A NoC Emulation/Verification Framework

Peng Liu, Chunchang Xiang, Xiaohang Wang, Binjie Xia, Yangfan Liu, Weidong Wang, Qingdong Yao
Department of Information Science and Electronic Engineering
Zhejiang University, Hangzhou 310027, China

Abstract

The emulation and functional validation are essential to assessment of the correctness and performance of networks-on-chip architecture. A flexible hardware/software networks-on-chip open platform (NoCOP) emulation framework is designed and implemented for exploring the on-chip interconnection networks architecture. An instruction set simulator and universal serial bus communicator control and configure the emulation parameters and process that are running on the host computer as active elements in the emulation framework. The experimental results show that the proposed emulation/verification framework can speed up the simulation, preserve the cycle accuracy, and decrease usage of the resource of field programmable gate array.

1. Introduction

With the growing complexity of integrated circuits, new tendencies envisage heterogeneous systems-on-chip (SoC) architectures consisting of complex integrated components of SoCs made of hundreds of cores will not be feasible using a single shared bus or a hierarchy of buses due to their poor scalability with system size and their shared bandwidth among all cores attached to them. To overcome these problems, networks-on-chip (NoC) [2, 4] has been proposed as a promising replacement for buses and dedicated interconnections. However, NoC-based architecture involves new design challenges, such as topology selection, router design, routing algorithm, communication protocols, system tools, and so on [11]. All these challenges require a very time-consuming and error-prone design process of on-chip interconnects to design power-efficient and high-performance NoC [6, 7, 10, 13].

Simulation and functional validation are essential to assessment of the correctness and performance of multi-processor SoC (MPSoC) architectures. After the decisions regarding the communication paradigm, intellectual property (IP) core selection and infrastructure are made; a simulation and emulation of the system are used to validate the design.

In this paper, we present a hardware/software NoC open platform (NoCOP) emulation framework. The NoCOP emulation framework has implemented on a field programmable gate array (FPGA) platform based on hardware accelerate technology [15]. An instruction set simulator (ISS) and universal serial bus (USB) communicator running on the host computer are used as active elements in the emulation software layer to control and configure the emulation parameters and process. The NoCOP emulation framework is able to test actual physical realizations of NoC on silicon up to four orders of magnitude faster than HDL simulator (ModelSim for example) while preserving cycle accuracy.

The remainder of the paper is organized as follows. In section 2, we briefly discuss some related works. In section 3, we describe the NoCOP emulation framework. In section 4 and 5, we describe the hardware and software layer of the NoCOP emulation framework. In section 6, we present experimental results and conclude the paper in section 7.

2. Related work

A number of cycle-accurate simulation frameworks in VHDL or SystemC have been proposed in the literature [3, 8, 12]. Though these simulators are flexible to perform NoC design space exploration, they cannot use real-life traces to extensively evaluate the entire NoC system at fast speed. The major issue with simulation-based approaches is the tradeoff between the level of implementation detail and the slow simulation time.

In [6, 7], the authors present a NoC emulation platform implemented on FPGA. The NoC hardware platform is implemented on a Virtex-II FPGA, which consists of network injection, reception and controller components. The processor core PowerPC is integrated into the hardware platform that functions as a controller. Instead of merely being the platform where the circuit is prototyped, the method can speed up functional validation
and add flexibility to the NoC configuration exploration. In [9], an FPGA emulation-based fast networks-on-chip prototyping framework is presented, which the main goal is tried to fast the synthesis process by partial reconfiguration of hard cores. However, these methods [6, 7, 9] have one major drawback, that need a processor core in the hardware to control and monitor the network at the cost of the limited resource of FPGA.

In [13], the simulator on the FPGA is implemented as a homogeneous wormhole switching network with virtual channel flow control with a torus topology. The system consists of a SoC board and an FPGA board. In [1], the scalable multi-FPGA platform is designed for NoCs emulation and debugging. The platform is constructed by XUP VirtexII Pro board using the high speed serial links to connect multiple FPGA boards. In [10], four real applications mapped into a NoC and prototyped in an FPGA are presented. These methods [1, 10, 13] lack flexibility and suffer the limited resources of FPGA in the case of data extraction.

In order to solve these problems and keep the emulation flexibility, the proposed NoC emulation framework consists of an ISS and USB communicator running on host computer, and hardware platform. The ISS and USB communicator are used as active elements to control and configure parameters during the emulation process. The hardware platform has been used to validate the NoC design.

3. NoC emulation of framework

The emulation framework of NoCOP is a combined hardware/software platform. Figure 1 shows the block diagram of emulation framework which can be divided into two layers: hardware layer and software layer.

3.1. Hardware layer

Hardware layer of NoCOP consists of four elements to emulate NoCs: network to be emulated, packet generator and packet receptors, packet controller and result analysis (PCRA) module, and interface to the host computer. The hardware layer of NoCOP contains four parts.

