Hardware/Software Partitioning based on Simulated Annealing

David Sánchez[1], Juan P. Castellano[1], Álvaro Suárez[1]

Abstract

A software implementation often can not satisfy embedded systems timing constraints. This problem can be solved by adding specific hardware to the system. Lately, it has been developed some design methodologies for this type of hardware/software systems. Our research group is developing a hardware/software codesign environment for designing this type of systems. In this paper, we present our Hw/Sw partitioning algorithm that is based on simulated annealing. Main contribution is the following: it supports process-level pipelining and estimates system power consumption. Thus, system designer can explore the design space to make latency, area and power trade-offs.

Introduction

In the last years, embedded systems are present in an ever increasing number of applications. In general, a software implementation does not satisfy the timing constraints of these systems. This problem can be solved by adding specific hardware to speed up system execution time. Lately, it has been developed some design methodologies for this type of hardware/software systems. They are named as hardware/software codesign methodologies [1][9].

Our research group is developing a codesign environment named GACSYS (*GAC's Codesign System*) [4][5] (Figure 1). At present, we have mainly developed the following work: a) we have designed a new system specification language named VSS (*VHDL-based System Specification*). Our language supports high level statements to make easy system specification. VSS allows a designer to decrease system specification time. We have also developed a compiler to translate from VSS code to VHDL one. b) We have designed an intermediate representation to support VHDL specifications. It is based on the ASCIS (*Architectural Synthesis of Complex Integrated Systems*) data flow graph. We have also developed a compiler to generate the intermediate representation from VHDL code. c) We have developed a hardware/software partitioning tool that allows a better design space exploration.

In this paper, we present our hardware/software partitioning tool. Partitioning is one of the most important phases of a codesign environment. It assigns a hardware or software

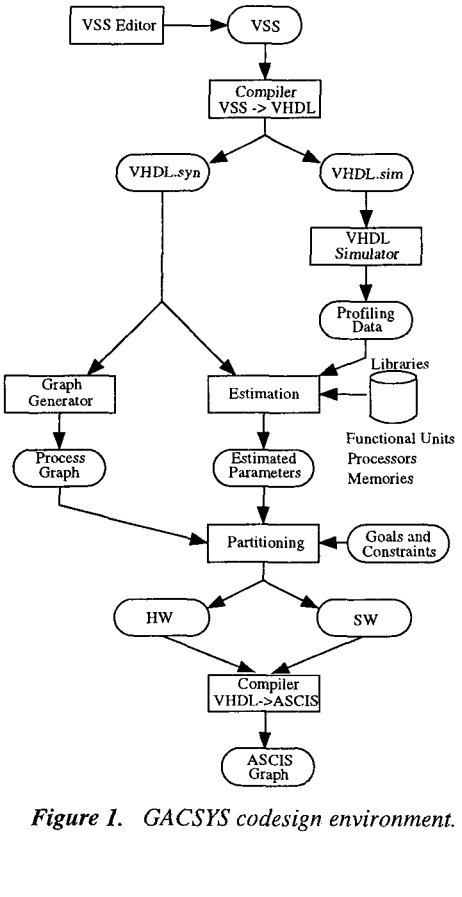

Figure 1. GACSYS codesign environment.

process_1

Software estimations /* Pentium */

T (seg)	A (mm^2)	E (mj)
6.82e-7	2.96e+2	1.09e-2

Memory : RAM_3

Hardware estimations

Memory : RAM_1

3 /* Number of VHDL operators */

Functional Unit Configurations

N°	Code	T (seg)	A (mm^2)	E (mj)
* + >	* + + <			
1 1 1	3 1 5	1.5e-6	7.1e-1	1.9e-5
1 1 1	3 2 5	1.6e-6	7.4e-1	1.3e-5
1 2 1	3 1 1 5	1.9e-5	8.6e-1	1.9e-5
1 2 1	3 1 2 5	2.0e-5	8.6e-1	1.6e-5
1 2 1	3 2 1 5	2.0e-5	8.6e-1	1.6e-5
1 2 1	3 2 2 5	2.1e-5	8.6e-1	1.3e-5

Figure 2. Estimation phase data.

implementation to each process of the system. Partitioning goal is to optimize design goals and satisfy design constraints. Main contribution of our partitioning tool is the following: a) it supports process-level pipelining, b) it estimates design power consumption. These features allows a better design space exploration. Previous tools do not support these features. We have developed a partitioning algorithm that is based on simulated annealing.

