layer is an utmost concern to realise reliable data transfer in a 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 summarised.

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

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, 100 MHz 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 modelled 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-synthesising 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., 8 × 8, 16 × 16 or even larger) by utilising the serial multigigabit transceivers on the 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 synchronisation 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 Figure 2.

In essence, the hardware part harbours the complete NoC architecture to be emulated. The memory supplies data storage space for the processing cores. The controller and monitor unit initialises 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.

The 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, the designer can utilise software tools residing and running on the host computer to configure the hardware emulation platform for different NoC emulation purposes. The software tools contain a USB communicator, an instruction set simulator (ISS), and a personal computer (PC) UART monitor program.

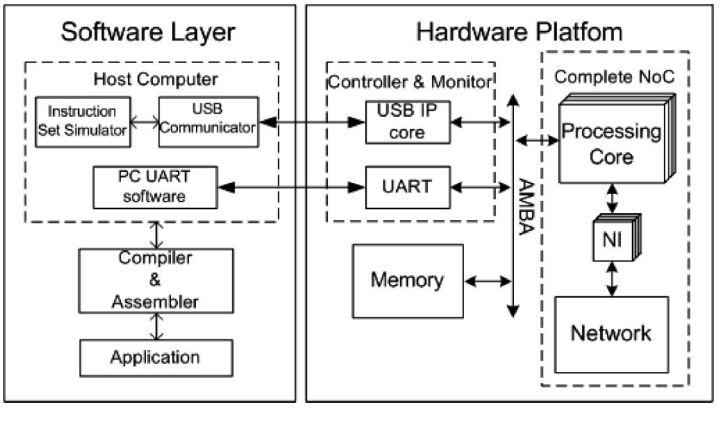

Figure 2. NoC emulation framework.

(1) The USB communicator is used to access USB hardware based on the USB 1.1 protocol. The whole 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 the processor core and collect the result that is stored in memory. This program is loaded during the initialisation and controls the emulation process that has several driver functions.
(2) The ISS is an instruction-accurate simulator whose instructions are compatible with the 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) The PC UART monitor completes the PC UART functions. Any commercially available PC UART software tools can be used to communicate with the hardware platform. The program can initialise UART, print the initial information, and wait for the user's input to configure the 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 Figure 3. The physical wires on the emulation module are organised 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 100 MHz with a total data throughput of 9 Gb/s. The 152-bit parallel LVCMOS interconnection wires are provided between the middle FPGA and the surrounding FPGAs to achieve a total of 15.2 Gb/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.

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 100 Mb/s to 3.75 Gb/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 60 Gb/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, two MGT transceivers are used to connect it with each of the four surrounding FPGAs and the remaining eight MGT transceivers are reserved for connecting to other FPGAs in another module board (off-board extensions).
- For each surrounding FPGA, two MGT transceivers are needed to connect with the middle FPGA, six MGT transceivers to connect with every adjacent surrounding FPGA, and the other two MGT transceivers are reserved for off-board connections.

Each FPGA on a module board has some reserved MGT channels for off-board connections. Each of these off-board MGT channels is connected to a small

