

A Hybrid NoC-based MPSoC Simulator

Linxiang Shi^a, Zhe Zhang^b

School of Computer and Information,
Shanghai Second Polytechnic University
Shanghai, China

^ashilixiang@yahoo.com.cn, ^bzhangzhe@it.sspu.cn

Keywords: Network-on-Chip, bus, instruction-set simulator.

Abstract. A hybrid MPSoC (Multi-Processor SoC) simulator was proposed in this paper. The simulator provides statistical traffic model and behavior-level model for computation, along with cycle-accurate model on basis of ISS (Instruction-Set Simulator) including ARM and TI DSP. The simulator also provides NoC (Network-on-Chip) and traditional bus for on-chip communication and interconnection. As shown in the case study, the proposed simulator may improve the efficiency of building-up a multi-core simulation platform, and then may be used for simulation and evaluation of the constructed multi-core and NoC architectures.

Introduction

MPSoC (Multi-Processor SoC) boosts the performance of the chips by parallel computing. In such a multi-core era, NoC (Network-on-Chip) has been proposed to improve the throughput for parallel nodes via multi-hops network, instead of bus, etc [1]. A multi-processor simulator was proposed characterized by hybrid architecture, called as HMPSim (Hybrid Multi-Processor Simulator). The simulator supports three kinds of computation models, i.e. statistical traffic model, behavior-level model and cycle-accurate model based on ISS (Instruction-Set Simulator). The ISSs include ARM and TI DSP processors. The simulator also provides NoC and traditional buses. HMPSim may help the developers to build-up various architectures via configuration files. A case study was shown in this paper to illustrate the use of such a tool and the importance of early-state exploration.

Overview of HMPSim

HMPSim consists of network components library based on SystemC and ISS library, along with scripts for translating the network configuration files (Fig. 1). The network library includes cycle-level routers, buses, links and wrapper for combining the network and process element together. The ISS library includes ARM processor simulator and TI DSP simulator. Users may develop behavior-level and cycle-level applications, and then map the applications onto the network nodes. In such way, the application models and the network models may act as a complete MPSoC simulator. The system modeling procedure consists of three steps. Firstly, application may be defined as statistical traffic models, or be developed as behavior-level and cycle-accurate tasks, the latter of which shall be running on the ISSs. Secondly, configuration files may be developed to assign all kinds of parameters for network components including processing elements, and to map the applications tasks to each nodes. Thirdly, HMPSim may translate the configuration files into compile commands (Makefile), which will instruct the compiler and links to build the executable specification.

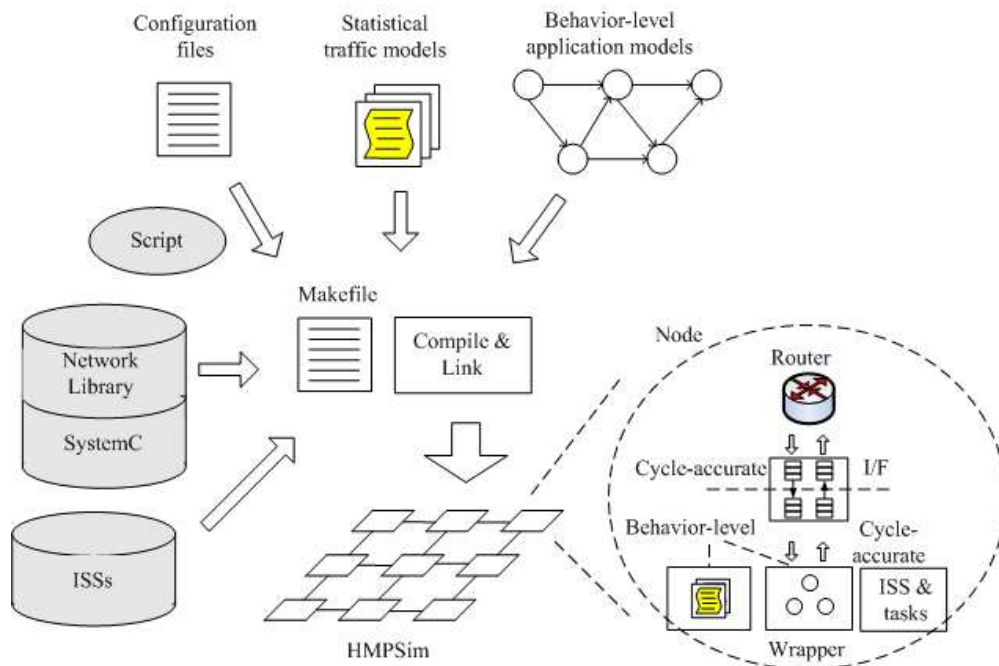


Fig. 1 Overview of the HMPSim simulator

In HMPSim, network components and wrappers have constructed the framework. The network components include routers and links models. The wrapper takes over the statistical traffic models and behavior-level application models. When the wrapper is bound with a processor simulator, it may ensure the signal interface between them to act as a whole node. As to implementation, the links were defined as *sc_signal* class and other models were defined as *sc_module* class, and all components share the same source clock defined as *sc_clock* class to build a global synchronous system. At the same time, all components were defined with parameterized constructors, which may construct instances following the arguments defined by the compilers. In this way, the applications may be bound with the network nodes to provide parameterized system modeling capability.

