Teaching Trade-offs in System-level Design Methodologies

K. Sakiyama, P. Schaumont, D. Hwang, and I. Verbauwhede
Electrical Engineering Department, University of California, Los Angeles
7440B Boelter Hall, Box 951594, Los Angeles, CA 90095-1594
{kazuo, schaum, dhwang, ingrid}@ee.ucla.edu

ABSTRACT

This paper summarizes two graduate-level class projects in EE201A/EE298 (VLSI Architectures and Design Methods) at the University of California, Los Angeles (UCLA). The purpose of the class is to explore the impact of system-level optimization for various target platforms using EDA.

1. INTRODUCTION

Nowadays, CAD and software development tools are indispensable to develop a complex system on chip in a short time. These tools must be supported with a system-level design methodology in addition to individual hardware/software design. While it is well known that design decisions at system level have most impact on the final performance, there are very few standard textbooks, educational materials or standard methodologies available.

The goal of system-level design is to translate a specification in high-level language (English, Matlab, C/C++) to implementation level, based on requirements such as the throughput, area, power, and the development time. It is important to give students an opportunity to consider a system-design flow in an educational setting and compare the implementation result of different platforms. This concept is illustrated in Fig. 1.

The class of Spring 2000 [1] was divided in 11 teams of 2 or 3 students. They started from an identical LPC Speech Coder and implemented this on five different signal processing platforms: three programmable DSP processors (TI C55x, C54x, and C6x) and two signal processing design environments (Ocapi [2] and A|RT Designer [3]). All five designs were compared based on energy, area, clock frequency/MIPS and design time.

The 13 teams of the class of Spring 2002 [4] conducted a similar experiment to design a high-speed JPEG encoder. In this class, 5 different approaches were taken to implement a JPEG encoder. Students used 3 different design languages (SystemC, HandleC, and SpecC [5,6]), and targeted 4 different platforms: 2 DSP processors (TI C54x, Analog Devices Blackfin), and 2 dedicated FPGA implementations.

Starting from a high-level specification, students thus could compare a wide variety of design approaches.

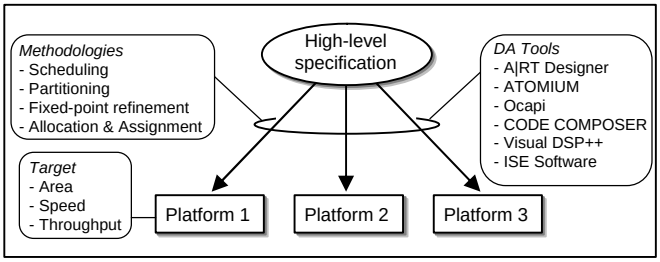

Fig.1. Concept of System Design flow

2. DESIGN METHODOLOGY

This section describes the design methodology for implementing LPC speech coder and JPEG encoder on the various platforms. In both projects, students started from a high abstraction level, optimized it in system-level perspective, and refined the spec to implementation. Students were separated into several teams and tried to find the best design given their platform. A crucial constraint for all these designs is the design time, which was limited to the time of one quarter (10 weeks).

2.1. The LPC Speech Coder project

The Linear Prediction Coefficients (LPC) speech coder was first designed in floating-point format in MATLAB [7,8]. Then, fixed-point refinement was used to efficiently map the algorithm onto fixed-point platforms.

For the platforms of TI DSPs (TI C54x, C55x, and C6x), the internal wordlengths are fixed to 16 bits for most arithmetic operations. There are several criteria which affect the fixed-point wordlength decision, including recognizable synthesized speech, pitch frequency matching, avoidance of signal overflow/saturation at each point in the algorithm, and avoidance of saturation of the synthesized speech. To obtain a fixed-point C++ code suitable for the fixed-point DSPs, students rewrote the entire algorithm using 16-bit C arithmetic.

Both Ocapi and A|RT Designer rely on a fixed-point C++ library. The refinement target for Ocapi is a cycle-true description, while A|RT Designer uses a compilation step to map behavior onto a VLIW architecture template.

All the implementation models were evaluated by simulation in this project.

2.2. The JPEG Encoder project

We started from a reference JPEG encoder implementation in C. At the first step, data-flow analysis is done of the C code to identify the individual processing stages in the JPEG encoder. In addition we also analyze the background memory requirements of the JPEG encoder in order to optimize the memory architecture of the target platform.

In a second step, each team translates the reference implementation in C into a system level modeling environment, which is one of SystemC, HandleC, or SpecC.

The third step deals with fixed point refinement. Depend-

ing on target platform, an appropriate fixed-point refinement has to be done, similar as with the LPC coder.

The last step is the implementation phase, where each team implements their design on one of 3 reference PCBs: A Spectrum Digital board with C5410 [9], an Analog Devices Blackfin evaluation board [10], and an Insight Electronics board with a Xilinx Virtex-II [11].