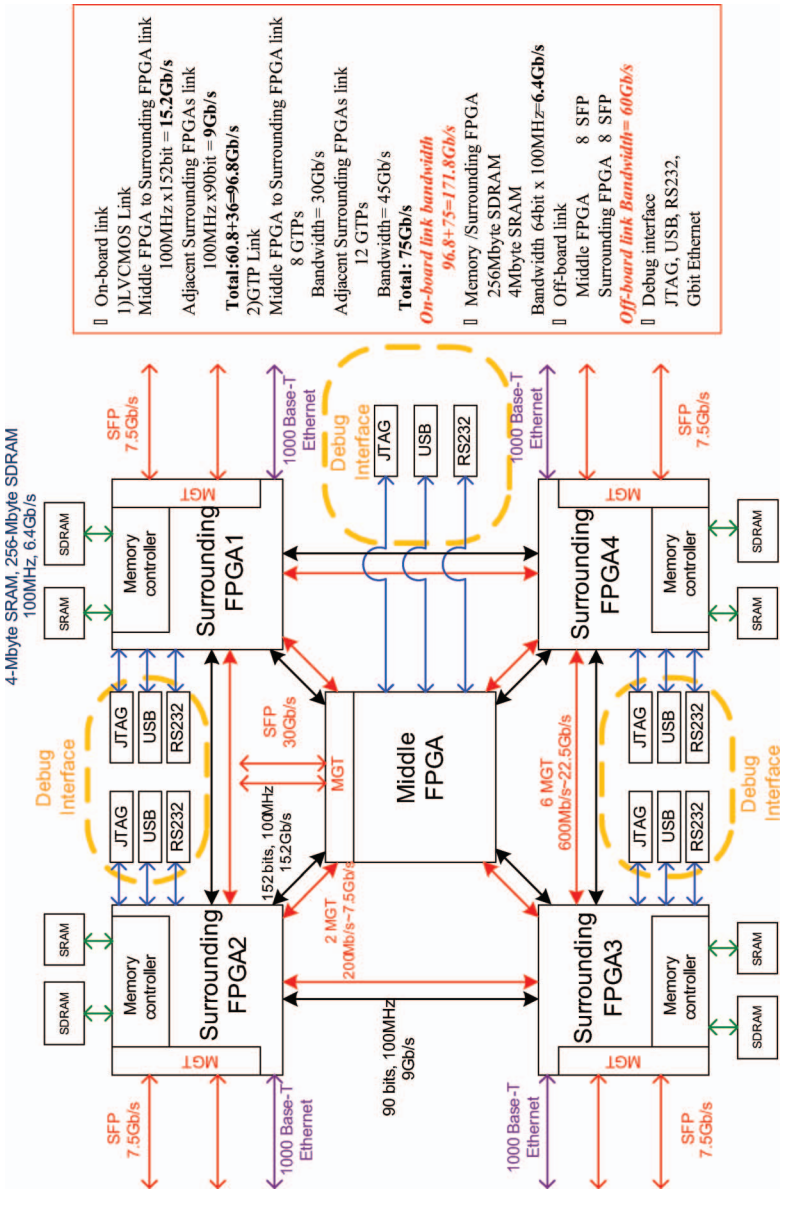

Figure 3. NoC emulation module block diagram.

form-factor pluggable (SFP) connector. The SFP transceiver is designed to support synchronous high speed data communications protocols, such as 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 the SRAM (4 Mbytes) and SDRAM (256 Mbytes) for data storage. Each memory channel runs at 100 MHz with a 32-bit or up to 64-bit data interface. As such, the peak aggregate memory bandwidth for one FPGA can reach 6.4 Gb/s.

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

4.3. *Modelling of wires*

With modern deep submicron and nanofabrication technologies, the interconnect wires, as opposed to the logic cells, dominate system performance in terms of timing delay and power consumption. It is significant to study the NoC architecture in wire modelling at the physical layer (Figure 1). The on-board wires and internal wiring inside an FPGA chip will be modelled 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 modelled as follows (Liu, Shen, Zheng, and Tenhunen 2003).

$$T_{\text{wire}} = 0.39\ rcl^2 \tag{1}$$

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

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

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

$$E_{pkt} = H \cdot \left(E_{\text{queue}} + E_{\text{SF}} + E_{\text{ARB}}\right) + L \cdot E_{\text{Link}} + E_{\text{Queue}} \tag{2}$$

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, Equation (2) is used to estimate the energy consumed by one flit travelling from the sender to the receiver. Estimation of these parameters can be

obtained by synthesis in the Synopsys Power Compiler tool. For example, the power consumption in wires consists of dynamic power and leakage power. The leakage power can be estimated with the Synopsys tool. The dynamic power can be estimated as $0.5\ cV^2fa$, where c is the capacitance of the wire, V is the voltage to be charged to, f is the clock frequency, and a is the number of switching activities per unit time. The switching activities per unit time can be acquired with a simulation tool, such as Modelsim. Thus, the dynamic power can be reported by the power compiler.

