Linear Transforms: From Math to Efficient Hardware

Extended Abstract

Peter A. Milder, Franz Franchetti, James C. Hoe, and Markus Püschel
Electrical and Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA, U.S.A.
{pam,franzf,jhoe,pueschel}@ece.cmu.edu

Introduction. Linear transforms (such as the discrete Fourier transform) are widely used building blocks in signal processing applications. The domain of linear transforms and their algorithms is well understood mathematically. In this work, we utilize this math-level knowledge to construct a formula driven domain specific high-level synthesis tool. This tool is able to generate efficient register transfer level designs from a mathematical description of the problem. The tool's flexibility allows its generated designs to span a wide cost/performance tradeoff space.

Formula Framework. A linear transform is defined as the matrix-vector multiplication $y = A_n \cdot x$, where x and y are (respectively) the n point input and output vectors, and A_n is an $n \times n$ matrix that defines the transform. Computing a transform by definition requires $O(n^2)$ operations.

So-called "fast" algorithms for linear transforms reduce the cost to $O(n \log n)$ operations. Using concepts from matrix algebra, one can precisely specify a fast algorithm as a *formula*, which describes an algorithm as the product of structured sparse matrices.

For example, the Pease Fast Fourier Transform algorithm is given by the formula

$$\mathrm{DFT}_{r^k} = \left(\prod_{\ell=0}^{k-1} L_r^{r^k} (I_{r^{k-1}} \otimes \mathrm{DFT}_r) T_\ell^{r^k} \right) R_r^{r^k}. \qquad (1)$$

This formula decomposes the discrete Fourier transform of size r^t (DFT_{r^t}) into: permutations (L and R), scaling factors (T), and parallel basic blocks ($I \otimes \mathrm{DFT}_r$). I is the identity matrix, and $\otimes$ is the tensor product of matrices. The iterative product term $\prod$ gives k repetitions of its internal block.

Formulas can be directly implemented as combinational datapaths, but as the transform size increases, a combinational datapath quickly becomes prohibitively expensive.

In order to construct efficient hardware, it is necessary to exploit structure in the computation. Regularity at the formula level can be mapped to datapath reuse in the hardware implementation, where computation blocks are reused sequentially. We support two types of sequential reuse and specify them with formula-level annotation. Thus, one annotated formula specifies both the computation to be performed and the structure of the desired datapath.

Formula-Driven Compilation. We have implemented an automated design flow that compiles a formula (as described above) into a register-transfer level (RTL) description of the datapath. It consists of three stages:

(1.) Formula generation. First, a transform is decomposed into a formula using one or more algorithms. At this point, the formula describes the computation but the datapath design is not yet specified.

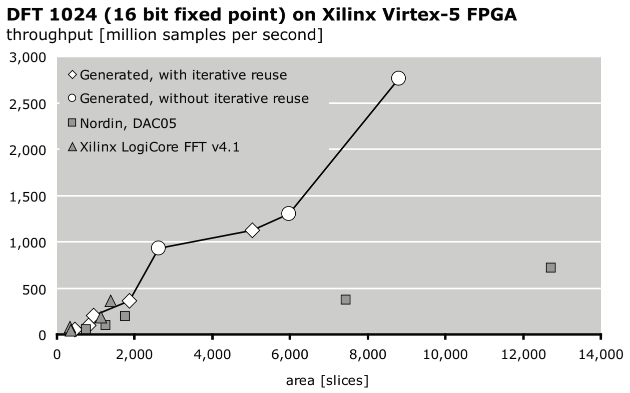

Figure 1: Throughput for varying implementations of $\mathbf{DFT_{1024}}$.

(2.) Formula optimization. Next, the formula produced above is restructured based on simple high-level datapath directives (i.e., parameters that describe the desired sequential reuse characteristics). Rewriting rules (based on mathematical identities) are used to modify the formula to match the given directives. After this rewriting is complete, the datapath's architecture is completely specified.

(3.) RTL generation. Lastly, an RTL generation stage maps the final formula into RTL Verilog. Permutations are constructed with memories and switching networks. Diagonals (e.g., T in (1)) map to multipliers with lookup tables. Basic blocks (e.g., DFT_r in (1)) are built as pipelined computational units. Currently, our RTL generation stage supports single precision floating point and fixed point data of any given width.

Example Results. Figure 1 presents results from an automated design space exploration of various implementations of DFT_{1024} on the Xilinx Virtex-5 FPGA. We compare throughput and area for our generated designs against the Xilinx LogiCore FFT family and our previous work. These results demonstrate that designs generated with our method are competitive with carefully tuned IP cores, but span a much wider cost/performance tradeoff space.

Further Information. For more information, please see our presentation at DAC 2008, where we will discuss research related to the formula representation used in this work. Additionally, a technical report that discusses the compilation framework in more detail is available from the authors.

Acknowledgments. This work was supported by the NSF through awards 0325687 and 0702386 and by DARPA through Department of Interior grant NBCH1050009 and ARO grant W911NF0710416.