The structure of this paper is the following: in the second section, we present related work. In the third section, we describe our hardware/software partitioning tool. In the fourth section, we show some experimental results. Finally, we present some conclusions.

Related work

We can classify partitioning tools according to representation model: a) process DAG (*Directed Acyclic Graph*) [17], b) control and data flow graph (CDFG) [11], c) other types

of graphs [16]. Partitioning tools can be grouped into three classes according to grain (node) size: a) process [6][17], b) subprogram [14], and c) basic block [11].

Optimization algorithm can also be used to group partitioning tools. On one hand, there are tools that generate an optimal hardware/software partition by using algorithms such as ILP (*Integer Linear Programming*) [13], *branch & bound* [6], and dynamic programming [11]. However, these algorithms are not suitable for complex systems because they need an excessive computation time. On the other hand, there are tools based on heuristic techniques such as *clustering* [15][17], group migration [3], and *simulated annealing* [2]. Partitioning tools define a cost function to optimize the partitioning task. We can identify two groups according to cost function parameters: a) some tools only use system execution time [2][14], and b) other tools also consider design area [3][17]. Anyway, previous tools do not consider system power consumption during partitioning process.

Main features of our tool are the following: *a) representation model:* process DAG, *b) granularity:* process-level, *c) optimization algorithm:* simulated annealing, *d) cost function parameters:* design execution time, area and power.

Hardware/software partitioning

Our partitioning tool assigns a hardware or software implementation to each process of the input specification. Partitioning goal is to optimize design goals and satisfy design constraints. Partitioning input data are the following: a) system process graph (a node represents a process and an edge specifies process data flow), b) hardware and software parameter estimation such as system execution time, area and power, c) design goals and constraints (Figure 1).

Hardware and software parameter estimation is generated by using a tool that we have developed [4]. This tool gives us the following information (Figure 2): 1) process name: next information corresponds to this process. 2) Software parameter estimation: estimated execution time (T), area (A) and energy (E). 3) Memories that have been used. 4) Number of different VHDL operators. 5) Functional unit configurations that can be used to implement the process in hardware. Configuration data is the following: a) number of functional units to implement VHDL operators, b) functional unit code, c) estimated system execution time (T), area (A) and energy (E).

Designer has to specify the following input parameters (Figure 3): a) maximum time to compute the process graph (T_E), b) number of pipeline phases (latency), and c) design area or power upper bound. Given these parameters, design goals and constraints are the following: *Goals*: a) execution time of a pipeline phase must be as close as possible to maximum execution time. In this way, we avoid designs with unnecessary extra cost. b) We must optimize design area or power. *Constraints*: a) execution time of a pipeline phase can not be greater than maximum execution time. b) Design area or power upper bound must be satisfied.

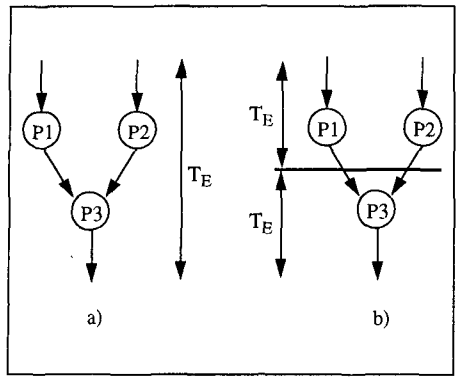

Figure 3. *(a) Graph without pipelining, (b) graph with 2-phase pipelining.*

$$T = \sum_{i}^{Phases} (T_E - T_{Fi})$$

Expression 1. *Execution time that is not used in pipeline phases.*

$$T_{Fi} = \sum_{Pi} max\{max\{T_{Hw}\}, \sum T_{Sw}\}$$

Expression 2. *Execution time of a pipeline phase.*

Main contribution of our partitioning tool is the following: a) *it supports process-level pipelining* (Figure 3). Previous work on partitioning does not consider this feature. We allow a designer to make a larger number of design cost (area or power) and latency tradeoffs. b) *It estimates design power consumption*: this parameter has not been considered in previous codesign environments. We have supposed an implementation with CMOS technology. Thus, main power consumption is due to switching power. We have also supposed power-down techniques (clock signal of some latches are disabled to avoid power consumption when a functional unit does not generate a useful result in a given cycle) [12].

