# C-MAP: Improving the Effectiveness of Mapping Method for CGRA by Reducing NoC Congestion

Shuqian An*†, Mingzhe Zhang†, Xiaochun Ye†, Da Wang†, Hao Zhang†, Dongrui Fan†* and Zhimin Tang†*
*University of Chinese Academy of Sciences, Beijing, China
†State Key Laboratory of Computer Architecture, ICT, CAS, Beijing, China
{anshuqian, zhangmingzhe, yexiaochun, wangda, zhanghao, fandr, tang}@ict.ac.cn

***Abstract*—The Coarse-Grained Reconfigurable Architecture (CGRA) is considered as one of the most potential candidates for big data applications, which provides significant throughput improvement and high energy efficiency. Unlike the *dynamic issue superscalar* method in conventional processors, the CGRA architecture uses the *static placement dynamic issue (SPDI)* execution method in which the compiler decides how to map the instructions onto the distributed processing elements (PEs) and the PEs executes one instruction when the required data is ready. Since the dataflow of the program is determined in the logical view, an improper mapping of instructions may leads to more network congestion and hurts the performance. Furthermore, the exploration for most optimized mapping in CGRA is proved to a NPC problem and can hardly be achieved in limited time. In this paper, we propose a novel mapping algorithm named *Congestion-MAP (C-MAP)*. C-Map improves the effectiveness of CGRA mapping in the perspective of reducing network congestion and enhancing the continuity of the data-flow. Furthermore, C-Map also accelerates the mapping optimization for CGRA by using network analysis method, which supports the fast comparison of mapping plan and parallel exploration. Additionally, with C-Map, we also analyze the impact of several key considerations in CGRA instruction mapping, such as NoC workload reduction and workload balance. The experiment result shows that C-Map improves the performance by $2.2\times$ as a geometric mean.**

***Keywords*-Instruction mapping, CGRA architecture, SPDI, performance, network**

## I. Introduction

In the Big Data Era, the processor is required to support various applications with high performance and energy efficiency [1]. Therefore, the Coarse-Grained Reconfigurable Architecture (CGRA) is considered as a potential candidate for both computation and data intensive applications, since it provides high throughput and flexibility. A series of CGRA structures have been proposed in the past years [2]–[4].

The typical structure of CGRA is as shown in Figure 1. In CGRA, a 2-D mesh routing network connects the Processing Elements (PE) and transmits the values among the PEs. Each PE contains the entire resources for computing, including ALU, the register files and the caches for instructions and data. The programs for CGRA organize the instructions in form of data-flow graph. The CGRA schedules the instructions with *static placement dynamic issue* (SPDI) model which relies the compiler to select PE for each instruction and allows the direct communication among the instructions according to the data-flow graph. The PE executes one instruction when its necessary input is ready and then sends the result to the next instruction. Since the CGRA works as a software pipeline, the network congestion significantly affects the performance. Therefore, it is important to optimize the placement of instructions and carefully organize the data movement.

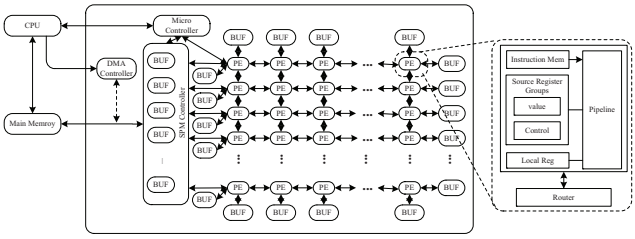



Figure 1. An example of CGRA architecture.

However, previous work on the mapping method cannot fully meet the demands of actual CGRA applications. First, most of the previous works focus on solving the mapping problem with heuristic algorithm, which are only applicable for small scale program. Besides, the previous mapping methods are based on specific CGRA features and can hardly utilize the new feature, such as *Static-Placement, Dynamic-Issue (SPDI)*. Additionally, previous works pay less attention on several key factors in the mapping, such as *workload balance*, *NoC workload reduction* and *inter-PE pipelining*.

Fortunately, the relevant research of network provides a new choice for CGRA mapping optimization. Previous work shows that the upper bound of packet transmission is determined by the *network diameter* and the congestion on transmission path [5], [6]. Based on such observations, Zhang *et al.* propose a NoC design space exploration model named COMRANCE [7], which uses congestion matrix to estimate the worst-case transmission delay. Similarly, for a given network, we can evaluate the placement of instructions by comparing the caused worst-case congestion.

In this paper, we propose a novel mapping method for instruction placement in CGRA, which is name Congestion-

Mingzhe Zhang and Shuqian An have equal contribution. Xiaochun Ye is the corresponding author.

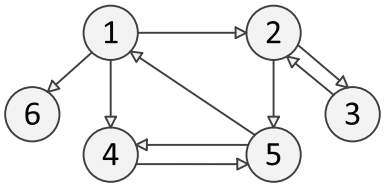



Figure 2. The example network with congestion=13 and dilation=4.

