# A Two-week Program for an Platform-based SoC Design

Sanggyu Park and Soo-Ik Chae
*Center for SoC Design Technology,*
*Seoul National University, Seoul, Korea, TN +82-2-880-5457*
*{sanggyu,chae}@sdgroup.snu.ac.kr*

## Abstract

*This paper describes a two-week program for a platform-based SoC design using SoCBase 1.0, a platform developed in the Center for SoC Design Technology. This program consists of 4 lectures and 9 labs. It covers several design steps from the transaction level to the FPGA prototype level for a Motion JPEG decoder. In this program we employed an SoC design flow based on SoCBase 1.0. It is targeted for graduate students and ASIC designers out in the industry. More than 100 engineers and graduate students have completed this program in 2004.*

## 1. Introduction

Recently platform based design (PBD) methodologies have been widely accepted in the SoC design community. A SoC platform contains a library of reusable components and pre-integrated subsystems as well as an effective design flow. By reusing the architecture of the SoC platform, a complex system can be designed with less effort in a shorter period of time. The platform users design derivatives from the already developed SoC platforms by others. Thus, a platform must be provided with enough information to make it easily understandable and usable by the users. SoCBase is an SoC platform developed in the Center for SoC Design Technology, Seoul National University, Seoul, Korea [1]. The SoCBase 1.0 includes a component library, a system template library, and a verification library as well as all its documentation.

A platform must also be accompanied with a good training program so that the platform users can exploit the platform effectively to design derivatives. For this purpose we developed an example design of the *Motion JPEG decoder to* guide the platform users to the design methodology of SoCBase. In this paper, we briefly introduce SoCBase and required EDA tools in Section 2, and describe its tutorial course in Section 3. which is followed by the conclusion.

## 2. SoC Platform : SoCBase

SoCBase includes a component library, a system template library, a verification library and a design methodology that utilizes those libraries effectively. The component library contains more than 30 hardware component IPs including AMBA bus components, memory controllers and I/O peripherals. We provide both transaction level model in SystemC and synthesizable RT-level model in VHDL for each component. Several I/O models including a serial terminal model and a TFT-LCD panel model are also provided for virtual prototyping of the transaction level and the RT level.

The system template library contains several system templates in which a processor core and peripheral components are pre-integrated with AMBA 2.0 buses. The current SoCBase version supports only ARM7TDMI and ARM926ejs cores. Fig. 1 shows an example of the system templates. Each system template includes three prototype models: a transaction level virtual prototype model, a RT level virtual prototype model, and a FPGA prototype model. Table I shows EDA tools for each prototype model.

The verification library is a collection of methods for rapid verification of a system and its components. We developed a SystemC-based verification environment for users not accessible to the commercial verification tools. They can use it without learning a new verification language such as OpenVera or e-Language required for the commercial verification tools [2][3].

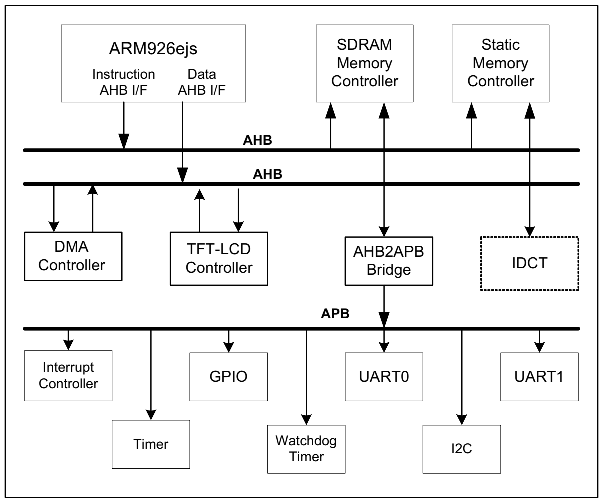

Fig. 1. Example architecture of system template

Table I. EDA tools used for each prototype models

| Prototype models | EDA Tools |
|---|---|
| Transaction level virtual prototype | SystemC<br>CoWare ConvergenSC$^{TM}$<br>ARM ADS$^{TM}$ 1.2 |
| RT-level virtual prototype | Mentor Graphics Seamless CVE$^{TM}$<br>Mentor Graphics ModelSim$^{TM}$ |
| FPGA prototype | ARM ARM Integrator$^{TM}$<br>ARM Multi-ICE$^{TM}$<br>HUINS SoCMaster$^{TM}$<br>Xilinx ISE$^{TM}$<br>Synplicity SynplifyPro$^{TM}$ |

## 3. Motion JPEG Design Program

Users need to know how to use the libraries of SoCBase as well as the EDA tools listed in Table I. The EDA vendors often provide 1-week courses for their specific tools. Therefore it might take several months just to learn all the EDA tools for designing a derivative SoC. Moreover, it is very important to practice not just the point tools but the entire design flow.

