# Contention Minimization in Emerging SMART NoC via Direct and Indirect Routes

Peng Chen, Hui Chen, Jun Zhou, Mengquan Li, Weichen Liu, *Member, IEEE,* Chunhua Xiao, *Member, IEEE,* Yiyuan Xie, *Member, IEEE,* and Nan Guan, *Member, IEEE*

**Abstract**—SMART (Single-cycle Multi-hop Asynchronous Repeated Traversal) Network-on-Chip (NoC), a recently proposed dynamically reconfigurable NoC, enables single-cycle long-distance communication by building single-bypass paths directly between distant communication pairs. However, such a single-cycle single-bypass path will be readily broken when contention occurs. Thus, packets will be buffered at intermediate routers with blocking latency from other contending packets, and extra router-stage latency to rebuild the remaining path when available, reducing the bypassing benefits that SMART NoC offers. In this article, we *for the first time* propose an effective contention-minimized routing algorithm to achieve maximal bypassing in SMART NoCs. Specifically, we identify two potential routes for packets: *direct route*, with which packets can reach the destination in a single bypass; and *indirect route*, with which packets can reach the destination in multiple bypasses via a (multiple) intermediate router(s). The novel feature of the proposed routing strategy is that, contrary to an intuitive approach, not the routes with minimal distance but the indirect routes via the arbitrary intermediate routers (even if they may be non-minimal) that avoid contentions yield the minimized end-to-end latency. Our new routing strategy can greatly enrich the path diversity, effectively minimize the conflicts between communication pairs, greatly balance the workloads and fully utilize bypass paths. Evaluation on realistic benchmarks demonstrates the effectiveness of the proposed routing strategy, which achieves average performance improvement by $35.48$ percent in communication latency, $28.31$ percent in application schedule length, and $37.59$ percent in network throughput, compared with the current routing in SMART NoCs.

**Index Terms**—SMART NoC, contention-minimized routing, bypassing, end-to-end latency, direct route, indirect route.

## 1 Introduction

NETWORK-on-chip (NoC) is a widely-used communication backbone for on-chip multi-/many-core systems (e.g., Xeon Phi [1], Teraflop [2], and Tilera [3]), in which the communication latency gradually becomes a bottleneck for system performance [4], [5] due to long-distance data transmission, especially for communication-intensive applications. Two main factors are influencing the latency of a source-destination communication pair: (i) the number of hops between the source and the destination, and (ii) contention issues that would introduce blocking latency from other contending packets. The former factor is inherently restricted by the distance of the source-destination communication pair, and the latter one is mainly related to task mapping and packet routing approaches.

To address the first factor influencing the communication latency mentioned above, an advanced reconfigurable NoC using the Single-cycle Multi-hop Asynchronous Repeated Traversal (SMART) NoC (SMART for short henceforth) was proposed [5], [6], which can effectively reduce the number of hops by dynamically building a single-cycle multi-hop path via bypass, and works especially well for long-distance communication pairs. To reduce the effective number of hops, SMART employs bypass control and integrated clockless repeaters based on traditional on-chip routers. It mainly takes advantage of the following two facts: (i) With asynchronous repeaters, the electric signal can propagate multi-millimeters in a single cycle; (ii) Packets can bypass the intermediate routers where no contention occurs. With the help of SMART techniques, the communication topology of the communication pairs can be dynamically reconfigured at runtime, and packets can fully bypass all the intermediate routers from their sources to destinations in the best case. Inspired by this advantage, the techniques of SMART NoCs are widely adopted in many recent academic works [7]–[11].

Nevertheless, when packets encounter contention at intermediate routers, SMART only builds the single-cycle multi-hop path from the source to the first conflicting intermediate router. If the contention issue is not well addressed, the bypass path from the source to the destination is readily broken, thus reducing the bypassing benefits that SMART offers. In particular, when packets frequently encounter contention at intermediate routers (in the extreme case, at every intermediate router), SMART performs the same as traditional hop-by-hop traversal NoCs. No more benefits can be acquired than the traditional counterpart in such cases, but incurring higher control overhead to establish the path (e.g., control link). Therefore, to fully and effectively utilize the dynamic reconfigurability of SMART, multi-hop bypassing must be utilized in a way such that contentions


- *P. Chen and M. Li are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, and also with the College of Computer Science, Chongqing University, Chongqing, China. E-mail: chnp616@gmail.com, mengquan@cqu.edu.cn.*
- *H. Chen, J. Zhou, and W. Liu are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: {hui.chen, edo.zhou, liu}@ntu.edu.sg.*
- *C. Xiao is with the College of Computer Science, Chongqing University, Chongqing, China. E-mail: xiaochunhua@cqu.edu.cn.*
- *Y. Xie is with the College of Electronic and Information Engineering, Southwest University, Chongqing, China. E-mail: yyxie@swu.edu.cn.*
- *N. Guan is with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China. E-mail: nanguan@cityu.edu.hk.*

are minimized as much as possible. This dynamic reconfigurability of SMART paves the way for performance scalability of future kilo-core (1000 cores) chips [12].

In this article, to our best knowledge, we *for the first time* address the contention reduction problem for bypassing in SMART from the perspective of the routing strategy. In the current XY routing for SMART, in which packets are susceptible to contention at intermediate routers, packets will suffer blocking latency from other contending packets, and extra router-stage latency to rebuild the remaining path. To reduce the contentions, the contributions are as follows:

- Firstly, we identified two potentially source-destination transmission routes for packets: *direct route* that is a *single*-bypass path, with which packets can reach the destination directly within a cycle, and *indirect route* that is a *multiple*-bypass path, with which packets can reach the destination indirectly via the intermediate routers. Compared with the current routing approach in SMART NoC, direct and indirect routes greatly increase the path diversity.
- Then, we proposed a design-time routing algorithm to assign an appropriate route for each packet such that the communication workloads can be well spatially isolated and balanced. Specifically, when conducting route allocation, we firstly try to exploit a contention-free *direct route* from the source to the destination according to the current resource state. If a direct route is not found, we instead turn to exploit an *indirect route* to eliminate blocking latency from other contending packets, at the expense of small router-stage latency at the intermediate routers.
- Finally, together with the consideration of real-time network state, we presented an improved hybrid routing algorithm that combines the advantages of design-time and runtime strategies, which can achieve higher communication performance due to finely-grained isolation of the communication workloads in both temporal and spatial dimensions, with the support of the resource manager [13], [14].

For the design-time and hybrid routing strategies in SMART NoCs, which one will be adopted depends on the traffic density, packet size, and performance requirements. In particular, to mitigate traffic congestion, the indirect routes allow the packets to be traversed with a non-minimal path. Contrary to the existing intuitive approaches in traditional NoCs with hop-by-hop traversal, not the routes with minimal distance but the indirect routes via an arbitrary intermediate router (even if they may be non-minimal) that avoid contentions yield the minimized communication latency. Our routing algorithms can well set up the dynamic reconfigurability of communication topology which in turn enables high system performance. Experimental results on realistic benchmarks show significant communication performance improvement with the help of our routing strategies, compared with the current routing in SMART NoCs. To the best knowledge, this is the first work on communication optimization from the perspective of routing in the emerging well-developed SMART NoCs [5], [9], [10].

The remainder of this article is organized as follows. Sec. 2 introduces the background, and motivation example. Sec. 3 provides the communication backbone, communication demand and problem definition. In Sec. 4, the routing strategy aiming to achieve contention reduced bypassing is proposed; and an improved hybrid routing strategy is proposed in Sec. 5. While the performance evaluation is conducted in Sec. 6. Sec. 7 finally concludes this article.

## 2 Background and Motivation

### 2.1 End-to-end Latency in Traditional NoCs

In traditional NoCs with hop-by-hop traversal [4], packets traverse from the source to the destination hop by hop through on-chip routers and links. To forward packets, a router conducts the following stages [4]: Route Computation (RC), Virtual-channel Allocation (VA), Switch Allocation (SA), Switch Traversal (ST), and Link Traversal (LT). Each packet is forwarded in a pipelined manner. The RC and VA stages are only conducted by head flit, while in the absence of stalls, the remaining flits (i.e., body and tail flit) enter the pipeline one cycle behind the head flit without computation.

In general, the packet latency covers the transmission latency of the head flit from the source to the destination, the serialization latency of the rest of the flits, and the blocking latency suffered from other contending packets. The complete route of the packet transmission composes of two *PE*(processing element)*-to-router* links and multiple *router-to-router* links. For the simplicity of analysis, we can only consider the *router-to-router* path since the transmission time in *PE-to-router* and *router-to-PE* links is constant and can be easily added. Therefore, the end-to-end latency $\mathcal{L}_{e2e-T}$ of a transmitted packet from the source router $R_s$ to the destination router $R_d$ in traditional NoCs is:

$$\mathcal{L}_{e2e-T} = (t_r + t_w) \cdot h_{sd} + t_w \cdot (\mathcal{N}_i - 1) + \mathcal{L}_b \qquad (1)$$

where $t_r$ is the router-stage latency to set up the path to the next stopping router (i.e., RC, VA); $t_w$ is the link latency between two adjacent stopping routers; $h_{sd}$ refers to the number of hops between the source router $R_s$ and the destination router $R_d$; $\mathcal{N}_i$ refers to the number of transmitting flits of a packet; and $\mathcal{L}_b$ refers to the summation of blocking latency suffered from other contending packets along the path to destination; NoC link width is one flit. For a given NoC platform and an analyzed packet, the parameters $t_r$ (also there are some works [15] to reduce such parameter.), $t_w$, and $\mathcal{N}_i$ are constants. From Eq. (1), the end-to-end latency is mainly determined by $h_{sd}$ and $\mathcal{L}_b$.

As a result, there are mainly two ways to reduce the end-to-end latency $\mathcal{L}_{e2e-T}$ in traditional NoCs: (i) reducing the hop count $h_{sd}$; (ii) reducing the blocking latency $\mathcal{L}_b$ with contention avoidance. For the first approach, contemporary researches (e.g., [16]) of task mapping and packet routing almost focuses on distance minimization, such as locality-centric task mapping and minimized-distance packet routing. To fundamentally improve performance, some architecture-based works, such as high-radix technique [17], application-specific long-range link [18], skip-links [19] and single-cycle multi-hop traversal NoC [5], are proposed to effectively reduce the hop count. As for the second one, the contention-aware task mapping approaches (e.g., [7], [20]) are proposed to reduce the number of contentions, such that the suffered blocking latency from other contending packets

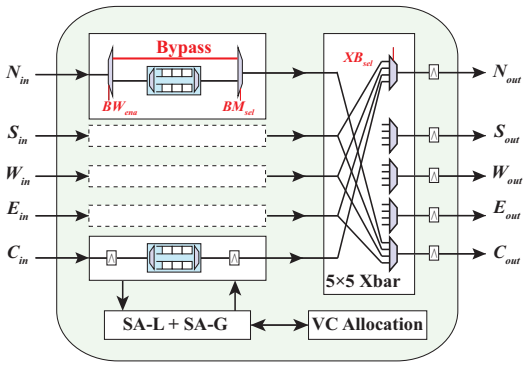



Fig. 1: SMART router micro-architecture with bypass path.

is reduced. In this article, we combine these two methods to co-optimize the NoC performance.

