Modeling and Analysis of SLDL-Captured NoC Abstractions

Ran Hao, Nasibeh Teimouri, Kasra Moazzemi, Gunar Schirner

Modeling and Analysis of SLDL-captured NoC Abstractions

Ran Hao, Nasibeh Teimouri, Kasra Moazzemi, Gunar Schirner
hao.R@husky.neu.edu, nteimouri@ece.neu.edu,
moazzemi.k@husky.neu.edu, schirner@ece.neu.edu

Department of Electrical and Computer Engineering
Northeastern University
Boston, MA, 02115

Abstract With increasing number of IP cores, parallel communication architectures including NoCs have emerged for many-core systems. To efficiently architect NoCs, early analysis of crucial run-time metrics such as throughput, latency and saturation time is required. This requires abstract modeling of NoCs. Modeling abstraction, and consequently the modeling granularity impacts the accuracy and speed of simulation. While a fine-grained model will slowly lead more accurate information, a coarser model simulates faster and yields less accurate predictions. This paper first identifies possible levels of abstraction for NoC models and correlating captured features with the accuracy/speed trade-off. Second, this paper proposes two NoC models at different abstraction levels: a finer grained Bus-Functional Model (BFM), and a coarser Transaction-Level Model (TLM). The BFM updates the system status after any events happening during data unit transmission, while the TLM updates the system status at the end of data unit transmission.
Our evaluation results show moving to higher abstraction (from BFM to TLM) gains 10x to 50x speedup at the cost of 10% - 20% accuracy loss on average. Our analysis approach and results guide system architects in exploring NoC architectural alternatives and help identifying suitable abstract levels.

1 Introduction

Chip Multiprocessing (CMP) as one essential solution for parallel processing and high performance computing has evolved to exploit parallelism in the form of integrating multiple processor cores on a single chip which is known as System on Chip (SoC). To power and performance efficiently connect the cores, reusable interconnect architectures are required that provide scalable bandwidth and parallelism.

These requirements cannot be met by traditional interconnect architectures like single shared bus or even hierarchy of buses due to their poor scalability as they allow only one (or a few number of) sender-to-receiver communication(s) at a time [1]. Promising alternatives are Networks on Chip (NoCs). NoCs avoid the need for dedicated wires for each individual communication, and connect IP cores through an on-chip network. Several advantages compared to dedicated wires include delivering high-bandwidth, low-latency and low-power communication over a flexible and modular medium [2] [3].

Different NoC design parameters (e.g. topology, communication mechanism, routing method and switching mode) impact a multi-dimensional trade-off space between latency,

throughput, communication load, energy consumption, and silicon area. Therefore, early evaluation of NoC is in high demand [4]. Recent approaches on NoC emulation aim for accuracy. However, they could be too slow, especially when considering the tight time-to-market [5].

System level modeling can relieve time to market pressures and the expense of NoC simulation/emulation tools with providing faster architecture exploration, performance evaluation, and functional validation [6]. Abstract modeling of an NoC poses the question of abstraction levels (the amount of detail to be retained in the model). Ultimately, this poses a trade-off between simulation speed and accuracy [7] as visualized in Fig. 1.

A highly abstract NoC model abstracts away much of the underlying communication details, and in result yields a very fast but inaccurate simulation. Conversely, an accurate model would capture more of the communication principles, resulting in a slower but more accurate simulation. Although [8] discusses about different abstraction levels, their precise definition and modeling abstraction rules are not clearly presented. Defining abstraction levels helps designers to select communication features to model given a desired speed/accuracy.

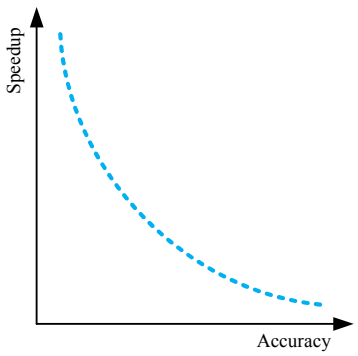


Figure 1: Speed Accuracy Trade-off

This paper first identifies different NoC abstraction levels according to the visibility of implementation details and communication granularity. A most abstract model treats the whole NoC as one black box, only revealing input and output traffic. Conversely, the most detail level exposes how individual flits are handled at the micro-architecture level. This paper defines accuracy impact factors at each level, highlights contention points over shared resources within an NoC and identifies the required arbitration points.

