FastTrackNoC: A DDR NoC with FastTrack Router Datapaths

Ahsen Ejaz and Ioannis Sourdis
Department of Computer Science and Engineering
Chalmers University of Technology, Gothenburg, Sweden
Email: {ahsen, sourdis}@chalmers.se

***Abstract*—This paper introduces FastTrackNoC, a Network-on-Chip router architecture that reduces latency by bypassing its switch traversal (ST) stage. FastTrackNoC adds a fast-track path between the head of a particular virtual channel (VC) buffer at each input port and the link of the opposite output. This allows non-turning flits to bypass ST when the required router resources are available. FastTrackNoC combines ST bypassing with existing techniques for reducing latency, namely, pipeline bypassing of control stages, precomputed routing and lookahead control signaling, to allow at best a flit to proceed directly to link traversal (LT). FastTrackNoC is applied to a Dual Data Rate (DDR) router in order to maximize throughput. Post place and route results in 28nm technology show that (i) compared to the current state of the art DDR NoCs, FastTrackNoC offers the same throughput and reduces average packet latency by 11-32% requiring up to 5% more power and (ii) compared to current state of the art Single Data Rate (SDR) NoCs, FastTrackNoC reduces packet latency by 9-40% and achieves 16-19% higher throughput with 5% higher power at the SDR NoC saturation point.**

I. Introduction

As current chips are able to integrate large amounts of resources, their interconnects become critical for their performance [1]. On-chip interconnection networks are expected to deliver low latency and high throughput within a limited power budget in order to accommodate the requirements of a system [2]–[5], however, this is a challenging task, especially for larger network sizes.

In the past decades, Network-on-chip (NoC) designs have improved their performance substantially. Packet latency has been reduced with improvements in various aspects of the network such as topology [6], routing algorithm [7] and the router architecture [2], [8]–[11]. On the other hand, network throughput has increased with the use of more efficient topologies [12] and sub-networks [13], [14] as well as with improvements in allocation [15]. Higher throughput can also be achieved by spliting switch traversal (ST) and link traversal (LT) to two pipeline stages, however, this puts the router control in the critical path [15]–[19]. Recent designs addressed this bottleneck by allowing the datapath of a router to be traversed twice in a cycle using both clock edges [19]–[21]. The control of such Dual Data Rate (DDR) NoC router utilizes an entire, longer cycle to make decisions, hence is no longer in the critical path, further improving NoC throughput to switching rates defined solely by a datapath stage (ST or LT) delay [21].

Although one could argue that the throughput of the NoC routers has reached a plateau, there is still room for improving routers latency. In theory, the latency for a single hop, i.e., crossing a tile, can be as short as the (register to register) delay of a link that traverses the tile dimension. On the other hand, current NoC routers have achieved to reduce their best-case hop latency to the time spent for ST plus LT [21]. This is achieved by a combination of techniques, such as precomputing routing [8], [9], lookahead (control) signalling [2], [10], and pipeline bypassing [2], [11]. Then, considering that ST and LT are pipelined and their delays are split evenly, the best case latency of the current state of the art NoC routers [21] is roughly twice the above theoretical minimum. It is worth noting that there exist interconnection approaches, such as using express links or routerless rings, that can reduce the minimum hop latency bellow *ST+LT*, however, express links change the network topology and therefore are orthogonal [6] to reducing routers latency, while the Routerless NoC introduces routing limitations, which need to be compensated for by extra wire resources limiting throughput [22].

This work introduces FastTrackNoC, a new NoC router architecture that strives to achieve a near minimal router latency, while maintaining the throughput achieved by previous DDR NoC designs. FastTrackNoC takes previously proposed pipeline bypassing NoC mechanisms a step forward and besides bypassing the SA control stages it offers bypassing of the ST stage, too. It does so by adding an extra FastTrack datapath between the head of a specific Virtual Channel (VC) at each input port of the router and its opposite output port. Then, incoming flits that do not turn have the opportunity to directly traverse the link using the FastTrack path. Such FastTrack traversal is possible for non-turning flits when particular router resources are available, i.e., input and output ports and the flits are stored at the head of VC buffer used for FastTrack. The proposed router architecture operates at DDR to offer the same throughput as current state-of-the-art DDR NoCs. Forcing the FastTrack path to start from the head of a particular input VC ensures that, besides the crossbar, the buffer- and input-multiplexing are avoided, too, so, the delay added to the link is mainly for multiplexing the FastTrack path with the regular path. Non-turning hops are very common in a network, particularly in larger network sizes and when minimal routing is used, therefore there are plenty of opportunities for FastTrack traversal in a network, especially at lower injection rates. Concisely, the contributions of this paper are the following:

- a new router architecture architecture, FastTrackNoC, that offers FastTrack paths to bypass switch traversal (ST) and is able to achieve a hop latency at best similar to the link traversal delay;
- A thorough evaluation and comparison against the current state of the art demonstrating 10-40% reduction in packet latency, while supporting the same maximum throughput;

The rest of this paper is organized as follows. Section II discusses related work in more detail. Section III describes the FastTrackNoC router architecture. Section IV details the experimental setup and presents evaluation results. Finally, Section V summarizes our conclusions.

II. Related Work

A brief overview of router architecture techniques for reducing packet latency and increasing network throughput is presented. In addition, the current state of the art (SDR and DDR) NoC routers used for comparison are described.

A. Increasing NoC throughput

There are three main approaches of improving throughput of single data rate (SDR) networks. They are: (i) use path diversity to avoid congestion [12], [13], (ii) improve matching quality of VC and switch

allocators [15] and (iii) increase network switching rate, i.e., clock frequency. The first two approaches improve utilization of network datapath resources to increase throughput and are orthogonal to the proposed router architecture.

A first step towards increasing the switching rate of a NoC router is to pipeline its datapath. Typically a NoC router is divided in two stages, namely, the switch and link traversal stages (ST, LT). Deeper pipelining of router's datapath can further increase throughput at the cost of higher latency. One such example is Intel's full custom NoC router which splits ST in two stages and shares a single crossbar among two lanes using it at [23]. Its simplified allocation and pipelined ST allow reduce its critical path to 15 FO4 delays, however, its 6-stage pipeline introduces high latency. Splitting router's datapath in just two stages, ST and LT, increases operating frequency and typically puts the control logic in the critical path, while the datapath has some slack [15], [17], [19]. As a consequence, the control of a router then defines the clock frequency of the network and the rate at which flits are routed. The fastest NoC router in literature that follows this approach is ShortPath, which uses among other techniques request queues for VC and Switch Allocation to reduce the number of simultaneous allocation request and simplify allocation logic [11]. Considering 2D mesh topology and 4 VCs, the critical path of ShortPath is estimated to be 25 FO4 delays, which is still about 20% slower than its ST delay. As a consequence, ShortPath offers about 20% lower throughput than what could be achieved if the critical path was on the ST [19].

DDR NoCs, address the above bottleneck by operating router's datapath at dual data rate using both clock edges. Then, the control has an entire, longer than ShortPath, cycle available to make decisions [19]–[21]. The clock period of a DDR NoC is 42 FO4 delays, twice its ST delay, and its switching rate half of that (21 FO4) [19]–[21]. As a consequence, the control logic is not in the critical path of a DDR NoC router and the ST delay defines the switching rate, which is about 20% higher than the fastest Single Data Rate NoC router, ShortPath.

B. Reducing NoC packet latency

There is a plethora of NoC designs that aim at minimizing packet latency. As mentioned in the previous section, the minimum latency per hop supported by current NoC router architectures is at best equal to the sum of their ST and LT delays. Then, changes in the network topology or the entire network structure can reduce latency further, but are either orthogonal or introduce significant limitations as explained next.

In the past, various networks have exploited a lower, relative to the router, link delay to transfer flits across multiple hops. This has been supported by richer topologies with longer links to non-neighboring routers, e.g. in flattened butterfly [12], multidrop express channels [6], and through express links [5], [24], [25]. Bypassing the ST stage, as the proposed here, is orthogonal to such topological changes and expected to reduce the latency of their routers. It is however noteworthy, that such solutions would increase the crossbar radix and in effect the ST delay increasing routers latency and limiting throughput. Finally, as more recent routers achieve delays comparable to the single-hop link delay [21], multi-hop links would need to be pipelined in order to maintain performance.

Another low-latency network is the Routerless NoC [22], which uses loops (rings) to interconnect its nodes. Its minimum latency per hop is equal to the delay of the link plus a 2:1 multiplexer. That is one 2:1 multiplexer delay less than the FastTrackNoC minimum latency or even equal to a FastTrackNoC that uses only a single clock edge.

TABLE I: State of the art NoC router architectures.

Network	Data Rate	Pipeline Bypass	# Pip. Stages	Routing rate	Min. hop latency
Intel [23]	(DDR)		5+LT	1/15 FO4	90 FO4
SCORPIO [2]	SDR	(SA)	3+LT	1/28 FO4	56 FO4
ShortPath [11]	SDR	SA	3+LT	1/25 FO4	50 FO4
DDRNoC [19]	DDR		2+LT	1/21 FO4	84 FO4
FreewayNoC [20]	DDR	(SA)	2+LT	1/21 FO4	42 FO4
HighwayNoC [21]	DDR	SA	2+LT	1/21 FO4	42 FO4
FastTrackNoC	DDR	SA, ST	2+LT	1/21 FO4	21 FO4

However, the Routerless NoC has several significant disadvantages. Using fixed loops limits the routing options. The Routerless NoC alleviates this problem by using n overlapping loops in an $n \times n$ network. However, this requires to partition the wire resources to n paths that are n times narrower substantially limiting throughput compared to conventional 2D-mesh networks with the equal wire resources. Moreover, Routerless NoC requires large buffers at their network interface to fit an entire packet in order to resolve conflicts. In turn, although Routerless NoCs can offer slightly lower latency than the proposed FastTrackNoC, their fixed loops either limit routing options and thus increase latency, or call for multiple overlapping loops which limit throughput considering equal wire resources.

The architecture of a NoC router has been improved in various ways to reduce latency. Pre-configuring preferred paths reduces ST delay, but has to tolerate misrouted, eagerly forwarded, dead flits [26]. XOR-based crossbars allows switching to be performed without waiting the arbitration result but suffers high costs in conflicts [27]. Less wasteful approaches have been the backbone of current state of the art NoC routers. Precomputing routing allows to perform virtual channel allocation (VA) immediately after a header flit arrives as the next route has already been computed by the upstream router [8]. Combined allocation [28] allows VA and switch allocation (SA) to be performed in parallel and saves a cycle in the router pipeline [9]. Lookahead (control) signalling enables allocation to start before the incoming flit is buffered [2], [10]. It is inspired by off-chip network designs [29] and uses a separate, narrow link to forward the control of a flit a cycle ahead of the data enabling the downstream router to process it one cycle earlier [2], [10]. Finally, pipeline bypassing allows incoming flits that find no contention to bypass some or all control stages of a router and proceed faster to the switch traversal stage [2], [11], [30].

