Transaction Level Model Simulator for NoC-based MPSoC Platform

Seungbeom Lee, Sung-Rok Yoon, Jin Lee, Min Li Huang and Sin-Chong Park
School of Engineering
Information and Communications University
119, Munji-ro, Yuseong-gu, Daejeon, 305-732,
Korea

Abstract: - Network-on-Chip (NoC) based Multi-Processor System-on-Chip (MPSoC) architecture is a promising SoC design solution, offering high computational power with lots of flexibilities. However, finding the optimal MPSoC architecture configuration remains an enormous challenge due to its high structural complexity and functional diversity. In this paper, we introduce a Transaction level NoC SIMulator (TraNSIM) to evaluate the performance of NoC based MPSoC architecture in the early design stage. Focusing on List Sphere Decoder (LSD) as a case study, we present the methodology to find an optimum multi-processor architecture, and demonstrate the performance variation with respect to different NoC topologies by using TranSIM.

Key-Words: - MPSoC, NoC, Transaction level model, SystemC.

1 Introduction

With the growing complexity in embedded system, decreasing time-to-market and a multitude of application standards, MPSoC has garnered much attention from the industry as single processor no longer suffice. Full exploration of MPSoC requires the paradigm shift from conventional computation-centric architecture to communication-centric. In recent years, this trend is exemplified by the increasing numbers of Network-on-Chip (NoC) architectures proposed for multi-processor based SoC integration, overcoming the scalability limits of traditional state-of-the-art shared busses [1].

However, reliable performance estimation of an MPSoC system remains an enormous challenge for MPSoC application designers. High structural complexity and functional diversified MPSoCs with random traffic transferring across heterogeneous PCs remain an issue to be solved. Without the help of proper tools, proposed solutions are most often sub-optimal and inefficient. Thus, this paper presents a transaction level approach to model the NoC architecture using SystemC that enables effective performance evaluation of an application specific MPSoC in various configurations; using a well-known wireless communication channel detector called List Sphere Decoder (LSD) as our case study. This methodology enables design space exploration in the early stage of SoC design flow, requiring only two kinds of information: approximated processing time for each PCs and data transaction among PCs.

Previous works including synthesizable NoC models that can be modularized and easily reconfigured are found in [2]. The SystemC based transaction level model proposed in this paper is not synthesizable, but it requires lower implementation effort while providing the ease of extending new protocols onto the MPSoC platform. The authors of [3] defined the handshaking based communication protocol for NoC using SystemC that only implements part of the NoC architecture. This work includes the entire NoC architecture and defines simpler and more efficient interface. The authors of [6] show the methodology to analyze MPSoCs architecture at transaction level, using bus as communications architecture. In [5], traces-based transaction level simulation is used to evaluate system performance for bused –based SoC architecture.

The rest of this paper is organized as follow: S Section 2 presents TraNSIM architecture and the know-how to model NoC architecture in application specific MPSoC system using SystemC. We chose LSD as our case study to demonstrate the proposed scheme and the corresponding simulation results are shown in Section 3. Our conclusion for this work is drawn in Section 4.

2. TraNSIM Model using SystemC

We introduce Transaction level Network-on-Chip SIMulator (TraNSIM) modeled using SystemC to evaluate the performance of MPSoC architecture with NoC communication architecture. Fig. 1 shows the

overview of TraNSIM. TranNSIM is composed of TraNSIM Library such as PC and NoC model, and NoC integrator (NoCI). NoCI is capable of generating a top level code named "top.cpp" for SystemC simulation and a routing table called "routing_table.dat" for switch router from input files ("topology.dat" and "tran_info.dat"), physical parameters (link width, buffer depth, number of switch and number of PC), and protocol parameters (route scheme and priority control scheme).