MAP (C-MAP). C-MAP optimizes the instruction mapping in the perspective of reducing NoC congestion. We show that, since the NoC acts as an important role in CGRA, the optimization of instruction placement by reducing NoC congestion can provide significant performance improvement in an intuitive way. Furthermore, C-MAP is easy to add the new factors in the mapping process, such as workload balance. Additionally, since C-MAP focus on the optimization of data stream distribution, the new features of the computing elements will not affect the effectiveness of C-MAP.

The experiment shows that, with proper configuration, C-MAP provides performance improvement by $2.2\times$ as a geometric mean and up to $3.96\times$ in specific application. Furthermore, we also provide the sensitivity study for the different factors and key parameters in the mapping process.

The rest of this paper is organized as follow. In Section II, we introduce the estimation model for upper packet transmission bound. In Section III, we present several important consideration for instruction mapping in CGRA and the possibility of optimizing the mapping in the perspective of reducing congestion. After that, Section IV introduces our proposed mapping method. Then the methodology and result analysis are presented in Section V and VI respectively. Finally, the related works are introduced in Section VII, followed by the conclusion in Section VIII.

## II. Background

Packet routing network, which transmits the message in granularity of packet, is widely used in various systems, scaling from multi-core processor to Internet. In general, packet routing network uses *store-and-forward* routing method, that the router stores the received packets until the switching components and the receiver of next hop are both available.

When designing a network, providing guaranteed QoS is one of the most important considerations, which indicates the transmission delay in the worst case. To estimate the up bound of the transmission delay, two important concepts are defined as follow: the first one is *dilation (d)*, which refers to *the longest acyclic path of all packets*, the other one is *congestion (c)*, which refers to *the largest number of packets which traverse same edge*. For example, a network is shown in Figure 2, which contains 6 routers and several unidirectional links. According to the definition, the *dilation* and the *congestion* of the network are 4 and 13 in respective.

Previous work proves that, for a given network G with dilation d and congestion c, there always exists a strategy of determine transmission priority for packet delivery in O(c+d) steps [5]. Furthermore, the strategy can be implemented by using the algorithm proposed in Ref [6]. Therefore, it is possible to estimate the performance of a network by comparing its congestion and dilation in the worst-case.

Based on such observations, a network analysis method named COMRANCE is proposed, which supports fast exploration for NoC design space [7]. COMRANCE abstracts the network as a *congestion matrix*, which describes the component occupation of all possible communications. As shown in Figure 3(a), a simple mesh NoC is presented, which connects all nodes with bi-directional links and each direction allows one packet to pass per cycle. COMRANCE represents all links with set $L$, and each element $l \in L$ denotes a specific link in the network. Suppose set $R$ denotes all possible communication requests in the network and each element $r \in R$ denotes a specific communication request. The *congestion matrix* is defined as $C = L^T \times R$, where each column (named in the format of $\langle x, y \rangle$) refers to a path from node x to node y, and each row refers to the occupation of one link. The value of each element in $C$ is assigned as

$$c(l,r) = \begin{cases} 1, & \text{request r occupies link l} \\ 0, & \text{request r doesn't occupy link l.} \end{cases} \tag{1}$$

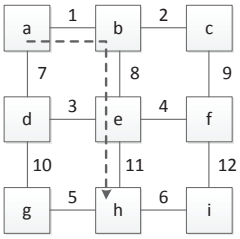
Since most of the practical NoC architectures use oblivious routing algorithms [8], the packets are transmitted in the deterministic path. Therefore, the components that one packet will pass are directly determined. For instance, the NoC shown in Figure 3(a) uses the *static X-Y* routing algorithm, which routes the packet in x- and y- dimension orderly. When node a sends a packet to node h, the path includes link 1, 8 and 11 orderly. Accordingly, the column corresponds to the path will be $(1,0,0,0,0,0,0,1,0,0,1,0)^T$ (as shown in the dashed box in Figure 3(b)).

As mentioned, the upper bound of the packet transmission delay can be estimated with the network *dilation* and *congestion*. Therefore, COMRANCE compares the different NoCs by comparing the sum of *dilation* and *congestion*. Additionally, the *congestion* estimation is based on the worst-case traffic, since the dynamic behaviors of packet switching network can be properly reflected by the worst-case traffic [9], [10]. Assuming that the set $\vec{X}$ denotes the communications requests in the NoC, and $\overrightarrow{load} = C \cdot \vec{X}$ denotes the workload of network. The congestion is then calculated according to the definition of Lp-norm as

$$congestion = \left\| \overrightarrow{load} \right\|_\infty = \left\| C \cdot \vec{X} \right\|_\infty \tag{2}$$

In the worst-case, the upper bound of *congestion* is calculated as

$$\begin{aligned} c_{max} &= \frac{\left\| C \cdot \vec{X} \right\|_\infty}{\left\| \vec{X} \right\|_\infty} = \|C\|_\infty \\ &= max \left\{ \sum_{i=1}^{r} |c_{1i}|, \sum_{i=1}^{r} |c_{2i}|, \cdots, \sum_{i=1}^{r} |c_{li}| \right\} \end{aligned} \tag{3}$$




(a) Sending a packet from node a to node h in a NoC with $3 \times 3$ mesh topology.

$$
\begin{array}{c}
 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
8 \\
9 \\
10 \\
11 \\
12
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc}
< a,b > & < a,c > & \dots & < a,h > & < a,i > & \dots & < g,h > & < h,i > \\
\end{array} \\
\left(\begin{array}{cccccccc}
1 & 1 & \dots & 1 & 1 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 1 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 1 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 1 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 1 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 1 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 1 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 1 & \dots & 0 & 0
\end{array}\right)
\end{array}
$$

(b) Corresponding congestion matrix. The dashed box part indicates the link occupytion of sending packet from node a to node h.