Implementation

In this section, all network components and wrappers were elaborated, including routers, buses, *ST_Wrapper* for statistical traffic model, *BC_Wrapper* for behavior-level tasks and *P_Wrapper* for ARM/DSP simulators.

Router. The router was modeled as classic pipelining wormhole router [2], including ports, buffers, crossbar, routing and scheduling components. All of these components are cycle-accurate except for the interface to wrapper. If the attached wrapper has included a behavior-level application task, the interface shall provide both behavior-level and cycle-accurate buffers for the router and the wrapper. HMPSim may configure the network topology, flit width, virtual-channels numbers, depth of buffers, routing and scheduling policy. In the same words, the network configuration may be implemented via definition of the routers in it.

Bus. HMPSim has employed a APB-like bus model. APB is one of the ARM's bus protocols, and has been used to interconnect the low-rate components. The APB bridge is only host node among all attached components. The bus model acts like a router in HMPSim, which has a pair of buffers for transmission and reception. For transmission buffer, it has a register to indicate the packet's target address, and its non-empty signal may be used as bus request. For receive buffer, the busy status indicates that it is receiving a packet, and will not receive other packets even the current packet is broken. The bus model has employed the round-robin scheduler for simplicity.

ST_Wrapper. HMPSim has defined the statistical traffic model as a group of files including random numbers, which may be produced by Matlab. Three such files may be employed by each node, which define the length, target address and interval of the packets to send. *ST_Wrapper* also has a pair of

buffers for transmission and reception. To produce packets, *ST_Wrapper* may get packet length and target address from the above files, and set the current system time in the packet as a time stamp. When the packet has been submitted to the bound router, the left random will be used to initialize the counter and start the counter. The wrapper will repeat above steps when the counter is set zero. The *ST_Wrapper* at the target will receive the packet and extract the time stamp to compute the delay for this packet to get through the network.

BC_Wrapper. HMPSim has employed a computation model something like KPN (Kahn Process Network) [3][4]. In such a model, behavior-level application may be defined as a group of tasks, which communicate via buffered FIFOs asynchronously. HMPSim has extended the model as below. Firstly, the distributed tasks write the transmission buffer in fashion of block, as the real system has only finite buffer resources. Secondly, the tasks read out the packets in buffers in fashion of non-block, as the tasks may poll the status of the buffers. As to implementation, a node may be bound with multiple tasks, all of them have implemented *HandleIt* method and defined its behavior in that method. To map the application-level channels to the buffers in network, each of the *BC_Wrapper* has a group of buffers just as the inner tasks' channels to other tasks. In this view, the point-to-point channels in the KPN have been implemented as the point-to-point node buffers and the common buffers in the network. The *BC_Wrapper* calls the inner tasks in data-driven mode, and will never call a task if only its transmission buffer is blocked.

P_Wrapper. To develop cycle-accurate application models, the applications shall be cross-compiled into binary for the target processor, and then be translated and executed by the ISSs. In such a way, the application exchanges the packets with the network through the interface between the ISS and the bound router. HMPSim has defined a group of fixed address as access to the transmission buffers and the reception buffers, along with the status of them.

Case Study

In this section, a classic network application for IPv4 packet transferring was developed and deployed in HMPSim, which is modeled at both behavior and cycle levels. The modeling approach and the results are as below.

System Overview. We constructed a 3×3 2D mesh with HMPSim, with the parameters set as following, dimension-order routing, 8-flit buffer depth, 8-bit flit width. The network application as defined as that in Fig. 2, in which the *mem_element* is the shared memory for other functional elements and the relationship between them is omitted for simplicity. The labels (x,y) indicate the locations and addresses of the elements.

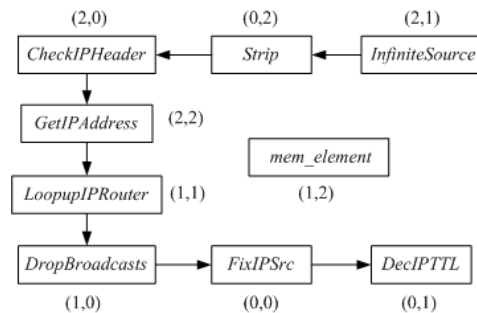


Fig. 2 IPv4 packet transferring applications and the mapping results

Configuration Approach. Part of the network configuration file and application configuration file are shown in Fig. 3. Each node was defined in the network configuration file in Fig. 3(a) and all parameters were defined between the labels of *Node_BEGIN* and *Node_END*. The connections between the nodes were defined between the labels of *Connection_BEGIN* and *Connection_END* in the same file. The applications were defined in the application configuration file including its accuracy, bound node, and application level connection as that in KPN. Fig.3(b) has illustrated a hybrid application model including statistical traffic, behavior and cycle level tasks.

```

// network configuration file
// global parameter
flit_width = 8 ;

Node_BEGIN // (0,0)
    node = N00 ; // node name
    port = [0..2] {
        // Three ports:L,N,E
        v = 2 ; d = 8 ;
        .....
    }
    pipes = 5 ; //5-stage pipelining router
    addr = 0 , 0 ; // node address
    routing = 0 ; // dimension-order
    // port mask for L,N,E,S,W
    routing_mask = 11100
Node_END

..... // other nodes
Connection_BEGIN
    N00 1 , N01 3 ; // N-S
    N00 2 , N10 3 ; // E-W
    .....
Connection_END

