EXACT AND APPROXIMATE TEMPORAL PARTITIONING METHODS FOR DYNAMIC RECONFIGURABLE ARCHITECTURE

Fadel Abdallah[a,*], Camel Tanougast[a,*], Imed Kacem[a], Camille Diou[a], Daniel Singer[a]

[a] *Université de Lorraine, LCOMS, EA 7306, 7 rue Marconi, Metz, F-57070,France*

Abstract

In this paper, we propose an exact method and a genetic algorithm to solve the temporal partitioning problem. The main goal behind the exact method is to divide the complete application (described in the data flow graph: DFG) into sub-applications taking into account the area constraint. These sub-applications are successively performed by the logic area referred through Dynamic Reconfiguration (DR) in order to minimize the total scheduling length (C_{max}). The exact method is compared to a Genetic Temporal Partitioning Algorithm (GTPA), already proposed in a previous work, on a randomly-generated benchmark. The computational experiments validate the hypothesis that the proposed exact method gives optimal solutions in exponential running time. These results show also that the genetic algorithm is better to reach good solutions in less running time.

Keywords: Temporal partitioning problem, Genetic algorithm, Exact method, Data flow graph (DFG), Scheduling, Optimization, Field-Programmable Gate Arrays (FPGA), Real time, Makespan, Dynamic reconfiguration (DR)

1. Introduction

Reconfigurable hardware devices, such as field-programmable gate arrays (FPGAs), are increasingly used and have become ubiquitous, especially in embedded systems that operate under real time constraints. In this paper, we consider the dynamic reconfiguration (DR) that consists in the division and successive execution of an application or algorithm with successive reconfigurations of configurable material part. The advantages of DR have been shown several times in [1, 2]. This approach minimizes the logic area necessary for the implementation of an algorithm or application while increasing the efficiency of silicon processing [3, 4]. This is very important for the development of embedded applications as it allows to avoid oversizing of the required resources for an application and thus optimizes its overall cost. More precisely, the implementation of dynamic reconfiguration consists in identifying the different parts sequentially executed (temporal partitions) according to a scheduling and dependency data.

In the literature, different authors have proposed many approaches and algorithms to solve the temporal partitioning problem [4, 5, 6, 7, 8]. For similar purpose, we have previously proposed a Genetic Temporal Partitioning Algorithm (GTPA) to solve the considered problem in order to optimize the makepsan [5].

In this paper, we propose an exact method for solving the temporal partitioning problem with respect to the physical area constraint and we compare this method to a genetic algorithm. More precisely, the exact method aims to divide the complete application described in the data flow graph (DFG) into sub-applications successively performed

*corresponding author. Tel.: +33 387547136; fax: +33 387547307.

E-mail addresses: fadel.abdallah@univ-lorraine.fr (Fadel Abdallah), camel.tanougast@univ-lorraine.fr (Camel Tanougast), imed.kacem@univ-lorraine.fr (Imed Kacem)

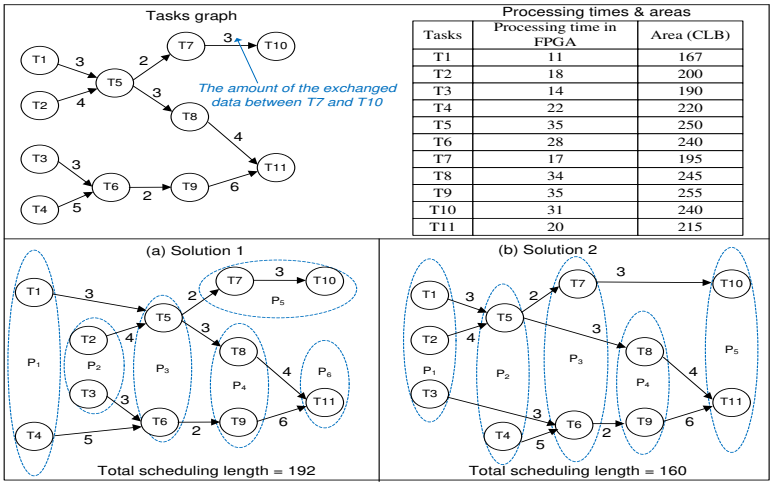
by the logic area referred through Dynamic Reconguration, taking into account the area constraint in order to minimize the makespan. Then, we compare the exact method to the Genetic Temporal Partitioning Algorithm (GTPA, [5]), which is designed to solve the same considered problem.

