A Hardware-Software Co-Design System for Embedded Real-Time Applications

Sérgio Vanderlei Cavalcante

A thesis submitted for the degree of Doctor of Philosophy

in the

Department of Electrical and Electronic Engineering

at the

University of Newcastle upon Tyne

June, 1997

ABSTRACT

This dissertation presents an environment for the co-design of hardware and software in critical real-time embedded systems. The environment is targeted to producing solutions suitable for multi-processor architectures. The challenges and difficulties faced in developing such an environment are shown.

The system includes three major parts:

- A specification tool which uses hierarchical high-level timed Petri nets, while enforcing a design methodology to ensure consistent specifications;
- A scheduling algorithm able to schedule a set of processes with deadlines, release times, precedence, exclusion relations, and which allows process pre-emption, in a heterogeneous multi-processing architecture;
- An analysis and partitioning algorithm that has real-time constraints as its primary concern.

At the end of the dissertation some examples are presented to illustrate the use of the system.

Table of Contents

Page

List of Illustrations

List of Tables

Acknowledgements

This work was made possible by the funding of CAPES - Brazilian Research Agency - under the project number 2779/92-8.

Many thanks to my wife for her support, love, care, understanding, and encouragement during our life together and specially throughout the development of this dissertation.

I am deeply indebted to my parents not only for their continual understanding and encouragement but also for the material support which provided me with conditions to reach this point in my carrier.

Thanks to my supervisor, Prof. David J. Kinniment, for his help and encouragement throughout the development of the project.

Thanks to all staff from the Department of Electrical and Electronic Engineering of the University of Newcastle upon Tyne for their help. In particular, many thanks to Dr. Gordon Russell for his invaluable guidance and friendship.

Thanks to Prof. Mauro Pezzé from the Politecnico de Milano, Italy, for providing the Cabernet environment and making its source code available.

My gratitude to Rogério de Lemos, Avelino Zorzo, and Marinho Barcellos for the discussions about the methodology. Thanks to Shirley Craig, the best librarian I have ever met, for the care and interest in forwarding all new information related to my topic which reached her hands.

Thanks to the many friends I made here, specially those from the Luso-Brazilian-Society, for great moments and the relaxation necessary to cope with the stress.

A Patricia,

aos meus pais,

a Júlia.

1.

Introduction

The high complexity found in current digital systems requires new tools and methodologies for design. By working at higher abstraction levels the designer can be kept from design details which can be handled by automated tools in CAD environments. The automation of low level tasks also helps to produce correct designs, which further reduces the overall design time and cost. Market competition produces a strong request for productivity. The damage in profitability due to a six month delay in a product release can be worse than a 50 per cent cost overrun during the design cycle [103]. At the same time, market pressures make it necessary to produce more adaptable and changeable hardware devices (hardware to software migration) and more performance efficient software systems (software to hardware migration).

In order to handle this complexity and productivity need, an approach called *hardware-software co-design* has become a hot research topic since the beginning of the decade. This approach is aimed to the co-development of the hardware and software parts of digital systems, promoting a better integration between those parts, and reducing cost and time-to-market by using defined methodologies and CAD tools.

Although co-design is not new, its development has been made possible by the availability of techniques, such as hardware high-level synthesis, which puts hardware

development at an abstraction level nearer to that found in software development [117]. Furthermore, the advent of application-specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs) has helped to change the boundary between software and hardware. This boundary has become fuzzier and techniques once used for software development have started to be used for hardware as well.

1.1 Embedded Systems and Co-Design

Co-design is being used in several applications, from the design of general-purpose processors and compilers to embedded digital systems. In the particular case of embedded systems, the design starts with a high-level functional description where implementation details are not taken into account. As the design proceeds, the system is split into hardware and software partitions. The choice on the best partitioning depends on characteristics required to be present at the final system implementation, such as area, speed, and upgradability requirements. Analysis techniques are used to estimate values for the design features, validate refinements, profiling of systems, etc. Due to the large number of different solutions that may be produced, the co-design process needs to allow for constant interaction with the user in order to guide the solution space exploration.

There has been an enormous increase in the number of systems which use dedicated processors. Examples of such systems are camcorders, microwave ovens, mobile phones, fax machines, printers, cash machines, and industrial control systems. It is a rapidly expanding market, as shown in Figure 1.1. Co-design is likely to play a major role in the development of such devices [35]. However, there are still many problems of interest to be tackled.