Cost function of our partitioning tool makes use of the following parameters: a) design area (A) and power consumption (P). b) Design execution time (T) that is not useful in each pipeline phase (Expression 1). This value is obtained by substracting maximum execution time (T_E) and pipeline phase execution time (T_F). Given a pipeline schedule and a hardware or software process configuration, execution time of a pipeline phase depends on process schedule. We have named this scheduling process as virtual scheduling (Figure 4).

Given a virtual schedule, Expression 2 shows how to evaluate execution time of a pipeline phase (T_{Fi}) (*Pi* represents one step of the virtual schedule). In general, a virtual schedule step can have hardware and software processes. Thus, execution time of a virtual step is the maximum value among the following ones: 1) slowest execution time among hardware processes ($max\{T_{Hw}\}$). 2) Software execution time. It is evaluated by adding each process execution time because we only have one processor (ΣT_{Sw}).

Expression 3 shows the cost function that our partitioning algorithm uses. P_{min} and A_{min} represent minimum design power and area, respectively. Parameters are normalized by dividing them with corresponding value ranges. α weight allows a designer to give priority to design area or power.

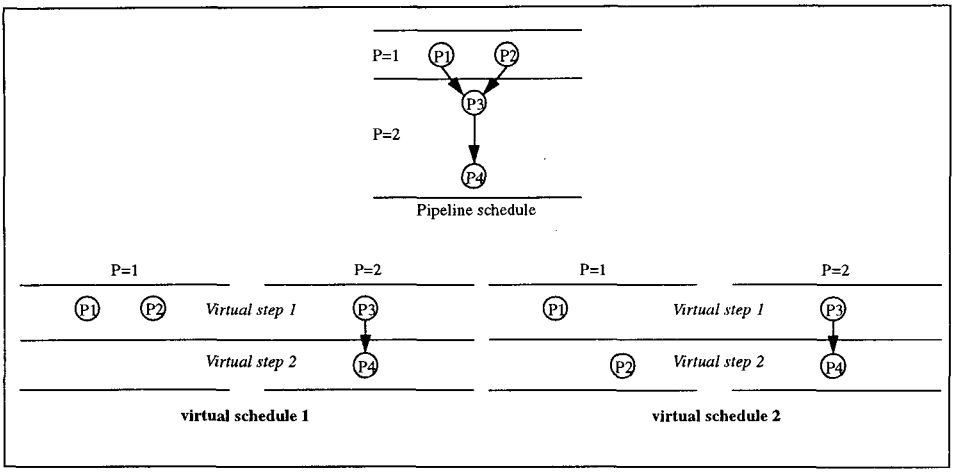

Figure 4. *Example of different virtual schedules.*

$$COST_P = \alpha \frac{P - P_{min}}{Range\,(P)} + (1-\alpha)\frac{T}{Range\,(T)}$$

$$COST_A = \alpha \frac{A - A_{min}}{Range\,(A)} + (1-\alpha)\frac{T}{Range\,(T)}$$

Expression 3. *Cost function to optimize power ($COST_P$) or area ($COST_A$).*

Hardware/software partitioning is a NP-complete combinatorial optimization problem [8]. In general, we can not use methods that give us an optimal solution because they need an excessive computation time. Heuristic methods are an alternative to solve this problem. Our partitioning algorithm is based on simulated annealing [7][10]. We select this method following experimental results published in [16], where it is concluded that simulated annealing gives better results than other heuristic techniques do such as clustering or group migration.

