This is a postprint of the paper; the final version of this work has been published in 2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), DOI: https://doi.org/10.1109/ISPASS48437.2020.00015.

BST: A BookSim-based Toolset to Simulate NoCs with Single- and Multi-Hop Bypass

Ivan Perez*, Enrique Vallejo*, Miquel Moreto†‡, Ramon Beivide*†
*University of Cantabria, Santander, Spain
†Barcelona Supercomputing Center, Barcelona, Spain
‡Polytechnic University of Catalonia, Barcelona, Spain

***Abstract*—Network-on-Chips are a critical part of modern multiprocessors and their relevance will grow with the number of cores. The development of future NoC designs relies on detailed simulation models that accurately estimate their performance, power and hardware cost.**

Bypass routers are promising proposals due to their improved performance. Bypass routers reduce latency thanks to a combination of speculation, pre-routing (lookahead routing) and buffer bypass, which also reduce energy consumption by avoiding unnecessary buffer writes and reads. Multi-hop bypass NoCs, known as SMART, even bypass the crossbar of multiple routers in a single cycle. However, publicly available NoC simulators, such as BookSim or Garnet, do not implement bypass mechanisms or do not model them accurately.

In this work, we present *Bypass Simulation Toolset* (BST), a set of tools to accurately simulate NoCs with single- and multi-hop bypass routers. BST combines and extends several simulation tools: an extension of BookSim with state-of-the-art cycle-accurate bypass router models and additional flow control mechanisms; an RTL implementation of multi-hop bypass mechanisms based on OpenSMART; an API to ease a modular integration of the BST NoC simulator in full system simulators; and a set of scripts to automate simulation execution and data collection.

To showcase BST, we i) validate BookSim SMART models with the RTL implementation; ii) compare bypass and traditional non-bypass router models; iii) integrate BookSim in gem5 using the proposed API and compare it with gem5's Simple and Garnet 2.0 NoC models; and iv) present a case study evaluating different combinations of router types and topologies recently proposed for NoCs, highlighting the flexibility of the BST toolset.

The toolset is available at www.atc.unican.es/software.html

***Index Terms*—BST, SMART, NoC, BookSim**

I. Introduction

The exponential growth of information required for and produced by High Performance Computing (HPC) or generated by humans and objects (*Big Data*), creates an explosion of digital data. In this new *data deluge* era, the explosion of connected devices and the necessity to preserve security and privacy will lead to more and more embedded processing. In such scenarios, large-scale multi-cores and heterogeneous manycores are key to provide such high performance and low power demands.

Modern System-on-Chip (SoC) performance highly depends on an efficient communication between processors and with memory. A Network-on-Chip (NoC) delivers such high-performance communication allowing for the integration of a large number of cores on a single SoC. Recent processor designs comprise hundreds [17], [22] or even thousands of cores [40], [44], with a critical role of the NoC design.

Being part of the memory hierarchy, low latency is critical for NoCs. Several techniques have been developed to minimize zero-load router latency, such as speculation [50], pre-routing or lookahead routing [23], and buffer bypass [37], which also reduces energy consumption by avoiding unnecessary buffer writes and reads. Multi-hop bypass mechanisms, such as SMART [35], bypass multiple routers in each hop, relying on broadcast control signals that coordinate the routers.

Selecting NoC parameters such as the NoC topology, links bandwidth, concentration, or the router architecture are critical for the overall NoC performance. The selection of such parameters requires NoC simulation and FPGA emulation with realistic applications and NoC models. Not modeling the NoC in detail may lead to an incorrect (due to deadlocks or data corruption) or suboptimal (due to a under- or over-provisioning of hardware resources) system design.

BookSim is one of the most widely used network simulator tools. The original version implements multiple topologies and routing mechanisms [16]. BookSim2 extends the original version with a focus on NoCs [30], including microarchitectural details, channel delays or traffic models. However, BookSim2 still lacks many mechanisms that are fundamental for current high performance NoCs, such as single- and multi-hop router bypass mechanisms [35], [37], [51], [52], flow control mechanisms such as virtual cut-through (VCT, used in many NoC designs [18], [35], [41], [58]), or bubble-based deadlock avoidance mechanisms [10], [13], [42], [49], [59]. Bypass mechanisms, in particular, are highly relevant for performance as NoCs typically operate in low-load regions. Some bypass mechanisms are implemented in Garnet 2.0 [2], [34], which is integrated in the gem5 simulator [8]. However, publicly available implementations are not cycle-accurate and recent patches are not compatible with the latest gem5 version.

This paper introduces *Bypass Simulation Toolset* (BST), a set of tools used to model, research and develop contemporary NoC designs. Figure 1 depicts the main elements that compose BST. BST contains a new iteration of the BookSim simulator, which incorporates multiple improvements including cycle-accurate models for single- and multi-hop bypass mechanisms. This tool is used for performance evaluation. A Bluespec implementation of multi-hop bypass routers based on OpenS-

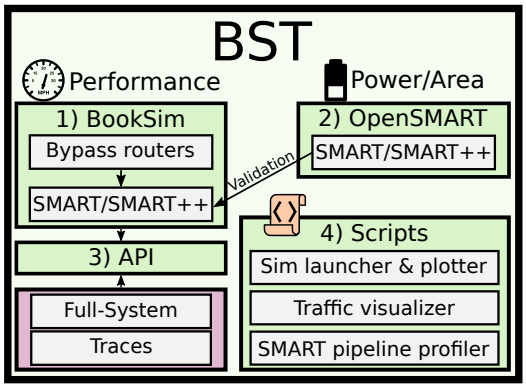


Fig. 1: BST components.

MART [39] provides power and area results, and has been validated with the Booksim model. The toolset is completed with an API to integrate the BookSim simulator with full-system simulators (such as gem5 [8]) and external NoC traffic generators, and a set of scripting tools to automate simulation execution and data analysis.

Specifically, the main contributions of this paper are the following:

- A new iteration of BookSim implementing cycle-accurate single- and multi-hop bypass and other advanced NoC features, and a working RTL implementation of SMART and SMART++ based on the OpenSMART framework.
- A flexible API that allows integrating BookSim with other simulation infrastructures such as cycle-accurate multi-core simulators (gem5), trace-based simulators or complex synthetic traffic models (SynFull [4]).
- A set of scripts to automate executions, represent simulation results, visualize traffic and profile the execution.
- An evaluation of the toolset, presenting its capabilities, performance metrics and a case study which highlights the benefits of bypass routers.

The rest of this paper is organized as follows: Section II introduces the background on bypass routers and NoC simulators. Section III describes in detail the BST toolset to simulate NoCs with bypass mechanisms. Next, Section IV describes the experimental environment and evaluates the proposed NoC simulator accuracy and advanced features. Section V describes the related work and, finally, Section VI concludes this work.

II. Background

BST is a toolset used to model and develop contemporary NoC routers with advanced features such as bypass. This section presents the required background on bypass routers (both single- and multi-hop bypass) and NoC simulators.

