Wavelet transform based
Adaptive Image Compression on FPGA

Sarin George Mathen
Masters Thesis
M.S. Computer Engineering
University of Kansas

Thesis Committee: Dr. Joseph Evans, Chair
Dr. Gary Minden
Dr. John Gauch
Presentation Outline

- Motivation
- Hardware Platform
- Wavelets, Wavelet transform
- Wavelet transform based image compression
- Design and implementation of the encoder
- Results
- Conclusions & Future Work
Motivation

Why adaptive image compression?

• Application specific compression requirements
  For example, different compression requirements for
  a video conference, a streaming movie ...
• Changing bandwidth availability of the underlying network,
  in case of a real time transmission
  For example, a congested network, peak usage time ...

Key requirements

• Support different levels of compression
• Real time performance in both encoding and switching between codecs
Motivation

**Why reconfigurable hardware (FPGA)?**

- For computationally expensive problems, von-Neumann model is not enough, need a hardware intensive solution/parallel implementation
  
  For example, a 30 tap FIR filter takes 30 cycles for advancing one unit of real time on a DSP microprocessor

**FPGAs are ideal**

- Same piece of silicon re-used for different configurations/codecs
- Real time performance in encoding, about 10 frames/second
- Real time performance in switching between configurations,
  
  108 ms to reconfigure a 4036 -36x36 CLBs (time taken by Wildforce APIs,
  
  290 ms to reconfigure a 4085 -56x56 CLBs actual device timing may be faster)
Hardware platform

Xilinx FPGAs:

• Look up table based FPGAs, organized as a matrix of CLBs
• Programmable elements: LUTs +MUXes in CLBs, routing switches ...
• Challenges: getting a high clock rate
  (routing resource run out fast, use about < 40% of the CLBs,
  avoid many logic levels, use pipelined design, use plenty of flops)
Hardware platform

Wildforce PCI plug-in board:

- 5 Processing Elements (PEs) - Xilinx 4085 FPGAs
- Embedded memory - 1MB attached to each PE
- Other interconnects - FIFO, Crossbar, SIMD ...
- Challenges: getting the best out of embedded memory
  (read/write turn around latency, host/FPGA sequential access)
Wavelets

What are wavelets?
• Localized functions - zero outside a finite interval
• Mean zero

What is a wavelet basis?
• Different basis vectors are formed by dilations and translation of the mother wavelet

What makes a basis desirable?
• Compact support: - localized in space/time
  zero outside a finite interval,
suitable for an FIR implementation
• Vanishing Moments:
  First $p$ moments be zero - localized in frequency
• Smoothness:
  Higher derivatives be zero
• Orthogonality/Bi-orthogonality - perfect reconstruction
Wavelets

Multi-resolution analysis

- Example: consider the sequence of pixels: 10 8 1 3 5 7 8 6
  averages: [9 2 6 7] differences: [-2 2 2 -2]
  averages: [5.5 6.5] differences: [-7 1]

- Each stage divides the band into 2 subbands - low frequency + high frequency coefficients
- Regions of discontinuities will have large coefficients, smooth regions will have smaller differences
- Error introduced by truncating a coefficient is proportional to its magnitude, can truncate small coefficients without considerable distortion
Wavelets

Lifting scheme

\[
\begin{align*}
s & \leftarrow (a+b)/2 \\
d & \leftarrow (b-a) \\
a & \leftarrow s \\
b & \leftarrow d \\
a & \leftarrow (a+b/2)
\end{align*}
\]

Wavelets that map integers to integers

- Though rounding/truncation introduces a non linearity, it is perfectly invertible if rounding is deterministic

\((2,2)\) Cohen Daubechies Feauveau wavelet

\[
\begin{align*}
s_i & \leftarrow x_{2i} \\
d_i & \leftarrow x_{2i+1} \\
d_i & \leftarrow d_i - (s_i + s_{i+1})/2 \\
s_i & \leftarrow s_i - (d_{i-1} + d_i)/4
\end{align*}
\]

\[
\begin{align*}
s_i & \leftarrow s_i - d_i/2 \\
d_i & \leftarrow d_i + s_i \\
x_{2i} & \leftarrow s_i \\
x_{2i+1} & \leftarrow d_i
\end{align*}
\]