Synchronisation in 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 a slower operating speed so that the clocks can be in phase with each other. The other option is that the transmitter sends both the clock and data to the receiver, which can run at a 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 Figure 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 modelled at the register transfer level (RTL) level using HDL, preferably Verilog. Figure 5 shows the synthesis flow to validate a 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 synthesised and mapped onto one FPGA.

The software tools running on the host computer are in charge of downloading the application programs to the FPGAs, setting up all the configuration parameters,

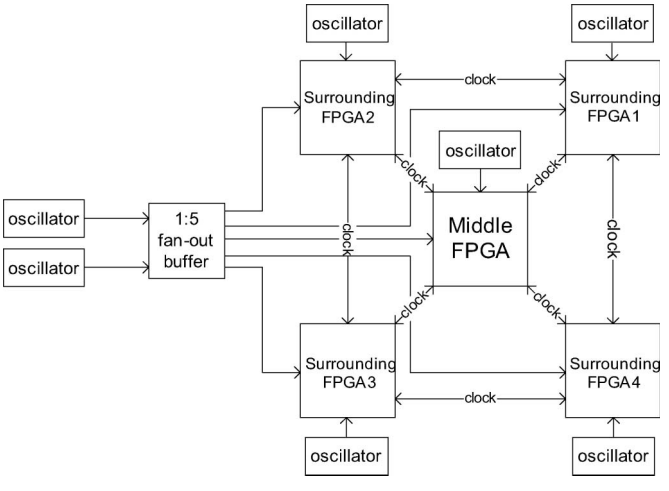

Figure 4. Synchronisation on emulation platform.

controlling the emulation process, and displaying the results collected from the emulation hardware.

The FPGAs on the 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 Figure 6. The devices connected in the JTAG chain are configured one by one according to NoC partition subsets shown in Figure 5.

5.2. *NoC partition*

With five FPGAs available on the emulation module board, the NoC architecture being emulated needs 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.

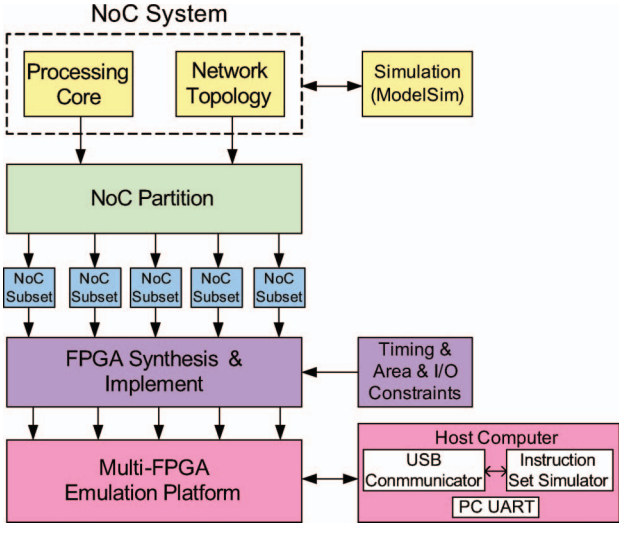

Figure 5. NoC emulation synthesis flow.