The rest of the paper is organized as follows. Section 2 illustrates and explains the temporal partitioning problem. Section 3 describe the proposed optimization methods for the considered partitioning problem. Experimental results are presented and discussed in Section 4. Finally, Section 5 concludes the paper and introduces some perspectives.

2. Position and description of the problem

The main goal of this work is to be able to perform on a dynamically reconfigurable FPGA an application, described in a data flow graph, which is too large to be completely mapped on the FPGA. Therefore, the application tasks has to be partitioned in such a way that allows mapping onto the target hardware using the Dynamically Reconfigurable capability (Run Time Reconfiguration - RTR).This section gives an overview about the main problem. The notations of the graph task and the dynamic reconfiguration architecture are presented by considering one specific example for our problem in Figure 1. The first part of Figure 1 presents an application that consists of eleven tasks (tasks graph) to be performed by the hardware FPGA, where the processing time and area required for each task are given in a specific table (processing time & areas).The link from tasks T7 to T10 indicates that the task T10 needs a data size of 3 from task T7, before starting the execution. The time cost to exchange this amount of data depends on the communication rate (we assume that the communication access cost is negligible). To explain this example, let us consider an FPGA area of 550 CLBs (Configurable Logic Blocks), a communication rate of the device FPGA of 2 with reconfiguration time RTR equal to 1.

The second part of the Figure 1 shows two feasible solutions of the considered partitioning problem (in parts (a) and (b)). We have two types of communication cost: intra-partitions and inter-partitions communication. In the first side, we note that in (a) the delay to send three data units from T7 to T10 in the same partition P_5 is 0 + 3/2 = 1.5 time units (access + communication costs). In the other side, each partition can execute when it receives all the data from all predecessor tasks of their own tasks. For example, in the partition P_3 (Figure 1 (a)), the communication cost from all previous tasks of P_3 (T1, T2, T4, T5) to the tasks in P_3 (T5, T6) is the maximum between 3/2, 4/2, 3/2 and 5/2 (which is equal to 5/2 = 2.5 time units). The makespan and number of partitions in (a) are 192 time units and 6, respectively. On the other hand, the makespan and number of partitions for the solution 2 of the same partitioning problem are 160 time units and 5, respectively. Hence, the difference between solutions (a) and (b) both depends on the mapping of tasks graph on the partitions. Indeed, to find the optimal solution, the mapping should be very effective.



Tasks	Processing time in FPGA	Area (CLB)
T1	11	167
T2	18	200
T3	14	190
T4	22	220
T5	35	250
T6	28	240
T7	17	195
T8	34	245
T9	35	255
T10	31	240
T11	20	215

Figure 1. General overview of the considered temporal partitioning problem.

3. Methodology

3.1. Exact method

In this subsection, we will present an exact method for solving the temporal partitioning problem under area constraints. Basically, the proposed method tries to partition the input application described in a data flow graph (DFG) into various partitions of smaller sizes that fit inside the FPGA area in order to minimize the makespan. This method makes it possible to find all the possible solutions (task list + partitioning list) and get the optimal solution as shown in Algorithm 1.

Algorithm 1: Exact method.

1 Generate a set E consisting of all possible solutions;
2 Select the subset F that contains all feasible solutions in set E;
3 Calculate the makespan and the number of partitions of all solutions in subset F;
4 The final solution is the best solution with a minimum makespan in the subset F;

3.2. Genetic Temporal Partitioning Algorithm (GTPA, [5])

In this subsection, we summarize our previous GTPA that proposed in [5] to solve the temporal partitioning problem in order to minimize the makespan. This algorithm inherits almost the same steps from the genetic algorithms (GAs). Figure 2 illustrates the different typical phases of GTPA.

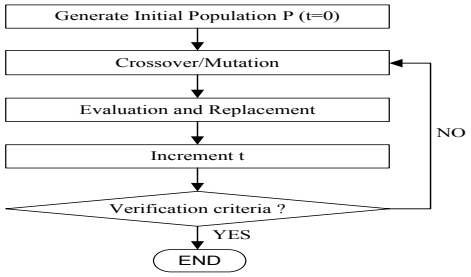


Figure 2. Flowchart of Genetic Temporal Partitioning Algorithm (GTPA, [5]).

4. Computational results and discussions

The exact method has been tested with a number of project examples in order to evaluate their performance. A project consists of n tasks with data dependencies (precedences) and where the number of tasks n varies from 5 to 100 with different graph topologies, as mentioned in the Table 1. We have developed the exact method using the C language. This method has been tested on a machine equipped with 8 Intel processor i7-4700MQ cores and 8GB RAM memory. The operating system is a 64-bytes Windows 8 Pro.

