

Exploration of Hardware/Software Design Space in the Co-design Process

Analysis methods based on estimation to approximate a priori the system performance/cost can play an important role in hardware/software co-design of real time systems. This paper presents a fast analysis approach for exploration of hardware/software design space. This approach is based mainly on synthesis and simulation results and guided by estimation of costs and performances. Estimation models are introduced in order to evaluate cost/performance for a given hardware/software solution. A study in applying the environment to the development of a real time robot arm controller is then described. The results show fast convergence of estimated to real solution.

INTRODUCTION

In order to lead to more efficient implementations and to improve overall system performance, reliability, and cost effectiveness, cooperative approach to the design of hardware/software systems is required. Hardware and software options can be considered together. The renewed interest in such an implementation is driven by advances in technologies that support both hardware and software [1].

Starting from the initial specification, the goal of hardware/software co-design is to produce an efficient implementation that satisfies the required performance and minimizes the cost. However, there exist a large variety of technological solutions based on available hardware/software components. The designer needs to explore all possible solutions either by using automated tools or selecting his choices manually. Actually much of the effort is spent doing the design. However, as systems become more complex, an increasing amount of time is spent exploring the hardware/software design space and verifying that the design is cor-

rect and meets the performance goals [2].

The traditional approach adopted by designers of mixed distributed systems consists of separating the designs of hardware, software and designing them independently of each other (figure 1 (a)). The hardware may be modelled in high description language, simulated and then synthesized into ASICs. The software parts are coded in an assembly language specific to a selected microprocessor; this code is tested on emulators, then compiled for execution on a fixed architecture. It is only at the lowest level that the different heterogeneous parts of the system are tested together. Only at such a low level that the designers can become aware of the design flaws and optimisations needed. Once the anomalies are localized, the designers go up to the initial hardware and software specifications to iterate towards the required improvements. This loop is often repeated several times, before a first functional prototype can be realized conforming to the initial specifications and constraints.

In hardware/software co-design, the goal is thus to cre-

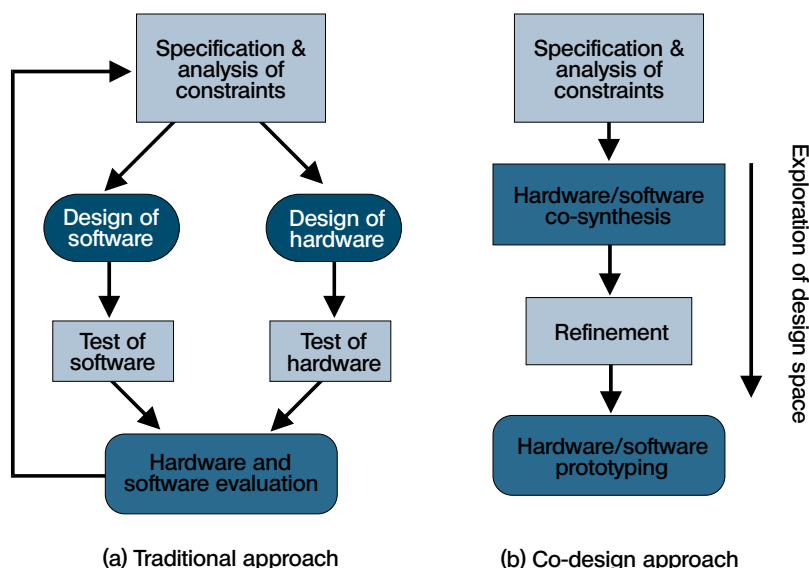


Figure 1. Design strategy of a hardware/software electronic system co-design.

