Tsinghua Science and Technology

Volume 26 | Issue 5 Article 11

2021

Efficient Scheduling Mapping Algorithm for Row Parallel Coarse-Grained Reconfigurable Architecture

Naijin Chen
School of Computer and Information Science, Anhui Polytechnic University, Wuhu 241000, China

Zhen Wang
School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China

Ruixiang He
School of Computer and Information Science, Anhui Polytechnic University, Wuhu 241000, China

Jianhui Jiang
School of Software Engineering, Tongji University, Shanghai 201804, China

Fei Cheng
School of Computer and Information Science, Anhui Polytechnic University, Wuhu 241000, China

See next page for additional authors

Efficient Scheduling Mapping Algorithm for Row Parallel Coarse-Grained Reconfigurable Architecture

Authors

Naijin Chen, Zhen Wang, Ruixiang He, Jianhui Jiang, Fei Cheng, and Chenghao Han

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/15 pp724–735
DOI: 10.26599/TST.2020.9010035
Volume 26, Number 5, October 2021

Efficient Scheduling Mapping Algorithm for Row Parallel Coarse-Grained Reconfigurable Architecture

Naijin Chen, Zhen Wang*, Ruixiang He, Jianhui Jiang, Fei Cheng, and Chenghao Han

Abstract: Row Parallel Coarse-Grained Reconfigurable Architecture (RPCGRA) has the advantages of maximum parallelism and programmable flexibility. Designing an efficient algorithm to map the diverse applications onto RPCGRA is difficult due to a number of RPCGRA hardware constraints. To solve this problem, the nodes of the data flow graph must be partitioned and scheduled onto the RPCGRA. In this paper, we present a Depth-First Greedy Mapping (DFGM) algorithm that simultaneously considers the communication costs and the use times of the Reconfigurable Cell Array (RCA). Compared with level breadth mapping, the performance of DFGM is better. The percentage of maximum improvement in the use times of RCA is 33% and the percentage of maximum improvement in non-original input and output times is 64.4% (Given Discrete Cosine Transfor 8 (DCT8), and the area of reconfigurable processing unit is 56). Compared with level-based depth mapping, DFGM also obtains the lowest averages of use times of RCA, non-original input and output times, and the reconfigurable time.

Key words: temporal mapping; Reconfigurable Cell Array (RCA); listed scheduling; communication costs

1 Introduction

Reconfigurable Hardware (RH) has become a hot topic as it can be configured in the spatial domain and programmed in the time domain. In the past 20 years, RH has been widely associated with Field Programmable Gate Arrays (FPGAs), which have distinct advantages in bit-width operations, but are inefficient in word-width operations. To overcome the limitations of FPGAs, various Coarse-Grained Reconfigurable Architectures (CGRAs) have been proposed in recent years[1]. In terms of their low power requirements, high performance, and flexibility, CGRAs have obvious advantages and can deal with many kinds of word-level and logic operations[1, 2]. However, it is not easy to map a computation-intensive Data Flow Graph (DFG) onto a Reconfigurable Cell Array (RCA), because there are many constraints. In previous studies, researchers have presented a wide range of mapping algorithms based on a variety of CGRAs[2–14]. Yoon et al.[2] proposed the spatial mapping algorithm, known as Split-Push Kernel Mapping (SPKM), to map several applications onto resource sharing and pipelining architecture. However, SPKM is not applicable to time-division-multiplexing mapping and the benchmarks used by SPKM have fewer nodes. Several multimedia applications are mapped to the Architecture for Dynamically Reconfigurable Embedded Systems (ADRES) processor by the dynamically reconfigurable embedded system compiler framework, and have obtained good speedup. However, this mapping method is only applicable to ADRES[3]. Some researchers have focused on extending CGRAs with Omega networks, but data transmission by Omega

• Naijin Chen, Ruixiang He, Fei Cheng, and Chenghao Han are with School of Computer and Information Science, Anhui Polytechnic University, Wuhu 241000, China. E-mail: chennaijin@ict.ac.cn; {809031856, 957189105, 1048551181}@qq.com.
• Zhen Wang is with School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China. E-mail: wangzhenqq@hotmail.com.
• Jianhui Jiang is with School of Software Engineering, Tongji University, Shanghai 201804, China. E-mail: jhjiang@tongji.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2020-07-30; accepted: 2020-09-08

networks generates very long interconnect delays, which significantly reduce the CGRAs' acceleration performance[4]. Krishnamoorthy et al.[5] presented an interconnect-topology independent mapping algorithm to map applications onto a REDEFINE architecture, but failed to fully utilize the resources of CGRAs by exploiting operation-level parallelism. Ahn et al.[6] proposed a mapping algorithm that includes three sub-problems: covering, partitioning, and layout. However, this spatial mapping algorithm is not applicable to temporal partitioning and mapping. Ansaloni et al.[7] used a novel scheduling strategy that considers both registered and unregistered communication among tiles. Lee et al.[8] and Jo et al.[9] introduced approaches for supporting floating-point operations for CGRAs. Kim et al.[10] proposed a fast modulo routing scheduling technique for mapping 3D graphics benchmarks onto CGRAs, which improved the compilation speed. Level-Breadth-Mapping (LBM) partitions the nodes by level. Level-Breadth and Depth Mapping (LBDM) considers both breadth and depth[11]. However, both LBM and LBDM have higher communication costs. Existing approaches do not sufficiently consider the use times of the RCA or the non-original input or output times. Xiao et al.[15] and Ouyang et al.[16] discussed gates and fault tolerant design.

The remainder of the paper is organized as follows. In Section 2, we define the problems associated with temporal mapping and provide the target architecture. In Section 3, we design and develop a Depth-First Greedy Mapping (DFGM) algorithm. In Section 4, we demonstrate the efficiency of our approach on several indexes and analyze our experimental results. Finally, we draw our conclusions in Section 5.

2 Target Architecture and Problem Definition

2.1 Target architecture

A Row Parallel Coarse-Grained Reconfigurable Architecture (RPCGRA) (e.g., REMUSs[12]) is a typical coarse reconfigurable processor consisting of a main processor, a Direct Memory Access (DMA) controller, a main memory, one or several Reconfigurable Processing Units (RPUs), an advanced high-performance bus, and other elements. An RPU contains one or several RCAs, which is a 2-D mesh array connected via an interconnect network, a global controller, a configuration controller, an instruction/data memory controller, local data memory, and local instruction memory. Each Reconfigurable Cell (RC) can be configured to realize many different kinds of operations. Each RC can communicate with its up-row RCs or down-row RCs by a router, such that each RC reads inputs from the up-row RCs, local memory, or the above RCA, and writes to the down-row RCs or local memory. Local memory can supply operands to the RCs and store the computing results from the RCs. The configuration controller can dynamically change the context register sets, which provide configuration information to the multiplexer, RCs, and router module. The configuration memory of the RPCGRA can store multiple sets of configuration words. One of the context registers is applied to control, and the rest registers are used to configure the buffer data. RPCGRA based on an RPU is shown in Fig. 1.

