Contention Minimized Bypassing in SMART NoC

Peng Chen[1,2], Weichen Liu[1], Mengquan Li[1,2], Lei Yang[3], Nan Guan[4]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore.
[2]College of Computer Science, Chongqing University, Chongqing, China.
[3]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, USA.
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China.

***Abstract*—SMART, a recently proposed dynamically reconfigurable NoC, enables single-cycle long-distance communication by building single-bypass paths. However, such a single-cycle single-bypass path will be broken when contention occurs. Thus, lower-priority packets will be buffered at intermediate routers with blocking latency from higher-priority packets, and extra router-stage latency to rebuild remaining path, reducing the bypassing benefits that SMART offers. In this paper, we *for the first time* propose an effective routing strategy to achieve nearly contention-free bypassing in SMART NoC. Specifically, we identify two different routes for communication pairs: *direct route*, with which data can reach the destination in a single bypass; and *indirect route*, with which data can reach the destination in two bypasses via an intermediate router. If a direct route is not found, we would alternatively resort to an indirect route in advance to eliminate the blocking latency, at the cost of only one router-stage latency. Compared with the current routing, our new approach can effectively isolate conflicting communication pairs, greatly balance the traffic loads and fully utilize bypass paths. Experiments show that our approach makes 22.6% performance improvement on average in terms of communication latency.**

I. Introduction

Network-on-chip (NoC) is a widely-used communication backbone for multi-/many-core systems, in which the communication latency is critical for system performance [1]. A higher latency also causes poorer throughput, especially for latency-sensitive applications. There are two main factors influencing the communication latency: the number of hops (hop count) between source and destination, and contention issues that introduce blocking latency from higher-priority packets. The former factor is inherently restricted by the distance of the source-destination communication pair, and the latter one is related to task mapping and routing schemes.

To address the first factor influencing the communication latency mentioned above, an advanced NoC using the single-cycle multi-hop asynchronous repeated traversal (SMART) design was proposed [1], [2], which can effectively reduce the hop count by dynamically building a single-cycle multi-hop path via bypass, and works especially well for long-distance communication source-destination pairs. SMART employs bypass control and integrated clockless repeaters based on traditional routers. It mainly takes advantage of the following two facts: (1) The electric signal can propagate multi-millimeters in a single cycle; (2) Packets can bypass the intermediate routers where no contention occurs. When packets encounter contention at intermediate routers, SMART only builds the single-cycle multi-hop path from source to the first conflicting intermediate router. If the contention is not addressed, the single-cycle single-bypass path from source to destination is broken, thus reducing the bypassing benefits that SMART offers. Especially, when packets frequently encounter contention at intermediate routers (in the extreme case, at every intermediate router), SMART performs the same as traditional NoCs with hop-by-hop traversal. No more benefits can be acquired than the traditional counterpart in such cases, but incurring more control overhead (i.e., control link). Therefore, to fully utilize the capability of SMART NoCs, the multi-hop bypassing mechanism must be utilized in the way such that contentions are minimized as much as possible.

In this paper, we *for the first time* address the contention minimization problem for bypassing in SMART from the perspective of the routing strategy. In current XY routing for SMART, if contention occurs at intermediate routers, packets will suffer blocking latency from higher-priority packets, and extra router-stage latency to rebuild the remaining path. We optimize the routing performance by identifying two different source-destination routes: *direct route* that is a contention-free single-bypass path, with which data can reach the destination directly, and *indirect route* that is a contention-free double-bypass path, with which data can reach the destination indirectly via an intermediate router. We firstly try to exploit a *direct route* from source to destination according to the current network state. If a direct route is not found, we instead turn to exploit an *indirect route* to eliminate blocking latency from higher-priority packets, at the cost of only one router-stage latency. Contrary to an intuitive approach, not the routes with minimal distance but the indirect routes via an arbitrary intermediate router (even if they may be non-minimal) that result in contention minimization yield the minimized communication latency. Experimental evaluation shows averagely 22.6% improvement using our flexible routing strategy, compared with current routing for SMART NoCs.