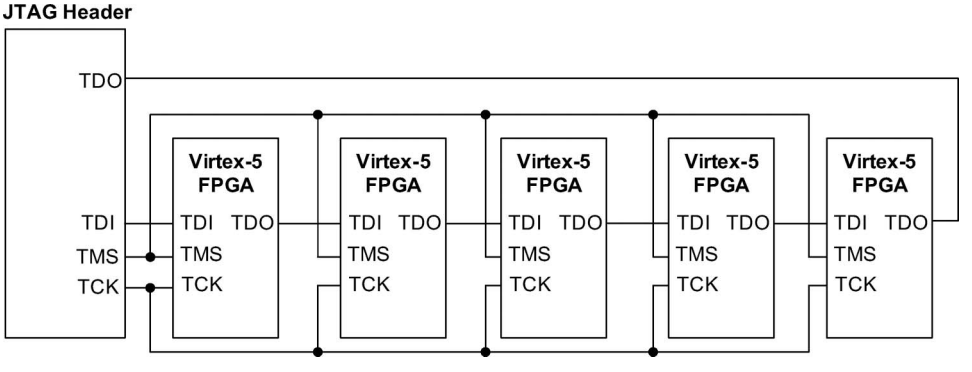

Figure 6. JTAG configuration chain for FPGAs.

5.2.1. 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 chip set 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 utilises 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 resynthesised and the rest of the FPGAs will be left untouched. For example, we can reconfigure only the network FPGA to alter the network topology or routing algorithm while the network interface between the processing core and the router remains 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 4×4 complete NoC systems with 16-bit data bandwidth of each channel between each processing core and its router. Figure 7a illustrates the function-based partition strategy using the example of a 4×4 mesh-based NoC system. The network FPGA is mapped with a 4×4 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 (in one direction) between a processing core and its router consists of 19 bits, including 16 data bits, 2 control bits and 1 clock bit.

5.2.2. 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 regions may be partitioned in various ways to achieve different objectives, such as better utilisation of resources on each FPGA or better utilisation of the parallel wires

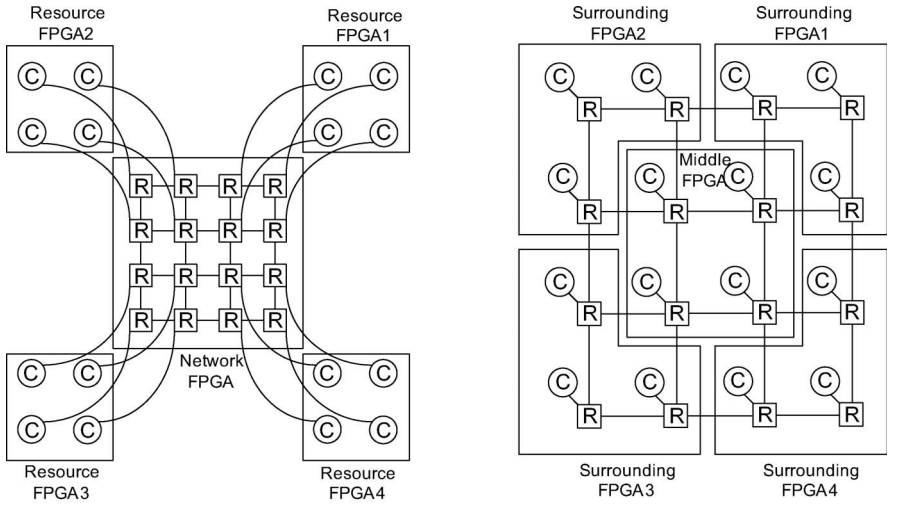

Figure 7. (a) Function-based partition. (b) Region-based partition.

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

on the board. Figure 7b shows one type of region partition for 4 × 4 mesh-based NoCs, which makes better use 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 channels (16 data bits, 2 control bits, 1 clock bit) between routers, for different 4 × 4 NoC topologies based on the partition in Figure 7b. 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, the wire multiplexing technique must be used.

Though wire multiplexing requires extra logic compared with the function-based partition strategy, the benefit of the region-based partition strategy lies in the possibility for supporting larger size NoCs as the resources in the middle FPGA are better utilised under this partition strategy. For instance, by implementing four processing cores and their routers on one FPGA, the emulation module can support

Table 2. Inter-FPGA connections of 4 × 4 NoC based on the partition shown in Figure 7b.