### 2.2 End-to-end Latency in SMART NoCs

Among the NoC architectures aiming to reduce hop count $h_{sd}$ between communicating source and destination, SMART [5] (single-cycle multi-hop asynchronous repeated traversal) enables single-cycle long-distance communication by dynamically building a single-bypass direct path from the source to the destination, which in turn effectively reduces the end-to-end latency. Unlike the above-mentioned ones [17]–[19], SMART NoCs have advantages on area, power, and layout complexity.

SMART is proposed by embedding the low-swing clockless repeated link and bypass control in traditional routers. It mainly is based on the following two facts: (i) With the embedded repeated link, the electric signal can be transmitted multi-millimeters in a single cycle; (ii) The incoming data can bypass the intermediate routers where no contention occurs. To establish the bypass path, SMART adds a set of dedicated SMART-hop Setup Request (SSR) links spanning up to $HPC_{max}$ neighbors [5] for each possible dimension-order routing (DOR) [4] path, which are used to forward bypass requests to downstream intermediate routers. Distinct from the conventional router, the Switch Allocation (SA) in SMART is further split into two stages: Local Switch Allocation (SA-L) and Global Switch Allocation (SA-G). SA-L, which is the same as the SA stage of the conventional router, can be achieved with different arbiters [4] (e.g., priority, round robin) according to the desired user requirements. While SA-G is implemented with the arbiters based on hops from source router and turns (more details are in Fig. 11 of the original SMART [5]). The working process of SMART consists of three steps: *Step 1:* Each start router performs SA-L to choose a winner from buffered flits for each output port; *Step 2:* SA-L winner broadcasts SSR request, which carries the desired path length in hops over SSR links, to downstream routers towards the destination to set up bypass path, and continually each SSR recipient conducts SA-G for competing SSRs. The bypass path is built at the end of this step by setting the control signals (i.e., $BW_{ena}$, $BM_{sel}$, $XB_{sel}$), as shown in Fig. 1; *Step 3:* The packet of the SA-G winner traverses the established single-bypass path with multiple hops up to $HPC_{max}$ ($HPC_{max}$ refers to the maximum number of hops that can be reached per cycle).

Distinct from traditional NoCs, the hop count $h_{sd}$ is eliminated if the maximum bypass hop count $HPC_{max}$ [5] is greater than or equal to $2 \times (\mathcal{N} - 1)$ for the allocated region with $\mathcal{N} \times \mathcal{N}$ in 2D-Mesh SMART NoCs. Correspondingly, the end-to-end latency $\mathcal{L}_{e2e-S}$ over SMART NoCs is:

$$\mathcal{L}_{e2e-S} = (t_r + t_w) \cdot (\mathcal{N}_c + 1) + t_w \cdot (\mathcal{N}_i - 1) + \mathcal{L}_b \quad (2)$$

where $\mathcal{N}_c$ refers to the contention count along the path of the source-destination communication pair; note that if $\mathcal{N}_c = 0$, the blocking latency summation $\mathcal{L}_b = 0$; in such case, the packet completely bypasses all the intermediate routers from the source to the destination. It can be seen from the Eq. (2), the end-to-end latency $\mathcal{L}_{e2e-S}$ in SMART is only determined by the contention count $\mathcal{N}_c$ if $HPC_{max}$ is large enough. Therefore, according to Eq. (2), the performance is mainly dependent on the contention parameter $\mathcal{N}_c$.

Despite the benefits brought by SMART NoCs, due to the large control overhead and throughput loss in some cases, some improved works (e.g., [7], [9], [10], [21]–[23]) are proposed to upgrade the performance of SMART. In addition, in light of this communication advantage of SMART techniques, many studies (e.g., [8], [24], [25]) employ SMART design as the communication fabric of multi/many-core systems. However, there are few studies to optimize the SMART communication performance with the consideration of contention reduction (i.e., $\mathcal{N}_c$).

### 2.3 Motivation Example

As mentioned in previous sections, SMART NoCs show great communication advantages by building a single-cycle multi-hop bypass path; on top of that, packets can completely/partially bypass intermediate routers up to $HPC_{max}$ hops towards the destination, provided that a bypass path is built. However, the current XY routing in SMART NoCs cannot be applicable for communication-intensive applications since it cannot effectively reduce the number of contentions even under lightweight traffics (e.g., collective traffic). We motivate the need for a flexible routing by presenting an example with a graph shown in Fig. 2(a), to demonstrate the benefits that our proposed flexible routing offers.

As shown in Fig. 2(a), the communication requirement is represented as a communication graph, where the number between source and destination refers to the packet size in flits. There are 4 pairs transmitting in 2D-Mesh SMART ($HPC_{max} = 6$, a flit can maximally bypass 6 hops per cycle), $\langle s_1, d \rangle$, $\langle s_2, d \rangle$, $\langle s_3, d \rangle$, and $\langle s_4, d \rangle$. The source and destination tasks are mapped like in Fig. 2(b) and Fig. 2(d), where the thick black links are occupied by other packets. Here, we only consider the transmission time of router-to-router path, such as the data path $R_{15} \rightarrow R_{14} \rightarrow R_{10}$ of $\langle s_4, d \rangle$ in Fig. 2(b). Assume all of the packets are released *simultaneously* starting from the Time 0, the non-preemptive *distance* and *direction* based local policy* [5] is employed when contention occurs at intermediate routers, and the NoC link width is equal to one flit size. The transmission timelines under the XY routing [4] (firstly route in the dimension X, then route

*. Under the *local* priority policy, the nearest SSR to the current router wins the arbitration; and direction-based arbitration (i.e., *straight* > *left-turn* > *right-turn*) is adopted for equal-distance SSRs.

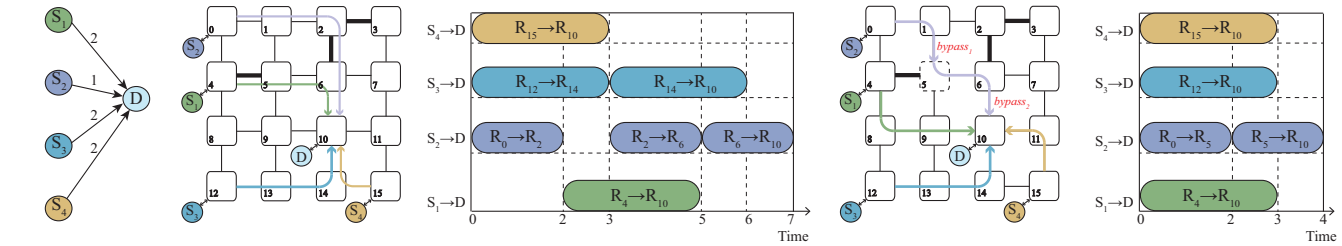



Fig. 2: A running example when using XY routing and our proposed flexible routing in SMART NoCs ($HPC_{max}$ = 6).

in the dimension Y) and our proposed flexible routing are shown in Fig. 2(c) and Fig. 2(e), respectively.

We observe that from the running examples for the same communication pairs, the communication performance of our proposed flexible routing is remarkably better than that of the XY routing. As shown in Fig. 2(c), when using XY routing, the schedule length of the given packets is 7 time units and the total communication latency is 18 time units; while for our proposed flexible routing, the schedule length is 4 time units, and the total communication latency is 13 time units. More specifically, for $\langle s_1, d\rangle$, its packet is blocked by other packets for 2 time units when using XY routing due to contention at link $R_4 \rightarrow R_5$, while the packet of $\langle s_1, d\rangle$ reaches the destination $R_{10}$ directly by alternatively choosing the YX routing, thereby eliminating the contention and bypassing the intermediate routers $R_8$ and $R_9$ in Fig. 2(d). Thus, the transmission completion of $\langle s_1, d\rangle$ in Fig. 2(e) is finished earlier than that of XY routing of Fig. 2(c) due to contention avoidance. For $\langle s_2, d\rangle$ under XY routing, it encounters two contentions and buffered at routers $R_2$ and $R_6$. In this case, it suffers not only the blocking latency from other contending packets but also the router-stage latency to rebuild the remaining path at the conflicting routers (i.e., $R_2$, $R_6$). Instead, since the links $R_4 \rightarrow R_5$ and $R_2 \rightarrow R_6$ have been occupied, our approach resorts to a contention-free double-bypass data path via the intermediate router $R_5$, and then the 2 router-stage latencies and blocking latency of original conflicting route are eliminated, at the cost of only one router-stage latency at $R_5$ to rebuild the second half of the path. As a result, the latency of $\langle s_2, d\rangle$ is reduced from 7 to 4 time units. Similarly, the potential contention between $\langle s_3, d\rangle$, and $\langle s_4, d\rangle$ is avoided through route isolation.

Through the motivation example in Fig. 2, the observations are twofold: (i) With another contention-free route via intermediate routers, the potential contention can be eliminated, which in turn eliminates the router-stage and blocking latency at originally conflicting routers, at the cost of only one router-stage latency at the intermediate router. (ii) To further reduce the contention, the multiple-bypass route can be non-minimal in distance, breaking the minimized-distance routing tradition. That is, any router (e.g., $R_{12}$ for $\langle s_2, d\rangle$) can be selected as the intermediate router, without latency increasing as long as each bypass route is within $HPC_{max}$. Also, there are some works [26]–[31] involved routing in traditional NoCs or for other purposes (e.g., multi-cast) in SMART NoCs. Inspired by these works, on top of SMART NoCs [5], we thus propose a routing strategy in this article to achieve contention reduced bypassing transmission, fully utilizing the bypass path and reducing the end-to-end latency.

## 3 Problem Definition

### 3.1 Communication Backbone

To guarantee low latency of communication-intensive applications, we employ the 2D-Mesh SMART NoC as the on-chip communication backbone for multi/many-core systems. For the given communication demand at design time, a portion of SMART is allocated, denoted by $\pi$, on which some resources (e.g., link) may be unavailable due to assignment to other communication pairs that are out of the current communication demand scope. Formally, $\pi$ consists of $\mathcal{N} \times \mathcal{N}$ array of tiles, where each tile includes a processing element (PE), network interface (NI), and on-chip router. The PEs can be homogeneous or heterogeneous (e.g., CPU, accelerator, memory). As for the maximal bypass hop count $HPC_{max}$, it is fixed as long as SMART is configured at design time (i.e., set as 9 for paths with turns within a 1GHz cycle [5]). The packet is divided into multiple fixed-size flits, and the virtual cut-through flow control is adopted. Note that, to avoid large control overhead, the routing path in SMART NoCs is only allowed for at most one turn. Thus, the dimension-order route (DOR) [4] is adopted for each bypass phase, i.e., XY, YX. For example, a packet with the XY route is forwarded first in the X dimension until it reaches the same column as the destination, then forwarded in the Y dimension to the destination.

