**Computer Science Graduate Course Announcement**

**“CS 650: Program Generation and Optimization”**


**Instructor:** Jeremy Johnson, Dept. of Computer Science (jjohnson@cs.drexel.edu)

**Term:** Winter 2010 (Wed. 6-9)

**Course Number:** CS 650 sec. 501

**Description:** This course introduces the student to the foundations and state-of-the-art techniques in high performance software development for numeric libraries and other important kernels. Topics include: 1) fundamental tools in algorithm theory, 2) optimizing compilers, 3) effective utilization of the memory hierarchy and other architectural features, 4) how to use special instruction sets, and 5) an introduction to the concepts of self-adaptable software and program generators.

**Course Objectives:** 1) To be able to optimize code for memory hierarchy, SSE, multicore, Cell, and GPU. 2) To be able to implement fast algorithms for various signal transforms. 3) To utilize techniques to automatically implement, optimize, and adapt programs to a variety of computing platforms. 4) To perform code transformations and optimizations. 5) To develop a domain specific language.

**Audience:** Graduate CS or ECE students interested in Programming Languages, Compilers, and High-Performance Computing with applications to Digital Signal Processing algorithms. This course counts as an advanced CS elective and should be taken after CS 550 and CS 540. Students, who are interested in the course but have only taken one of these courses, should contact the instructor to see if an exception is possible. Students should have background in Computer Architecture, Algorithms, Linear Algebra, and Programming Languages. Advanced undergraduates (GPA of 3.0 of better) and prerequisite knowledge (CS 282, CS 360, MATH 201 or ENGR 231) may take the course with instructor permission and count it towards the programming languages and compiler track.

The fast evolution and increasing complexity of computing platforms pose a major challenge for developers of high performance software and scientific/Digital Signal Processing (DSP) libraries. It is increasingly difficult to harness the available computing power; conversely, straightforward implementations may lose as much as one or two orders of magnitude in performance. Creating optimal implementations requires the developer to have an understanding of algorithms, compilers, and the target platform's microarchitecture. For these reasons, a recent trend in numerical computing is towards "self-adaptable" software to achieve optimal performance and portability with reduced coding effort. One approach to self-adapting software is the automatic generation of algorithms and implementations and the use of intelligent search to find the "best" implementation on a given platform.

This course introduces the student to the foundations and state-of-the-art techniques, including automatic tuning and self adaptation, in high performance software development for DSP libraries such as Intel's MKL (Math Kernel Library) and IPP (Integrated Performance Primitives). The course will focus on a domain specific language and code generation system (SPIRAL – www.spiral.net) for fast signal transforms such as the Fast Fourier Transform (FFT), Walsh-
Hadamard Transform (WHT), Discrete Trigonometric Transforms (e.g. Discrete Cosine Transform (DCT)), Convolution and Filtering, and Fast Wavelet Transforms.

**Topics** include:

1. Overview of modern processors, code optimization, optimizing compilers and the SPIRAL system for automatic code generation and optimization
2. SPL – Domain Specific Language and compiler for fast signal transform algorithms
3. Σ-SPL – Extension of SPL to allow loop transformations and optimizations
4. Tagged SPL – extension of SPL to assist in vectorization and parallelization
5. Optimization for SSE, multicore, Cell BE, and GPU platforms
6. Automatic library generation