2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery

Mapping DAG to CGRA using Min-max Ant Colony System

Li Zhou, Hengzhu Liu
Institute of Microelectronics and Microprocessor, School of Computer Science
National University of Defense Technology
Changsha, China
e-mail: zhouli06@nudt.edu.cn

***Abstract*—Coarse-grained reconfigurable array (CGRA) architecture has become popular because of its performance and flexibility. The efficiency of CGRA relies on an efficient application mapping algorithm to exploit parallelisms. In this paper, we formalized the temporal mapping problem and proposed the min-max ant colony system (MMAS) algorithm for CGRA mapping. Further optimization of MMAS is studied to reduce mapping time while maintaining the quality of solutions. Comparisons with other heuristic algorithms show that our approach obtains better results in less mapping time.**

Keywords-CGRA; application mapping; MMAS;

I. INTRODUCTION

Coarse-grained reconfigurable array (CGRA) is a highly efficient and flexible architecture for the computation intensive applications. Numerous implementations have been successfully employed in the digital signal processing and scientific domains, including multimedia, software-defined radio, and encryption [1]. These kinds of applications employ a repetitive computation pattern, so CGRA can exploit the inherent parallelism in these applications through a large number of processing elements (PEs) working simultaneously. In addition, the code pattern in these applications often varies because of the different standards and circumstances in which the system works. Thus, hardware devices need programmability to meet the flexibility requirement of such applications. CGRA is composed of an array of reconfigurable PEs, whereby each PE and the interconnected network can be configured at runtime by loading context words from the context cache. Moreover, unlike the case in field-programmable gate array, the granularity of reconfiguration in CGRA is word-level rather than bit-level. This sacrifice makes CGRA more efficient because function units in CGRA are customizable and specific to a certain application domain. Thus, a system constructed by CGRA can accelerate computations by multiple instruction multiple data-style (MIMD) executions and support the flexibility feature required by the application.

Mapping applications onto CGRA is a key factor that greatly influences efficiency. Certain system traits, such as execution latency and power consumption, are highly dependent on the application mapping scheme. Usually, kernels in application are represented by Directed Acyclic Graph (DAG), where nodes denote operations and edges denote data dependency between operations. Mapping includes determining when and where operations can be executed under resource and data dependency constraints. The problem was shown to be NP-complete [2]; thus, finding optimal solutions in a reasonable time becomes difficult. Two types of mapping strategies have been developed: spatial mapping and temporal mapping [3]. In spatial mapping, the configuration of a PE is fixed and the same operation is executed for an application kernel. This mapping style requires kernels to be simple enough such that the operation and data dependency are within the capability of the target CGRA architecture. Applications often need partition procedures to obtain several sub kernels that cannot ensure global optimized result [4]. In the case of temporal mapping, PEs change context dynamically during execution, whereas complex kernels are mapped for the large mapping space in the time dimension. However, temporal mapping is only considered in mapping the innermost loop bodies onto CGRA which is aimed at scheduling and mapping operations in a proper initial interval for performing loop pipelines [5]. A large mapping space greatly increases the complexity in finding a valid solution; thus, only a limited time space, such as the case in loop mapping, is feasible for searching.

In this paper, we propose a mapping algorithm based on the min-max ant colony system (MMAS). The algorithm is a temporal mapping approach that reduces the execution latency of an application running on a given CGRA; thus, optimized solutions can be found within a reasonable time. Compared with other methods, our algorithm has the following advantages

The rest of this paper is organized as follows. Section II provides an overview of existing approaches for scheduling and mapping for embedded systems. Section III shows the detailed mapping algorithm. Section IV presents the evaluations, and Section V concludes the paper.

II. RELATED WORK

Many compile technologies have been proposed with the development of CGRA architecture. Morphosys [6] uses manual tools based on the graphic user interface to map applications. It is a tedious job and does not work with complex designs. Thus, hand optimization cannot satisfy the requirements of efficiency.

978-0-7695-5106-7/13 $26.00 © 2013 IEEE
DOI 10.1109/CyberC.2013.96

M. Ahn [7] formalized the spatial mapping problems and split it into three sub problems. J. W. Yoon [8] noted that the problem of application mapping is similar to that of graph drawing.

B. Mei [9] was the first to introduce modulo scheduling into CGRA mapping. This approach exploits loop-level parallelism through abundant PE resources. The loop pipelining technique in very large instruction word (VLIW) processors is applied in CGRA through module scheduling. H. Park [10] improved the procedure by utilizing edge-centric modulo scheduling. The search is route-aware and focuses on how to assign PEs according to edges rather than on the placement of nodes.