Next, the paper proposes an abstract NoC model using Transaction Level Modeling (TLM). It realizes the structure of Hermes router model [4] with 5 bidirectional (both input/output) ports to the four neighboring IP cores and one port to the local IP core.The model implements XY routing and wormhole switching. The proposed TLM captures all the router features and arbitration events over its shared resources at cycle-level. It updates the system status at each cycle independent of how many arbitration events are notified during that cycle.

To evaluate the performance and accuracy of the proposed TLM versus an accurate model, this paper also introduces a more detailed model, a Bus Functional Model (BFM). The BFM follows the bus functional modeling principals [6] aiming to be as accurate as an RTL implementation. The BFM captures much more structural and micro-architectural detail and it updates the system status upon any arbitration event. This improves accuracy at the cost of simulation speed.Our TLM is 10 times faster than the BFM at the cost of 10% to 20% error.

The rest of this paper is organized as follows: Section 2 discusses related work in NoC modeling and analysis. Section 3 defines different abstraction levels for NoC modeling. Section 4 first presents the structure of modeled router, then proposes our

TLM and BFM NoC abstract models. Following that, Section 5 validates the proposed BFM as accurate against an RTL implementation of NoC and then evaluates the proposed TLM versus BFM. Finally Section 6 concludes the paper.

2 Related Work

Exploring the design trade-offs of NoC metrics including bandwidth, power, performance and silicon area can be done at different levels of accuracy and details based on the design requirements. For instance, SW development needs fast simulation. Conversely, performance estimation demands a finer set of details and higher accuracy for proper validation [9]. In general, work on evaluating NoC can be categorized in 3 groups: emulation/simulation frameworks, static analysis, and abstract modeling.

Emulation/Simulation Framework: Many NoC simulators and emulators have been developed; the emulation platform proposed by Dally et al. as a flexible emulation environment implemented on FPGA based on a complete mixed HW-SW NoC emulation framework, Xmulator [10] as an event-driven simulator and Booksim [11] as a cycle-driven simulator are a few instances of tools in this category. All instances impose high implementation cost, maintenance difficulty and long emulation/simulation time.

Some works [12] and [13] aim to reduce the emulation/simulation time by changing the kernel scheduler, simulation/evaluation semantics by adding local clocks/schedulers. Nevertheless, their improvements are case-specific, for instance [12] gains more as the size of NoC gets larger.

Static Analysis: Static analysis like [14] and [15]rapidly yields timing parameters such as router service time and packet arrival time. These methods have low accuracy as they abstract away dynamic behaviors influencing NoC performance and bandwidth.

Abstract Modeling: Abstract modeling might be placed in between two above categories. It abstracts away some implementation detail (such as bit-level communication details) and takes into account only the events occurring per transmission of coarser data granularity. The goal is to accelerate the NoC evaluation while maintaining some accuracy. The architectural model in [9] is one example. It models the HERMES [4] router architecture as a bus and all cores/routers connected to it as individual modules. With keeping track of all routers' active flows in different FIFOs and prioritizing their requests based on the pre-defined priorities, all the competitions over the shared resources are captured. The drawback of this work compared to the proposed TLM is its evaluation for worst-cases; when all possible contentions over the shared resources happen. Similarly, [16] proposes an accurate abstract model for on-chip interconnects. It uses bus protocol specifications to identify a reduced set of timing points. Finding the set of optimal timing points is the drawback of this work compared to the proposed TLM.

3 NoC Abstraction Models

Conceptually, many abstraction levels are possible that may range from an extremely coarse grain model the treats the whole NoC as one black box, to a very fine-grained model that exposes micro architectural implementation details of all the NoC elements. Abstracting NoC can occur with different levels of details; from low covering details

such as observing the whole NoC as a communication box to considering all micro-architectural implementation details of all NoC elements. When comparing abstraction levels, the following aspects should be considered:

Granularity of data defines the smallest unit of data transferred through the NoC.

Visibility defines the level of implementation details of NoC communication observable in the model.