A. Bypass Router Networks-on-Chip

NoC routers are implemented with a pipeline with several stages. For example, the traditional router model in [16] employs 5 stages, which implies 5 cycles per hop. Bypass-router NoCs reduce packet latency and power consumption by pre-allocating the switch and skipping pipeline stages,

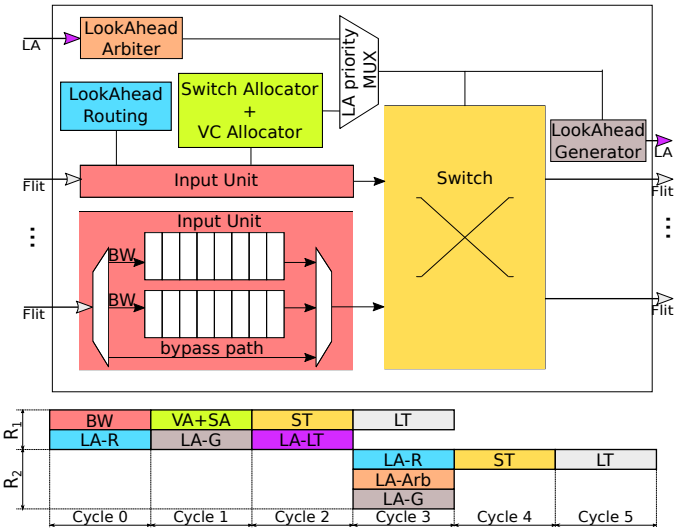


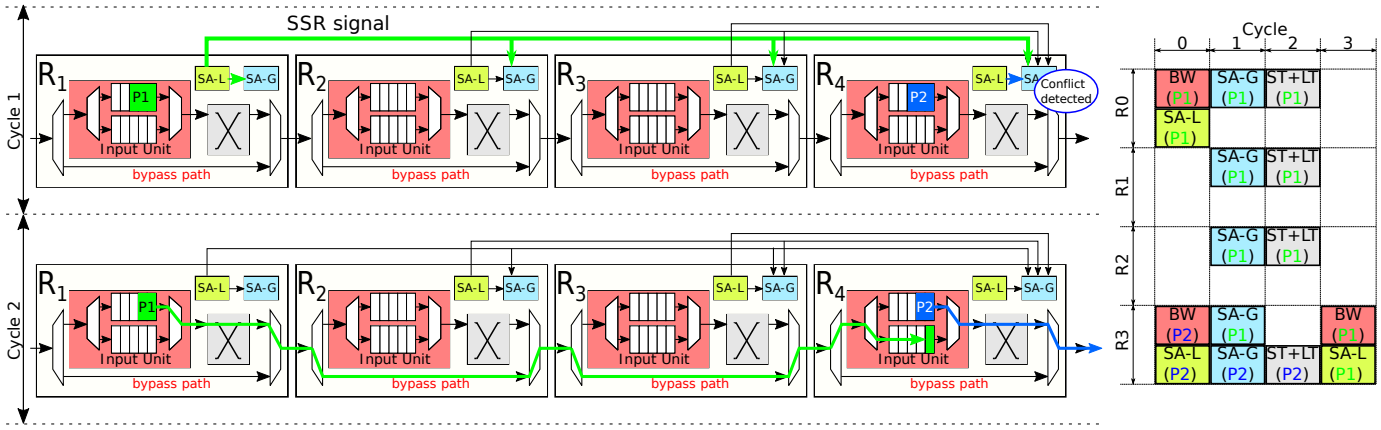
Fig. 2: Single-hop bypass router architecture and pipeline. The pipeline representation depicts two routers R_1 and R_2. R_1 shows the standard pipeline for flits that do not take the bypass, while R_2 depicts the bypass path.

avoiding unnecessary writes and reads to buffers. The routing information of each flit is sent to the next router one cycle before sending the flit, using the *Lookahead* (LA) signal. Lookaheads allow to precompute the route and enable a bypass path to the switch to skip buffering and allocation stages. Depending on the number of hops performed when using the bypass, there are single- and multi-hop bypass mechanisms.

1) Single-hop bypass: Kumar et al. [37] introduced single-hop bypass-routers. Figure 2 depicts the router architecture and pipeline of a generic bypass router, including a bypass path in the input unit. The architecture is similar to a traditional virtual channel (VC) router with LookAhead Routing (LA-R pipeline stage), input buffers (Buffer Write, BW), VC and switch allocators (VA and SA) and a switch (Switch Traversal, ST). The additional units required to handle Lookaheads are a *LookAhead Generator*, a *LookAhead Arbiter*, and a multiplexer *LA priority Mux*. These units create and forward LAs for flits that advance to the ST stage, arbitrate among LAs and resolve conflicts between LAs and local flits. As depicted in the pipeline in Figure 2, bypass routers may reduce per-hop delay to two cycles, for ST and link traversal (LT) stages.

Traditional bypass router implementations require the destination buffer (about to be bypassed) to be empty, in order to forward the packet. With this condition, a large number of VCs is required to obtain significant performance benefits. More recently, Non-Empty Buffer Bypass (NEBB) routers [51] is capable to bypass non-empty buffers, and achieve competitive performance with few VCs.

Three variants of NEBB have been introduced, based on the flow control mechanism used: i) *NEBB-WH* only bypasses non-empty buffers when packets are single-flit; ii) *NEBB-VCT* bypasses non-empty bufferes for single- and multi-flit packets, but only if there is room for the whole packet in the downstream router, even if the bypassed buffer is empty; and



(a) Architecture, showing two consecutive cycles (b) Pipeline

Fig. 3: SMART multi-hop bypass router architecture (left) and pipeline displaying the transmission of two packets (right).

iii) *NEBB-Hybrid* combines NEBB-WH and NEBB-VCT to maximize bypass utilization and provide the best performance.

2) Multi-hop bypass: Krishna et al. introduce multi-hop bypass mechanisms in SMART [35]. Figure 3 shows its architecture and pipeline. SMART implements a divided Switch Allocator (local, SA-L, and global, SA-G) and relies on broadcast control signals denoted Switch Setup Requests (SSRs). Routers broadcast SSRs for flits that win SA-L, and downstream routers configure their bypass, which skips the router completely. After this, flits are forwarded and they bypass all routers in the multi-hop, until they find a router that had a conflict in SA-G and did not set up the bypass path. Packet bypass in SMART requires three cycles per multi-hop: SA-L, SSR broadcast and data forward.

Like classic bypass-routers, SMART requires an empty destination buffer when allocating a VC to a packet, which implies that a large number of VCs is required for good performance. More recently, SMART++ [52] combines SMART with NEBB to provide efficient multi-hop bypass with just one or few VCs.

B. NoC Simulators and Current Bypass Support

This section presents the required background on NoC simulation tools and their relation to bypass routers, specifically BookSim2, Garnet and OpenSMART.

1) BookSim2: BookSim2 [30] is a flexible and detailed network and NoC simulator with a multitude of topologies and routing algorithms. It includes a cycle-accurate model of a traditional five-stage pipeline router [16], with multiple configurable elements such as allocators or buffering policies, and router optimizations like LookAhead routing and speculative switch allocation. Besides, it includes a variety of synthetic traffic patterns to feed the network. It also supports request-reply traffic models, which are characteristic of cache coherence communication. When using finite input queues, it can be used to model the performance of multi-core cache hierarchies with limited Miss Status Hold Registers (MSHRs), in which the injection rate is dependant on the network latency. However, BookSim2 does not include neither single-hop nor multi-hop bypass-router models.

BookSim has been used in a multitude of diverse studies, many of them mentioned in [30]. For example, in [4], it is used to develop and validate SynFull, a simulation methodology consisting of generating synthetic traffic models from traces that mimics the spatial, temporal and burstiness distribution of cache coherence traffic of full-system (FS) simulated applications. In [42], a deadlock avoidance mechanism for torus topologies called Flit Bubble Flow Control (FBFC) is proposed, primarily oriented to NoCs as it does not require multiple VCs or deep buffers like other alternatives, such as dateline [16] and Bubble routing [10]. In addition, BookSim is used as the interconnect model of GPGPU-sim [5], a GPU simulator that runs CUDA applications. Even though several works [12], [19], [29] have integrated it with the gem5 simulator [8], there is not a public distributed version of the integration interface which is compatible with the latest releases of gem5.

2) Garnet and Garnet2.0: Garnet [2] is a NoC model inside the gem5 full-system simulator. In previous versions of gem5 there were two versions of Garnet: *fixed-pipeline* and *flexible-pipeline*. *fixed-pipeline* modelled cycle-accurately a traditional five-stage pipeline router with a credit-based virtual channel flow control. By contrast, *flexible-pipeline* does not model the pipeline, implementing all router functions in a single cycle. The router latency is customized by adding a latency to each forwarded packet. Because of this implementation, dependencies of packets in different pipeline stages are not modeled in detail, which may hide NoC design flaws and/or provide inaccurate results.

