Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

Multi-Resolution Co-Design Modeling: A Network-on-Chip Model

Soroosh Gholami
Hessam S. Sarjoughian

Arizona Center for Integrative Modeling and Simulation
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ 85281, USA

ABSTRACT

This paper proposes a multi-resolution co-design modeling approach where hardware and software parts of systems are loosely represented and composable. This approach is shown for Network-on-Chips (NoC) where the network software directs communications among switches, links, and interfaces. The complexity of such systems can be better tamed by modeling frameworks for which multi-resolution model abstractions along system's hardware and software dimensions are separately specified. Such frameworks build on hierarchical, component-based modeling principles and methods. Hybrid model composition establishes relationships across models while multi-resolution models can be better specified by separately accounting for multiple levels of hardware and software abstractions. For Network-on-Chip, the abstraction levels are interface, capacity, flit, and hardware with resolutions defined in terms of object, temporal, process, and spatial aspects. The proposed modeling approach benefits from co-design and multi-resolution modeling in order to better manage rich dynamics of hardware and software parts of systems and their network-based interactions.

1 INTRODUCTION

Multi-Resolution Modeling (MRM) aims at describing individual parts of a dynamical system at different resolutions with support for composing them to represent also the whole system at different resolutions. Model resolution can be in terms of *time*, *space*, *process*, and *object* (Davis and Bigelow 1998). Models can vary in terms of their time granularity, spatial sizes, process mechanisms, and object states and functions. These can lead to a multitude of hierarchical model components with varying relationships. MRM may be practiced using domain-neutral *specification hierarchy* (Zeigler, Praehofer and Kim 2000). The levels within the system hierarchy establish different *model type* resolutions. MRM requires domain-specific abstraction levels. Target abstractions such as *Interface*, *Capacity*, *Flit*, and *Hardware* for Network-on-Chip systems can be cast into one or more levels of a system specification hierarchy. These NoC abstraction levels are derived from domain experts and targeted to satisfy certain needs. Within the system specification hierarchy, modelers can begin developing a high-level model of a system (e.g., for policy analysis) and gradually increasing the model's resolution toward an actual system (e.g., specifying blueprints for implementing the system).

As software continues to have a greater role in system complexity, the hardware/software co-design paradigm that is used in embedded systems is becoming a necessity for systems such as Network-on-Chips. In this paradigm, designers embrace separating HW and SW development early on. This is because regardless of the actual system we are working with, the requirements of the software and hardware pose design constraints on one another. Co-simulation can help tackle some design requirements and solutions (Liem et al. 1997). However, co-design must deal with the increasing heterogeneity, complexity, and integration issues of HW and SW in electronic systems (Teich 2012).

Network-on-Chip as an integrated hardware-software system can handle the increasing complexity of the communication subsystems of System-on-Chips. Some key capabilities include reduced messaging

overhead and packet-based communication using configurable network topology. Like other networked system, NoC can be modeled at multiple resolutions using *object*, *temporal*, *process*, and *spatial* abstractions (Davis and Bigelow 1998). In NoC, individual operations can have temporal resolution. NoC components have designated locations and dimensions (spatial resolution). Network processes such as routing, flow control, and arbitration define process resolution. Finally, NoCs with different number of objects and levels of hierarchy can be specified (object resolution).

New NoC designs are moving toward higher complexity (Lis et al. 2011), 3D structures (Pavlidis and Friedman 2007), real-time support (Bolotin et al. 2004, Wiklund and Liu 2003), energy efficiency (Hu and Marculescu 2004) and cache coherence (Martin, Hill and Sorin 2012). This calls for more sophisticated methods for modeling, simulating, testing, and verifying the NoC before and after implementation. Many modeling/simulation frameworks have been developed by the community to address these needs. However, few consider the software and hardware components separately; instead, NoC is viewed as a single system, ignoring its hybrid nature.

Our goal is to provide a new modeling framework for NoC where the static and dynamic aspects of NoC are specified using both multi-resolution modeling and co-design. This framework considers the NoC models at different resolution levels appropriate for various stages of SW/HW system co-design. According to (Dally and Towles 2004), NoCs are defined at four abstraction levels: 1) Interface, 2) Capacity, 3) Flit, and 4) Hardware. Although the abstraction levels have distinct hardware and modeling resolutions, it is useful to cast them within a multi-resolution modeling framework. From a modeling vantage point, NoCs can be described at multiple levels of abstraction. One approach is known as system modeling hierarchy (Zeigler, Praehofer and Kim 2000). A system can be specified from highest level to lowest level of abstraction: 1) I/O Frame, 2) I/O Relation, 3) I/O Function, 3) Iterative I/O Specification, 5) I/O System Specification, 6) Coupled System, and 7) Coupled Network of Systems (Zeigler, Praehofer and Kim 2000). The I/O Frame affords specifying the least knowledge and the Coupled Network of Systems affords specifying the most knowledge. In this modeling hierarchy, an abstraction with more knowledge incorporates the abstractions that have less knowledge. We elaborate on abstraction levels 5 and 6 in this paper. We have network software dedicated to NoC and application software dedicated to System-on-Chip. We consider simulation as a part of co-design which enables designers to synthesize the system at some desired levels of resolution.