Most specification methods used in co-design start with an initial language which is translated to an internal representation. This method makes it difficult for the designer to follow the design process. A useful characteristic would be the use of a single representation from the high-level specification through to hardware synthesis and software compilation.

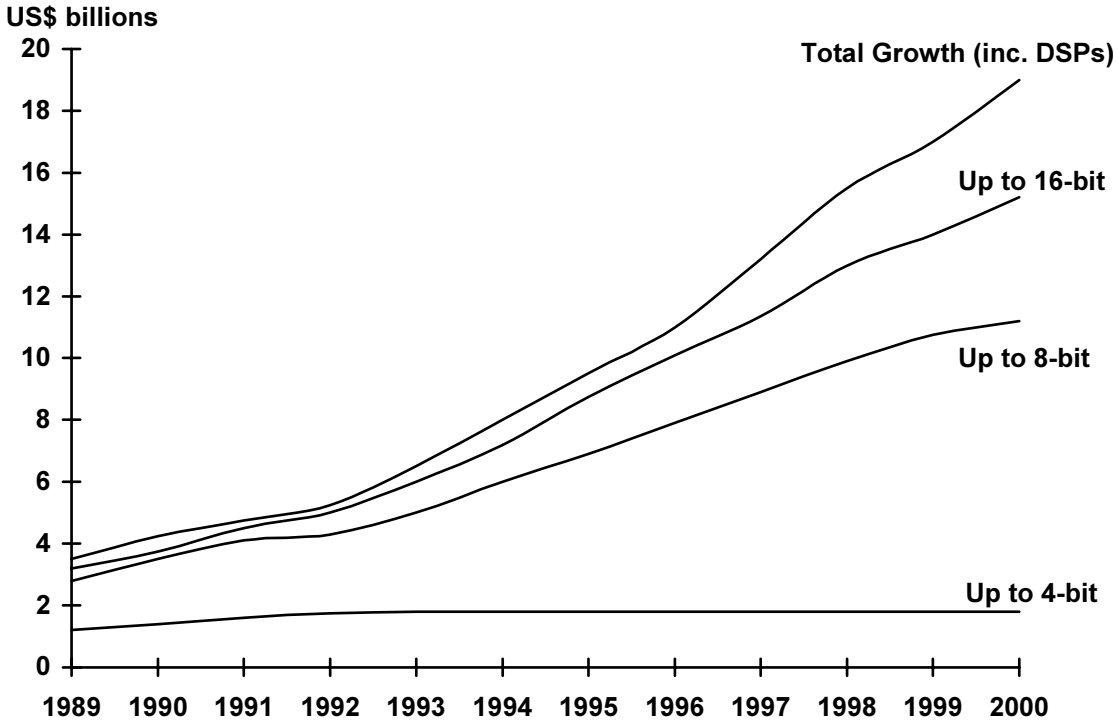

Figure 1.1. World-wide revenues for micro-controllers and DSP sales [35]

Architectures comprising multiple software processors are not common in co-design. Figure 1.2 shows a basic co-design architecture which is used by several authors (e.g. in [8, 52]). The use of several processors particularly benefits rapid prototyping by allowing the system to be emulated using minimal specially built hardware.

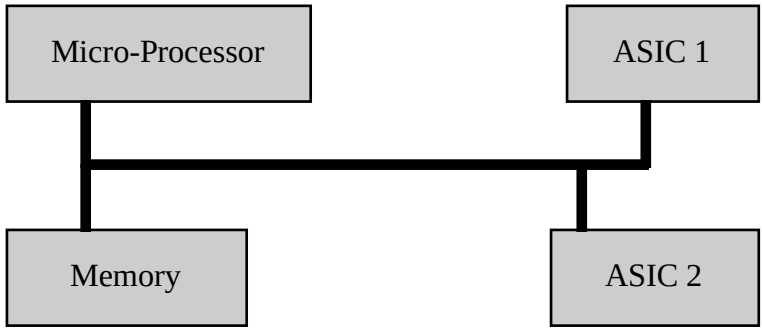

Figure 1.2. Basic co-design architecture

The general approach used in this area includes the exploration of hardware-software characteristics such as area, speed, memory limitations, power consumption, maintainability, upgradability, testability, reliability, etc. However, most of the systems developed so far do not consider critical real-time issues explicitly, or when they do so there are many restrictions such as not allowing more then a task to operate at the same

time on the architecture. This is done to overcome the difficulty of guaranteeing time properties in multi-processor architectures. However, in this approach the only advantage in implemented tasks in hardware is the execution speed-up of each task but no advantage in parallelism exploitation exists.