Arbitration points lists the shared resources for which contention is dynamically resolved.

Timing accuracy outlines the resulting estimation accuracy; meaning that at which level of accuracy, an NoC model can estimate the timing behavior of a real NoC.

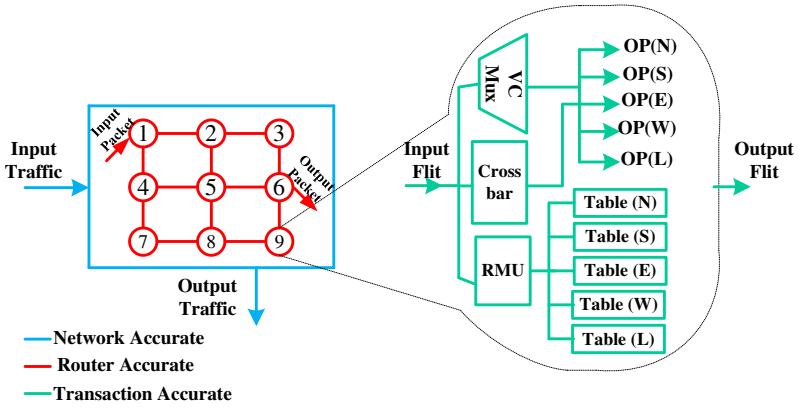


Figure 2: NoC Modelling Granularities

Given the characteristics above, we propose five abstraction levels. Table 1 summarizes the models, and Fig. 2 illustrates the 3 most abstract models.

Network-Level Model: models the whole NoC as a black box and only exposes the local ports. This model abstracts away everything inside NoC including traffic paths and contentions over the shared resources. The model estimates network latency based on statistical information like average/worst case network latency per pre-defined size of traffic and the amount of traffic transferred through the network.

Router-Level Model: realizes NoC as a set of routers connected to each other via physical channels. In this model, routers are modeled as black boxes which receive packets as input and sends output packets over a physical link to the next router. This model estimates the NoC performance/latency based on the number and size of packets as well as the length of path taken by each packet. It dynamically resolves contention on physical links.

Transaction-Level Model: more details over the router-level model by modeling router internal modules, including input/output ports, cross-bar, routing management, virtual channel (VC) allocation and flow control management.

In this model, at the end of any transaction, the contentions (and arbitration events to resolve them) which change the system status are collected and the system status is updated. Based on [17], a transaction is defined as injecting the header flit (first part of a packet) by initiator and receiving the last flit of the packet by the receiver.

Pin-Level Model: implements all the internal wires/pins per router modules and updates the system status after any individual contention (and arbitration to resolve that) happening per transmission of each bit of the transaction.

Micro-Architectural-Level Model: implements the Pin-Accurate model and all of it's router operations at gate level for final validation. This model is practically an RTL model, very close to the final implementation, most accurate but also the slowest.

From the network-level model to micro-architectural level model, communication and implementation details are added to the model, increasing accuracy at the cost of simulation speed. Table 1 summarizes the abstraction levels.

Table 1: NoC Abstraction Overview

Model	Visibiltiy	Granularity	Arbitration Point	Time Unit
Network-Accurate	-	Traffic	-	Loosely Time Estimated
Router-Accurate	Channel	Packet	Routers	Approximate Time Estimated
Transaction-Accurate	Channel	Flit	Router Modules	Cycle Estimated
Pin-Accurate	Wire	Bit	Router Modules	Cycle
Micro-Arch	Wire	Bit	Router Modules	Cycle

4 Proposed NoC Models

4.1 Router Architecture

Our router models are based on the HERMES router architecture [4] with slight changes.

Fig. 3 outlines the router's internal structure with 4 important functional units: Input/Output Ports, Routing Management Unit (RMU), Flow Control Unit (FCU) and crossbar unit. RMU includes Routing Logic Unit (RLU) and central table to record the status of virtual channels (VCs) of the output ports. FCU for promulgating free ports to the neighboring output ports. Crossbar unit forwards the packets to the next router determined by the RMU.