1) Network to be emulated: The routers and network interfaces (NIs) are organized into different topologies. The router is designed in parameter in terms of various bit-width, flow control, routing algorithm, and arbiter scheme, etc.

2) Packet generators (PGs) and packet receptors (PRs): Every node in the network has one PG and one PR. PGs are controlled by packet controller. PGs generate packets with designated control information, such as the source, destination, packet length, packet number, and interval cycle. PRs receive packets from the network.

3) Packet controller and result analysis (PCRA) module: PCRA unit initializes the network communication pattern and analyzes the network performance in terms of latency.

4) Interface between the host computer and NoC: The designer can configure network through the AMBA bus. The AHB bus and APB bus were implemented on platform. AHB is a high performance bus, which has an arbiter, a USB master module, a slave module (APB Bridge). APB is designed to connect peripheries. The UART controller and PCRA module are connected on the APB bus. The USB module is connect with the AHB bus.

3.2. Software layer

Software layer is used to configure the emulation parameters and control the emulation process of NoC architecture. The software layer of NoCOP contains three parts.

1) USB software, which is used to access USB hardware based on USB 1.1 protocol. All address space of AMBA bus can be accessed by the USB software.

2) Instruction set simulator (ISS), which is an instruction set simulator compatible with MIPS 32 instruction set. The ISS connects with USB software. It can access the AMBA bus through USB software. The ISS directly access AMBA APB bus when entering the special memory space (now the special memory address with most significant 8-bit of 0x1c). The processor IP soft core can replace the ISS for emulation that is similar to Genko’s method [7].

Figure 1. NoCOP emulation framework
3) Personal computer (PC) universal asynchronous receiver/transmitter (UART) software, which completes PC UART functions. We can use the commercial PC UART software tools to communicate with NoCOP hardware.

4. Hardware layer

The hardware layer is designed in a modular way, which consists of three modules relating to the network packet: packet generators, packet receptors, and packet controller and result analysis module.

4.1. Packet generator

PGs consist of three main parts, that are linear feedback shift register (LFSR) random generator, configure module, and packet generate controller. Figure 2 illustrates the block diagram of PGs. LFSR random generator is used to generate pseudo random number. This number is used for destination node number. The Xilinx FPGA series have an IP core named LFSR that can generate the LFSR pseudo random number. We use this LFSR IP core now for simplicity. For detail information about LFSR generator, please refer to the Xilinx datasheet (http://www.xilinx.com).

Configure module receives the configure information from packet control and result analysis module. The configuration information includes PGs startup, destination node, packet length, packet numbers, and transmitting interval. The configure unit has three counters: packet length counter, packet number counter, and interval timer. The packet generate controller based on these registers content generates the control signals of network.

Figure 2. Block diagram of network packet generator

Figure 3. State diagram for the packet generator controller

The packet generate controller has been designed into a four-state finite state machine which is illustrated in Figure 3.

1) IDLE is the idle state. When the cfg_start is asserted, the machine starts the PG, exits idle state and enters PKG_TRANS.

2) PKG_TRANS controls the PG injecting packets into network. After finished packet transfer, the PKG_TRANS state has alternatives to the next state. The PKG_TRANS changes to WAIT or STOP resting on the packet number counter.

3) WAIT is a waiting next-transfer state which a packet has finished the transfer but the whole packets have not been sent to destination. When the interval time counter reaches the predefined waiting cycles, the machine will roll back to PKG_TRANS state to transfer the next packet.

4) STOP is the end state of a transfer. The STOP state will roll back to IDLE after cfg_start is unasserted.

When a flit is injected into the network, the PG embeds a time stamp in its body. This time stamp is used by PR to calculate the transfer cycles.

4.2. Packet receptors

The packet receptors receive flits from the network. Every flit has a time stamp in its flit body that contains the time when the flit is injected into network. The PR picks up the time stamp information and subtracts the time when it receives the flit and the time stamp in the received flit body. The result indicates the flit transfer cycles in network. The time stamp is 32-bit long.
4.3. Packet controller and result analysis

PCRA module fulfils two functions: 1) controls all packet generators to generate packets; 2) keeps an account of the total number of flits that the PRs received, and the total cycles of these flits flying in network. Figure 4 illustrates the PCRA, which consists of three parts, APB bus interface connecting to APB bus, packet controller, and mean cycle statistics. The mean cycle statistics have been used to calculate the average message latency of flits transmitting in the network. The packet controller includes two packet injection control registers, a system control register, a system status register, and a latency register. Programmers use these registers to configure and monitor the network.

5. Software layer

The three tools: USB communicator, ISS, and UART monitor are located in software layer. The USB communicator has been used to communicate with USB core in slave mode. The interface of the USB communicator is illustrated in Figure 5. We should select “IPC only” to make USB communicator connect with the ISS. If the ISS is not used, the USB communicator can use to download the program to FPGA. The application program is simulated on the ISS. The simulator will access the APB bus through USB communicator by specified address. The UART monitor communicates between PC and hardware layer.