II. Background

A. End-to-end Latency in Traditional NoCs

In traditional NoCs with hop-by-hop traversal [3], packets traverse from source to destination hop by hop through on-chip routers and links. In general, the end-to-end latency of a packet covers the transmission latency of the head flit from source to destination, serialization latency of the rest of flits, and blocking latency suffered from higher-priority packets. Therefore, the end-to-end latency $\mathcal{L}_{e2e-T}$ of a transmitted packet from source Processing Element (PE_s) to destination Processing Element (PE_d) in traditional NoCs can be formulated as:

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

where t_r is the router-stage latency; t_w is the link latency between two adjacent routers; h_{sd} refers to the hop count between source PE_s and destination PE_d; $\mathcal{N}_i$ refers to the

978-1-7281-4123-7/20/$31.00

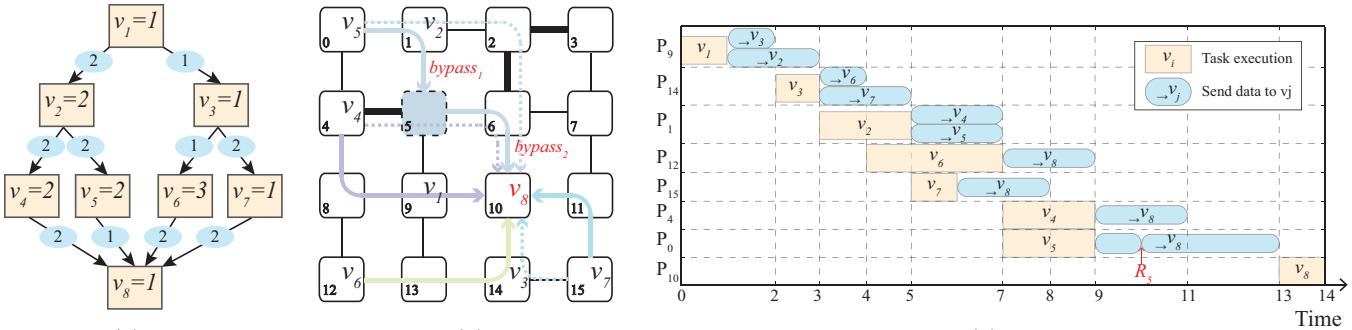


Fig. 1: Motivational example in a 2D-Mesh SMART (HPC_{max} = 6). (a). Directed acyclic graph (DAG) modeled application including execution time and communication volume; (b). *Our* proposed routes to the destination router $R_{10}(v_8)$, where the dotted lines are original routes; (c). Timeline of the computation and communication tasks with our routing in SMART NoC.

number of transmission data units (i.e., flit) of a packet; and $\mathcal{L}_b$ refers to the summation of blocking latency suffered from higher-priority packets along the path to destination. For a given NoC platform and an analyzed packet, t_r, t_w, and $\mathcal{N}_i$ are constants. As can be seen from Eq. 1, the end-to-end latency is only related to the hop count h_{sd} and blocking latency $\mathcal{L}_b$.

As a result, there are mainly two ways to reduce end-to-end latency in traditional NoCs: (1) reducing the hop count h_{sd}, and (2) reducing blocking latency $\mathcal{L}_b$ by contention avoidance. For the first approach, high-radix technique [4], application-specific long-range link [5] and skip-links [6] are proposed to reduce hop count. As for the second approach, contention-aware mapping [7], [8] are also proposed to avoid contention.

B. End-to-end Latency in SMART NoCs

Among the NoC architectures aiming to reduce hop count, another NoC architecture, SMART [1], [2], enables single-cycle long-distance communication by dynamically building a single-bypass direct path from source to destination, which in turn effectively reduces the end-to-end latency. Unlike the above-mentioned ones [4]–[6], SMART NoC has advantages on area, power and layout complexity.

SMART [1], [2] is proposed by embedding the low-swing clockless repeated link and bypass control in traditional routers. It mainly bases on the following two facts: (1) The electric signal can be transmitted multi-millimeters in a single cycle; (2) The incoming data can bypass the intermediate routers where no contention occurs. To realize single-cycle long-distance transmission, SMART adds a set of dedicated SMART-hop Setup Request (SSR) links spanning up to HPC_{max} neighbors [1] for each possible dimension-order routing path, which are used to forward bypass requests to downstream intermediate routers. The working process of SMART mainly consists of three steps: *Step 1:* Each start router performs Local Switch Allocation (SA-L) to choose a winner from buffered flits for each output port; *Step 2:* SA-L winner broadcasts SSR to downstream routers towards the destination to setup bypass path and continually conducts Global Switch Allocation (SA-G) for competing SSRs. The bypass path is built at the end of this step; *Step 3:* The packet of the SA-G winner traverses the established single-bypass path with multiple hops up to HPC_{max} in a single cycle.