Each router is connected to 4 neighboring routers through the 4 input/output ports. One local port connects the router to the local IP core. Each input port has a configurable (8bits in Hermes, 32bits in our model) size of VC-buffers to record the received data/control flits. A flit is the smallest part of a packet. There is no buffer in the output ports as the buffer in the next router's input port is used. For this, a credit-based flow control mechanism is employed to notify the sending side about the available space on the receiving side. This way, flit is only sent if there is space on the receiving side. The RLU (part of RMU) computes which output port to sent a packet to. The VC allocation unit selects which VC to use for a given output port. One of our changes over the Hermes architecture is supporting individual RLUs for each input port in order to avoid congestion inside the router. Similar to Hermes, the routing method is XY; approaching the destination always first horizontally, then vertically or vice versa. Routing decision is made per header flit which contains destination information and packet length. Flits are switched using the wormhole method.

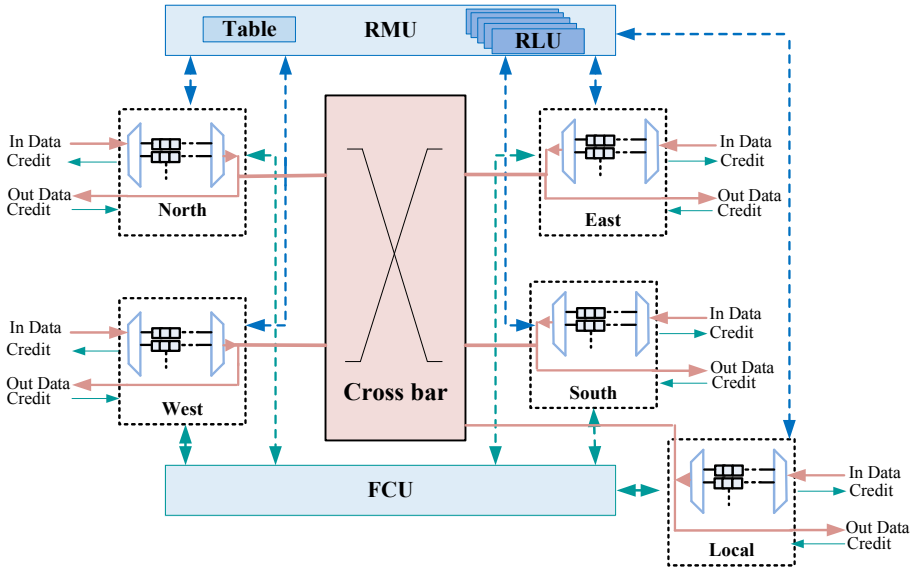


Figure 3: The Router Architecture in the Proposed NoC Abstract Models

4.2 Packet Transmission

Example a received flit is stored in the VC buffers of that input port. In case of a header flit, it's destination is forwarded to the RMU to determine the output port. After selecting the output port, the RLU consults RMU table to find an available VC on the output port.

After selecting output port and VC, the input port refers to FCU to check if there is enough room for receiving this flit in the next router. In our credit-based flow control, each VC has a credit and when the VC is used by a flit, its credit is decreased. When the flit leaves the input buffer of the destination router, a credit is sent back to the sending router, increasing the VC's credit count.

Assuming sufficient sending credit is available, the crossbar sends the flit from the input port to the output port (and subsequently sends a credit upstream). The remaining flits of this packet are then sent one by one consulting the FCU about receive buffer credits to the next router. Upon receiving the tail flit, the RMU de-allocates the output buffer and VC.

During packet transmission within a router, various shared resources are used for which accesses need to be arbitrated. Detecting and resolving/arbitrating the contentions impacts the accuracy.

4.3 Arbitration Points

One of the most important aspects impacting accuracy in modeling is detecting contentions over shared resources and resolving them. Shared resources are FCU, crossbar, and output ports. The way how access requests to these shared resources are collected and arbitrated affects the modeling accuracy. We identify one contention type for each resource (see also summary in Table 2):

1. **Connection Establishment**: if a router receives two header flits that target the same output port, their requests contend for the RMU. An arbiter is required to select one of requests. The selected request gets access to the RMU (central table of RMU), then starts connection and sends data.
2. **Request Flow Control Grant**: simultaneous flow control requests at the same FCU for different VCs create contention over FCU access. Concurrent requests are feasible, as they have already received the credit to send data. In order to guarantee that at one point of time, only one traverse is allowed to a specific output port, an arbiter is necessary to give flow control grant signal to one of the requests. We define this arbitration point as arbitration for same output accesses.
3. **Crossbar Access**: when more than one VCs at the same input port gets the flow control grant simultaneously, there is contention on the crossbar. In our design, to avoid this contention, we define an arbiter for crossbar access from the same input port. This arbiter grants crossbar access to only one of the requests.