Combining several of the above techniques, i.e., precomputed routing, speculative allocation, lookahead signalling, and pipeline bypassing, can result in a minimum latency of two stages (ST and LT) per hop. However, as mentioned in the previous paragraph for a Single Data Rate (SDR) NoC router the cycle time would be longer than the ST delay as it is limited by the control logic [15], [17], [19]. Operating the router datapath at DDR removes control from the critical path and allows faster switching rates [19]. Then, a DDR NoC router that employs precomputed routing, combined allocation, lookahead signalling, and pipeline bypassing, offers a minimum latency per hop equal to the delay of *ST+LT*. FastTrackNoC improves router latency further offering ST bypassing.

C. Summary

Table I summarizes the main characteristics of competing designs including performance estimations measured in FO4 delays[1]. The

[1]The reported performance results are either delays taken from the respective papers [23] or estimated based on post place and route results on 28nm technology, as explained in Section IV-A.

aforementioned Intel design offers a very fast router with oversimplified control, pipelined ST and a DDR crossbar, which achieves the fastest rate of routing flits (1/15 FO4), however without pipeline bypassing it suffers long hop latency of 90 FO4 delays [23]. Scorpio is a 3-stage SDR router that offers speculative allocation, lookahead signalling, and some pipeline bypassing support, and exhibits a cycle time of 28 FO4 delays [2]. ShortPath is a faster 3-stage SDR router with better pipeline bypassing and shorter critical path (25 FO4) than Scorpio [11]. DDRNoC is the first fully DDR router that increases the rate of routing flits to one flit per 21 FO4 (every half a cycle) offering higher throughput. It offers precomputed routeing, speculative SA, and lookahead signaling, however it has no pipeline bypassing support so it suffers high packet latency [19]. FreewayNoC improves on the DDRNoC offering limited in-network pipeline bypassing to non-turning flits, improving packet latency [20] and HighwayNoC adds bypassing support for flits that enter or exit the network. FastTrackNoC is applied to a DDR router so it maintains the fast switching rates and reduces the minimum router latency to half (21 FO4 delays) employing ST bypassing.

III. The FastTrackNoC Router Architecture

The FastTrackNoC router offers an extra bypassing path between every input port and its opposite output port enabling flits to proceed directly to the link without passing through the switch, thereby reducing latency. Its datapath uses both clock edges, i.e., it is able to route up to two flits per port per cycle at Dual Data Rate to enable faster switching rates. On the contrary, its control has an entire cycle available to make decisions. In order to minimize delay added to the link, the FastTrack (FT) path connects only the head of one specific input virtual channel, i.e., VC-0, to the output link. Then, if the way is free a non-turning flit registered in the input can directly traverse the link avoiding switch traversal (ST) as well as the allocation steps. Using the FT path enables flits to perform one hop doing only the LT stage when they do not turn; this is one hop per half a cycle at DDR. When FT traversal is not possible, incoming flits may still be able to bypass switch allocation (SA) and directly traverse the switch and the link when their way to the output port is free. This is possible for all VCs and hops including entrance and exit from the network, except in-network turns, and allows flits to be routed at one hop per cycle, similar to previous DDR NoC routers [21]. Flits unable to bypass SA follow the complete router pipeline (VS/SA, ST, LT) spending one cycle for allocation and two halves of a cycle for ST and LT resulting in two cycles for a hop. FastTrackNoC also offers lookahead signalling as it forwards control information of routed flits to the downstream router a cycle before the flits, to start earlier the allocation of datapath resources as well as to perform bypassing checks and next-route computation (NRC). In parallel to bypassing checks and allocation logic, a VC is selected out of the available free ones and allocated to successful header flits. Without loss of generality, our FastTrackNoC design considers a 2D-mesh network, with lookahead XY-routing, composed of routers with virtual-channels and credit-based flow control.

The top-level view of the FastTrackNoC router is shown in Figure 1. The datapath is composed of two stages: Switch Traversal (ST) and Link Traversal (LT), separated by the input VC buffers and the two output registers (one for each clock edge), which are multiplexed before the link. Alternate bypass paths also exist to support FT traversal by multiplexing flits stored in input VC-0 on the output link. Each stage can handle two flits per cycle, one at the high and one at the low phase of the clock. In addition, the FastTrackNoC router has the following control blocks: a combined allocator composed of the Virtual Channel Select (VS) and a Switch Allocation (SA), Next-Route-Computation (NRC) and bypassing check logic at the input and output ports.

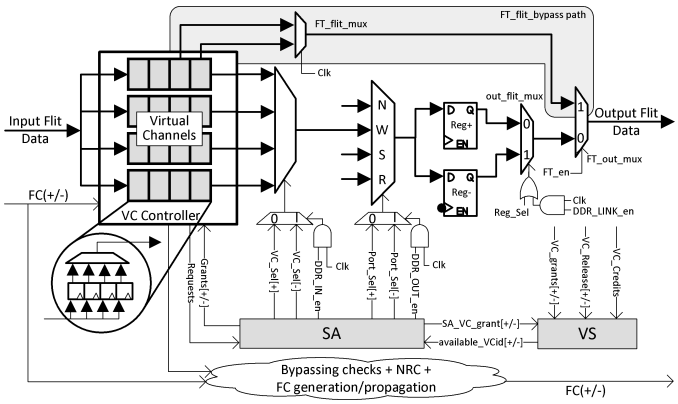


Fig. 1: The FastTrackNoC router architecture.

A. Router Datapath

The FastTrackNoC router datapath has two parts: the datapath of the data flits, which is further divided to the regular and the FastTrack path, and the datapath of the control bits forwarded separately to support lookahead signalling. Each of them is described separately below.

1) DDR Flit Datapath: The datapath of the flits carries flits from an input to an output port. It is composed of two parallel intra-router paths, called the regular datapath and the FastTrack datapath, which are multiplexed on inter-router output links.

Regular Flit Path: The regular datapath is composed of input VCs, crossbar switch, output registers and link wires and it is illustrated in Figure 1. This datapath is similar to the regular datapath in previous DDR NoC architectures [19]–[21]. The regular datapath can support two flits per cycle per port, one at each phase of the clock. At the input side, each register in a VC buffer can be written selectively, either at the negative or the positive clock edge, to store up-to two incoming flits per cycle. The multiplexer of the input VC buffer along with the input and output multiplexers of the crossbar then transfer flits stored in the input VCs to the requested output port. These multiplexers operate at DDR and can transfer up to two flits from one or two different VCs in a single clock cycle, i.e., one flit per clock phase. This is accomplished by changing the select signals of these multiplexers during each clock phase. After the output multiplexer of the crossbar, flits are registered in one of the two output registers ($Reg+$ and $Reg-$), one using the positive clock edge and the other the negative. Then, the contents of these registers are multiplexed to the link and sent to the downstream router at DDR.

FastTrack flit path: The FT datapath is parallel to the crossbar switch and the output registers, as shown in Figure 1, and transfers flits in FT mode. FT mode allows incoming flits to bypass SA and ST stages, in the absence of contention, and directly traverse the link in half a clock cycle. It offers FT support to flits traversing an in-network straight hop; in particular from West to East (shown in Figure 1), North to South and vice versa. FT support is also limited to flits at the head of input VC-0 but it is independent of the allocated output VC. In other words, while only VC-0 can transfer flits in the FT-mode, any downstream VC can receive these flits. This limitation is because offering full support, from each input VC to every output port would require large multiplexers, similar to the

router's crossbar, connected serially to the output link. Such a FT datapath would almost double the clock period of the router (by having delays similar to ST and LT in half a clock cycle), whereas, a key design objective of the FastTrackNoC is the network should operate at a clock frequency similar to previous DDR NoCs which do not have FT bypass paths and pipeline ST and LT, which then occupy consecutive halves of a clock cycle, as shown in of Figure 2d in the second half of cycle 2 and the first half of cycle 3. To this end, FT flit datapath is simplified exploiting the fact that it will only be active in the absence of contention and for more common in-network straight hops.

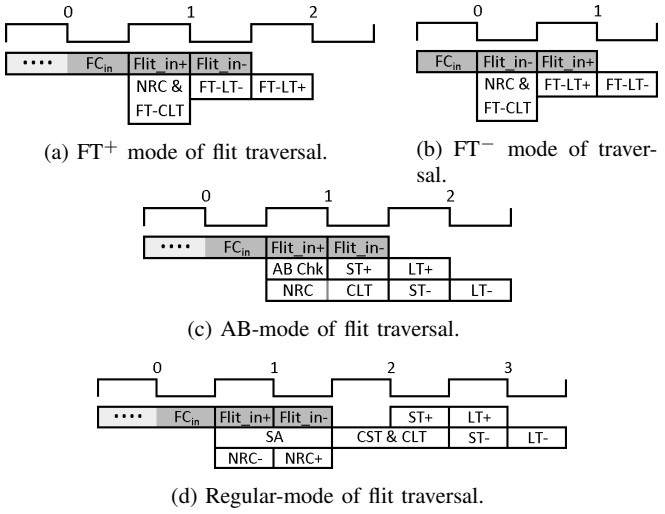
The simplified FT flit datapath of the proposed FastTrackNoC router only propagates incoming flits stored in the head registers 0 and 1 of an otherwise empty input VC-0 buffer. So, VC-0 FIFO implementation prioritizes storing its incoming flits in registers 0 and 1 when it is empty (contrary to a simple circular FIFO for other VCs) to facilitate FT traversal. This reduces the inputs of the FT_flit_mux, used to select incoming flits stored in input VC registers and shown at the top part of Figure 1, from number of registers in a VC buffer $\times$ number of VCs per input port (V), to only 2. Additionally, to route flits at DDR in FT-mode, the FT_input_mux also operates at DDR by having a gated-clock as its select signal. It is worth noting that, for an SDR NoC router, the FT path would only use the head register-0 of VC-0 and the FT_input_mux would not be needed. At the other end of the FT path, the FT_out_mux, is used to multiplex FT flits on the link as shown in Figure 1. This is a *2:1* multiplexing rather than *(P-1):1* as FT traversal is restricted to only in-network straight hops. The two aforementioned optimizations at the two ends of the FT path, significantly reduce the delay of the FT_flit_bypass datapath, enabling low latency FT DDR flit traversal. Compared to DDR NoCs without FT support, the delay of the intra-router FT wires as well as the delay of the two 2:1 multiplexers (FT_flit_mux, FT_out_mux) is added to the LT delay. This may reduce slightly either the network clock frequency or the link length. Our implementation opts for the latter to ensure the FastTrackNoC can operate at a clock frequency defined by the router's crossbar delays and maximize throughput. This is further analyzed in Section IV-D.

Altogether, FastTrackNoC routers support three modes of routing a flit, regular, allocation bypassing and FastTrack. The first two modes are similar to the previously proposed HighwayNoC [21] and therefore briefly described bellow, while the latter is added for the FT support and is discussed in detail. A mode is dynamically selected based on the VC and route of incoming flits, the local traffic conditions and downstream buffer space availability. All three of these modes can route flits at DDR (as shown in Figure 2) as well as at single data rate (SDR).

In the regular-mode, flits cannot bypass any router stage. SA is initially performed to select winning flits which are then allowed access to router crossbar and subsequently, the link, as shown in Figure 2d, for ST and LT, respectively. Regular mode is the slowest mode of flit traversal requiring at least two cycles per hop and can be used by flits at all input ports and VCs to all outputs.