Heuristic algorithms have been addressed in the mapping and scheduling of tasks in multiprocessor system-on-chip (MPSoC). In [11], a mapping technique based on the quantum-inspired evolutionary algorithm was proposed for heterogeneous MPSoC, consequently reducing mapping time to a large extent. In [12], the author applied the ant colony optimization algorithm for task-mapping and scheduling in MPSoC. The algorithm was shown to outperform other traditional techniques. However, mapping and scheduling for CGRA are quite different from those for MPSoC, it has more constraints on mapping and increase the difficulty.

III. MMAS FOR CGRA MAPPING

A. MMAS

Take the traveling salesman problem (TSP) for example, there are n cities, and the distance between cities i and j is d_{ij}. In an ant colony system, several ants are placed in one city to begin the construction of their solutions. Each time an ant stands in the city i, there is a set of candidate cities that is "visible" to the ant, and it will choose to which one city in the set to move forward. The probability of selecting city as its next city is calculated as follows:

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \bullet \eta_{ij}^{\beta}}{\sum_{j \in Candidate} \tau_{ij}^{\alpha} \bullet \eta_{ij}^{\beta}} & j \in Candidate \\ 0 & else \end{cases}$$

τ_{ij} is the global pheromone on the path and it is initialized as a constant. η_{ij} is the expectation of moving from city i to j. In TSP, $\eta_{ij} = 1/d_{ij}$. α and β indicate the importance of the global pheromones and local expectations. After every ant finds its solution, they release pheromones on the traveled path, and updated pheromones remain on the road by $\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \triangle\tau_{ij}(t)$. ρ indicates the evaporation factor of pheromones. Afterward, the ants will start to find their paths again. The algorithm will converge to an optimized solution after several iterations [13]. In order to speed up the convergence procedure, only the ant that finds the best solution releases pheromones in MMAS. The pheromone τ_{ij} is limited within $[\tau_{\min}, \tau_{\max}]$ to prevent falling into a local optimized result. MMAS is proved to be one of the best algorithms for discrete optimization problems [14].

B. Applying MMAS for CGRA Mappin

The key in applying MMAS is to design a way where ants can construct solutions stepwise while creeping through nodes. The solution of CGRA mapping contains $|V|$ elements that each has three attributes: operation, PE, and time. Once an ant moves to a node at step k, the attributes of the k th element is determined, and after $|V|$ steps, the solution is built. The mapping space is infinite, hence, a valid time slot is always found with feasible route if one PE is selected for an operation according to the occupation state of PEs. Thus, nodes that contain operation and PE information are applied in our mapping algorithm.

Figure 1a shows an example DAG and target CGRA architecture with one cluster. Operations v0 and v1 are to be mapped on CPEs, and v2 is to be mapped on SPEs. Figure 2b demonstrates how ants construct solutions by traveling to nodes that contain operations and PE information. A source node is added as the starting point of ants, and all PEs are idle at the start. Ant 1's path implies mapping v1 to CPE1, v0 to CPE3 at cycle 1, and v2 to SPE0 at cycle 2, whereas ant 2's path implies mapping v0 to CPE0 at cycle 1, v1 to CPE0 at cycle 2, and v1 to SPE0 at cycle 3.

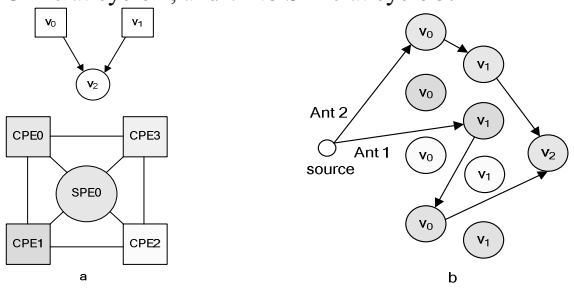

Figure 1. a) Examples of DAG and CGRA composed of one cluster. b)Construction of solutions by the traveling ant.

The local expectation of moving from node i to node j is related to when operation of node j can start execution on its PE, denoted by $startTime(j)$. The earlier an operation starts, the higher the expectation it obtains. We define $\eta_{ij} = 1/startTime(j)$ as the distance in the CGRA mapping. The pseudo code is shown in Algorithm 1.

In the algorithm, each ant maintains an occupy table to record the state of PEs. The table is updated when a new node is selected and the route data transferred. At each step, a candidate set, D, is calculated. The node *d=(op,pos)* can be put into the candidate set only if all parent operations of op have been mapped.

The earliest time that the op can be executed on pos is obtained according to the occupy table. This calculation is done by breath first search (BFS). It starts from the cycle and PE which the parents of op are mapped to. Then BFS iteratively searches the PEs that data can be delivered to at the next cycle until the PE pos reached. Route from data

producer to consumer is found as well as the $startTime(d)$. The BFS ensures that the route is always the shortest.