### 3.2 Communication Demand Model

The communication demand of one application is formally represented by $\mathcal{P}$ consisting of a set of source-destination communication pairs, $\mathcal{P} = \{p_1, p_2, ..., p_{\mathcal{M}}\}$. Each source-destination communication pair (*pair* for short henceforth) $p_i = \langle s_i, d_i\rangle \in \mathcal{P}$ is associated with the generated packets from the source router $s_i = (x_i^s, y_i^s)$ to the destination router $d_i = (x_i^d, y_i^d)$. Assume the communication demand $\mathcal{P}$ is not enforced to set the deadline constraints. Given any two pairs $p_i$ and $p_j$ ($p_i, p_j \in \mathcal{P}$), we say $p_j$ is *dependent* on $p_i$ if the packet generated by $p_j$ is strictly released after the transmission completion of the packet from $p_i$ due to data or control dependence. Such the *dependence* relationship is denoted by $p_i \prec p_j$. Since the contention situation only occurs when two pairs have both temporal and spatial overlap at runtime, the dependent pairs must be in the absence of contentions at runtime even if they overlap in the spatial dimension (e.g., share links). To fully utilize the

multi-hop bypass advantage of SMART NoCs, assume the Manhattan distance of the pairs is within $HPC_{max}$ hops, also discussed by Liu et al. [32]. The reasons are twofold. First, the control signal SSR is only broadcasted to the downstream routers ranging from 1 to $HPC_{max}$ hops. That is, even if a long-distance route that is more than $HPC_{max}$ is assigned, the transmitting packet will experience at least one stop at intermediate routers. Second, the long-distance route will lead to more potential contentions to other pairs since the communication resources are limited.

### 3.3 Problem Illustration

The packet routing problem to be solved in this article is defined as follows: We are given a SMART NoC $\pi$ with allocated array size of $\mathcal{N} \times \mathcal{N}$. For a given set of source-destination communication pairs $\mathcal{P}$ consisting of $\mathcal{M}$ pairs, the objective is to determine a routing path for each pair to effectively isolate and balance the potentially conflicting pairs in order that the single-cycle multi-hop bypass path is built as possible. With the consideration of the contention-aware routing strategy, the dynamic reconfigurability of communication topology can be well controlled to enable a single-cycle long-distance transmission in SMART NoCs.

## 4 Contention-Minimized Routing

As stated in previous sections, the single-cycle multi-hop bypass establishment of SMART is very susceptible to contentions. To our best knowledge, in SMART NoCs, XY routing is the currently available routing [5], since no related routings are studied and the routing approaches in traditional NoCs are not fully applicable. Since XY routing cannot effectively isolate conflicting pairs, the reconfigurability of communication topology is largely limited, thus degrading SMART performance. It is a natural requirement to optimize the reconfigurability from the perspective of the packet routing design. Note that, the application performance and reconfigurability capability are jointly determined by task mapping, scheduling, and routing strategies. The study of task mapping and scheduling is temporarily not considered in this work, and we mainly focus on the routing design.

### 4.1 Routing Overview

For a given communication demand $\mathcal{P}$ on the communication infrastructure $\pi$, multiple communication pairs are remaining to be assigned routes. In consideration of the relative order of these communication pairs to be assigned and the way for route selection from multiple route candidates, the route allocation is split into two steps, as shown in Alg. 1. Before the application begins execution (namely at design time), for its unallocated pairs, we firstly select a pair that has the least number of possible idle routes using the function `selectPair` in Line 2. Then, for all the idle route candidates of the selected pair, the route that has minimal resource usage and minimum impact on potential routes of unassigned pairs is chosen by the function `assignRoute` in Line 3. Finally, the route allocation state is updated, and the `while` loop is repeated if $\mathcal{P} \neq \phi$.

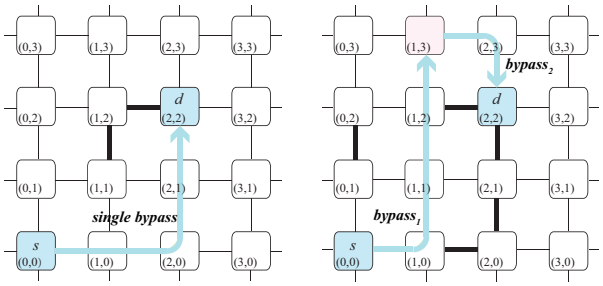
**Algorithm 1:** Outline of Route Allocation

**Input**: A set of unallocated source-destination pairs $\mathcal{P}$; Route allocation state $\Pi$;
**Output**: Route $\gamma_i$ for each pair $p_i \in \mathcal{P}$;

```
1 while (P ≠ φ) do
2     p_i = selectPair(P, Π); // Sec. 4.2;
3     γ_i = assignRoute(p_i, Π); // Sec. 4.3;
4     updateState(γ_i, Π);
5     P = P − {p_i};
```

### 4.2 Communication Pair Selection

In this section, we aim to select a pair to be assigned from $\mathcal{P}$. The way to select a pair is to choose the one that has the least number of possible routes, since other remaining pairs with a larger number of possible routes can tolerate more potential contentions. Thus, we firstly search for the number of potential routes for each pair. If the single-bypass paths through DOR routes (i.e., XY, YX) are not available, we instead turn to find a contention-free multiple-bypass path via intermediate routers to avoid potential contention and eliminate blocking latency. That is, select appropriate intermediate router(s) first, route to the intermediate routers, and then route from the intermediate routers to the destination. Due to only one turn permission within each $HPC_{max}$ quadrant in 2D-Mesh SMART NoCs [5], an appropriate DOR route is employed in each bypass phase. To facilitate analysis, for an analyzed pair $p_i$, we define its possible contention-free *direct route* and *indirect route* via intermediate router(s), as follows:




(a) Single-bypass Direct route (b) Double-bypass Indirect route

Fig. 3: Direct and indirect routes illustration for a source-destination communication pair from s = (0,0) to d = (2,2).

**Definition 1** (Direct Route)**.** *For an analyzed pair, if its packets are forwarded via a single bypass using a DOR route from the source to the destination, such route is defined as a* direct route, *highlighted in the light blue of Fig. 3(a).*

**Definition 2** (Indirect Route)**.** *For an analyzed pair, if the corresponding packets are forwarded via multiple bypasses from the source to the destination, in which a DOR route is employed in each bypass phase, such route is defined as an* indirect route, *highlighted in the light blue of Fig. 3(b).*

Note that, the thick black links (e.g., $R_{(1,1)} \rightarrow R_{(1,2)}$ in Fig. 3(a)) are assigned to other communication pairs. Distinct from a single DOR route, the indirect route via intermediate routers indeed increases the path diversity,

which in turn effectively contributes benefits to contention reduction and workload balance. To strictly decrease the end-to-end latency for a communication pair, in the following, we analyze how to determine the number of intermediate routers of an indirect route. Without loss of generality, consider a contention-free indirect route *path*$'$ with $\mathcal{N}_{imd}$ intermediate routers, and a conflicting direct route *path*.

**Theorem 1.** *For an analyzed pair, assuming the contention-free indirect route path*$'$ *has* $\mathcal{N}_{imd}$ *intermediate routers, if*

$$1 \leq \mathcal{N}_{imd} < \left\lceil \frac{t_r + t_w + min(\mathcal{N}_i)}{t_r + t_w} \right\rceil$$

*then, path*$'$ *strictly dominates the originally conflicting direct route path in terms of end-to-end latency.*

*Proof.* Based on Eq. (2), the end-to-end latency $\mathcal{L}_{e2e-S}$ of the contention-free indirect route *path*$'$ is derived as follows:

$$\mathcal{L}'_{e2e-S} = (t_r + t_w) \cdot (\mathcal{N}_{imd} + 1) + t_w \cdot (\mathcal{N}_i - 1)$$

Then, $\mathcal{L}_{e2e-S} - \mathcal{L}'_{e2e-S} = ((t_r + t_w) \cdot \mathcal{N}_c + \mathcal{L}_b) - (t_r + t_w) \cdot \mathcal{N}_{imd}$. In this article, we aim to ensure $\mathcal{L}_{e2e-S} - \mathcal{L}'_{e2e-S} > 0$ by eliminating the contention count $\mathcal{N}_c$ and blocking latency $\mathcal{L}_b$ through a contention-free indirect route *path*$'$. Let $\mathcal{L}_{e2e-S} - \mathcal{L}'_{e2e-S} = 0$, and then $\mathcal{N}_\lambda = \left\lceil \frac{(t_r + t_w) \cdot \mathcal{N}_c + \mathcal{L}_b}{t_r + t_w} \right\rceil$. To avoid the contention of the conflicting direct route, the contention-free indirect route with $\mathcal{N}_{imd}$ intermediate routers ($\mathcal{N}_{imd} \geq 1$) is selected. Let's consider the following two cases:

- If $\mathcal{N}_{imd} \geq \mathcal{N}_\lambda$, we have $\mathcal{L}_{e2e-S} - \mathcal{L}'_{e2e-S} < 0$. The end-to-end latency $\mathcal{L}'_{e2e-S}$ is increased when adopting the indirect route *path*$'$ with intermediate routers more than $\mathcal{N}_\lambda$, due to significant router-stage latencies (i.e., $(t_r + t_w) \cdot \mathcal{N}_{imd}$) that are higher than the blocking latency when using the originally conflicting direct route *path*.
- If $\mathcal{N}_{imd} < \mathcal{N}_\lambda$, we have $\mathcal{L}_{e2e-S} - \mathcal{L}'_{e2e-S} > 0$. The end-to-end latency is strictly decreased (i.e., $\mathcal{L}'_{e2e-S} < \mathcal{L}_{e2e-S}$) when adopting the indirect route with less than $\mathcal{N}_\lambda$ intermediate routers.

Since $\mathcal{N}_c \geq 1$ and $\mathcal{L}_b \geq min(\mathcal{N}_i)$ (i.e., link width = 1 flit size) in the conflicting direct route *path*, $\mathcal{N}_\lambda \geq \left\lceil \frac{t_r + t_w + min(\mathcal{N}_i)}{t_r + t_w} \right\rceil$. Therefore, regarding the end-to-end latency for an analyzed pair, the contention-free indirect route *path*$'$ strictly dominates the conflicting direct route *path* if $1 \leq \mathcal{N}_{imd} < min(\mathcal{N}_\lambda) = \left\lceil \frac{t_r + t_w + min(\mathcal{N}_i)}{t_r + t_w} \right\rceil$. The theorem is proved. □

Following Theorem 1, the number of intermediate routers $\mathcal{N}_{imd}$ is not the more the better, but $\mathcal{N}_{imd} < min(\mathcal{N}_\lambda)$ that can guarantee strictly latency reduction. Thus, $\mathcal{N}_{imd} < \left\lceil \frac{t_r + t_w + min(\mathcal{N}_i)}{t_r + t_w} \right\rceil$ to guarantee strictly latency reduction, where the maximum $\mathcal{N}_{imd}$ is mainly determined by the minimal packet size $min(\mathcal{N}_i)$ and router-stage latency $t_r + t_w$. The actual number of intermediate routers of a contention-free indirect route should be selected in [1, $\left\lceil \frac{t_r + t_w + min(\mathcal{N}_i)}{t_r + t_w} \right\rceil$).