We provide a 2-week program, which is one of the deliverables of SoCBase, for a Motion JPEG decoder. We selected the Motion JPEG decoder because its algorithm is relatively simple for students to understand. The course consists of four lectures and nine lab sessions. In the lecture sessions, we introduce basic concepts of the SoC design and the platform-based design methodology. In the lab sessions, students can design a Motion JPEG decoder and verify it step by step from the transaction level to the FPGA prototype level. Each lecture is a three-hour session and each lab is four hours long. All the sessions in this program are outlined in Table II.

This program is targeted for graduate students as well as ASIC engineers out in the industry. Although the participants are assumed to be familiar with the basics of C/C++, we cover the basics of SystemC and VHDL in the program. The transaction level modeling of a Inverse Discrete Cosine Transform (IDCT) accelerator in SystemC is scheduled for the 1st lab session and the RT level modeling in VHDL is scheduled for the 4th. These models are used in the later parts of the program. In the 2nd and 3rd lab sessions, the students exercise design space exploration for the M-JPEG SoC at the transaction level using ConvergenSC$^{TM}$ of CoWare [4]. The students execute and profile the fully S/W implementation of M-JPEG and find out that the IDCT operation is performance critical. And then, they replace the SW implementation of IDCT with the transaction level HW implementation which was described in section 1. In the 5th and 6th sessions, they practice system-level integration and simulation at the RT level with ModelSim$^{TM}$ and Seamless CVE$^{TM}$ of Mentor Graphics. All the H/W and S/W components are given to the students from the component library except the IDCT accelerator which is designed in the 4th lab. Finally, the students exercise FPGA prototyping with ARM Integrator$^{TM}$ in the last three sessions.

Table II. Outline of the Motion JPEG design program

| Category | No | Session Title |
|---|---|---|
| Lecture Session | L1 | Platform-based SoC design methodology |
| | L2 | Transaction level design |
| | L3 | RT-level design |
| | L4 | SoC implementation and verification |
| Practice Sessions | P1 | System modeling in System-C |
| | P2 | M-JPEG S/W Codec Implementation |
| | P3 | M-JPEG H/W-S/W Co-design |
| | P4 | System modeling in VHDL |
| | P5 | HW-SW cosimulation with seamless CVE$^{TM}$ |
| | P6 | RT-level integration of M-JPEG SoC |
| | P7 | Design exercise with ARM Integrator$^{TM}$ |
| | P8 | M-JPEG SoC prototyping |
| | P9 | M-JPEG verification with ARM Integrator$^{TM}$ |

## 4. Conclusion

We have offered this 2-week program four times in 2004, and more than 100 engineers and graduate students have completed this program. We found our program quite effective to educate students to find out what is important in designing a derivative with PBD, but on the other hand, it was a bit difficult for a novice to learn about all the materials in two weeks. Right after each program, we made a survey on students about the quality of the program. Many of them said that although they could follow all design steps in our program, they didn't understand them fully by the time. After several months, we could see some of them coming back and commenting that the design flow of our program was helpful on their personal design projects. Our two-week program does not cover all subjects for the SoC design although it can be a good starting point for beginners. We are presently improving the structure and contents of this program in two aspects; One by covering the back-end design issues such as logic synthesis, design for test, and place and route, and the other one by covering the component and system verification using the verification library of SoCBase, which can be a separate 1-week program.

## REFERENCES

[1] Sanggyu Park, Soo-Ik Chae, "SoCBase : An Integrated solution for platform based design," *International SoC Design Conference*, pp. 862-863, Oct. 2004

[2] Synopsys, "Constrained-Random Test generation and Functional Coverage with Vera," Feb. 2003

[3] Verisity, "Specman Elite," http://www.verisity.com/products/specman.html, 2004.

[4] CoWare, "ConvergenSC $^{TM}$," http://www.coware.com