ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.10, No.16, 1–6

Loop acceleration by cluster-based CGRA

Li Zhou[a)], **Hengzhu Liu, and Jianfeng Zhang**

Computer School, National University of Defense Technology, Changsha, China

a) *zhouli06@nudt.edu.cn*

Abstract: This paper presents a cluster-based coarse grained reconfigurable array (CGRA) architecture and a corresponding modulo scheduling method for the inner-most loop. The reconfigurable clusters in this CGRA are composed of generic processing elements (PE) and shared PEs. The local connectivity of a cluster is utilized in the proposed mapping heuristic. Routing in the PE array is avoided because data transmission is within a cluster or between adjacent clusters in the heuristic. Experiment shows that the architecture and method outperform other modulo scheduling algorithms on CGRA. Better execution delay and resource utilization ratio can be achieved at 9.8%.

Keywords: CGRA, modulo scheduling, cluster-based

Classification: Integrated circuits

References

[1] B. Mei, S. Vernalde, D. Verkest, H. D. Man and R. Lauwereins: Proc. 13th Field Programmable Logic and Application (2003) 61.

[2] G. Lee, K. Choi and N. D. Dutt: IEEE Trans. Computer-Aided Design Integr. Circuits Syst. **30** [5] (2011) 637.

[3] Y. Kim, M. Kiemb, C. Park, J. Jung and K. Choi: Proc. 2005 Design, Automation and Test in Europe Conference and Exposition (2005) 12.

[4] T. Oh, B. Egger, H. Park and S. Mahlke: Proc. ACM SIGPLAN/SIGBED 2009 Conference on Languages, Compilers, and Tools for Embedded Systems (2009) 21.

[5] L. Chen and T. Mitra: Proc. International Conference on Field-Programmable Technology 2012 (2012) 285.

[6] B. G. Phillip and S. S. Muchnick: Proc. 1986 ACM SIGPLAN Symposium on Compiler Construction (1986) 11.

1 Introduction

Coarse grained reconfigurable array (CGRA) architecture has gained attention since its emergence. CGRA is composed of reconfigurable processing elements (PE) connected in 2D network similar to ADRES [1]. Figure 1(a) shows a no-clustered typical CGRA. Each PE owns a register file and a context register. PE function units (FU) can be ALU, memory access (LD/ST), multiplier (MUL), or a combination. Data is delivered hop-by-hop with predefined path. CGRA achieves a software-like flexibility while offering parallel

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013

execution capability in different levels. Modulo scheduling is a frequently used technique for implementing loop level parallelism. Unlike in very long instruction word processor, modulo scheduling is more difficult in CGRA because the data should be transferred from the producer PE to the consumer PE. Both scheduling and routing are NP-complete [2]. Thus, several heuristic methods for the mapping problem are studied to exploit the potential capability of CGRA.

A cluster-based CGRA architecture is proposed in this paper. Simple PEs in a reconfigurable cluster shares complex PE, which reduces hardware overhead and improves efficiency. This approach is also convenient for delivering data within the cluster through shared register file. Modulo scheduling utilizes this feature by allowing each cluster execute its own iteration. It avoids routing and lowers the complexity of loop mapping. Experiments on real-life applications indicate that the cluster-based CGRA and method employed in this paper outperforms other modulo scheduling algorithms on CGRA architecture.

2 Cluster-based CGRA architecture

Figure 1 (b) shows the cluster-based CGRA which is constructed by a number of reconfigurable clusters. A cluster is composed of four generic PEs (GPE) and a shared PE (SPE). Each cluster is connected to its neighboring clusters in eight dimensions. GPE implements ordinary arithmetic and logic operations. The source operands of GPE are obtained from SPE or other clusters selected by two multiplexors. The operations performed on ALU cost a single cycle, with a data path width of 16 bits. The width of inter-cluster connection is doubled to 32 bits to provide more transmission bandwidth. The interior of the GPE has no registers and the connection network between GPE is removed, since data are stored and exchanged in the register file of SPE.