Item	Requested number of wires		
	Mesh	Torus	PRDT
Middle FPGA – surrounding FPGA	76	76	114
Adjacent surrounding FPGAs	38	114	152

up to a 4 × 5 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 2 × 2 NoCs.

5.3. Scalability

For an NoC architecture with many more processing nodes, multiple emulation modules can be built to construct even larger scale emulation systems 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. Figure 8 shows the connection of four 4 × 4 meshes to form an 8 × 8 mesh NoC architecture.

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 the network FPGAs will be used for the connection of adjacent modules to form an 8 × 8 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 or 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 × 12.4 inch). It consists of five Virtex-5 LX110T FPGAs with a total of 16 MB SRAM and 1 GB SDRAM for data storage. Besides, there are five USB, JTAG, and RS232 serial ports for debugging, four 1000 Base-T Ethernet interfaces for communication, and 16 SFP connectors for off-board extension. The picture of the NoC emulation module board is shown in Figure 9.

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 a NoC on the proposed emulation platform given in Table 3. Specifically,

- 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

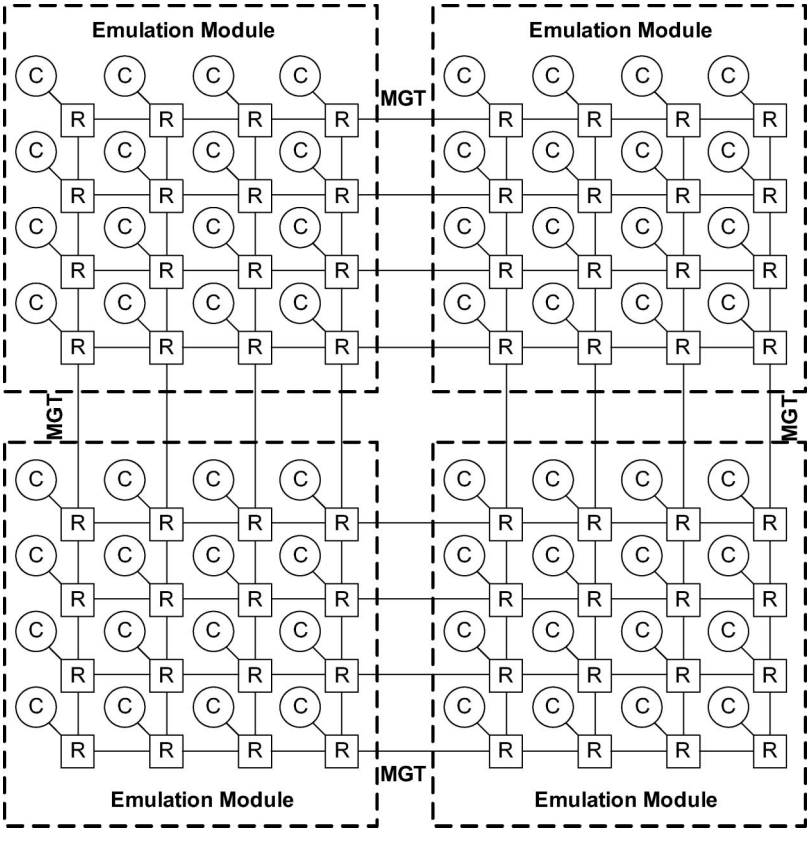

Figure 8. Interconnection of 8 × 8 mesh NoC using four emulation module boards.

types of NIs with different resource dependence: (1) AMBA bus-based NIs which connect the processor to the network through the AMBA bus; (2) direct memory access (DMA)-based NIs which access the processor's on-chip memory through a DMA channel; (3) memory-based NIs which connect memory elements with network that can responds 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.

Various applications can run on the emulated NoC architecture, 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.

