Mapping of Pipeline Flow Chart onto NoC Architecture

Haofan Yang, Ruijin Xiao, Liang Liu, Ming-e Jing, Zhiyi Yu, Xiaoyang Zeng, Dan Zhou
State Key Laboratory of ASIC and System, Fudan University, Shanghai 200433, China
School of Computer Science, Fudan University
mejing@fudan.edu.cn

Abstract - Among current outstanding research problems in NoC Design, mapping application onto NoC is one of the core issues to be explored. In this paper, we propose two methods for mapping a pipeline flow chart onto mesh-based NoC system: Communication Length Concerned (CLC) method and Space Restricted (SR) method after a simple pre-process. The former significantly reduces the latency and energy consumption in inter-core communication while the latter method provides a trade-off for run-time mapping based on the former one. Results show that, compared with normal mapping, the CLC method reduces about 43.3% and 28.6% in communication latency and energy consumption, respectively; while the SR method reduces about 37.6% communication latency and 24% energy consumption. The two methods discussed in this paper could be used as guidance for mapping pipeline flow charts on homogenous mesh-based NoC architectures.

1. Introduction

Recently, network-on-chip (NoC) emerges as a promising alternative to classical on-chip communication architectures with hierarchical bus structure in multi-core systems [1]. This novel communication architecture not only brings better performance and lower power consumption, but also enhances the scalability and flexibility of systems. Among different types of NoC systems, mesh-based network is a favorite one, for its flatness brings up many merits, such as simple routing strategy and better scalability [2-3]. In this paper, our topics focus on mesh-based NoC system.

Besides the consideration of how to design outstanding hardware architecture for NoC systems, proper methods for mapping target application tasks onto NoC systems are of equal importance to ensure a well performance. A good mapping algorithm can consummate application running on NoC systems without any hardware cost. So it is useful to give an effective mapping algorithm for special system architecture and application [1-3].

Generally speaking, the job of mapping an application onto NoC system consists of two processes: partitioning and mapping. The partitioning process is dividing the target application into basic task units, and a basic unit can be implemented by a processor unit (PE) in NoC system. The second process is mapping the obtained basic task units into NoC system.

The methods presented in this paper mainly focus on the mapping process; therefore, further details in the partitioning process will not be discussed in this paper. Among related works, [2] proposed a branch and bound algorithm of mapping a given set of IP cores onto a regular NoC architecture to minimize the total communication energy. [3] targets at real-time applications which are dynamically mapped onto embedded MPSoCs, and resources connected to the NoC have multiple voltage levels. Being different from above works, we focus on the mapping of pipeline flow chart on homogenous mesh-based NoC architectures.

2. Normal Rules in Pre-process

Although the first step is not the emphasis of this paper, we do have certain general rules as well as presumptions for this part of mapping work.

Discussion will be focused on how to map divided tasks of target application onto NoC system with the concern of high system performance.

It is obvious that different task arrangements on PEs will result in different performance, despite the same hardware architecture and task allotment. The effects are mainly in two aspects: latency and energy consumption in communication. They normally could be represented by two key characteristics in communication pattern: critical path length (CPL) and total communication length (TCL). When packet travels in network, it takes much time and energy to pass routers than links between adjacent routers [7]. In mesh type network, routers are placed in every node which connects a PE; number of routers that certain packet will go through in communication path, therefore, becomes the reasonable indicator of communication length. Since the transmission time is approximately in proportion of its transmission length; the CPL plays a key role in deciding the communication latency in system. The TCL, on the other hand, dominates the total energy consumption in system for it represents the total routers that packets would pass. Our evaluation on mapping algorithm will be mainly focused on CPL and TCL in system.
3. Basic Patterns in Flow Chart

Though flow chart varies in a myriad of ways, they do have some characteristics in common. Generally speaking, they could be viewed as the combination of FIVE basic patterns. They are respectively parallel, serial, convergent, radial and loop-like patterns (An example is shown in Fig. 2). Especially in the realm of communication and multimedia technology, such as applications of 3780 points FFT [4], transmitter [5], Reed Solomon Decoder [6] and so on, this disassembling analysis works well. It is important to see the whole chart in basic patterns; because it provides a general way to handle a variety of flow charts.

