VCore-based Design Methodology

Michiaki Muraoka, Hideyuki Hamada, Hiroaki Nishi, Toshihiko Tada,
Yoichi Onishi, Toshinori Hosokawa, Kenji Yoshida
Semiconductor Technology Academic Research Center (STARC)

Abstract

The VCore [1](*) based design methodology, which has been developed at the VCDS (**) Project, is a SoC design methodology using VCores. A VCore is a reusable, high level abstracted design component. We have developed the VCore based design methodology and the VCDS tool prototype. We used the developed tool and did a trial SoC design. The design result showed that SoC design productivity improved using the proposed methodology.

(*) VCDS: Virtual Core based Design System

(**) VCore: Virtual Core

I. Introduction

The re-use methodology at the register transfer level (RTL) became popular in the area of SoC design in late 1990's. However, the IP based methodology at RTL has not been in widespread use yet, because of several technical problems such as interfacing difficulties and process portability. These problems make the re-use of design components difficult.

In order to solve these problems, we have proposed the VCore, which is a reusable, high level abstracted design component, and VCDS, which is the design system using the re-use methodology based on VCores, for the high level design of SoCs in the VCDS project.

VCores, which are reusable, high level abstracted (such as the system level design or architecture level design) design components, are categorized into three types: Functional VCores used at the system level; Hardware VCores used in hardware parts of the architecture design; and Software VCores used in software the parts of architecture design.

At the system level design, the function of a SoC is defined by the combination of functional VCores and the definition of the communication between Functional VCores.

At the architecture level design, the hardware and software are partitioned based on the functional definition at the system level, the required performance, and the design constraints. From the partitioning a hardware architecture and a software architecture is obtained. The hardware architecture consists of Hardware VCores and

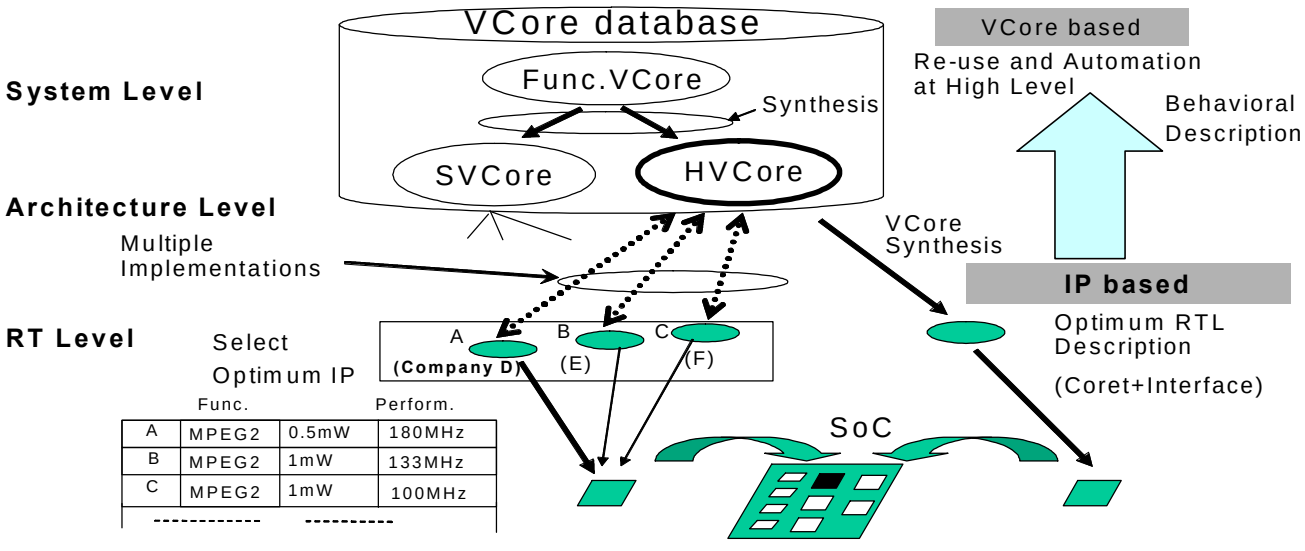

Figure 1. VCores vs IPs

the communication among hardware VCores. The software architecture consists of Software VCores and the communication among Software VCores. These VCores are described using C-based system level description languages such as SpecC or System C.

