PlugSMART: a pluggable open-source module to implement multihop bypass in Networks-on-Chip

Alireza Monemi
Barcelona Supercomputing Center
Barcelona, Spain
alireza.monemi@bsc.es

Iván Pérez
University of Cantabria
Santander, Spain
ivan.perezgallardo@unican.es

Neiel Leyva
Barcelona Supercomputing Center
Barcelona, Spain
neiel.leyva@bsc.es

Enrique Vallejo
University of Cantabria
Santander, Spain
enrique.vallejo@unican.es

Ramón Beivide
University of Cantabria
Santander, Spain
ramon.beivide@unican.es

Miquel Moretó
Barcelona Supercomputing Center
Universitat Politècnica de Catalunya
Barcelona, Spain
miquel.moreto@bsc.es

ABSTRACT

The integration of many processing elements per die makes it more difficult to provide low latency in the Network-on-Chip (NoC). Multihop bypass proposals, such as SMART, attack this problem by allowing flits to skip multiple routers in the path in a single cycle, drastically reducing latency while preserving a regular tiled layout. However, multihop bypass routers are more complex and relatively different from traditional NoC routers, since they rely on global broadcast signals and global allocation mechanisms. Additionally, the maximum number of nodes that can be bypassed within a single cycle is limited by the Critical Path Delay (CPD) of the NoC. Hence, a practical multihop bypass mechanism must also minimize this delay.

To simplify the design of multihop bypass mechanisms, this work introduces PlugSMART, an open-source pluggable Verilog module that extends a traditional router to support multihop bypass. PlugSMART follows a black box approach, requiring minimal modifications from the original router. As an application of PlugSMART, we introduce ProSMART, a multihop bypass extension of the efficient NoC router ProNoC. ProSMART is evaluated using simulations, FPGA, and ASIC synthesis. Results show that it is more performant and requires significantly fewer resources than previous open-source designs. The comparison with OpenSMART++, the most recent state-of-the-art SMART-based NoC, shows up to a 50% reduction in both area and CPD. Overall, PlugSMART constitutes a simple alternative for fast and efficient upgrading of existing NoC routers, allowing to implement multihop bypass and significantly improve performance while preserving the original characteristics of the router design.

CCS CONCEPTS

• **Networks → Network on chip**; **Network resources allocation**; *Network performance evaluation.*

NOCS '21, October 14–15, 2021, Virtual Event, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9083-5/21/10…$15.00
https://doi.org/10.1145/3479876.3481601

KEYWORDS

Network-on-Chip; SMART NoC; Low latency; Multihop bypass

1 INTRODUCTION

Thanks to the technology scaling, today's supercomputers, high-end servers, and GPUs employ chips that integrate tens to hundreds of processing elements (PEs) in a single die. Network-on-Chip (NoC), a scalable, high bandwidth inter-core communication infrastructure, is proposed to meet the performance demand in such many-core systems on chips. Multiple NoC router designs have been proposed with different characteristics, many of them open-source [12, 13].

Fundamentally, increasing the number of PEs in a NoC-based system may lead to higher communication latency. The latency is proportional to the number of intermediate nodes located between two PEs. Thus, latency reduction has become a key research element in Networks-on-Chip [3–5, 10, 15, 18].

Multihop bypassing is an essential mechanism for designing a near-optimal interconnection network. Multihop bypass, first introduced in single-cycle multihop asynchronous repeated traversal (SMART) [3], allows bypassing a user-defined number of intermediate hops in a single cycle, drastically reducing average latency. This proposal has been implemented and improved in several other related works [1, 7, 14, 15]. However, implementing multihop bypass is neither easy nor immediate, since it requires a significant redesign of the NoC router and implementing global signals that communicate far-away routers. Despite its benefits, most available NoC routers do not implement this feature.

Additionally, multihop bypass is only effective if the data/control packets reach their destinations satisfying a specific time constraint. This constraint is defined according to the target operating frequency of the NoC, and restricts the maximum number of hops that can be bypassed within a single cycle (HPC_{Max}). In other words, the HPC_{Max} parameter should be defined in such a way that the NoC CPD meets the predefined timing constraint. This is further discussed in Section 2.

This paper introduces PlugSMART, an open-source pluggable module that extends an existing NoC router incorporating multihop bypass capabilities. PlugSMART follows a black-box approach, requiring minimal adaptations from the original NoC design and allowing for fast design cycles. PlugSMART is demonstrated by designing and evaluating a multihop bypass NoC based on ProNoC [12], an open-source NoC RTL code.

We compare our proposed design against OpenSMART++ [15], a state-of-the-art SMART-based NoC, including a model of global-wire delays in an ASIC implementation. The obtained results show a significant improvement in the area and timing overheads of the proposed design. The proposed PlugSMART-based solution can be adopted by any wormhole (WH) NoC with minimal modifications.

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

- PlugSMART, an open-source pluggable module that simplifies the design of multihop bypass routers, allowing existent designs to be easily upgraded to support this feature.
- ProSMART, a multihop bypass NoC implemented by extending ProNoC [12] with PlugSMART, with two variants employing 2 and 3 stages per router.
- An exhaustive evaluation of PlugSMART-based ProSMART using simulation, FPGA, and ASIC implementations. Evaluations consider both the impact of logic and global wiring delays. Results show that ProSMART outperforms previous designs in performance, maximum frequency, power, and required resources. Results also provides insight on the impact of the logic and global wiring delays, with a 3-stage design outperforming simpler 2-stage design.

The rest of this paper is organized as follows: Section 2 provides background information regarding existing multihop bypass NoCs, the reference ProNoC router, and the logic and global-wiring delays in multihop designs. Section 3 describes the microarchitecture details of the proposed PlugSMART module and the ProSMART design based on it. The performance results of ProSMART are analyzed in Section 4. Section 5 concludes the paper.

2 BACKGROUND

This section presents a brief outline of multihop bypass in NoC design, and an analysis of the involved delays and an introduction of ProNOC, the base router employed to demonstrate PlugSMART.