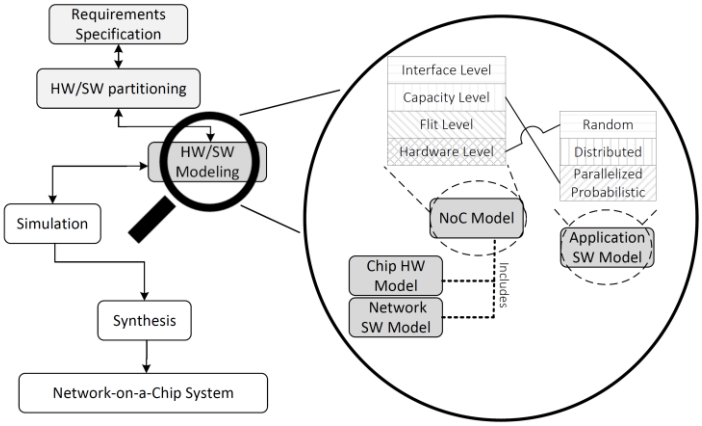


Figure 1: MRM-based HW/SW Co-Design Methodology.

2 BACKGROUND

This section provides an introduction to Multi-Resolution Modeling (MRM), Hw/Sw Co-design, and NoCs. We highlight some existing works for NoC design/simulation and the key roles MRM and Co-Design can play in meeting current and future NoC design challenges. In (Marculescu et al. 2009), the

authors identify five major categories of open research and future challenges for NoC design and designers: 1) application specification and modeling, 2) application optimization for communication, 3) communication architecture synthesis and optimization, 4) communication architecture analysis and evaluation, and 5) NoC design validation. In this research, we focus on the first, fourth, and fifth categories. In the first category, the challenge is application traffic patterns and bandwidth requirements. In the fourth category, the aim is identifying congestion points, hot spots, and performance evaluation. In the fifth category, the aim is testing and validating the design (Marculescu et al. 2009).

2.1 Multi-resolution Modeling

Multi-resolution modeling (Davis and Bigelow 1998) is used in various field of research such as graphics (Garland 1999) and defense systems (Davis and Bigelow 1998). The challenges associated with MRM have been recognized for many years and application domains. MRM in the context of NoC modeling is defined as a set of models, each focusing on serving a particular purpose. We use the NoC model resolutions defined in (Dally and Towles 2004). The resolutions are defined from lower to higher resolutions as Interface, Capacity, Flit and Hardware levels (see Figure 1). We can observe that lower-level resolution models are suited for verification using model checking while higher-level resolution models are suited for validation using simulation. Thus, verification and simulation offer unique, complementary capabilities for designing NoC systems.

One important point to note is that no MRM frameworks have been proposed for NoC and System-on-Chips (Berekovic, Stolberg and Pirsch 2002). Without such a framework, it is difficult to classify what could be highest or lowest resolution models (Catania et al. 2015). While one aspect of resolution (such as object, process, time, or space) of a model may be increasing, another aspect may be declining (Davis and Bigelow 1998). The power of MRM lies in defining levels of model abstractions and more significantly how models that have different levels of resolution can be related. We cast the NoC's levels of abstraction in terms of MRM with SW/HW co-design consideration.

2.2 Hardware-Software Co-design

Hardware-software co-design process is considered within our proposed NoC modeling framework. Co-design is needed as in embedded systems (Chiodo et al. 1994). Figure 1 depicts a high-level process flow of the NoC design approach. Phases colored in light grey are those applied to the NoC system as a whole. These phases include the requirement specification, partitioning, modeling, simulation, and emulation. After the partitioning phase, NoC design is carried out in two major branches (the hardware and the software). The hardware is further divided into chip hardware and network software. The three components (hardware, network software, and application software) are coordinated and tested against one another using co-simulation. Multi-Resolution Modeling (MRM) is applied to the dark grey blocks. Hardware and software models are devised at various resolutions and co-simulated for the purpose of validation or making design decisions. In our integrated design approach, we refer to the software for the Processing Elements as application software. As illustrated in Figure 1, the SW Model is not used anywhere but for co-simulation to ensure the hardware is optimized for the application software it is supposed to support. When a tangible model of hardware is reached, it is converted to real hardware specification and validated against sample application software (if exists) via emulation. Via emulation one can ensure the designed NoC possesses the desirable properties. After final model refinements, the hardware is synthesized into a target system.

NoC design and simulation without considering the software is useful but incomplete. Many modeling and simulation frameworks such as Booksim (Dally and Towles 2004), Noxim (Catania et al. 2015), DEVS-NoC (Gholami and Sarjoughian 2012), and NoC-Sim (Jantsch 2006) do not provide models for application software. Instead, designers develop abstractions within the confines of the modeling framework they are working with. This makes the design (and the analysis) unnecessarily complicated –

i.e., software and hardware model abstractions are not systematically separated. Without considering the network and application software, design of the hardware cannot be optimized.