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 2 × 2 mesh-based NoC configured

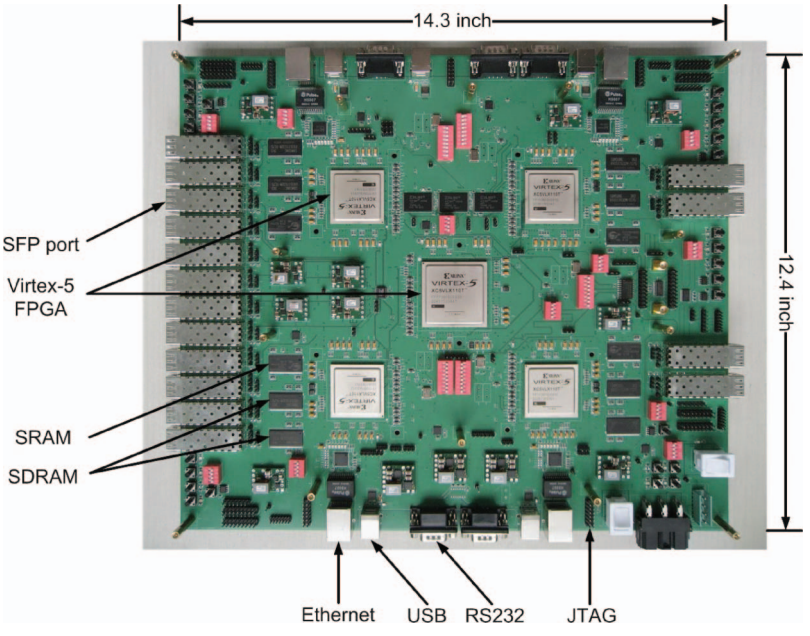

Figure 9. Picture of the NoC emulation module board.

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, 64 KB, 4-way, 16-bytes line size 2-level TLB: 3-entry ITLB and DTLB, 16-entry JTLB ScratchPad memory: 4, 8, 16 KB
	PG and PR	–
NI	Resource dependent	AMBA bus-based, DMA-based, Memory-based
Router	Input/output ports	Mesh with four ports, Torus with five ports, PRDT with nine ports
	Buffer depth	4, 8, 16-entry FIFO
	Routing algorithm	Deterministic, adaptive

with four RISC processor cores and a function-based partition strategy. Each core is a high-performance 32-bit RISC processor compatible with MIPS4Kc (Liu et al. 2005). Figure 10 shows the specific mapping of the 2 × 2 mesh-based NoC architecture. Each resource FPGA is configured with one RISC processor core which is attached to the AMBA bus in order to connect with peripheral memory, communication, and debugging interfaces. One of these four cores, named the control core, is responsible for initialising the NoC communication, starting the other cores, collecting the result data and calculating statistics, while the other three cores (called synergic cores) are only used for computation. The network FPGA is mapped with a 2 × 2 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).