2.1 SMART

SMART [3] is an efficient low-latency NoC architecture originally designed for Chip Multi-Processors (CMPs). It uses a bypass mechanism to allocate multihop paths so packets can traverse multiple routers in a single cycle. Thus, SMART reduces the effective number of hops in tiled topologies such as meshes or tori, as an alternative to other solutions that increase the degree of the network, which has a negative effect on power/area and timing [6].

Figure 1 depicts the high-level diagram of a multihop in SMART. SMART has a pipeline of 3 stages: the first stage computes the next multihop in the route of packets and performs Switch Allocation Local (SA-L), which allocates the crossbar for local flits. SMART relies on on/off flow control signaling, together with a Virtual Channel (VC) Selection (VS) mechanism that skips VC Allocation (VA), deferring this assignment to the upstream router upon flit reception. In the second stage, the routers broadcast Switch Set up Requests (SSRs) for local flits that won SA-L in the first stage. These SSRs place allocation requests in Switch Allocation Global (SA-G) in the routers involved in the next multihop to setup their bypass paths and crossbars. When

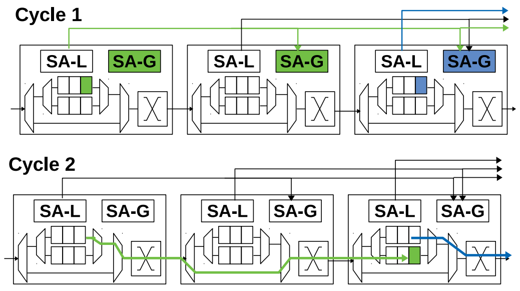
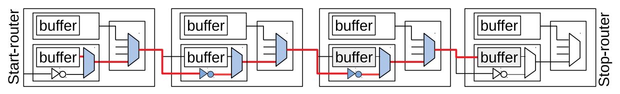
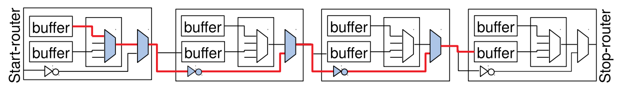


Figure 1: SMART multihop high-level diagram and pipeline.



(a) SMART_2D (also *buffer bypass*).



(b) SMART_1D (also *router bypass*).

Figure 2: SMART_2D VS. SMART_1D CPD.

SSRs conflict in SA-G with other SSRs or requests from local flits, priority is given to requests that come from the closer router to the one performing SA-G [3]. In the third stage, flits do Switch Traversal (ST) and Link Traversal (LT) in all the routers whose SSRs won SA-G in the previous cycle until encountering the first bypass path disabled.

Krishna *et al.* [3] propose two types of SMART NoCs: SMART_1D and SMART_2D. In the 2D version, flits are permitted to bypass any intermediate nodes, whereas the 1D prohibits bypass in turns. Figure 2 highlights the CPD of SMART_2D versus SMART_1D. Besides the complexity that SMART_2D introduces, it may also suffer from a longer CPD due to including the bypassed routers' crossbar module in the NoC's CPD.

2.2 Improvements to SMART

Multiple improvements to SMART have been proposed. This section presents three alternatives focused on improving buffer management, broadcast cost, and design availability.

2.2.1 SMART++. SMART++ [15] is an improved version of SMART that does not require VCs to take advantage of the network bandwidth. SMART implements flit-by-flit arbitration in combination with atomic virtual channel allocation (VA) to resolve conflicts between flits that request bypass paths. Thus, SMART needs multiple VCs to increment the available throughput with its associated implementation costs that affect the area, power, and CPD. SMART++ combines SMART with three mechanisms:

- Multi-Packet Buffers (MPB), or non-atomic VC reallocation [9], allows storing multiple packets in the same buffer or VC.MPB alleviates the need for multiple VCs of SMART.
- Packet-by-Packet Arbitration (PPA) is used to allocate the switch and bypass paths of the router in a packet basis, following Virtual Cut-Through.

- Finally, Non-Empty Buffer Bypass (NEBB) [14] enables packets to use the bypass even when the buffers to be bypassed are not empty. NEBB mitigates the Head of Line Blocking (HoLB) issues caused by having few VCs, as it enables packets to bypass a buffer that contains other blocked packets waiting for the required resources.

2.2.2 SHARP. SHARP [1] reduces the number of SSR wires by implementing propagation-based SSRs. SMART uses dedicated wires to broadcast the SSRs to the neighbor routers in the range of HPC_{Max}. This translates into a substantial number of wires required, which can be prohibitive if the design allows packets to take the bypass when they change the traveling dimension in SMART_2D [3].

SHARP reduces the number of SSR wires and consequently the complexity of SA-G by adding a module to the router that arbitrates among all the received SSRs and only propagates the winners.

2.2.3 OpenSMART(++). OpenSMART [7] is an open-source RTL implementation of SMART. It is written in Bluespec System Verilog (BSV), a high-level hardware description language. OpenSMART has four main differences with respect to the original SMART design. Firstly, OpenSMART implements SMART_1D, i.e., it does not allow packets to take the bypass when changing the traveling dimension. Thus, the bypass paths directly reach the input port with their opposite output port in the same direction. Secondly, ST is moved to the second stage of the pipeline, and the third stage only consists of the multihop LT. This is possible as OpenSMART gives priority to SSRs with the closest distance to the router, so it is guaranteed that local flits that win SA-L in the first stage can use the crossbar directly. Thirdly, OpenSMART uses credits instead of on/off signaling, which also implies the use of VA instead of VC Selection (VS). Lastly, OpenSMART implements an SSR propagation mechanism in its “SMART unit”, similar to the one proposed by SHARP.

BST [16] includes an implementation of SMART++ based on OpenSMART. It follows a similar architecture to OpenSMART, but it performs ST in the third pipeline stage and replaces some structures like registers and FIFOs with built-in ones to resolve some errors.