Table 2: Contention Events and Arbitration Points

Arbitration point	location	Resource	Arbitration (BFM)	Arbitration (TLM)
Connection Establishment	RMU	RMU table	FIFO	random
Request Flow Control Grant	FCU	output port	FIFO	random
Crossbar Access	IP	crossbar	Round-robin	random

4.4 NoC Abstract Models: TLM and BFM

This paper captures two abstract models of NoC; Transaction-Accurate Model (TLM) and Pin-Accurate Model (BFM) in the System-level Design Language (SLDL), SpecC[18]. Both models take into account the arbitration points explained in Section 4.3 as well as the characteristics of Table 1. However, they differ in the way that requests to the shared resources are collected and arbitrated. The BFM gathers resource access requests based on sampling and driving of every single wire at each cycle. Conversely, the TLM gathers the access information at transaction. Consequently, the BFM updates the system status at any cycle, while the TLM updates at transaction boundaries. As the granularity of updating the system status affects the accuracy, TLM is less accurate than BFM.

Both models implement the same arbitration policies. However, as the TLM makes a decision at a coarser granularity, it is more susceptible to the order in which access requests appear. Within the same time quantum, the TLM cannot distinguish between concurrent requests. As the execution order is not specified by the underlying discrete event simulation semantics, the effective arbitration policy for simultaneous (same quantum) becomes random.

Moreover, the BFM is driven by an explicit clock, while the TLM virtually times routers by using the instruction $wait for$. In some sense the TLM can be considered time driven, while BFM is event-driven. Both models also differ in the number of threads (sc_module in SystemC, or behavior in SpecC) used for simulation. The BFM employs active threads for each router module. Conversely, the TLM is mainly channel based (ie. is call driven) and only uses one behavior (or sc_module) for each VC. For instance, assuming 4 VCs per physical link, the BFM has 41 simultaneous threads and the TLM only 20. With the lower number of active threads, the TLM can perform faster (avoiding context switches). Table 3 compares BFM and TLM.

Table 3: Comparing TLM and BFM

	BFM	TLM
Communication Implementation	Behavior	Channel
Arbitration Policy	FIFO & Round-robin	Random
Timing	Event driven (explicit clock)	Time driven ($wait for$)

5 Experimental Results

This section explores the proposed BFM and validates its accuracy and functionality with respect to the RTL implementation. It then compares the TLM versus the BFM based on speed and accuracy.

For evaluation of the models, we mainly use hot spot traffic [19]; some nodes in the network receive most of the traffic.

5.1 BFM Validation

System performance and throughput are two important metrics for analyzing NoC architectures. Average packet latency is a representative of system performance, and link utilization is a representative of throughput. Packet latency is the packet life-time defined as the difference between its start time label and its end time label.

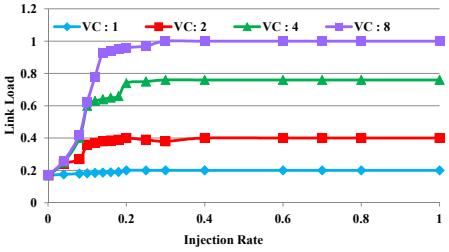
Link utilization/load is the ratio of link busy time over the whole simulation time. Link busy time is defined as the total time when the link is busy carrying traffic.

In this part and for validation of the proposed BFM, we adopted 40%-hot spot traffic and defined packet size as 10 flits. 40%-hot spot traffic means that 40% of the nodes are the destinations of total traffic injected to the network. Each injector (hot node) injects 100 packets to the network. Fig. 4 shows the simulation results including link load, average packet latency and simulation time for hotspot traffic injected into the 8*8 mesh. The results are correlated with the results of VC extended HERMES router structure on FPGA [20].

