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

Fang Fang
James C. Hoe
Markus Püschel
Smarahara Misra

Carnegie Mellon University
Conventional Approach: Static IP Cores

- IP cores improve productivity and reduce time-to-market.
- e.g. Xilinx LogiCore library:
  - FFT for N=16, 64, 256 and 1024 on 16-bit complex numbers

May not match the application’s needs: parameters, speed, power, area and their trade-off.
Alternative Approach: IP Core Generation

- Generate IP cores to match specific application requirements (speed, area, power, numerical accuracy, and I/O bandwidth...)

Application parameters

Speed / area / power requirements

Generator + Evaluator

Optimized IP cores
DSP transform design can be studied at several levels.

- More math knowledge involved
  - Bigger design space to explore.
Problem

- Problem: gap between *transform mathematics* and *hardware design*

A math guy

What I know:
- Linear algebra
- Digital signal processing
- Adaptive filter theory ...

A hardware engineer

What I know:
- Finite state machine
- Pipelining
- Systolic array ...
Bridge: Formula

- Solution: - Formula representation of DSP transforms
  - Automated formula manipulation and mapping

Formula example: $DFT_8 = (F_2 \otimes I_4) \cdot D \cdot (I_2 \otimes (I_2 \otimes F_2 \cdots)) \cdot P$

A math guy

What I know:
- Linear algebra
- Digital signal processing
- Adaptive filter theory ...

A hardware engineer

What I know:
- Finite state machine
- Pipelining
- Systolic array ...
Outline

- Introduction
- Technical Details (illustrated by WHT transform)
  - What are the degrees of design freedom?
  - How do we explore this design space?
- Experimental Results
- Summary and Future work
Walsh-Hadamard Transform

Why WHT?
- Typical access pattern for a DSP transform
- Close to 2-power FFT
- Study important construct ⊗

Definition

\[
WHT_{2^n} = \begin{bmatrix}
WHT_{2^{n-1}} & WHT_{2^{n-1}} \\
WHT_{2^{n-1}} & -WHT_{2^{n-1}}
\end{bmatrix}
\]

\[
WHT_2 = \begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

\[
WHT_{2^n} = F_2 \otimes F_2 \otimes \ldots \otimes F_2 \quad \text{n fold}
\]

\[
F_2 = \begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

Tensor product \( A \otimes B = [a_{k,l} \cdot B] \), where \( A = [a_{k,l}] \)
From Formula to Architecture

\[ WHT_{2^3} = F_2 \otimes F_2 \otimes F_2 = (F_2 \otimes I_4)(I_2 \otimes (F_2 \otimes I_2))(I_4 \otimes F_2) \]

an \( F_2 \) block

\( WHT_{2^3} \)

Addition
Subtraction
Pease Algorithm

\[ \text{WHT}_{2^3} = (F_2 \otimes I_4)(I_2 \otimes F_2 \otimes I_2)(I_4 \otimes F_2) \]

\[ = L_2^{8} (I_4 \otimes F_2) L_2^{8} (I_4 \otimes F_2) L_2^{8} (I_4 \otimes F_2) \]

Stride permutation \( L_2^N \)
Pease Algorithm

\[ WHT^3 = (F_2 \otimes I_4)(I_2 \otimes F_2 \otimes I_2)(I_4 \otimes F_2) \]
\[ = L_2^8 (I_4 \otimes F_2) L_2^8 (I_4 \otimes F_2) L_2^8 (I_4 \otimes F_2) \]

- Regular routing
- Possibility for vertical folding
- Possibility for horizontal folding
- 12 $F_2$ blocks total
- An $F_2$ block
Folding

Horizontal Folding (HF)

HF + Vertical Folding (VF)

Repeat 3 times

I_4 \otimes F_2

L_2^8

Folded L_2^8

4 F_2 blocks

1 F_2 block
Challenge in Vertical Folding

- **Straightforward approach: Memory-based reordering**
  - Extra control logic to reorder address
  - Computation speed is limited by memory speed

