Proceedings Paper:
Sun, X, Dong, Y, Chen, J et al. (2020) Network-on-Chip Aware Task Mappings. In: ACA 2020: Advanced Computer Architecture. 13th Conference on Advanced Computer Architecture (ACA 2020), 13-15 Aug 2020, Kunming, China (Online). Springer Nature, pp. 135-149. ISBN: 978-981-15-8134-2. ISSN: 1865-0929.

https://doi.org/10.1007/978-981-15-8135-9_10

Network-on-chip Aware Task Mappings

Xiaole Sun[1], Yong Dong[1], Juan Chen*[1], and Zheng Wang[2]

[1] National University of Defense Technology, Changsha, China
{sunxiaole18,yongdong,juanchen}@nudt.edu.cn
[2] School of Computing, University of Leeds z.wang5@leeds.ac.uk

Abstract. Energy and power density have forced the industry to introduce many-cores where a large number of processor cores are integrated into a single chip. In such settings, the communication latency of the network on chip (NoC) could be performance bottleneck of a multi-core and many-core processor. Unfortunately, existing approaches for mapping the running tasks to the underlying hardware resources often ignore the impact of the NoC, leading to sub-optimal performance and energy efficiency. This paper presents a novel approach to allocating NoC resource among running tasks. Our approach is based on the topology partitioning of the shared routers of the NoC. We evaluate our approach by comparing it against two state-of-the-art methods using simulation. Experimental results show that our approach reduces the NoC communication latency by 5.19% and 2.99%, and the energy consumption by 17.94% and 12.68% over two competitive approaches.

Keywords: Network on Chip · Performance Optimization · Many-cores.

1 Introduction

The network-on-chip (NoC) is an essential component of multi-core processor architectures. As parallelism is the best way to utilize multi-cores, parallel workloads are now commonplace on such systems. In such a setting, the NoC is often a performance bottleneck of a multi-core system and responsible for performance slowdown of parallel workloads [1]. The NoC is also a major energy consumer of modern multi-core systems. It can consume over 28% of the total energy consumption of a multi-core processor [2], and even account for over 40% of the CPU energy consumption for multimedia applications [3]. As we are moving into a many-core era, with an increasing number of processor cores integrated into a single chip, the NoC will play an increasingly important role for performance and energy optimization of computing systems.

There have been efforts on exploring hardware and software techniques to perform performance and energy optimization, specifically targeting the NoC. For example, Chen *et al.* [4] reduce the power consumption of the NoC, by closing idle routers to without blocking communication. Other works exploit a software-centric technique to partition the router resources of the NoC among running tasks [5]. Software-based approaches have the advantage of not requiring hardware modification and can work on commercial off-the-shelf chips.

Existing work on task mapping often ignores the real-time occupation of routers of an NoC. This is a significant drawback for multi-programmed workloads, where multiple tasks or jobs use the *shared routers* concurrently. In such scenarios, existing approaches can over-subscribe the shared resources, leading to resource contention and overall performance slowdown and increased energy consumption for competing workloads.

Because of the subtle interaction among concurrently running tasks, it is important to consider the occupancy of shared routers for resource allocation. The key to minimize network congestion of the NoC is to reduce the overlap in using shared routers among concurrently running tasks. Doing so can reduce communication latency and the related energy consumption of the NoC.

This paper presents a novel software-based approach to perform power and performance optimization for the NoC. Our work dynamically allocates computing resources to match the concurrent tasks to the underlying hardware to minimize the share of routers among running tasks. We achieve this by exploiting the NoC topology to perform shared router resource partition. By always trying to assign idle routers first, our approach can reduce the resource contention, which in turn leads to faster performance and lower energy consumption among running tasks.

We evaluate our approach using the NIRGAM simulator [6]. We compare our approach against three alternative methods, including a random allocation scheme, INC [5], and CASqA [7]. Experimental results show that our approach is able to reduce the communication by 59.73%,5.19% and 2.99% and energy consumption by 53.34%, 17.94% and 12.68%, over the random scheme, INC, and CASqA, respectively.

This paper makes the following contributions:

- It is the first to leverage the topology partition theory to model the resource requirement among multiple jobs for NoC.
- It presents a novel heuristic to reduce the resource contention of shared routers among multiple jobs, using the partial topology partition theory.

2 Background

2.1 The problem of Shared routers

In this section, a simple example is offered to show the impact of Shared routers on communication latency and energy consumption. Figure 1 shows the results of mapping job1 and job2, to a 5×5 mesh NoC under the XY routing rule. Suppose job1 maps before job2. Figures 1.a and 1.b show the different results caused by two mapping method. The mapped area and communication distance for each job is the same. However, 1.a has more shared routers than 1.b, where the two blue routers in the red area are the shared routers.