Garnet2.0 deprecates the detailed *fixed-pipeline* model, but introduces additional routing algorithms and topological support for creating custom routings, replacing the previous deterministic table-based routing with shortest-paths. Bypass routers in Garnet are modelled by simply reducing the latency of the *flexible* router. Additionally, there exists a patch to include a SMART-1D NoC model based on Garnet2.0's router. However, the patch is relatively old and, at the time of writing, it is not compatible with the Garnet code distributed in recent versions of gem5.

3) OpenSMART: OpenSMART [39] is an open-source tool to generate NoC synthetizable Verilog code. It provides models written in Bluespec and in Chisel. The Bluespec design includes both implementations of a conventional single-cycle and a SMART router, limited to single-flit packets. Since Bluespec code is compiled to Verilog, it allows to estimate area, energy and timing results for an implementation that targets actual FPGAs and ASICs, enabling design-space exploration.

However, actual tests of the Bluespec SMART model[1] bring up implementation issues that make packets reach incorrect destination nodes or lose packets, making the current model unsuitable for actual research or design work.

III. BST Tools

This section presents the components of BST. First, we describe the new iteration of BookSim with bypass support. Next, we detail the API of BST with emphasis on the BookSim integration with gem5. Finally, we go into the SMART++ implementation in OpenSMART.

A. BookSim with Single- and Multi-Hop Bypass Routers

BST includes an enhanced version of BookSim which adds several new router models to faithfully model different characteristics of contemporary competitive NoCs. Besides the three original router models in BookSim, BST adds four main new models: two models for single-hop bypass (base and NEBB-hybrid) and two models for multi-hop bypass (SMART and SMART++). Additional intermediate implementations are also included. In this section we first detail the contributions in flow control and deadlock avoidance mechanisms. Next, we mention changes required to support bypass routers, followed by single- and multi-hop bypass implementations, which are the most significant improvements.

1) Flow control mechanisms: The conventional input-queued router in BookSim only supports WormHole (WH) and Virtual Channel flow control mechanisms. The following alternative flow control mechanisms have been implemented in BST: Virtual Cut-Throught (VCT) [32], Bubble flow control [10], and Flit Bubble Flow Control Local (FBFC-L) [42]. The latter two are specifically designed to avoid deadlocks in k-ary n-cube (torus) topologies, using VCT or WH respectively. Single-hop bypass routers support most of these mechanisms, while multi-hop bypass routers use VCT. The implementation of the flow control mechanisms rely on credits to monitor the buffer occupancy of adjacent routers.

2) Bypass routers: The implemented bypass-router models are cycle-accurate and faithful representations of the microarchitectures described in [35], [37], [51], [52].

The original router model of BookSim models each pipeline stage in two phases, *evaluation* and *update*, in order to avoid the propagation of unintended information between stages within a cycle. The evaluation phase performs the logic of each stage without writing the inter-stage signals to avoid interfering with the next pipeline stage, while the update phase modifies these signals. The bypass-router models in BST follow a different strategy to correctly define cycle boundaries: the pipeline stages of each router are executed sequentially in reversed order, i.e. link traversal, switch traversal, arbitration stages, route computation, flit reception. Pipelines of different routers are separated by channels that introduce a delay, avoiding unintended data propagation in a single cycle. This strategy simplifies the code at the cost of not having customizable pipeline stage delays except for link traversal.

Bypass routers support most of the configurable parameters of BookSim. Besides parameters that select the router model, Table I collects the most relevant new parameters.

3) Single-hop Bypass: The enhanced BookSim simulator in BST has five different bypass router models divided in two categories. Classic bypass, which requires empty destination buffers, has two variants: with and without LookAhead arbitration. There are also three variants of NEBB: NEBB-WH, NEBB-VCT and NEBB-Hybrid.

Lookaheads (LAs) and LA channels are properly modeled in the network, following the same scheme of flits and flit channels in BookSim.

Among the specific configuration parameters, three are the most relevant: *disable_bypass*, which deactivates the bypass, is useful to obtain reference results; *lookaheads_kill_flits*, which indicates the priority policy of the LA priority Mux (see Figure 2); and *guarantee_order*, which preserves an ordered delivery of packets as required by some coherence protocols.

Dynamic input buffer management (shared buffers, implemented as DAMQs, [57]) has been adapted to support NEBB-Hybrid. Hybrid combines WH and VCT virtual channel allocation, and packets may have *bubbles* caused by channel interleaving. Buffer management must be aware of which flow control is used by each bypassed packet to prevent deadlock. When bypassing a packet using VCT, buffer slots are reserved for the whole packet, i.e. credits are reduced by the packet size. This guarantees that any packet that starts advancing to a buffer will have space for it, regardless of any bubble. In contrast, when transferring a packet using WH, credits are reduced flit by flit (both for bypass and non-bypass paths).

Besides the standard statistics of BookSim, simulations using single-hop bypass routers provide the following new metrics: *bypass utilization*, as the ratio of the number of hops that use bypass paths over the total number of hops done; *buffer* and *crossbar conflicts*, as the number of times that these resources are not available for flits or LAs; *number of SA winners killed*, as the number of local flits that win switch allocation and are later killed due to a conflict with LAs in the LA priority Mux.

4) Multi-hop Bypass: The BST version of BookSim includes two main multi-hop bypass routers representative of SMART [35] and SMART++ [52]. Partial versions of SMART++ evaluated in [52] are supported too. Each model implements VCT flow control. However, SMART++ is the only mechanism that guarantees consecutive reception of packet flits, since it allocates buffers, switches and bypass paths on a packet by packet basis, instead of flit by flit.

[1]Using the most recent commit: d4f5095 from 4 September, 2019

TABLE I: Representative parameters related to bypass routers in BookSim from BST.

BookSim parameter	Description
Single-hop Bypass	
bypass_empty_vc	Specifies if empty destination VCs are required to forward packets
disable_bypass	Used to enable or disable router bypass.
lookaheads_kill_flits	Specifies the priority used in the *LA priority Mux*: Priority to LAs or to local data flits.
guarantee_order	Avoid data reordering caused by buffer bypass requests that target the same output as a packet in a local buffer.
Multi-hop Bypass	
smart_type	Specifies the version of SMART. The router model must be SMART (router=smart)
smart_max_hops	Sets the maximum number on hops within a multi-hop (HPC_{max})

In these models, flits can traverse multiple routers and links in a single cycle. However, each channel in BookSim requires a minimum delay of one cycle, making impossible the traversal of multiple links in a single cycle without modifying the structure of BookSim to a large degree. Instead, BST Booksim accounts for the delay of a multi-hop link traversal in the sender router. In a multi-hop path, the sender router calls the reading function of the next router that evaluates whether the bypass is enabled or not. If it is enabled, the same procedure occurs with the subsequent router, until finding a disabled bypass or reaching the end of the multi-hop. The flit is directly saved in a pipeline register in the last router, prepared for buffer write (BW) in the following cycle.

SSR channels are not modeled due to the complexity that they introduce in the topology creation. Instead, SSR signals are placed directly in the routers within each multi-hop, after wining the local switch allocator. This implementation is cycle-accurate and functionally equivalent to the actual proposed implementation.

SMART simulations employ two additional parameters to select the version of SMART and to define HPC_{max}, as indicated in Table I.

In addition to the statistics given by BookSim, SMART simulations includes: number of multi-hops; length of each multi-hop; and bypass utilization as the ratio of flits bypassed over the total number of flits transmitted by each router.