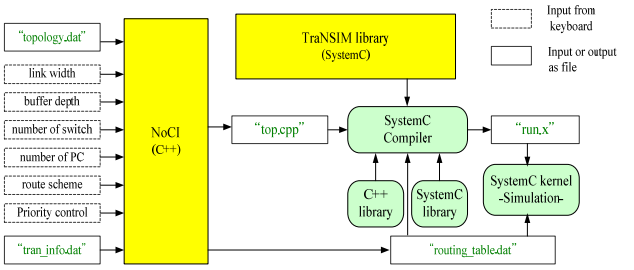


Fig. 1. Overview of TraNSIM

The physical parameters and protocols are defined in "top.cpp" file by using preprocessing commands of C++ program. The code syntax for instantiating switch or PC module is generated from input information such as 'number of switch' or 'number of PC'. The file, "topology.dat" is a table whose row and column size are the addition of the number of switches and PCs. Each position in the table represents the connecting condition between two modules. Based on the topology information provided, port binding for two modules is done in the automatically generated "top.cpp" file, as shown in the code below.

```
#define _WIDTH 64 //channel width
#define _ADDR0 3 //address for ex_appl
. . .
// make a channel instance
sc_fifo<sc_uint<_WIDTH> > pe2net;
. . .
// make a switch instance
swi4x4 swi0("switch0");
. . .
// make a resource instance
ex_appl dsp0("dsp0", _ADDR0);
// port binding
swi0.in_buff[0](pe2net);
. . .
```

NoCI also generates "routing_table.dat" for routing information. The row of this table represents the source PC, and the column stands for destination PC. The value in table is the number of output port of switch. The file, "trans_info.dat", is a table whose value is the transaction size between the source PC and destination PC. This row and column represents the source PC and destination PC, respectively. After "top.cpp" is compiled with TraNSIM library, "routing_table.dat", and "trans_info.dat", the simulation starts to run through SystemC kernel. The example files of input and output are shown in Section 3, using our LSD case study as an example. TraNSIM generate latency and throughput performance, which are the crucial performance metrics for wireless communication.

In order to model MPSoC with NoC communication architecture (TraNSIM library), basic components are modeled as in Fig. 2. Switch, NI and PC are implemented as module (SC_MODULE), respectively. The link between PC and NI module is implemented as tlm_transport_if [9] channel, and tlm_fifo [9] channel is used for the link between NI and Switch module. The resource and network are separated by independent modules and tlm_transport_if channel provides bidirectional blocking interface to access network switches.

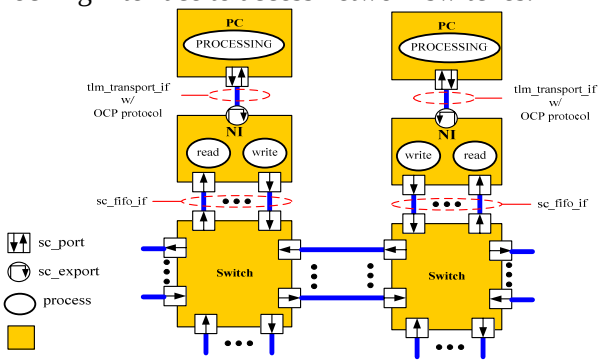


Fig. 2. Overview of TraNSIM Library using SystemC

PC and NI Modeling: PC and NI are implemented in a module, respectively. The PC module declares the transport method [9]. The transport method is defined in the NI module. The PC has a process namely PROCESSING, using the SC_THREAD method in SystemC. Transport method in PROCESS is called to generate write/read transaction. Read process of NI is woke up by the transport method call with read contents and starts to read a packet. When packet reading is completed, PROCESSING process waits for a pre-defined amount of time, which is equivalent to the processing delay of the actual PC. After which this process generates an event for Write process of NI by calling transport method with write contents. Write process in NI sends a packet to the network.

Switch Modeling: Switch contains a single process which is triggered by clock edge event coming from an external clock source. In order to make reconfigurable

and extendible switch, protocols such as switching, route, and priority control are implemented by independent C++ functions such as switching, route, priority control, crossbar, and flow control function. Each function call among several functions is controlled by control variables such as "pkt_header_on". For example, a route function starts to operate when the control variable, "pkt_header_on", is 1 which means that there are packet headers in coming flit. A switching function reads and stores incoming flits from the ports when there are no previous flit remaining in the switch and the connecting FIFO is not empty. A route function decides the output port by comparing and referencing the destination address in a packet header to a routing table. A priority control function assigns priority to each input port. A crossbar function links the incoming flits to the corresponding output ports. The flow control function treats flits which are not yet assigned to output ports due to contention.