2.3 Network-on-Chip Modeling Metrics

The effectiveness of NoC simulation models should be evaluated via metrics. We have provided a set of these in Table 1. Although there are many metrics for NoC framework evaluation (such as for verification or GUI), we have limited ourselves only to performance and modeling capabilities of the framework. Performance metrics evaluate the simulation framework based on its support for measuring the throughput, utilization, power consumption, flit latency, application run-time, area requirements of the chip, and the frequency required for the application. On the modeling metrics, we seek to measure the capabilities of the framework for levels of NoC abstractions. The levels of abstraction supported by the framework specify NoC model resolutions. Also, some frameworks have support for more NoC elements, which lead to higher resolution models. Other metrics are the levels of support for multi-resolution modeling and co-design.

Table 1: NoC modeling metrics.

Model Performance	Area	Area requirements of the current design
	Frequency	Minimum frequency to satisfy the application needs
	Latency	Average latency of delivering flits
	Power/energy	Average power usage w.r.t frequency
	Throughput	Transfer capability in the network
	Utilization	Utilization of the network capability
Modeling Features	Modeling elements	Structural (HW Components) and behavioral (Network SW)
	Level of abstraction	Supported levels of abstraction in modeling NoC components
	MRM	Environment support multi-resolution modeling and simulation
	SW/HW co-design	Environment capability for hybrid modeling and simulation

Booksim (Dally and Towles 2004) framework is a flit-level simulation engine with support for throughput, latency, and utilization measurement. However, Booksim 2.0 lacks capabilities such as area measurement or power consumption. From modeling point of view, Booksim provides a succinct textual format for specifying the NoC, experiments, and the traffic pattern. The elements it supports are at the flit-level (internal switch components). However, it does not take into account multi-resolution modeling, co-design, and application software. Similar to Booksim is Noxim (Catania et al. 2015). It shares the same capabilities and limitation as in Booksim.

Wormsim (CMU 2005) is a cycle accurate simulator developed in C++. This simulator supports a wide range of topologies, routing algorithms, and switching policies while measuring basic performance characteristics of the network. The traffic generations can be also trace-based in addition to synthetic. Trace-based traffic gives the modeler the option of resembling the real application better than a synthetic workload. This simulator can be coupled with Orion (Kahng et al. 2009) for NoC power modeling. This simulator also lacks multi-resolution modeling, co-design, and application software modeling.

TOPAZ simulator (Abad et al. 2012) supports configuration parameters for various components of the network such as router, topology, and traffic. One can integrate this simulator with full-system simulation tools such as GEM5 (Binkert et al. 2011) for holistic performance evaluation and Orion (Kahng et al. 2009) for power analysis. TOPAZ does not support application software modeling although by integrating with full-system simulators it can.

Finally, DART (Wang, Jerger and Steffan 2011) simulator is unique among the ones introduced in this section for its support for hardware (FPGA-based) execution. This capability substantially reduces simulation runtime compared to other simulators (such as Booksim, NoC-DEVS, and GEM5). The DART

simulator specifies the NoC in flit-level as well and provides accurate performance measures. However, it has limited or no support for hybrid multi-resolution modeling.

3 MRM-BASED CO-DESIGN FOR NOC

Depending on the resolution (object, time, space, and process) at which NoC is modeled (structure and behavior), various components of NoC should be included in the model. Prototypical NoC component belongs to one of three categories: Switch (SW), Link, and Network Interface (NI). NoC also must have *network software* components required for managing data communication (see Section 3.1). We usually add a fourth component known as processing element (PE). This is commonly used in developing System-on-a-Chip (SoC) models. Adding the processing element paves the way for adding *application software* where task execution, scheduling, and communication are necessary. Thinking of NoC as a 4-component system (PE, SW, NI, and Link) is considered a low-resolution view of the system from the object, process, and space resolution points of view. Higher resolution modeling of NoC reveals a number of sub-components in each of these components, new processes (network software), and spatial information, which the 4-component view does not provide. In flit-level abstraction, object resolution is increased by decomposing the switch into input ports, output ports, crossbar, switches, routers, arbiters, and allocators. Also, at this abstraction level, the NI is decomposed into packetizer/depacketizer components. The process resolution is higher as new processes (such as flow control, routing, and allocation) are added to the model. For the hardware, in addition to adding new components and processes for them (such as error checking modules and link reconfiguration models for error handling), spatial information is added so that heat generation and area requirements can also be calculated.

3.1 Multi-resolution Co-design Modeling

Our approach toward multi-resolution network-on-chip design leverages hardware/software co-design. In contrast, existing MRM concepts and methods do not divide a system in terms of software and hardware.. Similarly, co-design methodology does not consider MRM. It is concerned with gradual progression from low-resolution to high-resolution models with much as accuracy and precision as needed can be developed. Co-design is an established process in which software, hardware, and their integration are incrementally and iteratively specified, modeled, and implemented. A modeler cannot know the software or hardware requirements of the system before experimenting using co-simulation. Also, at early stages of system design, neither the hardware nor the software can be specified or implemented in detail. Therefore, low-resolution models of software and hardware are co-simulated and experimented with in order to make early-stage design decisions. Gradually, multiple models at different resolutions are developed which contribute to the making of the final product.

