Embedded Systems Design And Verification : Reuse Oriented Prototyping Methodologies

S. Raimbault, G. Sassatelli, G. Cambon, M. Robert, S. Pillement, L. Torres

LIRMM,, UMR 5506 – Université Montpellier II / CNRS
161, Rue Ada, 34392 MONTPELLIER CEDEX - FRANCE

Key words: Design Reuse, Co-design, Co-Verification, Prototyping

Abstract: The SIA roadmap plans for 50 millions transistors asics/SOC in 2008 [1]. The design of these chips cannot be achieved in the required *Time-to-Market* constraints without new methodologies. The key solution for saving design time is *Design Reuse*. However, while design reuse solves many design problems, it causes increased verification problems. The complexity of these new designs leads to simulation times that become prohibitive with regard to market pressure. The verification is thus achieved through high-speed emulation and *prototyping* technologies. The scope of this paper is to present these new methodologies. SPW [2] (from Cadence) can handle a wide variety of models in a co-simulation for virtual prototyping. Designs are verified by real prototyping on an Aptix *reconfigurable platform,* using DSP and FPGA components.

1. INTRODUCTION

In a competitive marketplace where many similar products compete for consumer attention, manufacturers must offer noticeably better features and performances. As the integration scale gets larger and larger, these extra features are provided by more and more logic in the design. At the same time, the developement of each unit of the design with the traditionnal methodologies will not meet the time to market constraints. As the required functionalities are often the same from one design to another, design time often is saved by reusing functionals blocks from an older system that has already been validated, or from IP vendors.

The design flow is moving to a higher level of abstraction; concepts such as co-simulation, co-synthesis and co-design are used and applied much more often and, in fact, have become an industrial reality. Around this vo-

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35498-9_57

L. M. Silveira et al. (eds.), *VLSI: Systems on a Chip*

cabulary, re-usable cores seem to be the right way to go toward the system on chip and to implement the interface between the system specification and the final physical level. One kind of block which is well suited for design reuse is the processor. Most complex embedded systems are built around one or more processors, with dedicated hardware for intensive tasks, analog parts and interfaces. Therefore, a system is built with hardware and software parts, that have to be designed together (Co-design).

The verification of such systems can be achieved in two ways: formal methods or simulation. Formal verification aims at getting the mathematical proof of the system's predictability. Although this approach can give a high level of confidence in the design, it is very difficult to obtain this result for complex designs, especially if an asynchronous communication scheme is used between the different parts of the system. The simulation approach consists of applying a sequence of stimuli (a testbench) and comparing the results with the expected response. The quality of the validation relies on the testbench's ability to detect the errors. Simulation is powerful for high level design, but the increasing complexity of systems results in prohibitive running time as implementation details become fine. Let's consider a low complexity system which performs edge detection on a 256x256p_ picture with the Sobel algorithm on a 3x3 matrix. Table 1 presents the simulation time for the whole picture. The order of magnitude between these results clearly shows that prototyping is an excellent issue to simulation time.

Description Level	ISS in SPW	VHDL Simulation*	Real world Prototype
Simulation Time	10 mn	20 h	0,5 s

Table 1 : Verification time for a DSP–based edge detection system
*: Vulcan VHDL simulator on a SPARC20 workstation

As systems become heterogeneous (HW/SW, IP/Specific), links between CAD tools and real prototyping seem to be the right way to mixed simulation of heterogeneous modules and/or models (Virtual/Real); designing those systems becomes incremental. We first present an overview of prototyping. The following deals with co-simulating heterogenous systems with heterogenous models in a SPW/Aptix framework.

2. FAST PROTOTYPING: STATE-OF-THE-ART

A prototyping system is actually built with FPGA components. There are 3 main topologies for prototyping systems: specific boards, emulation machines and reconfigurable platforms.

2.1 Specific Boards

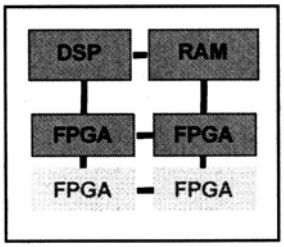

Figure 1. Specific board topolgy

The specific boards [3][4][5] are built for prototyping a given system, or at least systems having a given topology. These boards are actually made of one processor, connected to standard memories and FPGAs for custom logic. It is a low cost solution which allows prototyping of medium complexity HW/SW systems (for example: JPEG algorithm) [6]. The main drawback with these boards is their poor level of reusability.