Figure 3. A sample NoC with $3 \times 3$ mesh topology and its corresponding congestion matrix.

where the $\|C\|_\infty$ denotes the induced infinite norm of matrix C and $\sum_{i=1}^{r} |c_{li}|$ denotes the cumulative result of *row l*.

In this paper, we will use the method proposed in COMRANCE to accelerate the mapping process for CGRA.

## III. Motivation

Unlike super-scalar processor, CGRA architectures use a *Static Placement Dynamical Issue* execution model, which couples compiler-driven placement of instructions with hardware-determined issue sequence. In general, the program for CGRA architecture is constituted by the instructions (as shown in Figure 4(a)), the data and the data-flow graph(as shown in Figure 4(b)). Before loading instructions to the CGRA processor, the compiler generates the instruction mapping scheme (as shown in Figure 4(c) and 4(d)) for the program according to its data-flow graph and the structure configuration. The generated mapping scheme determines a specific *Process Element (PE)* for each instruction and the distribution of the on-chip communications. Since one instruction is issued only if the required data is ready, the CGRA performance might be hurt by frequent NoC congestion. Therefore, the mapping scheme significantly affects the actual performance and energy consumption of CGRA structure.

However, the optimal mapping is difficult to achieve, since the number of possible mapping increases dramatically with the number of instructions and PEs. Besides, several combined factors aggravate the complexity of mapping optimization. In this paper, the following factors are considered:

- **NoC Congestion** is regularly caused by the contention for switching crossbar, buffer capacity or link. Once an instruction is waiting for the data from other PE, the extra delay caused by NoC congestion will postpone the instruction issue and decrease the performance.
- **Load of NoC** indicates the number of NoC packets. Reducing the load of NoC can help to lower the energy consumption of NoC and reduce the probability of performance penalty caused by NoC congestion. However, over low NoC workload may hurts the hardware utilization and overall performance.
- **Inter-PE pipelining** indicates the execution pipeline constituted by several PEs. The instructions which are neighbors in the data-flow graph are placed in these PEs and their execution is driven by the on-chip data stream among the PEs. Inter-PE pipelining enhances the hardware utilization and the performance. However, it also enhances the probability of performance penalty caused by increasing NoC utilization.
- **Workload Balance** refers that the instructions should be equally placed in all PEs, which guarantees that the execution of different PE requires similar time and reduces the waiting time of PEs.

When generating the mapping scheme, all of above factors should be considered. However, the combined factors extremely increase the complexity of the mapping optimization. For example, reducing the load of NoC by placing more instructions in the same PE can reduce the congestion and the energy consumption of the NoC, but it also loses the opportunity to take the benefits of inter-PE pipelining. When generating the mapping scheme for the program as shown in Figure 4(a), different consideration provides various performance: the scheme with *Packet Reduction First Policy* maps the instructions to 2 PEs and finishes one loop iteration within 10 cycles; on the other hand, the scheme with *Pipeline First Policy* maps the instructions to 9 PEs and finishes one iteration in 9 cycles. Although the gap of execution time is only 1 cycle, the scheme with *Pipeline First Policy* manifests great advantages for the loop. For example, when loop the program showed in Figure 4(a) for 10000 times, the scheme with *Pipeline First Policy* only requires 10008 cycles in ideal case, while the scheme with *Packet Reduction First Policy* requires 100000 cycles. However, the scheme with *Pipeline First Policy* generates much more on-chip data transmission, which significantly improves the probability of performance affected by NoC congestion in actual cases.

Similar dilemma also lies in the consideration of workload balance. Although the mapping method guarantees that the execution time of all PEs is close to each other by carefully layout, the NoC delay and unexpected NoC congestion may still break the balance and cause more waiting time.