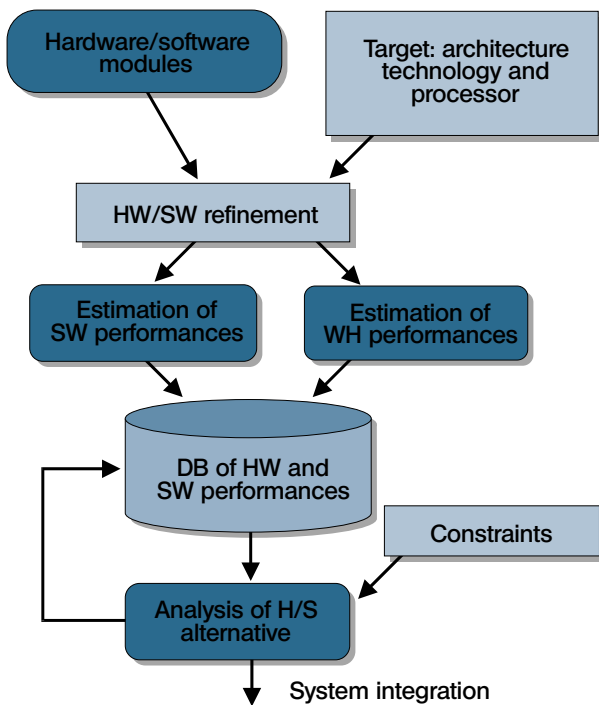


Figure 2. Analysis flow.

ate an unified environment encompassing the analysis in which the hardware and software design space can be explored before design tape out (figure 1 (b)). The design space exploration approach is a technique that is based mainly on estimation in order to evaluate cost/performance of different solutions. This enables performances models, performances estimation and hardware/software partitioning algorithm in order to deduce an optimal solution in the first steps of the design process.

Several related works have been reported dealing with different aspects for exploration of hardware/software design space. They differ in hardware/software partitioning algorithm [3, 4, 5], analysis method [6], and the degree of the exploration process automation [9]. Few frameworks address all aspects of hardware/software design space exploration.

In this paper, we address the entire problem of hardware/software design space exploration and we consider the different aspects for the analysis of hard-

ware/software solutions. However, our approach differs in the input description, performance/cost model, hardware/software estimation model and analysis method. The evaluation is mainly based on synthesis results and is guided by estimation of costs and performances for both hardware and software. Estimation models are proposed in order to estimate cost/performance for a given hardware/software solution. An objective function is achieved to define a solution quality. The process of preliminary hardware-software design and estimation is iterated in order to obtain a solution, which meets performance requirements with the minimum costs. Within this paper, a study in applying the analysis approach based on estimation to the development of a real time robot arm controller is performed.

The next section presents the analysis approach based on performances estimation. Section 3 gives an example of experimentation of the proposed method for exploration of design space. The last section draws some conclusions.

ANALYSIS APPROACH BASED ON PERFORMANCE ESTIMATION

Our goal in design of hardware/software real-time systems is to guarantee that the resulting solution satisfies the timing requirements with the minimum costs. Figure 2 shows the analysis flow based on performances estimation.

We combine some design tools with novel features for design space exploration via an analysis method. Let us assume that the target architecture is known. The analysis flow is composed of the following four steps:

- i. synthesis and compilation of different system modules;
- ii. estimation of hardware and software performances corresponding to the target architecture;
- iii. analysis of the hardware/software alternative.
- iv. If the result does not meet the desired performances repeat iii. We assume that it exists at least a hardware implementation that meets timing requirements.

The architecture model specified earlier in the hardware/software design process guides the detailed implementation choices. A modular, flexible and distributed architectural model is used. This architecture acts as a platform on which a mixed hardware/soft-

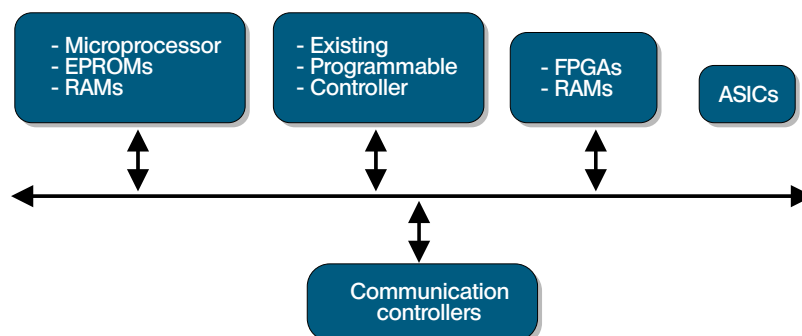


Figure 3. Typical Architecture.

