SoC Design Laboratory
Term Project Part II
Virtual Prototyping & IP Authoring for Baseline JPEG Codec

Instructor: Tian-Sheuan Chang
Announcement: 2004.04.06
Due date: 2004.05.11

Many problems can arise during the system integration process. Moving the system integration phase forward in the design cycle would help in detecting these integration problems earlier. This can be achieved by creating a HW/SW co-verification environment (Figure 1 (a)) early in the design cycle. Soft (or virtual) prototype (Figure 1 (b)) is one of such environment which uses a software representation of the design being verified. Another example of such environment is Mentor Graphics Seamless Co-Verification Environment (CVE) (Figure 1 (c)).

In term project part II, we make use of virtual prototype where the processor is modeled as an Instruction Set Simulator (ISS) and hardware functional blocks are represented with C models. Virtual prototype allows designers to do the following:

- Make trade-offs by modifying system parameters and checking the results
- Test system synchronization such as interrupt handlers
- Develop and test drivers for your IPs

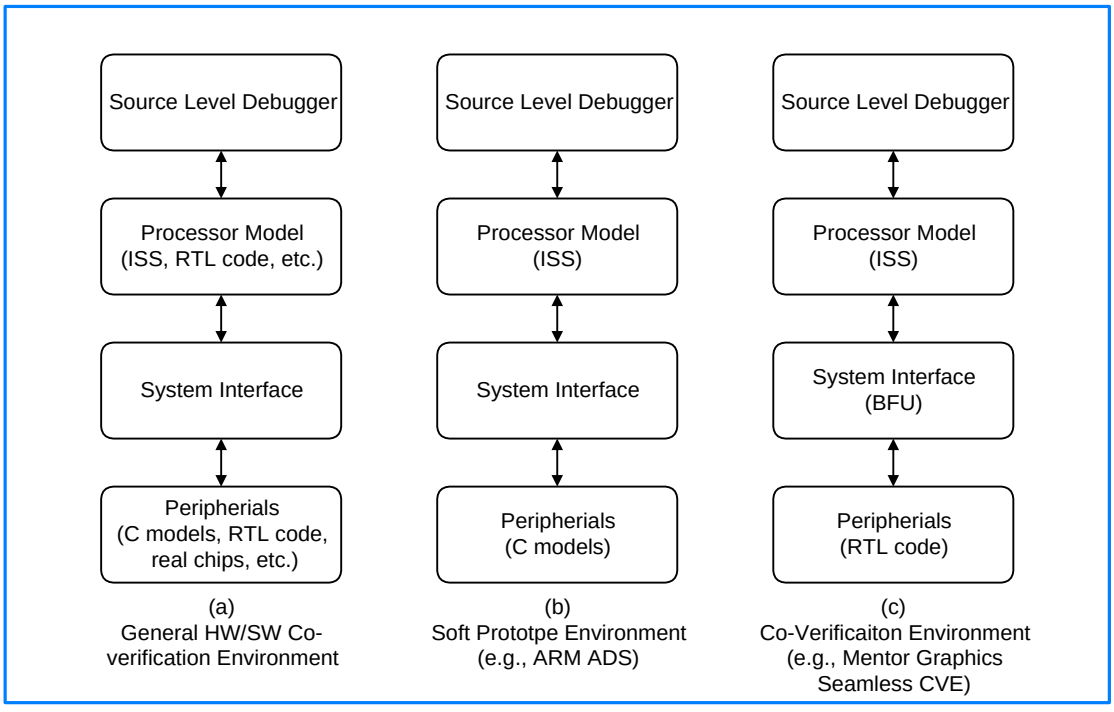

Figure 1. HW/SW co-verification environments.

The software/firmware developed using virtual prototype can be modified for

emulation and downloaded through an ICE to rapid prototype or target hardware system for testing. Two of the limitations of the virtual prototype we have to know are:

- **Accuracy of models**. The hardware models are functionally correct but not pin-accurate. Also, it is often difficult to model exact cycle-accurate hardware designs.
- **Synchronization**. It is usually difficult to resolve the synchronization requirements of the peripheral data dependencies.

As the design evolves, it is important to keep the hardware model in step so that the software development is based on the most accurate estimates of the available timing. If the timing assumption built into the original hardware model is proved impossible to meet in the course of the detailed hardware design, consistency between the models should be maintained.

Using the ARMulator, it is possible to build a complete, clock-cycle accurate hardware model of a system including MMU, physical memory, and peripherals. Since this is likely to be the highest-level model of the system, it is one of the best places to perform the initial evaluation of design alternatives before detailed RTL design. Once the design is reasonably stable, hardware development will probably move into a timing-accurate design environment, but software development can continue using the ARMulator-based model.