L_a represents the average actual communication latency of job1 and job2 in 1.a. L_b represents it in 1.b. Accordingly, E_a and E_b respectively represent

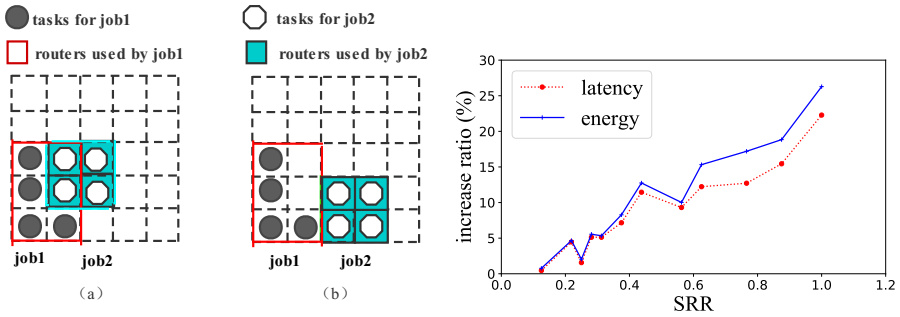

Fig. 1: The results of the two job mapping methods. (a) result with shared routers, and (b) result without shared router

Fig. 2: $r_{latency}$ and r_{energy}change with SRR

the energy consumed by all routers and their links occupied by jobs in 1.a and 1.b. Compared with 1.b, communication latency increases by 3.14% and energy consumption increases by 3.81% in 1.a. The Shared routers (1.a) can influence the communication latency and energy consumption.

2.2 Communication latency caused by Shared routers

Furthermore, we quantitatively analyze the rise in communication latency caused by the increase in Shared routers. According to the $SchedulingMethod$ [8], a packet containing n flits transfer from $s$to d, and the latency calculation formula is as follows:

$$T_{pkt\_cont}(s,d) = (T_{receive} + T_{handle} + T_{send}) \times R(s,d) + \\ T_{transfer} \times (n-1) \times MD(s,d) + T_{handle} \times M \times R(s,d)_{shared} \quad (1)$$

This includes the time it takes for $R(s,d)$ routers to receive, handle and send header flits from s to d(the first item in formula 1), the transfer time of the remaining flits at $MD(s,d)$ communication distance (the second item in formula 1), and the time it takes $R(s,d)_{Shared}$ routers to handle M packets in FIFO queues (the third item in formula 1). For Figure 1.b, the third item in formula1 is 0, because two jobs do not share the router. However, for Figure 1.a, it is not 0 because of the existence of shared routers. Therefore, the communication latency in 1.a is higher than it in 1.b.

Next, we quantitatively analyze the variation of the communication latency under different numbers of shared routers, which are generated by different mapping methods. The number of Shared routers is measured through the shared router ratio (SRR). Random mapping method is used to allocate resources for two jobs in an 8×8 NoC. We get different SRR and sort them in ascending order. The red line in the Figure2 shows how the average actual communication latency

increments Δ_L changes as SRR increases.

$$\Delta_L = \frac{L_a - L_b}{L_b} \times 100\% \tag{2}$$

It can be seen that with the increase of SSR, $Delta_L$ increases significantly. According to Formula (1), the third item will rise when SRR increase. When the SRR is 0.2, $Delta_L$= 4.46%, when SRR increases to 0.87, Δ_L= 13.44 %.

2.3 Communication energy caused by shared routers

The shared routers will not only impact communication latency but will also, increase the energy consumption of the NoC. The energy consumption of router buffering data increases because of Shared routers in 1.a. That means E_{bf} increases in Formula 3.

A message contains N packets and the size of one packet is L_{pk} bits. It transfers from processor s to processor d. The communication energy E_{com} is calculated by the Formula 3 [8].

$$\begin{aligned} E_{com}(s,d) = [(E_{xbar} + m \times E_{bf}) \times R(s,d) \times N + E_{c->r} + E_{r->r} \times \\ (R(s,d) - 1) + E_{r->c}] \times L_{pk} \times N + E_{handle} \times R(s,d) \times N \end{aligned} \tag{3}$$

It contains three terms. The first item is the energy consumed by N routers.E_{xbar} is the average energy to transfer a bit through a crossbar.E_{bf} is the average energy for buffering a bit. The second item is the energy consumed by the link. $E_{c->r}$ and $E_{r->c}$ respectively represent the transmission energy from the source core to the direct router, and from the last router to the destination core. $E_{r->r}$ represents the average energy to transfer a bit through an electrical interconnect between routers. The third item represents the energy consumption for the router to make decisions for a packet.E_{handle} is the energy of the router to handle header flits. When there are Shared routers, the energy consumed to buffer packets increases due to network contention, that is the part of E_{bf} in Formula 3.

We further quantitatively analyze the variation of the energy consumption increment Δ_E in the case of the different number of Shared routers. The blue line in Figure 2 shows how the energy increment Δ_E consumed by all the occupied routers and links changes as SRR increases.

$$\Delta_E = \frac{E_a - E_b}{E_b} \times 100\% \tag{4}$$

It can be seen that with the increase of SSR, Δ_E increases significantly. When the SRR is 0.2, Δ_E=4.66%, when SRR increases to 0.87, Δ_E=18.83%

Different mapping methods can produce different numbers of shared routers. How to design the mapping method that products as few shared routers as possible is the concern in this paper. At present, most mapping strategies usually allocate resource according to the idle cores and often ignore the real-time occupation of routers. So the situation with shared routers in Figure 1(a) is easy to

happen. Besides, it is inevitable for routers to be shared by multiple jobs because of the large number of jobs, the limited cores, and the fragmentation in allocation. To solve this problem, the mapping strategy must be reconsidered. The utilization of routers should be one of the crucial conditions for job mapping.