In order to develop multi-resolution model components of NoC, we ask two questions: 1) what can be the resolution of a component? and 2) what relationships model components at different resolutions can have? Co-design divides a system into software and hardware with mapping. What follows are answers to the two questions for the major components of the system (hardware and software).

Moving from Interface, to Capacity, to Flit, and to Hardware abstractions leads to more closely modeling physical NoC systems. Resolution for the hardware and network software is defined based on the abstraction levels introduced in (Dally and Towles 2004). Table 2 categorizes NoC abstraction levels in terms of accuracy and precision resolution levels. As defined, the Interface level model has the lowest level of resolution; its components are specified as objects without having Temporal, Process, and Spatial abstractions. Only a set of objects exchanging data in an ordered discipline. We do not consider this order as temporal specification because of its vast difference with temporal specifications in other hardware abstractions of NoC. The capacity level introduces timing for delivering messages between two nodes and higher resolution objects. The flit level extends the capacity level by introducing processes (for handling flits) and more detailed model of time and objects. Finally, the hardware level, in addition to extending all models with higher resolution concepts, adds chip spatial information to the specification.

Table 2: NoC abstraction levels with accuracy/precision resolutions.

Accuracy/ Precision → Resolutions ↓	NoC Abstraction Levels			
	Interface	Capacity	Flit	Hardware
Object	✓	✓	✓	✓
Temporal		✓	✓	✓
Process			✓	✓
Spatial				✓

Since the Interface level is too high level of a model, we model NoC hardware using the other three (capacity, flit, and hardware levels). Figure 2 (left) depicts a 2-node NoC at the Capacity level. Processing elements (PEs) communicate via packets, which are routed in the network by the switches and the BW/FW (backward/forward) links. Figure 2 (middle) depicts the NoC switch at flit-level abstraction. A comparison between capacity and flit levels shows their differences. At the Capacity abstraction, adding new components including Packetizer, Depacketizer, Crossbar, and Input ports increases **object resolution**. The Capacity abstraction has **temporal resolution**. Switches can exchange flits (every packet contains several flits) in accordance to clock cycles. Furthermore, allocators, routers, and flow controls (i.e., **process resolution)** are supported. The Router, VC Allocator, and SW Allocators represent the network software. Finally, in Figure 2 (right), the input and output ports of a single switch are modeled at hardware level. This example clearly shows the amount of details which go inside each level of NoC abstraction.

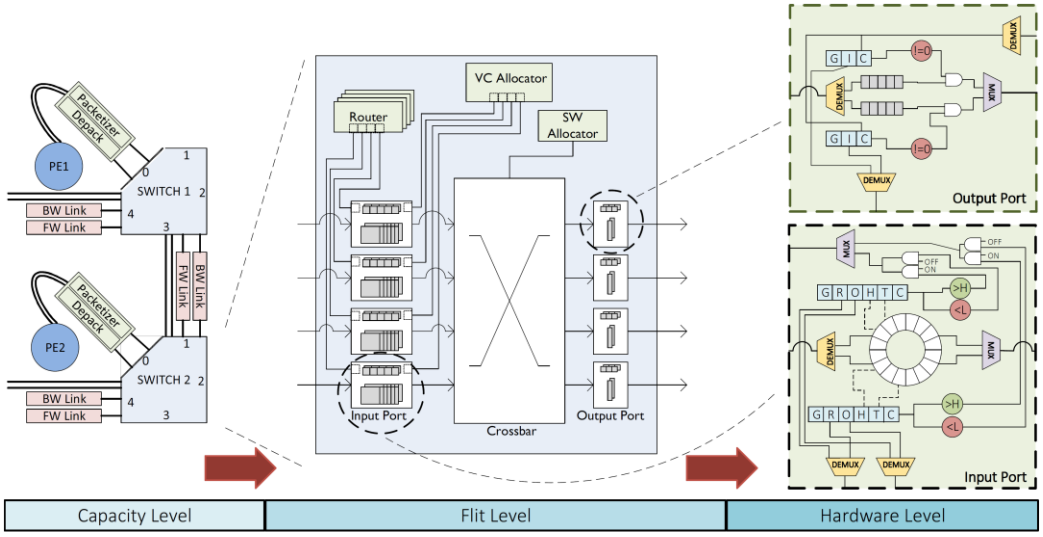


Figure 2: NoC hardware in three resolution levels.

Figure 2 provides not only the view of hardware parts of NoC (switch component in particular) but the network software as well. A switch network software at the capacity level controls the entire operation of the switch (receiving, scheduling, routing of flits and flow control), while the network software at the flit level switch is broken up into many pieces for switch allocation, virtual channel allocation, routing, flow control, etc. The network software would be broken even further at the hardware level.

Mutli-resolution modeling can also lend itself to developing application software. Application software can be modeled as a set of distributed tasks, which transmit random messages, set of tasks with pre-defined communication volume, or a set of tasks with additional specifications on the execution times, threads, dependencies, and function calls (Butler 1994). From process point of view, the software tasks are equipped with dependency, execution times, and probabilistic method calls as we model the

software in higher-resolution. For each abstraction level, a short description is provided along with the specification of resolution in terms of temporal, process, and object resolutions. Network software has no spatial resolution.

