AN UNDERGRADUATE FIXED POINT DSP COURSE

Wayne T. Padgett

Rose-Hulman Institute of Technology

ABSTRACT

A course is proposed to cover fixed point, quantization and overflow issues in DSP programming for undergraduates. The course covers the material using a simulation approach in a case study of a SSB communication system. The course is needed in the curriculum because most new graduates in DSP jobs need these skills immediately to be productive on the more common fixed point processors.

1. INTRODUCTION

Undergraduate DSP courses tend to emphasize theory, and applications are intended to motivate and illustrate. These courses are generally intended as preparation for more rigorous graduate courses, which also emphasize theory.

In contrast, engineers who program DSP's in industry are confronted with relatively little theory (which is largely left to research departments), and instead struggle with numerous practical issues to which they have had minimal introduction. While it would be impossible to fully prepare undergraduate engineers for their every need in industry (in DSP or any other specialization), I have become convinced that a single course emphasizing a methodical approach to the conversion of algorithms from floating point to fixed point arithmetic would greatly improve the ability of new graduates to step into DSP programming.

This course would be a senior level undergraduate course requiring a basic undergraduate DSP introductory course (ECE380 at Rose-Hulman) as a prerequisite, and offered as a complement to the graduate level DSP theory and programming courses already offered (ECE580 and ECE581).

2. METHODOLOGY

The goal of the course will be to give the students a very light introduction to the many practical issues in embedded DSP programming, and a very strong background in converting floating point algorithms to fixed point by working through a simple SSB communication system design. Note that although communication systems and DSP theory are not central to the course, it is expected that the students will benefit substantially in these areas by looking closely at the details of the system. Hopefully, this practical view will make the advanced theory course even more attractive.

In theory courses, various methods of analyzing filters for quantization error and overflow are usually discussed. Although the analysis is useful for gaining intuition, real systems may defy the required simplifying assumptions, and simulations are often required to test the necessary compromises.

The new course will emphasize a simulation approach based on comparing floating point to fixed point performance. This requires an appropriate simulation tool, which has not yet been finished. The planned simulation tool will be written in MATLAB and designed to simultaneously compute and compare fixed and floating point operations based on DSP primitive operations like those defined in ETSI speech coder standards [3]. The simulator will also collect statistics on signal range and computational complexity.

A methodical approach for fixed point development will be presented and applied to each subsystem of the SSB communication system example.

- Verify algorithm in floating point (MATLAB or C)
- Implement algorithm in a simulator using primitives
- Iterate on overflow and quantization issues until algorithm performance is satisfactory
- Identify algorithm inner loops which would justify hand coding in assembly (no assembly in course)

This process closely mirrors industry practice and will give the students a clear view of how limited precision and computational complexity interact in a relatively simple system.

3. COURSE SYLLABUS

The course will draw on my personal experience with a simple SSB communication design as a case study. After introducing the issues and giving a brief overview of some of the system considerations, the course will follow the signal flow in the SSB transmitter and receiver. Each subsystem will be examined carefully and several alternate implementations considered in terms of signal quality and computational complexity. Effects on system specifications like analog filter requirements and out-of-band energy and adjacent channel rejection will be considered as well. Because the course is not centered around teaching communication systems, a simple working MATLAB implementation of the system will be provided to the students at the beginning of the course for comparison purposes. Because the course is not centered around DSP theory, many of the subsystems will be provided to the students already implemented in fixed point. When appropriate, the students will implement an alternative subsystem, but generally the course will focus on measurements and modification/improvement of working implementations.

The course will be taught in a four hour per week, ten week (quarter) format, with no lab, but regular computer assignments. All students at Rose-Hulman have a required laptop computer, so in-class computer exercises are relatively easy to arrange.