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

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

where $\mathcal{N}_c$ refers to the contention count along the path of source-destination pair; note that if $\mathcal{N}_c = 0$, the blocking latency summation $\mathcal{L}_b = 0$; in such case, a packet completely bypasses the intermediate routers from source to 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 $\mathcal{N}_c$.

Based on SMART NoCs, many studies [8]–[13] have been made to upgrade performance. However, there are few studies considering contention minimization (i.e., $\mathcal{N}_c$) in SMART NoCs. To fully maximize the multi-hop bypassing benefits that SMART offers, we *firstly* achieve a nearly contention-free bypassing scheme from the perspective of routing strategy for SMART NoCs in the following sections.

III. Motivational Example

SMART NoCs show great communication advantage by building a single-cycle 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 cannot be applicable for latency-sensitive applications since it cannot reduce the number of contentions even under lightweight traffics (i.e., collective traffic). We motivate the need for flexible routing by presenting an example in Fig. 1.

As shown in Fig. 1(a), the given application is represented as a directed acyclic graph (DAG) with execution time and communication volume. The number inside the vertex is the execution time, and the number between two vertexes is the message size. Both kinds of these values are known beforehand at design time. Compared with the current XY routing of SMART, our proposed routing is flexible and performs better, since all the potential contentions are avoided in the example of Fig. 1(b). Specifically, the communication pairs (R_{15}, R_{10}) and (R_4, R_{10}) alternatively choose YX route to avoid contentions between R_{14} and R_{10}, R_4 and R_5. Especially for the communication pair (R_0, R_{10}), since the links $R_4 \rightarrow R_5$

and $R_2 \rightarrow R_6$ have been occupied, it alternatively resorts to a contention-free double-bypass path via an intermediate router R_5, and then eliminates 2 router-stage latency and blocking latency from contending packets at R_2 and R_6, at the cost of only one router-stage latency at R_5. As shown in Fig. 1(c), when using our proposed flexible routing, the schedule length of the given application only requires 14 time units. While for the current XY routing, similar to Fig. 1(c), the schedule length of the given application at least requires 23 time units.

To increase bypass utilization and reduce the end-to-end latency, the contention between source-destination pairs should be minimized based on the whole platform (i.e., Fig. 1). Here in this paper, on top of SMART [1], [2], we thus propose a routing strategy to achieve contention minimized bypassing. As a result, the bypass is fully and maximally utilized, and in turn, the schedule length of applications is further reduced.

IV. Contention Minimized Routing

Since SMART is very sensitive to contentions, the single-cycle single-bypass path from source to destination will be broken when contention occurs under the current XY routing even if alternative routes exist, thus degrading SMART performance. It is natural to achieve contention minimized bypassing from the perspective of flexible route selection.

A. Outline of the Routing Strategy

Algorithm 1: Outline of the Routing Strategy

Input: The set of unassigned source-destination pairs $\mathcal{P}$; Network state Π;
Output: Route γ_i for each pair $p_i \in \mathcal{P}$;

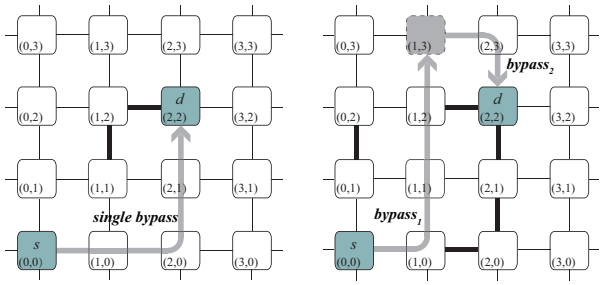
```
1 while (P ≠ φ) do
2     p_i = SelectPair(P, Π);//Section IV-B;
3     γ_i = AssignRoute(p_i, Π);//Section IV-C;
4     Update(γ_i, Π);
5     P = P − {p_i};
```

Our route assignment is conducted by SMART scheduler according to the current "network state" (actually the route allocation state) at design time. For a given DAG-modeled application (i.e., Fig. 1(a)), there are multiple source-destination pairs remaining to be assigned route. Considering the relative order of these pairs to be assigned and the way for route selection from multiple route candidates, our route allocation is split into two steps. The outline of our routing strategy is listed in Algorithm 1. Before the scheduled application begins execution, for its unassigned communication 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 "network state" is updated, and the loop is repeated if $\mathcal{P} \neq \phi$.