We specified resolution for software and hardware in the previous two paragraphs. Now, we should take a bird's eye view at the system. As mentioned, various levels of system model may be considered high-resolution with respect to some aspects of system resolution (time, space, process, and object) and low on some others. This is also the case in multi-resolution NoC modeling. While the software is at its highest resolution (most accurate) for the Interface-level model, the hardware is at its lowest resolution with respect to object and process resolution components. Similarly, the highest resolution of hardware (the bottom row in Table 2) in the hardware abstraction level has the least detailed software. The highest resolution software contains tasks, dependencies, communication volume, threads, and methods. Figure 3 visualizes high-resolution software executing on capacity-level hardware modules (inspired from (Salminen et al. 2009)). At the application level, the software is defined in detail. The tasks and threads are mapped to low-resolution hardware modules (PEs), and then the simulation is carried out. The application level software does not exist in flit-level or hardware-level resolutions (see below).

Multi-resolution modeling can instrumental in developing NoC and SoC models. Moving from one abstraction level to another (e.g., Capacity to Flit) can be systematic using accuracy/precision resolution levels and system hierarchy levels. Different resolution levels can be used to categorize abstraction levels as well as relating these levels to one another. Without having established relationships between the abstraction levels, multi-resolution modeling is vague. For example, Flit NoC abstraction at the I/O System level (level 5 in system hierarchy) can be related to the Object and Temporal resolution levels. The Flit NoC abstraction is defined at the Coupled System level (level 6 in system hierarchy) and Process resolution level (see Figure 2). Comparing models of NoC hardware at different resolutions reveals that moving toward higher-resolution converts atomic models into coupled ones and adds new processes in accordance with the new capabilities needed. For example, an atomic model of the switch at Capacity level is converted to a coupled model at Flit level (Figure 2). The switch model at the flit level possesses subcomponents inside the switch for input port, output port, crossbar, router, and allocators. Along with these new components comes the urge to implement new processes. A router in addition to being a new object, which increases the object resolution of the model, also requires new processes for the task it is supposed to execute. These relationships are useful for designing, validating, and testing new models relative to one another and ultimately to actual NoCs. The network software also changes from one resolution to another. The important point to note here is that network software resolution is always synchronized with the resolution of hardware. The relationship between hardware models (along with the network software) at different resolutions is defined via Object and Process resolution components. As explained above, higher resolution hardware model decomposes atomic models into coupled ones and with the new components comes the need for new processes. This also holds true for network software.

The application software models define relationships based on Object and Process resolutions as well. However, it might not be in sync with the hardware resolution (high-level hardware with low resolution application software). High-resolution application software can be modeled with low-resolution (Capacity level) as shown in Figure 3. The purpose of modeling the application software is to identify accurate benchmark packet traffic among Processing Elements. As a result, the network software is not used with high-resolution hardware model (Flit level). A good approximation of application software traffic) replaces the application software and then the focus of the simulation shifts to high-resolution NoC hardware and software modeling. For this purpose, a transducer model records the packet communication between processing elements. This record is then used in the flit-level hardware as application software model. The record can be used as only feed-forward (the software is reduced to tasks sending and receiving flits using the record) or a feedback-enabled software (tasks are still data sensitive, dependencies exist, and the record captures all of them). One way or another, there is no high-resolution software application anymore. This record is used for generating packets for NoC. The flit-level model

places more emphasis on the hardware while the capacity-level model places more emphasis on the application software. The NoC hardware model takes this even further by excluding data from flits. The communication between components at the flit-level are recorded and is used in the simulation using feedbacks. Similarly, no high-resolution software aspect such as the data in the flits, end-to-end communication, and flit sequence number exists in this stage.

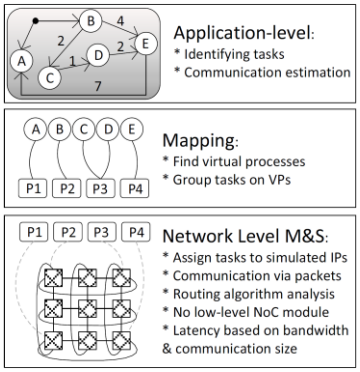


Figure 3: High resolution software at capacity level hardware (low-resolution).

4 NOC MODELS

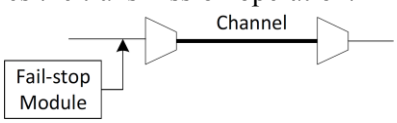
As discussed in Section 2.2, NoC contains two types of software: application software running on the hardware platform and Network Software as a collection of small pieces of software which control the network aspect of NoC. This Network Software has to deal with new problems as NoC moves toward more general software, size increase, energy-awareness, cache coherency, etc. Therefore, NoC cannot be viewed as pure hardware anymore. Software-defined NoCs resemble hybrid systems in which the software plays the crucial role of controlling the hardware. In this light, the NoC is a hybrid system with a traditional hardware and network software modules which control the most dynamic aspects of hardware relative to application software. As shown in Figure 1, simulation plays an important role at the latter stages of NoC design where high-resolution hardware sketches are tested against network and application software. It is possible to capture the hybrid nature of NoC system in one formalism too. In this work we have modeled the NoC in a single formalism and simulated them in one environment. However, the HW and network SW models are still inherently different due to the hybrid nature of NoC.