In this article, for the simplicity of analysis, we only consider the indirect routes with one intermediate router (i.e., $\mathcal{N}_{imd} = 1$), and this method can be easily extended to the indirect routes with multiple intermediate routers. To ensure a flit can finish transmission within one cycle in each bypass phase, the Manhattan distance of each bypass phase is within $HPC_{max}$. In the process of indirect route search, if we choose the source or destination itself as the intermediate router, the derived indirect route is essentially a direct route. To obtain the number of possible idle routes according to the current route allocation state, we search for the possible routes by choosing all of the routers in the allocated region as the intermediate router in turn, where the destination can be reached with the same latency through any double-bypass indirect route.

More specifically, in Line 3 of Alg. 2, the function `getDirectRoute` returns the direct routes by choosing the source or destination router as the intermediate router; while the function `getIndirectRoute` returns the indirect routes by choosing other routers as the intermediate router. Given current pair $p_i$ and an assigned pair $p_j$, assume $p_i \prec p_j$ or $p_j \prec p_i$, meaning that $p_i$ and $p_j$ would not be temporally overlapped. Thus, when considering the possible routes of $p_i$ via `getDirectRoute` and `getIndirectRoute`, the assigned route to $p_j$ can be regarded as available resources, since packet release of $p_i$ is strictly *after* or *before* the transmission completion of the packet from $p_j$. Then, summarize all the distinct routes as the contention-free route candidate set $S_i$ for each pair $p_i$ in Line 3. Finally, we select a pair that has the least number of possible routes to be assigned route in Line 7-9, since other remaining unassigned pairs that have a larger number of possible routes can tolerate more potential contentions. Notice that, when "$|S_i| = 0$" in Line 4-6, meaning that it cannot find a spatially contention-free direct or indirect route, $p_i$ is inserted into the set of unassigned communication pairs $S_{unassigned}$ and removed from the original pair set $\mathcal{P}$.

**Algorithm 2:** *selectPair*($\mathcal{P}, \Pi$)

**Input**: Unallocated pair set $\mathcal{P}$; Allocation state $\Pi$;
**Output**: A selected pair $p_i$ to assign route;

```
1  minRoute = INF, minPair = null;
2  for (p_i ∈ P) do
3      S_i = getDirectRoute(p_i, Π) ∪ getIndirectRoute(p_i, Π);
4      if (|S_i| == 0) then
5          S_unassigned = S_unassigned ∪ {p_i};
6          P = P − {p_i};
7      else if (|S_i| > 0 and |S_i| < minRoute) then
8          minRoute = |S_i|;
9          minPair = p_i;
10 Return minPair;
```

### 4.3 Route Allocation for Selected Pair

We start by observing that, for the selected pair of Sec. 4.2, there are multiple route candidates. In this section, we aim to assign an optimal route from these route candidates. To find such a route, there are two influence factors to be considered when making the decision. Firstly, since communication resources (e.g., router, link) are limited, a long-distance route holding more communication resources will consume more energy and cause more potential contentions on unassigned pairs. Thus, the distance of the desired route should be as short as possible. Secondly, if there are multiple

route candidates with the same distance, the impact of the desired route, which would cause to potential direct routes of unassigned pairs, should be also minimized. For the selected pair $p_i$, depending on whether the two types of route sets (i.e., direct route set $S_i^D$, indirect route set $S_i^I$) are empty or not, the proposed route selection is divided into three cases to deal with in Alg. 3.

**Algorithm 3:** *assignRoute*($p_i, \Pi$)

**Input**: The selected pair $p_i$; Allocation state $\Pi$ for the $\mathcal{N} \times \mathcal{N}$ 2D-Mesh SMART region;
**Output**: Route $\gamma_i$ for the selected pair $p_i$;

1 $\gamma_i = null, minFactor = 0, impactFactor = 0$;
2 //Case 1: contention-free direct route exists;
3 $S_i^D = getDirectRoute(p_i, \Pi)$;
4 **if** *($S_i^D$ is nonempty)* **then**
5 $\quad \gamma_i = minImpactPath(S_i^D)$;
6 //Case 2: contention-free indirect route exists;
7 $\mathcal{D} = ManD(R_s, R_d)$;//Manhattan distance;
8 **while** *($S_{imd} = \{R_i | ManD(R_s, R_i) + ManD(R_i, R_d) = \mathcal{D}, 0 \leq R_i.x < \mathcal{N}, 0 \leq R_i.y < \mathcal{N}\}$, $\mathcal{D}$+=2)* **do**
9 $\quad S_i^I = getIndirectRoute(p_i, S_{imd}, \Pi)$;
10 $\quad$ **if** *($S_i^I$ is nonempty)* **then**
11 $\quad\quad$ **for** *($path_\alpha \in S_i^I$)* **do**
12 $\quad\quad\quad$ **for** *($p_\beta \in S_{unassigned}$)* **do**
13 $\quad\quad\quad\quad S_j^D = getDirectRoute(p_\beta, \Pi)$;
14 $\quad\quad\quad\quad$ **for** *($path_\beta \in S_j^D$)* **do**
15 $\quad\quad\quad\quad\quad$ **if** *($path_\alpha \cap path_\beta \neq \phi$)* **then**
16 $\quad\quad\quad\quad\quad\quad$ *impactFactor*++; break;
17 $\quad\quad\quad$ **if** *(impactFactor == 0)* **then**
18 $\quad\quad\quad\quad$ *minFactor = 0*;
19 $\quad\quad\quad$ **if** *(impactFactor < minFactor)* **then**
20 $\quad\quad\quad\quad$ *minFactor = impactFactor*; $\gamma_i = path_\alpha$;
21 $\quad\quad\quad$ *impactFactor = 0*;
22 //Case 3: both direct and indirect route sets are empty;
23 **if** *($\gamma_i$==null)* **then**
24 $\quad \gamma_i = rand()\%2 == 0\ ?\ XY : YX$;
25 Return $\gamma_i$;

*Case 1: Direct route set $S_i^D$ is nonempty.* For the selected pair $p_i$, we firstly derive the direct route set $S_i^D$ by using the function `getDirectRoute` in Line 3. There are at most two contention-free direct DOR routes (i.e., XY, YX route). The route that causes less impact to unassigned pairs is selected if both direct routes exist in Line 5, where the impact is estimated using the method of *Case 2*.

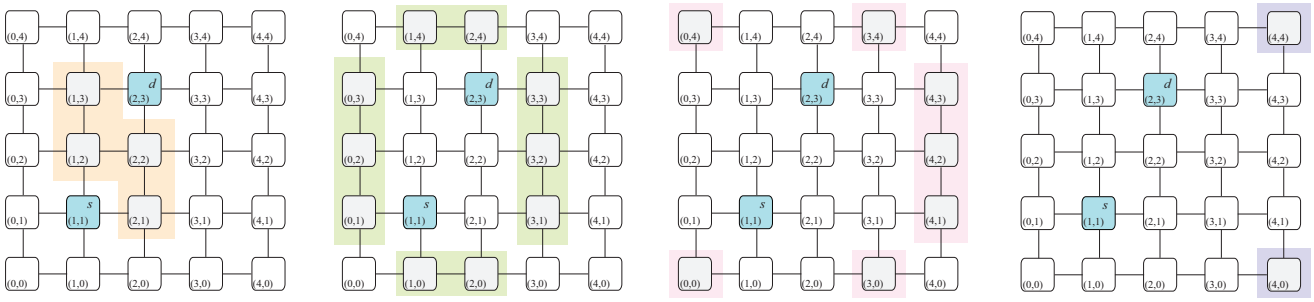
*Case 2: Indirect route set $S_i^I$ is nonempty.* In this case, since direct routes are occupied by the assigned pairs, we thus turn to find an optimal indirect route for the selected pair. The principle behind indirect route selection is to minimize resource usage and impacts caused to available direct routes of unassigned pairs. As an example, the source-destination pair from s = (1,1) to d = (2,3) in Fig. 4 illustrates how to select intermediate router. The candidates of the intermediate router are classified as multiple layers (e.g., L0, L1) based on Manhattan distance. For example in Fig. 4(b), the Manhattan distance of the indirect routes via the intermediate routers with green color in Layer 1 is 5. The indirect route via these intermediate routers belonging to L0 has a minimum distance from the source to the destination, and its distance is added by 2 via the intermediate routers in the adjacent outer layer. To reduce energy consumption and potential contention with unassigned pairs, the indirect route via intermediate routers of the innermost layer is chosen first, since the minimal route occupies minimal resources. If not found in the inner layer, the intermediate router candidates of the adjacent outer layer would be checked.

For each of the indirect route candidates of each layer, to estimate the impact caused to unassigned pairs, two variables are defined in Line 1: *minFactor* recording the minimum impact of route candidates, and *impactFactor* recording the impact of the current candidate. The variable $\mathcal{D}$ in Line 7 is initialized as the Manhattan distance between the source and the destination. Define $S_{imd}$ as the set of intermediate routers that $p_i$ transfers towards the destination. For the intermediate router set $S_{imd}$ of each layer, all of the possible indirect routes via $R_i \in S_{imd}$ are collected into $S_i^I$ according to the current allocation state in Line 9. To choose the route that minimizes the impacts caused to available DOR routes (derived by `getDirectRoute`) of unassigned pairs, the impact of each $path_i$ in $S_i^I$ is estimated from Line 11 to 16. Then, if $path_i$ shares links with any of DOR route of $p_j \in \mathcal{P} \backslash p_i$, the variable *impactFactor* will be added by 1. This is because, when the available direct routes of $p_j$ are assigned in advance, the packets generated by $p_j$ may turn to find indirect routes via an intermediate router, thus leading to extra router-stage latency. But for $p_i$, picking any of the indirect route of $S_i^I$ up can be acceptable, due to the same distance of these indirect routes. Thus, an existing indirect route with minimum impact is selected via an intermediate router of the inner layer. However, there is a chance that indirect routes are not found via an intermediate router of the inner layer. The function `assignRoute` in turn tries to find an alternative indirect route from the adjacent outer layer, without performance degradation of any found indirect route. We can finally find the optimal route (from $S_i^I$) that has minimum distance and meanwhile minimizes the impact caused to these unallocated pairs in $\mathcal{P} \backslash p_i$.

*Case 3: Direct and indirect route sets are empty.* When increasing the workloads, the indirect route still may not be found. In such a case, to isolate and balance the communication workloads, a DOR route is randomly returned in Line 24, and the packets generated by $p_i \in S_{unassigned}$ will transmit as far as they can towards the destination.

