

simplex sigillum veri

VaST™ SYSTEMS TECHNOLOGY

U S A • J A P A N • E U R O P E • A U S T R A L I A • K O R E A



Teaching System Level Design Using Virtual System Prototyping of a Digital Radio

simplex sigillum veri

VaST™

Abstract

High-performance, timing accurate models of complex systems (called Virtual System Prototypes or VSP) have enabled the University of Texas at Austin to design a complex system on a chip for a digital radio application. Virtual system prototyping proved to be a superior methodology for architectural optimization, development productivity and accuracy. Virtual system prototyping methodology and technology from VaST systems was used by graduate level students to map a digital radio receiver algorithm on to an ARM processor, build a SystemC based Viterbi decoder and then link to the software using a hardware abstraction layer (HAL).

1. Background and Motivation

The University of Texas at Austin, Department of Electrical and Computer Engineering has designed a graduate course in hardware/software co-design and co-verification with an emphasis on a technique called “virtual system prototyping”. This effort is part of an overall focus on “Systematic Design”. The complexity of modern electronic systems requires the ability to do simultaneous multi-metric optimization in order to converge on the design objectives. It is imperative that this discipline of doing systematic design be part of an engineer’s educational toolbox. A key objective for the course work is that it supports the use of systematic engineering practices from requirements and specification phase through the design, testing and validation phases.

This paper will describe the pedagogy of this new method of preparing student for the real world of system design plus explore the results of using these techniques in student projects. The course is intended to provide the students the following knowledge and skills:

1. A working understanding of the concepts, issues, and process of system-level design of embedded systems including hardware/software co-design, co-verification and debug.
2. Analysis of hardware/software tradeoffs, algorithms, and architectures to optimize the system based on requirements and implementation constraints.
3. Exposure to the specification and modeling of an embedded system at a high level of abstraction.
4. Hardware and software abstraction layers.
5. Use of virtual system prototyping to validate system functionality.
6. Use of multi-metric analysis and optimization techniques.
7. Partitioning architectures into control flow and data-flow dominated real time systems.

2. Selection of the Digital Radio Mondiale (DRM) code base

The DRM code base was selected because it is an open source implementation of a digital radio receiver utilizing algorithms which are used by modern radio systems. The code is designed to primarily run on a 3.0GHz Intel microprocessor and is highly instrumented to provide performance information of the receiver software. The challenge for the students is convert the code base from a PC based application to optimized embedded software and hardware running on a 200MHz ARM based SOC. The algorithms are written using floating point arithmetic operations and must be converted to fixed point operations before the performance analysis operations can be performed. Due to performance limitations some portion of the code must be partitioned into hardware accelerators.

2.1 Details of the DRM System [1]

The DRM is a new digital radio standard for the long-, medium- and short-wave ranges. The standard was formed by a consortium in cooperation with the International Telecommunication Union (ITU). The new system offers the radio stations and new service providers access to the multimedia age with small bit rates for large target areas and long distances. The DRM system uses a type of transmission called COFDM (Coded Orthogonal Frequency Division Multiplex). The DRM system can use three different types of audio coding, depending on broadcasters' preferences. MPEG4 AAC audio coding, augmented by SBR bandwidth extension, is used as a generalpurpose audio coder and provides the highest quality. MPEG4 CELP speech coding is used for high quality speech coding where there is no musical content. HVXC speech coding can be used to provide a very low bit-rate speech coder.

The bandwidth of a DRM bandpass signal is less than 20 kHz and the number of carriers used in the OFDM-modulation is relatively small (max. 460). This allows a real-time software implementation of a DRM-receiver on an embedded processor. A long, medium and short wave front-end with an intermediate frequency (IF) between 5 kHz and 15 kHz is used to receive the DRM signal.

2.2 ARM Platform

The design platform (Figure 1) that the students use consists of an ARM processor, I/O devices, memory components, hardware accelerators interconnected via the AMBA bus.

The final platform configurations will depend on the target markets that require DRM technology. The ARM processor was chosen because it is used for a majority of the components developed for the target market. The DRM system is given a budget of 35% of the ARM CPU cycles. This requires that the student partition some portion of the design into hardware accelerator(s) to meet the performance requirements. There is a power and die size budget that the student must consider when doing the hardware and software partitioning.