Now the question would be which resolutions of (network and application) software and hardware can be used with one another. In order to answer this question, we consider link, a simple component of NoC and model it's hardware at three resolution levels, and then show hybrid multi-resolution models can be defined given different abstractions for the network and application software. The resolution of hardware, network software, and application software are defined as follows. Hardware resolutions are stated with the level of abstraction used for modeling it (interface, capacity, flit, and hardware levels). As for application software, we model it at three abstraction levels (i.e., random, distributed, and probabilistic). Network software can be found in various NoC components such as processing elements (cache coherence algorithms), switches (routing/flow control) and links (error checking and retransmission). We use the same hardware resolution levels for network software due to its closeness to hardware.

For the model of the link provided in Section 4.1, we do not consider the link as only a wire which transfers electric charge; instead, based on the resolution the link is modeled at it may contain complementary logic from upstream and downstream nodes for retransmission, error checking, channel reconfiguration, error counter, and fail-stop.

4.1 Multiresolution Link HW/Network SW Models

Figure 4 depicts a low-resolution model of the link with the data (packets) that it can handle. A piece of network software exists for the Fail-stop module which is in charge of disabling the channel if need be. The network software inside the Fail-stop module is simple since the model is at the capacity level. For testing purposes, the role of this module might be to use a distribution function to inject errors. This way the system can be tested under erroneous conditions. The Fail-stop module communicates with the hardware at the channel entry point. That is where the software signal interacts with the hardware component and disables the transmission operation.



Header	Sender's Address Receiver's Address Packet Number
Payload	Data
Trailer	End of packet indicator

Figure 4: Low-resolution link model with the type of data it communicates.

The Fail-stop consists of a hardware logic which is governed by a network software. However, there are components in NoC that are purely hardware, such as buffers. For the low-resolution model of the link, channel is pure hardware while the Fail-safe module is both hardware and network software. The network software will be modeled as the behavior of the hardware component. The hardware component is defined by specifying the state space and input/output ports. The rest of the specification (how inputs are handled, how outputs are generated, how the state changes, etc.) is the network software which specifies how the hardware behaves. At each of the resolution levels (described below) the hardware and software components of the link are extended.

$$S = \overbrace{\{Acitve, Idle\}}^{Phase} \times \overbrace{\vec{\sigma}}^{sigma} \times \overbrace{\{0,1\}^{16}}^{Phit} \quad (1)$$

$$A = \{deliverPhit\} \quad (2)$$

$$\delta_{ext}\big(Idle, \sigma, Phit, e, (in, x)\big) = (Active, \delta t, x) \quad (3)$$

$$\delta_{int}(Active, \sigma, Phit) = (Idle, \infty, \emptyset) \quad (4)$$

$$\lambda(Active, \sigma, Phit) = (out, Phit) \quad (5)$$

$$\psi(Active, \sigma, Phit) = \{deliverPhit\} \quad (6)$$

$$InPorts = \{in\}, OutPorts = \{out\} \quad (7)$$

$$P_p = 1 - (1 - P_e)^N \quad (8)$$

$$\text{Fail-stop Signal} = \begin{cases} 0\ (enable) & P_p < 10^{-12} \\ 1\ (disable) & P_p \geq 10^{-12} \end{cases} \quad (9)$$

As a showcase of a simplified capacity-level model of the link, we modeled the channel using ALRT-DEVS (Sarjoughian and Gholami 2015) in equations 1-7. The model is defined to have a state set (S), input/output ports, external transition (δ_{ext}), internal transition (δ_{int}), and output functions (λ), actions set (A), and activity mapping (ψ). As for the network software operating on the Fail-stop module, we used the concept of BER (bit error rate) (Proakis and Salehi 2007). The software disables the channel if the actual frequency of channel malfunction (bit error) is greater than the packet error ratio (PER) which is characterized by P_p. The Fail-stop module can be developed within a DEVS model and thus simplify its composition with the channel model.

The link model is modeled at higher resolution in Figure 5 (left) by increasing its details in the object and process resolution dimensions. The hardware is extended by flit buffer and additional logic for retransmission and error checking. The network software is also extended with error checking algorithm and retransmission decision making module. The retransmission logic, upon receiving an error signal from Error Checking Module, reconfigures the MUX to pass the buffer data through and enables the buffer to transmit the previously stored data. This way, the data sent in the previous cycle and rejected by the error checking module is retransmitted. In addition to the extensions made to hardware and software, the data which is communicated is changed to flit which is the breakdown of a packet into 8 chunks of smaller data. The higher resolution for the packet and flit structures are shown in Figure 5 (bottom).

Finally, in Figure 5 (right), we have modeled the link at higher resolution. The hardware is extended with additional necessary modules for wires, error counter module, and channel reconfiguration module.