In this paper, we discuss the mapping problem with respect to RPCGRAs (e.g., REMUS, MorphoSys[13], and the other similar architectures). A fully interconnected RPCGRA makes operation easier, but our objective is to develop a temporal mapping algorithm that can be applied to a low-cost RPCGRA.

2.2 Problem definition

Here, we only consider a fixed-point integer operation. The relevant definitions of the mapping problem are as follows.

Definition 1. Row Parallel Execution RCA (RPER): A loop sub-DFG is partitioned and mapped onto an RCA. Each row of mapped nodes has the following properties: (1) Mapped nodes in the same row are non-dependent and can execute concurrently. (2) Each row of mapped nodes can be configured and executed simultaneously. An array having these properties is called an RPER. A reconfigurable computing architecture having these RPER properties is called an RPCGRA.

Definition 2. Use times of the RCA: Supposing an RPU contains one RCA, where the number of nodes for a loop sub-DFG is larger than the number of RCs on one RCA. The loop sub-DFG is divided into several sub-tasks, which are mapped onto the RCA under the hardware resource constraints. Based on an RCA, these sub-tasks are executed and scheduled regarding their repeated configuration and use. The number of times they are repeatedly configured and used is called the use times of the RCA.

Definition 3. Loop kernel DFG[14]: The loop kernel

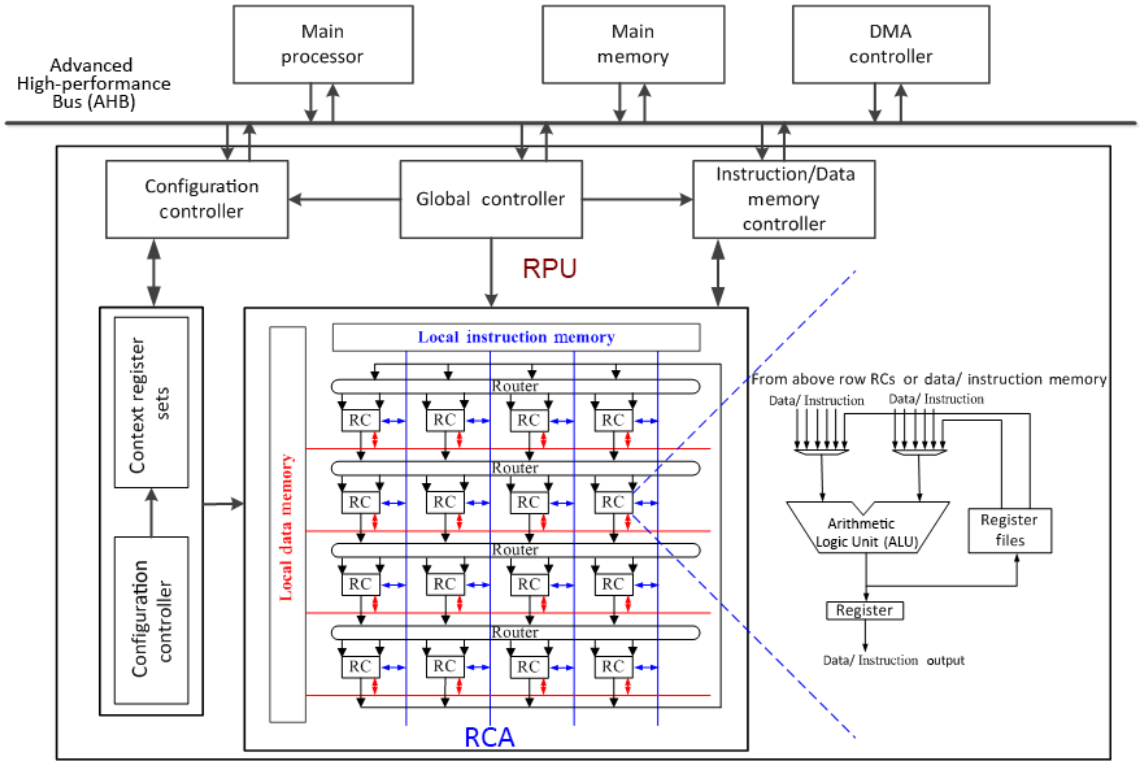


Fig. 1 General RPCGRA architecture.

DFGs of computation-intensive tasks or programs can be expressed as a four-tuple $G = (V, E, W, D)$, where V is the set of vertexes. $V = \{v_i|$ where each v_i is an ordered operation, $1 \leqslant i \leqslant n\}$, and $|V| = n$ is the number of operations. E is the set of data-dependent edges. $E = \{e_{ij}|e_{ij} = \langle v_i, v_j\rangle, 1 \leqslant i, j \leqslant n\}$, where each e_{ij} denotes a directed edge from v_i to v_j, v_i is the direct predecessor node of v_j, and v_j is the direct successor node of v_i. In other words, each e_{ij} indicates the dependency relationship between v_i and v_j, i.e., the execution of v_j depends on v_i. $|E| = m$ indicates the number of edges. $W = \{w_i|$ each w_i indicates the hardware resource area of each operation node v_i, $1 \leqslant i \leqslant n\}$. D represents the set of delays or latencies, and $d_i \in D$ represents the delay of the i-th operation execution time.

Definition 4. Reconfigurable Cell Array Model (RCAM): Let $R_{\text{row}} \times C_{\text{col}}$ denotes the size of a two-dimensional RCAM $= (R, L, \text{IN}, \text{OUT})$, where $R = \{r_{1,1}, r_{1,2}, \ldots, r_{k,h}\}$ is a finite set of reconfigurable processor units, and the basic unit is ALU. In multimedia applications, each $r_{k,h}$ $(1 \leqslant k \leqslant R_{\text{row}}$ and $1 \leqslant h \leqslant L_{\text{col}})$ unit can perform either general arithmetic, logical operations, or special operations, such as computing the absolute value, shift addition, comparison, and so on. $\text{IN} = \text{IN}(r_{1,1}) \cup \text{IN}(r_{1,2}) \cup \cdots \cup \text{IN}(r_{k,h})$, where $\text{IN}(r_{k,h})(1 \leqslant k \leqslant R_{\text{row}}$ and $1 \leqslant h \leqslant L_{\text{col}})$ is the set of $r_{k,h}$ input ports. $\text{OUT}{=}\text{OUT}(r_{1,1}) \cup \text{OUT}(r_{1,2}) \cup \cdots \cup \text{OUT}(r_{k,h})$, where $\text{OUT}(r_{k,h})$ is the set of $r_{i,j}$ output ports; $L \subseteq \text{OUT} \times \text{IN} = \{\langle o, i\rangle | o \in \text{OUT}, i \in \text{IN}\}$, where L is a finite set, in which each element represents the data-dependency connection relation between the output port of one RC and the input port of another RC.