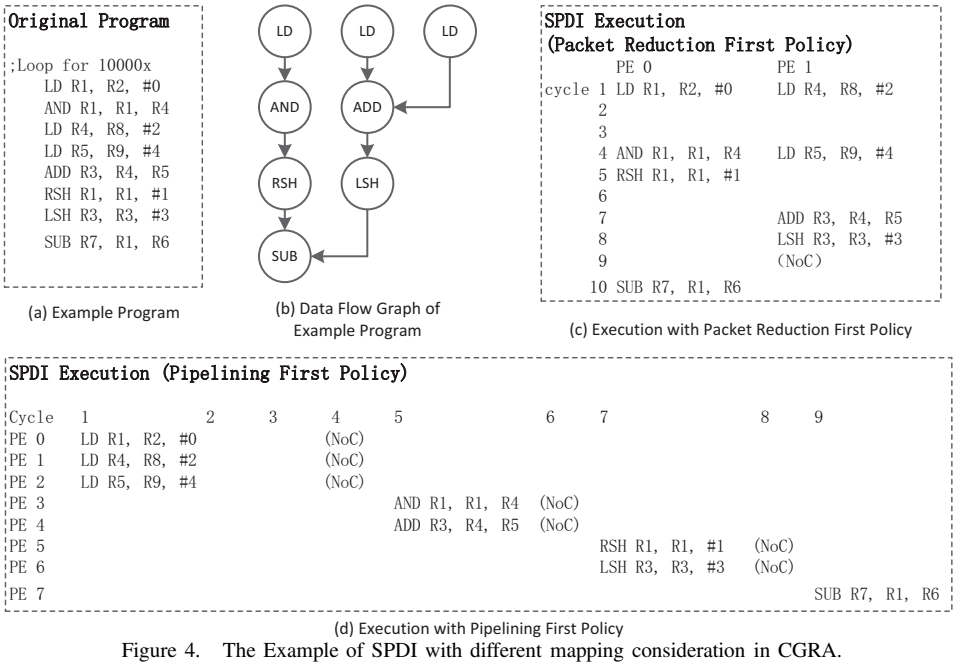



Figure 4. The Example of SPDI with different mapping consideration in CGRA.

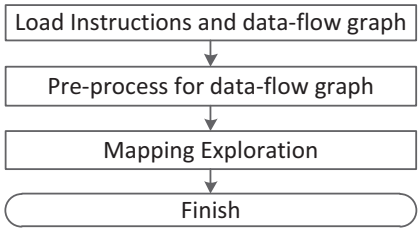



Figure 5. The overview of C-MAP mapping method.

Based on the above analysis, we observe that the NoC actually plays an important role in CGRA and has a remarkable impact on the overall performance. Since the mapping scheme determines the distribution of data streams, the generation of mapping scheme should reduce the NoC congestion as possible while considering other factors, such as workload balance, load of NoC and inter-PE pipelining. According to introduction of Section II, the congestion analysis model proposed in COMRANCE provides an intuitive way to estimate the worst-case congestion for NoC. Based on this model, we can estimate the impact on NoC congestion for different mapping plans. In addition, the mapping plan also includes the considerations for other factors. Based on such observation, we propose a novel mapping method for CGRA, which is named Congestion-MAP (C-MAP). In the next section, we will introduce the C-MAP in detail.

## IV. Scheme

### A. Overview

The overview of C-MAP method is shown in Figure 5. Note that, in the *pre-process* step, C-MAP method traverses the loaded data-flow graph with breadth-first traversal policy and reorders the instructions according to the traversal result. Then the C-MAP method explores the mapping plan based on the reordered instruction sequence. Besides, C-MAP also accelerates the exploration process with parallelism.

### B. Mapping Exploration

Figure 6 presents the workflow of mapping plan exploration. Since the global optimal mapping for long instruction sequence is hard to achieve, C-MAP uses a iterative method to get the near-optimal mapping plan. In each iteration, C-MAP sequentially selects $N$ non-mapped instructions and searches the most proper mapping plan for them. The obtained mapping plan for the newly selected $N$ instructions will be appended to the final mapping scheme. In this way, C-MAP provides a near-optimal mapping scheme at the end of the exploration. Note that, the number of selected instructions will affects the effectiveness of final mapping scheme and the performance of C-MAP: as number of selected instructions increases, the final mapping scheme gets more closer to the global optimal result, while the execution time of C-MAP significantly grows as the exploration space in each iteration dramatically expands. In section VI-E, we will analyze the influence of the number of selected instructions in each iteration with experiment results.

In the right-portion of Figure 6 shows the detail of proper mapping plan exploration in each iteration. For each selected instruction group, C-MAP generates all kinds of permutations and tries all possible mapping plan. For each possible mapping plan, C-MAP calculates following factors when a new instruction is placing to a PE and the ranking of each factor is in the ascending order:

- **Congestion Index** is estimated with the model introduced in Section II.

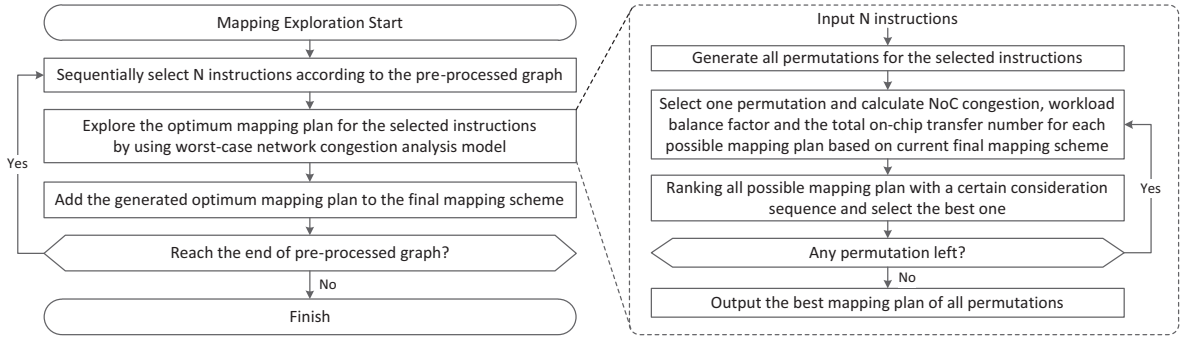



Figure 6. The detail of mapping plan exploration in C-MAP.