Algorithm1: MMAS for CGRA mapping

```
Initialize pheromones;
while exploration is not terminated
  for each ant
    Initialize PE occupy table;
    Initialize candidate set D;
    while D is not empty
      for each node d in D
        Calculate the earliest start time of d;
      end for
      Select a node to move onto;
      Update PE occupy table;
      Update candidate set D;
    end while
    end for
    Update pheromones and only the ant who found the
best solution release pheromone;
    end while
```

C. Further Optimization

Due to the large space of the temporal mapping problem, ants always take a long time to find results; most importantly, the BFS performed for each candidate node aggravates this situation. Usually, the search depth of the BFS depends on the location of the parent and child nodes. If the nodes are far away from each other, more steps need to be taken to route from source to sink. Thus, we define a principle for selecting the candidate *d=(op,pos)* as follows: if a parent of *op* is mapped onto PE *pos'*, then the distance between pos and *pos'* must be within a certain range.

Aside from the parent-child relationship, other affiliations influence the mapping result [15]. If two operations have a common consumer, they should be put closer. Thus, we define a second principle for selecting candidate *d=(op,pos)* as follows: if *d'=(op',pos')* has been visited and *op'* has a common child with op, then the distance between *pos* and *pos'* must be within a certain range.

Figure 2 gives a valid candidate selection example when the limited range is just the distance of one cluster, which implies that pos and pos' should be mapped to the same cluster or neighbor clusters; suppose v0 is already mapped to cluster 0. First, v1 is the candidate operation and can be mapped to cluster 0, 1, and 3 only because v0 and v1 have a common child. If cluster 1 is selected to map v1, then v2 is the candidate operation in the next step. According to the principle we defined, the distance between v2 and his parents, v0 and v1, must be within one cluster. Thus, only cluster 0 and 1 are the valid positions for v2.

The optimization of candidate selection greatly reduces searching time and avoids long paths by using simple local heuristics that reduce steps in BFS. The optimization of the result can also be ensured because the abandoned paths usually do not offer a viable mapping solution.

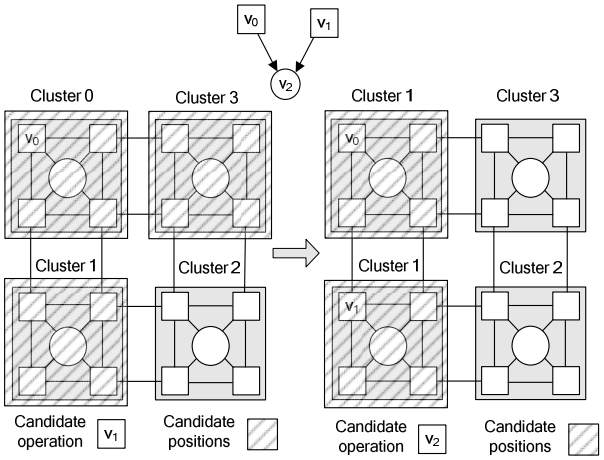

Figure 2. A candidate selection example.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

The validation of our algorithm is done on the CGRA target architecture shown in Fig. 1 and described in section II. We have run the following mapping algorithms:

ILP: The integer linear programming method of mapping based on our problem definition. It finds global optimal solutions for small problems.

MMAS: The proposed mapping algorithm in this work. The parameters we used are α=1, β=2, and the evaporation factor of pheromones is ρ=0.9.

Optimized MMAS (opt MMAS): Further optimization of MMAS shown in section IV is adopted. The two principles that reduce the number of candidates are implemented with MMAS.

Simulated annealing (SA): We designed a simulated annealing approach to map kernels based on the definition of our problem. SA finds results by continuously generating new valid solutions and accepting better solutions, or even inferior solutions, according to a probability function.

The applications we chose came from MIBench and DSPstone, which are frequently used algorithms in the signal processing domain. We identified kernels from these source codes and transformed them into DAG. Loop unrolling was performed for some loop kernels. We ran mapping algorithms five times on each kernel to obtain the average result.

To evaluate the effectiveness and robustness of the algorithms, we used five random generated DAGs (D1–D5) that each contains 20 nodes and 20 data dependencies. Small scale DAGs can be mapped quickly and the global optimal result can be calculated by ILP. Hence, we ran heuristic algorithms 100 times on each DAG and recorded average result of the algorithm's solution as well as the variance of the solutions.

B. Results

First, we evaluated the performance of our algorithm and other heuristic algorithms. ILP cannot map kernels in a reasonable time except for the inner product; thus, ILP's mapping time (>1 day) is not given for other some kernels. Table II shows the best solutions that were obtained using the four algorithms. The algorithms all have an optimized result, and nearly the same optimization effort is achieved.

