Optimization of logic area for System on Programmable Chip based on hardware-software partitioning

Mehdi Jemai, Sonia Dimassi, Bouraoui Ouni and Abdellatif Mtibaa

Laboratory of Electronic and Microelectronic, Faculty of Science at Monastir

Monsatir 500, Tunisia

jmehdie@gmail.com, sdimassi@yahoo.com, ouni_bouraoui@yahoo.fr, abdellatif.mtibaa@enim.rnu.tn

***Abstract*—T In this paper, we propose an approach based on hardware-software partitioning to minimize logic area of a SOPC circuit "System on a Programmable Chip". This approach minimizes the SOPC area while satisfying a time constraint. To minimize this area, we propose an algorithm to determine the critical path with the largest number of hardware tasks in a given data flow graph. Once these hardware tasks are determined, they will be implemented on the software. In this way we minimize the number of tasks used by the HW and increase the number of tasks used by the SW, where we have a minimization of the area.**

Index Terms— **Logic area, hardware-software partitioning algorithm, temporal constraint, SOPC.**

I. INTRODUCTION

A system on programmable chip SOCP is a circuit comprising multiple functions such as one or more processors, one or more reconfigurable areas, signal processor DSP (*Digital Signal Processor*), various peripherals and memory or analog parts. These circuits are increasingly used because of their small size and reduced costs compared to the use of various circuits for performing the same function. Using SOPC is increasingly common in embedded applications. Therefore, many hardware and software techniques must be developed to satisfy specific constraints in terms of area, performance, power consumption, etc. Thus, in this paper, we present an effective approach based on hardware-software partitioning to implement a control data flow graph on SOPC circuit while minimizing the logic area. In this paper, firstly we have implemented all tasks of the graph on the hardware part of architecture. However, implementation of the hardware modules may degrade the design in terms of area. Hence, to reduce the area, we have implemented the tasks of the hardware part on the software part of architecture. Therefore, the main objective of our hardware/software partitioning approach is to balance all the design parameters to find a better compromise between the logic area of the application and its execution time.

This paper is structured following six parts. After the introduction, we give an overview of the related work; in the third section, we present the hardware/software partitioning model. The fourth section shows our partitioning algorithm. In the fifth one, we present the experiments and results. Finally, we end up with a conclusion.

II. RELATED WORK

Cutting or hardware/ software partitioning is an important phase in the system design (and especially the architecture exploration phase) and is to seek the best compromise software/hardware. It is in this phase that are made choices leading to a realization hardware or software of the various constituent parts of the system. In general, the software is used to reduce design costs and equipment to increase performance. Many techniques and algorithms have been proposed to assist the designer in this task. The ultimate goal is of course to automate this task.

Behavioral specification is divided into sub-functions, the choice of their implementation, hardware or software, is posed. The question that arises is how to choose between software and hardware? The choices made in the partitioning step are definitive choices, which are not challenged in the following design steps. A bad choice at this level then requires restarting the design cycle of partitioning until the co-simulation. We must therefore pay great attention to this step.

In the context of the system design, the software may be defined as "a sequence of operations running on a programmable machine necessary for the functioning of a set of information processing". Programmable machines on which is running the software are of different types: general processor, signal processor (DSP), etc. The material is defined by opposition to the software. It is "a non-programmable physical structure." Its functionality is fixed in the design. It is not possible as for the software to make a change for a new feature. ASIC (*Application-Specific Integrated Circuit*) is hardware because this particular circuit is designed for a given application.

In system design, an optimization method generally involves applying an optimization algorithm on the set of sub-

functions of the specification. We then seek to minimize a criterion (or set of criteria) given as the surface, the execution time, consumption, etc. The partitioning algorithm is an optimization algorithm (in our case minimization) seeking one or more achievements optimized for a given problem. In our approach, we seek to minimize the logic area according to a time constraint. We also find a comparison of several minimization algorithms applied to software/hardware partitioning [1].

In the literature, different research has already been made on the partitioning software/hardware; we note that they have treated the problem of reducing the overall cost of the application in term of hardware resources or the improvement of its performance in term of execution time. In fact the implementation of a software module requires more flexibility and less cost, but more executing time, and vice versa in case of hardware. Therefore, the main goal of most approaches of hardware/software partitioning is to balance all the parameters to find a compromise between cost and application execution time. In this context, we quote the exact algorithms that based on: integer linear programming [2], dynamic programming [3] and branch and bound [4]. However, exact algorithms are very slow and can be applied only for small size graphs. Thus, to overcome the drawbacks of exact algorithms, researchers have tried heuristic algorithms that are more flexible and efficient as network throughput [5], simulated annealing [6], tabu search [7], genetic algorithm [8], combined algorithm [9] and greedy algorithm [10]. As mentioned earlier these optimization algorithms seek one or more achievements optimized for a given problem. In this paper, we have proposed an algorithm for hardware/software partitioning that minimizes the logic area of SOPC circuit.