- **Ad-hoc approach: Register routing**
  - Hard to automate the process

- **Our approach: formula-based matrix factorization**
Factorization of Stride Permutation

\[ L_2^N = (I_2 \otimes L_2^{N/2}) \bullet J_N \]
\[ = (I_2 \otimes (I_2 \otimes L_2^{N/4}) J_{N/2}) J_N \]
\[ = \ldots \]
\[ = (I_{N/Q} \otimes L_2^Q) \bullet \prod_{i=0}^{n-q-1} (I_{2^{k+i-1}} \otimes J_{2^{q+i+1}}) \]

\( L_2^Q \) has Q input ports

\( Q = 2^q, N = 2^n \)

\( J_N \) can be easily folded [1]

Example of \( (L_2^{64})_4 \) (N=64, Q = 4)

Freedom in Horizontal Folding

- WHT\(_2^n\) has \(n\) horizontal stages in the flattened design
  - The divisors of \(n\) are all the possible folding degrees
  - Example: HF degrees of WHT\(_2^6\) can be 1, 2, 3, 6

- Effects of more horizontal folding degree

<table>
<thead>
<tr>
<th></th>
<th>Same</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (cycle)</td>
<td></td>
</tr>
<tr>
<td>Throughput (op / cycle)</td>
<td>Lower</td>
</tr>
<tr>
<td>Area</td>
<td>less adders, more muxs &amp; wires</td>
</tr>
<tr>
<td>Speed</td>
<td>Not clear</td>
</tr>
</tbody>
</table>

Less pipeline depth ⇒ lower throughput
Freedom in Vertical Folding

- WHT_{2^n} has 2^n vertical ports in the flattened design
  - 1, 2, 4... 2^{n-1} are all possible folding degrees
  - Example: VF degrees of WHT_2^6 could be 1, 2, 4, ... 32

- Effects of more vertical folding degree

<table>
<thead>
<tr>
<th>Latency (cycle)</th>
<th>Longer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (op / cycle)</td>
<td>Lower</td>
</tr>
<tr>
<td>Area</td>
<td>less adders, more regs &amp; muxs</td>
</tr>
<tr>
<td>Speed</td>
<td>Not clear</td>
</tr>
</tbody>
</table>

Less I/O bandwidth ⇒ longer computation
Outline

- Introduction
- Technical Details
- Experimental Results
- Summary and Future work
Design Space Exploration

- HF factor: (1, 2, 3, 6)
- VF factor: (1, 2, 4, ..., 32) = 24 different designs
- Bit-width (8)
- Transform size (64)

Diagram:
- WHT Generator
- Xilinx FPGA Synthesis
- Xilinx FPGA Place&Route
- Performance requirement
- Evaluator
Area vs. Folding Degrees

To achieve the same area, multiple folding options are available.
Latency vs. Folding Degrees ($WHT_{64}$)
Latency vs. Folding Degrees (WHT_{64})

Latency (ns)

VF degree
Latency vs. Folding Degrees \((WHT_{64})\)

Latency is almost unaffected by HF, except comparing flattened design with folded design.
Throughput vs. Folding Degrees

Folding always lowers throughput
Comparison with an Existing Design

- \( WHT_8 \)
  - 8 bit fixed-point
  - FPGA: Xilinx Virtex xcv1000e-fg680  Speed grade: -8
  - Compare our fastest generated designs against results reported by Amira, et al. [2]

<table>
<thead>
<tr>
<th>Design</th>
<th>Area (#of slices)</th>
<th>Latency(ns)</th>
<th>Throughput(MOP/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design in [2]</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Our fastest</td>
<td>2</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

60% more area
80% reduction in latency
13 times higher throughput

Comparison with an Existing Design

- WHT_8
  - 8 bit fixed-point
  - FPGA: Xilinx Virtex xcv1000e-fg680  Speed grade: -8
  - Compare our **smallest** generated designs against results reported by Amira, et al. [2]

---

Summary

- Large performance variations over the design space of horizontal and vertical folding
- Automatic design space exploration through formula manipulation and mapping can find the best trade-off
Future work

More DSP transform

DFT
DCT
DST
DWT
...

Formula

Representation

Manipulation

Mapping

More design decisions

Pipelining
Systolic array
Distributed Arithmetic
Fix-point vs. Floating-point
...

More design decisions
Thank you!

Contact: Fang Fang
Email: ffang@cmu.edu
URL: www.ece.cmu.edu/~ffang