Consequently, the network software is also extended for channel reconfiguration management. The Fail-stop module works based on an error counter module. If the number of errors for the channel becomes greater than acceptable, the channel reconfiguration module orders the fail-stop module to block the channel. In bit reconfiguration scenario, the channel is reconfigured to change the data wires due to bit errors in one of them (Dally and Towles 2004). Thus, the channel is reconfigured to use less or a different set of wires for data communication. In high-resolution, the communication unit of data is still the flit.

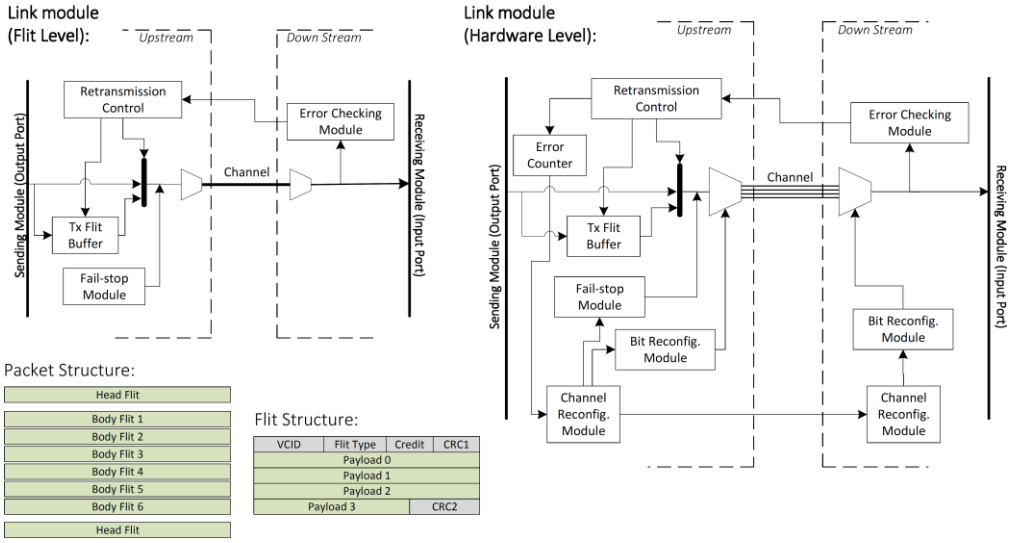


Figure 5: Link models at the Flit and hardware abstraction levels.

4.2 SW/HW NoC Models

In this section, we devise co-design of NoC in three parts: 1) hardware, 2) network software, and 3) application software. The reader must keep in mind that modeling application software is considered as future work. This helps in describing our approach to NoC multi-resolution co-design modeling.

At early stages of design, the model of the hardware (and consequently the network software) are low resolution. For the link model, this is shown in Figure 4 where the link consists of a channel and a simple piece of software in charge of fail-stop module. This is only modeling the link, however, once applied to the switch and network interface components, together they provide us with the capacity-level model of NoC. The model of NoC at this level incorporates the most detailed application software (probabilistic model) on Capacity-level NoC. The reason for this choice is that at early stages of design, the designers need to verify whether their high-level sketches of the hardware and network software are capable of handling the load imposed by the application software. Therefore, the most accurate model of the software is applied and the results analyzed in case changes in the design are necessary.

Furthermore, for the flit-level model of NoC (for which the link is depicted in Figure 5-left), the mid-resolution software (distributed tasks) are used. At this stage, the designer focuses more on the hardware (such as buffers and virtual channels) and network software (such as routing flow control). In order to test the design, they require a model of the software with a wide variety of communication patters. For this, the distributed model is most suitable since it can be configured to imitate a large variety of software applications with various end-to-end communication requirements. This enables designers to verify their design using various scenarios which can be easily produced by the distributed model of the software such as real-time/quality of service constraints and bandwidth requirements. The reason that probabilistic model is inappropriate for this level is reconfiguration inflexibility. The probabilistic model is devised to present a specific (or at least a limited number of) software applications. However, at this stage, configurability and flexibility of software for producing various sorts of traffic and scenarios is desirable.

Finally, at hardware level (presented in Figure 5-right), the circuit is designed at highest resolution. Here, the focus is on testing individual components (pure hardware and network software) to ensure correct behavior under various scenarios. For example, the model of the link in Figure 5-right, can be tested under severe electromagnetic interference to verify retransmission and error checking modules. For this form of testing/simulation, one does not need an actual software to test the component. The only thing needed is random flits (which their data is entirely random) travelling through the channel under electromagnetic interference. Therefore, the hardware-level NoC model uses the random model of application software as the most appropriate one for verification purposes.

5 CONCLUSION

In this paper we described a method for NoC modeling using Multi-Resolution Modeling and co-design in order to help tackle some of the challenges facing Network-on-Chip system design. In this method, NoC hardware and software can be designed in various levels of resolution and later simulated for verification purposes. We described the role of multi-resolution modeling in SW/HW co-design. We observed the domain-specific NoC abstraction levels can be succinctly defined in terms of the domain-neutral I/O System and I/O Coupled System hierarchy levels. We used the proposed framework to develop models for the NoC Link component and described how various resolutions of hardware and software can be coupled for simulation. In the MRM Co-Design framework, we observed that the existing NoC abstractions need to be extended to handle application software and network software to support multiple levels of resolution. Later all these can be used for verification purposes. Future research includes devising a specific NoC application software in various resolutions and couple it with our existing models of NoC system in different resolutions. This can be extended to synthesizing the hardware model on a FPGA and running the application software on it.