In the allocation bypassing mode, henceforth denoted as AB-mode, flits are able to bypass the SA stage but not the ST stage. Here, incoming flits initiate ST and then LT as soon as they are received, as shown in Figure 2c. AB-mode is supported from all input ports and VCs and to all outputs, including local ports, except for in-network turns (i.e. xy and yx turns). Flits using this mode propagate the network at a rate of one hop per cycle.

In the FastTrack mode, henceforth denoted as FT-mode, flits are



(a) FT^{+} mode of flit traversal.

(b) FT^{-} mode of traversal.

(c) AB-mode of flit traversal.

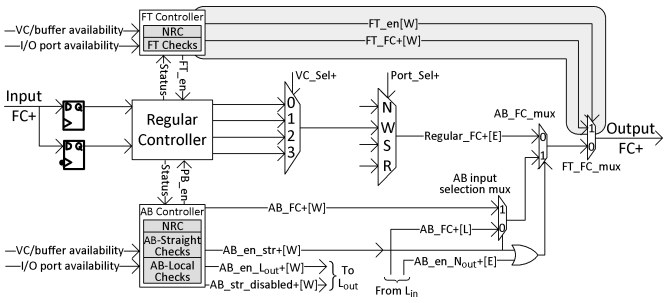
(d) Regular-mode of flit traversal.

Fig. 2: Modes of flit traversal in FastTrackNoC.

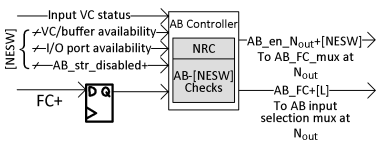
able to bypass both SA and ST pipeline stages. A received flit can use the FT bypass path to immediately reach the output port for LT in half a clock cycle, as described above. FT-mode has two further types depending on the clock edge at which the forwarded control information of incoming flits is received, as shown in Figures 2a and 2b and described later in section III-A2. FT-mode is only supported for incoming flits stored in registers-0 and -1 of VC-0 buffer at non-local input ports (N_{in}) and is activated for in-network non-turning hops. Flits using this mode propagate the network at a rate of two hops per cycle.

2) Datapath of Forwarded Control signals: The FastTrackNoC uses lookahead signalling [29] to forward control information ahead of the flits. This forwarded control (FC) information is received by the downstream router ahead of the flits using separate link wires. This information not only allows allocation stage in a router to be overlapped with LT stage of the upstream router, saving a cycle in regular mode of flit traversal, but it also informs downstream input VC registers to store subsequent incoming flits at the appropriate clock edges. There are two sets of FC signals, one for flit traversing the link during the high phase of the clock cycle (FC+) and one for flit traversing in the low phase (FC-). Within the router, there three parallel FC paths for each of the two FC signals, one for each of the routing modes, namely the regular, AB, and FT forwarded control paths. The signals of the three FC paths are generated by the respective controller for regular, AB and FT routing, carry a set of FC+ and FC- signals, and are multiplexed on the output link as shown in Figure 3a.

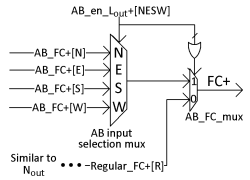
Regular Forwarded Control Paths: The regular FC path transfers once every cycle control information of up to two flits routed in regular mode in the two phases of the subsequent cycle. So, FC contains two sets of signals, one for the flit traversing the link during the high phase of the clock cycle (FC+), shown in Figure 3a, and one for the flit traversing in the low phase (FC-). When a flit at the input port is granted access to the switch by SA, its FC signal first undergo a switch traversal, called control switch traversal (CST), to reach the requested output port and then they are multiplexed on the link for a control link traversal (CLT). CST and CLT together take one complete clock cycle, as shown in Figure 2d and are not pipelined as opposed to the ST ad LT of the flits.



(a) Regular, AB and FT FC+ paths from a non-local input port (West) to its straight non-local output port (East) in the FastTrackNoC.



(b) Bypassing FC+ path and enable signals originating from the AB controller at the local input port.



(c) Multiplexing of regular and AB FC+ signals at the local output.

Fig. 3: FT and AB controllers and FC+ paths at the FastTrackNoC input and output ports. FC- paths are identical to FC+ paths.

Forwarded Control Paths for Allocation Bypassing Mode: Control information of flits traversing the router in Allocation Bypassing (AB) mode is forwarded also once a cycle using separate FC paths in the router, which skip CST and are multiplexed with the regular FC path at the output port. As in HighwayNoC [21], SA bypassing is supported for flits that either traverse an in-network straight hop, enter or exit the network. For in-network straight hops, each non-local input port, N_{in}, sends FC of respective flits to its straight non-local output N_{out}, as shown in Figure 3a for West input and East output ports. For flits entering the network, FC of AB flits goes from the local input port, L_{in}, to each N_{out}. Finally, for exiting flits, each N_{in} sends FC to the local output port, L_{out}.

The AB controller checks the respective input and output port availability, as well as for available VC and credits, to determine whether propagating flits in AB-mode is possible. These checks are performed in parallel to the regular CST. A successful AB check at an input port will enable FC of AB flits to be sent through the 2:1 *AB_FC_mux* and the *FT_FC_mux* directly to CLT, overriding the CST of regular FC, as depicted in Figure 3a. As shown in Figure 3, the local input and output ports have different AB support than the in-network ports. When bypassing from the L_{in} to the network, bypassing FC signals generated after AB checks (shown in Figure 3b) are multiplexed with the bypassing FC signals of the N_{in} using the *AB_input_selection_mux*, with higher priority for flits coming from the N_{in}, as explained in Section III-C2. However, when bypassing to the L_{out}, a 4:1 multiplexer is needed before the 2:1 *AB_FC_mux* (shown in Figure 3c) to select one of the N_{in} for bypassing using static priority. This adds latency to the bypassing FC path which is however compensated by the shorter CLT of the L_{out}; that is because there is a shorter link (relative to inter-router links) connecting the local output port of a router to the network interface.

Forwarded Control Paths for FastTrack Mode: FC signals of incoming flits routed in FT-mode are forwarded through separate paths in a router, called the *FT_FC_Bypass* paths, as shown at the top of Figure 3a. These paths carry FC signals from non-local input port

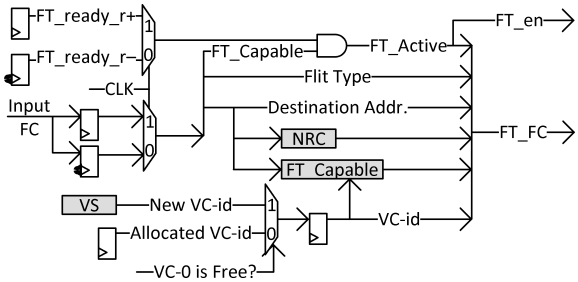


Fig. 4: FT Controller updates FT_FC information and sets FT_en signal before FT_CLT.

(N_{in}) to their straight non-local output port (N_{out}). They bypass the logic of the AB and regular FC paths and are multiplexed with them, last through a 2:1 multiplexer (*FT_FC_mux*) before they traverse the link.

The FT controller at each N_{in} checks the respective input and output port availability, as well as for available downstream VCs and credits, to determine whether FT traversal is possible, as discussed in detail in section III-C3. A successful FT check at an input port enables received FC signals of incoming flits to be updated with NRC result and output VC-id and transferred through the FT_FC_mux directly for CLT in half a clock cycle. Since ⟨FT_FC_Bypass + CLT⟩ (FT_CLT) is a half cycle path, it can be active during any one of the two phases of a cycle, right after the incoming FC signals are registered, as shown in Figures 2a and 2b. In order to select the correct incoming FC to be sent forward for FT_CLT a 2:1 multiplexer is used in FT controller, as shown in Figure 4.

To support the FT-mode, the two following FC signals are used. The first signal, FT_Active, indicates the special case where an incoming FC, announcing an arriving FT-routed flit, is registered downstream at the falling edge of the clock. FT_Active acts as load enable to the negative edge triggered FC register at the input port, shown on the left side of Figure 3a. The second signal, called FT_Capable, carries part of the pre-computed FT eligibility, check which is used to speed up control logic for FT_CLT. It indicates incoming flit(s) will be stored in input VC-0 and will traverse an in-network straight hop, as described in detail in Section III-C3.

The FT_FC signals are generated in the FT Controller at the input illustrated in Figure 4. Although most information is locally registered, Next Route computation is in the critical path. FastTrackNoC simplifies NRC for FT traversal to reduce the delay of updating the control signals of a flits routed in FT-mode as explained in Section III-C4. Moreover, multiplexing FT_FC signals on link for FT_CLT after AB multiplexer (AB_FC_mux), as shown in Figure 3a, and substantially narrow width of FC signals compared to flit datapath also contribute to achieving reducing the delay added to the link.

FC Signals Format: The FC signals at each port contain the following fields: two sets of VC-ids and NRC results (one for each incoming flit), encoded flit types of two flits, destination addresses of header flits required for the NRC when bypassing (as explained in Section III-C4), an *FT_Active* bit and an *FT_Capable* bit used to route flits in FT-mode. Moreover, similar to the HighwayNoC, various optimizations are performed to reduce the number of FC bits. In particular, only one header flit to bypass per cycle so only one destination address needs to be carried by the FC, and when non-header flits are transmitted, the NRC field is not used and its bits are reused to distinguish between body and tail flits.

B. Zero-Load Latency Analysis

Next, the Zero-Load Latency (ZLL) of FastTrackNoC, HighwayNoC and ShortPath networks are analyzed and compared to the minimum achievable latency of a direct connection between the source and the destination nodes using a 128-bit wide pipelined link, similar to links in other networks. Although a direct connection with an un-pipelined link would offer lower latency for single flits, however, it will have high serialization latency for packets with multiple flits and therefore is considered a worse option. Moreover, this analysis also takes into account the maximum operating frequency of each network, implemented as described in our evaluation section. A hop is considered to be 2.1 mm long, which is the longest hop distance supported by FastTrackNoC at its maximum frequency. For this reason, the distance between pipeline stages of the ideal network is also set to 2.1 mm. A pipeline stage of the ideal network has a delay of 15.6 FO4 stages and a operates at 4.22 GHz.