B. Simulator Integration API

BST provides an API to simplify the integration of BookSim with other simulation environments, such as gem5. The API abstracts the internal structure and presents simple commands to interact with the simulation tool. This section explains the integration with gem5, assuming the Ruby memory hierarchy model is employed. Ruby defines multiple modules for each element in the memory hierarchy and the coherence protocol, and they communicate using the NoC.

The API supports the compilation of BookSim as a library, instead of integrating its code into the other simulator like in GPGPUSim [6]. The main advantage of linking BookSim as a library is the automatic synchronization from the standalone to the integrated version. Additionally, this decouples the development and maintenance of both tools, and speeds up the (time-consuming) compilation process when changes are done only to one simulator.

Some code changes required to support the API and the compilation as a library include a new namespace that encapsulates the complete BookSim code; modified compilation parameters; changes in the traffic manager functions; and a new class to interact with the API functions.

1) API functions: Besides the constructor and destructor of the object (*BookSimWrapper*), the API consists of four functions called from gem5:

a) GeneratePacket: Sends a new packet if there is enough space in the injection queue. This function returns the packet identifier of the packet created. This identifier is used to create a dictionary to save the message data and destination of the packet in the Ruby domain (the memory hierarchy module from gem5).

b) RunCycles: Execute the number of cycles given. A value larger than 1 cycle defines a faster frequency in the BookSim NoC.

c) RetirePacket: Retires a packet from one of the ejection queues of BookSim. This function returns an struct that contains the packet identifier, the packet class (virtual network), and statistics data. The packet identifier is used to gather the message information from the dictionary entry previously created by *GeneratePacket*. Then, the message is enqueued in the correct Ruby switch and port.

d) CheckInFlightPackets: Checks whether there are flits inside the network, in order to schedule a new network simulation event for the next cycle. This is useful to reduce computation when the network is empty.

2) Topology mapping: Unlike the integration of the gem5 modules Simple Network and Garnet 2.0, BookSim ignores the topology defined in the parameters of gem5. The topology modelled in BookSim is defined by its own parameters, the same ones as when running in isolation. However, the user must be aware of the mapping of Ruby tiles to BookSim nodes.

A Ruby tile comprises cache controllers from different levels of the memory hierarchy and may include memory directory and DMA controllers. Figure 4 shows an example with a 16-tile network arranged in a 4×4 mesh. There are two types of tiles represented in yellow and red. Light yellow tiles have a private bank of L1 cache and a shared bank of L2 cache connected to a Ruby Router (RR). Dark red tiles also include memory directory (DIR) and DMA controllers.

The topology configuration from gem5 specifies a tile organization in Ruby, assigning a consecutive tile ID to each Ruby Router. The actual gem5 topology is irrelevant, since BookSim only employs the tile ID to map the resources.

A mapping algorithm has been adapted to take into account relative location of each resource in the NoC placement.

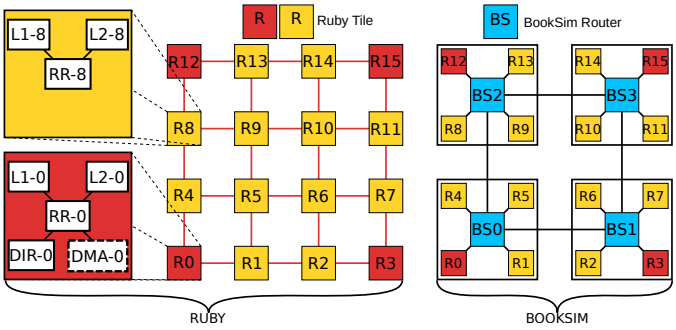


Fig. 4: Example of 16-tile network mapped as a 2×2 mesh with concentration 4 in BookSim.

Figure 4 presents an example in which tiles are arranged in a 2×2 mesh with concentration 4. A naive mapping using increasing tile index would map tiles with consecutive IDs to the same router; hence, the first router ($BS0$) would include two Ruby tiles with attached memory controllers ($R0$ and $R3$), altering the tile placement in the chip. The mapping of tiles to BookSim nodes implemented in BST preserves the the location of the tiles with memory controllers in the corners of the chip. To achieve this arrangement, the node mapping from the concentrated mesh topology in BookSim has been modified: tiles 0, 1, 4 and 5 are connected to the first router ($BS0$), and so on.BST includes support for the most frequent topologies: square meshes, tori and flattened-butterfly, with and without concentration.

Despite the incurred complexity, this solution provides flexibility. For example, it allows users to define a tile per cache, memory or DMA controller, and interconnect them as desired in BookSim's topology.

C. OpenSMART++ RTL Implementation

BST includes a Bluespec System Verilog implementation of SMART and SMART++ derived from OpenSMART [39]. This model allows to generate valid RTL code and to estimate area, power and frequency metrics. Like OpenSMART, the implemented models only support single-flit packets.

The pipeline model presented in OpenSMART significantly differs from the original proposal in SMART. For example, ST is implemented in the second stage after winning SA-L, instead of the third stage as discussed previously (see Figure 3b). This complicates the comparison between the functional models in BookSim and the results from Bluespec. Additionally, initial testing of the models presented execution errors, with packets that were missed or not delivered to their correct destination, preventing any productive use of the tool[2].

To be able to compare the BookSim and Bluespec models, we have reorganized the router stages. Our implementation follows the original organization from SMART depicted in Figure 3. Additionally, our model corrects some errors and provides working models that deliver all packets to the correct destinations. Some specific changes are detailed next.

Several issues with Bluespec rules prevented successful compilation, specifically cyclic dependencies of rules. Rules are the main coding block of Bluespec to describe how data is moved from state to state. The Bluespec compiler detects data dependencies and schedules rules in an appropriate order, which may be explicitly given in the code. Cyclic dependencies in these rules may prevent compilation.

The following changes simplified the implementation and mitigated the identified issues. First, OpenSMART employs a custom library to implement FIFO structures. These modules have been replaced with the Bluespec built-in modules FIFO and FIFOF (the latter including explicit *full* and *empty* signals). These changes are implemented in the Credit Unit, Input Unit, Smart Flag, Smart Router and Traffic Generator Units. Similarly, the CReg module that implements an Ephemeral History register [55] has been replaced by explicit combinational logic, which is less prone to generate cyclic rule dependencies when modifying code.

The implementation of SMART++ essentially required modifications to the VC Selector (VS) in OpenSMART. The original SMART VS implements a pool listing free VCs in the neighbor router. An empty VC is extracted when a packet wins SA-G, and inserted when a credit is received. The implementation is quite simple as SMART requires empty destination VCs to forward packets, ignoring buffer depth and occupation. In contrast, SMART++ employs buffer depth and occupation. Flow control credits, already implemented in OpenSMART, are leveraged to monitor the available space in each VC. The increase in logic is very moderate.

D. BST Scripting Tools

Finally, BST provides a set of python scripts used to automate the most frequent tasks with the simulation tools. Launching scripts are used to submit multiple simulation jobs to a compute cluster job scheduler. The provided scripts are specifically adapted to the SLURM workload manager [60]. Other scripts are used to ease simulation data recovery, visualize the evolution of the router pipelines, and plot the results.

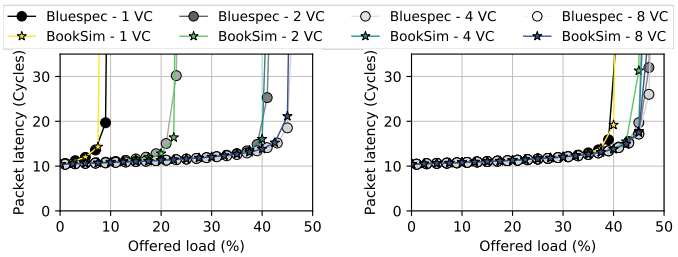
IV. Evaluation

This section presents an evaluation of the BST tools. First, we validate that the SMART models are cycle-accurate. Next, we compare the previous BookSim router model with the new single- and multi-hop bypass router models. Then, we show gem5 results obtained with the developed BookSim API. Finally, we present an example use cases that evaluates different topologies and the impact of bypass on each case.