2.3 SMART Delay Analysis

In general, two sources of delay in the NoC CPD are defined:

(1) The CPD's Logic-Gate-Delay (LG-CPD): The delay caused by logic gates located in the NoC's CPD, including the delay of local wires connecting these logic gates.
(2) The CPD's Global-Wire-Delay (GW-CPD): The delay caused by the global-wires that connect neighboring routers.

Hence, we have:

$$CPD = \sum_{i=1}^{HPC_{max}} (LG\text{-}CPD_i + GW\text{-}CPD_i) \qquad (1)$$

Related work in the literature often considers only one of GW-CPD or LG-CPD in the calculation of the HPC_{Max} value. In [3], achievable HPC_{Max} is analyzed according to the global-wire length only, but the impact of LG-CPD is ignored. However, the SMART units are complex, and resolving the conflict with local requests plus the propagation delay of bypass multiplexers can significantly increase the NoC CPD.

In OpenSMART [7], the NoC CPD is obtained by synthesizing a SMART NoC without considering/reserving the area for the processing tiles. As a result, the synthesizer maps all adjacent routers side by side, neglecting the impact of GW-CPD. Moreover, as technology shrinks, the GW-CPD can become the dominant source of delay in a system. Figure 3 shows well-known estimations of how GW-CPD increases when the technology shrinks. For instance, in 22 nm technology, each mm global-wire (with repeaters) causes a 120 ps propagation delay. OpenSMART [7] reports a maximum of 40 ps delay for bypassing one router in a 15 nm technology, which can be very optimistic. Despite ignoring GW-CPD in the CPD calculation, OpenSMART reports considerably high CPD, i.e., 490 ps on 15 nm technology. We note that a large portion of this CPD (380 ps) is reported for start and end routers. Our experiments show that OpenSMART++ has even higher CPD due to more complex allocator units.

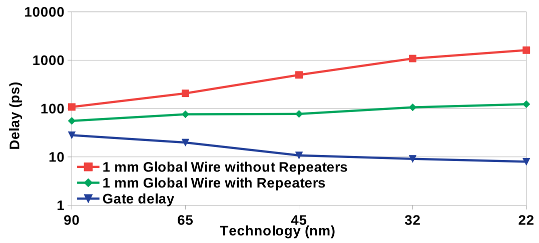


Figure 3: The interconnect delay scaling ITRS [2].

2.4 ProNoC

ProNoC [12] is an open-source NoC RTL code developed using Verilog HDL. The ProNoC's RTL code is fully parametrizable in terms of the number of VCs, virtual networks, network topology, and routing algorithms. ProNoC adopts MPB [9] and can be configured with hard-built Quality of Service (QoS) support [11]. ProNoC adopts WH which allows flits belonging to the same packet to be stored in the buffer space of several routers along the path.

A key feature of ProNoC is the optimization techniques that are adopted to minimize the area and timing overhead of the router. The VC and switch allocator modules are combined (VSA) in such a way that VA is done only upon successful switch allocation (SA) [8]. This significantly reduces the area and timing overhead of the allocator unit. Moreover, in ProNoC all VCs located in the same input ports share the same memory module. For FPGA platforms, this feature results in more efficient utilization of hard-built memory units such as Memory Slices (SLICEMs) or Block-RAMs (BRAMs). Besides that, input ports also consist of Output VC (OVC) status predictor modules that remove the delay of OVC-status signals forwarding (from output ports to input ports) from router's CPD. In ProNoC router, all packets involving in a turn face a zero-load two-cycle pipeline latency. However, packets heading to the same direction (straight direction) are allowed to bypass the first pipeline stage. Hence, they face only single-cycle latency. Multihop bypassing is not supported in ProNoC.

3 PLUGSMART ARCHITECTURE AND PROSMART

This section introduces PlugSMART, a pluggable module that simplifies the upgrading of existent routers to support multihop bypass. PlugSMART is then demonstrated by integrating it with the ProNoC router, in a design denoted ProSMART.

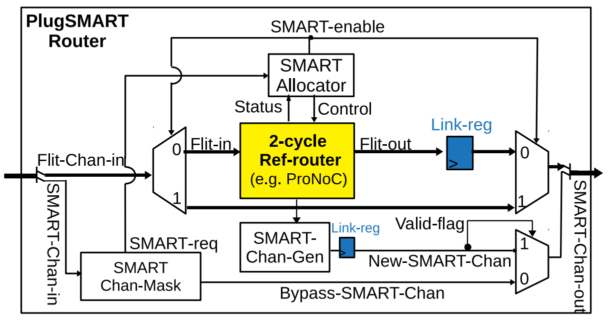


Figure 4: PlugSMART schematic based on a 2-cycle ref. router.

3.1 PlugSMART

Figure 4 shows a top-level functional block diagram of PlugSMART, which is built by adding bypass modules i.e., SMART-channel, SMART-channel-Gen (SM-G), SMART-channel-Mask (SM-M), and SMART-Allocator (SM-A) to the reference router, which is considered a black-box IP. These components are detailed in the following subsections.

3.1.1 SMART-Channel. The routers' connection links are divided into two channels. PlugSMART preserves the original router flit-channel and adds a new SMART-channel. Data flits are sent using the original flit-channel, and the SMART-channel carries some information regarding the incoming flit one clock cycle in advance.

SMART-channel includes the flit's VC number (SMART-VC), flit's final destination address, and a Bypass-Request-Flags (BRFs) field that codifies the Maximum-Hop-length value (MHL) at the start router, with a fixed number that has the lowermost 'MHL' bits at '1'. As an example, assuming HPC_{Max} is defined as 4, then MHL is 3, one unit smaller than HPC_{Max}, and BRFs becomes 3'b111.

If there is a valid new SMART-channel request in the static straight direction (SS-OPORT) from the current router, the incoming SMART-channel request from the upstream router is dropped. This is because the newly generated SMART-channel request has a higher priority towards the incoming ones as proposed in [1].

For a router with 3 pipeline stages (such as the ProSMART3 design introduced in Section 3.2) both flit-channel and SMART-channel are registered at the start router output ports. Link registers, shaded with blue color in Figure 4, remove the initial processing delay from the NoC LG-CPD in such a case. A two-stage version (such as ProSMART2) does not implement these pipe-registers.