Definition 5. DFG temporal mapping: In general, the number of $v_i \in V$ in a DFG is greater than the number of RCAs. Therefore, a DFG should be partitioned and mapped onto the RCA with several constraints. DFG temporal mapping is the process of executing a DFG by an RCA in the time domain and collecting the calculation results, which is denoted as the function: $G \longrightarrow$ RCAM.

Definition 6. RCA successful scheduling mapping: After mapping, an operation node $v_i (1 \leqslant i \leqslant n)$ occupies an RC. A 2-D RCA can be represented by the array graph RCAM $= (R, L, \text{IN}, \text{OUT})$. For any two units $r_{i1,j1}, r_{i2,j2} \in R$, $\exists\, l_1 = (r_{i1,j1}, r_{i2,j2}) \in L$, and a given $G = (V, E, W, D)$, if a mapping DFG of RCAM is found with a correct interconnection between $r_{i1,j1}$ and $r_{i2,j2}$, an RCA successful scheduling mapping is obtained. Otherwise, it is called invalid scheduling mapping.

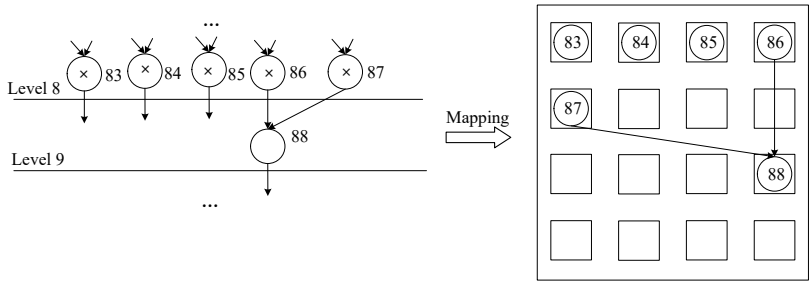
Definition 7. DFG non-original input and output

times: A DFG is partitioned according to many RCA constraints (i.e., area, interconnect modes, etc.). As the area of the RCA is finite, a DFG can be partitioned into several blocks. In this situation, the input times between one RCA and the other RCAs are called the DFG non-original input times, and the output times between one RCA and the other RCAs are called the DFG non-original output times.

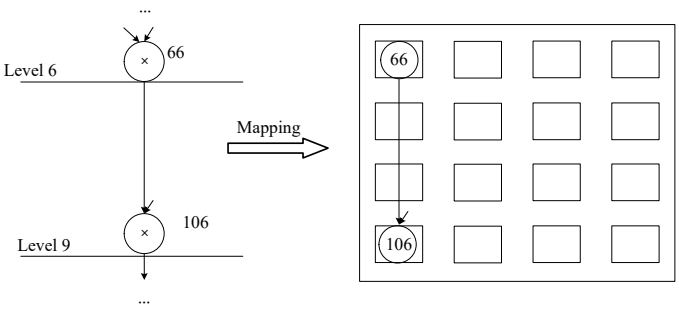
Definition 8. RCA communication costs: The RCA communication costs include original input times, non-original input times, original output times, and non-original output times. Generally speaking, the area of the RCA in one RPU is a constant value, which can be represented by A_{RPU}. Several sub-graphs are partitioned according to the size of the RCA and the interconnect mode of the RCA. From top to bottom and from left to right, each partitioned sub-graph is mapped to an RCAM in the proper scheduled order. The objective of mapping is to minimize the total execution delay.

Definition 9. Cross-level data transmission: With respect to the RPCGRA row pipeline framework, the computing result of the above row operation can only be sent to any node in the next row by the router, or the computing result of the above row operation can be saved to local data memory. Because the RC is used as a transitional node for data transmission, the cost of its configuration would increase greatly, and this situation is not considered in this article. The category of cross-level data transmissions includes misplacement, direct cross-level, and interlaced (see Fig. 2).

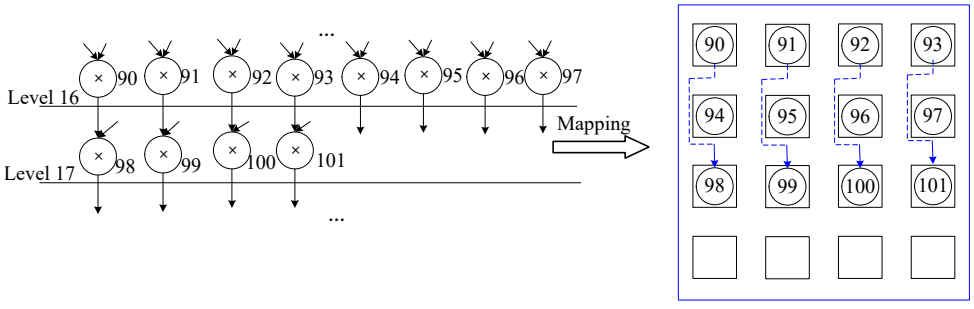
Definition 10. RC: An RC is a processing element, which is an important part of the CGRA arithmetic array. Its functions include arithmetic logic operations, data transmission, and configuration. An array of RCs is referred to as an RCA or Processing Element Array (PEA).



(a) Misplacement



(b) Direct cross-level



(c) Interlaced

Fig. 2 Illustration of misplacement, direct cross-level, and interlaced modes mapped in one RCA.

3 Proposed Approach

In this section, we propose and design a DFGM algorithm for a reconfigurable system comprising a main processor and an RPU with several hardware resource constraints.

3.1 Temporal mapping quantitative indexes for delays of the CGRA

Based on our analysis and research, we identified six temporal mapping quantitative indexes for the delays of a CGRA: N_1, N_2, M, S_{SD}, C_{CON}, and I_{ID}, where N_1 is the non-original input times, N_2 is the non-original output times, M is the use times of the RCA, S_{SD} is the sum of a DFG execution delay, C_{CON} is the reconfigurable time, and I_{ID} is the cross-level data transmission interconnect delay.

3.2 Two premise conditions

(1) Row parallelism: Dependency among row nodes does not occur after an operation node has been mapped. As such, operations in the same row can be executed simultaneously, by which parallelism degree maximization can be obtained, thereby reducing the execution delays of the computing nodes of the inner RCA.

(2) Operation execution time and configuration time: The time of each mod operation is 4 clock cycles, the time of each multiplication operation is 2 clock cycles, the time of other operations is 1 clock cycle, and the configuration time of each operation is 1 clock cycle.