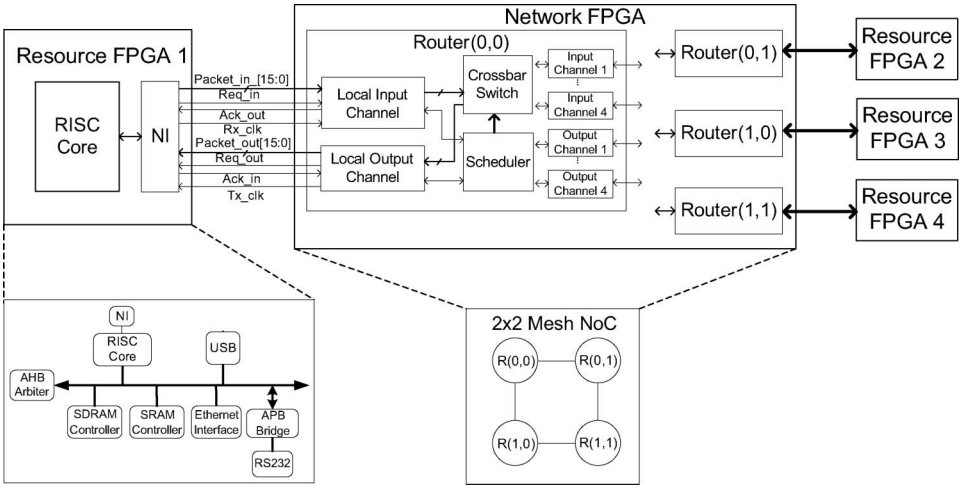

Figure 10. Specific mapping for 2 × 2 NoC architecture based on the partition shown in Figure 7a.

The H.264 decoding program is partitioned into four cores following a coarse-grained program partition strategy at the frame level (Figure 11). The program is first downloaded into the peripheral memory of the control core. The control core is started through an external manual reset signal. During the initialisation phase, the partitioned programs for the synergic cores are loaded to their corresponding cores by the control core though the 2 × 2 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 frames in Quarter Common Intermediate Format (QCIF) format are tested. The application program is partitioned statically in the frame level shown in Figure 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 the reference I-frame or P-frame needs to be transferred to 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 utilisation 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 amount of data transferred between RISC cores is 90,000 words (32-bit). The average latency to transfer one packet (32-bit data) on the 2 × 2 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 the program and balance the payload on each processor core.

As shown in Figure 12, compared with software simulation at the RTL using the Modelsim tool, our FPGA-based emulation platform achieves more than a 10^4 magnitude speedup for the same experiment using the foreman sequence. The speedup result is obtained in the following way. The execution time needed to decode

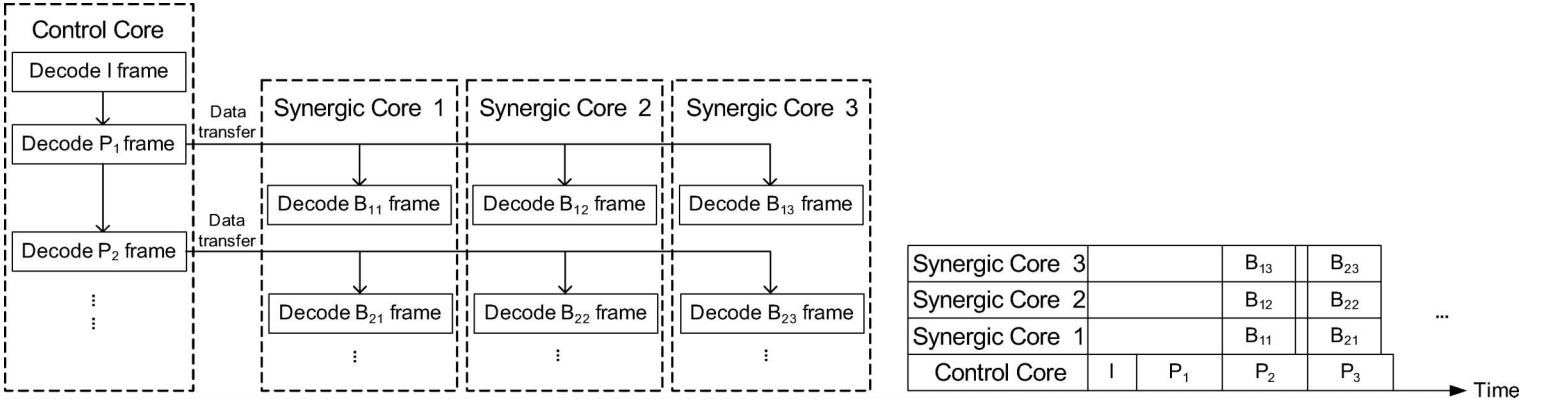

Figure 11. H.264 decoding program parallel partition and execution on a complete 2 × 2 mesh-based NoC system.

the 58-frame sequence on our FPGA-based emulation module operating at 54 MHz is 13 s. It takes 247 min to decode one frame on the software simulator, therefore, the simulation time needed to decode the whole sequence (58 frames) is approximately 239 h. The environment for the software approach is an AMD Opteron 2387 processor running at 2.80 GHz configured with 4 GB memory.