Here are the challenges: How to characterize the occupancy of routers on the chip? How to keep the number of Shared routers as small as possible?

Here are our solutions: Topology partition theory is used to depict routers for each job as well as the shared routers among multiple jobs. A heuristic algorithm based on topology partition is designed to reduce the number of shared routers.

3 Mapping Algorithm Based on Topology Partition

Suppose that a $N \times N$ 2D Mesh structure is designed for multi-core processor. There are already k jobs in the system, noted with JM. The $k+1$ job J_k is mapped on the NoC at t_0. Our job mapping algorithm based on topology partition is used to allocate resources for J_k. The algorithm is divided into two parts: core allocation and core mapping. Core allocation is to find a region satisfying the conditions for J_k, that is, to obtain a set of core C_{J_k}. Core mapping implements the one-to-one mapping of processes in J_k to cores in C_{J_k}.

3.1 Examples of core region selection

Here is an example to show the basic idea of region selection. There are 4 orderd jobs, J_1, J_2, J_3, J_4. The number of processes is $n_1 = 4, n_2 = 6, n_3 = 3, n_4 = 6$, respectively. They will be mapped in a 5×5 NoC. Figure 3 is the selected region for this group of jobs under our mapping method.

A bidirectional balanced mapping based on application size is used to guide the selection for a job: a small job seeks an appropriate area according to ascending order of idle routers. Instead, a large job according to descending order. For a $N \times N$ NoC, this paper takes $n_{th} = N$ as the boundary to distinguish a job, that is, if $n_{job} > n_{th}$, it is denoted as a large job, if not, it is denoted as a small job. For 5×5 NoC, n_{th} is 5.

Figure 3.a shows the region selected for J_1. Since $n_1 < 5$, select the region from the minimum idle router R_1. Start with R_1 and seek for a rectangular region with four idle cores (square is optimum). R_1 is the top left vertex. Once found, the cores in the region are got, which are c_1, c_2, c_6, c_7.3.b shows the region selected for J_2 and now J_1 is a part of JM. Since $n_2 > 5$, select the region from the maximum idle router R_{24}. Start with R_{24} and seek for a rectangular region with six idle cores (square is optimum).Then get the cores number in the region $c_{24}, c_{23}, c_{19}, c_{18}, c_{14}, c_{13}$. 3.c shows the region selected for J_3, Since $n_3 = 3$, Start from the minimum idle router and seek for a rectangular region with three idle cores (square is optimum). Then get c_3, c_4, c_8, c_9. Do the same steps for J_4 and select the region as Figure 3.d.

The use of a specific router is determined by routing rules and communication between processes. For example,the region selected for J_3 is c_3, c_4, c_8, c_9, but

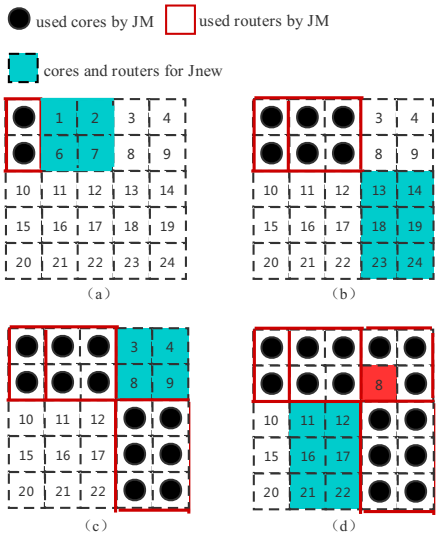

Fig. 3: The result of core allocation for J_1, J_2, J_3, J_4. (a)core allocation for J_1;(b)core allocation for J_2;(c)core allocation for J_3;(d)core allocation for J_4

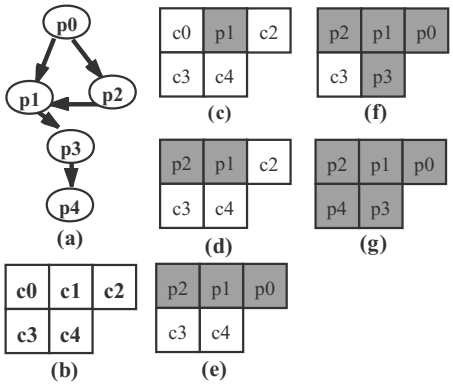

Fig. 4: Core Mapping for *job*. (a)communication graph for *job*; (b)The selected mapping region; (c)map p_1 to c_1; (d)map p_2 to c_0; (e)map p_0 to c_2; (f)map p_3 to c_4;(g)mapping results for all processes

actually J_3 only needs 3 cores. Therefore, its communication should to be considered during the core mapping, and $CoreMapping()$ in algorithm 1 is used to realize the mapping of job process to core in the selected region. Finally c_3, c_4, c_8 are selected for processes for J_3.

