Research Statement

Motivation
Due to the steady increase of complexity of modern processors, new generations of hardware do not necessarily translate in a proportional increase of the system performance. In fact, the system-performance equation is the result of an intricate relationship between the constituent parts of a processor, and the characteristics of various parts of the application code. What's more, new generations of machines are coming out at shorter and shorter intervals.

To ensure portability of a high-performance code across platforms and multiple generations, algorithms (and in particular libraries) must be designed such that they are easy to use and understand and equally important, that they can effectively exploit the ever changing hardware inner-works. These two requirements are in practice contradictory to some extent and thus difficult to achieve simultaneously. In embedded system, an increasingly dominant and pervasive area of computing, even more stringent requirements (besides absolute performance) are present in the form of real-time deadlines, and power/area constraints on the applications.

General Statement
My work is at the intersection of compilers/architectures, with emphasis on Software.

Indeed, in my thesis I have investigated how to design software that delivers top performance over a variety of (changing) architectures ranging from the high-performance modern single processor (i.e., superscalar) to the newly emerging multi-core parallel machines. As illustrations of my approach, I have derived fast scientific codes that improve on the state-of-the-art performance of several core numerical applications (*e.g.*, Dense Matrix Multiplication, FFT, etc).

In my post-doctorate work (in the department of Electric and Computer Engineering at Carnegie Mellon University), I have applied my expertise and expanded my research into the field of DSP applications and their co-design into custom embedded systems. I have been developing novel technology for the exploration and implementation of embedded solutions for DFT kernels.

My research interests are: embedded systems, and high performance computing, with an emphasis on compilers, architectures, algorithms, and the inter-relationship between these components in maximizing performance/minimizing power consumption. A related interest is linear algebra/DSP kernel implementations.

Specific Expertise

Post-Doc
I have explored two research topics: re-configurability of embedded processors (e.g., XScale) and SW/HW partitioning for DFT kernels using the SPIRAL framework.

> The XScale processor family provides user-controllable independent scaling configuration of CPU, bus, and memory frequencies. This feature introduces another handle for the code optimization with respect to energy consumption or runtime performance. I have quantified the effect of frequency configurations on both performance and energy for three signal processing

transforms: DFT, FIR filters, and WHT. I use SPIRAL to generate different implementations for different frequency configuration, optimized for runtime and energy consumption (physically measured, not simulation). I was able to deliver automatically generated (SPIRAL generated) that achieves competitive results with Intel's vendor (hand-tuned) library routines. Even more importantly, I obtained 20% performance improvements or energy reduction for selected transforms and problem sizes.

HW/SW partitioned implementations promise to offer the best of both worlds—the performance and efficiency of HW and the flexibility of SW. This remains an under-tapped paradigm because of its complexity, the inadequate support in current tools, and the lack of engineers trained in this design discipline. I developed (in SIRAL) the software framework for the co-design and thus fast exploration of SW/HW partitioning of DFT libraries for a VirtexIIPro FPGA. This co-design of DFT kernels achieved a threefold (3x) performance speed-up and a twofold (2x) energy efficiency saving (i.e., two times more operations per Joule) vs. a pure software solution alone for an entire library of DFTs.

Ph.D. Dissertation + Continuing recent work.
In my thesis, I have investigated the design of high-performance algorithms for linear algebra and graph manipulation. In particular, I investigated and implemented application reorganization techniques (e.g., data layout organization, instruction scheduling) and compiler optimizations for the optimal utilization of register files and data caches and also other level of the memory hierarchy. I have also investigated techniques for the analysis of data locality for Divide-and-Conquer algorithms in general, both recursive and non recursive and their utilization to drive novel memory hierarchy adaptations (statically and at run time).

A core technique developed in my dissertation is a static analysis for parameterized loops where I exploit data reuse through the static analysis of cache-line size adaptivity. I present an approach that enables the quantification of data misses with respect to cache-line size at compile-time using (parametric) equations, which model interference. My approach aims at the analysis of perfect loop nests in scientific applications; it is applied to direct mapped caches and is an extension and generalization of the Cache Miss Equation (CME) proposed by Ghosh et al. (1999). Part of this analysis is implemented in a software package, STAMINA. I presented analytical results in comparison with simulation-based methods and I showed evidence of both the expressiveness and the practicality of the analysis. Just as an example of the potential, my approach is capable to investigate and find the optimal cache line size for problems that were intractable using previous techniques.

While at CMU, I have discovered a new algorithm for fast matrix multiplication for Strassen-Winograd's variants that builds on, but also significantly improves performance over the prior state-of-the-art methods such as Atlas and Goto's. Indeed, preliminary results show that my algorithm can improve upon state-of-the-art libraries such as ATLAS (or GotoBLAS), achieving, even for relatively small problems and for a wide variety of architectures, speedups of 20-30%. Furthermore, my algorithm uses a balanced division of the problem and thus reduces the number of computation w.r.t. previous formulations, making the algorithm shorter (line of codes) simpler (easier to understand and maintain).