Systems Development Tools for Embedded Systems and SOC's

Óscar R. Ribeiro

Departamento de Informática, Universidade do Minho
4710–057 Braga, Portugal
oscar.rafael@di.uminho.pt

Abstract. A new approach for developing high-level performance models for system-on-a-chip (SOC) designs is presented. In this work we study the feasibility of using that approach to design RED, a novel architecture based on a standard microprocessor containing a reconfigurable datapath, for high-performance embedded systems.

1 Introduction

In this work we are going to study the feasibility of using a new approach for developing high-level performance models for SOC designs [1] in a particular SOC, the RED architecture [2]. This work does not present a tutorial of how analyse a RED, but instead identify some topics in the process to early analysis of SOC presented, that need to be changed for the RED analysis. Section 2 describes the RED architecture. Section 3, presents a new approach to the early analysis for SOC design, and the system-level tools to SOC design. Finally, section 4 presents some conclusions.

2 RED Architecture

Traditionally, we have two distinct ways of implementing computations: in hardware, using Application Specific Integrated Circuits (ASICs), custom VLSI, or gate-arrays; or in software running on processors, using the DSPs, microcontrollers, embedded or general-purpose microprocessors. Since the introduction of Field-Programmable Gate Arrays (FPGAs), there are an alternative to the traditional hardware and software alternatives. The using of FPGAs originated an important new structure for implementing computations, called Reconfigurable Computing [3, 4, 5, 6].

The reconfigurable devices, including the FPGAs, have associated an array of computational elements (sometimes know as logic blocks) whose task is determined by a multiple programmable configuration bits. These elements are connected using a set of routing resources that are also programmable. Thus, digital circuits can be mapped to the reconfigurable hardware by computing the logics functions of the circuit within the logic blocks, and using the configurable routing to connect the blocks together to form the necessary circuit.

The main goal of Reconfigurable Computing is providing architecture between the hardware and the software that has a higher level of flexibility than hardware, and much higher performance than software. The reconfigurable hardware increases the flexibility of hardware implementation, sacrificing some power or performance [7].

There are a concept closely to the reconfigurable, the configurable, which only support a one-time customisation prior to manufacturing. By comparison the reconfigurable ones supports both postman factoring and dynamic runtime configurability.

The RED architecture [2] is a novel architecture, based on a standard processor plus some reconfigurable logic, in the form of a pipelined datapath. This architecture can execute several complex operations of more than two operands at time, thus RED can be used for those operations of an application that are critical in time, while the rest are being executed by the processor. RED is a reconfigurable coprocessor, therefore it is not an FPGA.

In [2] is guarantee the easily integration of a RED on FLECOS methodology [8], where the above task can be easily performed by a compiler base on this methodology voiding the classical problem of hardware/software partitioning. FLECOS methodology is not only a design methodology but also a set of tools to develop complex heterogeneous systems. RED architecture is currently being prototyped, and will be soon integrated into FLECOS environment.

We can see the RED architecture as a concrete example of a SOC, which have a reconfigurable component, now we are interested in find the necessary changes of the SOC analysis to apply this analysis to the RED.

3 SOC Design

Nowadays there are an higher level of integration of many functionalities in the same chip, than in the past where the same system had many chips associated with it, and these chips are interconnected with a board, this higher integration is possible because the rapidly advance of the silicon technology, this system is usually called a system-on-a-chip (SOC).

Some embedded systems can be implemented on a SOC, depending on the volume of production that are required.

The designers of a system-on-a-chip (SOC) have an important task in the construction of the system, they must simulate the behaviour of the final system before the implementation, which usually have an expansive cost associated with it.

Because the complexity of the systems to be designed, it is useful the use of some components (usually called cores), that had been previously tested, thus they must have the expected behaviour. And after the correct selection of cores the designers must interconnect them, according to the functionalities pretended.

Usually the SOC designers work with a register-transfer-level (RTL) system specification, which includes the identification of all registers or memory elements, and the precise time and conditions of transferred data among them. Thus, the RTL analysis can have the same complexity as the implementation of the chip itself [1].

To increasing the SOC analysis, it is necessary to use an higher level of abstraction to represent the system (system-level representation), at an early stage of SOC design, the analysis of the system-level representation is called an early analysis. At an early stage, some of the questions that must be answered are the following:

- What functions should be implemented in hardware or software?
- Which embedded processor should be used? Or can the chosen processor handle the software functions within the real-time constrain of the system?
- What is the worst-case interrupt latency?
- What is the internal bus utilization?
- Can the chosen architecture/component be laid out with the available chip size?

