Generation of Custom DSP Transform IP Cores: Case Study Walsh-Hadamard Transform -

Fang Fang James C. Hoe Markus Püschel Smarahara Misra
Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

Abstract

Hardware designers are increasingly relying on pre-designed DSP (digital signal processing) cores from IP libraries to improve their productivity and reduce design time. Unfortunately, static DSP cores cannot accommodate application-specific trade-offs. To overcome this problem, we are proposing to automatically generate *customized* DSP cores that can be tailored for specific design requirements. This enables a designer, with no specific background in DSP transform mathematics, to quickly create and evaluate different design choices and to determine the one that is most suitable for the application. In this paper, we present a generator for the Walsh-Hadamard Transform (WHT). The generator accepts as input the WHT's sample size and two implementation parameters that control the degree of hardware reuse. The output is an RTL-level Verilog description of the desired implementation. We present experimental results that compare our different generated designs for the same transform with respect to resource requirements, computation latency, and throughput.

DSP Transform IPs

Creating an optimized hardware implementation of DSP transforms requires combined expertise in both transform mathematics and hardware design to take advantage of the full design space available through algorithmic and architectural choices. Although the field of DSP hardware implementation has been well studied, it remains a highly specialized field with a high entry threshold. Instead of hand-crafting custom implementation of DSP transforms, hardware designers typically use pre-designed static IP blocks from ASIC/FPGA vendors and third-party IP libraries. These IP libraries provide implementations of common DSP transforms of specific sample sizes and data formats. For example, Xilinx's LogiCore library provides implementations of FFT for N=16, 64, 256 and 1024 on 16-bit complex numbers. An obvious shortcoming of these pre-designed IP blocks is the loss of customization. A static IP makes no allowance for achieving higher performance (more concurrent computations) when more hardware resources or power budget is available, nor can a static IP trade excess performance for reduced size or power consumption.

We propose to systematically and automatically generate DSP cores that are optimized with respect to user specified requirements such as performance, size, power, numerical accuracy, and I/O bandwidth. Our approach is based on machine-manipulatable formula representations of fast transform algorithms. By encapsulating the mathematics of DSP transforms in a few well-designed transform factorization rules and algebraic formula manipulations rules, we can develop a systematic approach to explore the algorithmic space of linear DSP transforms. This approach has been successfully applied in the software domain by SPIRAL [1]. Here we present our experience in developing a parameterized generator for WHT.

[1] J. Moura, J. Johnson, R. Johnson, D. Padua, V. Prasanna, M. Püschel, B. Singer, M. Veloso, and J. Xiong. Generating Platform-Adapted DSP Libraries using SPIRAL. In *Proc. HPEC*, 2001. `http://www.ece.cmu.edu/~spiral`.

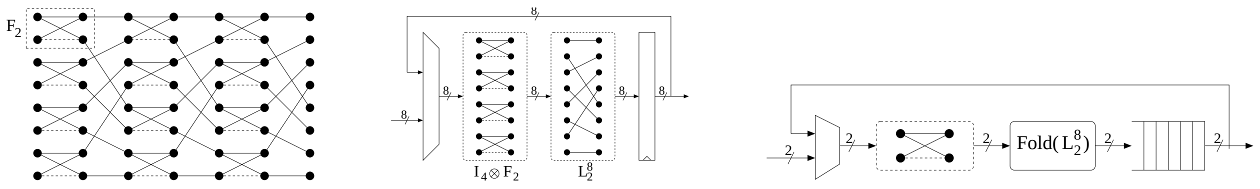

Figure 1: Pease WHT algorithm. From left to right: completely flattened, horizontally folded, fully horizontally and vertically folded. Two lines meeting at a node means addition; a dashed line scales by -1.

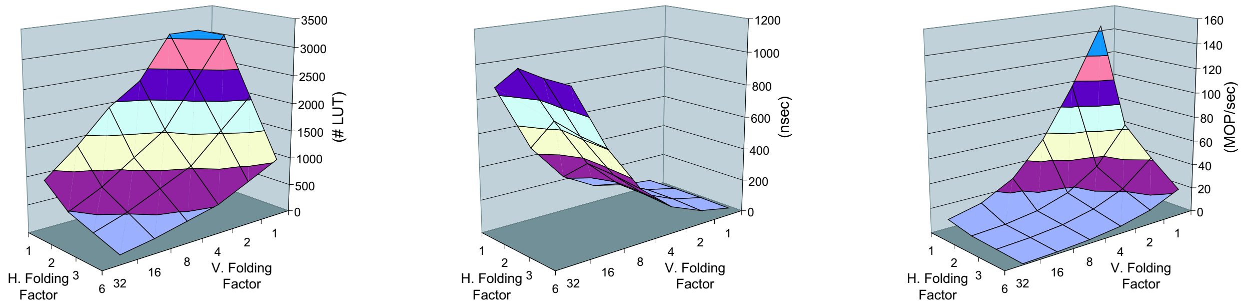

Figure 2: From left to right: area, latency, and throughput variations over the 24 implementations of WHT_{64}.

Generation of Walsh-Hadamard Transform

We chose WHT as our initial case study because this very simple transform has a dataflow that is typical for many DSP transforms (e.g., the discrete Fourier transform). Our generator is based on the Pease algorithm for the WHT, which we express in a formula notation as

$$\mathrm{WHT}_{2^n} = ((\mathrm{I}_{2^{n-1}} \otimes \mathrm{F}_2)\,\mathrm{L}_2^{2^n})^n, \quad \mathrm{F}_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1}$$

where 'I' denotes an identity matrix, '$\otimes$' the Kronecker product of matrices, and 'L' a stride permutation. Figure 1 shows a dataflow representation of (1) for $n = 3$. In general, (1) is a 2^{n-1}-by-n grid of F_2 blocks, which is reflected by the constructs '$\mathrm{I}_{2^{n-1}} \otimes$' and '$(\cdot)^n$' in the formula. The rectangular grid structure exhibits the possibility of *horizontal folding* and *vertical folding* (i.e., reusing the same F_2 hardware instances over multiple columns or rows of the conceptual F_2 grid). Figure 1 shows, for $n = 3$, a block diagram of a horizontally folded WHT (middle) and a horizontally and vertically folded WHT (right). Our WHT generator currently accepts h, v, and n as input parameters, where h and v represent the degrees of horizontal and vertical folding, respectively. For WHT of size 2^n, there are $H \times V$ folding strategies, where H and V are the number of divisors of n and 2^{n-1}, respectively. The main challenge is the design of the folded permutation network for $\mathrm{L}_2^{2^n}$ for arbitrary h, v, n.

For $n = 6$, the WHT core generator produces $4 \times 6 = 24$ implementations representing different trade-offs between the different design goals and constraints. Figure 2 shows how resource usage, latency and throughput varies over these 24 design choices. These results are estimates produced by Synplify after synthesizing the generated Verilog descriptions for Xilinx Virtex-E-8. The figures indicate that resource usage reduces almost linearly with increase in either h or v. The latency is little affected by horizontal folding, but increases with the degree of vertical folding. Both horizontal and vertical folding reduce throughput by increasing the number of times an F_2 block must be reused per computation. In summary, folding reduces the resource requirement but at the same time reduces performance. Our core generation methodology can help hardware designer achieve the optimal trade-off within the design space.