3.3 Optimization objective

Given: $[\mathrm{RCA}]_{\mathrm{row}\times\mathrm{col}}$

s.t.,

(1)$1 \leqslant i \leqslant \mathrm{row}$;

(2) $1 \leqslant j \leqslant \mathrm{col}$;

(3) $A_{\mathrm{RPU}} = \mathrm{row} \times \mathrm{col}$;

(4) $T_{\mathrm{TOTAL}} = \alpha \times (N_1 + N_{\mathrm{org1}} + N_2 + N_{\mathrm{org2}}) + \beta \times M + C_{\mathrm{CON}} + S_{\mathrm{SD}}$,

where α is an adjustment factor determined by different CGRAs. We obtained $\alpha = 0.5$ in an actual test of the REconfigurable MUltimedia System (REMUS) compiler. β represents the number of control registers. Taking REMUS as example, $\beta = 17$. N_{org1} is the original input times and N_{org2} is the original output times.

Optimization objective: Minimize N_1, N_2, M, and C_{CON}.

3.4 Three mapping strategies

Strategy 1: Adoption of point-to-point mapping method.

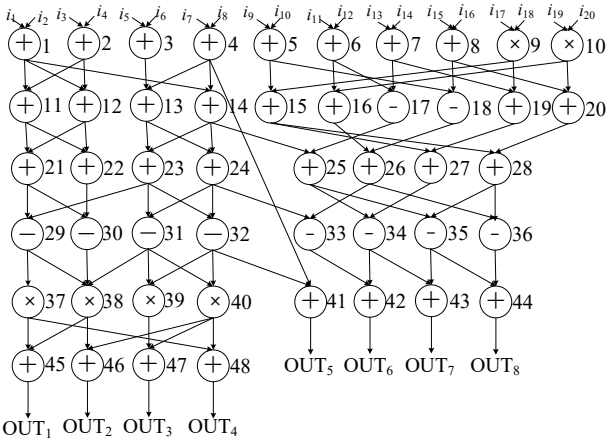
Cross-level data transmission involves three modes: misplacement, direct cross-level, and interlaced (see Fig. 2).

Cross-level data transmission interconnection can directly increase I_{ID}. Taking Fig. 2a as an example, because the mapping nodes in an RCA are executed by rows from top to bottom, the rows level of v_{86} is lower than that of v_{87}, so v_{86} is calculated first and the data sources of v_{88} are not synchronized, which results in a great increase of I_{ID}. Based on the above, cross-level mappings are not permitted and operation nodes are mapped successfully. Thus I_{ID} is zero and is not considered. To reduce the configuration time, the addition of bypass node mapping is also not permitted.

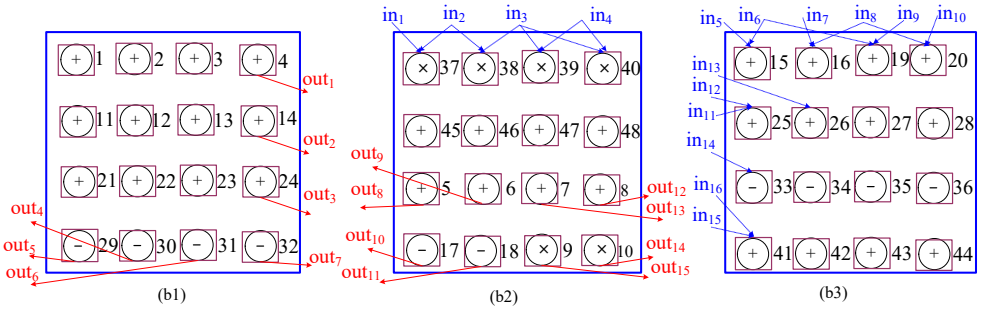
Strategy 2: Consideration of the communication costs of different RCAs.

Communication costs mainly depend on non-original input times and non-original output times after partitioning and mapping a DFG onto an RCA, whereby mapping methods with fewer input and output times have lower communication costs. Communication costs can be reduced using the following method.

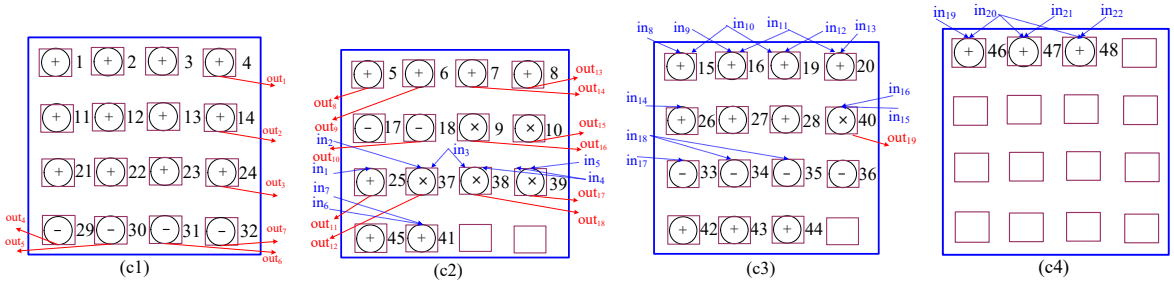
Example 1. As shown in Fig. 3a, a loop sub-DFG (from the assessment program h264_h_loop_filter_luma_intra_c (HHLFLIC)) contains 20 original inputs, 8 original outputs, 48 operation nodes, 76 non-original edges, 32 addition operations, 10 subtraction operations, and 6 multiplication operations. Based on the RPCGRA architecture, starting from the first row of the upper left corner in the RCA, the DFGM finds v_1 and maps it onto the RCA. As its immediate successor v_{11} is not ready, in order to reduce N_1 and N_2, v_{11} is placed in advance. Under the hardware resources constraints, we try to schedule and map its immediate predecessor v_2 onto the RCA. If v_2 is mapped successfully, the scheduling sequence is $v_1 \rightarrow v_2 \rightarrow v_{11}$. Otherwise, we look for the next starting point, and so on. Iterative mappings by depth are executed first. Based on $\mathrm{RCA}_{4\times4}$ of the RPCGRA, as shown in Fig. 3b1, the scheduling sequence of the first M is $v_1 \rightarrow v_2 \rightarrow v_{11} \rightarrow v_{12} \rightarrow v_{21} \rightarrow v_{22} \rightarrow v_{30} \rightarrow v_3 \rightarrow v_4 \rightarrow v_{13} \rightarrow v_{14} \rightarrow v_{23} \rightarrow v_{24} \rightarrow v_{29} \rightarrow v_{31} \rightarrow v_{32}$. M_2 and M_3 are shown in Figs. 3b2 and 3b3. Based on the same loop sub-DFG, the mapping results of the LBDM[11] are shown in Fig. 3c, and those