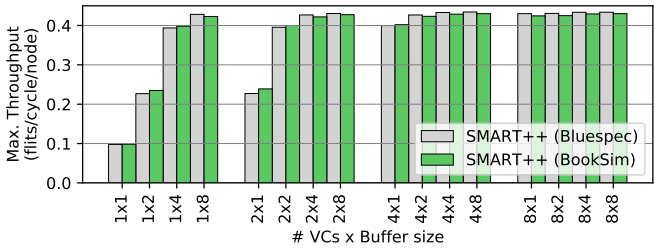
A. SMART Model Validation

This section demonstrates the accuracy of the multi-hop bypass models implemented. We compare the results obtained using the BookSim simulator and using OpenSMART RTL simulations with the Bluespec System Verilog simulator. The network simulated is a rectangular 4×8 mesh with

[2]Using the most recent repository version at the time of writing, commit d4f5095.



(a) Buffer size: 1 slot. (b) Buffer size: 4 slots.



(c) Maximum throughput.

Fig. 5: Packet latency and throughput of the SMART++ models implemented in Bluespec and BookSim.

$HPC_{MAX} = 4$. Both simulators generate random uniform traffic with single-flit packets, due to the limitation of the OpenSMART implementation mentioned in Section II-B3.

BookSim and the Bluespec simulator account injection and consumption latency differently: several constant cycles in both processes are only modelled in Booksim. For this reason, raw results differ in a constant value. We have identified this discrepancy and adjusted the results of OpenSMART to match the zero load packet latency of BookSim.

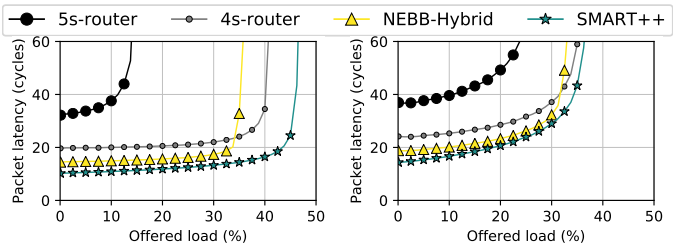
Figure 5 shows average packet latency (Figures 5a and 5b) and maximum throughput (Figure 5c) with both BookSim and OpenSMART models. It considers multiple combinations of VC buffer depths (1 and 4 slots in latency plots; 1 to 8 slots in the case of throughput) and number of VCs. The results use the SMART++ model; SMART results follow the same trend, but are omitted for the sake of space.

Despite comparing different router and bypass implementations in different simulation tools, the results of both models are nearly the same. Packet latency is identical in both models until reaching saturation, as shown in Figures 5a and 5b. After saturation, there are small differences; the maximum throughput difference is 5%, when using 2 VCs of 1 flit.

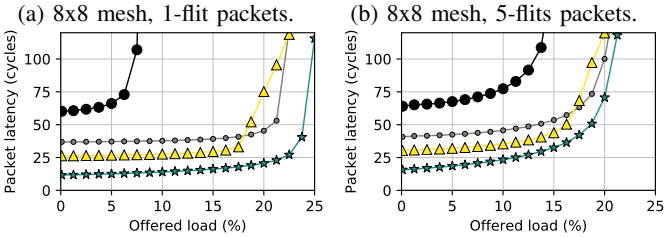
B. Bypass Router Model Comparison

In this section, we evaluate the router model of BookSim and the most advanced versions of bypass routers, i.e. NEBB-Hybrid single-hop bypass and SMART++ multi-hop bypass respectively, both of them implemented in BookSim. The networks simulated is an 8×8 and a 16×16 mesh with a single VC and 20-flit input buffers. Results are obtained with a random uniform traffic injection pattern; other traffic patterns produce similar results and are omitted for brevity.

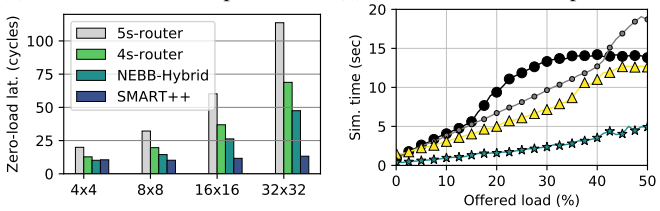
Two models from BookSim are compared: *5s-router* stands for BookSim's 5-stage router, which follows the traditional



(a) 8x8 mesh, 1-flit packets. (b) 8x8 mesh, 5-flits packets.



(c) 16x16 mesh, 1-flit packets. (d) 16x16 mesh, 5-flits packets.



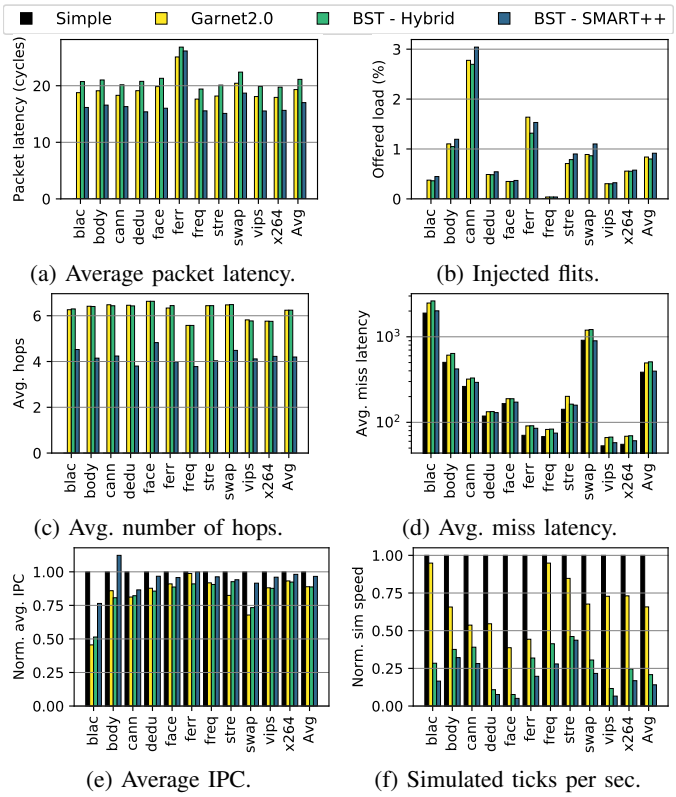
(e) 1-flit zero-load latency for different mesh sizes. (f) Simulation runtime 8x8 mesh with 1-flit packets.

Fig. 6: Evaluation of 5- and 4-stage routers (5s-router and 4s-router), NEBB-Hybrid and SMART++ using BookSim. Latency results for different network and packet sizes.

pipeline in [16]; *4s-router* reduces the pipeline depth to 4 stages, employing LookAhead routing and speculative VA, but no buffer bypass. These two models rely on WH flow control. The models with single-hop bypass is denoted *NEBB-Hybrid*. In this model, the network interface (NI) generates a LA to allow the bypass in the injection router, at the cost of an extra cycle delay in the injection link so the LA arrives one cycle before the data flit. Bypass has priority over local flits. The multi-hop bypass model is *SMART++*, and it employs $HPC_{MAX} = 8$. In this case, local flits have priority over SSRs to avoid pathological performance issues [35].

Figure 6 shows average packet latency, for packets of 1 flit (Figures 6a and 6c) and 5 flits (6b and 6d). In the 8×8 mesh, the single-flit zero load packet latency of *5s-router* is reduced by 39.47% with *4s-router*, 55.71% with *NEBB-Hybrid* and 68.06% with *SMART++*. With packets of 5 flits, this reduction is 34.7%, 48.55% and 60.52% respectively. Larger network sizes increase the benefit of speculation and bypass, as observed in Figures 6c and 6d. Saturation throughput, detected by the point in which latency skyrockets, also benefits from speculation and bypass; interestingly, the result for *4s-router* without bypass is slightly better than *NEBB-Hybrid*.