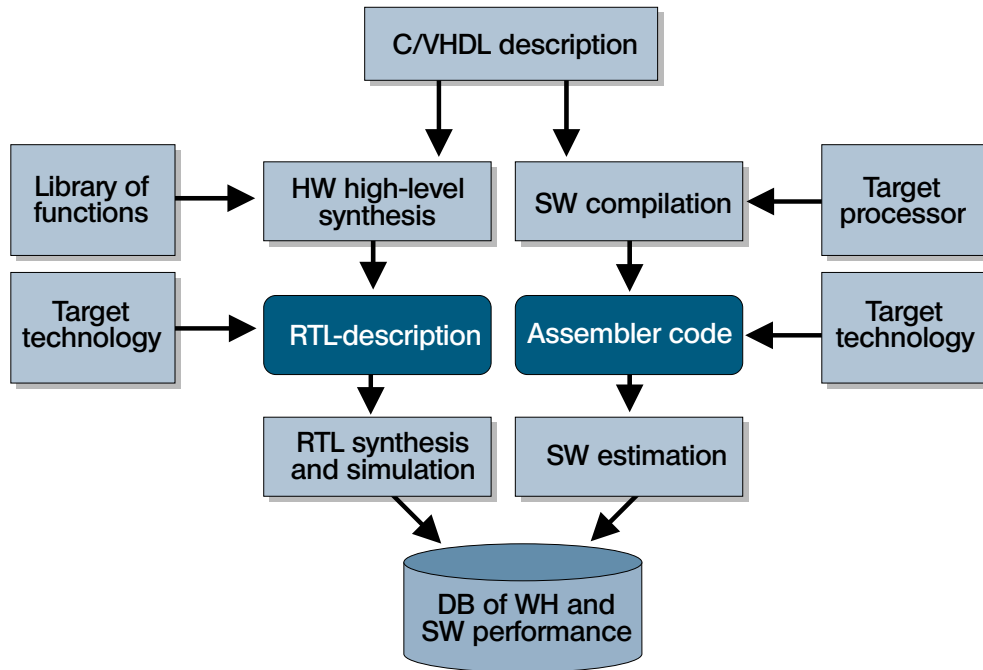


Figure 4. Deduction of hardware and software performances.

ware system is mapped. It is composed of three kinds of components: software, hardware, and communication components (figure 3). An example of platform using an Intel 80x86 microprocessor-based architecture and configurable 4013 Xilinx FPGA devices based board is performed. This platform allows to verify the deduced solution in real environment.

1 Performance/cost models

The exploration technique of design space has to consider several issues namely cost, performances, area, memory size, etc. that are defined as a design metrics to qualify a hardware/software implementation as optimal. In this work, performance (execution time constraint) and cost are considered.

The timing characteristics of the system modules implementation are specified by:

- Software Local Execution Time (SLET): the time interval at which the module function is executed in software (this time excludes all the scheduling and swapping overheads).
- Hardware Local Execution Time (HLET): the time interval during which the module function is executed in hardware.
- Global Execution Time (GET): the time interval at which all system functions are executed. It is expressed by: (1)

$$GET = \sum_{i=1}^m SLET(i) + \sum_{j=m+1}^n HLET(j)$$

Computation of execution time of the whole system depends on overlap in time. In the case where some modules are executed in parallel, the "sup" is considered; however if other modules are executed sequentially, the 'sum' is computed. For the sake of simplicity,

the extreme case is considered: GET is defined as the sum of execution time of a sub-set of m software modules and execution time of a sub-set of n-m hardware modules.

In order to evaluate different hardware/software implementations, the cost of each component is needed. The software implementation cost (Sicost) is defined by program memory size and data memory size for a given processor. The hardware implementation cost (Hicost) is evaluated by the number of cells needed to implement hardware using a library of gates. The implementation Cost (Icost) combines Sicost and Hicost. Since these values do not have the same unit, the constants ks and kh are introduced and fixed by the designer, to give a coherent and significant implementation cost. Thus, the implementation cost is as follows: (2)

$$Icost = k_s * Sicost + k_h * Hicost$$

Analysis of hardware/software implementation uses both the execution time (performance) and the cost. A cost-function (cost_function) is a single cost value that combines execution time (GET) and implementation cost (Icost) estimated metric values to define a hardware/software implementation quality. Different units, such as transistors and seconds are combined into a single value. The cost-function may be a weighed sum function in which two values, GET and Icost are weighed by the constants kt and kc respectively. The cost-function is as follows:

$$Cost_function = k_t * GET + k_c * Icost$$

In order to give a coherent and significant cost-function, the constants kt and kc are also introduced and fixed by the designer.

