Wavelet transform based Adaptive Image Compression on FPGA

Sarin George Mathen

Masters Thesis
M.S. Computer Engineering
University of Kansas
June 23, 2000.

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 <u>encoding</u> and <u>switching</u> between codecs

Motivation

Why reconfigurable hardware (FPGA) ?

- For computationally expensive problems, Von-nueman 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,

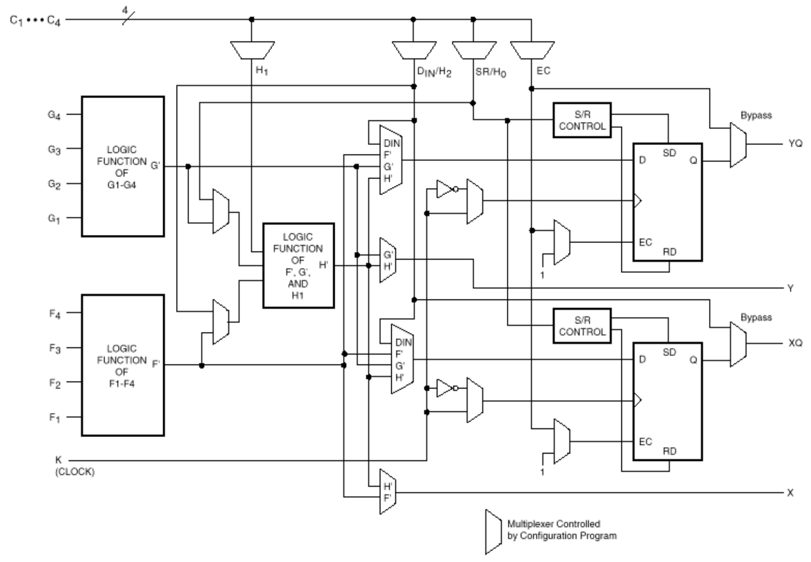
  108 *ms* to reconfigure a 4036 -36x36 CLBs (time taken by *Wildforce* APIs,
  290 *ms* to reconfigure a 4085 -56x56 CLBs actual device timing may be faster)

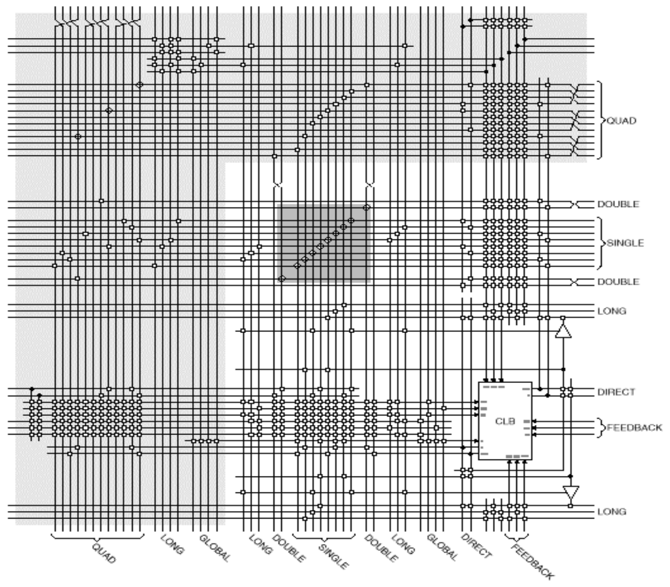
Hardware platform

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)



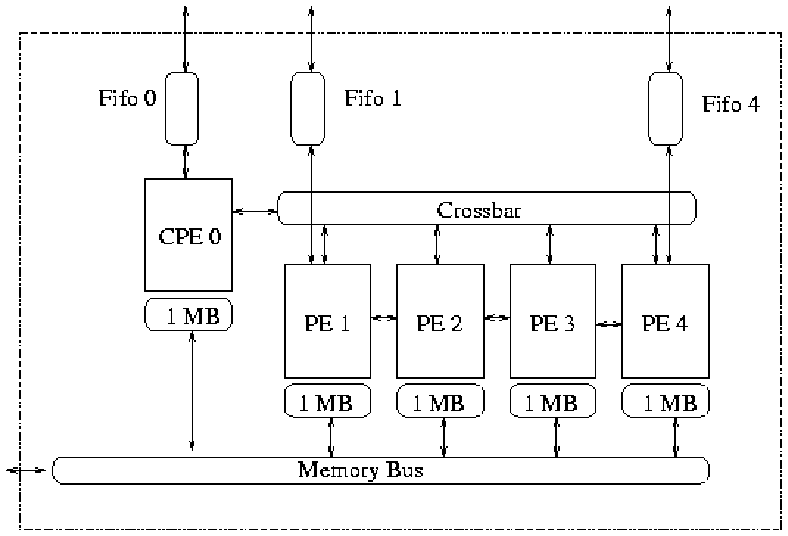


Hardware platform

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)