Lifting allows an in-place computation of coefficients - split, predict, and update
Wavelets

Wavelet transform based image compression

• DWT coefficients of input image - multiple levels of wave-letting
• Coefficients are quantized - coefficients in each subband is quantized separately
• Coefficients are zero thresholded, different subbands have different thresholds
• Longs spells of zero are run length encoded
• The coefficients are then entropy encoded

Achieving different compression ratios

• Use different sets of zero thresholds
Design and Implementation

Design Specs

• Input image: 512x512 pixel, gray scale frame, 8 bits/pixel
• Support 3 different configurations of encoder with varying levels of compression

Design Partition - 2 stages

• **Stage 1**: DWT coefficients over 3 stages of wave-letting
• **Stage 2**: Dynamic Quantization,
  Zero thresholding,
  RLE of zeroes and,
  Entropy encoding of DWT coefficients
• 2 stages are implemented on 2 separate PEs
Design and Implementation

Stage 1 - DWT coefficients

- **Input:** 8 bit pixels, **Output:** DWT coefficients - 16 bits
- 2 pixels/WORD, 512 Rows and 256 Columns, 0.5 MB
- From 512 pixels in a row, extract 256 low frequency coefficients + 256 high frequency coefficients
- Symmetric extension at the boundaries

![Diagram of Mallot ordering]

- Only \( f \) coefficients are used in next stage, write back 2 consecutive \( f \)s to one WORD, next stage reads only alternate WORDs, we save on memory READs

```
0 1 2 3 4 5 *** 508 509 510 511
```

```
0 f0 f1 f2 f3 f4 f5 f6 f7 *** f254 f255
```

- **Mallot ordering:** Write back 256 \( f \)s followed by 256 \( g \)s - needs extra temporary storage
Design and Implementation

Stage 1 - DWT coefficients

- Extend same scheme of interleaved memory access along Y direction
- But now the 2 values obtained in a READ are not consecutive pixel values of a column rather they are one pixel each of two parallel columns

3 stages of Wave-letting
- Stage 1 - On rows and columns of length 512
- Stage 2 - On rows and columns of length 256
- Stage 3 - On rows and columns of length 128

<table>
<thead>
<tr>
<th></th>
<th>a0</th>
<th>b0</th>
<th>f0_a</th>
<th>f0_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>b1</th>
<th>g0_a</th>
<th>g0_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a2</th>
<th>b2</th>
<th>f1_a</th>
<th>f1_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a510</th>
<th>b510</th>
<th>f255_a</th>
<th>f255_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a511</th>
<th>b511</th>
<th>g255_a</th>
<th>g255_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design and Implementation

Stage 2 - Dynamic Quantizer

- Coefficients from each block is quantized separately
  - different dynamic ranges
- Dynamic range divided into 16 levels
- Quantizer is implemented as a binary search tree look-up
- Pipelined design with a latency of 5 cycles.
**Design and Implementation**

**Stage 2 - Zero thresholding and RLE for zeroes**

- Coefficients with magnitude lesser than the threshold are truncated to ZERO
- Separate thresholds are used for each subband
- After thresholding, many coefficients are truncated to zero, long runs of zeroes are replaced by their count
- Maximum countable run is 240 - longer runs are broken up

\[\text{RLERunning} \text{ is asserted when in the middle of a run}\]
\[\text{RLEspellend} \text{ is asserted when a run ends - read the count now}\]
\[\text{Flush} \text{ is used to disable counting across subbands}\]

- Quantizer output (16): 00000000 to 00001111
- RLE output (240): 00011111 to 11111111
- Clk
- Reset
- Enable
- Flush
- RLE output (8)
- RLErunning (1)
- RLEspellend (1)
Design and Implementation

Stage 2 - Entropy Encoding

- Variable length encoding of 8 bit inputs to 3 to 18 bits output
- Implemented with 2 look-up tables on FPGA

![Diagram of LUTs for encoding](image)

- LUT for encoding length
- LUT for variable length bit encoding

![Bit allocation graph](image)
Design and Implementation

Stage 2 - Bit packing

Binary Shifter
- 5 register stages, each 32 bits wide
- Can shift by 16, 8, 4, 2, 1 at stages 1, 2, 3, 4, 5 respectively
- Key decision is whether to shift or not at each stage
- Last stage is partly double buffered
- OutputEn signals when 32 bits are available at the output
Design and Implementation