3.2 Single job mapping algorithm based on topology partition

The job mapping algorithm based on topology partition is shown in Algorithm 1. The input includes state information of current NoC (used processor cores $Cused$, unused processor cores C_{unused}, routing rule), job information (the number of processes n, the process ordered set based on the total communication volume P_{comm}) and threshold to distinguish jobs-n_{th}. The output consists of a set of cores assigned to J_{new}-$C_{J_{new}}$ and the corresponding relationship between the job process and core-MAP. In step 1, $FindUsedRouters$ gets the available routers R_{unused} by routing rules and the used core. R_{unused}, an ascending sequence sorted by the number, is used in $CoreAllocation$ to select a set of processor core whose region is a rectangle or close to a rectangle. Steps 2 to 4 determine the order of the traversal of $CoreAllocation$ according to the number of processes. For a large job, reverse the sequence, that finds the region from a large number of the router. The fifth step is to call the algorithm $CoreAllocation$, and select the mapping region for J_{new}. The optimal allocation is the minimum rectangular region containing n processor cores, and the output of $CoreAllocation$

Algorithm 1 Single job mapping algorithm based on topology partition

Require:
used processor cores C_{used}; unused processor cores C_{unused};
routing Rule $routing = XYrouting$;
process numbers of J_{new} n
the process ordered set based on the total communication volume
$P_{comm} =< p_0, p_1, ..., p_n >$;
threshold to distinguish jobs n_{th};
Ensure:
A set of processor cores assigned to J_{new} $C_{J_{new}}$;
The corresponding relationship between the job process and $C_{J_{new}}$
$MAP = \{p_i \leftarrow c_j | p_i \in P_{comm}, c_j \in C_{J_{new}}, 0 <= i < n, 0 <= j < n\}$.
1: $R_{unused} = FindUsedRouters(G, C_{used}, routing)$; //$R_{unused}$ is an ascending sequence sorted by the idle router number
2: **if** $n > n_{th}$) **then**
3: $R_{unused} = Reverse(R_{unused})$; //Reverse the sequence R_{unused}
4: **end if**
5: $C_{J_{new}} = CoreAllocation(n, C_{unused}, R_{unused})$;//Core allocation algorithm, get a set of processor cores
6: $MAP = CoreMapping(P_{comm}, C_{J_{new}})$;// Core Mapping algorithm, map each process to the corresponding core

is the set of processor cores in the selected region. According to the communication relationship between processes and the connection relationship between cores on NoC, $CoreMapping$ gets the specific mapping between process and core $MAP = \{p_i \leftarrow c_j | p_i \in P_{comm}, c_j \in C_{J_{new}}, 0 <= i < n, 0 <= j < n\}$.

$CoreAllocation$: As shown in Algorithm2, $C_{J_{new}}$ is obtained according to the number of J_{new}'s process n, the router sequence R_{unused}, and the idle cores C_{unused}. In step 1-14, search for the smallest rectangular area containing m cores first ($m <= n$). Through $GetRctange$ in step 3, obtain the rectangular region containing m cores with router $vertex$ as the top-left vertex. If it can be found, return the cores in the region C_{rect}; If not, use the next $vertex$ in R_{unused} to get the region. If the final returned rectangle contains less than n cores, an additional $n - m$ cores are still needed to meet the assignment requirement of J_{new}. Steps 12-22 are the steps to find them. The basic idea is to find the other $n-m$ cores closest to C_{part}(the found region with m cores).This $n-m$ idle cores are searched one by one through the while loop (step 15-21), and then added to C_{part}. Since the number of idle cores is assumed greater than or equal to n, the $n - m$ idle cores can be found when the loop is over. In step 17, select the idle core, which is closest to C_{part} and has the fewest unused neighbours around by $MinmdAndneighbor$. The neighbour here means the core with a Manhattan Distance of 1.

$CoreMapping$: In this paper, core mapping is based on the algorithm in Chou[5]. The process of mapping each process to the core is divided into two steps. First, an unmapped process is selected in p_{comm}. Then a suitable on-chip core is selected for it. But it's different between our method and Chou's in process selection: Chou defines 3 states of the process, $white$, $gray$, and $black$, two actions to switch between states:$DISCOVER$ and $FINISH$. If the neighbors(the

processes that communicate with each other are neighbors) of the process p are all *white*, $DISCOVER$ it and select all available cores on the slice and convert p to *gray*; If p's neighbour is *gray* or *black*, $FINISH$ it and select a particular core and convert p to *black*.Go back to the first node of the ordered set, change the state for nonblack process until they all *black*. For it takes two steps for a process mapping to a core, we get rid of *gray*. To reduce the communication distance among processes with high traffic, we strengthen condition for the $FINISH$ action.The basic idea is as follows: start by selecting the process with the largest traffic volume to map, and mark it *black*. If the neighbour process with the largest traffic of process p is *white*, choose p to be p_{next}, the next process that needs to be mapped. And p is $p_{neighbor}$, the core for p is $c_{neighbor}$. The process with the most traffic has already been mapped, so such a p_{next} can certainly be found. Select a specific core for p_{next} such that the distance between p_{next} and $c_{neighbor}$ is minimized. If more than one core gets the minimum distance closest to p_{next}, we choose the core that the number of whose neighbors closest to the number of nonblack neighbors of p_{next}.