B. Possible Routes for Unassigned Pairs

In this section, we aim to select a pair to be assigned from $\mathcal{P}$. The way to select 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 dimension-order routing (i.e., XY, YX) are not available, to avoid contention, we instead turn to find a contention-free double-bypass path via an intermediate router to eliminate blocking latency. That is, select an appropriate intermediate router first, route to that intermediate router, and then route from the intermediate router to destination. Due to only one turn permission within each HPC_{max} quadrant to reduce SSR overhead in 2D-Mesh SMART [1], an available dimension-order route [3] is employed in each bypass phase.

Why we did not choose a contention-free path via multiple intermediate routers? The reasons are twofold. On the one hand, due to the limitation of the single directed link between PE and its attached router, a tile (binding a PE and a router together) can at most simultaneously send and receive a packet, respectively. The number of source-destination pairs is upper-bounded by K^2 in $K \times K$ 2D-Mesh SMART. Besides, due to the phenomenon of dark silicon [13]–[15], a large portion of cores cannot be utilized to guarantee safe chip temperature. The communication resources are redundant since they are not fully utilized, thus we can find a contention-free double-bypass path via an intermediate router in most cases. On the other hand, adopting multiple intermediate routers will bring significant router-stage latency that may be higher than its blocking latency when using the original conflicting route. Hence, considering multiple intermediate routers might not be helpful, and in this paper, we only consider the routing path via at most an intermediate router. To facilitate analysis, for a given source-destination pair p_i, we identify its contention-free possible routes as *direct route* and *indirect route*:



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

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

- If there exists a contention-free *single-bypass* route using dimension-order routing from source to destination according to the current network state, such route is denoted as *direct route*, highlighted in gray of Fig. 2(a).
- If there exists a contention-free *double-bypass* route via an intermediate router (i.e., $R_{(1,3)}$) from source to destination, such route is denoted as *indirect route*, highlighted in gray of Fig. 2(b).

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 network 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. In Algorithm 2, the function `GetDiPaths` returns the direct routes by choosing the source or destination router as the intermediate router; while the function `GetIdPaths` returns the indirect routes by choosing other routers as the intermediate router. Then, summarize all the distinct routes as the contention-free route candidate set S_i for each pair p_i in Line 4. Finally, we select a pair that has the least number of possible routes to be assigned route, since other remaining unassigned pairs that have a larger number of possible routes can tolerate more potential contentions. Note that, when "$|S_i| == 0$" meaning that it cannot find a(n) direct or indirect route, we let it be assigned later.

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

Input: The set of unassigned pairs $\mathcal{P}$; Network state Π;
Output: A selected pair p_i to assign route;

1 $S_1 = GetDiPaths(p_1, \Pi) \cup GetIdPaths(p_1, \Pi)$;
2 $min = |S_1|$, $min\_pair = p_1$;
3 **for** $(p_i \in \mathcal{P}/\{p_1\})$ **do**
4 $S_i = GetDiPaths(p_i, \Pi) \cup GetIdPaths(p_i, \Pi)$;
5 **if** $(|S_i| > 0$ *and* $|S_i| < min)$ **then**
6 $min = |S_i|$;
7 $min\_pair = p_i$;
8 Return min_pair;

C. Route Allocation for Selected Pair

In this section, we aim to assign an optimal route from the route candidates to the selected pair of Section IV-B. To find such a route, there are two influence factors to be considered when making a decision. Firstly, since communication resources (i.e., router, link) are limited, the long-distance route holding more 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, caused to potential direct routes of these pairs that will be assigned route in the near future, should be also minimized. 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, our route selection is divided into three cases to deal specifically with in Algorithm 3.

Case 1: Direct route set is not empty. For the selected pair p_i, we firstly derive the direct route set S_i^D by using the function `GetDiPaths` in Line 3. There are at most two contention-free direct dimension-order routes (i.e., XY, YX route). The route that causes less impact to unassigned pairs is selected if both of direct routes exist in Line 5, where the impact is estimated using the method of *Case 2*.