TABLE I. BEST SOLUTION OF ALGORITHMS MEASURED IN THE CGRA CYCLE

Kernel	ILP	MMAS	optimized MMAS	SA
inner product	6	6	6	6
fir	N/A	19	20	20
idct	N/A	24	24	25
fft	N/A	41	42	44
cordic	N/A	46	46	49

The quality of the result is tested using D1–D5. Table III shows the comparison of mapping algorithms performed on D1–D5. The average cycle measurement and variance of 100 executions are given. The results of ILP are shown to be globally optimal to the mapping problem. The table shows that MMAS is more efficient and robust compared with SA and opt MMAS because of less average cycle counts and the more stable results obtained by MMAS. MMAS achieves a globally optimal result in more cases and has a lower variance. The optimized MMAS is slightly inferior to MMAS but still has a better performance than SA. The optimized MMAS has extra limits on the candidate node; this removal of several valid candidates decreases the degree of randomization in the search and leads to missed possible solutions but the probability is very small.

TABLE II. COMPARISONS ON THE QUALITY OF RESULTS OBTAINED BY 4 ALGORITHMS

Kernel	ILP	MMAS		opt MMAS		SA	
		avr	*var*	*avr*	*var*	*avr*	*var*
D1	6	6.43	0.32	6.48	0.37	6.71	0.43
D2	10	10.64	0.56	10.7	0.56	10.97	0.60
D3	7	7.57	0.42	7.61	0.44	8.43	0.45
D4	9	9.63	0.51	9.65	0.51	10.66	0.54
D5	5	5.26	0.30	5.29	0.33	6.3	0.34

V. SUMMARY AND CONCLUSIONS

In this paper, we presented an MMAS-based algorithm for mapping applications onto CGRA. Given a kernel's DAG and the target CGRA architecture, the proposed algorithm can find an optimized mapping scheme with routed data within a reasonable time. The experimental results show that solutions generated by the algorithm are better and more robust than those of other heuristic algorithms. Several optimizations we made greatly reduced the mapping time without loss of performance.

ACKNOWLEDGMENT

This research was partially funded by National NFSC of China (No. 06970037).

REFERENCES

[1] K. Choi, "Coarse-Grained Reconfigurable Array: Architecture and Application Mapping," IPSJ Transactions on System LSI Design Methodology, vol. 4, pp. 31-46, 2011.

[2] C. O. Shields Jr, Area efficient layouts of binary trees in grids: The University of Texas at Dallas, 2001.

[3] 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, pp. 637-650, 2011.

[4] M. D. Galanis, G. Dimitroulakos, and C. E. Goutis, "Partitioning methodology for heterogeneous reconfigurable functional units," The Journal of Supercomputing, vol. 38, pp. 17-34, 2006

[5] B. R. Rau, "Iterative modulo scheduling," International Journal of Parallel Programming, vol. 24, 1996.

[6] H. Singh, M. H. Lee, G. Lu, et al., "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," Computers, IEEE Transactions on, vol. 49, pp. 465-481, 2000.

[7] M. Ahn, J. W. Yoon, Y. Paek, et al., "A spatial mapping algorithm for heterogeneous coarse-grained reconfigurable architectures," In Proc. DATE'06, 2006 pp. 363-368.

[8] J. W. Yoon, A. Shrivastava, S. Park, M. Ahn, and Y. Paek, "A graph drawing based spatial mapping algorithm for coarse-grained reconfigurable architectures," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 17, pp. 1565-1578, 2009.

[9] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "DRESC: A retargetable compiler for coarse-grained reconfigurable architectures," In Proc. FPT'02, 2002, pp. 166-173.

[10] H. Park, K. Fan, S. A. Mahlke, T. Oh, and H. Kim, "Edge-centric modulo scheduling for coarse-grained reconfigurable architectures," In Proc. PACT'08, 2008, pp. 166-176.

[11] H. Yang and S. Ha, "Pipelined data parallel task mapping/scheduling technique for MPSoC," In Proc. DATE'09, 2009, pp. 69-74.

[12] F. Ferrandi, C. Pilato, D. Sciuto, and A. Tumeo, "Mapping and scheduling of parallel C applications with Ant Colony Optimization onto heterogeneous reconfigurable MPSoCs," In Proc. ASP-DAC'10, 2010, pp. 799-804.

[13] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 26, pp. 29-41, 1996.

[14] T. Stützlea and H. H. Hoosb, "MAX–MIN ant system," Future Generation Computer Systems, vol. 16, pp. 889-914, 2000.

[15] H. Park, K. Fan, M. Kudlur, and S. Mahlke, "Modulo graph embedding: mapping applications onto coarse-grained reconfigurable architectures," In Proc. CASES'06, 2006, pp. 136-146.

[16] T. D. Braun, H. J. Siegel, N. Beck, et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," Journal of Parallel and Distributed computing, vol. 61, pp. 810-837, 2001.