### 4.4 Complexity and Implementation Consideration

By combining Alg. 1 - 3, the total time complexity is $\mathcal{O}(\mathcal{M}^2\mathcal{N}^2)$, where $\mathcal{M}$ is the number of pairs and $\mathcal{N} \times \mathcal{N}$ is the mesh size of the allocated NoC region. Thus, our proposed algorithm can be solved within polynomial time. Based on the given pairs and allocated NoC, the route calculation can be conducted by the resource manager [13], [14] at design time, and the time overhead of the route calculation algorithm can be acceptable. After calculation, the routing information can be implemented using lookup tables within each router along the assigned route, and is delivered to them via dedicated control links before it demands. The start router of the pair broadcasts SSR requests toward the right output port to establish a bypass path at runtime. The deadlock-free transmission is guaranteed by additional efforts, e.g., the VCs are classified into 2 classes, $VC_0$ for XY




(a) Innermost Layer 0, D = 3 (b) Layer 1, D = 5 (c) Layer 2, D = 7 (d) Layer 3, D = 9

Fig. 4: Illustration of intermediate router selection for the communication pair from s = (1,1) to d = (2,3).

and $VC_1$ for the YX path. In the case of mixed paths (e.g., XY+YX path), two dedicated VCs in an input buffer are reserved for turn switch at intermediate stop routers, e.g., $VC'_0$ for the XY+YX and $VC'_1$ for the YX+XY path. To achieve this goal without increasing the input buffer size, we can use the variable number of VCs for each VC class and each VC queue can be variable-length according to traffics. Also, the livelock cannot occur since packets always make progress toward their destinations in our routing approach. Our routing is feasible for SMART, and can be also applicable for evolved SMART NoCs (e.g., [9], [10], [21]).

## 5 Improved Routing Strategy

The previously design-time routing strategy can effectively isolate and balance the communication workloads in the case of light congestion, but would become challenging in the case of heavier congestion. In this section, we will improve the above design-time routing algorithm with the consideration of the real-time network state and additionally generated traffics not considered at design time.

### 5.1 Observations

The previously routing strategy of Sec. 4 aims to heuristically search for the routes at design time before the application starts execution, without the support of the resource manager [13], [14]. It indeed can effectively isolate and balance the potentially conflicting communication pairs in the case of light congestion. However, take a panoramic view of the real runtime situations in SMART NoCs, there are twofold existing problems, shown as follows.

Firstly, although the calculation time of the previously design-time routing algorithm can be acceptable since it can be done before the application starts execution, it does not consider the real-time network state. Specifically, the contention between any two communication pairs occurs only when they overlap in both the temporal and spatial dimensions. Even if some communication pairs overlap in the spatial dimension, they may not actually contend in the temporal dimension, leading to non-optimal route allocation. Hence, in the case of heavy congestion, most of the pairs cannot find the spatially contention-free routes (i.e., *Case 3* in Alg. 3), and the previously design-time routing strategy cannot produce the desired results.

Secondly, the previously design-time routing strategy is conducted based on those communication pairs that are known beforehand at design time (e.g., ASIC applications), in which the application range is largely limited. Specifically, for the applications with conditional structures (e.g., *if-then-else* statement), some communication pairs only could be determined and generated at runtime, which is not considered before. On accounting of the real-time network state and runtime additionally generated communication pairs, an improved hybrid routing strategy needs to be proposed, which consists of two parts: design-time route candidates selection and runtime management.

### 5.2 Route Candidates Selection

According to the real-time network state, the pure runtime routing schemes can intuitively achieve better results, but the control overhead of runtime calculation time of the routing algorithm introduces new challenges. If this control overhead is not well addressed, the latency reduction benefits from the flexible routing might be less than the latency penalty of the control overhead. In light of this observation, we propose a hybrid routing by combining the advantages of the design-time and runtime methods while discarding their drawbacks to derive an appropriate route. By revisiting *Case 3* of Alg. 3, even if a communication pair $p_i \in S_{unassigned}$ cannot find the spatially contention-free routes, there is a chance that the contention could be avoided in the temporal dimension at runtime. Thus, for these unassigned pairs $S_{unassigned}$, we instead offer a set of route candidates for them, while an appropriate route will be chosen among them at runtime. With the objective of offering a set of route candidates, there could be different possible strategies. To be specific, we address it with Alg. 4. Notice that the number of route candidates can be regulated according to the actual considerations.

The rationale behind this algorithm is as follows: for the analyzed pair $p_i$, $\delta$ route candidates, which have less number of spatial contentions with the assigned pairs of *Case 1* and *2* in Alg. 3, are selected from the indirect route set $S_i^I$. Therefore, by choosing a route from these candidates, $p_i$ has a lower probability to contend with other pairs at runtime, compared with the remaining possible routes in $S_i^I$. Specifically, at first, all of the direct and indirect routes are derived using the functions `getDirectRoute` and `getIndirectRoute` in Line 1, without the consideration of previous allocation state. The elements of array *count*, which records the number of spatial contentions with the assigned

**Algorithm 4:** *offerRCandidates*($p_i$, $\Pi$)

**Input**: An unassigned pair $p_i$ of *Case 3* in Alg. 3; Allocation state $\Pi$ for the NoC;
**Output**: Route candidates for $p_i$;

1 $S_i^D = getDirectRoute(p_i)$; $S_i^I = getIndirectRoute(p_i)$;
2 Init(count, $|S_i^I|$, 0); // initialize the count elements with 0;
3 **for** ($path_\alpha \in S_i^I$) **do**
4  **for** ($\ell = 0; \ell < |path_\alpha|; \ell$++) **do**
5  **if** ($link_\ell$ *is assigned*) **then**
6  count[$\alpha$]++;
7 $\Gamma_i$ = *getCandidates*($\delta$, $S_i^I$, count);
8 Return $\Gamma_i \cup S_i^D$;

pairs, are initialized as 0 in Line 2. Then, for each route $path_\alpha$ in $S_i^I$, the element $count[\alpha]$ is increased by 1 whenever its $link_\ell$ is assigned to other pairs. Finally, we return the first $\delta$ route candidates from $S_i^I$ sorted by *count* in ascending order, together with the direct routes $S_i^D$. Therefore, the improved hybrid routing strategy is derived by replacing *Case 3* of Alg. 3 with Alg. 4. By combining the Alg. 1 - 4, the total time complexity of the hybrid routing strategy is $\mathcal{O}(\mathcal{M}^2\mathcal{N}^2 + \mathcal{M}\mathcal{N}^3)$, which also can be solved within polynomial time at design time.

### 5.3 Runtime Management

The proposed hybrid routing approach in SMART NoCs [10], [21] requires the centralized management on the global information (e.g., resource occupation state, inter-processor communication requests). This routing approach can be integrated into the resource manager [13], [14] of the Operating System (e.g., conduct task mapping, dynamic voltage, and frequency scaling). When achieving the run-time route allocation, the control latency consists of four parts, including ① The time of *request* control packet from the start router to its resource manager, ② Route candidates *checking* time, ③ *Response* time from the resource manager to the start/intermediate router, and ④ *SSR transmission* time to establish the multi-hop bypass path. Therefore, the control latency needs to be eliminated/hidden to ensure the strict latency reduction benefits from the hybrid routing strategy; otherwise, this control latency could defeat the latency benefits from the contention reduction.

To this end, for the route candidates of a pair, we proactively check the route state according to the current resource occupation state and assign/reserve an idle one before they demand. Based on the design-time task-to-core mapping information determined by a task mapping algorithm (e.g., [7]), the inter-core communication requirements are known beforehand. The *request* control packet, which has multiple bits and carries the pair information (e.g., destination address), can be sent to the manager before the corresponding data packet is generated. Thus, to hide the control latency, we observe that such proactive check operation can be conducted during a frequent time interval, in which the source (e.g., CPU, accelerator) of the communication pair is conducting computation, and the route information can be allocated to the requesting pair ahead of time.

Fig. 5 demonstrates the establishment of an indirect route from the source to the destination via an intermediate

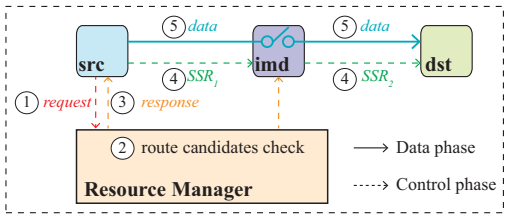



Fig. 5: Indirect route establishment (from the source *src* to the destination *dst* via the intermediate router *imd*).

router in SMART NoCs, which consists of the control phase ①-④ and data phase ⑤. While the establishment of the direct routes is similar to that of the indirect routes, in addition to a stop at the intermediate router.

① *Request.* Before a packet starts actual transmission, the source router of the packet will send the resource manager a *request* control packet to obtain the route information.

② *Route Candidates Check.* For each requesting pair, the resource manager selects an available route from candidates. The route candidates are pre-checked in turn from the direct routes (e.g., XY) to indirect routes according to the resource occupation state. Specifically, the link-state information can be indexed by the link ID, where "1" represents the busy link, while "0" for the idle one. The availability of the route candidate is known by checking all the links of the route candidate. The current route candidates and resource occupation state can be stored in on-chip memories (e.g., private/shared cache) of the resource manager. Once a certain route is assigned to a communication pair, the resource occupation state is updated instantaneously. Multiple requests can be conducted in parallel in the resource manager with multiple threads. Note that, to explore higher communication capability, this step also could be extended to a heuristic-based algorithm (e.g., [33]) with ultralow time complexity for the data transmission with large volume.

③ *Response.* The derived route information is delivered to all the routers along the assigned route. Similar to the design-time routing strategy, the routing information can be implemented using lookup tables. For the dynamic pair at runtime, the XY route is assigned by default. If the workloads are highly dynamic, it needs further investigation on top of our approaches to achieve higher communication performance, such as developing a fully runtime implementation, which is within the scope of our future work. However, there is a chance that all the route candidates are not available in the case of extremely heavy congestion. A DOR route is randomly generated to the current pair, in which the priority-based scheduling approach (e.g., distance and direction based policy [5], first-come-first-serve policy) is employed when encountering contention at intermediate routers during running time. Once the requested communication resources are available again, the start router of the blocked data will broadcast the SSR toward the right output port by checking the source router ID at the lookup table.

④ *SSR Broadcast.* After obtaining the route information, the source and intermediate routers send control packets in parallel to the downstream routers to build the fully multi-hop bypass path. Specifically, $SSR_1$ is to establish the first half of the path, while $SSR_2$ for the second half of the path.

⑤ *Data Transmission.* Instead of a stop at the intermediate router of an indirect route in the design-time routing approach, a flit in the hybrid routing approach traverses over the established path towards the destination directly within a cycle, if the distance of the full route is within $HPC_{max}$. When the data transmission finishes, the resource manager is notified of the resource idle state immediately through dedicated control links, and the network state is updated instantaneously. This update operation of resource occupation state can be conducted concurrently with the requests processing from communication pairs, and thus cannot delay the requests processing and route information delivery to the source/intermediate routers of pairs.

### 5.4 Implementation Consideration

For the design-time and hybrid routing strategies in SMART NoCs, which one will be adopted depends on the traffic density, packet size, and performance requirements. Specifically, for light traffic situations, the design-time strategy can be adopted to isolate the traffics. While for the high saturated traffic situations or large-size messages (e.g., the red AIR numbers in Fig. 6), the design-time strategy performs poorly but the hybrid one can be adopted with the help of the centralized resource manager. In this work, we mainly focus on the strategic design of the routing approach, and do not limit the implementation in software or hardware. Regarding the scalability problem of centralized management, one general solution is the distributed cluster-based management approach (e.g., [14], [34]). Specifically, SMART NoCs can be split into multiple cluster networks, in which the cluster size is user-defined. The global hardware overhead is limited to the cluster level, instead of the whole on-chip network, and thus the distributed cluster-based NoC can be extended to a larger-size network. On accounting of the contention minimization from the perspective of routing design, the communication topology can be well reconfigured at runtime and the long-range bypass path is built as possible. Thus, the communication performance of the (evolved) SMART NoCs can be correspondingly improved.