2.2 Emulation Machines

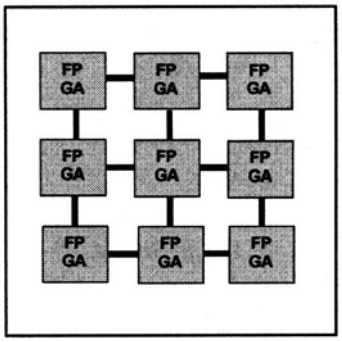

Figure 2. Emulation Board Topology

These machines are built as a network of FPGA, and have the capability to implement large designs [7]. Some of them are using proprietary FPGAs and can thus provide some estimations about the performances of the integrated system [8]. The HDL of the whole system is synthesized for the corresponding FPGA technology. One drawback with this approach is the availabilty of an HDL model for the cores. In the case of an hard-core processor, synthesizing its HDL code may not reflect some implementation details very well. Let's mention that some new emulation machines allow the use of off-the-shelf components, and so the evaluation of hard-cores [7][9]. In [9], the machine is built as an array of specific processors instead of FPGAs.

An other drawback is the price of these machines. They are very expensive, and confront the users with heavy maintenance problems.

2.3 Reconfigurable Platforms

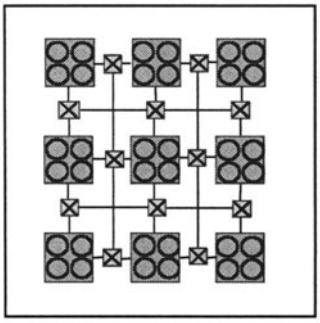

Figure 3. Reconfigurable Platform Topology

This is the most recent way in fast prototyping. These boards are built around Field Programmable Interconnexions Components (FPIC) [10]. The platform consists of a large number of interconnexion holes, which can be connected to each others through FPICs. FPGA or other off-the-shelf components lie on daughter card modules to be plugged into the holes. From a top-level view of the system and a description of each module's pinmap, the associated software automatically performs FPIC and FPGA configurations. The low delay (2-5 ns) introduced by the FPIC devices allows high-speed prototypes. A debug mode is also available in which one can probe any signal from the design into a logic analyzer. This platform has been succesfully used for a demonstrator we built with a d950 DSP (ST-Microelectronics) and an Altera FPGA, presented during the DATE99 exhibition (Aptix booth). A reconfigurable platform is very useful for system validation with a CAD environment, as presented in the following.

3. A GLOBAL PROTOTYPING ENVIRONMENT

Due to the wide variety of models used in the design of complex systems, a global simulation environment is needed. We have made the choice of an industrial frame work with SPW[2] and Aptix MP3 [10]. SPW is proposed as an algorithmic design tool dedicated to signal processing applications. It can handle synchronous or asynchronous dataflows, as well as cycle-based simulation. Overall, this tool offers the ability to co-simulate heterogeneous models. Fig. 4

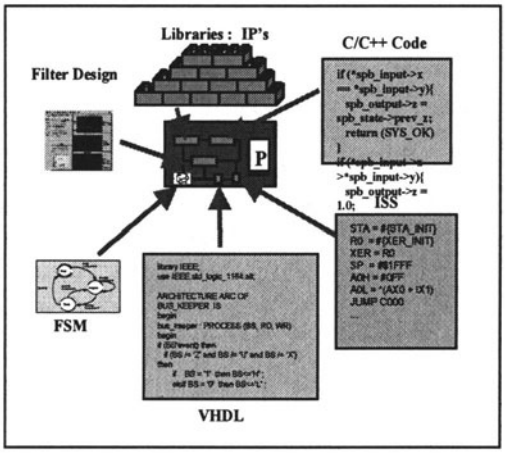

Figure 4. Virtual Prototyping with SPW

presents virtual prototyping with SPW. Within this framework, the system is first designed at a high level of abstraction as a dataflow. The system is built as a set of communicating blocks whose behavior can be specified with an iteration function in C or C++ code. Then an architecture can be developped and refined till the assembly code for software parts and RTL for hardware elements. During these design steps, the whole system is verified by simulations. As previously said, a fine grain of detail will result in long simulation times for complex designs. Therefore, the system is finally verified by running on real world prototype. Fig.5. represents this verification design flow.