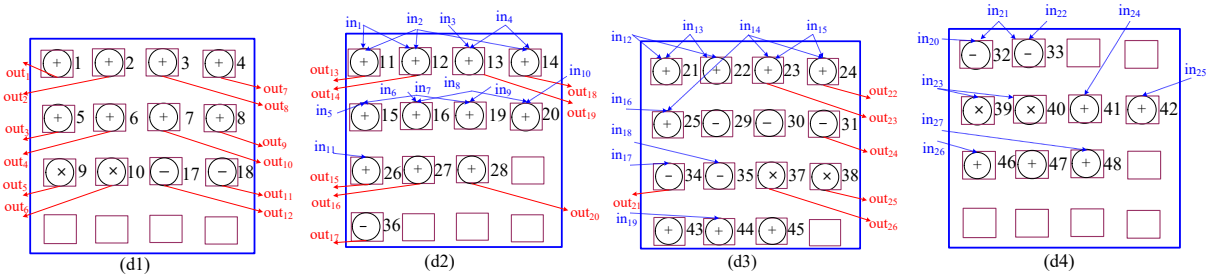
(a) Loop sub-DFG of benchmark HHLFLIC



(b) DFGM mapping result



(c) LBDM mapping result



(d) LBM mapping result

Fig. 3 Comparison of DFGs.

of the LBM[11] are shown in Fig. 3d. A comparison of LBDM and LBM reveals that DFGM can decrease the communication costs of one RCA and other RCAs, i.e., $N_1 = 16$ and $N_2 = 15$, whereas the communication costs of the LBDM are $N_1 = 22$ and $N_2 = 19$, and those of the LBM are $N_1 = 27$ and $N_2 = 26$.

Strategy 3: Decreasing M by greedy mapping.

When using this method, deadlock does not occur and the interconnect delay is minimized (i.e., no direct cross-level, interlaced, or misplacement mapping occurs). In this case, DFGM adopts a greedy mode (e.g., as shown in Fig. 3b, greedy mapping v_9 and v_{10}) to fill in each

RCA in the mapping scheduling process. By doing so, the number of M can be decreased. In Fig. 3, we can see that $M = 3$ (DFGM), whereas $M = 4$ (LBM or LBDM). More importantly, fewer M can result in less configuration time and less execution time, which reduces the total delays. Table 1 shows a comparison of the mapping results of LBM, LBDM, and DFGM, from which we can see that DFGM performs better than LBM and LBDM.

3.5 DFGM algorithm design

In this section, we present the DFGM algorithm, which satisfies both the above premised conditions and the related mapping strategies. The pseudo code is shown in Algorithm 1 as follows.

Supposing the 2-D RPCGRA is as shown in Fig. 1. In Algorithm 1, Step 2 applies Strategy 1 to filter illegal nodes and Strategy 2 to complete a recursive depth first search in DFS (node[i].id, matrix_level). Step 3 applies Strategy 3.

The time complexity of reading the data table is $O(n)$, where n is the number of DFGs. The time complexity of DFS(node[i]. id, matrix_level) is O(succ). Scanning the Max_graph_level and mapping by DFS(node[i]. id, matrix_level) have a total time complexity of $O(n^2 \times \text{succ})$. The time complexity of $S_{\text{SD}}()$ is $O(R_{\text{row}} \times C_{\text{col}})$. The time complexity of N_1_edges() and N_2_edges() is $O(M \times n \times \text{succ})$ and $O(n \times \text{succ})$, respectively, where succ is the nodes of the successors in a DFG. From the above, the average time complexity of DFGM is $O(n^2 \times \text{succ})$.

4 Experimental Result and Discussion

4.1 Benchmark

The A_{RPU} values selected at random are 42 ($\text{RCA}_{6\times7}$) and 56 ($\text{RCA}_{7\times8}$). For different A_{RPU} values, several benchmarks are adopted to verify the effectiveness of DFGM. Five indexes (i.e., M, N_1, N_2, S_{SD}, and C_{CON}) are considered in this paper, and ten benchmarks are used, including FEAL, DCT8, FFT8, EWF6, MATRIX4, put_h264_luma_mc_c_4×4(PHLMC4), HHLFLIC,

Table 1 Mapping parameter comparison of LBM, LBDM, and DFGM.

Algorithm	Parameter					
	M	N_1	N_2	S_{SD} (clock cycle)	C_{CON} (clock cycle)	T_{TOTAL} (clock cycle)
LBM	4	27	26	17	116	159.5
LBDM	4	22	19	16	116	152.5
DFGM	3	16	15	14	99	128.5

Algorithm 1 DFGM

Input: DFG

Output: Configuration information, M, N_1, N_2, S_{SD}, C_{CON}, and T_{TOTAL}

Constraint: RPCGRA architecture; $A_{\text{RPU}} = 42$ or 56; illegal dependencies are not happened; misplacement, direct cross level, and interlaced are not happened; nodes in each row can be executed in parallel, $\text{RCA}_{i\times j}$ ($1 \leqslant i \leqslant R_{\text{row}}$; $1 \leqslant j \leqslant C_{\text{col}}$)

Objective: obtaining optimization M, N_1, N_2, S_{SD}, C_{CON}, and T_{TOTAL}

Step 1:

Initializating array schedule [$R_{\text{row}} \times C_{\text{col}}$] = {0} & reading data table; sorting by level and node-number; $M = 0$; matrix_level = 0; $n = 0$; nodenumber = 0

Step 2:

for1 (graph_level = 1 to max_graph_level)

for2 (i = 1 to nodenumber)

if (the indegree of node is 0 && node[i].flag = 0 && node[i].level=graph_level)

Call misplacement(), direct-crossing-level(), and interlaced() to filter illegal nodes in DFS (node[i].id, matrix_level);

// matrix_level indicates the level of RCA

end if

for (j=1 to A_{RPU})

Mapping_Matrix[schedule [j].x][schedule [j].y]=schedule [j];

n++;

// mapping satisfied constraint nodes onto RCA

end for

Step 3:

if ((num[matrix_level]=C_{col})||(graph_level = max_ graph_level && matrix_level < row && flag_ last_node=0))

if (num[matrix_level]= C_{col})

while (matrix_level<R_{row})

if (num[matrix_level]= C_{col})

matrix_level++;

// Column is mapped completely

else break;

end while

Finding the starting row of the next sub-graph and greedy mapping;

else matrix_level = R_{row};

end if

Step 4:

if (the current RCA is mapped completely || the current RCA is not mapped completely, but the nodes do not satisfy the constraints), M++; matrix_level=0; graph_level=0; schedule[$R_{\text{row}} \times C_{\text{col}}$]={0};

Mapping_Matrix[schedule[j].x][schedule [j].y]={0};

end if

end for2

end for1

Step 5: if (n =nodenumber) break; M is obtained. N_1, N_2, and S_{SD} are got by N_1_edges(), N_2_edges(), and $S_{\text{SD}}()$, respectively; C_{CON} and T_{TOTAL} are got by delays();

Step 6: end DFGM

put_h264_luma_mc_c_8×8(PHLMC8), h264_h_loop_filter_luma_c(HHLFLC), and DCT32. The number of operands is listed in Table 2 (add represents addition; sub represents subtraction; mul represents multiplication; log represents logical comparison; less represents less than; le represents less than or equal; lsr represents logical shift right; lsl represents logical shift left; asr represents arithmetic shift right; mod represents modulus operator; and xor represents XOR operator). We implemented LBM, LBDM, and DFGM in C++.