- **Workload Balance Index** indicates the minimum number of mapped instructions in all PEs.
- **Transfer Index** indicates the sum of hops between one instruction and its parent for all mapped instructions.

For each permutation, C-MAP ranks the all possible mapping plans according to the above factors and selects the first one as the most proper mapping plan of the permutation. Then, the selected mapping plans in every all permutations are also ranked according to the factors and takes the first one as the output of the iteration. Note that, when ranking the mapping plans, the priority order of the factors will affect the effectiveness of the mapping scheme. In Section VI-D, we will show the impact of the priority order of the factors with experimental result.

### C. Acceleration

As shown in the right-portion of Figure 6, C-MAP explores the mapping plan for all possible instruction permutations. Since the explorations of different permutations have no data-interactivity, C-MAP accelerates the exploration with parallelism. For each possible permutation, C-MAP uses one thread to fulfill the mapping exploration, and each thread only returns the exploration result. In this way, C-MAP can achieve high performance mapping exploration for large scale program by fully utilizing the parallel resources in the host processor.

## V. Methodology

To evaluate our proposed C-MAP method, we implement a CGRA structure based on SimICT [11], which supports cycle-accurate simulation for large scale processor. To estimate the energy consumption, we use Synopsis PrimeTime to obtain the basic parameters and then calculate the energy consumption for PEs, integrated memory and NoC with our model. The simulated CGRA architecture is based on a newly proposed structure [12], which consists a $4 \times 4$ PE array with inter-PE pipeline support. The detail configurations of the CGRA structure is shown in Table I.

Table II presents the detail configuration of our proposed C-MAP method. Note that, the *consideration priority* and the *exploration granularity* have impact on the C-MAP in various perspective, which will be discussed with the experiment result in Section VI. Besides, we also implement two previously proposed mapping methods for comparison, which are named RS [13] and REGIMap [14].

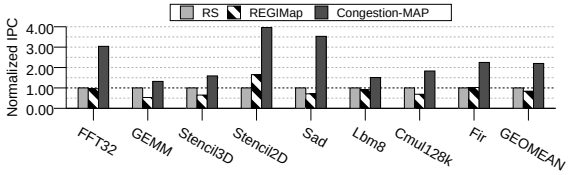



Figure 7. Normalized IPC comparison.

As is shown in Table III, we use 8 applications as the benchmark. In each experiment, we use C-MAP and other two mapping method to generate the mapping scheme according their data-flow graphs. All applications are completely executed on the simulated CGRA structure based on their corresponding data-flow graphs and mapping schemes.

## VI. Result

### A. Performance

In our experiment, we use *instruction-per-cycle (IPC)* to present the performance. As shown in Figure 7, C-MAP helps the CGRA system to achieve a significant performance improvement over the baseline work (RS) with a geometric mean of $2.2\times$ and up to $3.96\times$ in higher IPC with `Stencil2D`. This is because C-MAP reduces the congestion in CGRA NoC while considers the workload balance cross the PEs. Such optimization reduces the data-stall-time in PEs and leads to the performance improvement

Table I
Detail configuration of the implemented CGRA structure.

| | |
|---|---|
| PE number | 16 |
| Frequency | $1GHz$ |
| Process | 40nm |
| NoC | $4\times4$ Mesh Topology, bi-direction links with 1-cycle delay |
| SPM | 4MB module$\times2$ |
| Local Storage | SRAM-based, 4KB for instructions, 48KB for data, up to 128 instructions and affiliated data in each PE. |

Table II
Detail configuration of C-MAP method.

| | |
|---|---|
| Consideration Priority | Load Balance$\rightarrow$Congestion Reduction $\rightarrow$Transfer Reduction |
| Exploration Granularity | 3 instructions |
| Parallelism | 30 threads |

Table III
WORKLOAD DESCRIPTIONS

| Benchmark | Description | Total Inst. | Load/Store Inst. | Compute & Control Inst. |
|---|---|---|---|---|
| FFT | Fast Fourier Transform | 237,056 | 33,024 | 204,032 |
| GEMM | Dense Matrix Multiplication | 565,568 | 41,984 | 523,584 |
| Stencil3D | Iterative Jacobi solver of the heat equation on a 3-D structured grid. | 653,104 | 84,480 | 568,624 |
| Stencil2D | Iterative Jacobi solver of the heat equation on a 2-D structured grid. | 361,216 | 50,176 | 311,040 |
| Sad | Sum of absolute differences kernel, used in MPEG video encoders. | 1,051,344 | 42,384 | 1,008,960 |
| Lbm | Lattice-Boltzman Method | 608,400 | 91,200 | 517,200 |
| Cmul | The Vector Dot Product | 466,432 | 196,608 | 269,824 |
| Fir | Finite Impulse Response | 290,766 | 25,800 | 264,966 |

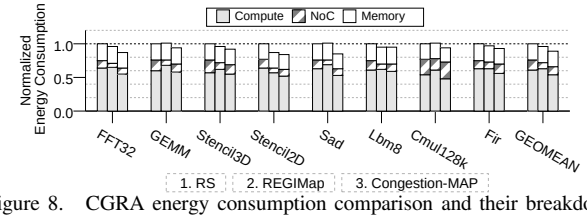


Figure 8. CGRA energy consumption comparison and their breakdown.