Case 2: Indirect route set is not empty. In this case, since direct routes are occupied by other 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 (*manhattan distance*) and impacts caused to potential direct routes of unassigned pairs. The source-destination pair from s = (1,1) to d = (3,4) in Fig. 3 illustrates how to select intermediate router. The candidates of the intermediate router are classified as multiple layers (i.e., L0, L1) based on manhattan distance. The indirect route via these intermediate routers belonging to L0 has a minimum distance from the source to 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 outer layer would be checked.

For each of indirect route candidates of each layer, to estimate the impact caused to unassigned pairs, two variables are defined in Line 1: min_factor recording the minimum impact of route candidates, and $impact\_factor$ recording the impact of the current candidate. The variable $\mathcal{D}$ in Line 7 is initialized as the manhattan distance between source and destination. Define S_{imd} as the set of intermediate routers that p_i transfers towards the destination. For the 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 network state in Line 9. To choose the route that minimizes the impacts caused to potential direct routes (derived by `GetDorPaths`) of unassigned pairs, the impact of each $path_i$ in S_i^I is estimated from Line 11 to 16. Let $S_{unassigned}$ denote the set of pairs remaining to be assigned. Then, if $path_i$ shares links with any of direct route of $p_j \in S_{unassigned}$, the variable $impact\_factor$ will be added by 1. This is because, when the potential direct route of p_j is occupied 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 accepted, 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. In this case, the function `AssignRoute` in turn tries to find an

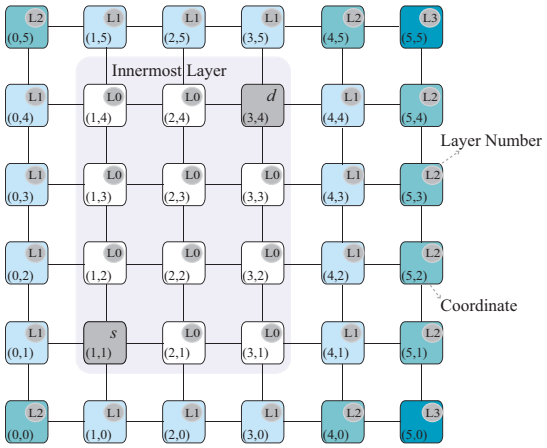


Fig. 3: Illustration of intermediate router selection for the communication pair from s = (1,1) to d = (3,4).

Algorithm 3: AssignRoute(p_i, Π)

Input: The selected pair p_i; Network state Π for allocated $K \times K$ 2D-Mesh SMART region;
Output: Route γ_i for the selected pair p_i;

1 $\gamma_i = null, min\_factor = 0, impact\_factor = 0$;
2 //Case 1: contention-free direct route exists;
3 $S_i^D = GetDiPaths(p_i, \Pi)$;
4 **if** ($|S_i^D| \neq 0$) **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** ($S_{imd} = \{R_i | ManD(R_s, R_i) + ManD(R_i, R_d) = \mathcal{D}, 0 \leq R_i.x < K, 0 \leq R_i.y < K\}$, $\mathcal{D} += 2$) **do**
9 $\quad S_i^I = GetIdPathsByImd(p_i, S_{imd}, \Pi)$;
10 $\quad$ **if** ($S_i^I \neq \phi$) **then**
11 $\quad\quad$ **for** ($path_i \in S_i^I$) **do**
12 $\quad\quad\quad$ **for** ($p_j \in S_{unassigned}$) **do**
13 $\quad\quad\quad\quad S_j^D = GetDorPaths(p_j)$;
14 $\quad\quad\quad\quad$ **for** ($path_j \in S_j^D$) **do**
15 $\quad\quad\quad\quad\quad$ **if** ($path_i \cap path_j \neq \phi$) **then**
16 $\quad\quad\quad\quad\quad\quad impact\_factor++$; break;
17 $\quad\quad\quad$ **if** ($min\_factor == 0$) **then**
18 $\quad\quad\quad\quad min\_factor = impact\_factor$;
19 $\quad\quad\quad$ **if** ($impact\_factor < min\_factor$) **then**
20 $\quad\quad\quad\quad min\_factor = impact\_factor$;
21 $\quad\quad\quad\quad \gamma_i = path_i$;
22 $\quad\quad\quad impact\_factor = 0$;
23 //Case 3: both direct and indirect route sets are empty;
24 **if** ($\gamma_i == null$) **then**
25 $\quad \gamma_i = GetXYPath(p_i)$;
26 Return γ_i;

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 minimizes the impact caused to these pairs that will be assigned route later.