To facilitate the NoC emulation, the designers use an application program running on the ISS to log and control the network. This program completes initialization and controls the emulation process that has several driver functions. The flow chart of this program is shown in Figure 6. The program can initialize UART, print initial information, wait for user’s input, and choose transfer mode.

Three different transfer modes have been supported in the software layer.
1) In script mode, a script file to configure the network is read, and then the results are displayed on the host computer.
2) Some predefined communication patterns work in internal mode, which include butterfly, matrix transpose, bit reversal, and so on.
3) The user can configure all the parameters of the packet generator in user mode.
If choosing the script mode, we need to load a file through UART with some designated format. The program will configure the network according to the script file and display the result. If choosing the internal mode, the network will be automatically configured using internal parameters, started a transfer, and displayed the result.

If choosing the user mode, the network will wait for next input to decide what to do. You have three choices: “configure node”, “startup configuration”, or “display”. If you choose “configure node”, you need to firstly choose a source node, then you can configure the destination, packet number, packet length, and packet interval time of the source node. If you choose “startup configuration”, you need to type all the source nodes which need to start, then it will start the network. If you choose “display result”, the node configuration information and the mean cycle can be displayed.

6. Experimental results

The NoCOP FPGA board [14] is illustrated in Figure 7. The central part of the board is a Xilinx Virtex-4 LX160. The synthesis tool of FPGA is Synplify Pro8.1. The FPGA resource usage of building blocks is listed in Table 1. The FPGA delay information of building blocks is listed in Table 2. The NoCOP emulation system can run on 110MHz.

We use the typical workload to evaluate the network latency. The workload model is defined by three parameters: distribution of destinations, injection rate, and message length. Figure 8 shows the average message latency verus injection rate on a 4 x 4 mesh when using the deterministic, partially adaptive, and fully adaptive routing algorithms under different workloads. We use the distributions to evaluate the network latency that consist of uniform, hot spot, bit reversal, and matrix transpose.

As can be seen from the Figure 8, the router uses adaptive routing algorithm in the case of a heavily network load and non-uniform distribution, the average latency decreases more than 50%. When the router uses uniform distribution in the case of high injection rate, the deterministic routing algorithm has better performance than the adaptive routing algorithms.

7. Conclusion

By introducing the NoCOP, the designers can use it to validate and test different NoC implementations in MPSoC. A method for overcoming emulation system drawbacks derived form inflexibility and lack of debugger software has been presented. The emulator gives us detailed insights in the behavior of our building blocks in the network. The software tool can control the emulation process, and the hardware can track the packet information. As a demonstration of the approach, the NoCOP framework has been presented and mapped on our designed platform.

We plan to develop multi-FPGA system and powerful tools using the proposed method to explore the MPSoC architectures design space.

### Table 1. Area requirements of building blocks

<table>
<thead>
<tr>
<th>Building blocks</th>
<th>Register</th>
<th>LUTs</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB IP</td>
<td>991</td>
<td>1054</td>
<td>959</td>
</tr>
<tr>
<td>Router_D</td>
<td>625</td>
<td>2395</td>
<td>1489</td>
</tr>
<tr>
<td>Router_P</td>
<td>649</td>
<td>2447</td>
<td>1522</td>
</tr>
<tr>
<td>Router_A</td>
<td>668</td>
<td>2447</td>
<td>1526</td>
</tr>
<tr>
<td>NI</td>
<td>738</td>
<td>900</td>
<td>792</td>
</tr>
<tr>
<td>PG</td>
<td>53</td>
<td>88</td>
<td>56</td>
</tr>
<tr>
<td>RG</td>
<td>53</td>
<td>117</td>
<td>62</td>
</tr>
<tr>
<td>PCRA</td>
<td>2497</td>
<td>1332</td>
<td>1607</td>
</tr>
</tbody>
</table>

Note: The router has five ports, that each port has two virtual channels. The physical channel is 75 bits wide. The configuration of routing algorithms in router consist of deterministic routing, west-first routing algorithm, and Duato’s protocol [5], which named as router_D, router_P, and router_A, respectively.

### Table 2. Delay estimation of building blocks

<table>
<thead>
<tr>
<th>Building blocks</th>
<th>Delay /ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router_D</td>
<td>7.602</td>
</tr>
<tr>
<td>Router_P</td>
<td>8.345</td>
</tr>
<tr>
<td>Router_A</td>
<td>8.746</td>
</tr>
<tr>
<td>NI</td>
<td>6.220</td>
</tr>
<tr>
<td>PG</td>
<td>4.085</td>
</tr>
<tr>
<td>RG</td>
<td>2.253</td>
</tr>
<tr>
<td>PCRA</td>
<td>3.499</td>
</tr>
</tbody>
</table>

Figure 7. NoCOP FPGA board
Figure 8. Comparison of routing algorithms

References