3.1.2 SMART-Channel-Gen (SM-G). PlugSMART implements a specific module called SMART-channel Generator (SM-G) that forwards SMART-channel fields from input ports to output ports while the router is performing SA. The SMART-channel fields (SMART-VC and destination address) are originally stored inside the start router's input ports. Forwarding these fields from input ports to output ports in the LT stage would result in some initial delay to LG-CPD. To avoid this problem, our design forwards them in an earlier phase (SA stage). This way, once a packet wins the local SA requests, it is needed to forward these fields to the granted output port in the same clock cycle. As a result, the new SMART-channel is fed directly from link-registers to the downstream routers while the actual flit is traversing the crossbar and heading to the flit-link register. However, waiting

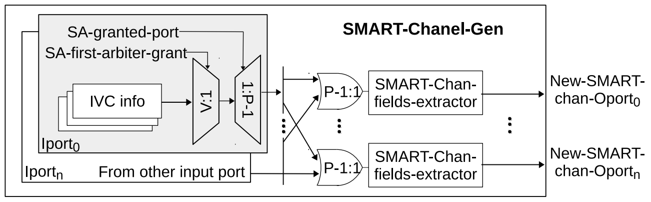


Figure 5: SMART-Channel-Generator (SM-G)

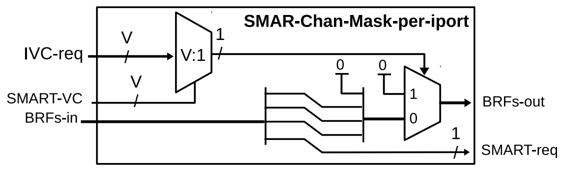


Figure 6: SMART-Channel-Mask (SM-M)

for SA results would add additional delay to the router's CPD. To minimize the impact of this delay on the routers' CPD, SM-G forwards SMART-channel fields in parallel while the router is performing SA.

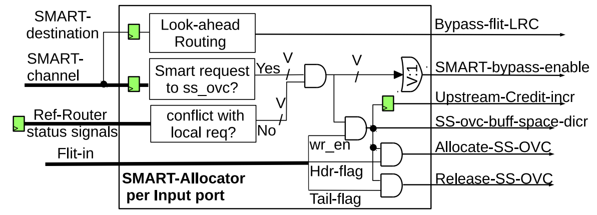
Figure 5 illustrates the SM-G schematic. PlugSMART considers a router with two-level SA. The SM-G module employs information from the first level of arbitration to forward the winner Input VC (IVC) fields to all output ports in parallel, while the router performs the second SA arbitration level. Once the second-level arbitration results are ready the SM-G uses them to generate the data for the new SMART-channel request.

3.1.3 SMART-Channel-Mask (SM-M). SM-M, which is shown in Figure 6, controls the activation of the bypass in the downstream router by manipulating the BRFs field. The bypass will be disabled in the next router if a zero BRFs value is detected in the SMART-channel. SM-M (zero-fill) shifts the BRFs field one bit to the right when bypassing the SMART-channel to the downstream router. This prevents the bypass hops from exceeding the MHL value. Moreover, PlugSMART requires that the IVC that receives a bypass request is empty, because it does not implement non-empty buffer bypass. As a result, the BRFs is set to zero once the IVC is not empty.

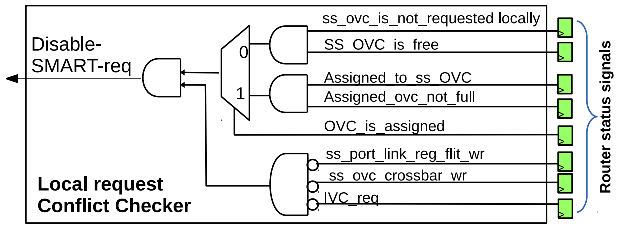
3.1.4 SMART-Allocator (SM-A). SM-A is responsible for enabling bypass signals and creating allocation control signals for flits that are successfully bypassed by the router. Figure 7a illustrates the functional block diagram of SM-A. Each routers' input port has one of this SM-A unit that handles bypass requests to its respective SS-OPORT. This section considers the specific design of ProNoC.

For routers (e.g., ProNoC) that work with look-ahead routing (LRC), the destination port is pre-calculated one router in advance, and is sent by the header flit. Hence, the LRC field must be updated at each successful bypass for the next router. For this reason, the packet destination core address is sent via SMART-channel one-cycle in advance to all routers in the downstream direction. Each SM-A has an LRC module (adopted from the reference router) that pre-calculates the destination port for the header flit bypassing the router.

The least significant bit of BRFs is called SMART-request. If it is asserted, it means that the number of bypassed routers has not reached MHL yet. To enable the bypass flag, the SM-A first needs to check if the incoming flit can be sent to the straight direction. To



(a) SMART-Allocator



(b) Conflict Checker

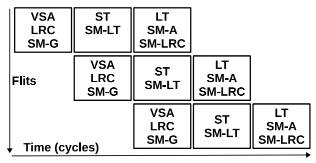
Figure 7: SMART-Allocator block diagram.

do that, the SM-A adopts the reference router's route compute (RC) unit fed by the SMART-channel destination address. If the straight output port is not included in RC's output, the bypass is disabled. By adopting the LRC and RC units of the reference router, PlugSMART can work with any type of topology and routing algorithm that are supported by the reference router.

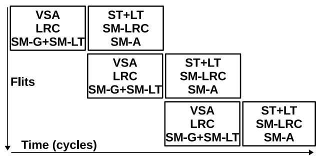
Packets are allowed to allocate any requested OVC using the local VA of the routers. The allocated OVC is then sent via SMART-channel (called SMART-VC) one cycle in advance to all downstream routers. One optimization technique that is applied in SM-A design is to restrict the SM-A to perform VA only on the SMART-VC. As a result, all the routers in a multihop allocate the same OVC index for the flit. If the SMART VC is not available, the bypassing is disabled regardless of the availability of other IVCs. This is a tradeoff that simplifies the SM-A unit and, more importantly, it removes the VC arbitration stage at bypass links, reducing LG-CPD. In subsequent multihops the packet can allocate any OVC that is available in its router.