There have been several research works on high level design methodology such as Ptolemy [2] [3], Polis [4], OCAPI [5], which are targeted to the system level design of SoCs. But the cited research works have not yet established, or considered the re-use of design components at a high level design abstraction. Although there is a research work [6] on the re-use at the architecture level, the re-use at the system level design will be a major research issue towards the era of 65nm process technology, when a SoC will have more than 100 million transistors.

II. IP vs. VCore

In order to solve the problems of RTL-IP re-use, the VCore has been proposed as one of the major candidates for re-usable design components at the high level design of SoC. Figure 1 shows the relationship between VCores and IPs.

There are three types of VCores: Functional VCores used at the system level, Hardware VCores and Software used at the system level, Hardware VCores and Software VCores used at the architecture level. Functional VCore is the component of the function of a SoC at system level. The algorithm to implement the function is described in a Functional VCore using a system level description language. This description does not include details of the actual implementation of hardware or software. Hardware VCore is the component of the hardware architecture. The algorithm to implement the hardware is described in Hardware VCore using a system level description language. This is the upper concept of a RTL-IP. Software VCore is the component of the software architecture. The algorithm to implement software is described in Software VCore using a system level description language. This is the upper concept of embedded software.

Figure 1 shows the correspondence between a RTL-IP and a VCore. When the engineer designs a SoC by re-using RTL-IPs, a large number of RTL-IPs must be provided, and the designer chooses the best IPs from them to design a SoC. Although each application domain requires 30 to 60 types of IPs to design a chip, a much larger number of types of IPs must be provided because each IP must support a variety of interfaces (such as bus interfaces) for it to be useful in the design of a SoC.

On the other hand, VCores are described at a higher abstraction level than RTL, such as function or behavior. The VCore interface consists of an interface protocol description between VCores (hardware to hardware and hardware to software). This description is synthesized and the correspondent hardware or software interfacing description is obtained. Using this approach will drastically reduce the interfacing problems.