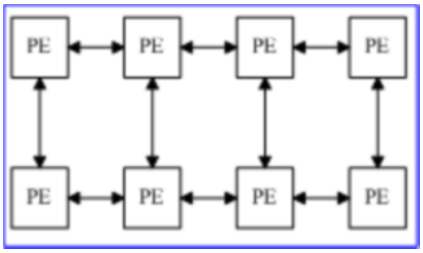
4.2 Grid PEA and row router PEA

Most recent mapping and related research has been based on grid PEA[17–19], which has the advantage of easy interconnection (see Fig. 4). However, for a DFG with multi-outputs or multi-inputs, grid PEA has the disadvantage of low throughput compared with row router PEA. Here we take a sub-DFG of DCT32 as an example (Fig. 5).

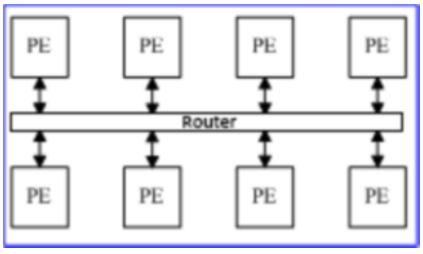
The benchmarks of most DFGs (Table 2) are characterized by multi-outputs or multi-inputs. However, as we can see in Table 3, based on a sub-DFG of DCT32, row router PEA has a more comprehensive optimization performance than grid PEA. So in this paper, we primarily study the mapping of row router PEA. At present, there are more optimized algorithms, such as LBM and LBDM based on row router PEA, but in-depth study and comparison reveal that LBM has high communication cost and LBDM must make a trade-off between communication costs and parallelism, its overall performance is still poor. Comparisons of the performances of LBM, LBDM, and DFGM are presented above.

4.3 Comparison of DFGM and LBM based on RPCGRA

In Table 2, ten benchmarks are mapped onto CGRAs based on several constraints of DFGM and LBM, seperately, the experimental results are shown in Tables 4 and 5 for $A_{\mathrm{RPU}} = 42$ ($\mathrm{RCA}_{6\times7}$) and $A_{\mathrm{RPU}} = 56$ ($\mathrm{RCA}_{7\times8}$). The results and improved percentages ($\triangle\%$) are listed in the corresponding columns. In the M column, taking DCT8 as an example, when $A_{\mathrm{RPU}} = 42$,



(a) Grid PEA



(b) Row router PEA

Fig. 4 Two kinds of PEA interconnection.

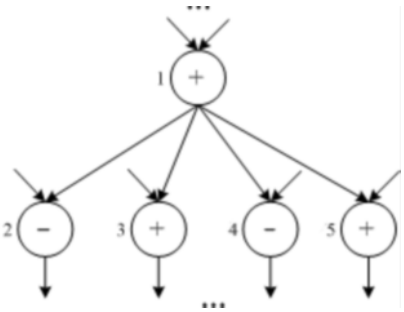


Fig. 5 Sub-DFG of DCT32.

Table 2 Set of benchmarks.

Benchmark	Number of operations											
	Total	add	sub	mul	log	less	le	lsr	lsl	asr	mod	xor
FEAL	34	6	0	0	0	0	0	0	4	0	4	20
DCT8	90	40	16	34	0	0	0	0	0	0	0	0
FFT8	36	12	12	12	0	0	0	0	0	0	0	0
EWF6	204	168	0	36	0	0	0	0	0	0	0	0
MATRIX4	112	48	0	64	0	0	0	0	0	0	0	0
PHLMC4	336	128	48	80	32	0	16	0	32	0	0	0
HHLFLIC	596	172	132	212	0	20	0	0	36	24	0	0
PHLMC8	624	224	96	160	0	0	32	0	48	64	0	0
HHLFLC	648	152	212	136	0	24	0	12	60	52	0	0
DCT32	562	192	129	129	0	0	0	112	0	0	0	0

Table 3 Mapping parameter comparison of grid PEA and row router PEA.

Structure	Parameter					
	M	N_1	N_2	S_{SD} (clock cycle)	C_{CON} (clock cycle)	T_{TOTAL} (clock cycle)
Grid PEA	2	2	2	3	39	49
Row router PEA	1	0	0	2	22	29

$M = 4$, which is obtained by LBM, where $M = 3$ is obtained by DFGM, thus $\triangle\% = -25.0\%$ (the negative value indicates improvement), and the rest can be performed in the same manner. Compared with LBM, the values of the five indicators, M, N_1, N_2, S_{SD}, and C_{CON}, are improved overall by DFGM. DFGM also obtains the lowest average execution delay (i.e., S_{SD}) with an increase in the RCA area.

4.4 Comparison of DFGM and LBDM based on RPCGRA

We adopted the ten benchmarks shown in Table 2. Merging breadth-first partitioning with depth-first partitioning, LBDM considers M, N_1, N_2, S_{SD}, and C_{CON} comprehensively and obtains a good result, but DFGM ($A_{RPU} = 42$ and 56) still obtains certain degree of optimization for M, N_1, N_2, S_{SD}, and C_{CON}. DFGM also obtains the lowest average of the five key indicators and the lowest total average. Details of the results are listed in Tables 6 and 7.

5 Conclusion

In this paper, we present a DFGM mapping algorithm for CGRAs. A comparison of the experimental results obtained by LBDM and LBM in the same pipelining RCA structure reveals that DFGM obtains better results. DFGM exhibits advantages with respect to $M, N_1, N_2, S_{SD}, C_{CON}$, and S_{SD}, especially with respect to reducing M, N_1, N_2, and C_{CON}. As such, we conclude that the proposed DFGM is reasonable and feasible.

Table 5 Comparison of DFGM and LBM (M, N_1, and N_2).