Stage 2 - Over all architecture

- **Read 001**: fire a READ
- **Read 010**: waiting for READ results, do a WRITE if needed
- **Read 100**: read result arrives
- **Write**: do a WRITE if needed
# Results

## Throughput measurements

<table>
<thead>
<tr>
<th>Action</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory write from host 0.5 MB</td>
<td>4.017</td>
</tr>
<tr>
<td>PE1 running time @ 24MHz</td>
<td>57.402</td>
</tr>
<tr>
<td>Memory read from host 1.0 MB</td>
<td>8.401</td>
</tr>
<tr>
<td>Memory write from host 1.0 MB</td>
<td>7.948</td>
</tr>
<tr>
<td>PE2 running time @ 24MHz</td>
<td>5.510</td>
</tr>
<tr>
<td>Memory read from host 1.0 MB</td>
<td>8.394</td>
</tr>
</tbody>
</table>

Various delays along a single thread of execution
Results

PSNR and RMS computation

\[ MSE = \frac{1}{512 \times 512} \sum_{i=1}^{512} \sum_{j=1}^{512} [p(i, j) - p'(i, j)]^2 \]

\[ RMSE = \sqrt{MSE} \]

\[ PSNR = 20 \log_{10}(\frac{255}{RMSE}) \]
Results

Test images used and reconstructed images

Barbara - Original Image

Minimum compression  Medium compression  Maximum compression
Results

Test images used and reconstructed images

Goldhill- Original Image

Minimum compression  Medium compression  Maximum compression
## Results

### Compression ratio, bpp, RMS, and PSNR

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Lena</th>
<th>Barbara</th>
<th>Goldhill</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Comp ratio</td>
<td>bpp</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>Config. 1 Minimum compression</td>
<td>9.11</td>
<td>0.878</td>
<td>30.783 (31.102)</td>
</tr>
<tr>
<td>Config. 2 Medium compression</td>
<td>47.18</td>
<td>0.169</td>
<td>29.630 (30.015)</td>
</tr>
<tr>
<td>Config 3 Maximum compression</td>
<td>69.58</td>
<td>0.114</td>
<td>28.040 (28.120)</td>
</tr>
</tbody>
</table>
## Results

### Implementation Costs - Device Utilization

<table>
<thead>
<tr>
<th>Block</th>
<th>LUTS (4)</th>
<th>LUTS (3)</th>
<th>CLB flops</th>
<th>Total CLBs</th>
<th>I/O Bufs</th>
<th>I/O flops</th>
<th>Gate count</th>
<th>Timing (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td>547</td>
<td>109</td>
<td>406</td>
<td>399 (12%)</td>
<td>75</td>
<td>88</td>
<td>8244</td>
<td>26.553</td>
</tr>
<tr>
<td>Stage 2 Conf. 1</td>
<td>1248</td>
<td>356</td>
<td>924</td>
<td>890 (28%)</td>
<td>77</td>
<td>88</td>
<td>17058</td>
<td>36.381</td>
</tr>
<tr>
<td>Stage 2 Conf. 2</td>
<td>1297</td>
<td>367</td>
<td>975</td>
<td>948 (30%)</td>
<td>77</td>
<td>88</td>
<td>17937</td>
<td>31.254</td>
</tr>
<tr>
<td>Stage 3 Conf. 3</td>
<td>1297</td>
<td>373</td>
<td>965</td>
<td>925 (29%)</td>
<td>77</td>
<td>88</td>
<td>17830</td>
<td>34.632</td>
</tr>
</tbody>
</table>
Conclusions and Future work

Conclusions
• We have achieved a throughput of 10 frames/second (frames = 512x512)
• The encoder supports 3 different configurations offering different compression levels at the expense of precision
• The same piece of silicon is reconfigured for different encoders

Future work
• Implement a similar decoder in FPGA, demonstrate the adaptability of the encoder-decoder pair
• Use Crossbar/SIMD for transferring data from PE1 to PE2
• Investigate other methods to achieve variable compression levels - changing number of wave-letting stages