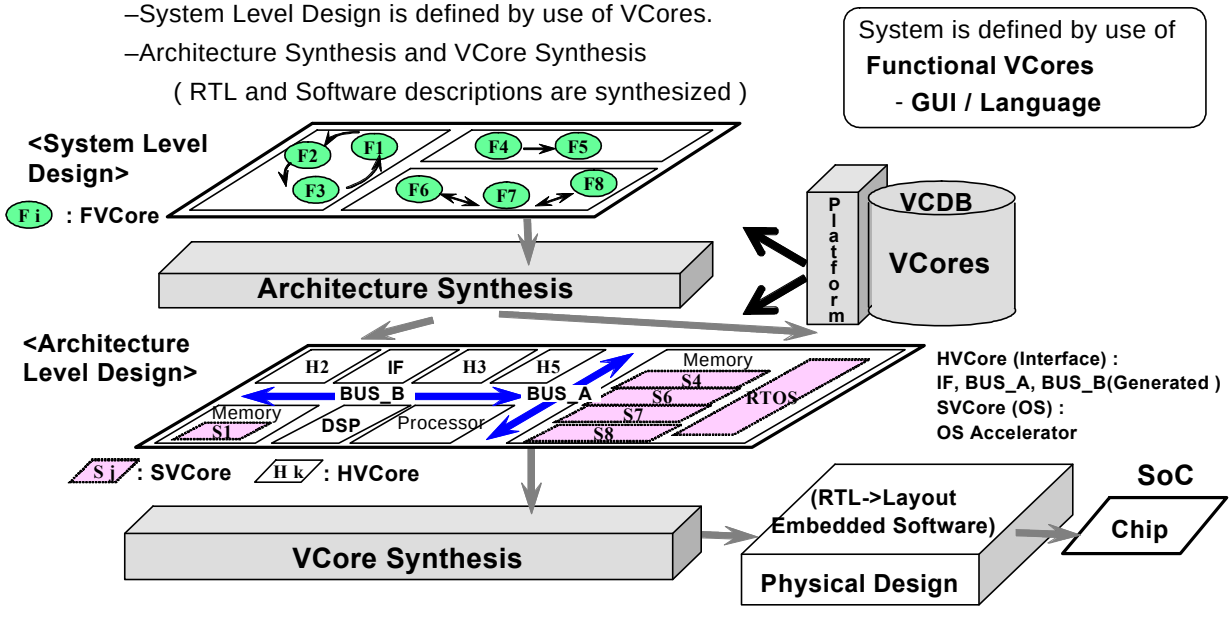

Figure 2. Design by VCDS

III. Design Automation Approach in VCDS

The objective of the VCDS is to accelerate the VCore based re-use design by developing tools to automate the methodology. The design image of VCDS is shown in Figure 2.

At the system design level, the function of a SoC is defined by selecting the adequate Functional VCores stored in VCore database. A graphical user interface tool is used to select the Functional VCores, which are graphically connected to generate a function diagram. The combination of Functional VCores and the description of the communication between VCores define the function of a SoC at the system level.

At the architecture design level, processor, buses and RTOS are selected from the VCore database to generate the basic architecture. Functional VCores in the functional diagram (defined at the system level) are mapped to Hardware or Software VCores to generate hardware and software architecture through the architecture synthesis tools. The performance of the synthesized architecture is evaluated using estimation tools in order to check whether it satisfies the design requirements or constraints. This synthesis procedure is repeated until the design conditions are satisfied, obtaining the final architecture.

The VCore synthesis consists of the hardware synthesis, the software synthesis, and the interface synthesis. The hardware synthesis synthesizes the hardware parts of the architecture to generate a RTL description. The software synthesis synthesizes the software parts of the architecture to generate a software description such as ANSI-C. The interface synthesis synthesizes the protocol description of hardware-to-hardware interface and hardware-to-software interface to generate an interface description. To improve VCore re-use, the VCore platform will be organized to support SoC design. The VCore platform consists of VCore database, VCore development tools.

Figure 3 shows the overview of the VCDS technologies and the descriptions follow.

3.1 System Level Descriptions and Architecture Synthesis

The system level specification description technology consists of the system level modeling and the system level simulation. The system level modeling, which is based on the graphic user interface, is the functional diagram editor. This has the capability of editing the functional diagram by utilizing the objective VCores from the VCore database, putting them on the diagram, and defining the communication between VCores to define the specification of SoC. The system level simulation technology has the capability to simulate the