Case 3: Direct and indirect route sets are empty. When increasing the traffic in SMART NoCs, the indirect route still may be not found although this event occurs with low probability. In such a case, the XY route is returned by default in Line 24, and the packets generated by the selected pair will transmit as far as they can towards the destination.

D. Analysis of Time Complexity and Feasibility

Combining algorithms 1 - 3, the time complexity is $\mathcal{O}(|\mathcal{P}|^2 \cdot K^2)$, where $|\mathcal{P}|$ is the number of communication pairs and K is the mesh size of the allocated region. Thus, our proposed algorithm can be solved in polynomial time. To achieve flexible route selection, such a proposed algorithm can be integrated into the SMART scheduler that conducts task mapping and route calculation. We can calculate the routing path based on the given vertex-to-core mapping of DAG task (i.e., Fig. 1(a)) at design time according to the remaining link utilization in the allocated region. In this way, the derived route information is delivered to the source of communication pairs through the control network in advance before they demand [1], thus the running time of the proposed routing algorithm is not the main concern. After that, the multi-hop path setup process follows the original SMART after obtaining the route information at runtime. The deadlock-free transmission is guaranteed by additional efforts. For example, the virtual channels are classified 2 classes, one for XY routing and another for YX routing. Therefore, the proposed routing which takes negligible time is feasible for SMART without hardware modification, and can be also applicable for evolved SMART NoCs (i.e., [16]–[18]). The bypass is fully utilized by employing nearly contention-free routing, and thus the performance of SMART is greatly enhanced.

V. Experimental Evaluation

A. Experimental Settings

In this section, we conduct a set of experiments to verify the performance of our proposed routing in SMART NoCs. The platform we considered in this paper adopts 4×4, 6×6 and 8×8 2D-Mesh SMART NoCs. Based on these platforms, assume the maximum bypass hop count $HPC_{max} = 9$ within a 1GHz cycle [1]. We conduct experiments with C++ programming environment, on which we achieve our proposed routing algorithm based on four classic synthetic traffic patterns [3]. For a given source coordinate (s_x, s_y) with radix-k, the destination coordinate (d_x, d_y) of $UniformRandom$ = random node, $BitComplement = (k-1-s_x, k-1-s_y)$, $Transpose = (s_y, s_x)$ and $Tornado = (s_x + \lceil \frac{k}{2} \rceil - 1 \ mod \ k, s_y)$. To compare the performance of our proposed routing algorithm, we divide our algorithm into two slightly different schemes. Let RA_1 denote the basic routing scheme which only considers resource usage minimization; while a more advanced routing scheme RA_2 considers both resource usage and potential impact minimization. Besides, these results obtained from applicable XY routing in traditional and SMART NoCs are also compared. All of the final results are averaged and normalized to the results obtained with XY routing in SMART NoCs.

B. Evaluation Results

Fig. 4 shows results of our experiments. Since our routing always considers to select a contention-free single-bypass direct route or double-bypass indirect route, conflicting pairs are isolated due to flexible routing. Thus, our routing algorithm can improve the performance significantly, compared with the current XY routing in SMART. Specifically, through the shown results in Fig. 4, there are some derived observations.

Firstly, due to the increase of single-cycle bypass utilization, the performance of SMART can obtain averagely 40.7% improvement, compared with traditional counterpart. More specifically, in SMART NoCs, packets always bypass the intermediate routers where no contention occurs along the path to the destination. In the best case, packets are transmitted to the destination via a single-cycle single-bypass path.