3. Case Study: List Sphere Decoder

3.1. Overview of List Sphere Decoder

The detailed algorithm description for LSD is well described in [4], [7-8]. We consider the pipelined architecture LSD proposed in [7].

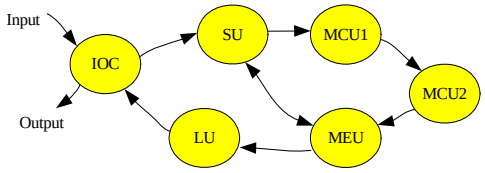


Fig. 3. Functional units of pipelined LSD

Fig. 3 shows the functional Units (FUs) of this architecture. We assume the number of transmitter antennas and receiver antennas is 4 respectively, 64 QAM modulation is used, and the number of candidates is 16. IOC controls sequence of inputs and outputs. MCU1 calculates common factor for Partial Euclidean Distance (PED). MCU2 calculates PEDs for all child nodes. MEU selects the next mother node through SE enumeration. SU is the storage for U-matrix, PEDs, and node history for node pruning purposes. LU generates the list of candidates and updates the radius value.

3.2. Transaction Analysis of LSD

The input of LSD is a 4×4 R matrix and a 4×1 QHY matrix [7-8] at the first clock cycle. Because R matrix is an upper triangular matrix, number of non-zero elements in R matrix is 10. When floating value of the input is assumed to be expressed by 16 bits and complex-valued operation is assumed, input transaction size of IOC from input during the first clock cycle is (16×2×10 + 16×2×4)=448 bits. Other transaction size can be calculated in the same way. From the LSD algorithm, transaction size and the average processing cycles of each FU are obtained and shown in Table 1. Cproc, Dread, and Dwrite are processing cycle, data size to be read, and data size to write, respectively.

Table 1. Average processing cycle and transaction size (bits) of each FU

FU	Cproc (cycle)	Dread (source FU)	Dwrite (destination FU)
IOC	5	448 (Input), 648 (LU)	640 (Output), 456 (SU)
MCU1	22, 31, 43, 56 for each tree depth	72, 108, 136, 168 (SU)	56 (MCU2)
MCU2	960	56 (MCU1)	22x64+8=1416 (MEU)
MEU	154	1416(MCU2)	1422, 8(SU), 9~649(LU)
SU	97, 576	456 (IOC), 1422 (MEU)	72,108,136,168 (MCU1), 16(MEU)
LU	328	9~649(MEU)	48 (IOC)

3.3. Mapping to MPSoC Architecture

Before applying LSD to NoC, multi-processor architecture exploration is performed in order to determine the number of PCs and the assignment of FUs to PCs. In this case we consider three mapped MPSoC architectures as shown in Fig. 4. Latency Minimization (LM) is the architecture that minimizes latency by mapping MCU2 to multiple processors. Throughput Maximization(TM) maps the entire FUs on to a single processing core (PC2). Balancing Load (BL) balances the processing load of all PCs. It maps SU, MCU1, MEU and LU to one processing core (PC2), then decides the number of PC2s similar to PC(M+2) for MCU2.

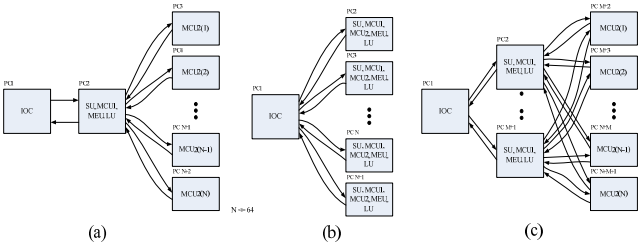
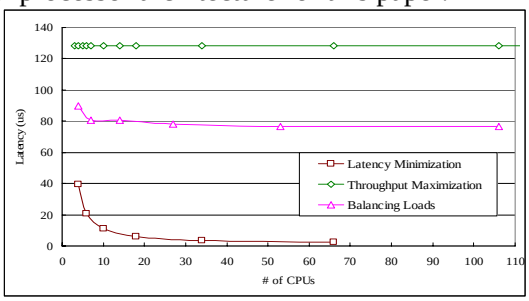