Week 1: System Implementation Overview
Interrupts, block processing, memory speeds, DMA, loop unrolling, software pipelining, power

Week 2: SSB Generation and Reception
Fourier Analysis, Hilbert Transformers, Phase Cancellation, System Specifications

Week 3: Fixed Point Arithmetic
FIR and IIR implementation, Q format, fractional arithmetic, scaling and quantization, double precision accumulator and guard bits, simulation tool introduction

Week 4: System Specification and Testing
Measurements of out-of-band energy, adjacent channel interference

Week 5: Hilbert transformers
FIR filtering, scaling, phase delay

Week 6: Raising the Sample Rate
IIR interpolation

Week 7: Raising the Sample Rate again
Polyphase FIR interpolation

Week 8: Modulation
Generating the sin/cos signal, lookup tables, oscillator

Week 9: Demodulation
Lowering the sample rate and anti-aliasing or image rejection

Week 10: Project
Term project will center on a further optimization of an existing subsystem, or on a new subsystem drawn from conversion of the SSB channel to a data modem.

The term project will probably be assigned at midterm, with the last week of the quarter devoted to resolving issues and encouraging apathetic students.

4. SIMULATION TOOL

A simulation tool is in development using MATLAB functions to do fixed point arithmetic (variable bit precision, to support 16, 32, 40, or others) and floating point arithmetic using a single set of "primitive" functions. The tool's design is intended to allow simple application of the theoretical concepts of fixed point arithmetic. The goal is to simplify the measurement of quantization effects enough that the students will be able to compare the performance of various structures and scaling approaches in a reasonable amount of time. This is important because a full theoretical analysis of several 2^{nd} order filter structures can be extremely tedious.

The simulation tool will make it easy to capture impulse responses at every node in the system, so that overflow values can be predicted and dealt with. Impulse response capture of the fixed point result will also allow frequency response comparisons with unquantized coefficients. With real data inputs, the simulation tool will facilitate the monitoring of actual overflow events, and because both fixed and floating point values are available, a signal to noise ratio computation can be done at any (or every) point in the system. Capturing the noise signals induced at each node should make it possible to measure not only the noise power at the system output, but also the noise spectrum so that a node that contributes to noise in a particular frequency band could be identified individually. The tool will also keep operation counts so that the system complexity can be easily measured. With this tool the students will be able to easily recognize the tradeoffs between scaling and roundoff noise as they experiment with different implementations of the system components.

There are still a number of contentious issues regarding the simulation tool. One obvious question is "why not use the fixed point blockset in MATLAB?" The primary answer is that because the blockset is a graphical tool, the results don't lend themselves to direct translation to assembly language implementation, whereas the primitive operations nearly correspond to individual assembly instructions on most DSP chips. Other reasons

for this choice include the potential for porting the tool to C++ for improved speed, and of course, the cost of adding the fixed point blockset to our software suite. The fact that the fixed point blockset is a mature product with excellent features makes this decision a tough one.

Since the development of the simulation tool is underway, I can comment on some important design choices. A key issue is the method of designating computations which are tracked. At the highest level of automation, all arithmetic operations should be tracked for overflow, etc. and impulse responses at every adder output should be recorded for "potential" overflow analysis. However, it is not trivial to organize the list of arithmetic operations into a set of nodal impulse responses. Even giving a unique name to each computational output can be problematic because some computations may occur different numbers of times for different iterations. On the other hand, manual assignment of nodes (using a logging function) runs the risk of missing important points in the code, and is also subject to computational variations. Another important design issue will involve the tradeoff between speed and complexity: if the tool logs all possible data, it may run too slowly to be useful for any but the most trivial of algorithms. On the other hand, with a full log of all operations, post processing functions could more easily analyze data ranges, compiling histograms of signal values, noise powers induced at each node, and probabilities of overflow.

5. COURSE TEXT

A number of texts have been evaluated for potential use in this course, and since this is a relatively unique course outline, it is not surprising that none matches extremely well. However, it is useful to consider the coverage given to the practical issues of fixed point implementation in a number of current text books.

Oppenheim, Schafer and Buck [2] devote about half of Chapter 6 to fixed point effects, in the context of various filter structures. This is an excellent place for the discussion, since there is little justification for discussing the difference between a direct form II and a direct form II transposed structure in the absence of fixed point effects. Although the theory is presented, there is not much discussion of a methodology for dealing with these effects.