Noticing that a normal pipeline flow chart starts with certain blocks and ends with some others, so the distribution of blocks within a flow chart is not completely random. In fact, the general trend of information stream in pipeline flow chart could be indicated if we drew a virtual axis from starting task blocks to ending blocks. Contents within the flow chart could be seen as basic modules one by one aligned in this line. This is a key characteristic of flow chart in mapping, because it not only allows us to have a certain order to process our mapping but also gives axil symmetry to the topological layout of task arrangement on NoC system.

Since tasks in one flow chart are associated, it’s common that parallel patterns in flow chart are accompanied by one of radial and convergent pattern. This is another important feature in flow chart task, and it allows us to apply a simple but effect method to allocate parallel pattern along with radial and convergent patterns. Furthermore, if two parallel patterns lie in adjacent stages in flow chart, it’s common that the parallel pattern has connection with both radial and convergent point (task block) first. Then, place every task in line.

4. CLC and SR mapping algorithms

Having seen the regularities of flow chart from the analysis above, we now introduce general ideas of our mapping methods:

A. The whole mapping process is serial, in the order of the virtual line in flow chart.

B. When dealing with the serial patterns in flow chart, just place those tasks sequentially in line.

C. When dealing with the parallel patterns, decide the position of related radial and convergent point (task block) first. Then, place every task in the parallel pattern that has communication need with the target point sequentially according to communication weight. The rule for the spots selection is that always try to fill available nearest (distance of two points \( d(x_i, y_j) \) and \( d(x_k, y_l) \) in a mesh structure could be expressed by \( |x_i - x_k| + |y_j - y_l| \) spot around the target point first. As for the situation that the parallel pattern has connection with both radial and convergent point, put those two points close to each other first.

D. When dealing with the loop-like patterns, since the formation of loop is simplified in pre-process by rules illustrated before, just unfold the loop and place tasks in a compact and regular pattern.

E. When the placement couldn’t fit in the available space in NoC system, try to adjust the placement by putting outsider task blocks into available spots. The rules of sequence is that always choose outsider which has the heaviest communication weight first, and move blocks to the spot that might bring the minimal effects among all available areas.

General Instruction of our methods.

Both the Communication Length Concerned method and the Space Restricted method are conducted by general rules presented above, but they are designed to fit in different hardware resource situation.

The general ideas discussed before are only informative. In order to demonstrate our methods in a clear and practical way, we will pick up a typical example to illustrate detail issues. The flow chart example is shown in Fig. 4, which consists of four steps. Among those stages, the first and third one includes heavy computation, such as matrix operation, which might be speeded up by dividing them into parallel operation structure. Noticing that stage 2 is an iterative computation process, we could therefore follow the general pre-process rules introduced in section 2 to eliminate this loop. Several serial task blocks will substitute the iterative computation part.

As for the last stage, it doesn’t require heavy computation, so we can just leave it alone. The specific number of blocks in each stage will be judged by the situation of computation load, in order to guarantee that the allocation is reasonable.

In this example, we assume that stage 1 consists of 5 blocks, stage 2 consists of 5 blocks and stage 3 consists of 10 blocks. The demo of following methods will be based on this specific flow chart; it is actually a real example we met when dealing with RS decoder mapping. In order to facilitate explanation, we name each block in following way: blocks in stage 1 are from \( \#0 \) to \( \#4 \); blocks in stage 2 are from \( \#5 \) to \( \#9 \); blocks in stage 3 are from \( \#10 \) to \( \#19 \); block in stage 4 is \( \#20 \); in addition, each stage is colored differently.

Although energy consumption and time delay vary dramatically in different NoC systems, to get an intuitive picture of the mapping result, we assume that both time and energy consumption in communication gather in routers; and each packet passes through a router would cost approximately 15ns and 3mW. It is important to notice that the relative value is the one of real significance, rather than the absolute value. Details results of each method are shown in Fig. 6 and Table 1.

4.1 Normal Method

When mapping a pipeline application flow chart, an easy and intuitive way is described in Method 1.