- What is the expected system power consumption? Is this within the limits for the chosen packing technology?

We need an extended design process, which evaluate options and make critical architectural decisions based on a system-level representation in advance of an RTL, to address the challenges described above. As described in [1] this process can be divided in the following tasks:

- System-level representation;
- Performance analysis;
- Floorplanning;
- Power estimation;
- Predictable path to RTL flow.

The first task of an early analysis is the specification of a system-level representation of the SOC to be designed. This includes the following steps:

- Identifying the SOC components;
- Interconnecting the components;
- And describing Clock domains.

Every components, including buses and power-management circuit, are selected from a given technology library of cores. Then are described the connections between the chip interface and the constituent cores, that work need the knowledge of the supporting architecture to the connections and the way of each pin should be connected. After that it is necessary to describe the clock domains, which includes the clock frequencies, their source, and their interfaces with various components. The most important criterion of a SOC architecture is its performance, that is the performance defines the how efficient are the decisions about others criterions.

The chip size, the number of wiring layers, and congestion depend on the chip floorplan, and all of which contribute to the cost of the SOC. The process of floorplanning must determine: the chip die size with pin assignments, the placement of the components, the core from factors, the component pin assignments, the global routes and the assignment of the terrains and voltage islands. Today is designers have another key to consider, the power consumption.

The figure 1 shows the environment of an early analysis. As we can see, at the system level representation we can apply the tool for early analysis to the SOC created by the SOC editor, eventually using cores from libraries, and from this we can answer to the questions about performance analysis, power estimation and floorplanner. Thus, the system-level representation can be seen as a front end for the RTL design. The bridge from the system-level representation and the RTL is obtained with the ”hand-of” function.

This approach has the benefit of operating on a unique central description of the system, which is then automatically mapped to different description levels for use by different design, simulation, and analysis tools. Therefore, many of the problems of these tools, namely their operation on different types of descriptions and on different levels of abstraction, are avoid with the above approach.

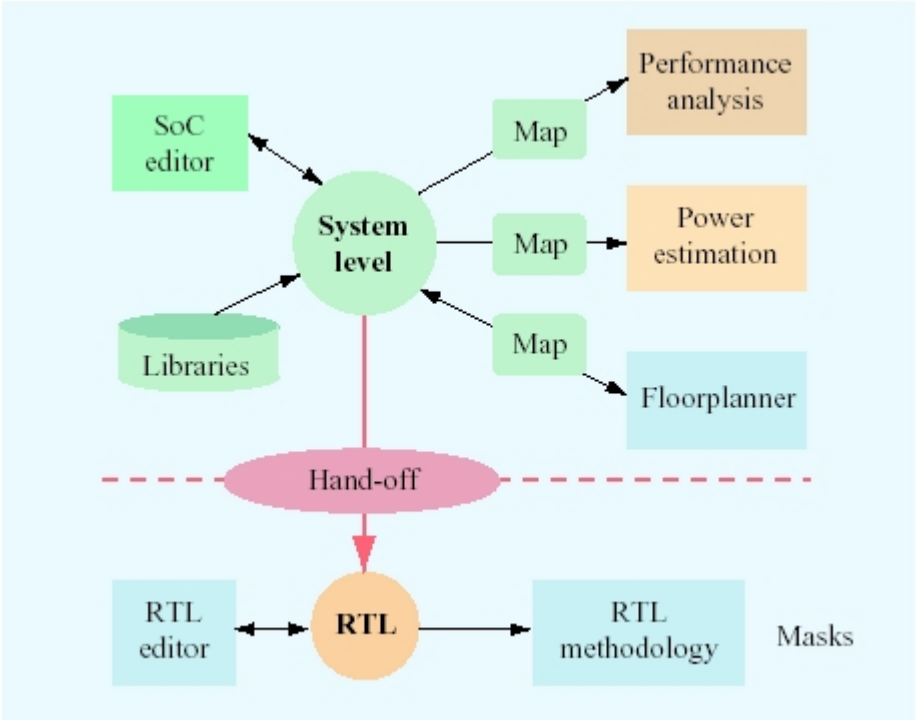

Fig. 1: Early analysis environment. This figure has been adapted with permission from [1].