6.3. *Demonstrative evaluation*

Under the function-based partition strategy, as discussed in section 5.2, a complete 4 × 4 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 4 × 4 mesh of routers. The utilisation 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.

Table 4. Execution time of H.264 decoding program on 2 × 2 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

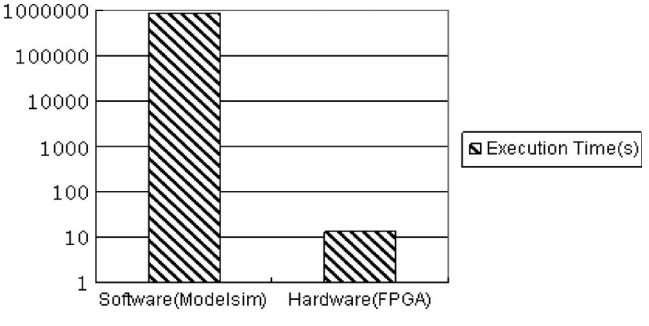

Figure 12. Comparison between hardware emulation and software simulation.

Table 5. Resource usage of NoC elements.

Partition	Block	Cell	Usage	Percentage
Function-based	Four RISC processor cores	Registers	34,079	49
		LUTs	54648	79
	4 × 4 routers network	Registers	13442	19
		LUTs	26635	38
Region-based	Four processing cores with their own routers	Registers	37526	54
		LUTs	57481	83
–	PG and PR	Registers	1361	1
		LUTs	1624	2

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 4 × 5 mesh-based NoC system.

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

7. Conclusion

A flexible and scalable multi-FPGA emulation platform that can be utilised 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 behaviours, 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 the 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 2 × 2 mesh-based NoC. The run time speedup for this application is shown to be four orders of magnitude faster than 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', in *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*, 35, 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 Multi-processor Systems-On-Chip', *IEEE Transactions on Parallel and Distributed Systems*, 16, 113–129.

Chan, J., and Parameshwaran, S. (2004), 'NoCGEN: A Template Based Reuse Methodology for Networks On Chip Architecture', in *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*, 50, 129–163.

Dally, W.J., and Towles, B. (2001), 'Route Packets, Not Wires: On-Chip Interconnection Networks', in *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*, 7, 42–51.

Jantsch, A., and Tenhunen, H. (2003), *Networks On Chip*, New York: Springer.

Kouadri-Mostefaoui, A.M., Senouci, B., and Petrot, F. (2008), 'Large Scale On-Chip Networks: An Accurate Multi-FPGA Emulation Platform', in *Proceedings of the 11th EUROMICRO Conference on 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', in *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', in *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*, 14, 148–160.

Liu, J., Shen, M., Zheng, L.R., and Tenhunen, H. (2003), 'System Level Interconnect Design for Network-On-Chip Using Interconnect IPs', in *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', in *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*, 35, 817–836.

Liu, P., Xiang, C., Wang, X., Xia, B., Liu, Y., Wang, W., and Yao, Q. (2009b), 'A NoC Emulation/Verification Framework', in *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*, 11, 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*, 28, 3–21.

MIPS Corp (2005), 'HMIPS32 Architecture for Programmers Volume II: The MIPS32 Instruction SetH', 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*, 38, 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*, 27, 86–95.

Xia, B., Wu, K., Xiang, C., Yang, M., Liu, P., and Yao, Q. (2010), 'Network Interface Design Based on Mutual Interface Definition', *International Journal of High Performance Systems Architecture*, 2, 168–176.

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', in *International Conference on Information Technology*, pp. 729–733.