Mitra [3] devotes all of Chapters 8 and 9 to fixed point effects, and provides the most complete treatment of fixed point effects in a general DSP text that I am aware of. He provides many MATLAB examples of how to automate the analysis of basic filter structures and covers many important topics in detail.

Kuo and Lee [4] emphasize system implementation by specializing on the the TMS320C55 DSP chip, which is a fixed point device. They discuss the fixed point issues in an introductory chapter, and in subsections along with most major topics. One excellent feature of this text is that it discusses intrinsic functions thoroughly, with the TI C compiler intrinsics as a fine example. Notably, they outline an implementation methodology similar to that mentioned above in Section 2. Because the text is devoted to real-time processing, it deals with many of the practical issues mentioned in Week 1 of the syllabus above. However, because it is tied to the C55 chip, it would be difficult to use without an appropriately equipped lab (our DSP lab is based on the C67).

Ackenhusen [5] is much more focused on higher level system design and architecture issues. These are all important, especially for guaranteed real-time performance and optimization, but there is not much discussion of arithmetic precision.

Ifeachor and Jervis [6] take a "Practical Approach" which is quite consistent with the theme of the proposed course. They devote all of Chapter 13 to finite precision effects, including several examples of scaling the canonical forms. Apparently a supplement with a software tool for studying filter implementations is forthcoming. This text seems to treat many of the practical issues throughout, and may be a useful text for the course.

There are a number of other texts that might be considered, such as those for the C6X by Kehtarnavaz or Chassaing, but these tend to focus on the more on the specifics of the C6X rather than on the algorithm.

I should also mention the work of the military academies [7] in developing tools to take students from the floating point world into real hardware and implementation. Incorporating ideas from their work has helped to define my ideas for this course.

6. CONCLUSIONS

This paper is intended to promote discussion of the need for an introductory fixed point course in the DSP curriculum, as well as to generate ideas for the improvement of the proposed outline. A major obstacle will be finishing the development of the simulator tool, although a graduate student may well be employed to aid the process. Use of primitive operations similar to those in the ETSI speech coders will reduce the amount of work required in choosing operations, and also familiarize students with real world terminology.

Reasonable parties could easily disagree over the need for, or focus of, such a new course. A number of other issues also have been decided somewhat arbitrarily and may be subject to change. Among these issues are: the substitution of generic primitives for a real assembly language; the decision to run the course without a lab time to focus on the computer assignments; the decision to

focus on fixed point arithmetic rather than the many software/system issues encountered in a DSP embedded system (see Week 1). It is possible that yet another new course could be designed around embedded DSP system software issues, but there are limitations to the number of DSP electives that can be offered at a small school. It is even possible that the fixed point material could be compressed enough to make room for a significant treatment of some of the assembly and embedded issues. However, I am quite wary of the temptation to put too much material into a course and thus ruin the experience for both myself and the students.

11. REFERENCES

[1] ETSI Standard, *ANSI-C Code for the GSM Half Rate Speech Codec,* (GSM 06.06 version 8.0.1 Release 1999), EN300 967, http://www.etsi.org .

[2] A. Oppenheim, R. Schafer, and J. Buck, *Discrete-Time Signal Processing,* 2nd ed., Prentice-Hall, 1999.

[3] S. Mitra, *Digital Signal Processing: A Computer-Based Approach*, McGraw-Hill, 2001.

[4] S. M. Kuo and B. H. Lee, *Real-Time Digital Signal Processing,* John Wiley & Sons, 2001.

[5] J. G. Ackenhusen, *Real-time Signal Processing*, Prentice-Hall, 1999.

[6] E. C Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach, 2nd ed.*, Prentice-Hall, 2001.

[7] C. H. G. Wright, T. B. Welch, D. M. Etter, and M. G. Morrow, "Teaching hardware-based DSP: Theory to practice," in *Proceedings of ICASSP*, May 2002.