After virtual prototyping, you have to represent the target IP in synthesizable HDL. Make sure that your IP is AMBA-compliant.

You have to take the following considerations into account:

- The hardware accelerated IP should occupy the most portion of the overall execution time in pure software. This ensures that the hardware IP would really improve the performance at most. Yet remember NOT to over design your hardware IP with redundant functionalities which the ARM core performs better in software.
- The driver codes would cost some execution time overhead and code size overhead. Lousy driver codes programming might reduce the improvement of the hardware accelerator.
- Pass the Rule Checking of HDL coding conventions that are included in attached file. Explain the reasons of any violation.
- ***Statement***, ***Branch*** and ***state*** (optional: ***trigger***, ***condition***) Metrics of all the sub-modules of your IP must be 100% (exclude your testbench). If not, please explain the reasons.
- Separate your IP kernel and AHB bus interface into different modules and design files. Examples for AHB bus interface can be found in ***Install\LogicModules\LM-XCV600E*** (e.g., C:\Program Files\ARM\Logic Modules\LM-XCV600E\) of PCs in ED414.

The following documents except the first one can be found in ***$VNAVIGATOR/help/pdfs/*** (e.g., /RAID/EDA/VN/help). If you write your design in VHDL, you have better read the relative documents.

- AMBA 2.0 Specifications, ***http://www.arm.com/***

- VN-Check User Guide, 2002.05.02.01
- VN-Cover User Guide, 2002.05.01.01
- Verification Navigator User Guide, 2002.05.00.01
- Verilog rules supplied with VN-Check, version 2002.05
- VHDL rules supplied with VN-Check, version 2002.05

[1] Michael Keating and Pierre Bricaud, "Reuse Methodology Manual for System-on-A-Chip Designs," 3ed. 2002.
[2] FPGA Reuse Field Guide, by Qualis Design Corporation Revision 03_2000_r1, 2000.
[3] Openmore information: http://www.openmore.com

Deliverable

Your deliverable has to include:

1. Report that describes your idea and result.

Port to ARM Integrator :

First of all, finish the ARM integrator part of project1:

- Point out the differences of code segments between your design in ARMulator and ARM integrator, and explain the reasons.
- Performance: use the timers/counters or the time function to record the time your program spends and show it on the host console. Please annotate that cache is enable or not.
- Memory requirement: describe your memory organization for each stage of data processing in detail and explain how it works. Evaluate the maximum memory requirement during your program. Note that the same memory space can be shared with different data structures if their life times are not overlapped. Also, the memory requirement for the program itself and variables have to be mentioned if you modify your program.
- Compare and discuss the results in ARMulator and ARM integrator.
- Evaluate which parts should be improved.

Virtual Prototyping :

Design and model the portions of JPEG baseline codec in ARMulator to improve the coding process using hardware accelerators. All the details of the registers, memory addresses of these registers, bit definitions, and drivers (initialization and interrupt behavior) of your IP(s) should be included in the hardware models. To facilitate testbench migration from this functional level to lower levels, use the bit-true, fixed-point representations in the functional testbench.

Your report should include the description of the functionality of the IPs, the programmer's model (e.g., Section 4.6 of Integrator/AP User Guide), the behavior of the driver (initialization and interrupt behavior), and the coordination of hardware and software.

IP Authoring :

- Architecture of your IP:

I. Describe the architecture and control scheme of your IP.
II. Draw a block diagram of your IP such that each block corresponds to the sub-module of your HDL design.

– Memory requirement:
I. Describe the memory usage of your IP.
II. If your IP requires a large memory, you have to incorporate a memory model, e.g., SRAM model, SDRAM model, embedded memory model.

– Quality of your HDL code:
I. List the results of the check of coding conventions and the coverage of statement and branch metrics in table format.
II. Explain the reasons that the design does not satisfy the rules and the coverage scores.

– Features of your IP:
I. Describe the flexibility of your IP, such as the parameterized design presented in page5-9, chapter 4 of handout.
II. Describe the superior features of your IP compared to the off-the-shelf one.

– Verification plan:
I. Describe the ways you verify the features, flexibility and AHB-compliant interface of your IP.

2. Source code of your JPEG codec, hardware models, drivers, and your IP.
3. All setting and information required for regenerating the result shown in your report. If the results of the check of coding conventions and the coverage of statement and branch metrics can not be regenerated by using your information, the report will not be recognized.
4. Please acknowledge in the last section of your report that you've used the reference code and related documents.

State your approaches, key ideas and results clearly and formally, and avoid redundant description. Your report can be written in Chinese or English. However, make sure your report is readable. A manual report won't degrade your score, unless it is scrabbled.

For more information

- This document is written by Hui-Cheng Hsu, huijane@twins.ee.nctu.edu.tw