Table 1. Set of data used for simulation.

Data Flow Graph	Nodes (n)	Edges ($\|A\ (G)\|$)
DFG1	5	4
DFG2	10	22
DFG3	20	29
DFG4	30	67
DFG5	50	65
DFG6	100	145

To assess the performance of our proposed method, the running time was fixed to 20 minutes. Then, the best solution in terms of makespan was taken as a final solution.

4.1. Comparison results between the proposed GTPA and Exact Method

In this subsection, the proposed exact method are compared with the GTPA [5] in terms of performance. The GTPA gives always approximate solutions that can be optimal in case of small instances [5]. However, the exact method provides always the optimal solutions. For example, in DFG1 and DFG2 both algorithms gave optimal results. In this case, the exact method ends the running time in less than 20 min. However, for the other instances, the running time is stopped at 20 min, and the obtained results were approximate and not optimal. Similarly using the GTPA, the corresponding solutions were approximate and calculated in a short running time. We can see in this case that the GTPA gives better results than the exact method, in less running time. Hence, it offers a good tarde-off between the quality of the solution and the running time compared to the exact method.

Table 2. The results obtained to study the performance of proposed algorithm.

	GTPA			Exact method		
DFG	C_{max}	number of partitions	Time (ms)	C_{max}	number of partitions	Time (ms)
DFG1	61.00	4	1.00	61.00	4	15.585
DFG2	159.00	7	94.98	159.00	7	70478.406
DFG3	307.00	14	4989.65	326.00	15	1200000.00
DFG4	402.00	19	3590.46	440.00	21	1200000.00
DFG5	692.50	36	15572.03	766.50	38	1200000.00

5. Conclusions and Future Work

In this paper, we have considered the temporal partitioning problem existing in FPGA under area constraints, where the objective is to minimize the makespan. We have proposed an exact method to solve the given problem. Furthermore, the conducted computational experiments permits the evaluation of proposed exact method as well as comparison with the GTPA. This study reveals that the GTPA is effective enough to find the approximate solution in a significantly less running time than the exact method. To conclude, the exact method cannot offer exact optimal solutions in a resonable time and we can observe the GTPA is able to solve effectively various partitioning and scheduling problems.
As future work, we can compare the proposed algorithms to other existing methods to evaluate their effectiveness. In addition, different metaheuristic techniques can be used to solve temporal partitioning problems.

References

[1] Wirthlin, M.J., Hutchings, B.L. 1998. Improving functional density using run-time circuit reconfiguration [FPGAs], *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(2), pp 247-256.

[2] Tanougast, C., Berviller, Y., Weber, S., Brunet, P. 2003. A partitioning methodology that optimises the area on reconfigurable real-time embedded systems, *EURASIP Journal On Applied Signal Processing (J.A.S.P.), Special Issue on Rapid Prototyping of DSP Systems*, 2003(6), pp 494-501.

[3] Brunet, P., Tanougast, C., Berviller, Y., Weber, S. 2003. Hardware partitioning software for dynamically reconfiureable SoC design, *in Proc. the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, IEEE circuits and Systems Societys Techincal Committee on VLSI and on Communication*, pp 106-111.

[4] Tanougast, C., Berviller, Y., Brunet, P., Weber, S., Rabah, H. 2003. Temporal partitioning methodology optimizing FPGA resources for dynamically reconfigurable embedded real-time system, *Microprocessors and Microsystems*, Elsevier 27(3), pp 115-130.

[5] Abdallah, F., Tanougast, C., Kacem, I., Diou, C., Singer, D. 2016. Temporal partitioning optimization for dynamically reconfiguration architecture, *In Proc. of 46th International Conference on Computers & Industrial Engineering*, pp. 643-650, Tianjin, China.

[6] Ouni, B., Ayadi, R., Mtibaa, A. 2011. Combining temporal partitioning and temporal placement techniques for communication cost improvement, *Advances in Engineering Software*, 42(7), pp 444-451.

[7] Ouni, B., Ayadi, R., Mtibaa, A. 2011. Temporal partitioning of data flow graph for dynamically reconfigurable architecture, *Journal of Systems Architecture*, 57(8), pp 790-798.

[8] Wu, G-M., Lin, J-M., Chang,Y-W. 2001. Generic ILP-based approaches for time-multiplexed FPGA partitioning, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(10), pp 1266-1274.