Algorithm 2 *CoreAllocation*

Require:
the number of J_{new}'s process n;
the router sequence R_{unused};
the idle cores C_{unused};
Ensure:
A set of processor cores assigned to J_{new} $C_{J_{new}}$;
for m=n,n-1,n-2,...,1 **do**
  //Look for a rectangle with n points, if not, look for a rectangle with n-1 points, and so on.
  $C_{part} = \varnothing$;
  for each vertex in R_{unused} **do**
    $C_{rect} = GetRectangle(vertex, m)$;
    //Return a rectangle with m-points with *vertex* as the vertex
    if ($C_{rect}! = \varnothing$) **then**
      $C_{part} = C_{rect}$;
      return;
    end if
  end for
end for
if (the rectangle contains less than n cores **then**
  //The size of the rectangle is less than n, so still need to find an additional n-m cores
  $C_{unused} = C_{unused} - C_{part}$;
  while $m < n$ **do**
    //seek for the other n-m cores one by one. The rule is to look for other idle cores closest to C_{part}
    $c = MinmdAndneighbor(C_{unused}, C_{part})$;//select the idle core in C_{unused}, which is closest to C_{part} and has the fewest unused neighbors
    join c to C_{part};
    remove c from C_{unused};
    $m = m + 1$;
  end while
end if
$C_{J_{new}} = C_{part}$;
END

As shown in Figure4, job with five processes in (a) is mapped to the selected region shown in (b). (c),(d),(e),(f) and (g) are its mapping processes. Assume now that, based on the total communication volume, the process ordered set is $p_{comm} =< p_1, p_2, p_0, p_3, p_4 >$. We start with p_1, since it has the largest communication volume. And it is mapped to c_1 who has the most neighbors, as shown in (c), $p_1 \leftarrow c_1$ is joined to MAP. At this time, p_2, the process that have the most traffic with p_1, is chosen as p_{next}. $p_{neighbor}$ is p_1, $c_{neighbor}$ is c_1. The unmapped process that communicate with p_2 is p_0. Select the core closest to c_1 and the available neighbor is 1. c_0 and c_4 are both meet the requirements. Select the one with the smaller number, and as shown in (d),$p_2 \leftarrow c_0$ is added to MAP. Follow this step to get $p_0 \leftarrow c_2$, $p_3 \leftarrow c_4$, $p_4 \leftarrow c_3$, and the mapping result is shown in (e).

3.3 Computation complex

$CoreAllocation$:A job with n processes is allocated to $N \times N$ NOC, where $n <= N^2$. Scanning the R_{unused} list executes $|n|$ times.For each router in R_{unused}, $GetRectangle$ and while loop (in line 15) executes $|n|$ times.So the total run time for $CoreAllocation$ has a complexity of $O(n^2)$.

$CoreMapping$:An ACG for a job with n processes and e edges is mapped to the selected region. The total run time of our algorithm has a complexity of $O(n^2 + e)$ the same with Chou[5].

4 Experimental Results

4.1 Experimental Platform

Simulation Environment In this paper, NIRGAM[6] is used to simulate an 8×8 mesh NoC. Table 1 is the configuration of NIRGAM in this experiment:

Table 1: Configuration for NIRGAM platform

Parameter name	values	Description
TOPOLOGY	MESH	2-d mesh topology
NUM ROWS	8	8 rows
NUM COLS	8	8 columns
RT ALGO	XY	XY routing algorithm
NUM BUFS	16	The number of buffers in input channel FIFO is 16
CLK FREQ	1GHz	Clock frequency is 1GHz
PKT SIZE	32	Packet size is 32bytes
FLIT INTERVAL	1	Interval between succesive flits is 1 clock

Job sequence generation Several sets of jobs with 4 to 16 tasks are generated using the TGFF[9]. It's 4 to 8 for small-scale-job, 9 to 16 for large-scale-job. Adjusting the proportion of large jobs to 0%, 25%, 50%,, 75% and 100%,we get 5 sets of jobs. An arrival sequence is generated in each set. These sequences are used to simulate the order in which the OS allocates resources for actual jobs.NPB[10] traces with 4 and 8(9 for BT and SP) and 16 processes are get by HPC-NetSim[11]. An arrival sequence is generated for NPB.

The mapping algorithms we compare include random, INC[5],CASqA[7] and the Job mapping algorithm based on topology partition (JMATD) proposed in this paper. FT2000+ under the condition of not binding cores, allocates resource for jobs in a random way by default. INC is a convex region mapping algorithm, which can reduce communication energy consumption and improve the application's performance. CASqA has multiple mapping levels by adjusting threshold(α), where set $\alpha = 0$ to improve performance and reduce communication energy consumption and latency.