The ZLL of a FastTrackNoC packet is as follows: FC is received at local input at rising clock edge; entering and exiting the network takes one cycle each because of AB support; in-network straight hops take half a cycle because of FT support; in-network turns take two cycles because they require SA; extra half cycle is required when flit exits the network at an odd hop because the exit node will receive flits in FT^{-} mode (shown in Figure 2b) and process them in the next cycle in either regular or AB mode; for the same reason, extra half cycle is also required when turn is at an odd hop; the serialization latency is half a cycle per flit:

$$ZLL_{FastTrackNoC} = \lceil hops/2 \rceil + 1 + 1.5 \times hops_{turn} + odd\_hops_{turn}/2 + N/2 \ cycles$$

The ZLL of a HighwayNoC packet is as follows: FC signals are registered at local input at rising clock edge; each non-turning hop takes a cycle because of AB support; turning hops require two cycles; the serialization latency is half a cycle per flit:

$$ZLL_{HighwayNoC} = hops + hops_{turn} + N/2 \ cycles$$

The ShortPath router which uses dynamic pipeline bypassing to reduce pipeline stages to 2 (VA\SA\ST in one cycle and LT in the other) has a ZLL of:

$$ZLL_{ShortPath} = 2 \times hops + N - 1 \ cycles$$

Finally, the ideal latency of a pipelined direct connection is one cycle per hop, considering one less hop compared to 2D mesh networks because L_{in} to N_{out} hop is no longer required, and serialization latency of one cycle per flit:

$$IdealLatency = hops + N - 2 \ cycles$$

Taking into account the cycle time of these networks and the above ZLL equations, we perform a sensitivity analysis of the ZLL with respect to the number of hops a packet traverses. We consider a 3-flit packet and assume it always takes a turn at an odd hop, if the hop count allows. As shown in Figure 5, the ShortPath network scales worse than HighwayNoC and FastTrackNoC and its latency is 3.3× higher than ideal latency for large number of hops. This is because it introduces control logic in its critical path which leads to slower clock and it has minimum hop latency of 50 FO4 delays as it requires all flits propagate both ST and LT stages. The HighwayNoC operates its datapath at DDR to have a critical path independent of control logic, which improves network clock frequency and reduces hop latency to 42 FO4 delays. Although HighwayNoC offers lower ZLL compared to ShortPath, it still has a widening gap compared to the ideal because it also requires all flits to propagate both ST and LT stages. With large number of hops, the ZLL through the HighwayNoC can be 2.9 × higher than the ideal. Finally, FastTrackNoC significantly reduces the latency gap to the ideal by allowing flits to bypass both, SA and ST stages. It operates at the same clock frequency as the HighwayNoC and has a minimum hop latency of 21 FO4 delays. This enables the FastTrackNoC ZLL to be 40% lower than the HighwayNoC and 75% higher than the ideal for large number of hops. More importantly, the ZLL of FastTrackNoC also scales better than other SDR and DDR networks.

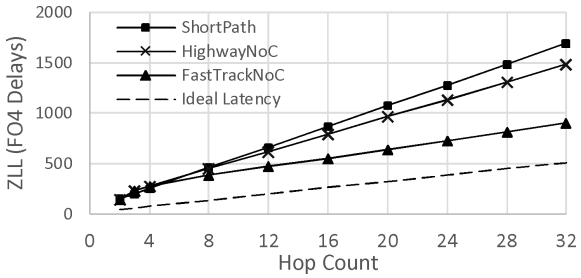


Fig. 5: ZLL analysis with respect to hop count of the FastTrackNoC, HighwayNoC and the ShortPath networks versus an ideal pipelined connection.

C. Router Control

The control of the FastTrackNoC router is described next. First the DDR allocation algorithm is described. Then the control logic for FT and AB modes of flit traversal is discussed. Finally, the route computation and flow control are discussed.

1) Combined VC and Switch Allocation: The FastTrackNoC utilizes a combined allocator which grants requesting flits access to the switch and assigns available output VCs to head flits of new packets. It is composed of VC Selection (VS) and Switch Allocation (SA) modules as depicted in the top of Figure 1. Overall, VS and SA architectures in FastTrackNoC are quite similar to the HighwayNoC [21] and are only briefly described below.

The VS performs V:2 fixed priority arbitration of available VCs (unassigned downstream VCs with credits) to select up to two VCs which are provided to new packets in the next cycle, when their header flits bypass SA or ST stages or win SA. VS in FastTrackNoC uses fixed priority arbitration and give packets higher priority access to output VC-0 which supports FT traversal, thereby improving chances for flits to be routed in the FT-mode.

Parallel to the VS, the SA receives requests for switch access from flits going to particular outputs.A head flit SA request is considered only if the VS indicates one or more available downstream VCs. Similarly, a non-head flit SA request is first checked for availability of credits in the assigned output VC. Besides filtering out requests that lack a VC or credits, SA also ignores requests from flits that propagate using FT or AB modes. All remaining requests undergo allocation using a separable input first dual grant switch allocator. In the first stage, input arbitration is performed on allocation requests from input VCs. This uses P *V:2* arbiters, one per input port. After input arbitration, up to two winning requests (one for each clock phase) from each input port, undergo *P:1* output arbitration using two different sets of output arbiters, one per clock phase. Finally, SA uses additional logic to allow two flits of the same packet to be granted access to the switch in DDR mode [19]. When an input VC receives an SA grant, a second packet flit can follow if available under the condition that (i) there is enough space in the downstream

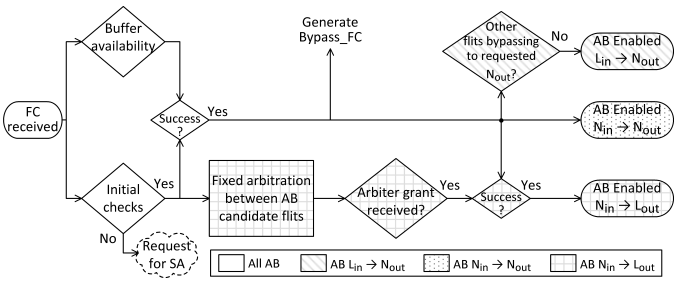


Fig. 6: Checks performed in order to guarantee conflict free propagation of flits in AB-mode.

VC buffer and (ii) the required input and output ports have not been promised by the SA to another flit.

2) Allocation Bypassing Controller: Similar to its predecessor, the HighwayNoC, FastTrackNoC also reduces end-to-end packet latency by allowing flits to skip SA stage in the absence of contention, as shown in Figure 2c. Contrary to the FT-mode which offers latency of half a cycle per hop to flits propagating an in-network straight hop from specific input VCs, AB-mode offers latency of one cycle per hop but to all except the in-network turning hops and from all input VCs. FastTrackNoC offers AB-mode in spite of the faster FT-mode, to reduce latency of flits which are unable to use the more restrictive FT-mode of traversal. Our implementation of AB mode is similar to the HighwayNoC [21], so, it is only briefly described below.

AB-mode allows incoming flits to bypass SA stage in three cases: (i) when flits go straight through a router, (ii) when flits enter the network, and (iii) when flits exit the network. It is not offered for in-network turns because this would require more complex AB checks and arbitration or kill logic to manage competing flits attempting to bypass to the same output at the same time. This extra control logic does not fit the available timing budget, which is constrained by datapath delays and should be independent of any control overhead.

AB check modules at input ports, shown in Figure 3, determine eligibility of incoming flits to bypass the SA stage. As shown in Figure 6, the checks performed are categorized into two types: (i) initial bypassing checks, and (ii) buffer availability checks. Initial bypassing checks use simple logic on registered information and passing them does not guarantee bypassing. Instead, they are used to filter out SA requests by incoming flits, holding back those that pass these initial checks. They are used because performing complete AB check before SA would be too slow to fit the target cycle time. These checks verify that:

- The flit traverses in a direction which supports bypassing;
- There are no buffered flits in the input VC which will receive the incoming flit (to ensure in-order delivery of flits in a packet).
- FT-mode is inactive for the input port that receives flits and the requested output port in the cycle when flits are received.
- The input port is free, i.e. for incoming $flit^+$ ($flit^-$), the ST^+ (ST^-) timeslot for the input multiplexer of the crossbar is not allocated by the SA to any input VC;
- The output port is free, i.e. for incoming $flit^+$ ($flit^-$), the ST^+ (ST^-) timeslot for the output multiplexer of the crossbar (which corresponds to LT^+ (LT^-) from the output port) is not allocated by the SA to any input port.

As shown in Figure 6, parallel to the initial AB checks, the received FC bits of incoming head and body/tail flits are used to check downstream *buffer availability* by accessing counters of available VCs and credits, respectively. Higher priority access to available downstream VCs is given to AB header flits already in the network over those coming from the local input. However, none of the AB header flits has priority to available VCs over non-bypassing flits.

Additional bypassing checks need to be performed for flits that enter or exit the network, as depicted in Figure 6. Bypassing from the L_{in} to a N_{out} is allowed only when there is no other flit, already in-network, bypassing to the same output. For flits exiting the network, *P-1:1* fixed priority arbiters are used to resolve contention if multiple incoming flits fulfill initial bypassing checks and buffering criteria to bypass to the L_{out}. The delay of these arbiters can be easily accommodated exploiting the slack present in the shorter L_{out} link.

All AB checks and generation of new bypassing FC signals occur in parallel to each other and in parallel to the normal CST path after which the FC bits reach the *AB_input_selection_mux* as shown in Figure 3. Moreover, after a successful AB check, the SA aligns input and output multiplexers of the crossbar to the input VC of the bypassing flit to enable ST during the second half of the current cycle or the first half of the next, as shown in Figure 2c.

3) FastTrack Controller: The FastTrackNoC utilizes the FT-mode to enable incoming flits to bypass both SA and ST stages and traverse an in-network straight hop in half a clock cycle, to reduce packet latency. At the same time, it also aims for the critical path of the router to be defined by its datapath and not by its control logic, to maintain high routing rate. FastTrackNoC achieves this by pre-computing checks required to enable FT-mode in a router.

At each N_{in} FastTrackNoC implements a FT controller which controls the flow of flits in FT-mode in a router. When FC signals are received at N_{in}, FT controller needs to verify if incoming flits should be allowed to undergo FT traversal. This requires checks which can be categorized into two types: (i) to ensure incoming flits meet FT criteria and (ii) to ensure internal state of the router is conducive for FT traversal. Passing these checks activates the FT-mode for incoming flits and sets the FT_en control signal (shown in Figure 3a) to multiplex FT_FC of these flits on the link through FT_FC_mux for FT_CLT.

The first category of checks should be performed on received FC information to determine FT eligibility of corresponding incomng flits. The FT controller should verify (i) valid incoming flits using flit type, (ii) flits will be stored in an input VC which supports FT-mode (i.e. VC-0) and (iii) flits will propagate a straight hop. This should be checked for up to two incoming flits to identify if one or both of these flits can traverse the hop in FT-mode. Performing these checks before generating the select signals of FT_FC_mux for FT_CLT increases network clock period by introducing control logic in the FC datapath, which goes against one of the design goals of the FastTrackNoC. To avoid this problem, FastTrackNoC simplifies and pre-computes these checks in the upstream router and transmits the result downstream in form of an additional signal, called the $FT\_Capable$ signal, as part of the FC information.

Pre-computing eligibility of incoming flits in the upstream router to generate the FT_Capable signal is possible because AB and regular modes offer sufficient time for the above mentioned checks to be completed parallel to other operations.However, in FT-mode of traversal, where FT_Capable signal needs to be computed and multiplexed through FT_FC_mux for FT_CLT in half a clock cycle, (pre-)computing it is more challenging because of the limited timing budget. Here, FastTrackNoC simplifies the (pre-)computation by utilizing the fact that the two flits being transferred are from the same specific VC which allows FT-mode (i.e. VC-0) and therefore are part of the same packet, will be stored in the same VC and routed to the same output port in the downstream router. So, the allocated

VC-id and the route for only one of the two transmitted flits need to be matched to a hard-coded value to generate the FT_Capable signal. Moreover, the select signal of the FT_FC_mux (FT_en) can be used to identify valid flits traversing in FT-mode, instead of checking multiple bits of flit type for each of the two propagating flits.