Since NoCs load is typically very low, zero-load latency is one of the most relevant metrics. Figure 6e combines base latency results for different router models and networks. BST allows to quantify the benefit of speculation and bypass.



(a) Average packet latency. (b) Injected flits.

(c) Avg. number of hops. (d) Avg. miss latency.

(e) Average IPC. (f) Simulated ticks per sec.

Fig. 7: Evaluation of Simple, Garnet2.0 and BST in gem5.

Interestingly, for small networks (4×4) the use of multi-hop bypass is not beneficial from single-hop bypass. This is explained because average distance is very small and the multi-hop bypass pipeline has an additional stage. By contrast, in large networks the benefit of multi-hop bypass is enormous.

Figure 6f shows the simulation time for the different router models and offered load in an 8x8 mesh with 10,000 simulated cycles. Logically, the time grows with the offered load, linearly before reaching the saturation point and exponentially later, until injection buffers become saturated. Both bypass models are faster than the original models of BookSim2 and SMART++ is the fastest model because each of its packets performs a lower amount of hops.

C. BST Integration with gem5

This section presents a test and evaluation of gem5 with the BookSim API introduced in Section III-B. We simulate a system that consists of 64 ARM *DerivO3CPU* running at 3GHz. The gem5 Ruby memory model is enabled, using the *MOESI_CMP_directory* coherence protocol. The memory hierarchy implements 32KB L1I and 64KB L1D private caches, both with associativity 2. The L2 cache is shared and distributed into 64 banks of 256KB, 16MB in total. There are four memory controllers located at the corners of the mesh. The interconnect frequency is 3GHz and four NoC models are evaluated: Simple Network, Garnet 2.0 (both part of gem5), BST NEBB-Hybrid and BST SMART++. In Garnet 2.0 we set a router latency of 2 cycles including the link latency, similar to the NEBB-Hybrid latency. We use the default buffer configuration of the Simple Network; the other detailed models implement 3 virtual networks of 1 VC with buffers of 10 slots each, and a DOR-XY routing mechanism.

We run multiple PARSEC suit benchmarks [7] in Full-System mode. We use the simlarge input set and 64 software threads in all of them. Due to the long simulation times, the results correspond to the first 300 million simulated cycles.

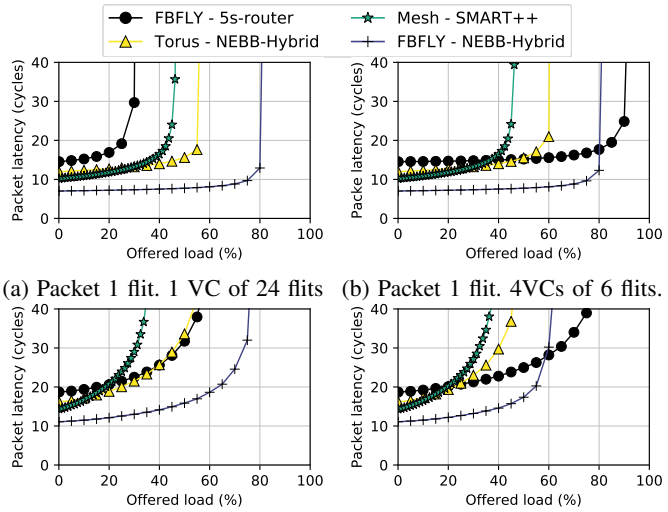
Figure 7 shows different metrics of the executions, including an average for all the benchmarks. Notice that Simple does not provide NoC statistics. In general, the injected traffic load (Figure 7b) is extremely low, as observed in the experimental results of different studies [27], [47]. In this situation, the most relevant parameter for performance is the base latency of the system. Canneal with BST SMART++ produces the highest offered load, of only 3.03%. BST SMART++, with the lowest packet latency (Figure 7a), injects more flits, which translates to lower miss latency (Figure 7d) and higher IPC (Figure 7e).

The average number of hops per packet (Figure 7c) are mostly unaffected by the benchmark. They are practically identical between Garnet2.0 and BST Hybrid, and around 2 hops lower in SMART++ because of multi-hop bypass. The average packet latency results (Figure 7a) are close to the zero-load latency, as a result of the low injection rates. Ferret is the only exception, with a higher packet latency in the three models, with similar results among them despite the hop count reduction in SMART++. In this case, the latency increase comes from packets waiting in injection queues, i.e. before packets are injected into the NoC. The miss latency (Figure 7d) and IPC results (Figure 7e) show the effect of packet latency in the overall performance. The lower the latency, the lower the miss latency and the higher the IPC. Excluding the unrealistic Simple, the BST SMART++ model achieves the best results.

The simulation speed is presented in Figure 7f normalized to the speed of Simple, which is the fastest due to its higher level of abstraction. Garnet2.0 presents a significant slowdown, with an average normalized execution time of $0.66\times$ despite being designed as part of the Ruby memory system, so it is clear that a detailed NoC modelling has a high impact on the overall simulation time. The BST models are slower, owing to their highly detailed implementation. Additionally, unlike the event-driven gem5, BookSim is time-driven, which complicates the synchronization between the simulators. A low injection rate increases the impact of the overhead of BST, suffering the complete NoC simulation overhead for only a few in-flight packets. Specifically, applications with very low offered load in Figure 7b (blackscholes, dedup, facesim, freqmine, vips, x264) present a larger overhead when switching from Garnet to BST in Figure 7f. The average normalized execution time of BST is $0.21\times$ in Hybrid and $0.14\times$ in SMART++.

D. Case Study: Topology Evaluation

This section presents a study of topology selection using the BST toolset. The topology and router architecture are the main factors that define the base latency of a NoC. Mesh is the most commonly used topology for NoCs. However, other topologies such us tori and the flattened butterflies (FBFLY)



(a) Packet 1 flit. 1 VC of 24 flits (b) Packet 1 flit. 4VCs of 6 flits.

(c) Packet 5 flits. 1 VC of 24 flits (d) Packet 5 flits. 4VCs of 6 flits.

Fig. 8: 8×8 mesh, torus and FBFLY topologies evaluation.

have been studied as alternative designs for NoCs [33]. In this example, we compare different combinations of topologies and routers proposed for NoCs. The configurations evaluated are a flattened butterfly with 5-stage router (*FBFLY - 5s router*); folded torus with NEBB-Hybrid (*Torus - NEBB-Hybrid*); mesh with SMART++, (*Mesh - SMART++*); and flattened butterfly with NEBB-Hybrid (*FBFLY - NEBB-Hybrid*). Note that multi-hop bypass makes no sense in a FBFLY, since only one hop occurs per dimension. All the simulated networks have a size of 8×8 and two configurations of VCs are used: 1 VC of 24 flits and 4 VCs of 6 flits. To simplify the analysis, the link latency of the folded torus and flattened butterflies is 1 cycle independently of the link length and HPC_{MAX} in SMART++ is 8. The traffic generator serves random uniform traffic with packets of 5 flits.

Figure 8 shows average packet latency for all configurations. The zero load latency of each topology and router combination is the expected. *FBFLY - 5s router* presents the highest base latency despite the high degree of the routers ($7 + 7 = 14$ ports per router), because of its high per-hop latency. *Torus - NEBB-Hybrid* and *Mesh - SMART++* have similar base latency, but *Torus - NEBB-Hybrid* achieves higher throughput with the same router degree (4 ports maximum). Overall, the best combination in terms of performance is *FBFLY - NEBB-Hybrid* but with a high cost given the router degree 14.