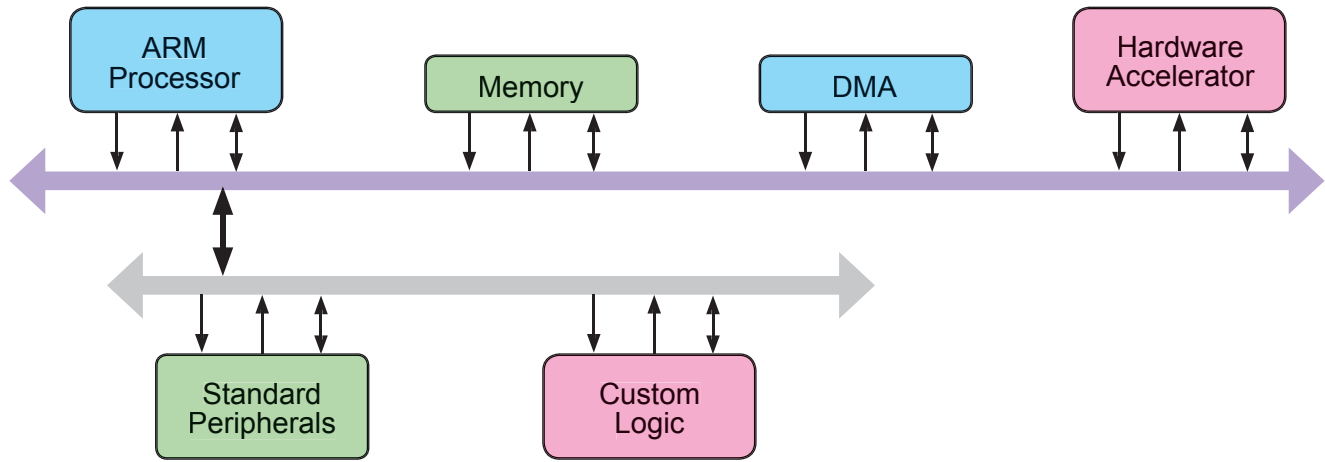


Figure 1.

3. Virtual System Prototypes [3]

A Virtual System Prototype (VSP) is a high performance, timing accurate model of a complete embedded system including software. The hardware platform component of the VSP is called a Virtual Prototype.

Characteristics such as performance and power for a complex system cannot be represented and computed as a formal mathematical problem. The only realistic solution for determining such characteristics is some form of simulation. One option for this simulation is hardware acceleration and/or emulation. Unfortunately, in addition to providing only limited visibility into the inner working of the system, the highest level of abstraction supported by these solutions are register transfer level (RTL) representations. As a result, development and evaluation cannot commence until a long way into the design cycle when the hardware portion of the design is largely completed. In turn, this limits the design team's ability with regard to exploring, evaluating and optimizing the hardware architecture. In addition, FPGA implementations of processors typically are slow, executing software at around 1 MIP – about 50 times slower than a virtual processor model of the same processor.

The related concepts of virtual prototypes (VPs) and virtual system prototypes (VSPs) provide a solution. A VP is a functionality-accurate and timing-accurate software model of the hardware portions of an electronic system. Such a model will typically include processor cores, memory subsystems, peripherals, buses, bridges, mechanical and RF devices, and so on. By comparison, a VSP is a model of the entire system: that is, the combination of the VP and the software that will run on it. Fully evaluating the characteristics of a complex system may require performing many hundreds of experiments on various system configurations. Furthermore, it is not unusual for a single simulation to require that 100 billion instructions be run to reproduce a problem or to compute a representative result. This represents less than one hour of simulation time using a high performance, timing-accurate VSP. By comparison, the same software simulation would take between 100 to 500 hours or more using a typical timing-accurate structural instruction set simulator (ISS) model and 100,000 hours or more using an RTL model. A key

advantage of using a VSP is that the hardware and software portions of the system can be developed and evaluated concurrently. A VSP allows different hardware architectures to be quickly and easily tested and analyzed under real software workloads. While hard real-time software code is being developed, its execution yields trace data (from probes inserted into the models) from which performance (timing, reaction times, latency times, etc.) and power data alongside normal debug data.