Benchmark	M						N_1						N_2					
	LBM		DFGM		△%		LBM		DFGM		△%		LBM		DFGM		△%	
	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$
FEAL	3	2	3	2	0	0	11	10	11	10	0	0	11	10	11	10	0	0
DCT8	4	3	3	2	–25.0	–33.3	57	45	23	16	–59.6	–64.4	57	45	23	16	–59.6	–64.4
FFT8	4	3	4	3	0	0	17	16	17	16	0	0	17	16	17	16	0	0
EWF6	7	6	6	5	–14.3	–16.7	141	121	116	104	–17.7	–14.0	116	108	94	94	–19.0	–13.0
MATRIX4	4	3	3	3	–25.0	0	80	70	50	51	–37.5	–27.1	80	70	50	51	–37.5	–27.0
PHLMC4	9	8	8	7	–11.1	–12.5	209	177	142	145	–32.1	–18.1	201	167	140	140	–30.3	–16.2
HHLFLIC	16	13	15	12	–6.3	–7.7	547	517	314	309	–42.6	–40.0	537	506	301	300	–43.9	–40.7
PHLMC8	16	13	15	12	–6.3	–7.7	578	516	293	298	–49.3	–42.2	529	484	286	296	–45.9	–38.8
HHLFLC	19	15	19	14	0	–6.7	560	547	345	340	–38.4	–37.8	558	545	341	340	–38.9	–37.6
DCT32	17	14	15	12	–11.8	–14.3	352	276	204	181	–42.0	–34.4	334	266	194	179	–41.9	–32.7
Average △%	0	0	0	0	–14.3	–14.1	0	0	0	0	–39.9	–34.8	0	0	0	0	–39.6	–33.8

Table 6 Comparison of DFGM and LBM (S_{SD}, C_{CON}, and T_{TOTAL}).

Benchmark	S_{SD}						C_{CON}						T_{TOTAL}					
	LBM		DFGM		△%		LBM		DFGM		△%		LBM		DFGM		△%	
	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$
FEAL	29	24	29	24	0	0	85	68	85	68	0	0	142.5	119.5	142.5	119.5	0	0
DCT8	25	23	23	18	–4.0	–21.7	158	141	141	124	–10.8	–12.1	293	262	240	211	–18.1	–19.5
FFT8	10	9	10	9	0	0	104	87	104	87	0	0	145	126	145	126	0	0
EWF6	49	39	42	34	–14.3	–12.8	323	306	306	289	–5.3	–5.6	575.5	534.5	528	497	–8.3	–7.0
MATRIX4	27	23	27	26	0	13.0	180	163	163	163	–9.4	0	359	328	312	312	–13.1	–4.9
PHLMC4	69	60	70	65	1.4	8.3	489	472	472	455	–3.5	–3.6	931	872	851	830.5	–8.6	–4.8
HHLFLIC	138	129	144	127	4.3	–1.6	868	817	851	800	–2.0	–2.1	1808.5	1718	1563	1492	–13.6	–13.2
PHLMC8	123	105	134	117	8.9	11.4	896	845	879	828	–1.9	–2.0	1868.5	1746	1598	1538	–14.5	–11.9
HHLFLC	148	139	152	133	2.7	–4.3	971	903	971	886	0	–1.9	1972	1882	1745.5	1653	–11.5	–12.2
DCT32	153	142	149	133	–2.6	–6.3	851	800	817	766	–4	–4.3	1660	1526	1478	1392	–11.0	–8.8
Average △%	0	0	0	0	–0.5	–1.8	0	0	0	0	–5.7	–4.5	0	0	0	0	–12.3	–10.3

Table 7 Comparison of DFGM and LBDM (M, N_1, and N_2).

Benchmark	M						N_1						N_2					
	LBDM		DFGM		△%		LBDM		DFGM		△%		LBDM		DFGM		△%	
	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$
FEAL	3	2	3	2	0	0	11	10	11	10	0	0	11	4	11	4	0	0
DCT8	3	3	3	2	0	–33.0	28	20	23	16	–18.0	–20.0	28	20	23	16	–18.0	–20.0
FFT8	4	3	4	3	0	0	17	16	17	16	0	0	17	16	17	16	0	0
EWF6	6	5	6	5	0	0	116	108	116	104	0	–3.7	94	96	94	94	0	–2.1
MATRIX4	3	3	3	3	0	0	57	53	50	51	–12.0	–3.8	57	53	50	51	–12	–3.8
PHLMC4	9	7	8	7	–11.0	0	148	147	142	145	–4.1	–1.4	146	145	140	140	–4.1	–3.4
HHLFLIC	15	12	15	12	0	0	310	305	314	309	1.3	1.3	309	308	301	300	–2.6	–2.6
PHLMC8	15	12	15	12	0	0	293	296	293	298	0	0.7	295	294	286	296	–3.1	0.7
HHLFLC	18	15	19	14	5.6	–6.7	354	353	345	340	–2.5	–3.7	349	352	341	340	–2.3	–3.4
DCT32	16	13	15	12	–6.3	–7.7	209	190	204	181	–2.4	–4.7	202	183	194	179	–4.0	–2.2
Average △%	0	0	0	0	–3.9	–16.0	0	0	0	0	–6.3	–4.4	0	0	0	0	–6.6	–4.6

Table 8 Comparison of DFGM and LBDM (S_{SD}, C_{CON}, and T_{TOTAL}).

Benchmark	S_{SD}						C_{CON}						T_{TOTAL}					
	LBDM		DFGM		△%		LBDM		DFGM		△%		LBDM		DFGM		△%	
	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$	$RCA_{6\times7}$	$RCA_{7\times8}$
FEAL	29	24	29	24	0	0	85	68	85	68	0	0	142.5	119.5	142.5	119.5	0	0
DCT8	23	22	23	18	0	–18.2	141	141	141	124	0	–12.1	245	236	240	211	–2.0	–10.1
FFT8	10	8	10	9	0	12.5	104	87	104	87	0	0	145	125	145	126	0	0.8
EWF6	45	39	42	34	–6.7	–12.8	306	289	306	289	0	0	531	505	528	497	–0.6	–1.6
MATRIX4	23	27	27	26	17.4	–3.7	163	163	163	163	0	0	315	315	312	312	–1.0	–1.0
PHLMC4	74	62	70	65	–5.7	4.8	489	455	472	455	–3.5	0	878	831	851	830.5	–3.1	–0.1
HHLFLIC	142	128	144	127	1.4	–0.8	851	800	851	800	0	0	1563	1495	1563	1492	0	–0.2
PHLMC8	128	116	134	117	4.7	0.9	879	828	879	828	0	0	1597	1535	1598	1538	0.1	0.2
HHLFLC	146	143	152	133	4.1	–7.0	954	903	971	886	1.8	–1.9	1760	1692.5	1745.5	1653	–0.8	–2.3
DCT32	153	141	149	133	–2.6	–5.7	834	783	817	766	–2.0	–2.2	1505.5	1423.5	1478	1392	–1.8	–2.2
Average △%	0	0	0	0	1.8	–3.3	0	0	0	0	–1.2	–5.4	0	0	0	0	–1.3	–1.8

Acknowledgment

This research was supported by the Natural Science Foundation of Anhui Province (No. 1808085MF203) and the National Natural Science Foundation of China (No. 61432017).