Unlike the GPE, complex and area consuming units such as MUL are implemented in SPE. These units are fully pipelined to facilitate operation execution on each cycle. LD/ST unit is placed in SPE to access external RAM. This distributed memory structure makes CGRA scalable. The function units in SPE and the fours GPEs share a 64×16 registered file. Write-enable signals for registers are generated by the write control unit according to the destination operands indicated in the config word. The configuration register loads the config word from the context cache at each cycle, which defines the behavior of the entire reconfigurable cluster.

Two benefits can be obtained by the cluster-based architecture. First, the splitting of ordinary PE and area-critic or delay-critic PE will improve efficiency both in performance and area without much degradation for some applications. Researches have already unveiled the advantages of resource sharing [3]. Second, by partitioning processor resources into several groups or clusters, the PEs in a cluster can be laid out in close proximity, which will reduce data transmission delays. This local connectivity will increase the PE utilization ratio and ease the mapping algorithm if used properly.

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013

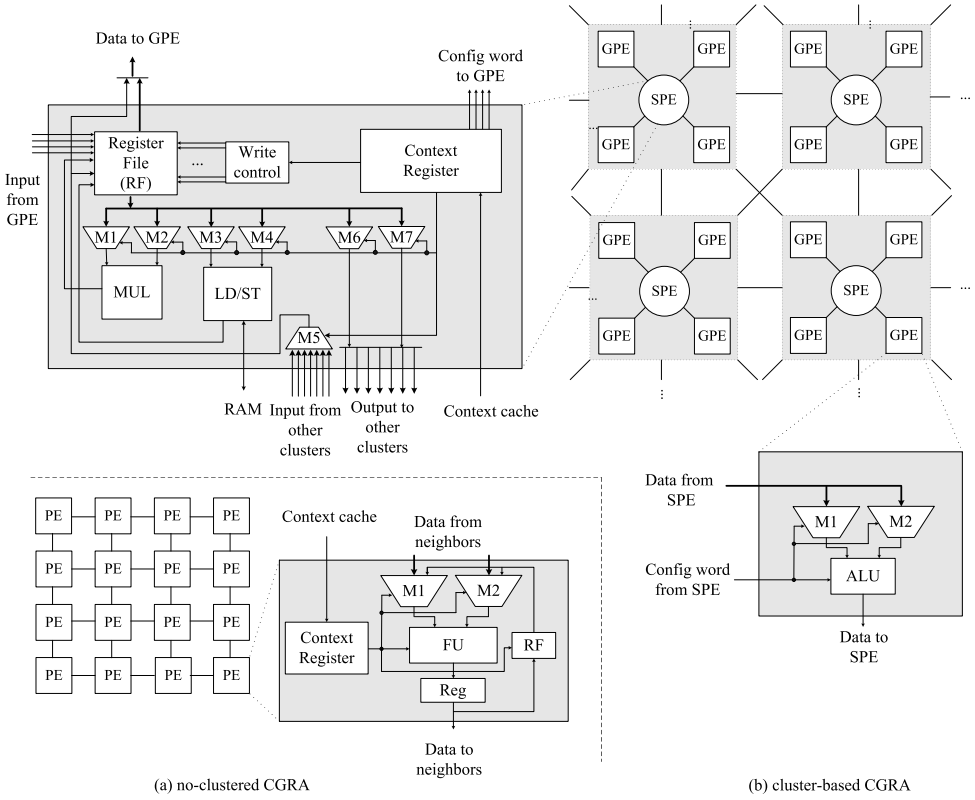

Fig. 1. (a) no-clustered CGRA architecture; (b) cluster-based CGRA architecture

3 Modulo scheduling on cluster-based CGRA

Modulo scheduling identifies an initial interval (II) to launch the iteration repeatedly after the II cycles. The loop kernel is represented by a data flow graph (DFG), and the target CGRA can be modeled by a 2D graph. II is initially set to the minimal II (MII) value between the resource-minimal II (resMII) and the recurrence-minimal II (recMII). Modulo scheduler maps all operations in the DFG to a 3D space, which is an II duplication of the CGRA 2D graph. Failure of the scheduler increases II and arranges another schedule until a feasible solution is achieved. Computation and route resource constraints influence the minimization of II, especially the route resource constraints [4]. Figure 2 (a) provides a simple loop kernel. Figure 2 (b) illustrates its modulo scheduling on a no-clustered CGRA. The $recMII = resMII = 2$. However, mapping the kernel into $MII = 2$ is impossible because of the limitation of routing resource. Four route node were added. Thus, $II_1 = 3$. The loop body runs it times and the latency of the executing iteration is K_1. Total execution time is represented by the following equation:

$$T_1 = II_1 \times (it - 1) + K_1 \tag{1}$$

Let α denote the PE utilization ratio of mapping the kernel into II_1 cycles (route node excluded); $\alpha = \frac{\#\ of\ operations}{\#\ of\ PEs \times II_1}$. α usually ranges from 0.5 to 0.6 [5].