4.2 Experimental Result

The number of shared routers The number of Shared routers produced by the four algorithms is different. In order to compare the differences, the number of shared routers in the five job sequences is statistically analyzed. (a)(b)(c)(d)(e)(f) in Figure 5 respectively reflect the change in the number of Shared routers per job sequence during the mapping process. For random, the number of Shared routers is the largest due to the overlap of jobs. JMATD reduces the number of Shared routers by partitioning the topology when a single job is mapped. For each job sequence, JMATD has a good optimization effect. In (d), the number of Shared routers in random, INC and CASqA is 9.56x, 3.48x and 2.10x higher than that in JMATD, respectively. For both INC and CASqA, due to the continuous convex mapping region, the same effect can be achieved with JMATD for the job sequences with more fragment, as shown in (b). Figure 7 shows the average number of Shared routers in the mapping results for each group of job sequences. On average, the number of Shared routers generated by random, INC and CASqA is 5.78x, 1.25x and 0.67x higher than that of JMATD.

Communication power for jobs JMATD is effective in reducing the number of Shared routers. We measured the communication power curve changing under different mapping algorithms for each set of jobs, and the results are shown in Figure 6.(a)(b)(c)(d)(e)(f) represent different job sequences. Although the power curve of each mapping method fluctuates somewhat, JMATD method is relatively low compared with other methods on the whole.Figure 8 shows the average power consumption in the job mapping process. On average, Compared with random, INC, and CASqA, the communication energy consumption is decreased by 53.34%, 17.94%, 12.68%, respectively. Run time is assumed to be the same for a job in different mapping method. Therefore, the communication energy consumption of the job sequence is proportional to the power consumption.

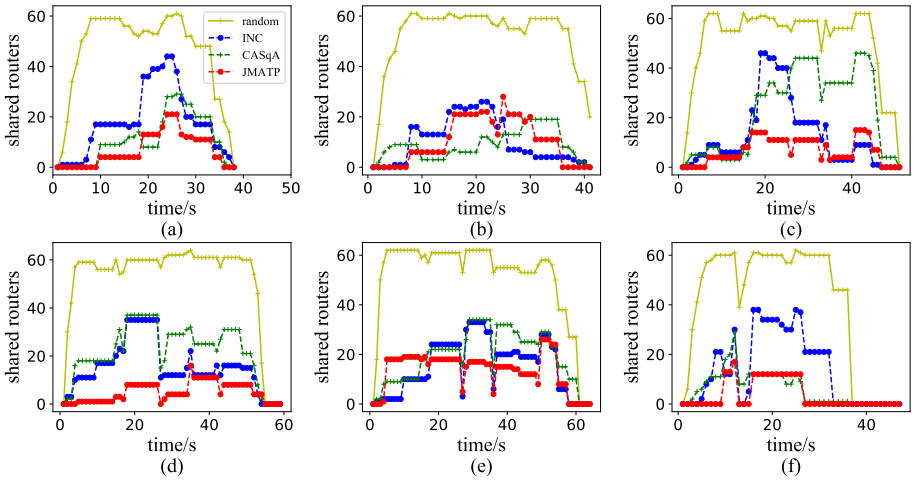

Fig. 5: Changes in the number of Shared routers per job sequence, (lower is better);(a)l%=0%;(b)l%=25%;(c)l%=50%;(d)l%=75%;(e)l%=100%;(f)NPB

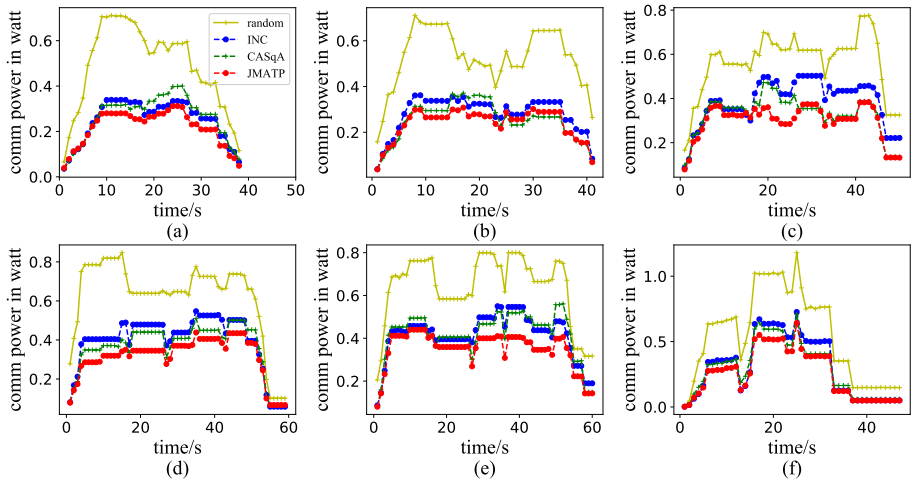

Fig. 6: Changes in communication power during each job sequence mapping, (lower is better);(a)l%=0%;(b)l%=25%;(c)l%=50%;(d)l%=75%;(e)l%=100%;(f)NPB

Communication latency for jobs The communication latency of a job is an important factor affecting performance.To compare the effect of JMATP in reducing communication latency,the average latency of each job is calculated.As shown in Figure 9, the data is normalized.According to the average results of the five job sets, the latency of random, INC, CASqA method is 59.73%,5.19%,2.99% higher than that of JMATP, respectively.