3.1 System-Level Tools

The system-level tools can be classified in three different categories:

- Design;
- Verification;
- And performance analysis.

Now we are interested in the system-level tools of design. There are many new tools to work at the system-level design, which have been the focus of academic research during several years [1].

One of the must important decisions at an early stage of the analysis is the selection of functions that want to be implemented in hardware or in software.

The above tools follow a vertical functional approach, i.e. first the system is partitioned in its basically parts, which are associated to hardware, or software, and then the global system is constructed. There is another approach to the co-design of hardware/software, called the structured approach, which is the opponent approach of functional one that is first a global structure is designed and then this structure is divided in small parts.

With a third alternative, the reconfigurable architecture, associated to the hardware/software ones, we have a different partitioning problem, because it is necessary define the “hardware” of the architecture and the software to run on it [5], which need the knowledge of the circuit designs of the reconfigurable systems.

In the RED architecture we have a traditional processor and a reconfigurable hardware, a datapath, so any program that will run in there must be partitioned into sections to be executed on the reconfigurable hardware and others sections to be executed in software on the processor.

In general, complex control sequences such as variable-length loops are more efficiently implemented in software, while fixed datapath operations may be more efficiently executed in hardware, according to [5]. Thus, the increasing of the execution efficiency will pay the costs associated to the reconfiguration of the hardware associated to the datapath, because hardware specific is created, in run-time, to the datapath.

The work of design a RED, depends on the compiler to be used. There are two solutions to do the compilation hardware/software on a RED architecture:

- Hardware compilation is performed separately from the software;
- Or the same compiler manages both the hardware and software aspects of application.

In the first solution there are an effort on the designer work to identify the sections that should be mapped to hardware, and to translate these into special hardware functions. In the second one, the hardware/software partitioning can be performed either manually, or automatically by the compiler itself, which is the software programmer can use the reconfigurable hardware without any direct intervention. The circuits created using these methods, are quickly and easily created, but they are generally larger and slower than manually created versions.

4 Conclusions

Reconfigurable computing is a new structure for the execution of algorithms. With this structure the problem of hardware and software partitioning has an increase of complexity, because now we have a new alternative to execute computations, the reconfigurable hardware.

The RED architecture includes a processor and a pipelined datapath created by some reconfigurable logic.

We described a different approach of early analysis, which works at the system-level representation, that approach gives to the SOC design a higher level of abstraction than the classic one at the level of the RTL. Therefore, selecting the adequate level of abstraction is very important.

The SOC with a reconfigurable part, will increase the complexity of SOC analysis, because it introduces a new architecture, which will received some functionalities of the global system, that are attributed by the SOC designer.

The reconfigurable in this architecture work at level of operations or systems calls. Thus, the changes at the SOC designer will be theoretically insignificant.

References

[1] Darringer, J.A., Bergamaschi, R.A., Bhattaccarya, S., Brand, D., Herkersdorf, A., Morrell, J.K., Nair, I.I., Sagmeister, P., Y.Shin: Early analysis tools for systemon-a-chip design. IBM J. Res. & Dev. 46 (2002)

[2] Rinc´on, F., Moya, J.M., L´opez, J.C.: RED: A Reconfigurable Datapath. Technical report, Universidad de Castilla-La Mancha, Dept. de Inform´atica (2002)

[3] Estrin, G.: Reconfigurable Computer Origins: The UCLA Fixed-Plus-Variable (F+V) Structure Computer. Annals of the History of Computing, IEEE 24 (2002)

[4] DeHon, A., Wawrzynek, J.: Reconfigurable computing: what, why, and implications for design automation. In: Proceedings of the 36th ACM/IEEE conference on Design automation conference, ACM Press (1999) 610–615

[5] Compton, K., Hauck, S.: Reconfigurable computing: a survey of systems and software. ACM Computing Surveys (CSUR) 34 (2002) 171–210

[6] Schaumont, P., Verbauwhede, I., Keutzer, K., Sarrafzadeh, M.: A quick safari through the reconfiguration jungle. In: Proceedings of the 38th conference on Design automation, ACM Press (2001) 172–177

[7] Dutt, N., Choi, K.: Configurable Processors for Embedded Computing. Computer 36 (2003) 120–123

[8] Moya, J.M., Moya, F., Domnguez, S.: Multi-language specification of heterogeneous systems. Forum on Design Languages (FDL'2000) (2000)