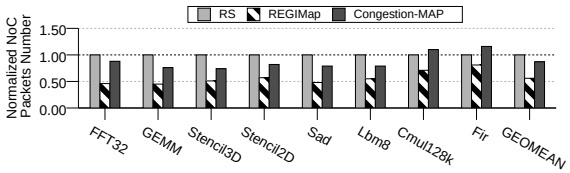


Figure 9. CGRA NoC workload comparison.

by enhancing the utilization of the PEs. However, we can also find that the effectiveness of our proposed C-MAP is related to the application characteristics. As shown in Figure 7, the performance improves by $32\%$ in `GEMM`, which is much lower than other applications.

### B. Energy Consumption

We calculate the energy consumption of CGRA with different mapping algorithms based on the parameters shown in Table I. As shown in Figure 8, comparing with the baseline, the C-MAP reduces the NoC energy consumption with a geometric mean of $20\%$. As mentioned in Section VI-A, the reduction of NoC workload and congestion help to minimize the data-stall in PEs and such improvement is also presented as the reduction of energy consumption in PEs, which is up to $18.75\%$ in `Stencil2D` and with a geometric mean of $10.31\%$. Besides, the reduction of execution time also leads to less energy consumption of memory, since the total leakage is reduced. Overall, the C-MAP achieves the energy consumption reduction for the whole CGRA is up to $16\%$ in `Stencil2D` and with a geometric mean of $9.6\%$.

### C. NoC Workload

As mentioned in Section VI-B, C-MAP achieves the energy consumption reduction in the NoC optimization approach. Such optimization includes the reduction on both network congestion and workload. In this part, we compare the number of NoC packets to show the effectiveness of C-MAP in NoC workload reduction. As shown in Figure 9, with most of the applications, C-MAP significantly reduces the NoC workload by locating communication-intensive instructions in the same PE. With a geometric mean, the NoC workload is reduced by $13\%$. Although REGIMap provides least NoC workload, C-MAP still provides better overall performance and energy consumption, since REGIMap contains less consideration of NoC congestion and workload balance.

### D. Sensitivity to Priority Order of Mapping Factors

As is discussed in Section IV, there are three considerations in the CGRA mapping process: the *load balance (L)*, the *transfer reduction (T)* and the *congestion reduction (C)*. The different priority sequence of these considerations will affect the effectiveness of the mapping algorithm. Here we examine the C-MAP with different sequence of these considerations to check how such sequence affects the NoC workload and the overall performance. As shown in Figure 10, not surprisingly, the NoC workload is minimized when the *transfer reduction* is considered with the first priority. However, as shown in Figure 11, *load balance* acts a more important role in affecting the performance. The geometric mean of IPC improvement can be up to $2.19\times$. Interestingly, when the priority of *load balance* is fixed, the sequence of *congestion reduction* and *transfer reduction* has minor impact on the performance.

### E. Sensitivity to Exploration Granularity

Recall from Section IV-C that C-MAP accelerates the mapping process by using parallelism. Each parallel thread iteratively selects one or several instructions and finds the most optimized mapping for them. The number of selected instruction in one iteration is identified as *exploration granularity*. Here we examine the C-MAP with different *exploration granularity* to check its impact on the effectiveness and exploration time cost. As shown in Figure 12, we can see that as the *exploration granularity* increases from $1$ to $8$, the system performance minor grows in most of the applications. This is because larger *exploration granularity* helps to get the result closer to the global optimum.

However, when considering the execution time, different result can be obtained. As shown in Figure 13, as the *exploration granularity* increases from $1$ to $5$, the exploration time significantly reduces. This is because larger *exploration granularity* helps to finish the mapping with less iterations. But when the *exploration granularity* reaches $6$, the exploration time starts to grow and increases dramatically when the *exploration granularity* forwards to $7$ and $8$. This is

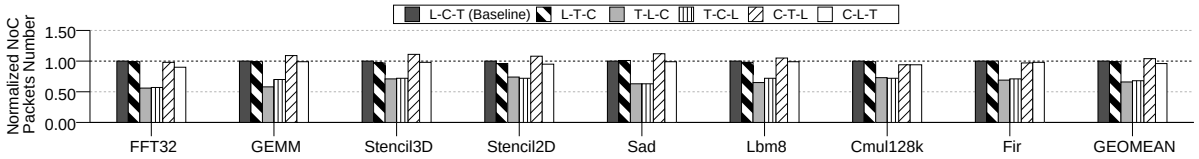



Figure 10. Normalized NoC Packets Comparison for different priority orders of mapping factors in C-Map.