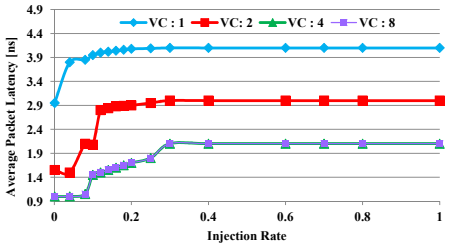
Fig. 4a shows link load for different numbers of VCs as the injection rate increases. At small injection rate, the link load increases linearly for all VCs. However, the link load starts to level off from a specific injection rate, around 10%, 15%, 20% and 30% respectively for 1 VC, 2 VCs, 4 VCs and 8 VCs. Based on [20], this point is called saturation point. The saturation degree of network for multiple VCs in our model is higher than what is reported in [20]. The reason lies in the different implementations of the network interfaces. In our work, we define multiple VCs for local connections as well, which means at the destination node, traffic from different ports can sink into the local PE without being blocked. With the improved mechanism, the network throughput for VCs of 8 can reach 100% under hotspot traffic.

Fig. 4b shows the average latency as injection rate increases. Up to the saturation point, the average latency is constant for all VCs. With increasing the number of VCs, the average latency drops in a half when VCs goes from 1 to 2. Similar situation for 2 VCs to 4 VCs. However, the average packet latency for VCs of 4 and 8 are similar. The reason is that using 4 VCs has already eliminated most of packet blocking.

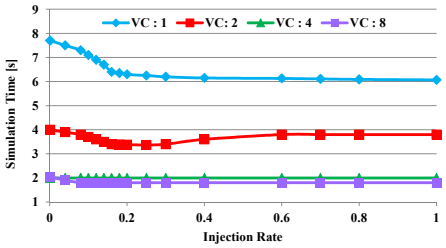
Fig. 4c shows the overall finish/transmission time as the injection rate increases. This time is when all the packets reach their destinations. Since more VCs leads to more traffic overlap on the fly, 8 VCs yields the shortest overall transmission time.



(a) Link Load



(b) Peak Latency



(c) Completion Time

Figure 4: BFM Performance Validation (8x8 mesh, 40%-hot spot)

5.2 TLM Evaluation

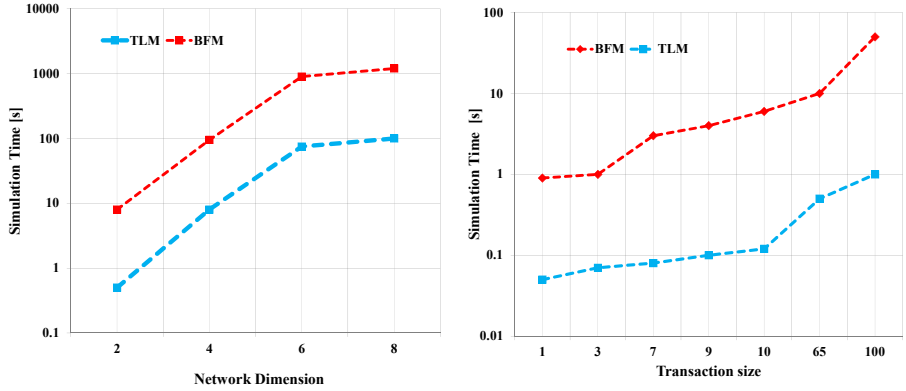
This section evaluates the TLM compared to the BFM with respect to speedup and accuracy.

SpeedUp: depending on the amount of implementation details and number of context switches, the simulation time varies. For comparing the models, simulation speedup is reported. Simulation speedup of the model with higher level of abstraction (H) compared to the model with lower level of abstraction (L) is defined as 1.

$$Speedup_{H2L} = Simu.\,Time_L \;/\; Simu.\,Time_H \tag{1}$$

As the simulation time strongly depends on the number of context switches, the simulation speed is closely correlated with the network size and traffic intensity. Both network size and traffic intensity affect the the number of behaviors and context switches. Network size is the number of nodes in the network. Network intensity is defined as the number of transactions (number of packets) from sender nodes to the receiver nodes. With larger network or intense traffic simulation time increases.

