HW/SW Implementation from Abstract Architecture Models

Ahmed Amine Jerraya
TIMA Laboratory - 46, av. Félix Viallet - 38031 Grenoble – France

1. Embedded System Design Challenges both System and Semiconductor Houses.

The evolution of technologies is enabling the integration of complex platforms in a single chip; called System-on-Chip, SoC. Modern SoC may include one or several CPU subsystems to execute software and sophisticated interconnect in addition to specific hardware subsystems. This is no more an advanced research topic for academia. 90% of SoCs designed since the start of the 130nm process include at least one CPU. Multimedia platforms (e.g. Nomadik and Nexperia) are already multi-processor systems-on-chip (MPSoCs) using different kinds of programmable processors (e.g. DSPs and microcontrollers). This trend of building heterogeneous multi-processor SoCs will even accelerate. It is easy to imagine that the design of a SoC with more than a hundred processors will become a current practice in a few years time, e.g. with 45nm technology in 2008. Compared with conventional ASIC design, such a multi-processor SoC is a fundamental change in chip design. These chips will include very sophisticated interconnect such as networks-on-chips (NoC). Moreover, to achieve the required communication performances, each processor may use different local architectures and communication schemes (fast links, non standard memory organization and access).

The design of a system-on-a-chip generally requires the development of complex software, entailing hundreds of thousands of lines of code, to run on the SoC platform. All that work must be accomplished within the context of short time-to-market window constraints and ever increasing functional complexity. Current ASIC design approaches are hard to scale up to such a highly parallel multi-processor SoC. Designing these new systems by means of classical hardware methods gives unacceptable realization costs and delays.

Mastering the design of these embedded systems is a challenge for both system and semiconductor houses that used to apply only software strategy or only hardware strategy. In addition to classic software and hardware that can be designed by software and hardware engineers, SoC design requires the design of hardware-dependent software and software-dependent hardware. In order to meet performances requirements, these two parts need to be jointly designed. This requires 2 key technologies (i) how to abstract HW/SW interfaces to capture the high level architecture model and (ii) how to design efficient hardware/software interfaces, in addition to designing hardware and software starting from this abstract architecture model.

2. Traditional Design Methodologies Feature Discontinuities.

Traditional SoC design approaches are based on serial methodologies. After finishing the hardware platform design, an operating system and/or middleware is chosen and tested on the hardware platform, and then the application software is ported to the operating system and/or middleware. This means that the software design can be finished only after finishing the hardware platform design**.** This often leads to poor hardware designs, since problems caught during software development cannot be fixed in the platform. It also means that the design process takes too much time.

SoC designers need abstract models of both software and hardware components. Ideally one would like to have a set of SW tasks communicating with a set of HW subsystems. Because software components run on processors, the abstraction needed to describe the interconnection between software and hardware components is totally different from the existing abstraction of wires between hardware components as well as the function call abstraction used to describe software. The HW/SW interface needs to handle two different interfaces: one on the software side using APIs and one on the hardware side using wires. This heterogeneity makes HW/SW interface design very difficult and time-consuming because the design requires the knowledge of both software and hardware and their interaction.

In SoC design, HW/SW interface abstraction must take into account the fact that the ultimate hardware/software interface is the CPU and thus any abstraction of HW/SW interfaces requires the hiding of the CPU. A CPU is a hardware module that executes a software program. From a software side, the abstraction hides the CPU under a low level software layer; examples range from basic drivers and I/O functions to sophisticated operating systems and middleware. From the hardware point of view, HW/SW interface abstraction hides the details of the CPU bus

978-3-9810801-2-4/DATE07 © 2007 EDAA

through a hardware adaptation layer generally called the CPU interface. This may range from simple registers to sophisticated I/O peripherals including DMA queues and sophisticated data conversion and buffering systems. This double definition of HW/SW interfaces has created a great confusion in the community and left HW/SW interfaces as an unexplored no man's land.

The same discontinuity existed in computing design where system designers must also consider both hardware and software, but the two are generally more loosely coupled than in SoC design. As a result, computing systems generally model HW/SW interfaces twice. HW designers use a HW/SW interface model to test their hardware design, and software designers use a HW/SW interface model to validate the functionality of their software. **Using two separate models induces a discontinuity between hardware and software**. The result is not only a waste of design time but also less efficient, lower-quality hardware and software. This overhead in cost and loss in efficiency are not acceptable for SoC design. A single HW/SW reference model needs to be shared between both hardware and software designers in order to create the hardware-software continuum required for efficient SoC design.

3. Combining ARTEMIS & ENIAC to Create a Design Continuum

The combination of both strategic agendas of ENIAC and ARTEMIS is aimed to create a design continuum from system level down to Silicon. The concept of "cross-domain architecture" proposed by ARTEMIS is aimed to abstract both software and hardware aspects of embedded systems in a unified platform model. Such a platform will be made of software and hardware components interacting through abstract hardware/software interfaces. This concept opens new vistas that will bring fundamental improvements to the design process:

- concurrent design of both hardware and embedded software, leading to a shorter time-to-market;
- modular design of hardware and software components, leading to clearer design choices when building complex systems; and
- easier global validation of embedded systems including hardware and embedded software, leading to increased reliability and improved quality of service.

ENIAC complements this agenda by offering a path to Silicon starting from these abstract architecture models by providing all the tasks related to hardware implementation of the abstract platform. It also provides the Hardware dependent Software (HdS) required to run the software components of the abstract architecture model on the final hardware platform.

Thus a seamless integration is achieved between the higher level of abstraction dealt with by ARTEMIS and those dealt with by ENIAC. This continuous flow will allow better architecture exploration at system level (e.g. different partitioning and component selection) and at implementation level (e.g. different organization to achieve different performances requirements).

For example, a car maker would like to use a standard cross-domain architecture model to develop the car's software while keeping the right to select the hardware implementation as late as possible. This scheme opens the design process to several new optimizations that were not possible when using the classical design scheme. The most obvious optimization is a better adaptation of the CPU to both HW and SW interfaces. For example, new flexible processor technologies can be used to optimize performances of the HW/SW interfaces by introducing an application-specific I/O operation. Another example may be the use of the capabilities of reconfigurable hardware to optimize hardware interfaces to an embedded CPU.

4. Summary

Early HW/SW codesign research concentrated on HW/SW partitioning, but without solving the problem of abstracting the hardware/software interfaces. Rather than using ad-hoc models of hardware as has been done with traditional design approaches, system-on-chip designs demand a well-thought-out approach to the hardware/software interface. Architecture models can abstract HW/SW interfaces at different abstraction levels such as data transfer, synchronization, interconnect, communication and finally HW/SW partitioning. These different abstraction levels correspond to different refinement processes that require specific cooperation between hardware and software designers. Separate SW and HW design methodologies do not meet the requirements of complex SoC design. Thus the key challenge is the creation of continuum between hardware platform and embedded software. The combination of ENIAC and ARTEMIS platform provides such a continuum and opens new vistas leading to master the ever growing complexity of embedded systems.

5. Bibliography

[1] ENIAC Strategic Research Agenda 2005, <http://www.eurosfaire.prd.fr/7pc/doc/1147246667_eniac_strategic_research_agenda_full__2005.pdf>

[2] ARTEMIS Strategic Research Agenda 2005, <http://www.artemis-office.org/DotNetNuke/Portals/0/Documents/sra.pdf>

[3] MEDEA+ Design Automation Roadmap 2005, <http://www.medeaplus.org/web/communication/publ_eda.php>