3.1 Design space exploration

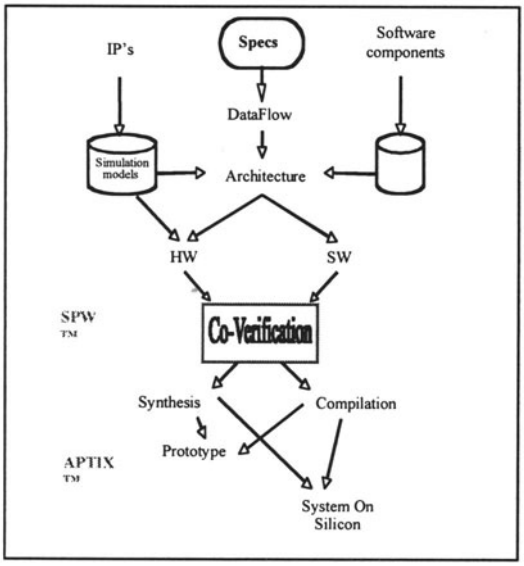

Figure 5. Co-Verification Design Flow

Let's consider the edge detection system described above. The specification was the Sobel algorithm coded in C. We first build a data-flow model of the system, reusing the specification's C code for the computing block. The second step consists of creating the I/O buffering parts, since the original code was using a file system. Here begins the design space exploration. Multiple buffering architectures can be evaluated : using a RAM implies using a communication protocol. Because we need three new points for each pixel, it would mean using either a three times faster clock or three parallel RAMs. On the other side, buffering with FIFOs would solve those problems, but is less reusable (fixed size) and could result in a prohibitive area for larger picture sizes. Note that this is dependant of the target architecture; for a DSP-based system, the processor's memory may be used. It is thus important to complete early evaluations of different solutions.
The system has been first implemented on a DSP and simulated with the processor's ISS. The system was working, but because of the huge data volume of pictures, the image processing rate was not satisfying the specs' constraints.

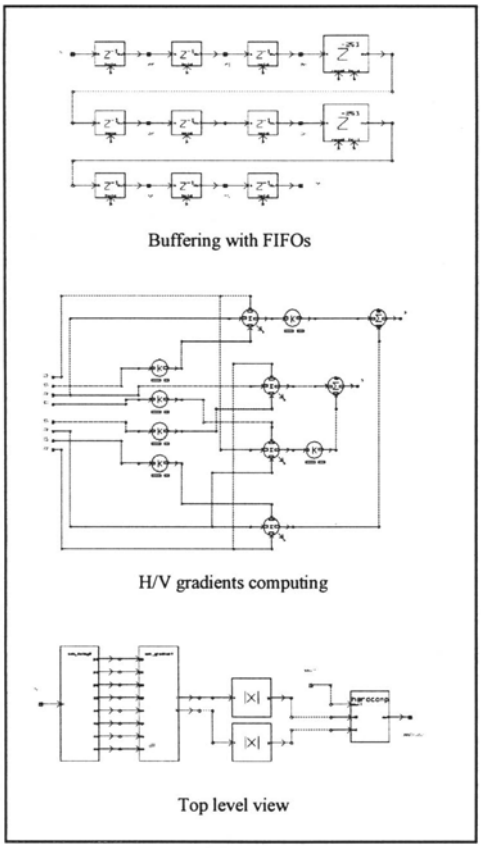

Figure 6. Edge Detection System : Full Hardware Architecture

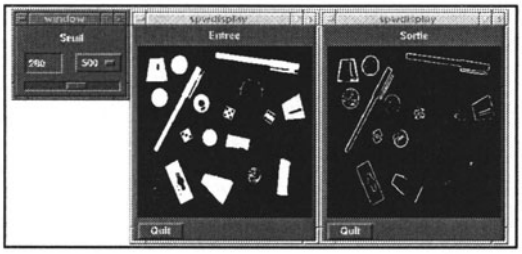

Figure 7. Interactive Parameter Tuning : Subjective Testing

A full hardware solution was then designed with the HDS Library from SPW. The global architecture is given in Fig. 6. The inputs are buffered with 3 FIFOs which can provide the eight required pixels. Then the horizontal and vertical gradients are computed, and their absolute values are added. Comparing this result with a threshold will give the output value for the current pixel. The system was first designed with floating point blocks. The next step was the conversion to the fixed-point system. Overflow and truncation effects are enlightened in order to find the smallest buses sizes (leading to smaller area and power consumption) guaranteeing a correct behavior. At least, the HDL code for the whole system is automatically generated.