Method 1. Normal method.

1. Get to know the total number \( N \) of task blocks in the flow chart.
2. Decide a square area with the length of \( n \), which satisfies that \( n = \text{ceil}(\sqrt{N}) \). In this equation, function \( \text{ceil}(x) \) round the elements \( x \) to the nearest larger integer.
3. Place each task blocks one after one, row according to their order.

4.2 Communication Length Concerned Method

Normal method described above has only considered the regularity of placement, but the communication cost and latency is very large as shown in Fig. 6. For example, when block \#4 in stage 1 sends a packet to block \#5 in stage 2, packet needs to pass 5 routers. In order to get a placement of task blocks with a better communication situation, Communication Length Concerned (CLC) method tries to explore the way to minimize CPL and TCL mentioned above.

However, finding the optimal placement is a NP-hard problem, it could only be gained by method of exhaustion, which has no practical significance. Thus CLC method heuristically provides a practical approach towards the optimal result.

To make problem simple and focused, we assume that the resource of available PE is abundant, so there isn’t any space restriction in CLC method. CLC method would be described in Method 2.
5. Experiment Result

In order to examine CLC and SR methods, our example flow chart has been tested with different blocks distribution in each stage. Our ten examples are taken from the uniform integer distribution of [2-6]-[2-6]-[8-11]-[1]. These tests are conducted under programs written in perl, and running on PC hardware with Windows OS environment. It only takes less than 3 seconds to generate all results of a single example. The results are shown in Fig. 7 and Fig. 8.

As shown in Fig. 7, the communication performance of the CLC method is the best for all the examples, while SR method which emphasizes on regulation gives relatively poor performance. According to our statistic data which is also shown in Fig. 7 and Fig. 8, the CLC method reduces about 43.3% and 28.6% in latency and energy consumption, respectively; while the SR method reduces about 37.6% latency and 24% energy consumption.

When we analyze latency of different application mapping, congestion is not taken into account, for this is another open problem in mapping research. But this does not prevent us carrying out a valid comparison.

Method 3 Space Restricted method.

4.3 Space Restricted Method

We could see from Table 1, though placements generated by CLC method possess a well feature in minimizing total communication length, its distribution is often too dispersive, as shown in Fig. 6; therefore, it is not feasible for system with multiple application. This is because the PE resource in a NoC system is limited; in run-time mapping situation, irregular available space left would impede incoming application acquiring high communication performance. Communication length, therefore, might have to be sacrificed in order to make the placement fit the available space in system. Based on the result of CLC method, Space Restricted (SR) method concentrates on this imperfect situation, and offers solution that tries to bring minimal negative effects on total communication length. SR method would be described in Method 3.

### Table 1. Detail information of each result.

<table>
<thead>
<tr>
<th>Latency (ms)</th>
<th>Normal</th>
<th>CLC</th>
<th>SRI(6&lt;4)</th>
<th>SRI(7&lt;5)</th>
<th>SRI(8&lt;6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption (mW)</td>
<td>267</td>
<td>153</td>
<td>174</td>
<td>165</td>
<td>159</td>
</tr>
</tbody>
</table>

Fig. 6. Sample results of each method.

The results of SR method would vary according to the shape of restricted area. For this example with 21 task blocks, the available regular mesh region might be 6×4, 7×3 or 8×3. These results also provide flexible choices for system design. From Table 1 and Fig. 6, we could see that these results offers approximately the same performance compared to CLC method, but with a much more regular space.

6. Conclusion

In this paper, we proposed two methods for mapping the altered flow chart onto mesh-based NoC system. Compared with normal mapping, our CLC method significantly reduces the latency and energy consumption in inter-core communication while SR method offers a good trade-off between performance and regulation. In order to get a complete set of mapping methods that could directly mapping application onto NoC system, there’re still many issues remained to be studied. As for the extension work in the future, we plan to focus on the partitioning process work: deciding how many tasks we shall divide the target application into and assigning detail task to individual PE with the concern of even task distribution and high system performance.

Reference:


Fig. 7. Power consumption of 10 examples.

Fig. 8. Latency of 10 examples.