4. Use of Virtual System Prototyping in the SOC Class

The SOC design course makes extensive use of the system design tool, COMET, from VaST Systems Technology for hardware/software cosimulation and co-verification purposes. The speed and the accuracy of the system simulation is important in the system design process, since often time, it takes from hours up to days to finish one simulation. Given the tight schedule of the labs and the project of this course, students cannot afford to wait for that much time to see the results. The COMET tool offers fast simulation time with timing accurate models and this considerably reduces the overhead time that students needed to spend on simulation. Also, COMET offers various ways of profiling the system design which helps students and instructors to judge different system design on fair basis. In addition to COMET, the ARM Development Suite (ADS) and the GNU tools are used for this course. The design flow is shown in figure 2.

In this section, we will describe the flow of the labs and the term project of the course and how the COMET tool is used in each stage to help students designing the DRM system. The students are required to complete four labs and a term project. It takes approximately 3 weeks to complete each lab and the students have to finish each lab completely to proceed to next lab since they are closely related with each other.

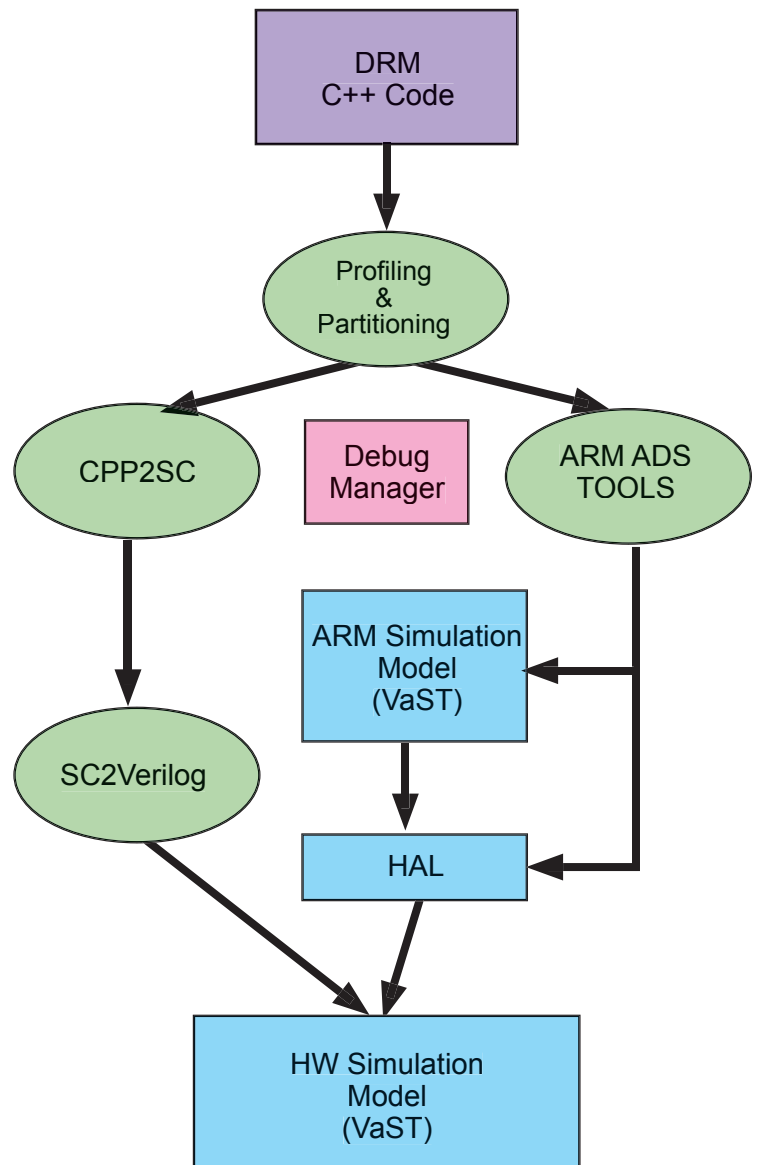


Figure 2.

The students are encouraged to work as a team, but, they need to turn in their own results. Starting LAB-4, the students work as a team of 3-4 people. This policy helps students to learn from each other and come up with several different approaches of designing embedded DRM system that can be used for their term project.