The virtual prototype allows verification of course, but also parameter tuning for subjective testing. The threshold constant is determined by using a scroll bar from the Interactive Simulation Library. During simulation, we can tune this parameter according to the subjective quality of the output picture (Fig. 7).

4. TRENDS FOR PROTOTYPING METHODOLOGIES

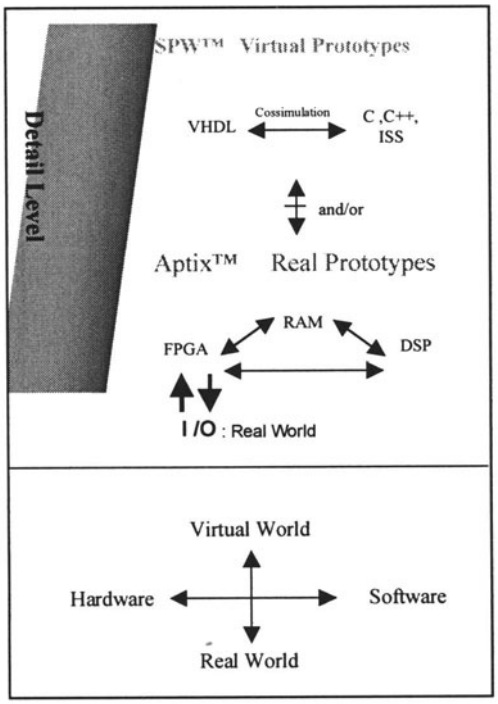

Figure 8. Co-verification Space

Because verification handles heterogeneous models, the verification space can be seen within two dimensions (Fig. 8). Therefore, we believe that an efficient verification environ ment must be able to perform validation with models located anywhere in this verification space. It represents an issue to prohibitive simulation time and also offers new prospects.

First, it allows providing the software team with more and more rapid and accurate proto types. There is another benefit with this approach: the availability of the models. If you plan to use a hard-core, the vendor may not distribute an HDL netlist, but will be able to provide an off-the-shelf component or at least a bounded-out core. In such an open verification environment, it becomes possible to develop the system, even at high level, with high-speed simulation and fine details.

Moreover, it bridges the implementation gap. When a circuit is designed and verified at high level with a virtual prototype, and later globally refined for implementation, there is a lot of work (and thus many possible errors) between them. This problem is more and more frequent with the increasing complexity of systems. When the system does not work after implementation, a lot of work is required to localize and identify the faults sites. An environment which can handle virtual models as well as real models allows the incremental migration from virtual prototype to real world prototype. After the migration of each part of the system, a verification is performed. If the system does not have the required behavior, the last implemented part of the design is analyzed to find the errors. In fact, this approach provides an easier diagnosis of faulty systems during implementation.

5. REFERENCES

[1] http://www.semichips.com

[2] http://www.cadence.com/alta/products/new datasheets/spw.html

[3] J.M Arnold, D.A. Buell, E.G. Davis, *"Splash2"*, Symposium on Parallel Algorithms and Architectures, ACM, June 1992, pp.316-322.

[4] P. Bertin, D. Roncin, J. Vuillemein, *"Introduction to Programmable Active Memories"*, Systolic Array Processor, Prentice Hall, pp. 301-309, 1989.

[5] R.W Hartenstein, J. Becker, R. Kress, H. Reinig, *"High-performance Computing using a Reconfigurable Accelerator"*, CPE Journal, Special issue of concurrency: Practice and experience, John Wiley & Sons Ltd. , 1996.

[6] S. Pillement, L. Torres, M. Robert, G. Cambon, "*Rapid Prototyping – A case study : the JPEG compression algo*rithm", Intl. Conf. On Rapid System Prototyping, Clearwater, Florid, June 15th –17th 1999.

[7] N. Zafar, "*Using emulation to cut Asic and system verification time*", Computer's design, Pennwell Publishing Company, April 1994.

[8] http://www.mentorg.nl/celaro/index.html

[9] http://www.quickturn.com/products/cobalt.htm

[10] http://www.aptix.com