Furthermore, FastTrackNoC also restricts FT traversal when FC information indicates two flits per port and only one of the two is eligible for FT traversal. In other words, FT_Capable signal is set to indicate FT eligibility when either both propagating flits are FT eligible or there is only one propagating flit, occupying a single LT timeslot, which is FT eligible, while the other LT timeslot is idle. This simplifies the logic for FT_capable signal and reduces FT_Capable bits in the link from two to one, while increasing average latency by at most 1%.

We now turn to the second category of checks performed by the FT controller, which determine if internal state of a router, in terms of contention on output links and available downstream buffering resources, is conducive for FT traversal. The registered result of these checks is ANDed with the received FT_Capable signal to generate the FT_en select signal of the FT_FC_mux to allow FT_CLT, all in half a clock cycle.

Evaluating internal state of a router after FC information is received, again affects the router critical path by delaying FT_en signal, and reduces network operating frequency. Here too, the solution is to pre-compute the router state and have the result ready in form of a single bit indicating if FT-mode could be enabled for eligible incoming flits. Since FT-mode is supported for FC signals which can be received at rising (FT^+ mode as shown in Figure 2a) and falling (FT^- mode as shown in Figure 2b) clock edges, internal state of the router needs to be computed twice in a cycle as well. More precisely, for FT^+ mode, router state is evaluated in the cycle before FC signals are received (i.e. cycle 0 in Figure 2a) and it is registered in *FT_ready_r+* at the rising clock edge to be used during the high phase of the next clock cycle (i.e. in the first half of cycle 1 in Figure 2a). In case of FT^- mode, router state is evaluated half a cycle before FC signals are received and it is registered in *FT_ready_r-* at the falling clock edge (i.e. in the first half of cycle 0 in Figure 2b), to be used during the subsequent low phase of the clock (i.e. in the second half of cycle 0 in Figure 2b). The internal state of a router is judged to be conducive for FT traversal by verifying that:

- There will be no buffered flits in input VC-0 when incoming flits are received in the next cycle to ensure these flits can be stored in registers which support FT traversal (i.e. VC-0 registers 0 and 1) and to guarantee in-order delivery of flits in a packet.
- The input port will be free for a complete cycle starting from the clock edge at which FC signals are received.
- The output port will be free for FT_CLT for one complete cycle after FC signals are received for FT^+ mode (i.e. cycle 1 in Figure 2a) and during the cycle when FC signals are received for FT^- mode (i.e. cycle 0 in Figure 2b).
- The output port will be free for FT_LT, for one complete cycle, half a clock cycle after FC signals are received for FT_LT- and FT_LT+ as shown in Figures 2a and 2b.
- Free downstream VCs will be available in case incoming flit is a header.
- At least two credits of the allocated output VC will be available for non-header incoming flits
- For FT^- mode, the input port should not be requesting SA in the cycle when FC information is received, in order to keep the upstream credit lines free in the next clock cycle. This is because when FT^- mode is activated (as in the second half of cycle 0 in Figure 2b), then the input port will need to return credits upstream in the next cycle (i.e. in cycle 1 in Figure 2b). More details are provided in Section III-C5.

Although above mentioned checks to evaluate router state are in essence mostly the same for FT^+ and FT^- modes, their implementation differs because they occur in different halves of a cycle and the state of a router can change from one half of a cycle to the next. For instance, when output port availability is checked for FT^+ mode, it is sufficient to verify that the SA has not allocated the required LT timeslots to any requesting flit. But for FT^- mode, in addition to SA grants, it is also important to verify AB mode from L_{in} to the required N_{out} is not under way either. This is because FC signals received in the middle of a clock cycle should not traverse in FT^- mode if it interrupts an already initiated transfer of flits in the same cycle using either regular or AB modes. For the same reason FT^- mode also has lower priority access to available output VCs for header flits, compared to flits being allocated by the SA. This priority is enforced while checking for available VCs to generate FT_ready_r-.

When the result of pre-computed FT eligibility checks, conveyed downstream via FT_Capable signal, and the evaluated internal state of a router allow then FT-mode of traversal is activated and FT_en control signal is set by a 2-input AND gate, as shown in Figure 4. For flits allowed FT-mode of traversal, this also disables AB to the involved output port and filters out SA requests by these flits from being considered.

When FT-mode of traversal is not allowed, incoming flits first try to traverse the router in AB-mode and then regular mode. However, the timing of when these two modes of traversal are used differs for FC information received at the two clock edges. When FC information is received at the beginning of a cycle, AB checks are also performed in parallel to verifying the received FT_Capable signal and the evaluated router state. If flits cannot be routed in FT^+ mode, then the FastTrackNoC router attempts to router them in the same cycle using AB-mode, if required conditions for AB are met. For FC information received at falling clock edge, if FT^- mode is not allowed then the FastTrackNoC router will not attempt to transfer these flits in AB-mode until the next cycle because AB control paths require a complete cycle for the checks to be completed and FC information transmitted. If AB fails as well, then flits request SA in the current or the next cycle depending on the cause of failure as is described in more detail in Section III-C2.

4) Next Route Computation: Next Route Computation (NRC) is performed only for header flits and its result is stored in the input VC state registers to be used by the subsequent flits of the packet. Its timing differs for flits routed in regular, AB and FT modes. Bypassing (FT and AB) flits have their destination address carried by the FC. Then NRC is performed after FC is registered and before the new FC is created and sent to the downstream router. More precisely, based on Figure 2a (2a) for FT^+ mode (FT^- mode) of traversal, NRC is performed in first half of cycle 1 (second half of cycle 0), before FT_FC is generated and multiplexed over the link for FT_CLT. For AB mode, NRC is performed in first half of cycle 1, parallel to AB checks and half a cycle before updated AB_FC is multiplexed over the link for CLT (in second half of cycle 1), as shown in Figure 2c.

Non-bypassing flits that are transmitted in the low phase of the clock use the same approach as bypassing flits in order to have NRC ready together with allocation results. This is presented in Figure 2d where NRC- for flit_in- is performed in first half of cycle 1. On the contrary, non-bypassing header flits sent during the high phase of the clock are treated differently. For these flits, the FC does not need

to carry their destination address. The header flit arrives during the first half of the cycle and carries its own destination address which is registered at the falling edge of the clock. Then, NRC+ for flit_in+ is performed during the second half of the cycle to be ready together with their allocation result, as shown in Figure 2d.

In FT and AB modes of traversal, tight time budget of the bypassing logic, which has NRC of bypassing flits in sequence with the bypassing multiplexers and CLT, limits the routing algorithm alternatives. To make matters worse, in FT mode, NRC, FT_FC multiplexing and FT_CLT, all must fit in half a clock cycle. So, when a bypassing mode is selected, the routing algorithm needs to be simple to fit in the available time budget (about 5-6 FO4 stages). In our implementation we use XY routing, but any other (dimension-order) routing algorithm with similar complexity could also be used. To further simplify NRC for FT mode, FastTrackNoC implementation utilizes the fact that with simple dimension-order routing (DOR), a single downstream router-ID can be hard coded in each N_{in} corresponding to its straight N_{out}. This optimizes NRC logic delay for FT_FC signals. This is not applicable to AB mode where the downstream router-ID first needs to be calculated before NRC is performed, e.g., in the case of AB from L_{in} to N_{out}.

However, when a flit is not bypassing then half a cycle is available for NRC, which allows more complex routing algorithms to be implemented. On one hand, pipeline bypassing is enabled when there is no congestion. On the other hand, advanced algorithms yield most of their performance benefits when there is contention and congestion. Therefore, FastTrackNoC could use more advanced algorithms when a header flit is routed normally and fall back to dimension-order routing when bypassing, as long as deadlocks are avoided.

5) Flow Control and Minimum Buffer Size: FastTrackNoC uses credit based flow control, with one wire per VC indicating to the upstream router the release of VC credits and an additional wire per port to signal the release of two credits of one VC. Except for the FT-mode, all other modes of traversal have one complete cycle available to transmit the credit upstream and to update the credit counters. FT-mode is different because it is activated in the middle of a clock cycle and it has only half of a cycle available to transmit the credit upstream and to update the credit counters, which is insufficient. In this case the credit is registered locally and then transmitted upstream in the next cycle.

Moreover, in case of a regular flit propagation, credits can be reused after four clock cycles. However, for AB mode, credits can be reused after two clock cycles because the flits do not require SA pipeline stage in either of the two routers. Finally, for FT mode, credits can be reused after 2 clock cycles after accounting for the delay in transmitting the credit for FT- mode.

D. Timing Example

The timing of a FastTrackNoC router is described using an example of a five-flit packet traversing three routers in FastTrackNoC network, illustrated in Figure 7. In our example, we consider all three routers are aligned and are neither source nor destination nodes of the packet. Therefore flits passing through them are routed straight and have the opportunity to use all three modes of traversal (FT, AB and regular mode). In this example we do not consider in-network turns and flits entering and exiting the network because FT-mode is not supported on these routes, and flits in such cases are routed in a manner similar to HighwayNoC [21]. Note that the LT and ST for flits that traverse the link in the high clock phase are marked in the example with "+", and for flits that traverse the link in the low clock phase are marked with "-".

In cycle zero, Router-1 receives the FC of the first two flits (H, B1) which carries flit types, assigned VC, next route, destination and an asserted FT_Capable bit indicating the flits are eligible for FT traversal. As the flits enters Router-1 from an N_{in}, so depending on the feasibility of FT traversal, based on the router state pre-evaluated in cycle 0, and AB checks performed in cycle 1, they can either undergo or bypass SA in cycle 1. Since FT checks pass, incoming flits are allowed to bypass SA and ST stages. Moreover, the data flits of H and B1 arrive and get stored in registers 0 and 1 of an empty input VC-0 buffer in the first and second half of cycle 1, respectively. As the first flit is a header, NRC is performed and an output VC-id is acquired from registered output of VC select (VS). Additionally, FT_Capable bit is set because output VC-0 has been allocated to the packet and it traverses an in-network straight hop in the downstream router. Moreover, FT_Active bit is set to indicate output FC signals should be registered at the falling edge of cycle 1 in router-2. After FT_FC is updated using the result of NRC and the allocated output VC, it reaches the FT_FC_mux through the FT_FC_Bypass path, shown in Figure 3a. At the same time, FT_en_str select signal sets the FT_FC_mux to allow FC_FC signals to traverse the link (FT_CLT) and reach router-2 at the falling edge of cycle 1. In the second half of cycle 1 the H flit stored in VC-0 register 0 uses the FT_Flit_Bypass (shown in Figure 1) to reach the FT_output_mux which allows it traverses the link (FT_LT-). B1 follows half a cycle later. The remaining flits are routed through router-1 in a similar way.