The bypass can be enabled if there is no conflict with routers' local requests. As SM-A is the main component defining the LG-CPD, care should be taken in designing the conflict checker not to increase the CPD. Figure 7b shows the router status registers which are checked for resolving local conflicts. The bypass-flit does not pass via the intermediate routers' crossbar. This is because we have targeted SMART_1D (see Figure 2a). Hence, a router can perform the local SA without considering the SMART request. In case the SA locally grants an output port in the current clock cycle, the bypass will be disabled in the next clock cycle for that output port.

The main challenge here occurs when a specific OVC is requested by both local and bypass header-flits at the same time, as only one of them should get the OVC allocation grant. The SMART solution is to give higher priority to the local VA. Otherwise, the bypass flits might always be granted by SM-A, causing starvation for the locally stored flits. To set higher priority to local VA requests, the SM-A would need to wait for the results of the local VA first. Then, in case the desired OVC is not granted locally, it would allow the incoming header flit to allocate that OVC and successfully bypass the router. However, this method would introduces a significant timing overhead on both router's CPD and LG-CPD.



(a) 3-stage pipeline



(b) 2-stage pipeline

Figure 8: ProSMART pipeline stages.

To overcome this problem, as an optimization technique, we have set a limitation on who can request a specific OVC in such a way that in any given clock cycle, a specific free OVC is only available to one of SM-A or local VA:

- The free static straight OVC$_n$ (SS-OVC)$_n$ is available for SM-A in the current clock cycle if a bypass request has been received in the last clock cycle for that OVC and the desired SS-OVC$_n$ is not requested locally by any IVC$_n$ in the last clock cycle, where n is the IVC/OVC index number in its input/output port.
- A free SS-OVC is available for the local VA in the current clock cycle if it is not available for SM-A.

According to this policy, the available OVCs are masked at the beginning of the allocation stage. Moreover, the masks are fed from registers. Consequently, both SM-A and VA can perform VC allocation independently.

3.2 ProSMART Implementation and Pipeline Stages

ProSMART is a multihop bypass NoC built using PlugSMART with a reference router ProNoC. Currently PlugSMART is validated with 2- and 3-stage reference routers so far. PlugSMARTrequires minor modifications in the reference router to communicate the control signals. This includes adding one status and one control interface. The status interface is defined as output and consists of SA's first and second arbitration level grants, IVC's destination address, IVC-not-empty-flag, assigned OVC number to body and tail flits, granted OVC number for header flits, and free-OVC flags. The control interface is defined as input in the reference router. The control interface defines a 4-bit control-value for each OVC, where each asserted bit indicates if the corresponding OVC is allocated, released, granted, or masked for local VSA, respectively. For each OVC, the final results of local VSA are ORed with the first 3-bit control signals. Moreover, the corresponding free OVC is masked at the beginning of VSA if the 4th bit is asserted.

ProSMART routers can be configured with two or three pipeline stages (ProSMART2 and ProSMART3) as illustrated in Figure 8. In the first pipeline stage, the ProSMART3's router performs VSA, LRC, and SM-G in parallel. The results of SM-G are stored in the link registers at the granted SS-OPORT. In the second stage, the flits that won the SA grants in the previous stage traverse the router's crossbar (ST). At the same clock cycle, their SM-G results traverse the connection links of all downstream routers until finding the first SMART-bypass disabled router (SM-LT). In the third pipeline stage, flits traverse the routers' connection links until reaching the first flit-bypass disable router (LT). The flit-bypass flag is generated in the same cycle in parallel by each downstream router's SM-A module.

The 2-stage router (ProSMART2) is simply obtained by removing the link registers from both SMART and Flit channels. As a result, at the first pipeline stage, the SM-G results also traverse the SMART-channels until they reach the first SMART-bypass disabled router. In the second stage, flits perform ST followed by LT. The evaluation section shows how a 2-stage design seems to be more efficient when considering functional simulations, but not when GW-CPD is taken into account.

4 EXPERIMENTAL RESULTS

This section evaluates the efficiency of PlugSMART by comparing ProSMART2 and ProSMART3, introduced in Section 3.2, with OpenSMART++ [16] using three pipeline stages (OpenSMART3). Evaluations consider both performance (load-latency curves) and FPGA/ASIC implementation complexity.

4.1 Simulation Results

We employ the Verilator [17] simulator to obtain the ProSMART3 and ProSMART2 performance results in an 8×8 Mesh topology with an HPC_{Max} value of 7. The NoC simulation configuration parameters are taken according to OpenSMART++ reported performance results in [15] i.e., using Dimension-Order-Routing (DoR), VC-depth of 4, flit width of 32-bit, with both transpose and uniform traffic patterns, with simulations running for one million cycles.

Figure 9 shows the obtained performance results in terms of average communication latency (in cycles) versus offered load in percentage. Figure 9a shows the obtained results for a NoC with only one buffer slot per router port (no-VC) with different buffer depths. Buffer depths are set to 5, 10, and 20, and performance is obtained using uniform random traffic with 5-flit packets. In this graph, ProSMART3 and OpenSMART3 show very similar performance while ProSMART2 has a 15% to 22% lower average packet latency compared to both 3-stage NoCs.

Figure 9b shows the performance comparison for no-VC configuration under a uniform random traffic that consists of bimodal packets, with 80% single-flit and 20% 5-flits packets. ProSMART3 with 5 and 10 buffer-slots outperforms OpenSMART3 with 18% and 10% higher saturation throughput, respectively. The 3-stage NoCs with 20 buffer slots show very similar performance. ProSMART2 outperforms ProSMART3 and OpenSMART3 with 30% lower average packet latency for all configurations. ProSMART2 with 5-buffer slots has 28% and 10% higher saturation throughput compared with ProSMART3 and OpenSMART3, respectively.