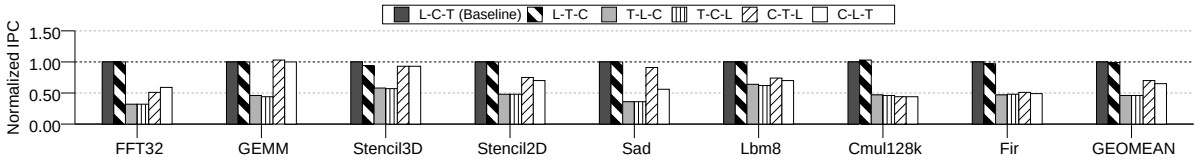



Figure 11. Normalized IPC comparison for different priority orders of mapping factors in C-Map

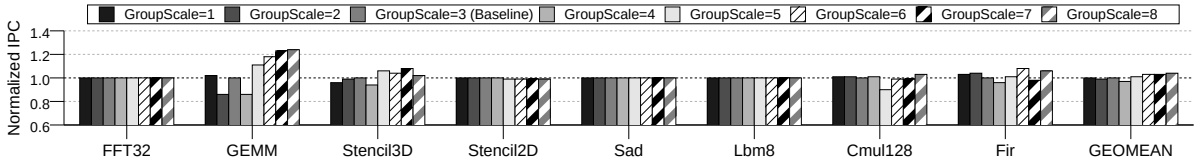



Figure 12. Normalized IPC comparison for different exploration granularity in Congestion-Map.

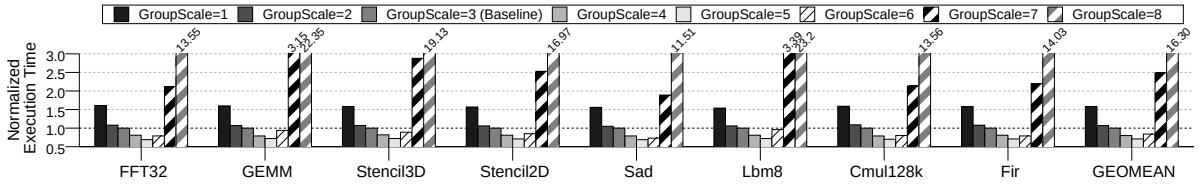



Figure 13. Normalized exploration time comparison for different exploration granularity in C-Map

because over large *exploration granularity* enlarges the sub-design-space and slows the iterations.

## VII. Related Work

In the past few years, several mapping methods have been proposed for CGRAs. Chin et al. present integer linear programming (ILP) approach with new ILP formulation, which was valid over any architecture generated by an open-source CGRA architecture evaluation framework [15]. Similarly, DNestMap is proposed for mapping deeply-nested loops onto CGRAs [16]. In DNestMap, the mapping problem was transformed into a 3D knapsack problem, and the branch and boundary method was used to find better mapping. In Ref [17], a novel algorithm based on the branch and bound method is proposed to solve the mapping problem under the constraints of CGRA. In addition, Yin et al. propose MEMMap, which uses memory as the routing resource to reduce the additional occupation of PE and registers [18]. The Branch Aware Loop Mapping proposed by Hamzeh et al. uses dual issue scheme to map loops for conditional Branch in CGRA [19]. The EPIMap proposed by Hamzeh uses re-computing technique to find feasible routing and mapping [20]. Then REGIMap uses the registers as the routing resource, and puts forward the novel register allocation algorithm [14]. The RS method with the global synchronization mechanism, made the DFG take up as few nodes and edges as possible on time-extended CGRA by using global synchronization mechanism [13]. Additionally, the RS method also introduces new revision of DFG to minimize the resource occupation, and finally generates the proper mapping plan by using the resource-monitoring mapping algorithm. RAMP [21] algorithm firstly explores the routing by memory , PE, registers and re-computing before scheduling. Then the method that can produce mapping and occupy the least resources are selected to generate mapping plan with better performance. Zhao et al. propose SPDI CGRA framework to solve the scheduling and routing problems with hardware, which automatically schedules the operations in PE with the token buffer, and the data flow can be transformed effectively in PE array [22].

Although significant improvement of effectiveness and performance have been achieved, the existed methods still have a lot of limitations. Heuristic algorithms( [15]–[17]) are only applicable to small scale DFG. For example, it takes a long time to finish mapping process for a program with 600 instructions [17]. Moreover, they are not applicable to large scale DFG. In [19], only conditional branch is optimized, and the performance improvement is limited. In Ref [13], [14], [18], [20], [21], scheduling and routing the loop kernel instructions require additional resources, which can be solved by hardware in Ref [22] and achieves better results. Besides, most of the mapping algorithms ( [13]–[21])

are based on the SISP execution model. The optimization goals includes execution performance improvement and minimizing the *initiation interval* (II) between two consecutive iterations. Additionally, such solutions combine the mapping optimization for routing, while lacks of particular optimization for instruction placement. Therefore, we propose a new algorithm for mapping instructions in SPDI CGRA, which provides performance improvement by reducing NoC congestion and improving the instruction level parallelism.