Another issue in combining various metrics into single

value is normalisation of metrics' units. The constants k_s and k_h may include the metric constraints so that they will be considered by cost-function. For example, the constants fixed by the designer may be divided by the metric constraints.

2 The Analysis Method

In order to analyse the different alternatives of the system implementation, we make use of the environment depicted in figure 4. The process of hardware/software design space analysis starts with a modular system description, in which hardware and software modules are described in VHDL and C respectively. All needed data for analysis is generated automatically by hardware and software synthesis and simulation tools.

For software modules, a standard compiler transforms the C description of each system module into assembly code. This code is simulated to determine the number of cycles and the software memory size using a dynamic simulation [7]. In fact, since the estimation accuracy is an important aspect for the design of real time systems, the dynamic simulation is adopted. In order to make the estimation credible, a judicious modelling of the target microprocessor is used. A functionality accurate simulation compiles and executes the design model directly on a host machine without paying special attention to simulation time.

Since using the performance of the target architecture (the cycle time of the microprocessor is fixed to be 10 MHz), the maximum execution time is determined by calculating the software critical path.

For hardware modules, the execution time and the area are determined using the synthesis and simulation results of the VHDL description. "Exampler" and

```

- initialisation
F={f1,
S=F, H=0
cost_function=lc

- first step
for i=1 to t
  S=S/{fi}
end

- second step
loop
  for each fi of
    if SLET(fi) < LTC
      S=S/{fi}
  end loop

- third step
repeat
  for each fi of
    Mov
  end
until no moves improve
procedure
  compute
  compute
  if GTE(fi) < GTC and
    S=S/{fi}
  for each fj
  end

```

Figure 5. Analysis Algorithm.

"V_System" tools are used for synthesis and simulation respectively. The maximum execution time of each system module is obtained by the product of the frequency and the number of cycles in the critical path.

The characteristics of timing requirements are specified by:

- Local critical-Time Constraint (LTC): the time interval at which each module (critical-time function) should be executed.
- Global critical-Time Constraint (GTC): the time interval at which all system functions should be executed.

The number of possible solutions for a given system can be extremely large. Some considerations are done in order to guide the exploration of design space and to converge rapidly to a solution that satisfies the constraints. These considerations are as follows:

- Many solutions will be not analysed if they are not possible a priori.
- Functions, which are considered unconstrainable, are initially placed in hardware.
- A solution is said to be feasible if the timing requirements of all critical-time functions can be satisfied.

Figure 5 shows the analysis algorithm. Given a set of functions of system modules $F=f_1, f_2, \dots, f_n$ and a set of constraints of the system under design, exploration of design space consists of three main steps:

- Exploration of design space starts with all functions in software. t functions, which are considered unconstrainable, are initially placed in hardware.
- The second step consists of analysing the software design space in order to study the feasibility of software implementation of each critical-time function. We deduce the sub-set of functions which the implementation in software is feasible: LTCs are satisfied ($SLET < LTC$). The others functions that are not feasible are considered in hardware.
- The third step consists of finding a performance satisfying solution with a minimum cost. An extension of the Greedy algorithm is performed. We start with $F=SUH$ and the cost-function of hardware solution (actual cost-function). For each f_i of S , GET and cost-function are calculated for F when f_i is moved from S to H . If calculated GET and cost-function are less than GTC and actual cost-function respectively, f_i will be moved from S to H and the cost-function of the implementation become the calculated cost-function. This procedure is iterate with each f_j successors of f_i . The process can be stopped as soon as the first solution that meets timing constraints. However, it can be continued to find a solution that meets timing constraints with minimum costs.