## 6 Experimental Evaluation

In this section, we conduct a set of experiments to quantitatively evaluate the communication performance of our proposed routing strategies in SMART NoCs, compared with the current approaches. We conduct the experiments by using a set of realistic benchmarks and derive their communication performance and energy consumption respectively.

### 6.1 Experimental Settings

We conduct experiments on $\mathcal{N} \times \mathcal{N}$ $(= 4, 8, 16)$ Alpha 21264 array cores in 22-nm technique as the experimental infrastructure and build the realistic infrastructure by gem5 [35], on which the StreamIt benchmarks [36] are adopted, including *Autocor, Audiobeam, FMRadio* and *H264*. The employed benchmarks are broadly representative of the real-world applications, as they have different orders of magnitude in workloads and cover audio processing, video processing, scientific processing, etc. Gem5 is a widely-used open-source computer architecture simulator, which provides a cycle-accurate timing model as well as full-system simulation. In gem5, a large number of models are implemented, such as CPU models (e.g., in-order designs), on-chip interconnection, a detailed DRAM model, coherent caches, etc. Thus, on top of the gem5, the task mapping, scheduling and routing behaviors of benchmark applications can be precisely simulated as realistic systems.

The application performance (e.g., application running time) of a running application is jointly determined by the task mapping, scheduling, and routing schemes. To be specific, in the simulator, we model the task mapping with the state-of-the-art method in [7], first-come-first-serve (FCFS) scheduling as well as our proposed routing strategies, and record the communication performance of each experiment. To ensure a fair comparison, the task mapping and scheduling schemes are the same and fixed under different routing strategies. Assume the maximum hop count $HPC_{max} = 9$ within a 1GHz cycle for SMART NoCs [5]. The width of the NoC data links is one flit size, and the transmission latency (i.e., $t_w$) of a flit between two adjacent stopping routers is one cycle. For the hybrid routing, the maximal number of indirect route candidates $\delta$ is set as 3 in the experiments, such that the route candidates checking time is not large.

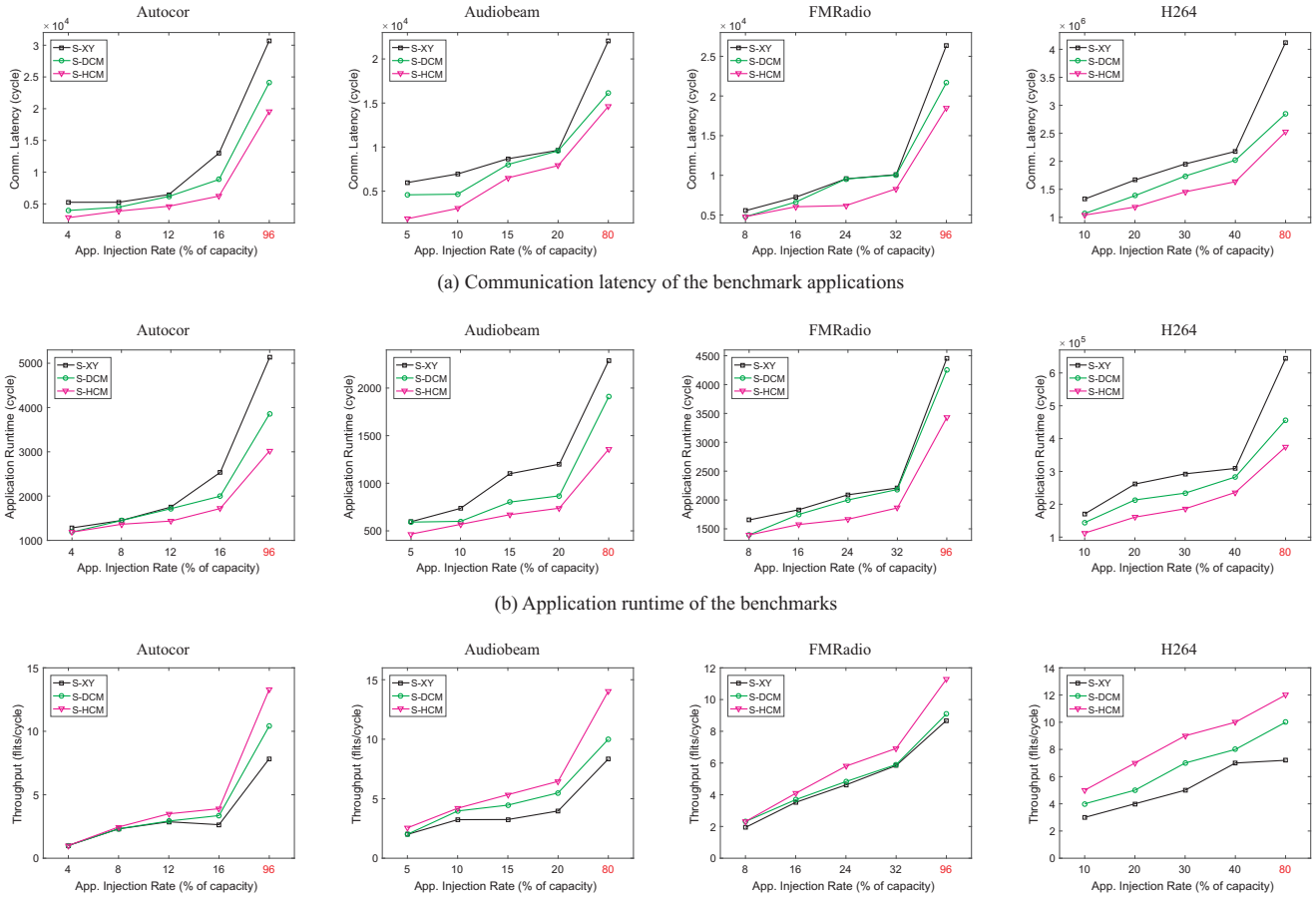
To facilitate quantitative comparison, the proposed routing approaches and baselines are summarized as follows:

- *S-XY*. The XY routing in SMART NoCs, to our best knowledge, which is the current routing approach, since no related routings are studied in SMART NoCs [5] and the routing approaches in traditional NoCs are not fully applicable to SMART NoCs.
- *S-DCM*. Our proposed basic design-time contention minimized routing in SMART NoCs.
- *S-HCM*. The improved hybrid contention minimized routing in SMART NoCs, which combines the advantages of design-time and runtime routing methods.

To reflect the communication performance variation with the traffic injection rate, the StreamIt benchmarks are mapped on the SMART NoCs with the application injection rate (AIR) [32], which is generated by the negative exponential distribution [37]. A larger AIR will contribute to a higher packet injection rate, heavier traffic congestion, more packet contention, and a larger number of communication pairs. To illustrate the effectiveness of the proposed routing strategies in the case of heavy congestion, we set a high AIR (shown in red AIR value in the figures) in the experiments.

### 6.2 Comparison of Communication Performance

In this section, we first conduct the experiments on StreamIt benchmarks to validate the communication performance of the proposed design-time and hybrid routing strategies, i.e., *S-DCM* and *S-HCM*, in communication latency, application runtime, and throughput. The communication latency is derived by summarizing the latency of all the packets during the runtime of an application; the application runtime refers to the total length of the execution time for a running application; and the network throughput denotes the number of received flits per cycle in the network. The benchmark applications are mapped on $8 \times 8$ mesh SMART NoCs with the variation of application injection rate. The




(a) Communication latency of the benchmark applications

(b) Application runtime of the benchmarks

(c) Network throughput of the SMART NoC

Fig. 6: Experimental results comparison in terms of *communication latency*, *application runtime* and *throughput* of realistic StreamIt benchmarks under different routing strategies in SMART NoCs.

communication performance is compared under different routing strategies and parameter settings.

As the experimental results are shown in Fig. 6, because of the contention reduction with direct/indirect routes, the communication performance of our proposed routing strategies (*S-DCM* and *S-HCM*) in terms of communication latency, application runtime, and network throughput are dramatically improved, compared with the baseline *S-XY* in SMART NoCs. Specifically, in the total collected data traces, the proposed *S-DCM* and *S-HCM* illustrate an average of 16.13 and 35.48 percent reduction in communication latency, an average of 14.66 and 28.31 percent reduction in application runtime, and an average of 16.09 and 37.59 percent improvement in throughput, respectively, compared with the baseline *S-XY* in SMART NoCs.

With the increase of the application injection rate, shown in Fig. 6, the communication latency and application runtime are expectedly increased due to the more contention encounters, while the throughput is increased due to the more injected traffics. We can observe that the results are consistent in different degrees of traffic congestion, showing the good scalability of our approach. In particular, in the case of heavy congestion (i.e., the red AIR value in Fig. 6), the improvements in terms of communication performance are most apparent when using our proposed routing strategies *S-DCM* and *S-HCM* in most cases. This is because, our routing strategies always select a contention-free direct/indirect route according to real-time network state/route allocation state as possible, such that the communication performance is largely improved by the maximal utilization of long-distance bypass transmission.

Although *S-DCM* and *S-HCM* both consider flexibly select a contention-free route with reduced contention, they are remarkably different in routing performance. Specifically, *S-HCM* is averagely 23.08, 15.99 and 18.52 percent better than *S-DCM* in terms of communication latency, application runtime and throughput, respectively, in Fig. 6. This is because, *S-DCM* avoids the contention at design time only in the spatial dimension. While *S-HCM* provides several direct/indirect route candidates that are derived at design time, such that there are more chances of finding a contention-free route from these candidates at runtime according to real-time network state. Therefore, the communication pairs can be isolated and balanced in a more fine-grained manner from the perspective of both spatial and temporal dimensions. Note that, since we have considered realistic benchmark applications, the performance improvements can be slightly different due to the different task mapping and scheduling schemes. But the relative performance improvement when using such benchmark applications can be correctly reflected to some extent.

Then, we conducted another two sets of experiments to

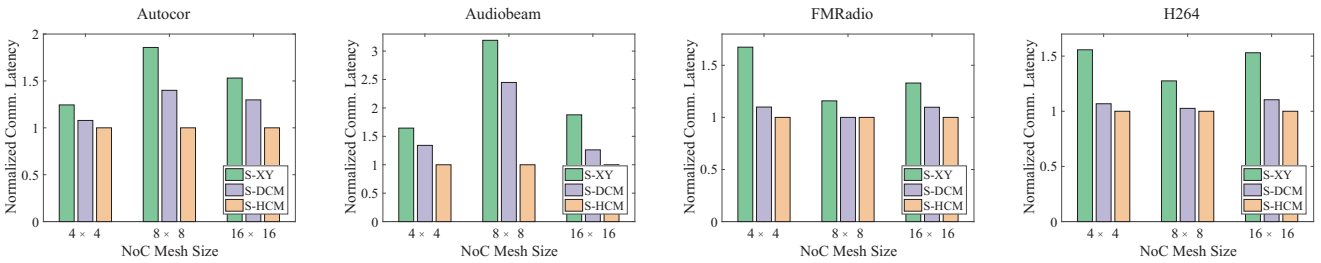



Fig. 7: Comparison of *communication latency* in different mesh size on StreamIt benchmark applications.