Secondly, in SMART NoCs, both our routing schemes RA_1 and RA_2 perform better than the current XY routing. Specifically, RA_1 and RA_2 achieve averagely 19.7% and 22.6% improvement. This is because, our routing strategy always alternatively selects a contention-free single-bypass direct route or double-bypass indirect route when potential contention occurs, and then eliminates blocking latency from higher-priority packets. Note that, although packets utilizing a double-bypass indirect route will be buffered at intermediate routers

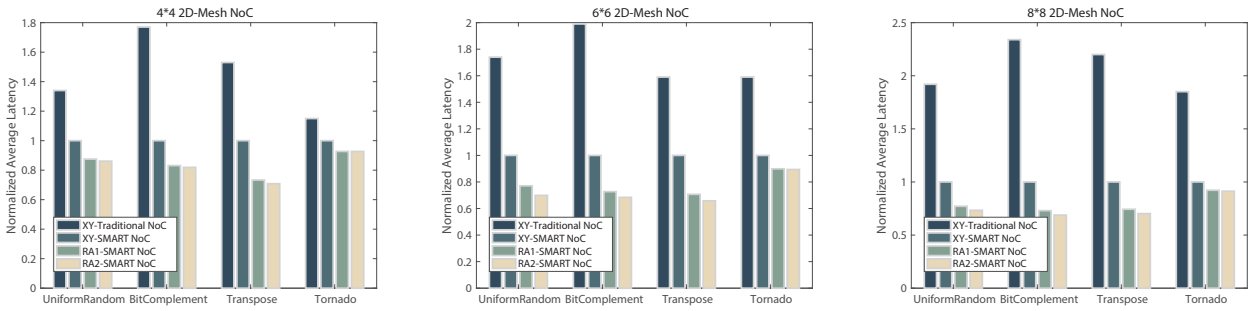


Fig. 4: Performance comparison in terms of normalized average latency (in cycles) of XY routing in traditional NoC, XY routing, a proposed basic routing RA_1 and a proposed more advanced routing RA_2 in SMART NoCs under synthetic traffics.

with a router-stage latency, such latency is almost negligible compared with blocking latency of original conflicting route.

Finally, although RA_1 and RA_2 both consider to flexibly select a contention-free route with minimum manhattan distance, they are slightly different in routing performance. Specifically, the results obtained by the more advanced routing RA_2 demonstrates 2.9% improvement than RA_1. This is because, when there are multiple contention-free indirect route candidates with the same distance, RA_2 tends to select the one that causes minimum impact to potential direct routes of unassigned pairs, such that these unassigned pairs have more probability to find a single-bypass direct route successfully, thus in turn further reducing communication latency.

VI. Conclusion and Future Work

Although SMART is promising for NoC-based many-core systems, the contention issue cannot be efficiently solved, and then bypass cannot be fully utilized, which in turn reduces the benefits that SMART offers. In this paper, we propose a system-level contention minimized routing strategy at design time: when potential contention is detected with original route, we alternatively select a contention-free single-bypass direct route or double-bypass indirect route. The contention-induced communications are isolated by using alternative routes, which in turn improves SMART performance due to the improvement of bypass utilization. In future work, to increase resource utilization, we are planning to extend the design-time route decision to a runtime one, and hide its path calculation time.

Acknowledgments

This work is partially supported by MoE AcRF Tier 2 MOE2019-T2-1-071 and Tier 1 MOE2019-T1-001-072, NTU NAP M4082282 and SUG M4082087, Singapore. Thanks to Dr. Jun Zhou, Hui Chen and the others from NTU of Singapore for their efforts to improve the equality on this paper.

References

[1] 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.

[2] 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 *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 2013, pp. 338–343.

[3] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[4] J. Kim, W. J. Dally, and D. Abts, "Flattened Butterfly: A Cost-efficient Topology for High-radix Networks," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 126–137.

[5] 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.

[6] 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.

[7] 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.

[8] L. Yang, W. Liu, P. Chen, N. Guan, and M. Li, "Task Mapping on SMART NoC: Contention Matters, not the Distance," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[9] B. K. Daya, L.-S. Peh, and A. P. Chandrakasan, "Quest for High-performance Bufferless NoCs with Single-cycle Express Paths and Self-learning Throttling," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[10] K. Duraisamy and P. P. Pande, "Performance Evaluation and Design Trade-offs for Wireless-enabled SMART NoC," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 1360–1365.

[11] 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.

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

[13] 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.

[14] 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.

[15] L. Yang, W. Liu, N. Guan, M. Li, P. Chen, and H. Edwin, "Dark Silicon-aware Hardware-software Collaborated Design for Heterogeneous Many-core Systems," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 494–499.

[16] 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.

[17] 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, p. 64, 2019.

[18] 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*. ACM, 2019, p. 5.