3. RESULTS

3.1. The LPC Speech Coder project

Table I shows the results for the LPC coder. The Ocapi solution is slightly over half the size of the A|RT Designer solution. However, these figures are somewhat deceptive as both designs were mapped onto a different technology (0.25-µm vs 0.35-µm).

Energy figures for each design are given in Table I in units of energy per frame. These values are based on the simulation and assumed to run the processor at the lowest clock frequency that still guarantees to meet the real-time constraints. The circuit design with Ocapi resulted in the lowest energy per frame.

The technology varies with the platform (0.15-µm for the DSPs, 0.25-µm for Ocapi and 0.35-µm for A|RT Designer). The energy for each design refers to the energy of the core and the memory and not for the I/O. The DSP processors have a much larger memory than needed for the application, which results in a large energy consumption. On the other hand, they are built in a more advanced technology.

The details of this project are reported in [12].

3.2. The JPEG Encoder project

The number of memory accesses was optimized by each team using ATOMIUM [13]. More than half of the teams were able to reduce the access count by 50% or more.

The design time for system-level modeling was two weeks. We observed a big variation in the result obtained with SpecC, HandleC and SystemC [14]. We attribute this variation in results mostly to the lack of a well-documented design approach for some of the design languages at the time of the project.

All of the teams using a DSP platform could implement their design successfully onto the evaluation PCB. However, no team targeting the FPGA board could obtain an implementation within the 10 weeks time limit. The low programming abstraction level was found to be the prime culprit here. Based on the data reported by the teams, the JPEG encode performance is calculated and compared in Table II.

4. SUMMARY AND CONCLUSIONS

Throughout the projects, students observed that system-level optimization had great impact on the final implementation. The obtained results varied strongly with both the target platforms as well as the tools used. DSPs offer high flexibility, easy learning curves and fast design times. More heterogeneous or application-specific architectures offer better performance at the cost of extra design time. Students reported that the material covered in these courses is new and complementary to the material offered in a traditional EE curriculum.

Table I. The LPC Speech coder performance

Platform	Area—Memory	Cycles / Frame [K]	Energy / Frame [µJ]	Power Supply [V]
TMS320V C5410A	8.7 KB	240	144	1.6 core
TI C5510	10.2 KB	120	53	1.5 core
TMS320 C6201	16.0 KB	30	66	1.5 core
Ocapi	1.4 mm^2	11	2.1	2.5
A|RT Designer	3.2 mm^2 2.3KB ROM 1 KB RAM	3	4.3	3.3

Table II. The JPEG encode performance

Platform (source code)	Average of Required Cycles [cycles/64blocks]	Code Length [lines]	JPEG encode Performance [blocks/s]
Blackfin (C code)	1,524K	879	12,602 @300MHz
TMS320VC5410 (C code)	1,499K	707	4,270 @100MHz
A|RT Designer (SystemC)	677K	1,015	-
DK1 (HandleC)	700K	1,312	1,357@15MHz (Simulation Value)

ACKNOWLEDGEMENTS

The authors thank all of the graduate students who enrolled in EE298/EE213A of Spring 2000 and EE201A of Spring 2002 at UCLA. We would like to acknowledge Hitachi, Ltd., Semiconductor & IC Division, 2002 DAC Graduate Scholarship, the Fannie and John Hertz Foundation, and UC-Micro Grant #02-079. We also acknowledge the logistical contributions from Analog Devices, Adelante Technologies, Celoxica, and Texas Instruments.

REFERENCES

[1] http://www.ee.ucla.edu/~ingrid/ee213a/
[2] http://www.imec.be/design/design/ocapi_intro.shtml
[3] http://www.adelantetech.com/en/html/algemeen/AboutAdelante/Partners/Academic/EducationalProgram.asp
[4] http://www.ee.ucla.edu/~schaum/ee201a/
[5] http://www.systemc.org/
[6] http://www.ics.uci.edu/~specc/
[7] L. Rabiner, R. Schafer, *"Digital Processing of Speech Signals"*, Prentice Hall, Englewood Cliffs, New Jersey, 1978.
[8] M.M. Sondhi, *"New Methods of Pitch Extraction"*, IEEE Trans. Audio and Electroacoustics, Vol. AU-16, No.2, pp.262-266, June 1968.
[9] http://www.ti.com/
[10] http://www.analog.com/
[11] http://www.insight-electronics.com/
[12] A. Gatherer, E. Auslander, *"The Application of Programmable DSPs in Mobile Communications"*, John Wiley & Sons, Ltd., New York, Chapter 15.
[13] http://www.imec.be/design/multimedia/atomium/
[14] K. Sakiyama, P. Schaumont, I. Verbauwhede, *"Finding the Best System Design Flow for a High-Speed JPEG Encoder"*, ASP-DAC 2003, Kitakyushu, Japan, January 2003