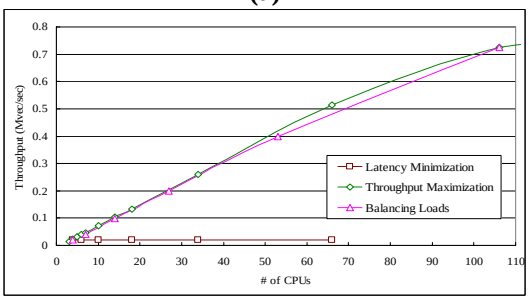
Fig. 4. MPSoC architecture (a) Latency Minimization (b) Throughput Maximization (c) Balancing loads

Fig. 5 shows the performance of multi-processor architecture in terms of latency and throughput. The LM architecture is not efficient in terms of throughput because the processing load is different among PCs. TM shows the best efficiency in throughput

performance but it has no gain in latency. BL balances the processing load of all PCs. BL shows almost similar throughput performance as in TM but with gain in latency. We thus choose BL as the multi-processor architecture for this paper.



(a)



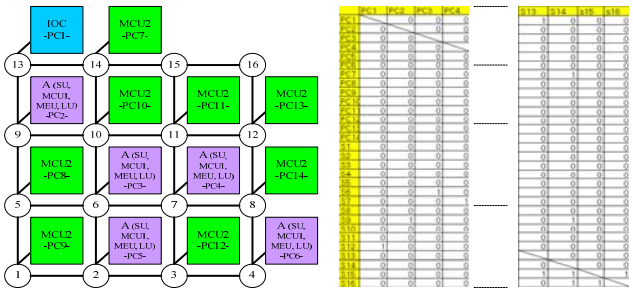
(b)

Fig. 5. Latency and throughput of multi-processor architecture (a) latency (b) throughput

3.4. Performance Evaluation

BL with 14 PCs is mapped to NoC and a 4x4 mesh topology is chosen as interconnection architecture. PCs are then arbitrarily mapped to NoC. Next, NMAP mapping algorithm [2] is applied. NMAP finds the mapping case which minimizes the communication cost. Last but not least, the topology and mapping are changed to minimize the communication cost. As a result, SPIN and mesh topology which provides the minimum communication cost is chosen in this paper.

Input of TraNSIM: The Link width is separated into two cases, 32bits and 64bits. The buffer depth is 64. During the simulation, TraNSIM checks and notifies whether the buffer is overloaded or not. The number of switch is 16, the number of PCs is 14, route scheme uses deterministic route scheme, and priority control scheme is random. As mentioned previously, "topology.dat" file decides the NoC topology. For example, mesh topology with 14 PCs is shown in Fig. 6 (a) and the converted "topology.data" file is shown in Fig. 6 (b). The column or row size of table in "topology.dat" is 30 because there are 14 PCs and 16 switches. The file, "trans_info.txt" can be extracted from Table 1 and is shown in Fig 7 (a). From these input, "top.cpp" and "route_talbe.dat" file are automatically generated by NoCI.



(a) (b)