```

(a)

```

// application configuration file
// statistical traffic
// statistic = ( dest, interval, length )
// { node = N00 ; // node name }
// behavior-level tasks
actor = InfiniteSource { // element
    args = 1000, 1 ; // user parameters
    .....
}
actor = FixIPSrc { // element
    node = N00 ; // Bound node
    index = 0 ; // Inner index
    in_port [0..2] ;
    out_port[0..2] ;
}
// cycle-level tasks
actor = Strip { // element
    .....
    out_pOrt[0..2] ;
    Processor = ARM;
    UserApp = strip.out;
}
.....
Connection_BEGIN
    // Actor, OutPort, Actor, InPort
    FixIPSrc 1 DecIPTTL 1
    FixIPSrc 1 mem_element 13
    .....
Connection_END

```

(b)

Fig. 3 IPv4 packet transferring applications and the mapping results

Results. In every experiment, the *InfiniteSource* injected 1000 IPv4 packets and the functional elements passed the packet by the pointer to its location in the memory (*mem_element*) or passed the packet itself. At the same time, the application has been developed as behavior-level or cycle-level. The results may be concluded as follows. Firstly, the cycle-accurate application needs more effort in development and leads to worse efficiency in simulation. The efficiency gap between the cycle and behavior level application is more than 15 times. It is believed that the gap has been something reduced for the bottleneck in network. Secondly, the gap of network performance is distinct between the two packets transferring mode, i.e. passing the pointer or packet itself. The gap is for the hop-spot known as the shared memory. When passing the pointer, all functional elements shall access the packet in the memory at almost all time. When passing the packet itself, the elements transfer heavy data packet among them, but the hop spot was eliminated. As shown in Fig. 4, the latter passing mode may get more efficiency and less packet latency.

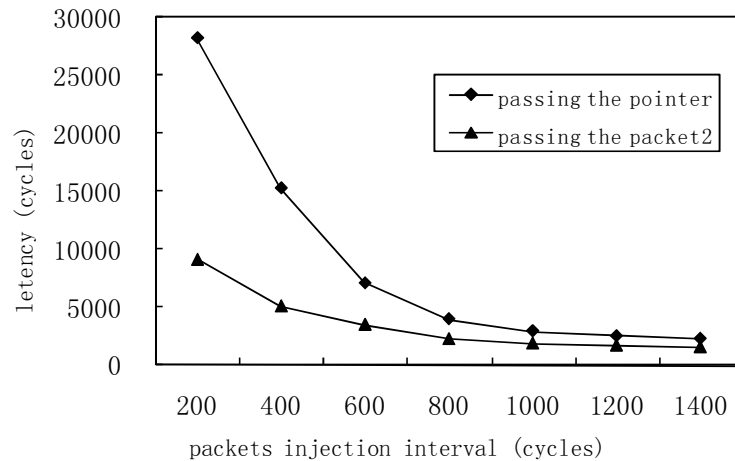


Fig. 4 Comparison of the latency when the elements transfer different packets payload

Conclusion

HMPSim supports different processors and interconnection models. With such a simulation-based tool, applications may be developed at different levels to meet different performance and accuracy requirements. As shown in the case study, HMPSim may be used to evaluate the architecture at early-stage, and help to eliminate the bottleneck in the design.

Acknowledgment

This work was supported by the project of “Highland of undergraduate education in Shanghai”, “Innovation Program of Shanghai Municipal Education Commission (11YZ251)”, “Shanghai University Scientific Selection and Cultivation for Outstanding Young Teachers in Special Fund”, and the “Startup Funding of Shanghai Second Polytechnic University”.

References

- [1] Benini L., De Micheli G. Networks on Chips: A New SoC Paradigm[J]. IEEE Computers, 2002, V35(1):70-78
- [2] LS Peh, WJ Dally. A Delay Model and Speculative Architecture for Pipelined Routers[C]. International Symposium on High-Performance Computer Architecture(HPCA), Nuevo Leone, Mexico, 2001:255-266.
- [3] TM Parks. Bounded Scheduling of Process Networks[D]. University of California, 1995.
- [4] T. Kangas. Methods and Implementation for Automated System on Chip Architecture Exploration[EB/OL].
http://www.martes-itea.org/public/papers/Kangas_-_Methods_and_Implementatio.pdf