The performance comparison under uniform random, single-flit packet traffic with is depicted in Figure 9c. Here, routers have only one buffer slot where the buffer size per input port is smaller, i.e., 2, 4, and 8. The OpenSMART3 shows considerably higher saturation throughput compared with ProSMART3 in dealing with single-flit sized packets that are 42%, 26% and 10% for 2,4 and 8 buffer slot, respectively. This is because OpenSMART3 leverages two extra pipeline registers inside each routers' input port [15] compared to ProNoC. Once a flit is sent to the next pipe-reg, another flit can be injected into the router. Since each input port's internal buffer space is small in this experiment, the influence of these pipeline registers is noticeable.

ProSMART3 shows a better performance compared to OpenSMART3 in dealing with matrix transpose traffic consists of bimodal packets (see Figure 9d) leading to 36%, 18%, and 10% higher saturation throughput for 5, 10, and 20 buffer-space, respectively. ProSMART2 shows similar saturation throughput as ProSMART3 while reducing the average packet latency between 30% to 40%.

Figures 9e and 9f show the performance comparison for NoCs with different numbers of VCs per port under uniform random traffic injecting 5-flit and bimodal sized packets, respectively. Each VC is configured with a 5-flit depth. The obtained results for each experiment are similar to those shown in Figures 9a and 9b under the same traffic pattern and equal total buffer space per input port. Here, adding VCs reduces Head of Line Blocking, resulting in slightly (8% to 10%) increased saturation throughput.

The simulation results show both ProSMART2 and ProSMART3 outperform OpenSMART3 in most cases, particularly with bimodal packets. Notably, ProSMART2 presents the best results, but this is without considering clock cycle nor GW-CPD.

4.2 FPGA Implementation Results

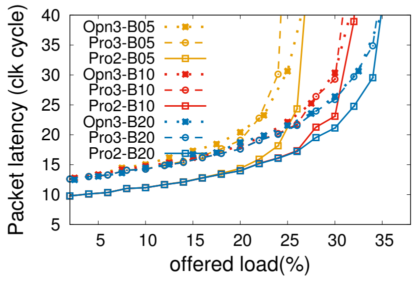
Figure 10 illustrates the FPGA implementation requirements for 4×4 Mesh NoCs with different number of VCs on a XC7K325T-2FFG900 Xilinx FPGA. Other NoCs parameters are set identically in all models as follows: HPC_{Max} of 3, VC-depth of 4, flit width of 32 and, DOR.

The FPGA implementation results show a significant improvement over OpenSMART3 in both area and timing overhead. A 4×4 ProSMART3 can be clocked with 2.5 to 3 times higher operating frequency compared to OpenSMART3. ProSMART2 has 8 to 10% lower operation frequency than ProSMART3. ProSMART2 with 8 VCs can still be clocked with 30% higher operating frequency than OpenSMART3 having no VC.

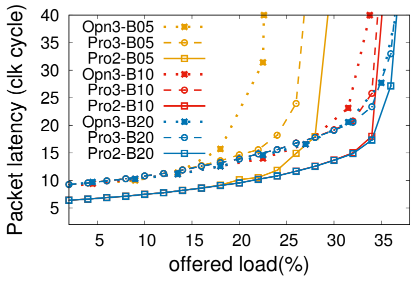
ProSMART2 and ProSMART3 report on average 60% lower slice utilization than OpenSMART3. Moreover, the improvement in memory utilization is much more noticeable. ProNoC is originally designed as an FPGA optimized RTL code that allows all VCs located in the same input port to share the same memory unit. As a result, for a NoC with up to 4 VCs, the same number of SLICEMs (LUTs) is used to implement memory units. For 8 VC NoC, the input buffers are implemented using inbuilt BRAMs. OpenSMART3 with no VC has 60% higher SLICEM utilization than ProNoC(2,3). Moreover, the SLICEMs utilization is doubled when doubling the VC number, which results in 500% higher SLICEMs utilization for a 4-VC NoC.

4.3 ASIC Implementation Results

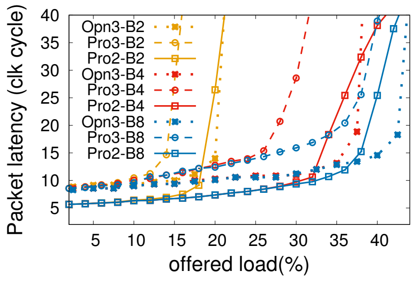
Synopsys Design Compiler with (GLOBALFOUNDRIES 22FDX) 22 nm technology library is used for ASIC realization of both the proposed design and the OpenSMART++ design. In order to investigate the impact of global wiring delay, we model the adjacent router's connection wires with a predefined number of serial repeaters and force the compiler not to optimize the wire model (keep all repeaters). Two experiments are conducted, one with 9 (denoted W1) and another with 17 (W2) repeaters. Each repeater introduces a delay of 13.5 ps according to the used technology library, resulting in an accumulation of 120 ps and 230 ps delays for global connection wiring



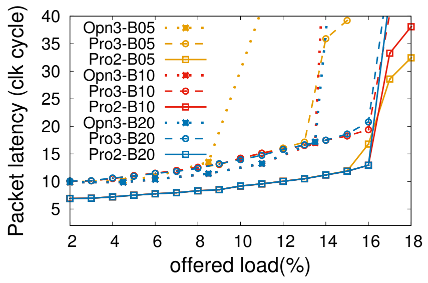
(a) Uniform Random, 5-flit packets, no VC



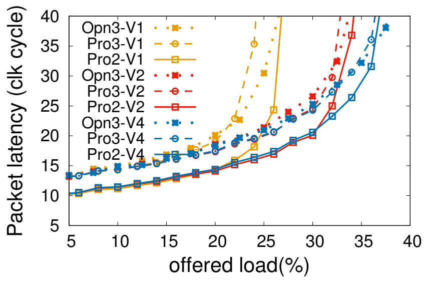
(b) Uniform Random, bimodal packets, no VC