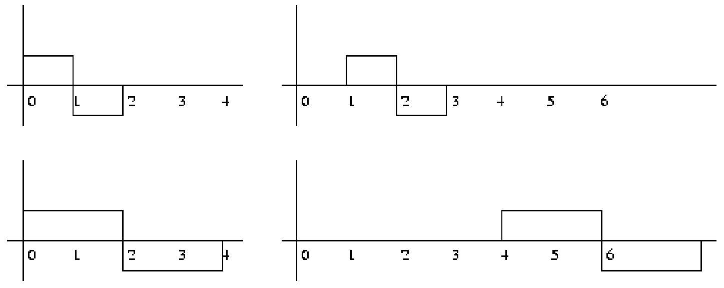
Wavelets

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



Haar Wavelet

What makes a basis desirable?

- Compact support: - *<u>localized in space/time</u>*
  zero outside a finite interval,
  suitable for an FIR implementation
- Vanishing Moments:
  First *p* moments be zero
- Smoothness:
  Higher derivatives be zero

$\left.\right]$ *<u>localized in frequency</u>*

•Orthogonality/Bi-orthogonality - *<u>perfect reconstruction</u>*

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]

averages:[6] differences: [1]

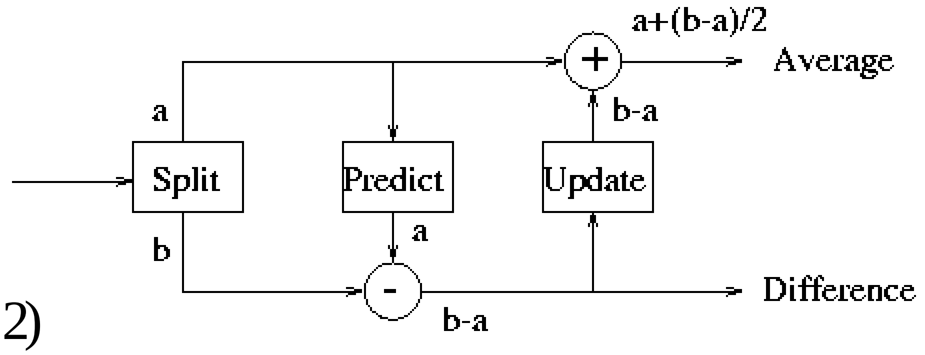
• Each stage divides the band into 2 subbands - low frequency + high frequency coefficients
• Regions of discontinuities will have large coefficients, smooth regions will have smaller differences
• Error introduced by truncating a coefficient is proportional to its magnitude, can truncate small coefficients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

Wavelets

Lifting scheme

$$s \leftarrow (a+b)/2$$
$$d \leftarrow (b-a)$$
$$a \leftarrow s$$
$$b \leftarrow d$$

$$b \leftarrow (b-a)$$
$$a \leftarrow (a+b/2)$$



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

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

$$s_i \leftarrow x_{2i}$$
$$d_i \leftarrow x_{2i+1}$$
$$d_i \leftarrow d_i - (s_i + s_{i+1})/2$$
$$s_i \leftarrow s_i - (d_{i-1} + d_i)/4$$

$$s_i \leftarrow s_i - d_i/2$$
$$d_i \leftarrow d_i + s_i$$
$$x_{2i} \leftarrow s_i$$
$$x_{2i+1} \leftarrow d_i$$