Following again H and B1 flits, their control information arrives in router-2 at the falling edge of cycle 1 while the data flits are received in the second half of cycle 1 and first half of cycle 2. During the second half of cycle 1, these flits are unable to initiate FT-mode of traversal in router-2 because busy output port disables FT-mode. Although these flits are unable to propagate in FT-mode, they still attempt to utilize the AB-mode in the next cycle. In first half of cycle 2, AB check for these flits is successful, indicating no contention on router switch and link, allowing SA to be bypassed. In parallel, to AB check, NRC is performed, output VC-id acquired from VS and FT_Capable bit is set because once again output VC-0 has been allocated to the packet and it traverses an in-network straight hop in the downstream router. The updated FC signals bypass regular CST using the AB_FC_mux (shown in Figure 1), go through the FT_FC_mux and traverse the link (CLT) to reach router-3 at the beginning of cycle 3. In the second half of cycle 2 the H flit traverses the switch of router-2 (ST+) and is registered in $Reg+$ before it performs LT in the first half of cycle 3 (LT+). B1 flit follows half a cycle later. It performs ST and LT in the first and second half of cycle 3, respectively (ST-, LT-). The remaining flits of the packet also cannot use the FT-mode to traverse router-2 because the output port is busy transferring the earlier flits of this packet, e.g. LT of H and B1 flits in cycle 3 when B2 and B3 flits could have used FT-mode. Therefore, the remaining flits are also routed through router-2 in a similar way as H and B1 flits using AB-mode.

Following again H and B1 flits, their control information arrives in router-3 at the end of cycle 2 while the data flits are received in cycle 3. FT_Capable bit in the received FC as well as the registered result of locally pre-computed FT check inform router-3 that incoming flits can fast-track to the output. Therefore, these flits are routed using the FT-mode in a manner similar to router-1. Updated FC signals perform FT_CLT in the first half of cycle 3 and H and B1 flits undergo FT_LT- and FT_LT+ in the second half of cycle 3 and first half of cycle 4, respectively. However, flits B2 and B3 cannot be routed using the FT-mode because of lack of downstream VC credits. These flits then

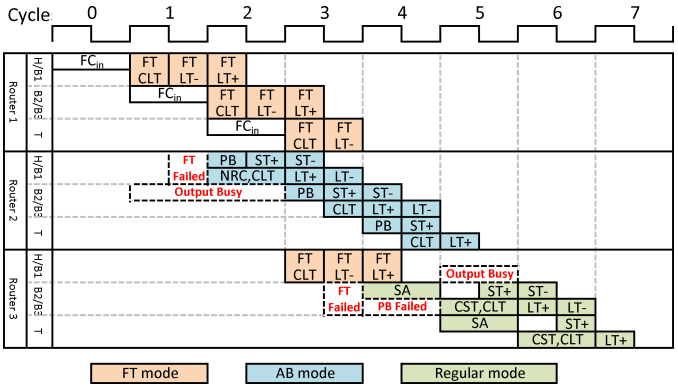


Fig. 7: Timing diagram showing the flow of a five flit packet through three routers of the FastTrackNoC.

attempt AB in cycle 4, which also fails because output port is busy during cycle 5. Allocation requests for flits B2 and B3 are then sent to SA in cycle 4. After successful SA, FC of these flits undergo switch traversal and link traversal (CST, CLT) in cycle 5. Flit B2 undergoes ST in the second half of cycle 5 (ST+) and LT in the first half of cycle 6 (LT+). Flit B3 follows half a cycle later. Finally, T flit received in the first half of cycle 5 cannot bypass any router stage because its VC still contains previously buffered flits. So, the T flit requests SA and is routed similar to B2 and B3 flits in router-3.

IV. Evaluation

The FastTrackNoC is evaluated and compared against HighwayNoC and ShortPath, the two current state-of-the-art NoC designs that achieve the highest throughput and lowest packet latency, respectively. First, we discuss the experimental setup and present post place-and-route implementation results. Then, we evaluate and compare the performance and energy of FastTrackNoC.

A. Experimental Setup

The FastTrackNoC, HighwayNoC and ShortPath are fully implemented in Register Transfer Level (RTL) abstraction, to accurately measure operating frequency, area and power consumption. All networks were also modelled in SystemC at a cycle accurate level to obtain network performance results for longer simulations. Table II lists the implementation details of all network routers.

The networks were implemented in a 28nm CMOS FDSOI (Fully Depleted Silicon on Insulator) 1.10V standard cell technology. The designs were placed and routed (P&R) with Cadence Design Systems Innovus Implementation System 17.1. We consider square tiles with their side being 2.1 mm based on the Chip multiprocessors parameters used by Sewell et al. after scaling down to 28 nm (CPU core with 32kB L1 I- and D-cache and a 512kB L2 cache slice) [31]. For the local links we consider a length of 0.5 mm. Finally, the registers in the datapath support clock gating to reduce power consumption when idle.

Power analysis is performed simulating post-P&R netlists of the NoCs in QuestaSim with back-annotated delays. Then, gate-level switching activity for each router in a network is recorded in a VCD file which was then used to get power estimates of the entire NoC using Synopsys PrimeTime PX.

Performance was measured by injecting synthetic traffic as well as application-driven traffic. Synthetic traffic injects 80 bytes (5 flits) of data packets and 16 bytes (1 flit) of control packets using the following traffic patterns: (i) uniform random (UR), (ii) hotspots (HS) with 25% of the traffic going to 4 hotspots, one at each NoC corner, and the rest of the traffic being uniform random, (iii) nearest neighbour (NN), and (iv) bit reverse (BR). In addition, traces based on application driven workloads are obtained using SynFull [32]. These traces capture the application behaviour of various PARSEC [33] and SPLASH-2 [34] benchmarks including messages generated by the cache coherence protocol and message dependencies. Simulations run until completion, all below 100 million cycles, generating packets 16 or 80 bytes for a 32 node (4×8) network. In these experiments, average packet latency is measured per benchmark.

B. Implementation results

Table III summarizes the post P&R results of a single router for all three networks. The FastTrackNoC operates at the same clock frequency as the HighwayNoC. The critical path of the FastTrackNoC is the FT_Flit_Bypass path, shown in Figure 1. The actual cycle time of FastTrackNoC is then double the above delay as two flits can traverse the router per cycle in the FT mode of traversal. Post P&R results confirm that FastTrackNoC control is not in the critical path and fits in a clock period of 654 ps. According to Psarras et al. ShortPath's control logic is in the critical path; more precisely cycle time is defined by the delays of credit-check, an N:1 arbiter (for the second part of SA), two 2:1 multiplexers and the crossbar delay [11]. Our physical implementation of the ShortPath router in 28 nm technology confirms the above indicating that the maximum operating frequency of ShorthPath is 2.56 GHz. Finally, the area of FastTrackNoC is about 4.7% higher than HighwayNoC due to the added datapath multiplexers and wires and control logic to support FT flit traversal. Compared to ShortPath, the area of FastTrackNoC is 13.2% higher because FastTrackNoC requires more complex bypassing and allocation logic as well as additional multiplexers and link control wires.

C. Performance evaluation

The performance of FastTrackNoC, HighwayNoC and ShortPath for 8×8, 16×16 and 32×32 sizes is measured using different traffic patterns and considering the maximum operating frequencies of the networks reported above.

Figure 8 shows the performance of 8×8 networks. At low injection rates, average latency of packets through FastTrackNoC is 11-13% lower than HighwayNoC for UR, HS and BR traffic patterns. This is because FastTrackNoC employs FT mode for 45-50% of hops and AB mode for another 36-39% hops, while HighwayNoC only offers AB mode which is utilized for 83-85% hops, as shown in Figure 11a for uniform traffic. For NN traffic, the average packet latency through FastTrackNoC is similar to the HighwayNoC because traffic sent to 1-hop neighbours cannot utilize the FT mode in the FastTrackNoC. Thus, in the absence of FT the performance of FastTrackNoC is reduced to that of the HighwayNoC. Similarly, compared to ShortPath, packet latency through FastTrackNoC is 9-11% lower for UR, HS and BR traffic patterns; for NN traffic, average latency is 3.2% higher.

The throughput of FastTrackNoC is similar to the HighwayNoC in all traffic patterns. This is because the clock period of FastTrackNoC is similar to the HighwayNoC. Compared to ShortPath, throughput with FastTrackNoC is 16-20% higher for all traffic pattern because the FastTrackNoC, like the previous DDR NoCs, routes flits every half a clock cycle (327 ps) as opposed to ShortPath that does so every cycle (390 ps).

TABLE II: Implementation parameters of the Networks.

Design	FastTrackNoC	HighwayNoC [20]	ShortPath [11]
Router arch.	2-stages: ST, LT (VA/SA parallel to LT of upstr. router)		4-stages: VA, SA1, SA2/ST, LT
FastTrack Support	From non-local input to non-local straight output	No	No
Pipeline Stage Bypassing	(i) Non-local input to straight output, (ii) Non-local input to local output, (iii) local input to non-local output		Dynamic pipeline-stage bypassing
Flow ctrl	Credit based flow control		
Link	128b data, 4+1b credits FC:- 1b ft_active, 1b ft_capable, 3b flit type [2], 2×2b VC-id, 2×2b NRC, 6b dest.[3] Total:- 152	128b data, 4+1b credits FC:- 3b flit type, 2×2b VC-id, 2×2b NRC, 6b dest. Total:- 150	128b data, 4b flit-credits, 1b pkt-credit, 3b flit type, 2b VC-id Total:- 138 bits
Cycles/hop	Min: 0.5, Max: 3	Min: 1, Max: 3	Min: 2, Max: 4
VC Config.	4 VCs per input port.		
Buffer size	5-flit flip-flop based VC buffers. [4]		
VC allocator	VC Select with V:2 arbiter, fixed priority	VC Select with V:2 arbiter, RR priority	V:1 in arbitr., P:1 out arbitr. VC alloc. ReqQ depth = 8
Switch allocator	Speculative in.-first separ. alloc., RR priority, V:2 in arb., P:1 in arb.	Speculative out-first separ. alloc., RR priority, PV:2 out arb., V:1 in arb.	2-stage pipel.: SA1: In-arb. V:1, SA2: Out-arb. P:1. SA ReqQ depth = 2
Routing	XY with NRC		

Performance results for 16×16 networks are presented in Figure 9. FastTrackNoC packet latency is 17-21% lower than HighwayNoC and 20-25% lower than ShortPath for traffic patterns with high average hop count (UR, HS and BR). For NN traffic, FastTrackNoC latency is similar to the HighwayNoC and 4.5% higher than ShortPath. We observe that the latency of FastTrackNoC improves relative to ShortPath with increasing network size where flits have a higher average hop count. This confirms our analysis in Section III-B, which showed that packet latency in FastTrackNoC scales better with increasing hop count compared to HighwayNoC and ShortPath.

The throughput of FastTrackNoC is similar to the HighwayNoC, whereas it is 12-19% higher than ShortPath.

Finally, performance results for 32×32 networks are presented in Figure 10. The packet latency at low injection rate in FastTrackNoC further improves compared to HighwayNoC and is 30-32% lower for traffic patterns with high average hop count because its ZLL scales better. Compared to ShortPath, FastTrackNoC packet latency is 38-40% lower for traffic patterns with high average hop count. For NN traffic, which has low hop count, latency in FastTrackNoC is similar to HighwayNoC and 4% higher than ShortPath.

The comparison in terms of network throughput is very similar to the previous network sizes.