The initial partition is constructed for partitioning feasibility as described in the figure 5 (steps 1 and 2). The algorithm uses a greedy approach to select a function for move into H . There is no backtracking since a function moved into H stays in that set throughout rest of the algorithm. For each function move, the change in partition cost-function and global execution time are computed in constant time. The procedure is iterate for

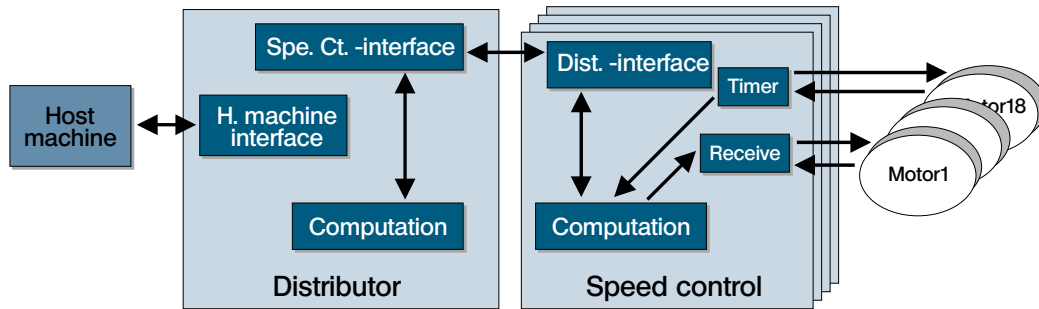


Figure 6: Control of robot arm motors

all functions in S. Thus, the complexity of the algorithm is $O(n)$. The proposed approach leads to reduced run-times of partitioning heuristics and in many cases leads to lower cost partitions. However, the limitation of the used algorithm is that it can produce a local minimum.

EXPERIMENTS

In this section, we treat an application of analysis environment for the design of a robot arm controller. The experiments should show the feasibility of the analysis environment for usage in hardware/software design space exploration.

1 The Robot Arm Controller

The robot arm controller under discussion acts between a "Host Machine" and a robot arm that makes use of 18 stepper motors. The host fixes the trajectory of the arm. It then sends the commands to the robot arm. The controller makes use of an adaptive speed control to smoothen the motion of the arm. The distribution system feeds the adaptive controller in parallel. The motor controllers continuously send back information's about the distance they still have to go. So the system performs two principal tasks: adaptive distribution of pulse packets and adaptive speed control in real-time.

As shown in figure 6, the system is decomposed into computation modules and interface modules. The Distribution sub-system receives data from the Host Machine and provides the travelling distance to the Speed Control sub-system. This control sub-system computes the number of pulses for speed control and

translates them into motor control signals. The motor returns the pulses already executed. Thus the Speed Control sub-system controls the motors step by step in real-time. The response time of each motor is 6 ms (the time needed to update the speed). Thus, GTC is 6 ms. In order to illustrate the advantage of hardware/software co-design, a case study of this application was performed [8]. The rest of this paper will be consecrated to explore the hardware/software design space regarding the performance/cost and using the analysis approach described above.

2 Performances/cost of hardware and software modules

For software, the execution times and the memory sizes of different modules are obtained using dynamic simulation. For hardware (Xilinx 4013 FPGA), the execution time and the gate sizes are also deduced from description VHDL using results of simulation and synthesis. The total execution times for each module are obtained with regard to speed. Table 1 presents the execution times (HLET and SLET) and the costs (Hlcost and Slcost) of each module for hardware and software respectively. Hardware cost represents the number of gates while software cost represents the number of bytes.

3 Analysis of the hardware/software design space

The performances of hardware/software implementations depend on the target architecture technology. The frequency is fixed upon the used target architecture (10 MHz). The execution times of a sub-set of soft-

	SLET(cycles)	HLET(cycles)	Slcost(bytes)	Hlcost(gates)
Distributor	1345.50	8.74	1915	9915
H.Machine Interface	1,27	0,21	195	840
Computation	1332.50	5.33	1550	8610
Spe.Ctr. Interface	11.73	3.20	170	465
Speed Controller	24570.66	259.20	2145	3150
Dis. Interface	10.66	3.20	170	450
Computation	23680.00	96	1640	2090
Timer	400	64	160	260
Receive	480	96	175	350

Table 1. Execution Times and Implementation Sizes.