Now, we describe main steps of our algorithm. First, algorithm generates an initial solution that has two components: a) initial pipeline schedule, b) hardware/software process configuration. Initial schedule is obtained by using an ASAP-based algorithm (*As Soon As Possible*). Initial configuration is randomly generated. Once we have an initial (current) solution, annealing process is started. Algorithm selects a neighbour solution of the current

one. It is randomly generated by selecting a process. Then, selected process is randomly changed from current pipeline phase to a new one. However, process phase change is not carried out each annealing process iteration. In this way, pipeline schedule remains stable various iterations. It allows remaining processes to adapt itself to process phase change. Finally, if neighbour solution cost is lower than current solution is, then first one is accepted as new current solution. On the other case, there is still a possibility to accept it that depends on both solution cost and annealing process temperature.

Experimental results

In this section, we present some experimental results. We have used a voice recognition system as example. This system generates a set of coefficients from a voice signal that is sampled at 8 KHz and packed in frames of 128 bytes. Thus, maximum graph execution time is 1.6E-2 seconds ($128 \cdot 8000^{-1}$). System specification has eight processes (process graph is illustrated in Figure 5) and 2670 lines of VHDL code.

Now, we show some results to illustrate design space exploration. On the one hand, designer can make area and power tradeoffs. Figure 6 shows different design space solutions. It can be observed that designs with higher latency have a lower power consumption. However, it is not always a good tradeoff to increase design latency. Figure 7 shows that designs with a 3-phase pipeline have equal power consumption than a 2-phase ones have. In short, designer can select a hardware/software partition among a set of solutions with different area, power and latency tradeoffs. For instance, designer could select a minimum power consumption partition if system under design is to be integrated in a portable equipment.

Finally, we have evaluated results quality. Our results are close to the optimal ones (Figure 8). In particular, there is only a difference from 5% to 10%.

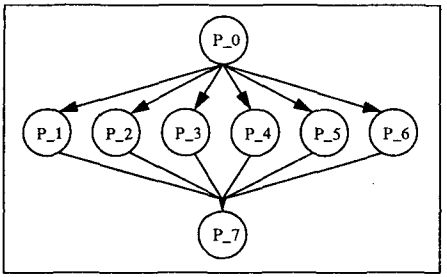

Figure 5. Process graph.

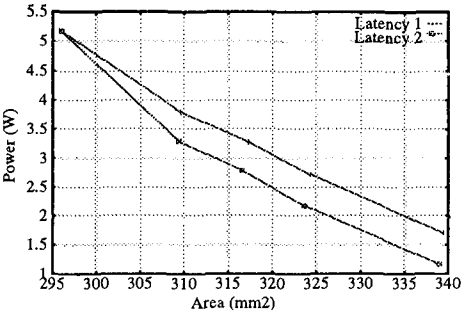

Figure 6. Power and area tradeoffs.

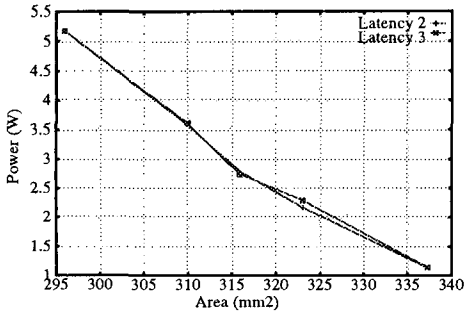

Figure 7. Power and area tradeoffs.

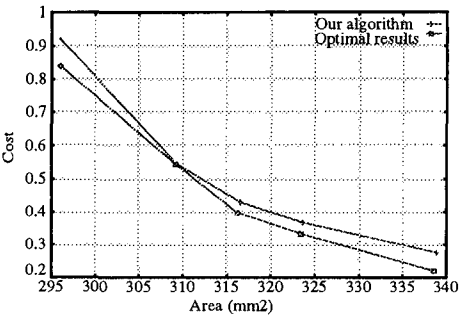

Figure 8. Results quality.