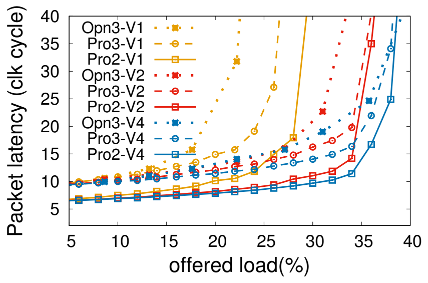
(c) Uniform Random, single flit packets, no VC



(d) Transpose, Bimodal packets, no VC

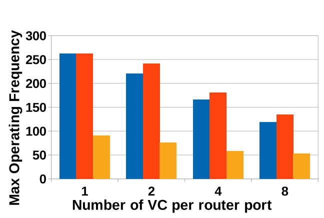


(e) Uniform Random, 5-flit packet, B=5

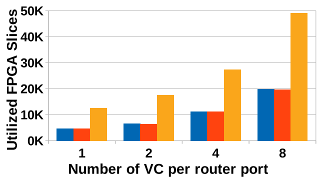


(f) Uniform Random, Bimodal packet, B=5

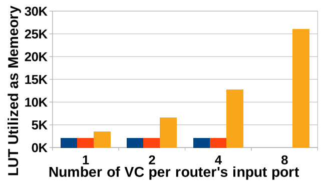
Figure 9: Performance comparison of ProSMART versus OpenSMART++ on 8×8 mesh topology and HPC_{max}=7. Pro3, Pro2, Opn3, B, V stand for ProSMART3, ProSMART2, OpenSMART3, Buffer depth per VC in flit, and, VC number per port, respectively.



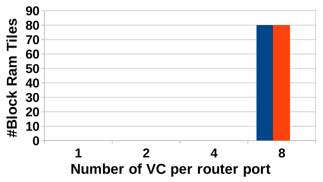
(a) Max. Operating Frequency (MHz)



(b) FPGA Slices



(c) LUT as memory



(d) Block Ram usage

Figure 10: FPGA Implementation results of a 4×4 NoC on XC7K325T-2FFG900 Xilinx FPGA.

between two adjacent routers, using W1 and W2, respectively. This represents a 1 mm and 2 mm wire length according to the prediction graph in Figure 3. Without adding this long-wire model, the synthesizer places all adjacent routers as close as possible, resulting in an unrealistic timing report.

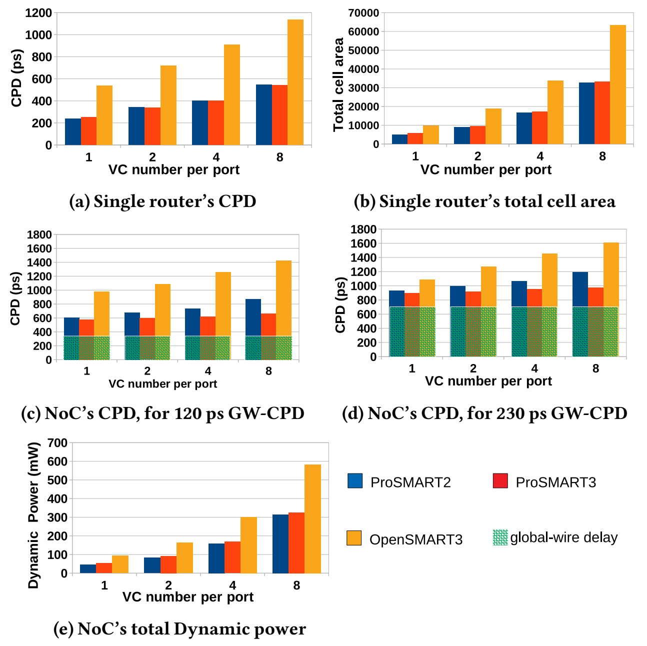
We employ a modular bottom-up approach. In the first level of compilation, a 5-port router is synthesized. In the second level, the synthesized routers are connected using global-wire models to generate the network. We set the same NoC configuration parameters as in the FPGA implementation section.

Figures 11a and 11b plot the timing and area reports obtained from synthesizing a single router at the first compilation level. The CPD and area utilization of OpenSMART3 is at least 2 and 1.7 times higher compared to ProSMART(2&3), respectively. In both ProSMART routers, the CPD was reported for VSA unit. As a result, the reported CPDs are very close to each other.

The synthesis results using W1 and W2 global-wire models are illustrated in Figures 11c and 11d, respectively. These graphs show the impact of both the router's pipeline stage and the global-wire delay model in the overall NoC's CPD. The reported CPD is the accumulation of HPC_{Max} times the single wire-delay model (360 ps for W1 and 690 ps for W2) plus the LG-CPD of (HPC_{Max}+1) number of routers. The accumulation of global-wire delay is shown with the Hatch pattern in these graphs.

To reach a target operating frequency of 1 GHz using the W2 wire model, the total LG-CPD should be bounded by 300 ps. All configurations of ProSMART3 with VC numbers of 1, 2, 4 and 8 meet the 1 GHz timing constrain. For ProSMART2, only the configurations with 1 and 2 VCs meet this timing constraint. On the other hand, none of the OpenSMART3 configurations can meet this timing constraint.

By halving the global-wire delay (W1 model), all ProSMART configurations meet the 1GHz timing constraint, whereas only OpenSMART3 with one VC can meet it.



(a) Single router's CPD (b) Single router's total cell area (c) NoC's CPD, for 120 ps GW-CPD (d) NoC's CPD, for 230 ps GW-CPD (e) NoC's total Dynamic power

Figure 11: ASIC results for HPC_{Max}=3, on 22 nm technology.

ProSMART3 without VCs shows 200 ps total LG-CPD. We can see that increasing the VC number from 1 to 8 in ProSMART3 results in only 40% increases in LG-CPD. By contrast, for ProSMART2 and OpenSMART3 this increase is more than 200%. In ProSMART3, the CPD is set from the flit-channel bypass links, which consist of both GW-CPD and LG-CPD (note that SM-A processing contributes to LG-CPD). Thanks to the optimization done in the SM-A design, the influence of increasing the VC number is reduced.