4.3 Discussion

The experiment shows that the number of Shared routers has an effect on the communication power consumption of the system, and the trend in Figure 8 is consistent with that in Figure 7. However, when l%=25%, the router is shared equally in INC, CASqA and JMATP, but the communication power is quite different.JMATP has lower power. We compared the communication distances of the jobs in each algorithm. In this article, weighted Manhattan distance(WMD, the sum product of MD and the corresponding weight of communicating processes)[12] is a metrics of the quality of the job mapping. Figure10 is the WMD comparison of 4 mapping methods adopted for 6 sets of jobs. From the perspective of single job mapping, compared with other algorithms, JMATP can effectively reduce the communication distance between job processes by 63.82%, 22.39% and 19.07% respectively for random, INC and CASqA in WMD. JMATP not only reduces the number of Shared routers but also reduces the communication distance of jobs.

Formula 1 indicates that the latency is related to the router and the communication distance of the process where the contention occurs. Since the packet transmission is concurrent, the latency is not wholly positive related to the relationship between the two. It means that the marginal benefits of optimizing communication latency from reducing communication distance are not high. Therefore, it is necessary to optimize shared routers while reducing the communication distance.

The influencing factors of communication latency include external congestion (router and link contend by different jobs) and internal congestion (router and link contend by packets of the same job). Memory[13] and disks[14] also impact the communication. Because of the interaction of these factors, communication latency optimization is not significant compared to communication power. It inspires us to work on what we're going to do next: How to reduce resource contention between job processes. How job communication characterizations[15] affect communication latency.

5 Related work

The performance of NoC is closely related to network congestion, which not only increases network latency but also affects the communication power consumption. That is why there are many works to diminish network congestion from software and hardware aspects. Ebragimi[16] optimizes communication from routing algorithms to reduce network conflicts. Jiang[17] proposed a new switching

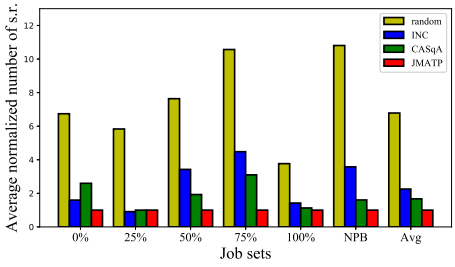

Fig. 7: Average number of Shared routers for jobs(lower is better)

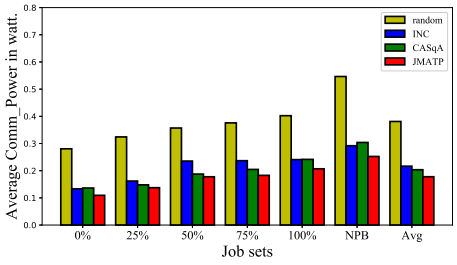

Fig. 8: Average communication power for jobs (lower is better)

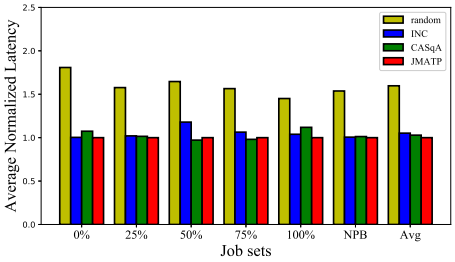

Fig. 9: Average latency for jobs (lower is better)

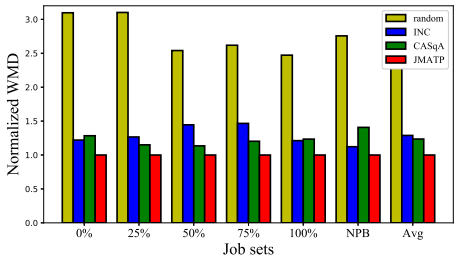

Fig. 10: WMD comparison for jobs (lower is better)

mechanism to reduce network latency. Based on the STT-RAM router, Yang[18] reduces communication latency by calculating the contended flit.

Job mapping is one of the effective ways to reduce network conflicts. Two types of congestion can be defined during dynamic application mapping: external and internal congestion. External congestion occurs when a network channel is competing with packets from different applications; internal congestion is related to packets from the same application.

To decrease the external congestion probability by mapping algorithm Chou[5] proposes an incremental (INC) approach. They first select the near convex region to reduce communication links and try to keep both the selected region and remaining nodes contiguous, then allocation node according to the total communication volume inside the selected region. Das[19] proposes new mapping policies to improve system performance by reducing inter-application interference in the on-chip network and memory controllers. Cores are clustered into a subnetwork. Fattah in [12], proposed a SHiC algorithm to guide how to find the optimum first node among all the available nodes for the run-time application. Then, in [7] they proposed a run-time mapping algorithm, CASqA. In this algorithm, the contiguousness of the allocated processors can be adjusted in a fine-grained fashion according to α. Zhu[20] proposed an efficient heuristic-based algorithm to balance minimized on-chip latency in multi-application mapping.