Solution	GET	Real-Time violation	Icost	Cost Function	possible Solution
1	0.035 ms	No	66615	1.005	Yes
2	44.3 ms	Yes			No
3	44.2 ms	Yes			No
4	5.048 ms	No	21220.	1.159	Yes
5	0.160 ms	No	58615	0.905	Yes
6	4.91 ms	No	29220	1.256	Yes

Table 2. Result of Analysis.

ware modules and a sub-set of hardware modules are then calculated. For each hardware/software implementation, the global execution time (GET) of whole the system is calculated by applying Equation 1. The implementation cost of the whole system and then the cost-function are determined by applying Equations 2 and 3, and combining the information from table 1 about the hardware and software implementation cost. However, some assumptions are considered concerning the constants k_s , k_h , k_t and k_c :

- $k_s=k_h=1$. We consider that the costs of a byte and a gate are equal.
- $k_t=1/6$. k_t is fixed regarding the GTC (6 ms).
- k_c is calculated regarding the hardware cost of the whole system. Thus, the analysis of a hardware implementation is done first.

Pure hardware implementation: If all functions are realised in hardware with FPGAs modules, the independent 18 Speed Control Sub-Systems may be designed as concurrent state machines. On considering the FPGA Xilinx 4013 capacity, the number of speed control sub-systems integrating on one chip is 3. Therefore this implementation has to use one FPGA for the distributor and 6 FPGA chips for the 18 Speed Control Sub-Systems (solution 1). The GET is 0.035 ms while the cost is 66615. Thus $k_c=1/66615$. Note that the cost-function of this solution is 1.005.

Pure software implementation using one software processor: We assume that all the modules will be executed on the same processor (solution 2). This needs an extra scheduling step that will arrange the execution order of the modules. The total execution time is about 44.3 ms (thus larger than 6 ms). This solution does not meet the GTC.

Pure Software implementation using two software processors: The system can be designed as two asynchronous parts: the distributor may be done concurrently to the 18 motor Speed Control Sub-Systems. Thus each part can be run on a separate microprocessor (solution 3). The GET of the 18 Speed Control Sub-Systems is almost 44.2 ms (>6 ms). This software solution does not meet the GTC as well.

Pure-Software implementation using 10 software processors: The Distributor and the 18 Speed Control Sub-Systems are independent and may therefore be designed as concurrent state machines. On considering the execution time of each module, the number of speed control sub-systems executing on one microprocessor is 2. Therefore this implementation has to

use one microprocessor for the distributor and 9 microprocessor for the 18 Speed Control Sub-Systems (solution 4). The GET of whole system is almost 5.048 ms (<6 ms) and the implementation cost of whole system is 21220. Thus, The cost-function is 1.159 (larger than 1.013: cost-function of pure hardware implementation).

Hardware/software solutions: Because external real-time signals are better analysed in hardware, we associate hardware components to the 18 Speed Control Sub-Systems and a microprocessor to the distributor. Thus, this implementation has to use one microprocessor for the distributor and 6 FPGA chips for the 18 Speed Control Sub-Systems (solution 5). The execution time is 0.160 ms while the cost is 58615. Thus, the cost-function is 0.905.

Note that another alternative in hardware/software solutions can also be analysed. We associate hardware component to the distributor and 9 microprocessors to the 18 Speed Control Sub-Systems (solution 6). The execution time is almost 4.91 ms; however the cost is 29220. Thus, the cost-function is 1.256 (larger than 0.905).

Four solutions meet the real time requirements as illustrated in Table 2. However, the solution 5 represents the optimal solution. It has a better cost-function compared to others solutions.

A rapid prototype of solution 5 is performed using an Intel 8086-based architecture (software part) and Xilinx 4013 FPGAs (hardware part). This solution is confirmed by prototyping platform in order to verify the system in its real environment [10]. An analysis indicates that this solution correctly implements the system functionality while meeting real-time requirements.