Fig. 6. Mesh topology (a) schematic (b) topology.dat

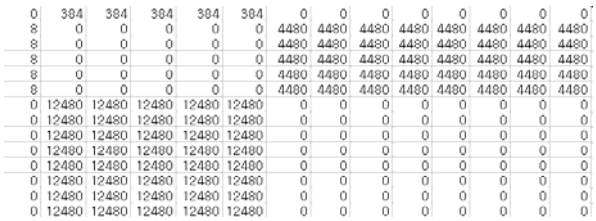
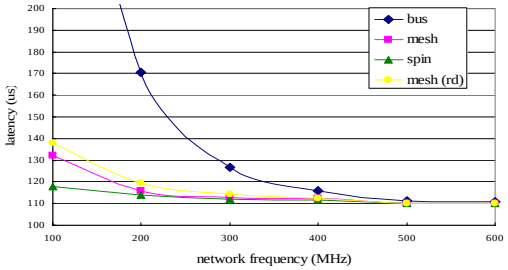
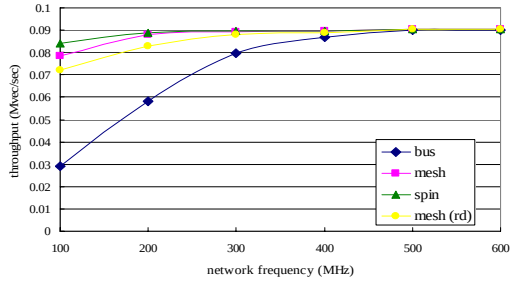


Fig. 7. File example of "trasn_info.txt"



(a)



(b)

Fig. 8. Latency and throughput of LSD with NoC architecture (rd : Random Mapping) (a) latency (b) throughput

Evaluation Results: Fig. 8 shows the latency and throughput for LSD with 14 PCs. Comparing NoC with bus, NoC requires 200 MHz network frequency to achieve peak performance with bus requirement of 400 MHz. When the latency performance differentiation is compared between different mapping

cases in 100 MHz network frequency, mesh with NMAP shows 5 us and 15us less latency when compared with mesh with random mapping and SPIN topology, respectively. Note that when the computation time is only considered, latency is 80 us. When the communication time is added as shown in Fig. 9, latency is saturated at about 110 us. This means that the communication performance can be increased up to 40% according to the configuration of NoC architecture.

4. Conclusion

In order to implement a specific application on a flexible and scalable MPSoC architecture, this work presents the methodology to evaluate various NoC architectures at the early stage of SoC design. The Transaction Level NoC SIMulator – TraNSIM – was specially designed and built to support this methodology. It is a fast and reconfigurable simulator, well-suited for the purpose of finding an optimal NoC based MPSoC. A case study which applies a List Sphere Decoder to an MPSoC architecture is introduced. The design exploration and tradeoffs between different multi-processor architectures and the evaluation of communication architectures are clearly shown in this work.

References:

[1] Benini, L., Micheli, G. D., "Networks on Chips: Technology and Tools", 1st edn. Elsvier Inc., 2006.

[2] Bertozzi, D., Jalabert, A., Murail, S., Tamhankar, R., et al., " NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip", IEEE Trans. On Parallel and Distributed Systems, Vol. 16, No. 2, 2005, pp.113 - 129

[3] Chan J., Parameswaran S., "NoCGEN:a template based reuse methodology for Networks On Chip architecture", Pro. International Conference on VLSI Design, 2004, pp.717-720

[4] Burg, A., Borgmann, M., Wenk, M., Zellweger, M., et al., "VLSI Implementation of MIMO Detection Using the Sphere Decoding Algorithm", IEEE Jnl. Of Solid-State Circuits, Vol. 40, Issue 7, 2005, pp.1566-1577

[5] Wild, T., Herkersdorf, A., Ohlendorf, R., "Performance Evaluation for System-on-Chip Architectures using Trace-based Transaction Level Simulation", Proc. Design, Automation and Test in Europe 2006, Vol. 1, 2006, pp.1 - 6

[6] Lee, J., Park, S.-C., "Methodology of High-Level Transaction Level Modeling using 802.11 PHY Example", IEICE Trans. Information and Systems. Vol. E88-D, No.7, 2005.

[7] Lee, J., Park, S.-C., Park, S., "A pipelined VLSI architecture for a list sphere decoder", Proc. International Symposium on Circuits and Systems 2006, 2006.

[8] Lee, J., Park, S., Zhang, Y., Parhi, K. K., Park. S.-C., "Implementation Issues of a List Sphere Decoder", Proc. IEEE International Conference on Acoustics, Speech and Signal Processing 2006. Vol. 3, 2006, pp. III-996 - III-999

[9] http://www.systemc.org