III. Hardware/Software Partitioning Model and Formulation

The HW/SW partitioning model defined in this section considers the following characteristics: granularity, metrics associated with the functional blocks, computational model, representation of the solution and the cost function.

The behavioral description given in a high-level language is transformed into a control data flow graph. Each node $v_i \in V$ corresponds to a part of the application that itself belongs to the base granularity (a single instruction or a basic block). Hence, a control data flow graph G (V; E) is a directed acyclic graph describing the dependencies between the operations of an application. Where $V= \{v_1, v_2, \dots, v_n\}$ is the set of nodes, n is the number of nodes and E is the set of edges $\{e_{ij} | 1 \leq i, j \leq n\}$. The critical path, CPg, of a graph G is a path from its source node to its sink node.

Once the system is represented under this model, values for the metrics are associated to each node v_i. The following metrics are used: software latency ($L_S(v_i)$), occupied hardware area in slice ($A(v_i)$), and the hardware latency ($L_H(v_i)$).

In this model, a partitioning solution is expressed as an indicator vector X_m that is defined as follows: $X_m=X_m(i)$; Where: $i \in [1,n]$ and X_m (i) = 1, if node (i) will be implemented in hardware; however, X_m (i) = 0, if the node (i) will be implemented in software.

Hence, our optimization problem can be modeled as follows:

$$\begin{cases} minimize \ \sum_{v_i \in G} X_m(i) A(v_i) & (1) \\ subject\ to\ L(G) \leq T & (2) \end{cases}$$

IV. The Algorithm

In this section, we propose an algorithm to solve the partitioning problem. The overall methodology of the proposed algorithm is illustrated in figure 1. The methodology starts by transforming Control Data Flow Graph G (V, E) to the Data Flow Graph G' (V, E), and assign all nodes of the transformed graph G' (V, E) to the hardware part of the architecture. Then we finds the critical path (CPg) in G' (V, E) that has the highest cost function related to the hardware area in slice, and assigns all nodes of this critical path to the software part of the architecture. However, we calculate the latency of the graph related to the new indicator vector X_m. In each iteration, the latency of the graph of each indicator vector X_m is evaluated until the latency outdated the temporal constraint T. The HW/SW partitioning solution is the penultimate indicator vector. The detail of the algorithm is shown in figure 2.

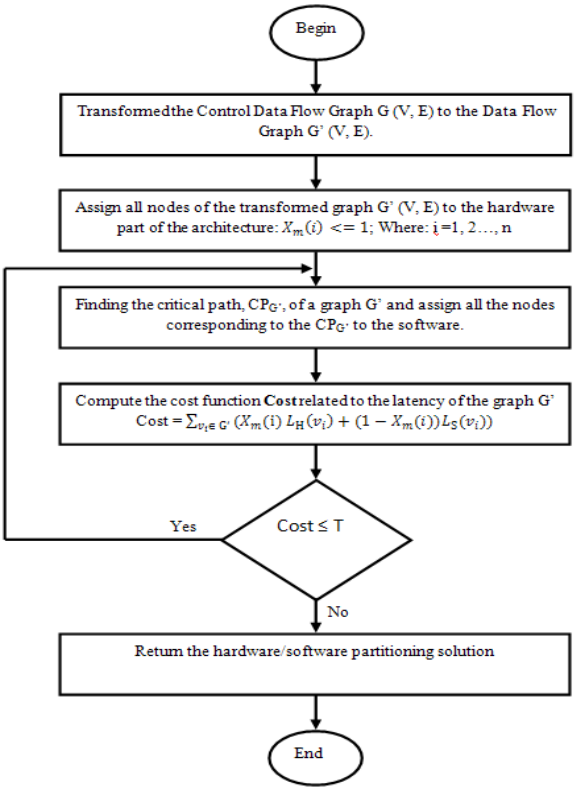

Fig.1: Main procedures for the proposed algorithm

1: **Begin**
2: Transformed Control Data Flow Graph G (V, E) to the Data Flow Graph G' (V, E).
3: Assign all nodes of the transformed graph G' (V, E) to the hardware part of the architecture $X_m(i) <= 1$; where: $i \in [1, n]$
-: // X_m *is the indicator vector and* n *is the number of nodes* //
-: ///– *Finding the critical path,* CP_G, *of a graph G'* ///
4: $CP_G = \{ v_0 \}$;// v_0 *is the source node of graph G'*//
5: **For all** $v_i, v_j \in V$:
6: **While** $((v_j) \neq v_f)$ **loop** // *where the* v_f *is the sink node of graph G'*//
7: **If** $\forall_{v_i, v_j \in V} (\beta_{ij} = 1)$ **then**
8: $CP_G <= CP_G + \{v_j\}$;
9: **End if**;
10: **End loop**
11: Return max $(A(CP_G) = \sum_{v_i \in CP_g} X_m(i)\, A(v_i))$
12: **End for**
13: **For all** $v_i \in CP_G$
14: $X_m(i) <= 0$;
15: **End for**
16: **For all** $v_i \in V$:
17 $X_m(i) <= X_m(i)$; // *Update the indicator vector Xm*
18: **End for**
19: Compute Cost= $\sum_{v_i \in G} (X_m(i)\, L_H(v_i) + (1 - X_m(i)) L_S(v_i))$
20: **If** (Cost ≤ T) **then**
21: S<=X_m;
22: go to the line 5;
23: **Else**
24: go to line 26
25: **End if**;
26: Return the partitions solution of original graph G (V, E) extracted from the transformed graph G' (V, E).
27: **End begin**