In ProSMART2, the CPD consists of the SMART-channels. It includes the start router CPD for generating the new SMART-channel (SM-G delay) plus GW-CPD and the delay caused by masks and multiplexers in bypass links (SM-M delays). Increasing the VC number enlarges the start router CPD and, as a result, raises the NoC's CPD.

OpenSMART3 with one VC has a total of 400 ps LG-CPD, which doubles ProSMART3 with the same configuration. Moreover, increasing the VC number significantly increases LG-CPD (up to 200% for 8-VC NoC). Finally, ProSMART reduces the dynamic power consumption by 45%, compared to OpenSMART.

5 CONCLUSION

This work introduces PlugSMART, an open-source module that seeks to simplify the implementation of multihop bypass in NoC routers. Based on this module, an optimized RTL implementation based on ProNoC is presented. The proposed design is optimized in both area and timing overhead. ProNoC RTL code is adopted as the reference 2-stage router. The SMART-channels and bypass modules are exclusively designed for supporting single-cycle multihop bypassing with low area and timing overhead. The proposed design ProSMART shows a significant reduction in timing and area overhead of the NoC in comparison with state-of-the-art SMART-Based NoCs OpenSMART++. Specifically, ProSMART achieves up to 3× higher maximum operating frequency and 60% lower slice utilization in FPGA prototyping and up to 50% reduction in both area and CPD in an ASIC implementation.

ACKNOWLEDGMENTS

This research was supported by the Spanish Ministry of Science and Innovation (FPI Grant BES2017-079971 and contracts PID2019-107255GB-C21 and PID2019-105660RB-C22 / AEI / 10.13039 / 5011000 11033) and by the Catalan Government (contracts 2017-SGR-1414). This work also received funding from the European Union Horizon 2020 research and innovation programme under grant agreements number 779877 (Mont-Blanc 2020), 826647 (EPI) and 946002 (MEEP).

M. Moreto was also partially supported by the Spanish Ministry of Economy, Industry and Competitiveness under Ramón y Cajal fellowship No. RYC-2016-21104.

REFERENCES

[1] Yashar Asgarieh and Bill Lin. 2019. Smart-Hop Arbitration Request Propagation: Avoiding Quadratic Arbitration Complexity and False Negatives in SMART NoCs. *ACM Trans. Des. Autom. Electron. Syst.* 24, 6 (Nov. 2019), 1–25.

[2] Semiconductor Industry Association et al. 2008. International technology roadmap for semiconductors. *http://www.itrs.net* (2008).

[3] T. Krishna, Chia-Hsin Owen Chen, Woo Cheol Kwon, and Li-Shiuan Peh. 2013. Breaking the on-chip latency barrier using SMART. In *International Symposium on High Performance Computer Architecture (HPCA)*. 378–389.

[4] Amit Kumar, Li-Shiuan Peh, and Niraj K Jha. 2008. Token flow control. In *41st IEEE/ACM International Symposium on Microarchitecture*. 342–353.

[5] Amit Kumar, Li Shiuan Peh, Partha Kundu, and Niraj Kumar Jha. 2007. Express virtual channels: Towards the ideal interconnection fabric. In *ISCA'07: 34th Annual International Symposium on Computer Architecture*. 150–161.

[6] Amit Kumary, Partha Kunduz, Arvind P. Singhx, Li-Shiuan Pehy, and Niraj K. Jhay. 2007. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *2007 25th International Conference on Computer Design*. 63–70.

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

[8] Ye Lu, Changlin Chen, John McCanny, and Sakir Sezer. 2012. Design of interlock-free combined allocators for Networks-on-Chip. In *2012 IEEE International SOC Conference*. 358–363.

[9] Sheng Ma, Natalie Enright Jerger, and Zhiying Wang. 2012. Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip. In *IEEE International Symposium on High-Performance Comp Architecture*. 1–12.

[10] George Michelogiannakis, Dionisios Pnevmatikatos, and Manolis Katevenis. 2007. Approaching ideal NoC latency with pre-configured routes. In *First International Symposium on Networks-on-Chip (NOCS'07)*. IEEE, 153–162.

[11] Alireza Monemi, Farshad Khunjush, Maurizio Palesi, and Hamid Sarbazi-Azad. 2020. An enhanced dynamic weighted incremental technique for QoS support in NoC. *ACM Transactions on Parallel Computing (TOPC)* 7, 2 (2020), 1–31.

[12] Alireza Monemi, Jia Wei Tang, Maurizio Palesi, and Muhammad N Marsono. 2017. ProNoC: A low latency network-on-chip based many-core system-on-chip prototyping platform. *Microprocessors and Microsystems* 54 (2017), 60–74.

[13] M. K. Papamichael and J. C. Hoe. 2012. CONNECT: Re-examining conventional wisdom for designing NoCs in the context of FPGAs. In *Proceedings of the ACM/SIGDA international symposium on FPGA*. 37–46.

[14] Iván Pérez, Enrique Vallejo, and Ramón Beivide. 2018. Efficient Router Bypass via Hybrid Flow Control. In *11th International Workshop on Network on Chip Architectures (NoCArc)*. 1–6.

[15] Iván Pérez, Enrique Vallejo, and Ramón Beivide. 2019. SMART++: Reducing Cost and Improving Efficiency of Multi-hop Bypass in NoC Routers. In *International Symposium on Networks-on-Chip (NOCS)*. 5:1–5:8.

[16] Iván Pérez, Enrique Vallejo, Miquel Moreto, and Ramón Beivide. 2020. BST: A BookSim-Based Toolset to Simulate NoCs with Single- and Multi-Hop Bypass. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, Boston, MA, USA, 47–57.

[17] W Snyder, P Wasson, and D Galbi. Online; accessed may 2017. Verilator-Convert Verilog code to C++/SystemC. http://www.veripool.org/wiki/verilator.

[18] Ling Xin and Chiu-sing Choy. 2010. A low-latency NoC router with lookahead bypass. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 3981–3984.