To analyze the effectiveness of the FT-mode in FastTrackNoC, we count the flits which traverse the router in FT or AB modes in the FastTrackNoC, flits that bypass SA in the HighwayNoC and flits which propagate a hop in 2 cycles (ST + LT) or 3 cycles (VA + ST + LT) in the ShortPath. The percentages of bypassing flits for the three network is presented in Figure 11 for three network sizes and UR traffic. At low injection rates, FastTrackNoC is able to fast-track up to 46% of the flits in an 8×8 network; which is more than half of all the bypassing flits in the HighwayNoC. Furthermore, FastTrackNoC is also able to bypass another 37% of the flits, bringing the sum of fast-track and bypassing flits to the same level as the bypassing flits in HighwayNoC. For 16×16 and 32×32 networks, which have a larger average number of hops, the percentage of fast-track flits at low injection rates increases further to 61% and 80%, along with another 28% and 15% of bypassing flits, respectively in the FastTrackNoC. For these network sizes too, the sum of fast-track and bypassing flits in FastTrackNoC is comparable to bypassing flits in the HighwayNoC. Compared to bypassing flits in ShortPath, combined fast-track and bypassing flits in FastTrackNoC lag behind by 4% to 12% because it does not support bypassing of turning flits. However, FastTrackNoC still manages to reduce average packet latency for all network sizes compared to ShortPath because the hop latency in FT and AB modes is 58% and 16% lower than the minimum 2 cycles/hop ShortPath can offer when bypassing VA and SA stages.

D. Performance analysis for networks with longer hops

For FastTrackNoC, the distance traversed in a hop is defined by the half cycle FT_LT path. This path constitutes FT_Flit_Bypass and FT_LT shown in Figure 1 and it defines network half clock period of 327ps. During FT_LT, a flit travels 2.1 mm in half a clock cycle, from the input VC-0, through FT_Flit_mux, FT_output_mux and the link, to the downstream router input port.

For the HighwayNoC and ShortPath networks, the distance traversed in a single hop is composed of two parts: (i) the ST distance and (ii) the LT distance. The ST distance is from the input VC registers to the output registers and it is traversed during the ST stage. During physical implementation, we restrict HighwayNoC router placement to a region of 0.6×0.6 mm^2 and ShortPath placement to 0.5×0.5 mm^2. So, we consider 0.6mm and 0.5mm as the ST distances for HighwayNoC and ShortPath networks, respectively. The LT distance is the length of the link which is traversed during the LT stage. It is dependent on network clock frequency. For HighwayNoC and ShortPath networks, it is 2.16mm and 4.13mm, respectively. The link length of HighwayNoC is shorter than the ShortPath because it has a multiplexer before LT to transfer flits at DDR from two different registers and it has 327ps available for LT compared to 390ps for the ShortPath network [11], [21]. Thus, the total hop

TABLE III: Post place & route implementation results.

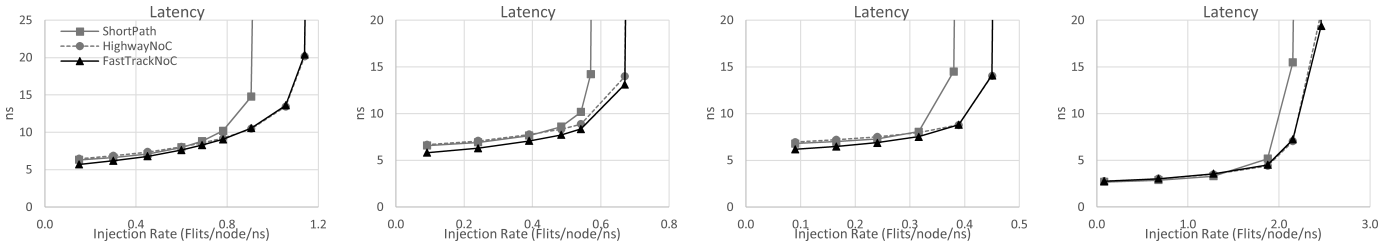
Volt.	Design	Area (#Gates)	Max Freq. (GHz)	FO4 delay
1.10 V	FastTrackNoC	178 654	1.53	21*
	HighwayNoC	170 688	1.53	21*
	ShortPath	157 810	2.56	25

* Half cycle delay is reported for DDR routers, which is the latency for a ST in HighwayNoC and ⟨FT_flit_Bypass + FT_LT⟩ in FastTrackNoC.

[2]For non-header flits, NRC bits are reused to distinguish body and tail flits.

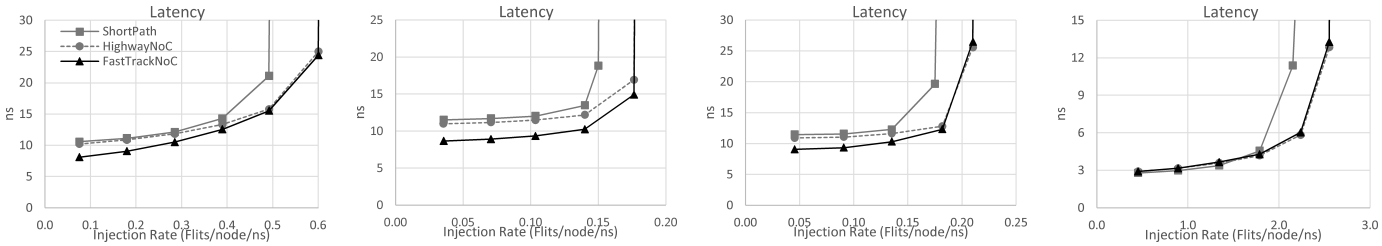
[3]Our implementation restricts only one incoming header flit to bypass per cycle per port, in order to have only one destination address field in the forwarded control. This destination address field indicates destination of a header flit to be used for NRC during AB and FT modes.

[4]This is sufficient for ShortPath as it has a credit-round-trip-time of 5 cycles and for FastTrackNoC and HighwayNoC as they handle up to 5 flits packets.



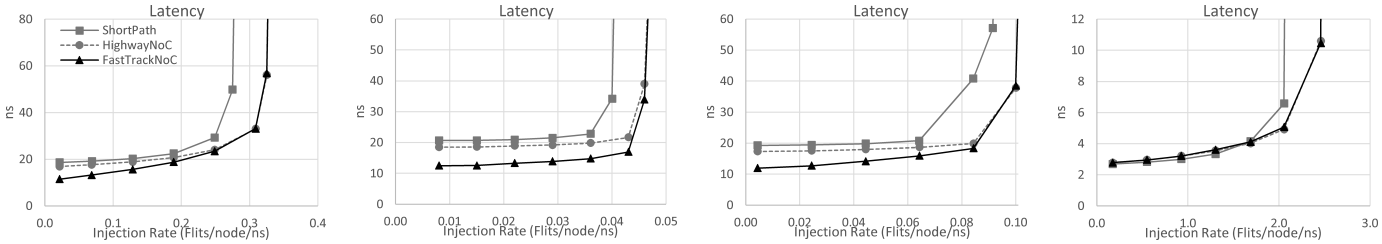
(a) Uniform random (b) Hotspot (c) Bit reverse (d) Nearest neighbour

Fig. 8: Average packet latency of 8×8 2D mesh networks for 4 different traffic patterns.



(a) Uniform random (b) Hotspot (c) Bit reverse (d) Nearest neighbour

Fig. 9: Average packet latency of 16×16 2D mesh networks for 4 different traffic patterns.



(a) Uniform random (b) Hotspot (c) Bit reverse (d) Nearest neighbour

Fig. 10: Average packet latency of 32×32 2D mesh networks for 4 different traffic patterns.

distance traversed by HighwayNoC and ShortPath routers is 2.76mm and 4.63mm respectively.

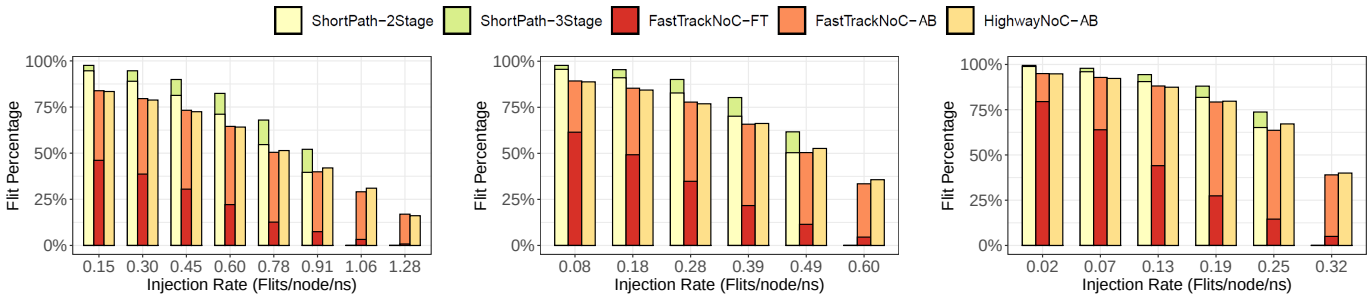
In order to transfer flits over a longer hop, links need to be pipelined to avoid slowing down network clock frequencies and to maintain their throughput. The length of these pipelined links differs for DDR and SDR networks, because DDR links require two registers and a multiplexer per pipeline stage to transfer flits at DDR. Moreover, the length of the link is constrained by half clock period for DDR NoCs and full clock period for the SDR network. Physical implementation of these pipelined links shows that the length of DDR and SDR links, with a delay equal to critical paths in our analyzed networks, i.e. 327ps and 390ps, is 2.89mm and 4.13mm, respectively.

Now, we analyze the effect of increasing hop distance on minimum hop latency for FastTrackNoC, HighwayNoC and ShortPath networks, as presented in Figure 12. This analysis takes into account the maximum clock frequency of the analyzed networks and hop distance they can traverse without and with pipelined links. For comparison, we also plot the ideal latency of transmitting a flit over an un-pipelined link which covers the required hop distance and has registered end points. For hop distances less than 2.1mm, FastTrackNoC offers minimum hop latency of 327ps because it can forwards flits in half a clock cycle. FastTrackNoC hop latency is 50% lower than the HighwayNoC which requires 1 cycle per hop, and 58% lower than ShortPath which requires two 2 cycles per hop. For hops longer than 2.1mm but shorter than 4.99mm, FastTrackNoC requires links with one DDR pipeline stage and has a hop latency of 654ps. Now, the latency of FastTrackNoC is similar to HighwayNoC until 2.76mm, after which HighwayNoC also requires a DDR pipeline stage for its links and has a latency of 981ps. This makes FastTrackNoC hop latency 33% lower than HighwayNoC for hops longer than 2.76mm. Compared to ShortPath network, for hops longer than 2.1mm, the latency of FastTrackNoC is 16% lower until 4.63mm, at which point ShortPath network requires its first SDR pipeline stage for its links and the latency of FastTrackNoC becomes 44% lower. Furthermore, compared to HighwayNoC and ShortPath networks, the hop latency offered by FastTrackNoC is closer to the ideal latency.

E. Energy efficiency

Figure 13 summarizes the energy efficiency results on an 8×8 mesh with UR traffic for the FastTrackNoC, HighwayNoC and ShortPath. The following are measured: total power, energy per transferred bit, energy-delay product (EDP), and energy throughput ratio (ETR).