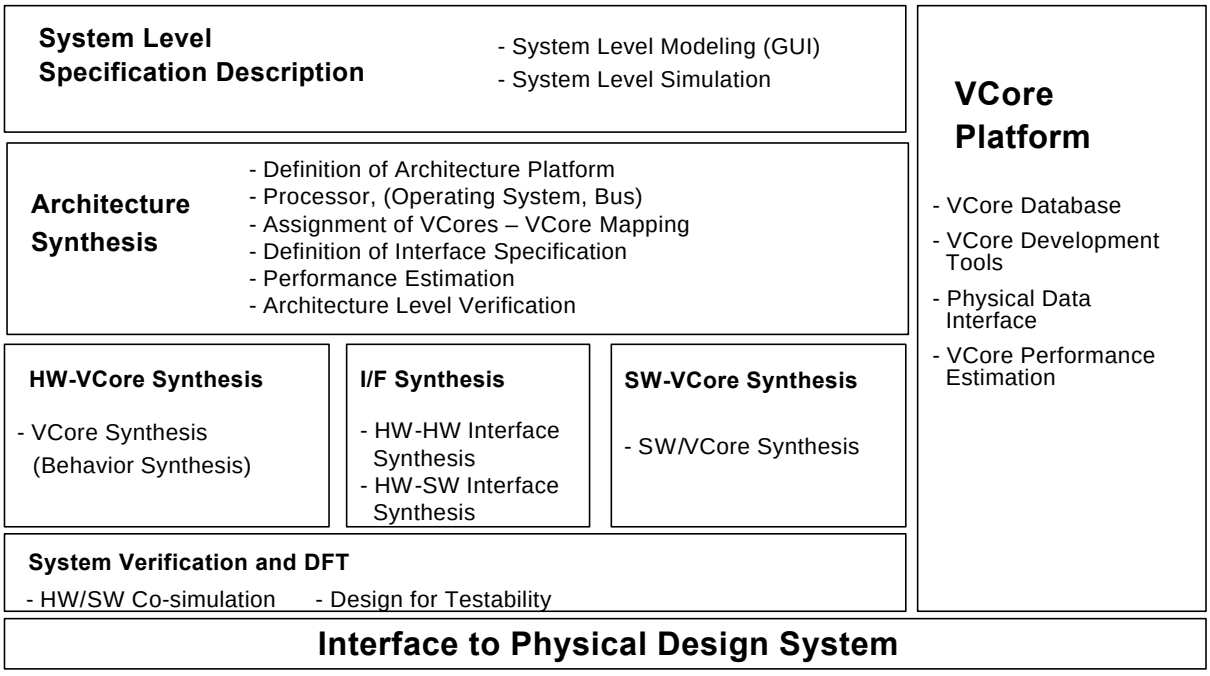

Figure 3. VCDS Technologies

function diagram at the system level rapidly, when the function is defined or changed.

The architecture synthesis technology consists of the architecture template definition, the VCore mapping (the VCore assignment), the interface specification definition, the architecture performance estimation, and the architecture verification. The architecture template (platform) definition is the technology to define the basic architecture for a SoC, and it contains a set of the processor, buses and RTOS with related VCores. The architecture synthesis selects the architecture template and generates the basic architecture, which consists of the processor, buses, RTOS and related VCores. The VCore mapping is the technology to map the hardware VCores or software VCores to the functional VCores in the functional diagram with the consideration of the required performance and the design constraints. The interface specification definition is the technology to generate the protocol description by selecting the protocol template. The architecture performance estimation is carried out by using the characteristics of VCores which are measured or estimated for each VCore. When the estimated performance is not satisfied with the required conditions, the VCore mapping and estimation is repeated until it matches to the requirement. The architecture verification is carried out by hardware/software co-simulation technology. This simulation approach is described in 3.3.

3.2 Synthesis of Hardware and Software VCores

VCore synthesis consists of the hardware VCore synthesis, the software VCore synthesis and the interface synthesis. The hardware VCore synthesis generates the RTL description from the behavior description of VCore. The software synthesis generates the ANSI-C software code and the task assignment description to realize the parallel execution from the software description of VCore written by the system level description language. The interface synthesis generates the interface hardware and software such as the protocol converter and the device driver from the protocol description of the communication between VCores. These VCore synthesis technologies generate the RTL description and the software codes from the VCore description in the architecture. This enables the interface from the architecture level to the RTL and the software code smoothly.

3.3 Hardware/Software Co-simulation by VCores

The simulation approach of VCDS is shown in Figure 4. The target of VCDS simulation is to achieve high speed hardware/software co-simulation by utilizing VCore description, which is the high level abstracted description. The target is to get the speed of 1 to 100 MIPS (#1, 2, 3 in figure 4), although the conventional hardware/software co-simulation at RTL is at the speed of 1 to 10 KIPS (#4, 5 in figure 3).

3.4 Design for Testability

The design for testability (DFT) technology [7] [8] of

VCDS focuses on system and architecture level simulation (#A, B, C) to achieve high speed simulation.