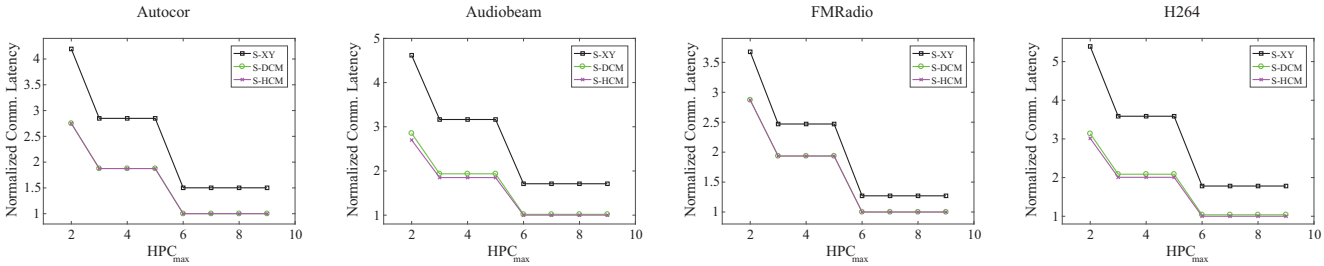



Fig. 8: Comparison of *communication latency* in different $HPC_{max}(\leq 9)$ on StreamIt benchmark applications.

evaluate the scalability of the routing strategies in terms of different mesh size and $HPC_{max}$ parameter, in which we recorded the communication latency of a set of StreamIt benchmark applications. Fig. 7 demonstrates the results of normalized communication latency are consistent in different mesh sizes, showing the scalability of our proposed routing strategies. This is because, in different mesh size, the design-time routing *S-DCM* has well isolated and balanced the communication pairs, such that it can achieve the higher communication efficiency than the baseline *S-XY* in SMART NoCs; while the hybrid routing *S-HCM* further improves the communication efficiency, since it additionally considers the real-time network state, which in turn further achieves workload balance and contention avoidance. Due to the random distribution of some successor tasks in task-to-core mapping [7] in different mesh size, the performance of the same routing strategy may not be strictly decreased/increased with the increase of mesh size, since the communication performance is jointly determined by the task mapping, scheduling, and routing schemes.

Fig. 8 illustrates the results in terms of normalized communication latency are better than the baseline *S-XY* in different $HPC_{max}$ parameter, also showing the scalability of our proposed routing strategies. With the increasing of $HPC_{max}$, by and large, the communication latency is reduced due to more utilized bypass paths except for some $HPC_{max}$ intervals (e.g., from 3 to 5) where the latency almost remains unchanged. The reason is that the effective number of hops of a communication pair remains the same during a $HPC_{max}$ range, thus the latency cannot be reduced apparently. In particular, *S-HCM* has very close performance with *S-DCM* in *Autocor* and *FMRadio*, since the contention issue of these two benchmarks is not serious in the experiments and there is less room to avoid the contention in the temporal dimension to improve the performance. While *S-HCM* performs apparently better than *S-DCM* for *Audiobeam* and *H264*, since the congested traffics are further isolated and balanced in the temporal dimension.

### 6.3 Comparison of Energy Consumption

In this section, we record and illustrate the energy consumption of each benchmark application in the experiments. For an application running in NoCs, we generally consider the systematic energy consumption as the total energy consumption $E$ of each run including task execution $E_{exe}$ and data communication $E_{com}$. Specifically, $E = E_{exe} + E_{com}$, and

$$E_{exe} = P \cdot t,$$

$$E_{com} = \sum_{p_i \in \mathcal{P}} \{[(h_i + 1) \cdot E_{router} + h_i \cdot E_{rrl} + 2 \cdot E_{rcl}] \cdot \mathcal{N}_i + (h_i + 1) \cdot E_{cu}\}.$$

Where $P$ (i.e., $P = CV^2f$, where *V/f* is the voltage/frequency level; *C* is the capacitance) refers to the per-core power of task execution, which is a constant parameter and derived by the power model McPAT [38]; and $t$ is the total task execution time of each application on processors. $h_i$ is the number of transmitted hops from the source to the destination of the packet $p_i$. $E_{router}$ is the energy consumption of a single bit crossing a router, and $E_{router} = E_{xbar} + buff \times E_{buff}$. Averagely, $E_{xbar} = 0.108\ pJ/bit$, $buff = 8$, $E_{buff} = 0.0078\ pJ/bit$. $E_{rrl}$ and $E_{rcl}$ denote the energy consumption of a single bit transmitted through the router-to-router link (*rrl*) and router-to-core link (*rcl*), respectively, where $E_{rrl} = 0.031\ pJ/bit$ and $E_{rcl} = 0.008\ pJ/bit$. $\mathcal{N}_i$ is assumed the packet size in bits, where $\mathcal{N}_i = 128\ bit$ [5]. And $E_{cu}$ denotes the control overhead to make an arbitration for each packet, where $E_{cu} = 0.917\ pJ/bit$. These energy constants are derived by hardware synthesis with a circuit simulator Hspice in 22-nm library [39]. According to the above two formulas, for a specific packet, the energy consumption is increased with the distance increase of the source-destination communication pair.

For the application footprints in Sec. 6.2, the experimental results in Fig. 9 and Fig. 10 demonstrate the ratio of the

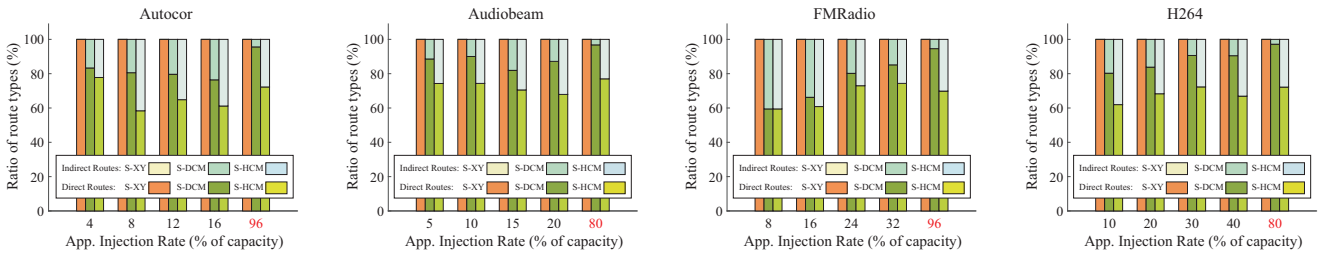



Fig. 9: The ratio of direct/indirect routes derived by different routing strategies in SMART NoC.

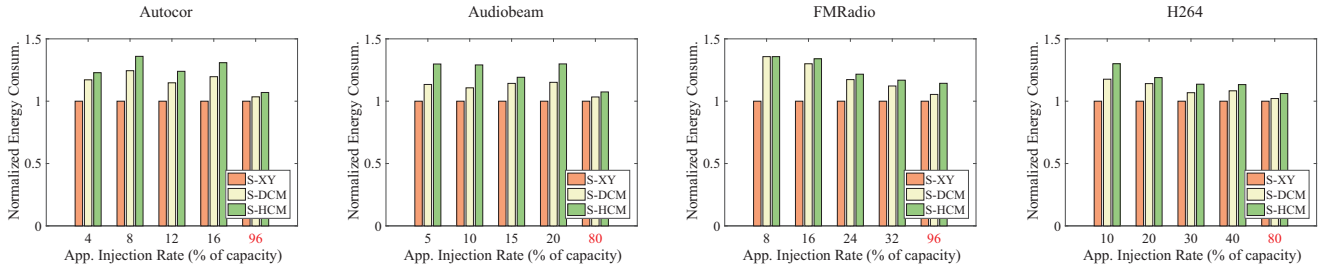



Fig. 10: Results comparison of systematic energy consumption of StreamIt benchmarks in SMART NoC.

derived route types and the systematic energy consumption in different routing strategies, respectively. The systematic energy consumption consists of energy consumption from task execution and communication transmission. Once the core parameters are configured (e.g., frequency of the cores), the task execution time is the same and the energy consumption from task execution also remains the same. Thus, systematic energy consumption is mainly determined by communication transmission. From the experimental results in Fig. 9, to fully avoid the communication contentions, our routing strategies *S-DCM* and *S-HCM* averagely employ $15.60$ and $31.12$ percent indirect routes in the total derived routes. Thus, as shown in Fig. 10, *S-DCM* and *S-HCM* averagely consume $14.3$ and $22.05$ percent more energy than the baseline *S-XY*, respectively, due to the adoption of indirect routes. Note that, in the case of heavy congestion (e.g., red AIR values), *S-HCM* still cannot find a contention-free route for some communication pairs, and a DOR route with minimal distance is assigned by default. Besides, the distance of some indirect routes via the inner intermediate routers (e.g., $R_{(2,2)}$ in Fig. 4) is also minimal. Thus, the energy consumption of the same routing strategy is not strictly decreased/increased with the increase of AIR.

### 6.4 Overhead Analysis

In this subsection, we will discuss three kinds of the overhead of the proposed routing strategy in SMART NoCs: energy, hardware/area, and control time overhead.

*Energy overhead.* According to the equations of the energy consumption in Sec. 6.3, the communication energy consumption is proportional to the number of transmitted hops $h_i$ from the source to the destination of packet $p_i$. Experimental results show the hybrid routing strategy *S-HCM* can improve the communication efficiency up to $37.59$ percent with $22.05$ percent additional energy consumption, compared with the baseline routing approach *S-XY* in S-MART NoCs. In the above experiments, we did not limit the Manhattan distance of the indirect routes, and thus additional techniques can mitigate the route energy consumption of the indirect routes according to actual requirements, such as slightly limiting the Manhattan distance of the non-minimal indirect routes during path selection.

*Hardware/area overhead.* Due to the need for task mapping [32] and chip thermal distribution [40], NoCs are necessarily equipped with the resource manager [13], [14] to achieve the cluster-level global management in communication efficiency and chip thermal reliability in the dark silicon era. The hardware/area overhead of control links (e.g., forwarding control signals) and network information storage (e.g., resource occupation state, processor temperature) is a necessary part in distributed cluster-based management [14], [34], in which the existing hardware resources can be shared. Regarding the storage of network information and allocated routes, like the task-to-core information, they can be partly/fully stored in on-chip memories (e.g., private/shared cache) of the resource manager. For example, one bit can represent the state of a link. For an $\mathcal{N}\times\mathcal{N}$ cluster NoC, the required storage space of the $4\mathcal{N}(\mathcal{N}-1)$ links is $4\mathcal{N}(\mathcal{N}-1)$ bits. In consideration of the $HPC_{max}$ limitation [5] and NoC scalability [34] in SMART NoCs, generally the cluster size $\mathcal{N} \leq 8$. As shown in [5], each core of SMART NoCs has 32KB private L1 cache and 1MB private/shared L2 cache. This additional storage space is acceptable when equipped with the KB- and MB-level cache.