CONCLUSION

This paper presented an analysis approach defined as the process and techniques employed to produce the optimal implementation in a short time. The analysis give rapidly predictions on the outcome of applying existing high-level synthesis tools and added estimation tools. The efficiency of this technique is done by the use of performances models and a cost-function guided by automated tools for evaluation.

Despite the fact that the example contains only a single timing constraint ($GTC=6ms$), combined real time constraint and the implementation cost make the analysis useful to deduce the optimal solution of the hardware/software implementation. The experimental results show fast convergence of estimated to real

solution.

Our method includes a performance/cost model, a hardware/software estimation model and an estimation method that does not depend on a particular target architecture. In this paper, target architecture using 80x86 microprocessor for software modules and FPGA for hardware modules is considered. However, the proposed approach may be considered in order to allow other target architectures. In later case, the used existing synthesis tools and added estimation tools have to include technologies of these target architectures. For example, if a hardware module will be implemented in an ASIC, the used technology must be included in the library of synthesis tools.

For the sake of simplicity and since this approach can be used for evaluation of different hardware/software solutions earlier in the design process, some assumptions are considered. However, these assumptions do not have an influence on the results of design space exploration to produce a solution meets the real time requirements with the minimum cost.

Future work aims management tools that are required to orchestrate the design process, i.e. for systematic control of the design data, tools, and flow. On the other hand, automatic schedule will be achieved to prioritise modules that request execution simultaneously.

Moreover, this approach is flexible and provides an open environment for consideration of other hardware/software technologies. FPGAs circuits were considered for integration. Further implementation consists of replacing the FPGA technology by semi-custom technology.

Acknowledgements: This work based upon work supported by the CMCU Franco-Tunisian collaboration. Author thanks Professor R. Tourki (FSM-Monastir, Tunisia) and Dr. A. Jerraya (TIMA-INPG, France) for their assistance ■

Mohamed Abid is an assistant professor of electrical engineering at Sfax University in Tunisia. He holds a Diploma in electrical engineering in 1989 from University of Sfax in Tunisia and received the Ph.D. degree in computer science in 1989 at University of Toulouse in France. His research interests includes hardware/software co-design, design space exploration and prototyping strategies for real-time systems.

REFERENCES

1. AD & T Roundtable on Hardware/Software Co-design, IEEE Design and Test, (March 1997), 75-83.
2. Collet R., Panel: Complex System Verification: The Challenge Ahead, 31 st IEEE/ACM Design Automation Conference,, San Diego CA, (June 1994), 320.
3. Gupta, R. and DeMicheli G., A Co-synthesis Approach to Embedded System Design Automation, International Journal of Design Automation for Embedded System, Vol. 1, Nos. 1-2, (January 1996), 69-120.
4. Ernst R., Henkel J., Benner Th., Ye W., Holtmann U., Hermann D. and Trawny M., The COSYMA environment for hardware/software co-synthesis of small embedded systems, Microprocessors and micro-systems, (May 1996) 20, 159-166.
5. Kavalade A. and Lee E. A., A Global Criticality/Local Phase Driven Algorithm for the Constrained Hardware/Software Partitioning Problem" Third Int. Workshop on Hardware/Software Co-design, France (1994), 42-48.
6. Yen T. Y. and Wolf W., Sensitivity Driven Co-Synthesis of Distributed Embedded Systems, International Symposium on System Synthesis, Cannes, France (September 1995)
7. Barros E., Rosentiel W. and Xing X., A Method for Partitioning UNITY Language in Hardware and Software, Proc. Euro-DAC'94, France (September 1994).
8. Vahid F., Gong J. and Gajski D., A Binary Constraint Search Algorithm for Minimising Hardware during Hardware/Software partitioning, EuroDAC'94, France (September 1994).
9. Madsen J., Grode J. and Knudsen P. V., Hardware/software Partitioning using the Lycos System, Hardware/software Co-Design: Principles and Prattice, Edited by J. Staunstrup and W. Wolf, Kluwer Academic Publishers, Boston/Dordrecht/London, (1997).
10. Abid M., Prototyping Environment for Design of Hardware/Software Systems, Accepted for publication in the International Journal of Computer Systems Science & Engineering (1998).