In terms of throughput, there are several worth mentioning observations. *FBFLY - 5s-router* is very sensitive to Head of Line Blocking (HoLB) issues, since the configuration with 4 VCs in Figure 8b triples the throughput of 1 VC in 8a. The other three configurations show a very small dependence on the buffer count, since they rely on Non-empty buffer bypass.

V. Related Work

A. Bypass Router Support

Section II-A presents the minimal background on bypass routers. There exist actual NOC architectures using bypass, such as SCORPIO [18] or SWIFT [36]. Alternative bypass mechanisms have been implemented in different proposals, such as Token Flow Control [38], SWIFT [36] or Shortpath [53]. These mechanisms are variations of the general idea, and have not been implemented in BST.

Similarly, different proposals modify the implementation of SMART multi-hop bypass presented in Section II-A2. For example, the original paper by Khrisna et al. [35] introduces several variants, such as the use of SMART_2D, consumption bypass, low-load router stage bypass, and different priority policies. Other variations include improvements to SSR arbitration in SHARP [3], the SSR network in [15] or the use of wireless links for the broadcast signals in [21]. These alternatives have not been considered in the BookSim implementation presented in Section III-A4, but some of them would require minimal code changes to be implemented.

B. Simulation Tools and Analytical Models

Section II-B already discussed BookSim, Garnet and OpenSMART. There exist many other open-source NoC simulators, such as [1], [11], [14], [46], [48]. While these tools provide very diverse and relevant characteristics, as far as we know they do not support bypass router models.

In some cases, such NoC simulators are integrated with other tools in order to evaluate shared-memory systems. Most simulation platforms at this level tend to be cycle-accurate to faithfully model the processing cores and the memory hierarchy. However, most of such proposals make use of simple NoC architectures [9], [56] or have employed router architectures without bypass features [8].

Even the largest state-of-the-art industrial FPGA platforms cannot simulate large multicores with tens of high performance cores, accelerators, a complex NoC and high bandwidth memory controllers. As an alternative, SynFull [4] proposes to abstract the core details and simulate or emulate only the NoC and memory hierarchy with a simple finite state machine that represents the traffic injected by the cores to the NoC. Such approach can be extended to incorporate accelerators although it is left for future work in the original paper.

For larger deployments, subsystem simulators are common tools that allow to obtain performance predictions and assist computer architects into designing specific parts of HPC systems. Mubarak et al. [45], for example, propose CODES, a fast and flexible simulation framework to model state of the art Torus and Dragonfly networks at a large-scale. Compared to this, our work focuses on a detailed NoC model that should be included in a state-of-the-art simulator like gem5, oriented to evaluate the nodes (servers) composing such big systems.

In this direction, prior work proposed simulation methodologies to evaluate the performance of large-scale parallel applications on distributed systems [20], [26], [61]. Most proposals use analytical models to estimate node performance (no cycle-accurate models are used) or system software interactions.

SST is a multi-scale simulator often used in combination with other simulators to model distributed applications. In BE-SST, authors combine SST with coarse-grained behavioral

emulation models abstracting from microarchitectural details in favor of simulation speed. Other implementations integrate SST with a highly accurate simulator but require too costly full system simulations to produce a wide set of experiments [28], [54]. Since SST is compatible with gem5, it could benefit from the proposals in this paper.

The MUlti-level SimulAtion methodology (MUSA) enables fast and accurate performance estimations in scenarios with several thousands of cores [24], [25]. MUSA takes into account inter-node communication, node-level architecture, and system software interactions. MUSA combines sampling techniques with different simulators based on analytical models and cycle-accurate traces. The NoC model in MUSA is a simple multi-bus without any notion of the router architecture. Integrating BookSim in MUSA is an interesting future work.

With the same objective, application specific analytical models [31], [43] use a small set of parameters to predict performance for a single application on large systems. Once those models are created and validated they are able to accurately predict performance with negligible compute and time cost. The main downside of these models is that they have little flexibility; any significant change in the application or hardware architecture requires the model to be updated, refined and validated again. Our methodology focuses on hardware microarchitectural exploration and iterative fast co-design; new features can be tested on all applications at the moment they are included in the simulator.

VI. Conclusions

This work presents BST, a simulation toolset to design and evaluate NoCs with single- and multi-hop bypass-routers. The toolset includes cycle-accurate simulation models of state-of-the-art single- and multi-hop bypass mechanisms for BookSim; an operational RTL model of SMART and SMART++ based on OpenSMART; and an integration API for BookSim to simplify its incorporation in other simulators.

We have shown the accuracy of the different SMART models, the utilization of the API to integrate BookSim in gem5, and the flexibility of the toolset to carry out evaluations of bypass routers in different topologies and configurations. The toolset is available at www.atc.unican.es/software.html.

Acknowledgment

This work was supported by the Spanish Ministry of Science, Innovation and Universities, contracts TIN2015-65316-P, TIN2016-76635-C2-2-R (AEI/FEDER, UE) and PID2019-105660RB-C22; the European Union's Horizon 2020 research and innovation program under the Mont-Blanc 2020 project (grant agreement 779877); and the HiPEAC Network of Excellence. I. Pérez is partially supported by an FPI grant, BES-2017-079971. M. Moreto has been partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Ramon y Cajal fellowship number RYC-2016-21104. Bluespec Inc. provided access to Bluespec tools.

References

[1] P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J. Gregorio. Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 99–106, May 2012.

[2] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 33–42, 2009.

[3] Y. Asgarieh and B. Lin. Smart-hop arbitration request propagation: Avoiding quadratic arbitration complexity and false negatives in smart nocs. *ACM Trans. Des. Autom. Electron. Syst.*, 24(6), Oct. 2019.

[4] M. Badr and N. D. E. Jerger. Synfull: Synthetic traffic models capturing cache coherent behaviour. In *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 109–120, 2014.

[5] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt. Analyzing cuda workloads using a detailed gpu simulator. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 163–174, 2009.

[6] A. Bakhoda, G. L. Yuan, W. W. L. Fung, H. Wong, and T. M. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 163–174, April 2009.

[7] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[9] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 52:1–52:12, 2011.

[10] C. Carrion, R. Beivide, J. A. Gregorio, and F. Vallejo. A flow control mechanism to avoid message deadlock in k-ary n-cube networks. In *Proceedings Fourth International Conference on High-Performance Computing*, pages 322–329, Dec 1997.

[11] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 162–163, July 2015.

[12] C. Chen and A. Joshi. Runtime management of laser power in silicon-photonic multibus NoC architecture. *IEEE Journal of Selected Topics in Quantum Electronics*, 19(2), March 2013.

[13] L. Chen, R. Wang, and T. M. Pinkston. Critical bubble scheme: An efficient implementation of globally aware network flow control. In *International Parallel Distributed Processing Symposium (IPDPS)*, pages 592–603, May 2011.

[14] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston. Simulation of noc power-gating: Requirements, optimizations, and the agate simulator. *Journal of Parallel and Distributed Computing*, 95:69 – 78, 2016. Special Issue on Energy Efficient Multi-Core and Many-Core Systems, Part I.

[15] X. Chen and N. K. Jha. Reducing wire and energy overheads of the SMART NoC using a setup request network. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(10):3013–3026, Oct 2016.

[16] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, San Francisco, USA, 2003.

[17] S. Davidson, S. Xie, C. Torng, K. Al-Hawai, A. Rovinski, T. Ajayi, L. Vega, C. Zhao, R. Zhao, S. Dai, A. Amarnath, B. Veluri, P. Gao, A. Rao, G. Liu, R. K. Gupta, Z. Zhang, R. Dreslinski, C. Batten, and M. B. Taylor. The Celerity open-source 511-core RISC-V tiered accelerator fabric: Fast architectures and design methodologies for fast chips. *IEEE Micro*, 38(2):30–41, Mar 2018.

[18] B. K. Daya, C. O. Chen, S. Subramanian, W. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L. Peh. SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering. In *International Symposium on Computer Architecture (ISCA)*, pages 25–36, June 2014.