Internal congestion can also be reduced by the mapping algorithm. In [8], a heuristic algorithm, unified priority-based scheduling (UPS), is put forward to effectively solve the contention problem in polynomial time by assigning priorities to messages. Once an instruction is waiting for the data from other PE, the extra delay caused by NoC congestion postpone the instruction issue and decrease the performance. An[21] proposed C-Map for the delay of the instructions existing in CGRA, which improves the effectiveness of CGRA mapping in the perspective of reducing network congestion and enhancing the continuity of the data-flow.

6 Conclusion

This paper analyzes the influence of Shared routers among multiple jobs on communication latency and energy consumption. When the number of Shared routers increases significantly, it affects the communication latency of a single job and the energy consumption of NoC. To reduce this impact, this paper proposes a task mapping method based on topology partition. When allocating resources for a single job, cores connected to an idle router are considered first to minimize the number of shared routers between multiple jobs. NIRGAM Simulator is used to compare the mapping method proposed in this paper and three other typical ones (including random, INC, and CASqA). Communication latency and energy consumption of the jobs under each mapping method are get based on an 8×8 NoC. The communication performance is improved to 59.73%,5.19%,2.99% and energy consumption is decreased by 53.34%, 17.94%, 12.68%, respectively. Shared routers exist not only between jobs but also between processes in a job. Next, We focus on how to reduce Shared routers in the same application . We also consider the impact of memory and disks on communication.

References

1. 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.
2. A. B. Kahng, B. Li, L. Peh, and K. Samadi, "Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration," pp. 423–428, 2009.
3. C. Wu, C. Deng, L. Liu, J. Han, J. Chen, S. Yin, and S. Wei, "An efficient application mapping approach for the co-optimization of reliability, energy, and performance in reconfigurable noc architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1264–1277, 2015.
4. L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of noc routers," pp. 378–389, 2015.
5. C. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 10, pp. 1866–1879, 2008.

6. NIRGAM, "A simulator for noc interconnect routing and application modeling," https://nirgam.ecs.soton.ac.uk/.
7. M. Fattah, P. Liljeberg, J. Plosila, and H. Tenhunen, "Adjustable contiguity of run-time task allocation in networked many-core systems," pp. 349–354, 2014.
8. L. Yang, W. Liu, W. Jiang, M. Li, J. Yi, and E. H. M. Sha, "Application mapping and scheduling for network-on-chip-based multiprocessor system-on-chip with fine-grain communication optimization," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, no. 10, pp. 3027–3040, 2016.
9. "Tgff," http://ziyang.eecs.umich.edu/~dickrp/projects/tgff/index.html.
10. "Nas parallel benchmarks," https://www.nas.nasa.gov/publications/npb.html.
11. W. Zhou, J. Chen, C. Cui, Q. Wang, D. Dong, and Y. Tang, "Detailed and clock-driven simulation for hpc interconnection network," *Frontiers of Computer Science in China*, vol. 10, no. 5, pp. 797–811, 2016.
12. M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–6.
13. F. Wu, J. Chen, Y. Dong, W. Zheng, X. Pan, Y. Yuan, Z. Ou, and Y. Sun, "A holistic energy-efficient approach for a processor-memory system," *Tsinghua Science Technology*, vol. 24, no. 4, pp. 468–483, 2019.
14. D. Yong, C. Juan, T. Yuhua, W. Junjie, W. Huiquan, and Z. Enqiang, "Lazy scheduling based disk energy optimization method," *Tsinghua Science and Technology*, vol. 25, no. 2, pp. 203–216, 2019.
15. J. Chen, W. Zhou, Y. Dong, Z. Wang, C. Cui, W. U. Feihao, E. Zhou, and Y. Tang, "Analyzing time-dimension communication characterizations for representative scientific applications on supercomputer systems," *Frontiers of Computer Science in China*, vol. 13, no. 6, pp. 1228–1242, 2019.
16. M. Ebrahimi, M. Daneshtalab, and F. Farahnakian, "Haraq: Congestion-aware learning model for highly adaptive routing algorithm in on-chip networks," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, May 2012, pp. 19–26.
17. G. Jiang, Z. Li, F. Wang, and S. Wei, "A low-latency and low-power hybrid scheme for on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 4, pp. 664–677, April 2015.
18. L. Yang, W. Liu, N. Guan, and N. Dutt, "Optimal application mapping and scheduling for network-on-chips with computation in stt-ram based router," *IEEE Transactions on Computers*, vol. 68, no. 8, pp. 1174–1189, 2019.
19. R. Das, R. Ausavarungnirun, O. Mutlu, A. Kumar, and M. Azimi, "Application-to-core mapping policies to reduce memory system interference in multi-core systems," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 107–118.
20. D. Zhu, L. Chen, S. Yue, T. M. Pinkston, and M. Pedram, "Balancing on-chip network latency in multi-application mapping for chip-multiprocessors," pp. 872–881, 2014.
21. S. An, M. Zhang, X. Ye, D. Wang, H. Zhang, D. Fan, and Z. Tang, "C-map: Improving the effectiveness of mapping method for cgra by reducing noc congestion," in *high performance computing and communications;*, Aug 2019, pp. 321–328.