Fig.2: Pseudo-code of our algorithm

V. EXPERIMENTS AND RESULTS

To confirm our approach, we have implemented the 16-DCT task graph on FPGA Xilinx Virtex®-5. The Xilinx Virtex®5 development kit enables high performance embedded design in Xilinx FPGAs. In our approach software resource is the PowerPC and the hardware resources are configurable logic blocs (CLBs). Hence, to compute the parameters of each node and to access to the PowerPC, we have used Xilinx ISE tool and Xilinx EDK tool. These Xilinx design tools provide resources and timing report incorporates timing delay and resources to provide a comprehensive area and timing summary of the design. Our algorithm has been written in JAVA language and executed under Windows-7 on Acer-PC (Intel Core 2 Duo T5500; 1, 66 GHz; 1GB of RAM). In order to demonstrate the effectiveness of the proposed algorithm, we compare it with simulated annealing and genetic algorithm. The simulation results are presented in the table 1.

Table 1: DESIGN RESULTS

Algorithm	Run time (ms)	Latency (ns)	Area (Slice)
Proposed algorithm	1.377	830	5215
simulated annealing algorithm	1.723	759	5724
Genetic algorithm	1.882	810	5460

As shown in the experiment results, we can find that the genetic algorithm needs more time, which means more iteration to satisfy the temporal constraint. Furthermore, the hardware area get by the simulated annealing algorithm is higher. Therefore, we conclude that our proposed algorithm produces low hardware area, and generates the solution at a faster speed.

VI. CONCLUSION

Partitioning software/hardware has become a marginal, even with some work. Many methods and algorithms have emerged in the late 90s and these works do not provide satisfactory solutions. The evolution of design, synthesis and compilation, the characteristics of the components and the complexity of applications and architectures are certainly responsible for the ineffectiveness of solutions in this field. In this paper we have presented a new approach based on high level hardware/software partitioning to reduce logic area. Our partitioning algorithm has been tested and compared to simulated annealing and genetic algorithm. Design result, shows that our approach provides better design results in term of logic area.

REFERENCES

[1] M. LOPEZ, J. LOPEZ., "On the Hardware-Software Partitioning Problem: System Modeling and Partitioning Techniques", ACM Transactions On Design Automation of Electronic Systems (TODAES), vol.8, n° 3, pp. 269-297, juillet 2003.

[2] S. Banerjee, E. Bozorgzadeh, and N. D. Dutt, "Integration physical constraints in hw-sw partitioning for architectures with partial dynamic reconfiguration," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no. 11, pp. 1189 -1202, nov. 2006.

[3] J. Wu and T. Srikanthan, "Low-complex dynamic programming algorithm for hardware/software partitioning," Information processing letters, vol. 98, no. 2, pp. 41- 46, 2006.

[4] K. Chatha and R. Vemuri, "Hardware-software partitioning and pipelined scheduling of transformative applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 3, pp. 193-208, 2002.

[5] H. Liu and D. F. Wong, "Efficient network flow based multiway partitioning with area and pin constraints", IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 17, no. 1, pp. 50–59, Jan. 1998.

[6] Z. Peng and K. Kuchcinski, "An Algorithm for Partitioning of Application Specific Systems," Proc. European Conf. Design Automation(EDAC'93),pp.316-321, 1993.

[7] T. Wiangtong, P.Y.K. Cheung, and W. Luk, "Comparing Three Heuristic Search Methods for Functional Partitioning in Hardwaresoftware Codesign," Design Automation for Embedded Systems, vol. 6, no. 4, pp. 425-449, 2002.

[8] Hong jun He; Qiang Dou and Weixia Xu , "Hardware/Software Partitioning for Heterogeneous Multicore SoC Using Genetic Algorithm", IEEE Intelligent System Design and Engineering Application (ISDEA), pp 1267 – 1270, Jan. 2012

[9] Yu Jiang, Hehua Zhang, Xun Jiao, Xiaoyu Song, William N. N. Hung, Ming Gu, and Jiaguang Sun," Uncertain Model and Algorithm for Hardware/Software Partitioning", IEEE Computer Society Annual Symposium on VLSI,2012.

[10] K.S. Chatha and R. Vemuri, "Magellan: Multiway Hardware-Software Partitioning and Scheduling for Latency Minimization of Hierarchical Control-Dataflow Task Graphs," Proc. Ninth. Hardware/Software Codesign (CODES '01), pp. 42-47, 2001.