4.1 LAB-1

The primary goals of LAB-1 are:

1. Learn how to use CoMET
2. Analyze the source code of the target system which is a software radio, DRM
3. Profile the DRM software on Linux to find the time-consuming functions
4. Design and simulate a simple SOC design in CoMET, the design environment for the course

LAB-1 consists of two parts. In the first part, students are expected to profile the DRM code and find which functions are bottleneck of the DRM system. They are required to profile the DRM code on two different platforms – X86 and ARM. GNU command ‘gprof’ is used for Linux (X86) profiling and the ARMulator for ARM7TDMI is used for ARM profiling. In both cases, the Viterbi decoding function is on top of the most time-consuming function list. The difference is, unlike the Linux profiling result, ARM profiling results have floating point operations which consume large amounts of time. This indicates that floating point to fixed point conversion is required to reduce the workload of the ARM processor (assuming that there is no floating-point unit).

In the second part of the lab the students are given a simple SOC system which uses memory mapped peripherals and an interrupt mechanism and are required to make it run on the Virtual System Platform in CoMET tool. SystemC code is used to implement a Fabric module and the ARM code coded in C++ syntax is used as target code. By doing this task, they get to know the functionality of the CoMET tool with hands-on experience and understand the concept of the HAL (Hardware Abstract Layer) and the VSP (Virtual System Prototype).

4.2 LAB-2

The primary goals of LAB-2 are

1. Understand floating point to fixed point conversion
2. Understand the structure of the SystemC code

In LAB-2, students are required to create a fixedpoint version of Viterbi decoder code in System-C for hardware simulation. At first, they convert the floating point variables in Viterbi decoder function to fixed point variables and verify the functional equivalence to the floating-point version. After the floating-to-fixed point conversion, students extract the Viterbi decoder from the DRM code and make it a stand-alone module. In the course of doing this task, they learn the code structure of the DRM code which is important in doing the project. Once they finish making the stand-alone Viterbi decoder, they are expected to verify the functionality of this stand-alone code using their own testbench. The Viterbi decoder module that the students have created in LAB-2 will be integrated into the VSP platform in subsequent labs and used as hardware accelerator for DRM simulation.

4.3 LAB-3

The primary goals of lab3 are

1. Understand the interface between the hardware accelerator and the ARM processor
2. Understand the architecture of a Viterbi decoder

In LAB-3 students are required to integrate the stand-alone Viterbi decoder into the COMET tool. To complete this task, students need to work on both software and hardware part. In this lab, software part corresponds to ARM target code and hardware part corresponds to the Viterbi decoder code written in System-C.

For the hardware part, the students create a System-C peripheral project and interface a System-C device model to CIF (Communication Infrastructure Fabric) platform. This is done by wrapping the System-C device inside a Fabric module wrapper. Template code for this wrapper is generated automatically by the tool and students are required to modify this template and the System-C Viterbi decoder code to make the Viterbi decoder module run in the CIF framework [4].

For the software part, the students modify the DRM source code to get it to run on the ARM platform. (ARM simulator for ARM926EJS is provided by the COMET tool.) Specifically they need to modify the communication scheme to Viterbi decoder. Most students are using a memory-mapped I/O scheme as communication protocol between the ARM and the Viterbi decoder (hardware). The AMBA AHB bus model is used as communication channel between ARM processor and the Viterbi decoder.

At the end of the lab, students are required to run the VSP simulation and verify the functional equivalence to the DRM system in pure software environment.

4.4 LAB-4

The primary goals of LAB-4 are

1. Understand HW/SW design tradeoffs
2. Evaluate performance of the HW accelerator with the metrics from product requirements document.

In LAB-4 the students are expected to analyze the cost metrics of Viterbi Decoder hardware IP. These metrics consists of power, size and speed. To estimate these numbers, students are advised to break down their SystemC code to basic block modules. In this lab the basic block module correspond to basic ALU component such as adders, multipliers, shifters, and logic gate component such as AND, OR, XOR, INV, etc. Students are given a library for each of these modules. This library contains the size and speed information of each module in two different cases which are the area-focused case and the speed-focused case. For example, the adder can be designed to have either small area or fast speed. The small adder can be implemented using carry-ripple adder and the fast adder can be implemented using carry-lookahead adder or tree adder. The library contains the size and speed of both of these implementations so that the students can decide which one to use based on their hardware implementation. The students can include this timing information into the code and run the simulation in COMET to get the total cycle count the DRM simulation. They can use the various API callback function to get the exact number of cycle counts for the various implementations. For the power estimation, students are given the library of power estimation for the basic block modules. This numbers are coming from the TSMC process technology library.

The cost metrics that the students found in lab3 can be used as a reference to optimize their system design in the project.