References

[1] J. M. P. Cardoso, P. C. Diniz, and M. Weinhardt, Compiling for reconfigurable computing: A survey, *ACM Computing Surveys*, vol. 42, no. 4, pp. 1301–1365, 2010.

[2] J. W. Yoon, A. Shrivastava, S. Park, M. Ahn, and Y. Paek, A graph drawing based spatial algorithm for coarse-grained reconfigurable architectures, *IEEE Transactions on Very Large Scale Integration Systems*, vol. 17, no. 11, pp. 1565–1578, 2009.

[3] M. Berekovic, A. Kanstein, B. Mei, and B. D. Sutter, Mapping of nomadic multimedia applications on the ADRES reconfigurable array processor, *Microprocessors and Microsystems*, vol. 33, no. 4, pp. 290–294, 2009.

[4] R. S. Ferreira, J. M. P. Cardoso, A. Damiany, J. Vendramini, and T. Teixeira, Fast placement and routing by extending coarse grained reconfigurable arrays with Omega networks, *Journal of Systems Architecture*, vol. 57, no. 8, pp. 761–777, 2011.

[5] R. Krishnamoorthy, K. Varadarajan, and S. K. Nandy, Interconnect-topology independent mapping algorithm for a coarse grained reconfigurable architecture, in *Proc. of 2011 International Conference on Field Programmable Technology*, New Delhi, India, 2011, pp. 1–5.

[6] M. Ahn, J. W. Yoon, Y. Paek, Y. Kim, M. Kiemb, and K. Choi, A spatial mapping algorithm for heterogeneous coarse grained reconfigurable architectures, in *Proc. of the Conference on Design, Automation and Test in Europe*, Munich, Germany, 2006, pp. 363–368.

[7] G. Ansaloni, K. Tanimura, L. Pozzi, and N. Dutt, Integrated kernel partitioning and scheduling for coarse grained reconfigurable arrays, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 12, pp. 1803–1816, 2012.

[8] G. Lee, K. Choi, and N. D. Dutt, Mapping multi-domain applications onto coarse grained reconfigurable architectures, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5,

pp. 637–650, 2011.

[9] M. Jo, D. Lee, K. Han, and K. Choi, Design of a coarse-grained reconfigurable architecture with floating-point support and comparative study, *Integration, the VLSI Journal*, vol. 47, no. 2, pp. 232–241, 2014.

[10] W. Kim, Y. Choi, and H. park, Fast modulo scheduler utilizing patternized routes for coarse-grained reconfigurable architectures, *ACM Transactions on Architecture and Code Optimization*, vol. 10, no. 4, pp. 1–24, 2013.

[11] N. J. Chen and J. H. Jiang, Mapping algorithm for coarse-grained reconfigurable multimedia architectures, in *Proc. of 19th IEEE International Parallel & Distributed Processing Symposium* (*IPDPS*) *Workshop*, Shanghai, China, 2012, pp. 281–286.

[12] S. D. Yu, Research on the software/hardware co-design for reconfigurable processor, PhD dissertation, School of Information Science and Technology, Tsinghua University, Beijing, China, 2009.

[13] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, and E. M. C. Filho, MorphoSys: An integrated reconfigurable system for data parallel and computation intensive applications, *IEEE Transactions on Computers,* vol. 49, no. 5, pp. 465–481, 2000.

[14] N. J. Chen, J. H. Jiang, X. Chen, Z. Zhou, and Y. Xu, An improved level partitioning algorithm considering minimum execution delay and resource restraints, *Acta Electronica Sinica*, vol. 40, no. 5, pp. 1055–1066, 2012.

[15] J. Xiao, Z. H. Shi, W. D. Zhu, J. H. Jiang, Q. W. Zhou, J. Lou, Y. Huang, Q. Ji, and Z. Sun, Uniform non-Bernoulli sequences oriented locating method for reliability-critical gates, *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 24–35, 2021.

[16] Y. M. Ouyang, Q. Wang, Z. Li, H. G. Liang, and J. Li, Fault-tolerant design for data efficient retransmission in WiNoC, *Tsinghua Science and Technology,* vol. 26, no.1, pp. 85–94, 2021.

[17] O. Sangyun, L. Hongsik, and L. Jongeun, Efficient execution of stream graphs on coarse-grained reconfigurable architectures, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 12, pp. 1978–1988, 2017.

[18] I. Bae, B. Harris, H. Min, and B. Egger, Auto-tuning CNNs for coarse-grained reconfigurable array-based accelerators, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2301–2310, 2018.

[19] J. Y. Gu, S. Y. Yin, L. B. liu, and S. J. Wei, Stress-aware loops mapping on CGRAs with dynamic multi-map reconfiguration, *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 9, pp. 2105–2120, 2018.

Naijin Chen received the PhD degree in computer science and technology from Tongji University, Shanghai, China in 2013. He obtained postdoctoral certificate from Tianjin University, Tianjin, China in 2016. He is a member of China Computer Federation. He is currently a professor in Anhui Polytechnic University, Wuhu, China. His current research interests include reconfigurable computing and compiling, fault tolerant computing, reliability evaluation of high-level circuits, approximate computing, formal verification, semantic big data representation and reasoning, and pattern recognition and image processing.

Ruixiang He received the MS degree from Anhui Polytechnic University, Wuhu, China in 2018. He is currently an engineer in Paneng Electric Power Technology Co. Ltd, Nanjing, China. His current research interests include reconfigurable computing and compiling, fault tolerant computing, reliability evaluation of high-level circuits, and approximate computing.

Zhen Wang received the PhD degree in computer science and technology from Tongji University in 2008. She ever worked as a senior engineer in Synopsys from 2008 to 2013. She is a member of China Computer Federation. She is now working in Shanghai University of Electric Power. Her main research interests include fault tolerant computing, reliability evaluation of high-level circuits, and approximate computing.

Jianhui Jiang received the PhD degree in traffic information engineering and control from Shanghai Tiedao University (in April 2000, it was merged to Tongji University) in 1999. Since 2011, he has been the associate dean of the School of Software Engineering, Tongji University. He is a professor and PhD supervisor in Tongji University. He is a senior member of China Computer Federation. His main research interests include reconfigurable computing and compiling, dependable systems and networks, software reliability engineering, and VLSI test and fault tolerance.

Fei Cheng received the BS degree from Anhui Polytechnic University, Wuhu, China in 2019. He is now a master student at School of Computer and Information Science, Anhui Polytechnic University, Wuhu, China. His current research interests include reconfigurable computing and compiling, formal verification, fault tolerant computing, semantic big data representation and reasoning, and pattern recognition and image processing.

Chenghao Han received the BS degree from Suzhou University, Suzhou, China in 2020. He is now a master student at School of Computer and Information Science, Anhui Polytechnic University, Wuhu, China. His current research interests include reconfigurable computing and compiling, formal verification, and fault tolerant computing.