The routing for inner dependencies of the loop body in the cluster-based

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013

CGRA can be trimmed if each cluster is allowed to execute a single iteration. Besides, the loop body can be unrolled until distance of all loop-carried dependency becomes 1. Then, the data only need to be transferred within clusters or between adjacent clusters. Another advantage is the reduced communication between clusters. Source code analysis suggests that the number of loop-carried dependency is lesser than that of inner dependency. Thus, the pressure on inter-connection becomes acceptable. Figure 2 (c) demonstrates this method on cluster-based CGRA. Four PEs are divided into two clusters. The loop kernel is successfully mapped in $II_2 = 2$ without additional route node. The nth and the $n+1$th iteration, as well as the nth and the $n+2$th iteration, can be executed in parallel.

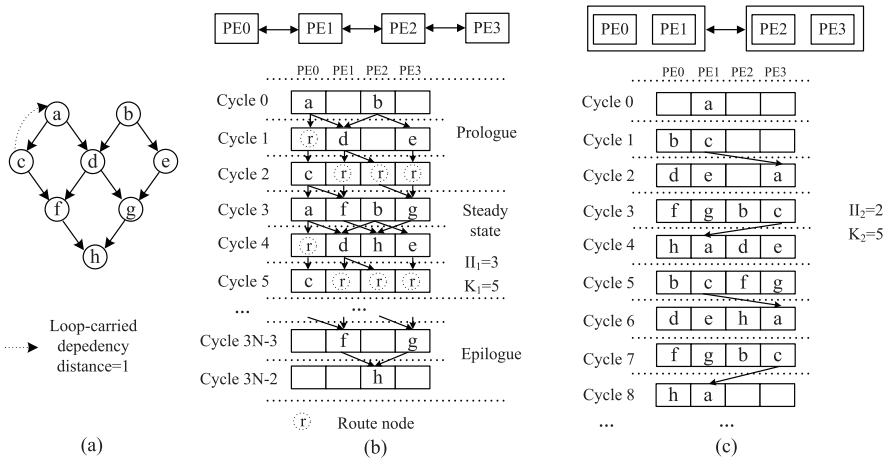

Fig. 2. (a) loop kernel; (b) modulo scheduling on no-clustered CGRA; (c) modulo scheduling on cluster-based CGRA

The total execution time can by calculated by the following equation if M reconfigurable clusters are present and the loop body can be scheduled in K_2 cycles on one cluster:

$$T_2 = K_2 \times \lceil \frac{it}{M} \rceil + II_2 \times (it \ mod \ M) \tag{2}$$

Let β denote the PE utilization ratio of one cluster, $\beta = \frac{\#\ of\ operations}{\#\ of\ PEs\ in\ a\ cluster \times K_2}$. CGRA is composed of M clusters, $\alpha II_1 M = \beta K_2$. Then,

$$T_2 = \frac{\alpha}{\beta} II_1 M \times \lceil \frac{it}{M} \rceil + II_2 \times (it \ mod \ M) \tag{3}$$

where β easily achieves more than 0.67 (concluded from experiments) because no route node is added. A comparison between Eq. (1) and Eq. (3) indicates the high probability of obtaining $T_2 < T_1$ because it is often in hundreds magnitude, whereas M typically ranges from 4 to 16.

The steps of modulo scheduling scheme on the proposed cluster-based CGRA are described below:

1) Unroll the loop until the distance of all loop-carried dependency becomes 1.

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013

2) Extend the kernel L times to obtain a new DFG. The potential parallelism among nth, $n+M$th, $n+2M$th,...$N+(L-1)M$th iterations can be explored by this extension. L in the proposed CGRA is set to the maximum parallelism level of 5.