To evaluate the effects of network size and traffic intensity, 40%-hot spot traffic is simulated with 4 VCs per physical link and 100% injection rate. Fig. 5a shows the simulation time for increasing network (mesh) size from 2*2 to 8*8. The TLM is 10x to 16x faster than the BFM. With larger networks, TLM achieves higher speedup as a result of abstracting away higher ratio of communication details. Fig. 5b illustrates the simulation time for network intensity. With increasing the transaction size (from 1



(a) Simulation over network size

(b) Simulation over network intensity

Figure 5: Comparing simulation time of BFM and TLM (8*8 mesh, 40%-hot spot)

packet to 100 packets), TLM achieves an increasing speedup (from 14x to 50x) again as res ult of the more abstract simulation.

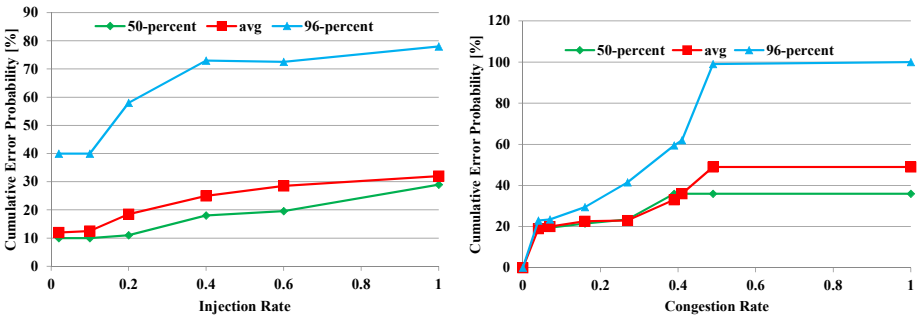
To measure the accuracy loss, we define accuracy error for each packet. As Equation (2) defines, this accuracy error has correlation with the difference between packet latency in the TLM and BFM. Packet latency is the difference between start time label and end time label of the packet.

$$Error = |PacketLatency_L / PacketLatency_H| / PacketLatency_L \qquad (2)$$

As the TLM differs from the BFM in the effective arbitration policy (due to collection of requests and arbitration among them), measuring the accuracy loss in the TLM requires simulation scenarios with different amount of contentions (requests) over the shared resources. The amount of contentions over the shared resources is determined by the amount of traffic injected to the network. The more the injection rate, the higher the number of simultaneous requests for the same resources and higher contention and accuracy loss as a result. To demonstrate this, 40%-hot spot traffic is adopted into the TLM and BFM models of 6*6 NoC with 4 VCs per physical links and 100% injection rate. We simulate 100 transactions through the NoC and measure the transmission delay of packets in both models BFM and TLM. To aggregate the results, we report the average error, as well as the cumulative error for 50-percentile and 96-percentile as Fig. 6a. The 50-percentile (96-percentile) cumulative error indicates the maximal error experienced by 50% (96%) of transactions. As Fig. 6a represents, increasing the injection rate, increases the cumulative error probability. Increasing the injection rate from 0.1 to 0.2 increases the average error from 10% to 20%. At 0.1 injection rate 96% of packets observe less than 40% error, while 50% see less then 10% error. Increasing the injection rate to 0.6, makes 50% of packets experience up to 30% error.

Increasing the injection of rate increases the contention over shared resources. One indicator is the congestion rate over physical links. Fig. 6b shows the cumulative error probability over increasing congestion. All three metrics are strongly related to congestion. And increase until congestion hits 50%. Then, 50% show at most 38% error, while

the maximum error measured for 96% of packets reaches 100%. Conversely, at lower congestion rate, e.g. 5%, 96% of packets experience less than 20% error.



(a) Injection Rate (b) Congestion Rate

Figure 6: Cumulative Probability of Accuracy Error

6 Conclusion

Modeling of NoCs is important for early exploration of NoC design alternatives. In this context, fast and accurate simulation is important. However, when abstracting NoC models, a trade-off between simulation speed and accuracy exists. This paper has identified NoC abstraction levels, differing in data granularity, visibility of internal structures and modeling of contention points. This paper has introduced two NoC models, an detailed Bus-Functional Model (BFM) which models and arbitrates at each clock cycle and a more abstract Transaction Level Model (TLM) that operates on coarser transactions. Both models have been captured in the SpecC SLDL. We have validated our BFM to sufficiently match the RTL implementation. The TLM simulates 10x to 50x faster than BFM at a cost of 10%-20% accuracy. Both speedup and accuracy loss increase with network size and traffic.