*Control time overhead.* Compared with the originally distributed routing (i.e., without the support from the resource manager) in SMART NoCs [5], the control timing overhead is the control latency (i.e., the steps ①-③ in Sec. 5.3) to obtain an available route. The control latency can be fully hidden in a frequent time interval, as stated in Sec. 5.3, in which the source (e.g., CPU, accelerator) of the pair is conducting computation. It is possible since the *request* control packet can be sent to the resource manager before the packet generation according to the design-time task-to-core mapping information. Specifically, steps ① and ③

take two cycles when the *request/response* control packets are transmitted between the packet source and resource manager via the dedicated control links; while step ② takes a short time to check the route candidates, since it can be conducted with multiple threads in the OS of the resource manager. To guarantee the high computation parallelism, the cooperating heavy tasks are usually mapped to different cores, and the execution time of the single partitioned tasks should not be too short with the consideration of the inter-core communication overhead. Otherwise, the communication latency overhead will defeat the benefits from computation parallelism. Thus, the control latency can be strictly less than the minimal task execution time by design.

## 7 Conclusion

Although SMART NoC is promising for NoC-based multi/many-core systems, the contention issue cannot be effectively solved, and then the multi-hop bypass cannot be fully utilized, which in turn reduces the benefits that SMART offers. In this article, we propose a system-level contention-minimized routing strategy: when potential contention is detected with a direct route, we alternatively select a contention-free multiple-bypass indirect route. The contention-induced communications are isolated in both temporal and spatial dimensions, which in turn improves SMART performance due to the improvement of bypass utilization. In particular, our proposed approach breaks the minimal-distance routing tradition without the latency penalty, paving the way for performance scalability of future kilo-core chips. In future work, we will explore more communication capability and reconfigurability by jointly studying the task mapping, scheduling, and packet routing strategies, and eliminating one turn permission in each bypass phase with architecture optimization.

## Acknowledgments

This work is partially supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MoE2019-T2-1-071) and Tier 1 (MoE2019-T1-001-072), and NTU, Singapore, under its NAP (M4082282) and SUG (M4082087). An earlier conference version [23] of this article has been accepted in the *Proceedings of the 25th Asia and South Pacific Design Automation Conference* (ASP-DAC 2020).

## References

[1] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y.-C. Liu, "Knights landing: Second-generation intel xeon phi product," *Ieee micro*, vol. 36, no. 2, pp. 34–46, 2016.

[2] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE micro*, vol. 27, no. 5, pp. 51–61, 2007.

[3] D. Chavarria-Miranda, M. Halappanavar, and A. Kalyanaraman, "Scaling graph community detection on the tilera many-core architecture," in *2014 21st International Conference on High Performance Computing (HiPC)*. IEEE, 2014, pp. 1–11.

[4] W. J. Dally and et al, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[5] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh, "Breaking the on-chip latency barrier using smart," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 378–389.

[6] C.-H. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "Smart: A single-cycle reconfigurable noc for soc applications," in *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2013, pp. 338–343.

[7] L. Yang, W. Liu, P. Chen, N. Guan, and M. Li, "Task mapping on smart noc: Contention matters, not the distance," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[8] B. K. Joardar, K. Duraisamy, and P. P. Pande, "High performance collective communication-aware 3d network-on-chip architectures," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 1351–1356.

[9] I. Pérez, E. Vallejo, and R. Beivide, "Smart++ reducing cost and improving efficiency of multi-hop bypass in noc routers," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–8.

[10] Y. Asgarieh and B. Lin, "Smart-hop arbitration request propagation: Avoiding quadratic arbitration complexity and false negatives in smart nocs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 6, pp. 1–25, 2019.

[11] H. Chen, P. Chen, J. Zhou, L. H. Duong, and W. Liu, "Arsmart: An improved smart noc design supporting arbitrary-turn transmission," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[12] N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. G. Dreslinski, D. Blaauw, and T. Mudge, "Scaling towards kilo-core processors with asymmetric high-radix topologies," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2013, pp. 496–507.

[13] M. A. Al Faruque, R. Krist, and J. Henkel, "Adam: run-time agent-based distributed application mapping for on-chip communication," in *2008 45th ACM/IEEE Design Automation Conference*. IEEE, 2008, pp. 760–765.

[14] G. Castilhos, M. Mandelli, G. Madalozzo, and F. Moraes, "Distributed resource management in noc-based mpsocs with dynamic cluster sizes," in *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2013, pp. 153–158.

[15] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Shortpath: A network-on-chip router with fine-grained pipeline bypassing," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3136–3147, 2016.

[16] A. Davare, J. Chong, Q. Zhu, D. M. Densmore, and A. L. Sangiovanni-Vincentelli, "Classification, customization, and characterization: Using milp for task allocation and scheduling," *Systems Research*, 2006.

[17] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proceedings of the 34th annual international symposium on Computer architecture*, 2007, pp. 126–137.

[18] U. Y. Ogras and R. Marculescu, ""it's a small world after all": Noc performance optimization via long-range link insertion," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 14, no. 7, pp. 693–706, 2006.

[19] C. Jackson and S. J. Hollis, "Skip-links: A dynamically reconfiguring topology for energy-efficient nocs," in *2010 International Symposium on System on Chip*. IEEE, 2010, pp. 49–54.

[20] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *2008 IEEE international conference on computer design*. IEEE, 2008, pp. 164–169.

[21] X. Chen and N. K. Jha, "Reducing wire and energy overheads of the smart noc using a setup request network," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3013–3026, 2016.

[22] W. Liu, P. Chen, L. Yang, M. Li, and N. Guan, "Work-in-progress: fixed priority scheduling of real-time flows with arbitrary deadlines on smart nocs," in *2017 International Conference on Embedded Software (EMSOFT)*. IEEE, 2017, pp. 1–2.

[23] P. Chen, W. Liu, M. Li, L. Yang, and N. Guan, "Contention minimized bypassing in smart noc," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 205–210.

[24] M. Karunaratne, A. K. Mohite, T. Mitra, and L.-S. Peh, "Hycube: A cgra with reconfigurable single-cycle multi-hop interconnect," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[25] P. Chen, W. Liu, H. Chen, S. Li, M. Li, L. Yang, and N. Guan, "Reduced worst-case communication latency using single-cycle multihop traversal network-on-chip," *IEEE Transactions on Computer-*

*Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1381–1394, 2020.
[26] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. IEEE, 2008, pp. 203–214.
[27] T. Krishna and L.-S. Peh, "Single-cycle collective communication over a shared network fabric," in *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*. IEEE, 2014, pp. 1–8.
[28] A. Psarras, J. Lee, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Phasenoc: Versatile network traffic isolation through tdm-scheduled virtual channels," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 844–857, 2015.
[29] M. Ramakrishna, V. K. Kodati, P. V. Gratz, and A. Sprintson, "Gca: Global congestion awareness for load balance in networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2022–2035, 2015.
[30] I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Automatic generation of peak-power traffic for networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 1, pp. 96–108, 2018.
[31] M. S. Rahman, S. Bhowmik, Y. Ryasnianskiy, X. Yuan, and M. Lang, "Topology-custom ugal routing on dragonfly," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–15.
[32] W. Liu, L. Yang, W. Jiang, L. Feng, N. Guan, W. Zhang, and N. Dutt, "Thermal-aware task mapping on dynamically reconfigurable network-on-chip based multiprocessor system-on-chip," *IEEE Transactions on Computers*, vol. 67, no. 12, pp. 1818–1834, 2018.
[33] A. Hansson, K. Goossens, and A. Rdulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2005, pp. 75–80.
[34] M. Ruaro, N. Velloso, A. Jantsch, and F. G. Moraes, "Distributed sdn architecture for noc-based many-core socs," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–8.
[35] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, B. Beckmann, S. Bharadwaj *et al.*, "The gem5 simulator: Version 20.0+," *arXiv preprint arXiv:2007.03152*, 2020.
[36] W. Thies and S. Amarasinghe, "An empirical characterization of stream programs and its implications for language and compiler design," in *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 2010, pp. 365–376.
[37] A.-M. Rahmani, I. Kamali, P. Lotfi-Kamran, A. Afzali-Kusha, and S. Safari, "Negative exponential distribution traffic pattern for power/performance analysis of network on chips," in *2009 22nd International Conference on VLSI Design*. IEEE, 2009, pp. 157–162.
[38] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.
[39] M. Karimian, M. Dousti, M. Pouyan, and R. Faez, "An improved macro-model for simulation of single electron transistor (set) using hspice," in *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*. IEEE, 2009, pp. 1000–1004.
[40] M. Li, W. Liu, L. Yang, P. Chen, and C. Chen, "Chip temperature optimization for dark silicon many-core systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 941–953, 2017.

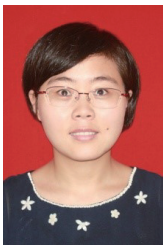
**Peng Chen** received the B.E. degree from the School of Big Data and Software Engineering, Chongqing University, Chongqing, China, in 2015, where he is currently pursuing the Ph.D. degree with the College of Computer Science. He is currently a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include embedded and real-time systems, and network-on-chip.

**Hui Chen** received the B.E. degree from the School of Computer Science and Technology at Zhejiang University, Hangzhou, China, and the M.Sc. degree from the School of Computing, National University of Singapore, Singapore, in 2016 and 2018, respectively. She is currently pursuing the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Her current research interests include network-on-chip.

**Jun Zhou** is a Postdoctoral Research Fellow in Nanyang Technological University, Singapore. He received the Ph.D. from Institute of Computing Technology, Chinese Academy of Sciences. His main research interests include embedded systems, wireless sensor networks, reliability and security for IoT and VLSI.

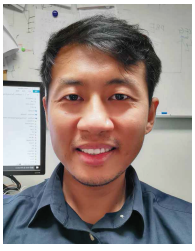
**Mengquan Li** received the B.E. degree from College of Computer Science, Chongqing University, Chongqing, China, in 2015, where she is currently pursuing the Ph.D. degree with the College of Computer Science. She is currently a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Her current research interests include optical network-on-chip, on-chip temperature modeling and optimization.

**Weichen Liu** received the B.Eng and M.Eng degrees from the Harbin Institute of Technology, China in 2004 and 2006, respectively, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong in 2011. He is a Nanyang Assistant Professor at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include network-on-chip, and machine learning acceleration.



**Chunhua Xiao** is currently an associate professor at the College of Computer Science, Chongqing University, China. She received the Ph.D. degree from the Beijing University of Technology, China. Her research interests include MPSoCs, hardware and software co-design, and embedded systems.

**Yiyuan Xie** received the Ph.D. degree in optical engineering from the Chinese Academy of Science, Beijing, China, in 2009. He is currently a Professor at the School of Electronic and Information Engineering, Southwest University, Chongqing, China. His current research interests include optical networks on-chip, ultra-high optical communications, optical chaotic secure communication, and optical data centers.



**Nan Guan** received his B.E. and M.S. from Northeastern University, China in 2003 and 2006, respectively, and the Ph.D. from Uppsala University, Sweden in 2013. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China. His research interests include real-time embedded systems, cyber-physical systems (CPS) and Internet-of-Things (IoT).