3) The operations in the recurrence cycle are grouped into a super node. Let $II = recMII$ to perform list scheduling. CGRA contains pipelined function units. Thus, the latency weighted depth is used to define the priority of nodes [6].

4) Check whether the scheduled DFG can start with II. If not, $II = II + 1$, and another list scheduling is set until II becomes feasible.

Operations in the recurrence cycle should be executed in successive cycles when scheduling a super node. $II = recMII$ if the super nodes can be scheduled this way; otherwise, II may be more than $recMII$ because of the increased latency in recurrence cycles. Thus, the validation of II should be checked every time a solution is achieved. When scheduling the extended DFG, the consumer of loop-carried dependency in the $L+1$th extension should be executed $M \times II$ cycles later than the its producer in the Lth extension. Therefore, a new scheduling is required in step 4 once II changed.

4 Experiments

4.1 Experimental setup

A CGRA with four clusters is adopted in the experiments. This method is described it in Verilog HDL and synthesized by Synopsys design compiler using Chartered 90 nm CMOS standard cell technology. The total area of the array (without SRAM) is 1.82 mm^2. Maximal frequency is 667 MHz. A conventional no-clustered architecture is used, as shown in Figure 1 (a). This architecture has 4×5 PEs (4×4 ALU PEs and 1 column of four MUL+LD/ST PEs).

Three actual applications from telecommunication, multimedia, and security domain were selected, namely, WCDMA channel decoder, H.264 decoder, and AES cryptography. A number of loop kernels are extracted for acceleration. Two other modulo scheduling algorithms are also chosen for comparison, namely, the recurrence cycle aware edge centric modulo scheduling (RAMS) algorithm [4] and the graph minor (GM) approach [5]. GM performs better than the simulated annealing method used in ADRES [1].

4.2 Results

Table I shows the execution time (measured in CGRA cycles) and the PE utilization ratio (route node excluded) of the three algorithms on no-clustered and cluster-based CGRA. GM and RAMS degrade when mapping applications onto cluster-based CGRA. This mechanism can be attributed to the competition for inter-cluster connection. The communication between clusters will bottleneck if the limited network resource is not considered. Local connectivity is utilized by confining each iteration into a single cluster, whereas the contention on communication resources between clusters is allevi-

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013

Table I. Comparison on scheduling results

Applications	number of loops	no-clustered CGRA				cluster-based CGRA					
		RAMS		GM		RAMS		GM		Proposed	
		T_1	α	T_1	α	T_1	α	T_1	α	T_2	β
WCDMA channel decoder	24	10505	0.53	8963	0.62	10541	0.53	9152	0.61	8033	0.69
H.264 decoder	61	46379	0.52	40537	0.60	47289	0.51	39965	0.60	36204	0.67
AES cryptography	8	9278	0.55	7998	0.64	9624	0.53	8016	0.64	7291	0.70
Total compilation time (sec)		809		837		806		845		782	

ated. Therefore, 23.6% and 9.8% cycle count is saved compared with the two other algorithms if cluster-based mapping heuristic is applied on the cluster-based CGRA. The complication time of the three algorithms is in the same magnitude. Although we may unroll and extend the loop body more times, the proposed algorithm is still faster because of the routing time saved. This algorithm achieves 68.6% resource utilization on average, whereas RAMS is 52.8% and GM is 61.9%.

5 Conclusion

This paper presents a cluster-based CGRA architecture and a heuristic to map loop kernels. Simple PEs and shared complex PE construct a reconfigurable cluster, and several connected clusters are included in the CGRA. A special modulo scheduling on the cluster-based CGRA is conducted by confining each iteration into a single cluster. The advantages of cluster-based CGRA are fully utilized which facilitate faster loop acceleration on this proposed CGRA by using the cluster-based mapping algorithm. The experiments show that CGRA and the method employed in this paper achieve less execution delay and better resource utilization ratio within less compilation time.

DOI: 10.1587/elex.10.20130506
Received July 01, 2013
Accepted July 17, 2013
Publicized July 25, 2013
Copyedited August 25, 2013