References

1. M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vencentelli, “Addressing the System-on-a-chip Interconnect Woes Through Communication-based Design,” in *Design Automation Conference (DAC)*. ACM, 2001, pp. 667–672.
2. J. Owens, W. Dally, R. Ho, D. N. Jayasimha, S. Keckler, and L.-S. Peh, “Research Challenges for On-Chip Interconnection Networks,” *Micro, IEEE*, vol. 27, no. 5, pp. 96–108, 2007.
3. W. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Design Automation Conference (DAC)*, 2001, pp. 684–689.
4. F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, “HERMES: An Infrastructure for Low Area Overhead Packet-switching Networks on Chip,” *Integr. VLSI J.*, vol. 38, no. 1, pp. 69–93, 2004.

5. S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya, "An analytical method for evaluating Network-on-Chip performance," in *Design, Automation Test in Europe (DATE)*, 2010, pp. 1629–1632.
6. G. Schirner and R. Dömer, "Abstract Communication Modeling: A Case Study Using the CAN Automotive Bus," in *From Specification to Embedded Systems Application*, Springer. Springer, 2005.
7. ——, "Quantitative Analysis of Transaction Level Models for the AMBA Bus," in *Design, Automation and Test in Europe (DATE)*, vol. 1, 2006, pp. 1–6.
8. K. Lu, D. Muller-Gritschneder, and U. Schlichtmann, "Accurately timed transaction level models for virtual prototyping at high abstraction level," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 135–140.
9. L. Indrusiak and O. dos Santos, "Fast and accurate transaction-level model of a wormhole network-on-chip with priority preemptive virtual channel arbitration," in *Design, Automation Test in Europe Conference (DATE)*, 2011, pp. 1–6.
10. A. Nayebi, S. Meraji, A. Shamaei, and H. Sarbazi-Azad, "XMulator: A Listener-Based Integrated Simulation Platform for Interconnection Networks," in *International Modeling Simulation (AMS)*, 2007, pp. 128–132.
11. N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, "A detailed and flexible cycle-accurate Network-on-Chip simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.
12. M. Hosseinabady and J. Nunez-Yanez, "SystemC architectural transaction level modelling for large NoCs," in *Forum on Specification Design Languages (FDL)*, 2010, pp. 1–6.
13. E. Viaud, D. Potop-Butucaru, and A. Greiner, "An Efficient TLM/T Modeling and Simulation Environment Based on Conservative Parallel Discrete Event Principles," in *Design, Automation and Test in Europe (DATE)*, 2006, pp. 1–6.
14. S. Suboh, M. Bakhouya, J. Gaber, and T. El-Ghazawi, "Analytical modeling and evaluation of network-on-chip architectures," in *High Performance Computing and Simulation (HPCS)*, 2010, pp. 615–622.
15. U. Ogras, P. Bogdan, and R. Marculescu, "An Analytical Approach for Network-on-Chip Performance Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 2001–2013, 2010.
16. H. van Moll, H. Corporaal, V. Reyes, and M. Boonen, "Fast and accurate protocol specific bus modeling using TLM 2.0," in *Design, Automation Test in Europe (DATE)*, 2009, pp. 316–319.
17. F. Ghenassia, *Transaction-Level Modeling with Systemc: Tlm Concepts and Applications for Embedded Systems*. Springer-Verlag New York, Inc., 2006.
18. A. Gerstlauer, R. Dömer, J. Peng, and D. D. Gajski, *System Design: A Practical Guide with SpecC*, 2001.
19. M. L. Fulgham and L. Snyder, "Performance of Chaos and Oblivious Routers Under Non-uniform Traffic," Tech. Rep., 1993.
20. A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC," in *Integrated Circuits and System Design*. ACM, 2005, pp. 178–183.