[19] D. Deb, J. Jose, and M. Palesi. Performance enhancement of caches in TCMPs using near vicinity prefetcher. In *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*, pages 13–18, Jan 2019.

[20] W. E. Denzel, J. Li, P. Walker, and Y. Jin. A framework for end-to-end simulation of high-performance computing systems. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, pages 21:1–21:10, 2008.

[21] K. Duraisamy and P. P. Pande. Enabling high-performance SMART NoC architectures using on-chip wireless links. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3495–3508, Dec 2017.

[22] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, W. Zhao, X. Yin, C. Hou, C. Zhang, W. Ge, J. Zhang, Y. Wang, C. Zhou, and G. Yang. The Sunway TaihuLight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, Jun 2016.

[23] M. Galles. Spider: a high-speed network interconnect. *IEEE Micro*, 17(1):34–39, Jan 1997.

[24] C. Gómez, F. Martínez, A. Armejach, M. Moretó, F. Mantovani, and M. Casas. Design space exploration of next-generation HPC machines. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 54–65, 2019.

[25] T. Grass, C. Allande, A. Armejach, A. Rico, E. Ayguadé, J. Labarta, M. Valero, M. Casas, and M. Moretó. MUSA: a multi-level simulation approach for next-generation HPC machines. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 526–537, 2016.

[26] E. Grobelny, D. Bueno, I. Troxel, A. D. George, and J. S. Vetter. FASE: A framework for scalable performance prediction of HPC systems and applications. *Simulation*, 83(10):721–745, 2007.

[27] J. Hestness and S. W. Keckler. Netrace: Dependency-tracking traces for efficient network-on-chip experimentation. *The University of Texas at Austin, Dept. of Computer Science, Tech. Rep*, 2011.

[28] M. Hsieh, K. Pedretti, J. Meng, A. Coskun, M. Levenhagen, and A. Rodrigues. SST + gem5 = a scalable simulation infrastructure for high performance computing. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, SIMUTOOLS, pages 196–201, 2012.

[29] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh. NoC architectures for silicon interposer systems. In *IEEE/ACM International Symposium on Microarchitecture*, pages 458–470, Dec 2014.

[30] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 86–96, 2013.

[31] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings. Predictive performance and scalability modeling of a large-scale application. In *Supercomputing, ACM/IEEE 2001 Conference*, pages 39–39. IEEE, 2001.

[32] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks (1976)*, 3(4):267 – 286, 1979.

[33] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. In *International Symposium on Microarchitecture (MICRO)*, pages 172–182, 2007.

[34] T. Krishna. Garnet2.0: A detailed on-chip network model inside a full-system simulator. In *gem5 workshop, ARM Research Summit*, 2017.

[35] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh. Breaking the on-chip latency barrier using smart. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 378–389, Feb 2013.

[36] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P. Chiang. SWIFT: A swing-reduced interconnect for a token-based network-on-chip in 90nm CMOS. In *International Conference on Computer Design (ICCD)*, pages 439–446, 2010.

[37] A. Kumar, P. Kunduz, A. Singhx, L.-S. Pehy, and N. Jhay. A 4.6 Tbits/s 3.6 GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *25th International Conference on Computer Design (ICCD)*, pages 63–70, Oct 2007.

[38] A. Kumar, L.-S. Peh, and N. K. Jha. Token flow control. In *International Symposium on Microarchitecture (MICRO)*, pages 342–353, 2008.

[39] H. Kwon and T. Krishna. OpenSMART: Single-cycle multi-hop NoC generator in BSV and Chisel. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 195–204, 2017.

[40] S. Lie. Wafer scale deep learning. In *Hot Chips Symposium*, 2019.

[41] S. Ma, N. E. Jerger, and Z. Wang. Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1–12, 2012.

[42] S. Ma, Z. Wang, Z. Liu, and N. E. Jerger. Leaving one slot empty: Flit bubble flow control for torus cache-coherent NoCs. *IEEE Transactions on Computers*, 64(3):763–777, March 2015.

[43] V. Marjanović, J. Gracia, and C. W. Glass. Performance modeling of the hpcg benchmark. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pages 172–192. Springer, 2014.

[44] K. Matsumoto, N. Nakasato, and T. Hishinuma. Effectiveness of performance tuning techniques for general matrix multiplication on the pezy-sc2. In *International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART)*, pages 8:1–8:6, 2019.

[45] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns. Enabling Parallel Simulation of Large-Scale HPC Network Systems. *IEEE Trans. Parallel and Distr. Syst. (TPDS)*, 28(1):87–100, Jan. 2017.

[46] A. Norollah, D. Derafshi, H. Beitollahi, and A. Patooghy. Pat-noxim: A precise power thermal cycle-accurate noc simulator. In *International System-on-Chip Conference (SOCC)*, pages 163–168, 2018.

[47] M. Ortín-Obón, D. Suárez-Gracia, M. Villarroya-Gaudó, C. Izu, and V. Viñals-Yúfera. Analysis of network-on-chip topologies for cost-efficient chip multiprocessors. *Microprocessors and Microsystems*, 42:24–36, 2016.

[48] M. K. Papamichael and J. C. Hoe. Connect: Re-examining conventional wisdom for designing NoCs in the context of FPGAs. In *International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 37–46, 2012.

[49] M. Parasar and T. Krishna. Bindu: Deadlock-freedom with one bubble in the network. In *International Symposium on Networks-on-Chip (NOCS)*, pages 3:1–3:8, 2019.

[50] L. S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *International Symposium on High-Performance Computer Architecture (HPCA)*, pages 255–266, 2001.

[51] I. Perez, E. Vallejo, and R. Beivide. Efficient router bypass via hybrid flow control. In *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*, pages 1–6, Oct 2018.

[52] I. Pérez, E. Vallejo, and R. Beivide. SMART++: Reducing cost and improving efficiency of multi-hop bypass in NoC routers. In *International Symposium on Networks-on-Chip (NOCS)*, pages 5:1–5:8, 2019.

[53] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos. Shortpath: A network-on-chip router with fine-grained pipeline bypassing. *IEEE Trans. Comput. (TC)*, 65(10):3136–3147, Oct 2016.

[54] A. Ramaswamy, N. Kumar, A. Neelakantan, H. Lam, and G. Stitt. Scalable behavioral emulation of extreme-scale systems using structural simulation toolkit. In *International Conference on Parallel Processing (ICPP)*, pages 17:1–17:11, 2018.

[55] D. L. Rosenband. The ephemeral history register: flexible scheduling for rule-based designs. In *International Conference on Formal Methods and Models for Co-Design (MEMOCODE)*, pages 189–198, 2004.

[56] D. Sanchez and C. Kozyrakis. Zsim: Fast and accurate microarchitectural simulation of thousand-core systems. In *International Symposium on Computer Architecture (ISCA)*, pages 475–486, 2013.

[57] Y. Tamir and G. L. Frazier. Dynamically-allocated multi-queue buffers for VLSI communication switches. *IEEE Trans. Comput.*, 41(6):725–737, June 1992.

[58] B. Towles, J. P. Grossman, B. Greskamp, and D. E. Shaw. Unifying on-chip and inter-node switching within the Anton 2 network. In *International Symposium on Computer Architecuture (ISCA)*, pages 1–12, 2014.

[59] R. Wang, L. Chen, and T. M. Pinkston. Bubble coloring: Avoiding routing- and protocol-induced deadlocks with minimal virtual channel requirement. In *International Conference on Supercomputing (ICS)*, pages 193–202, 2013.

[60] A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[61] G. Zheng, G. Gupta, E. J. Bohm, I. Dooley, and L. V. Kalé. Simulating large scale parallel applications using statistical models for sequential execution blocks. In *International Conference on Parallel and Distributed Systems (ICPADS)*, pages 221–228, 2010.