REFERENCES

Abad, P., P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J. A. Gregorio. 2012. "TOPAZ: an open-source interconnection network simulator for chip multiprocessors and supercomputers." *Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*. 99-106.

Berekovic, M., H-J Stolberg, and P. Pirsch. 2002. "Multicore system-on-chip architecture for MPEG-4 streaming video." *IEEE Transactions on Circuits and Systems for Video Technology* (IEEE) 12 (8): 688-699.

Binkert, N., B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, Joel Hestness, et al. 2011. "The GEM5 simulator." *SIGARCH Comput. Archit. News* (ACM) 39 (2): 1-7.

Bolotin, E., I. Cidon, R. Ginosar, and A. Kolodny. 2004. "QNoC: QoS architecture and design process for network on chip." *Journal of systems architecture* 50 (2): 105-128.

Butler, J. M. 1994. "Quantum modeling of distributed object computing." *ACM SIGSIM Simulation Digest* 24 (2): 20-39.

Catania, V., A. Mineo, S. Monteleone, M. Palesi, and D. Patti. 2015. "Noxim: an open, extensible and cycle-accurate network on chip simulator." *IEEE International Conference on Application-specific Systems, Architectures and Processors*. Toronto: IEEE.

Chiodo, M., P. Giusto, A. Jurecska, H. C. Hsieh, A. Sangiovanni-Vincentelli, and L. Lavagno. 1994. "Hardware-software codesign of embedded systems." *IEEE Micro* 14 (4): 26-36.

CMU. 2005. "Wormsim 4.2." available: http://www.ece.cmu.edu/~sld/software/worm_sim.php.

Dally, W. J., and B. Towles. 2004. *Principles and practices of interconnection networks.* Morgan Kaufmann.

Davis, P. K., and J. H. Bigelow. 1998. *Experiments in multiresolution modeling.* DTIC Document.

Garland, M. 1999. "Multiresolution modeling: survey & future opportunities." *State of the art report* 111-131.

Gholami, S., and H. S. Sarjoughian. 2012. "Real-time network-on-chip simulation modeling." *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques.* Desenzano, Italy.

Hu, J., and R. Marculescu. 2004. "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints." *Design, Automation and Test in Europe Conference and Exhibition.* 234-239.

Jantsch, A. 2006. "NoCSim: A NoC simulator." *School of Information and Communication Technology, Royal Institute of Technology.* Stockholm.

Kahng, A. B., B. Li, L. S. Peh, and K. Samadi. 2009. "ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration." *Proceedings of the conference on Design, Automation and Test in Europe.* 423-428.

Liem, C., F. Nacabal, C. Valderrama, P. Paulin, and A. Jerraya. 1997. "System-on-a-chip cosimulation and compilation." *IEEE Design Test of Computers* 14 (2): 16-25.

Lis, M., P. Ren, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, and S. Devadas. 2011. "Scalable, accurate multicore simulation in the 1000-core era." *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* 175-185.

Marculescu, R., U. Y. Ogras, L. S. Peh, N. E. Jerger, and Y. Hoskote. 2009. "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (IEEE) 28 (1): 3-21.

Martin, M. K., M. D. Hill, and D. J. Sorin. 2012. "Why on-chip cache coherence is here to stay." *Communications of the ACM* (ACM) 55 (7): 78--89.

Pavlidis, V. F., and E. G. Friedman. 2007. "3-D topologies for networks-on-chip." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15 (10): 1081-1090.

Proakis, J. G., and M. Salehi. 2007. *Digital communications.* McGraw-Hill Education.

Salminen, E., C. Grecu, T. D. Hamalainen, and A. Ivanov. 2009. "Application modelling and hardware description for network-on-chip benchmarking." *IET Computers & Digital Tech.* 3 (5): 539-550.

Sarjoughian, H. S., and S. Gholami. 2015. "Action-level real-time DEVS modeling and simulation." *Simulation: Transactions of the Society for Modeling and Simulation International* 91 (10): 869-887.

Teich, J. 2012. "Hardware/Software codesign: the past, the present, and predicting the future." *Proceedings of the IEEE* 100: 1411-1430.

Wang, D., N. E. Jerger, and J. G. Steffan. 2011. "DART: a programmable architecture for NoC simulation on FPGAs." *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip.* New York: ACM. 145-152.

Wiklund, D., and D. Liu. 2003. "SoCBUS: switched network on chip for hard real time embedded systems." *Parallel and Distributed Processing Symposium.*

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems.* Academic press.

AUTHOR BIOGRAPHIES

SOROOSH GHOLAMI is a Computer Science PhD student at ASU. He can be contacted at sgholami@asu.edu.

HESSAM S. SARJOUGHIAN is Associate Professor of Computer Science & Engineering at Arizona State University and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His research focuses modeling & simulation theory, model composability, distributed co-design modeling, visual simulation modeling, and agent-based simulation. He can be contacted at sarjoughian@asu.edu.