When it comes to time critical systems there must be analysis methods to guarantee that the requirements are met. Furthermore, any automatic partitioning algorithm targeted to such systems needs to take time issues into consideration in order to make decisions which would increase the probability of finding feasible architectures.

1.2 Objectives of This Dissertation

Based on the above discussion, the target systems to be dealt with in this dissertation are embedded real-time applications. As such, the primary objectives are:

- To provide more flexibility in the basic architecture by allowing several different software processors to operate concurrently. This would permit the use of the best type of processor to each application, e.g. DSPs, micro-controllers, etc.
- To tackle the problem of dealing with real-time constraints in co-design, while permitting parallelism among partitions and processors.
- The use of a single, simple, neutral specification language. This language should follow the trend of using graphical design methods. At the same time it needs to permit real-time requirement specifications. The neutrality mentioned before refers to the ability of modelling hardware and software tasks without imposing any bias. It needs to allow for refinements to be done from the initial abstraction level down to the generation of models for hardware synthesis and software compilation.

1.3 Proposed Method

In order to achieve the objectives mentioned, the following approach is used here:

- Use of a well-established graphical formalism as the basis of the specification formalism. Specifically, Petri nets [93] are being utilised here due to their widespread use in several areas, from concurrent system analysis and specification to hardware synthesis. Also, Petri nets have formal semantics which allows for the development of formal analysis techniques and transformations. Nevertheless,

extensions to the formalism were necessary in order to provide the facilities required for the co-design approach used here.

- Use of a task scheduler in order to verify time properties. A new algorithm was developed in order to allow for the pre-runtime scheduling of sets of tasks distributed over a multi-processor architecture. Tasks with release times, deadlines and inter-task dependencies such as precedence, mutual exclusion, and pre-emption relations are considered. The algorithm is also used to produce information for the partitioning phase.

- A dual-phase automatic partitioning is being applied, where starting from an unfeasible specification, tasks are moved to hardware in an attempt to produce a feasible solution. The partitioning phases are:

  1. Pre-Partitioning: This takes care of individual-task time constraints and constraints generated by inter-task dependencies. These constraints can be solved only by execution speed-up, i.e. by moving tasks to hardware (movement of tasks among software processors are not performed by the system).

  2. System Partitioning: This is concerned with problems derived from excessive software processor utilisation, i.e. contention for the use of a same resource by several tasks. At this phase, the speed-up gain when moving a task to hardware is interesting but not strictly required. The parallelism acquired when moving tasks to other hardware processors is enough to solve real-time problems here.

1.4 Organisation of the Dissertation

Chapter 2 provides an overview on the state-of-the-art in hardware-software co-design and an introduction to issues of importance to the proper understanding of this dissertation, such as Petri nets and real-time systems.

The general co-design methodology proposed here is shown in Chapter 3. In particular, the specification formalism is presented together with the graphical CAD tool which implements it. This tool is used for design specification and initial architecture selection, which may present more than one software processor. Tasks may also be initially allocated to any hardware device (e.g. ASICs) that the user specifies. New hardware

devices are created during partitioning, if necessary, to guarantee the schedulability of the system.

The methodology is also used to guide the refinements necessary to produce an output suitable to be used by the scheduling and partitioning algorithms. This guidance is performed through consistency checks performed by the CAD tool which compares the design with the rules imposed for the generation of the output model.

The scheduling algorithm is presented in Chapter 4. It is able to find a solution, whenever one exists, for the scheduling of a set of processes with deadlines, release times, precedence, exclusion relations, and which allows process pre-emption, in a heterogeneous multi-processing architecture. It is based on a branch-and-bound technique developed by Xu and Parnas [119].

Chapter 5 presents the partitioning algorithm developed to tackle real-time constraints. Two definitions are key to the proper understanding of the partitioning approach proposed:

1. A process implementation is considered feasible if, in the absence of resource constraints, its related time constraints are met. This is dealt with during pre-partitioning.

2. A whole system is considered feasible if all its processes are feasible and there exists an allocation and scheduling of those processes such that the time constraints are met. System feasibility is considered during system partitioning.

Some examples of the use of the methodology are presented in Chapter 6. A mixture of academic and real-life designs are employed to enhance specific characteristics of the co-design approach developed in this dissertation.

Finally, general conclusions and plans for future work are presented in Chapter 7.