Conclusions

Our research group is developing a hardware/software codesign environment named GACSYS. In this paper, we have presented GACSYS's hardware/software partitioning tool. Main contribution of our tool is the following: a) it supports process-level pipelining. b) It estimates design power consumption. This features allow a designer to make new design cost (area or power) and latency tradeoffs. We have developed a partitioning algorithm based on simulated annealing. Our results are close to the optimal ones. In particular, there is only a difference from 5% to 10%

Acknowledgements

This work has been partially supported by Spanish Government by means of CICYT research project TIC98-1115-C02-02.

References

[1] F. Balarin, et al. *Hardware-Software Codesign of Embedded Systems: The POLIS Approach*. Kluwer Academic Publishers, 1997.

[2] J. Becker, R. W. Hartenstein, R. Kress. *Two-level Partitioning of Image Processing Algorithms for the Parallel Map-oriented Machine*. Proc. of the Intl. Workshop on Hw/Sw Codesign, 1996, pp. 77-84.

[3] C. Carreras, J. C. López, M. L. López, C. Delgado-Kloos, N. Martínez, L. Sánchez. *A Co-Design Methodology Based on Formal Specification and High-level Estimation*. Proc. of the Intl. Workshop on Hw/Sw Codesign, 1996, pp. 28-35.

[4] Juan P. Castellano, David Sánchez, Onassis Cazorla, Juan Bordón, Álvaro Suárez. *GACSYS: a VHDL-based Hw/Sw Codesign Tool*. Proc. 2nd Intl. Workshop on Design and Diagnostics of Electronic Circuits and Systems, September 1998, pp. 293-299.

[5] J. Castellano, D. Sánchez, O. Cazorla, A. Suárez. *Pipelining-based Tradeoffs for Hardware/Software Codesign of Multimedia Systems*. Proc. 8th Euromicro Workshop on Parallel and Distributed Processing, January 2000, pp. 383-390.

[6] J. G. D'Ambrosio, X. Hu. *Configuration-Level Hardware/Software Partitioning for Real-Time Embedded Systems*. Proc. of the Intl. Work. on Hw/Sw Codesign, 1994, pp. 34-41.

[7] K. A. Dowsland. *Modern Heuristic Techniques for Combinatorial Problems*. Ed. by C. R. Reeves, McGraw-Hill, 1995.

[8] D. S. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997.

[9] T.B. Ismail, A.A. Jerraya. *Synthesis Steps and Design Models for Codesign*. IEEE Computer Magazine, February 1995, pp. 44-52.

[10] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. *Optimization by Simulated Annealing*. Science, Vol. 220, Nº 4598, May 1983, pp. 671-680.

[11] P. V. Knudsen, J. Madsen. *PACE: A Dynamic Programming Algorithm for Hardware/Software Partitioning*. Proc. of the Intl. Work. on Hw/Sw Codesign, 1996, pp. 85-92.

[12] E. Macii, M. Pedram, F. Somenzi. *High-Level Power Modeling, Estimation and Optimization*. Proc. of the 34th Design Automation Conference, 1997, pp. 504-511.

[13] R. Niemann, P. Marwedel. *Hardware/Software Partitioning using Integer Programming*. Proc. of the European Design and Test Conference, 1996, pp. 473-479.

[14] M. Theibinger, P. Stravers, H. Veit. *Castle: An Interactive Environment for HW/SW Co-Design*. Proc. of the Intl. Workshop on Hw/Sw Codesign, 1994, pp. 203-209.

[15] F. Vahid, D. Gajski. *Clustering for improved system-level functional partitioning*. Proc. of the 8th Intl. Symposium on System Synthesis, France, 1995, pp. 28-33.

[16] F. Vahid, T. dm Le. *Towards a model for hardware and software functional partitioning*. Proc. of the Fourth Intl. Work. on Hw/Sw Codesign, 1996, pp. 116-123.

[17] W. Wolf, J. Hou. *Process Partitioning for Distributed Embedded Systems*. Proc. of the Intl. Workshop on Hw/Sw Codesign, March 1996, pp. 70-76.

1. GAC (Architecture and Concurrency Group). Departamento de Ingeniería Telemática. Universidad de Las Palmas de G.C. Campus Universitario de Tafira s/n - 35017- Las Palmas de G.C. e-mail: {francis, dcsr, alvaro}@cic.teleco.ulpgc.es