4.5 Class Project

The purpose of the project is to do a HW/SW co-design of the embedded SOC which is a low power SOC implementation of the public domain DRM. On the course after doing the previous labs, students are equipped with the knowledge of DRM code structure and the trade-off between various cost metrics in system design process. Now, they are using this knowledge to actually implement the low power DRM system design. At the end of the project, each group comes up with the SOC implementation of their design and each group's design is compared with other group's design based on cost metrics such as power, area and the speed. Each group need to deliver their final outcomes in form of Virtual System Platform which can be run in COMET tool.

Since this is project open to various approaches and ideas, each group comes up with quite different outcomes focused on different areas of the SOC design optimization. Following is a few examples of various approaches for DRM SOC design optimization.

1. Optimizing HW/SW communication – It takes considerable time to communicate between hardware and software. Some groups implemented DRM scheme to reduce the memory access time and increase the CPU usage. Also there were some groups that compress the length of data sent across the AMBA bus to reduce the bus access time.
2. Software optimization – Public domain DRM code is not optimized to run on ARM processor. Some groups worked on optimizing the software to convert various floating point operation to fixed-point operation and remove the unnecessary function call.
3. System level optimization – Some groups used the system level approach to reduce the size of the Viterbi Decoder. They researched the specialized Viterbi decoder algorithm for the hardware implementation and implemented those algorithms in VSP.

5. Observations

This system level design class using the DRM code base has been taught for the last 5 semesters. It has been an opportunity to observe some very interesting aspects of not only teaching system level design, but using system level design tools for a real world project. Some of the observations include:

- Student team demographics directly impact the solution space chosen to implement the class project. A team which is software centric will tend to implement more of the solution in software and conversely a hardware centric team will try to convert more of the software into hardware accelerators.
- Software centric projects generally do not meet the performance specifications. The problem is such that the only solution is to move the Viterbi decoder into a hardware accelerator.
- Hardware centric projects will generally meet the performance specifications but not meet the power and die size requirements as there is a tendency to throw too much hardware at the problem.
- It is exceptionally hard to teach hardware engineers to use System-C. They have a tendency to generate Verilog like implementations of the hardware accelerators. This results in long run times and missed schedules. By the end of the semester there is a strong push by the students to use System-Verilog instead of System-C.
- Most teams do not read the specification clearly enough to execute the project correctly despite the emphasis on “systematic” design. Initially there was some attempt by the instructors and teaching assistants to correct the student designs early on in the semester. However, the “learning” aspects of not doing the project correctly provide valuable lessons for the students.

At the end of each semester there is a design review. One of the review requirements is for each team to comment on what worked, what didn't work and what would they do different if they could redo the semester. Some of the more salient comments include:

- Always verify feasibility of the specifications that marketing provides before agreeing to meet them. Specs were unattainable, failed to realize this until late in the semester. Glad this wasn't a commercial venture!
- When measuring performance, always ask what this means for the target platform.
- No amount of hardware acceleration can overcome fundamentally poor software.
- Move some operations done in HW accelerator to software to reduce the complexity of the HW accelerator.
- Focus on the "system" and not only on the Viterbi. Don't use software Viterbi as basis for hardware design.
- COMET tool is a good environment for transaction level modeling. The DeviceAdapter class consists of function calls using DeviceInterface. However, if the API manual had been provided, it would be more convenient.
- Low level timing cycle accurate design hard to implement in System-C.
- VAST tool documentation was good.

6. Conclusions

The VaST environment greatly improves the learning process. Ability to do rapid exploration of design options is essential when teaching metric based system level design.

7. Future Work

There are three areas which need to be improved in the future. These include:

1. Improved debug environment. Students spend a large percentage of the semester dealing with the various debuggers.
2. Improved hardware model generation environment. The transformation from a software algorithm to a hardware accelerator is slow and problematic. This may involve moving to System-Verilog.
3. Generate a MATLAB or SPW model for the DRM system. This may be a better starting point for the project.

8. References

- [1] Volker Fischer, *Software Implementation of a Digital Radio Mondiale (DRM) Receiver, Part I (Framework)*, Institute for Communication Technology, Darmstadt University of Technology
- [2] Source Forge DRM code source
- [3] Hellestrand, G.R. *Using Virtual System Prototyping Technology to Optimize Real-time Systems for Power*, White Paper. VaST Systems Technology Corp.
- [4] *Peripheral Device Builder User guide*, VaST Systems Technology Corp.