Compared to the HighwayNoC, FastTrackNoC power consumption is 5% higher when idle, 3.7% higher at low injection rates and 1.6% higher at high injection rates, a consequence of wide flit bypass paths, additional control logic and link wires added to support FT-mode of flit traversal. This is also reflected in energy per bit and ETR, both of which are also increased by 1.6% to 3.7% for the FastTrackNoC. Finally, EDP of the FastTrackNoC is up to 8% lower at low injection rates, in spite of the higher power consumption, because of significant latency reduction achieved by FT traversals.



(a) 8×8 (b) 16×16 (c) 32×32

Fig. 11: Percentage of bypassing flits in the FastTrackNoC, HighwayNoC and ShortPath networks for UR traffic. Bars are not shown for ShortPath network after it saturates.

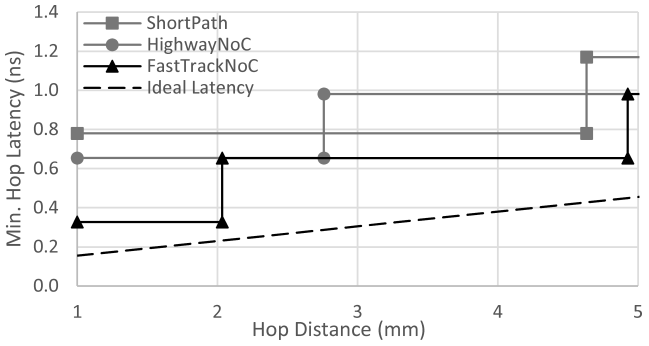


Fig. 12: Minimum hop latency with respect to hop distance for FastTrackNoC, HighwayNoC and ShortPath networks as well as for an ideal un-pipelined link with registered end-points.

Close to network throughput, where FT-mode is active for less than 3% of the hops, as shown in Figure 11a, EDP is increased by 3%.

FastTrackNoC has up to 79% higher power than ShortPath at low injection rates because it uses more complex control logic and lookahead signalling. However, at the saturation throughput of ShortPath, FastTrackNoC power is only 5.2% higher as the two networks converge to the same percentage of bypassing cases. The energy per bit differences between the networks are more profound at lower injection rates; that is because the energy costs include the idle energy of the networks, which in these points is shared among fewer transferred bits. Consequently, compared to ShortPath, FastTrackNoC has up to 36% higher energy per transferred bit at low injection rates, but the difference is negligible at high injection rates.

At low injection rates, FastTrackNoC has up to 20% higher EDP and 30% higher ETR compared to ShortPath. At higher injection rates, FastTrackNoC is better than ShortPath due to its better performance and achieves 40% lower EDP and 16.5% lower ETR.

F. Evaluation using application-driven traffic

Using SynFull, we measure the average packet latency of 32-node (4×8) networks for FastTrackNoC, HighwayNoC and ShortPath using various PARSEC and SPLASH-2 benchmarks. In order to stress networks throughput, in these experiments we considered 32-bit network datapaths, rather than 128-bits and SynFull packet generator step of 200ps (400ps for *fft* and *radix* benchmarks which saturate all networks). Average throughput does not provide any useful insight and therefore is not reported since for all networks the traffic generated by a benchmark is entirely delivered. Network performance in terms of throughput is reflected in packet latency, too.

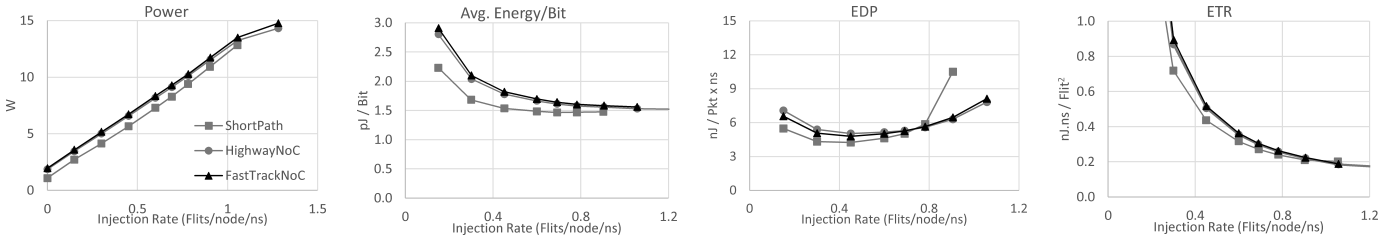
Figures 14 reports the average packet latency per benchmark for each of the three networks as well as the geometric mean. FastTrackNoC offers the lowest latency for all benchmarks. On average, FastTrackNoC reduces packet latency by 26% compared to ShortPath due to its significant throughput advantage and competitive router latency. Compared to HighwayNoC, FastTrackNoC reduces packet latency by 3.7% due to its extended bypassing support. On average, the amount of bypassing traffic in FastTrackNoC is similar to the HighwayNoC, out of which 38% traffic bypasses both SA and ST stages. FastTrackNoC offers about $\frac{3}{4}$ of the bypassing opportunities compared to ShortPath as it does not support bypassing for turning flits.

V. Conclusion

This paper proposes the FastTrackNoC network architecture which enables incoming flits to bypass ST and SA pipeline stages and directly initiate LT, in order to reduce packet latency. This is in addition to bypassing only the SA stage when flits propagate straight or enter and exit the network. FastTrackNoC implements bypassing on a NoC with DDR datapaths because of their higher throughput compared to SDR networks. Incoming flits use bypass paths in a router to bypass router crossbar when router in FT mode and traverse a hop in half a clock cycle. To reduce complexity of bypass paths and satisfy tight timing constraints, the FastTrack mode is restricted to flits in a specific input VC propagating an in-network straight hop and requires pre-computation of FT checks. As a result, FastTrackNoC operates at the same frequency as previous DDR NoCs, offers similar throughput and reduces latency by up to 32% at low loads. Moreover, it substantially outperforms existing SDR NoCs as it achieves up to 20% higher throughput and reduces packet latency by up to 40% at low loads.

References

[1] S. Borkar, "How to stop interconnects from hindering the future of computing!" in *2013 Optical Interconnects Conf.*, May 2013, pp. 96–97.
[2] B. K. Daya *et al.*, "Scorpio: A 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *Int. Symp. on Computer Architecuture*, ser. ISCA, 2014, pp. 25–36.
[3] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *MICRO-43*, 2010, pp. 421–432.
[4] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "Noc-out: Microarchitecting a scale-out processor," in *IEEE/ACM MICRO-45*, Dec 2012, pp. 177–187.
[5] A. Psathakis *et al.*, "A systematic evaluation of emerging mesh-like cmp nocs," in *ANCS*, 2015, pp. 159–170.
[6] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *HPCA*, Feb 2009, pp. 163–174.
[7] J. Kim *et al.*, "A low latency router supporting adaptivity for on-chip interconnects," in *Design Automation Conf. (DAC)*, 2005, pp. 559–564.



(a) Total Power (b) Energy per bit (c) Energy-Delay product (d) En.-Throughput ratio

Fig. 13: Energy and power costs on a 8×8 network with UR traffic.

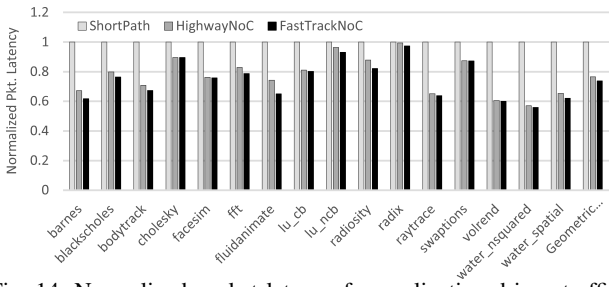


Fig. 14: Normalized packet latency for application driven traffic.

[8] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

[9] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Int. Symp. on HPCA*, 2001.

[10] T. Krishna *et al.*, "Swift: A swing-reduced interconnect for a token-based network-on-chip in 90nm cmos," in *IEEE ICCD*, 2010, pp. 439–446.

[11] A. Psarras *et al.*, "ShortPath: A Network-on-Chip Router with Fine-Grained Pipeline Bypassing," *IEEE ToC, 65, 10*, pp. 3136–3147, 2016.

[12] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *MICRO-40*, 2007, pp. 172–182.

[13] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Int. Conf. on Supercomputing*, 2006, pp. 187–198.

[14] A. Psarras *et al.*, "Networks-on-chip with double-data-rate links," *IEEE Trans. on Circuits and Systems*, vol. 64, no. 12, pp. 3103–3114, 2017.

[15] S. Rao *et al.*, "Vix: Virtual input crossbar for efficient switch allocation," in *Design Automation Conf. (DAC)*, 2014, pp. 103:1–103:6.

[16] H. Kim *et al.*, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *MICRO-46*, 2013, pp. 136–147.

[17] A. K. Mishra *et al.*, "A case for dynamic frequency tuning in on-chip networks," in *MICRO-42*, 2009, pp. 292–303.

[18] C. Nicopoulos, V. Narayanan, and C. R. Das, *Network-on-Chip Architectures - A Holistic Design Exploration*, ser. Lecture Notes in Elect. Eng. Springer, 2010, vol. 45.

[19] A. Ejaz, V. Papaefstathiou, and I. Sourdis, "DDRNoC: Dual data-rate network-on-chip," *ACM Trans. Archit. Code Optim.*, vol. 15, no. 2, 2018.

[20] ——, "Freewaynoc: A ddr noc with pipeline bypassing," in *Int'l Symp. on Networks-on-Chip (NOCS)*, 2018.

[21] ——, "Highwaynoc: Approaching ideal noc performance with dual data rate routers," *IEEE/ACM Transactions on Networking*, 2020.

[22] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *IEEE Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2018, pp. 492–503.

[23] Y. Hoskote *et al.*, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[24] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, 2007.

[25] C. Chen *et al.*, "Physical vs. virtual express topologies with low-swing links for future many-core nocs," in *NOCS*, 2010, pp. 173–180.

[26] G. Michelogiannakis *et al.*, "Approaching ideal noc latency with pre-configured routes," in *Int'l Symp. on NOCS*, May 2007, pp. 153–162.

[27] M. Hayenga and M. Lipasti, "The NoX router," in *MICRO-44*, 2011.

[28] Y. Lu, C. Chen, J. McCanny, and S. Sezer, "Design of interlock-free combined allocators for networks-on-chip," in *2012 IEEE International SOC Conference*, 2012.

[29] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34–39, Jan. 1997.

[30] M. Azimi *et al.*, "Flexible and adaptive on-chip interconnect for tera-scale architectures," *Intel Technology J.*, vol. 13, no. 4, pp. 62–77, 2009.

[31] K. Sewell *et al.*, "Swizzle-switch networks for many-core systems." *IEEE J. Emerg. Sel. Topics Circ. Syst.*, vol. 2, no. 2, pp. 278–294, 2012.

[32] M. Badr and N. E. Jerger, "Synfull: Synthetic traffic models capturing cache coherent behaviour," in *ACM/IEEE ISCA*, June 2014, pp. 109–120.

[33] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Dept. Comput. Sci., Princeton, NJ, USA, 2011.

[34] S. C. Woo *et al.*, "The splash-2 programs: Characterization and methodological considerations," *SIGARCH CA News, 23(2)*, pp. 24–36, 1995.