## VIII. Conclusion

The effectiveness of mapping method is proved to have a remarkable impact on the performance of CGRA. In this paper, we propose a novel mapping method based on the network congestion analysis model, which is named C-MAP. C-MAP optimizes the instructions mapping in the perspective of reducing NoC congestion while considers for other key factors that affects the CGRA performance. The experiment result shows that C-MAP provides performance improvement by $2.2\times$ as a geometric mean, while reduces the energy consumption by $9.6\%$. Besides, we compare the impact of different key factors in mapping scheme generation based on C-MAP. Additionally, we also provide sensitivity study for the key parameters in C-MAP.

## Acknowledgment

This work was supported by the National Key Research and Development Program (2018YFB1003501), the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDPB12, the National Natural Science Foundation of China (61732018, 61872335, 61802367), Austrian-Chinese Cooperative R&D Project (FFG and CAS) Grant No. 171111KYSB20170032, and the Innovation Project Program of the State Key Laboratory of Computer Architecture (CARCH3303, CARCH3407, CARCH3502, CARCH3505).

## References

[1] D. Fan, W. Li, X. Ye, D. Wang, H. Zhang, Z. Tang, and N. Sun, "Smarco: An efficient many-core processor for high-throughput applications in datacenters," in *HPCA'18*.

[2] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting ilp, tlp, and dlp with the polymorphous trips architecture," in *ISCA'03*.

[3] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin, "Wavescalar," in *MICRO'03*.

[4] O. Akbari, M. Kamal, A. Afzali-Kusha, M. Pedram, and M. Shafique, "Px-cgra: Polymorphic approximate coarse-grained reconfigurable architecture," in *DATE'18*.

[5] F. T. Leighton, B. M. Maggs, and S. B. Rao, "Packet routing and job-shop scheduling ino (congestion+ dilation) steps," *Combinatorica*, vol. 14, no. 2, pp. 167–186, 1994.

[6] T. Leighton, B. Maggs, and A. W. Richa, "Fast algorithms for finding o (congestion+ dilation) packet routing schedules," *Combinatorica*, vol. 19, no. 3, pp. 375–401, 1999.

[7] M. Zhang, Y. Shi, F. Zhang, and Z. Liu, "Comrance: A rapid method for network-on-chip design space exploration," in *IGSC'16*.

[8] E. Salminen, A. Kulmala, and T. D. Hamalainen, "Survey of network-on-chip proposals," *white paper, OCP-IP*, vol. 1, p. 13, 2008.

[9] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson, "Adversarial queuing theory," *Journal of the ACM (JACM)*, vol. 48, no. 1, pp. 13–38, 2001.

[10] M. Andrews, A. Fernández, A. Goel, and L. Zhang, "Source routing and scheduling in packet networks," *Journal of the ACM (JACM)*, vol. 52, no. 4, pp. 582–601, 2005.

[11] X. Ye, D. Fan, N. Sun, S. Tang, M. Zhang, and H. Zhang, "Simict: A fast and flexible framework for performance and power evaluation of large-scale architecture," in *ISLPED'13*.

[12] Z. Zhao, W. Sheng, W. He, Z. Mao, and Z. Li, "A static-placement, dynamic-issue framework for cgra loop accelerator," in *DATE'17*.

[13] Z. Zhao, W. Sheng, N. Jing, W. He, and Z. Mao, "Resource-saving compile flow for coarse-grained reconfigurable architectures," in *ReConFig'15*.

[14] M. Hamzeh, A. Shrivastava, and S. B. K. Vrudhula, "Regimap: register-aware application mapping on coarse-grained reconfigurable architectures (cgras)," in *DAC'13*.

[15] S. A. Chin and J. H. Anderson, "An architecture-agnostic integer linear programming approach to cgra mapping," in *DAC'18*.

[16] M. Karunaratne, C. Tan, A. Kulkarni, T. Mitra, and L.-S. Peh, "Dnestmap: mapping deeply-nested loops on ultra-low power cgras," in *DAC'18*.

[17] T. Ruschke, L. J. Jung, and C. Hochberger, "A near optimal integrated solution for resource constrained scheduling, binding and routing on cgras," in *IPDPSW'17*.

[18] S. Yin, X. Yao, D. Liu, L. Liu, and S. Wei, "Memory-aware loop mapping on coarse-grained reconfigurable architectures," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 5, pp. 1895–1908, May 2016.

[19] M. Hamzeh, A. Shrivastava, and S. B. K. Vrudhula, "Branch-aware loop mapping on cgras," in *DAC'14*.

[20] ——, "Epimap: using epimorphism to map applications on cgras," in *DAC'12*.

[21] S. Dave, M. Balasubramanian, and A. Shrivastava, "RAMP: resource-aware mapping for cgras," in *DAC'18*.

[22] Z. Zhao, W. Sheng, W. He, Z. Mao, and Z. Li, "A static-placement, dynamic-issue framework for CGRA loop accelerator," in *DATE'17*.