Design Level	Time Abstraction	OS	CPU Model	Software	Hardware	Simulation Performance (IPS)	#	Remarks
System Level	Untimed	-	-	Functional VCore		Slower one or two orders of magnitude than Host CPU	A	Depends on Host Processor
Architecture Level	Untimed	(Host OS)	-	Software VCore	Hardware VCore	10^8	B	
	Timed (BCA)	RTOS (μITRON etc.)	ISS	Software VCore	Hardware VCore +Interface	10^6	C	BCA: Bus Cycle Accurate
RTLevel	Timed	RTOS (μITRON)	ISS	ANSI C/C++	RTL Description	10^2~10^3	D	ISS Model for processor
			RTL	ANSI C/C++	RTL Description	< 10^2	E	Full RTL Simulation

Figure 4. Simulation Approach

VCDS consists of the external test of the VCores and the test architecture synthesis. The external test, which has been developed to reduce the time for the SoC test, is based on the non-scan DFT, which we call NS-DFT, and is the replace of the full-scan DFT. The DFT technology between VCores synthesizes the test architecture, which realizes the parallel test of VCores to reduce the time of SoC test. The DFT technology of VCDS is able to reduce the real time test of SoC more than 100 times.

3.5 VCore based Re-use System

The VCore platform technology consists of the VCore database management system, VCore entry tool, VCore performance estimation system and VCore distribution system. These technologies accelerate the re-use of VCores and utilize them efficiently.

IV. VCDS prototype and design experiment

The VCDS project has developed the VCDS prototype (the limited functional version), which could be demonstrated the design flow to prove the VCore based design methodology. We designed VCores and a SoC, which has the size of 200 Kgates, for Wearable Computers as a pilot project by use of the VCDS prototype, and the result could be proved the efficiency of the design methodology of VCDS. Through the designing the SoC, the improvement of the productivity was more than 5 times by use of the prototype and the estimated improvement by use of the enhanced VCDS will be 20 times respectively.

V. Summary and future works

We have developed the VCDS technologies as described. The key technologies such as the architecture synthesis and the HW/SW co-simulation of VCDS have to be brushed up to enhance VCDS as the beta site version. As to the future work, the formal verification technology at the high level abstracted description, which the simulation technology could not cover, will be the next important research area.

Acknowledgements

This work is sponsored by New Energy and Industrial Technology Development Organization (NEDO) as “SoC advanced design technology development project” (VCDS Project).

References

[1] M.Muraoka, VCDS: Virtual Core based Design System”, ASP-DAC 1999.

[2] Shuvra S. Bhattacharyya, Praveen K. Murthy, and Edward A. Lee, “ Synthesis of Embedded Software from Synchronous Dataflow Specifications,” Journal of VLSI Signal Processing Systems, Vol. 21, No. 2, June 1999.

[3]J.T. Buck, S. Ha, E.A. Lee and D.G.Messerschmitt, “ Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems,” Int. Journal of Computer Simulation, special issue on Simulation Software Development, vol.4, pp. 155-182, April, 1994.

[4] F. Balarin, E. Sentovich, M. Chiodo, P. Giusto, H. Hsieh, B. Tabbara, A. Jurecska, L. Lavagno, C. Passerone, K. Suzuki, and A.Sangiovanni-Vincentelli, “ Hardware-Software Co-Design of Embedded Systems: The Polis Approach.” Kluwer Academic Press, Boston, 1997.

[5] G.Vanmeerbeeck, P.Schaumont, S.Vernalde, M.Engels, and I.Bolsens, “Hardware/software partitioning of embedded system in OCAPI-xl "Proceedings of the 9th International Symposium on Hardware / Software Codesign - CODES, pp. 30-35, 2001.

[6] Reinaldo A. Bergamaschi, William R. Lee, "Designing Systems-on-Chip Using Cores", in Proc. of 37th Design Automation Conference (DAC2000), pp.420-425, June 2000.

[7] T.Hosokawa, H.Date and M.Muraoka, "A Test Generation Method Using a Compacted Test Table and a Test Generation Method Using a Compacted Test Plan Table for RTL Data Path Circuits”, in Proc. of 20th VLSI Test Symposium (VTS'02), pp.328-335, Monterey, April 2002.

[8] H.Date, T.Hosokawa and M.Muraoka, "A SoC Test Strategy Based on a Non-scan DFT Method", in Proc. of 11th Asian Test Symposium (ATS'02), pp.305-310, November 2002.