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

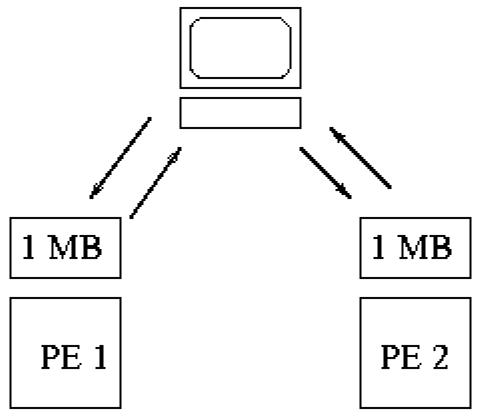
Design and Implementation

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

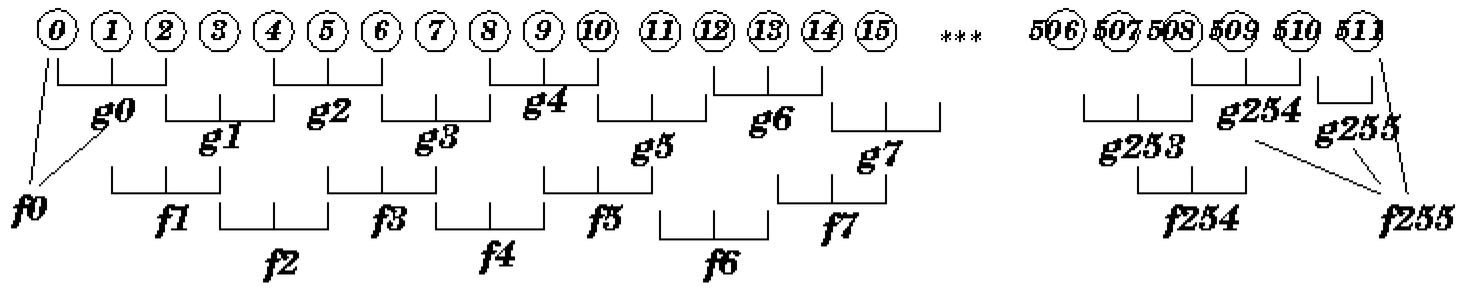
- Stage 1: DWT coefficients over 3 stages of wave-letting
- Stage 2: Dynamic Quantization,
  Zero thresholding,
  RLE of zeroes and,
  Entropy encoding of DWT coefficients
- 2 stages are implemented on 2 separate PEs



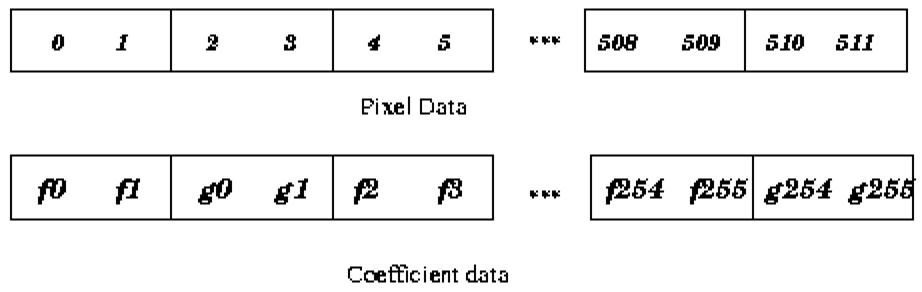
Design and Implementation

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


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 *** 506 507 508 509 510 511
g0 g1 g2 g3 g4 g5 g6 g7 g253 g254 g255
f0 f1 f2 f3 f4 f5 f6 f7 f254 f255

- 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



- **Mallot ordering:** Write back 256 ***f***s followed by 256 ***g***s - needs extra temporary storage

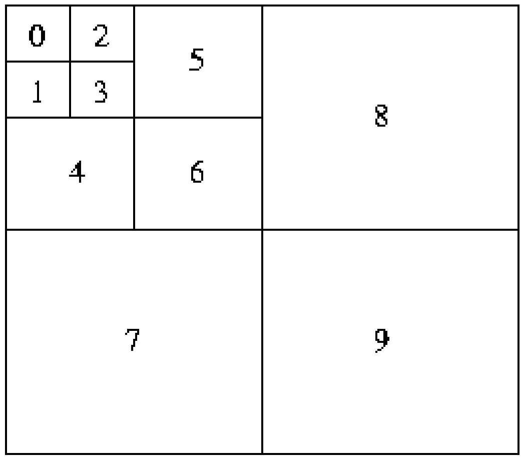
Design and Implementation

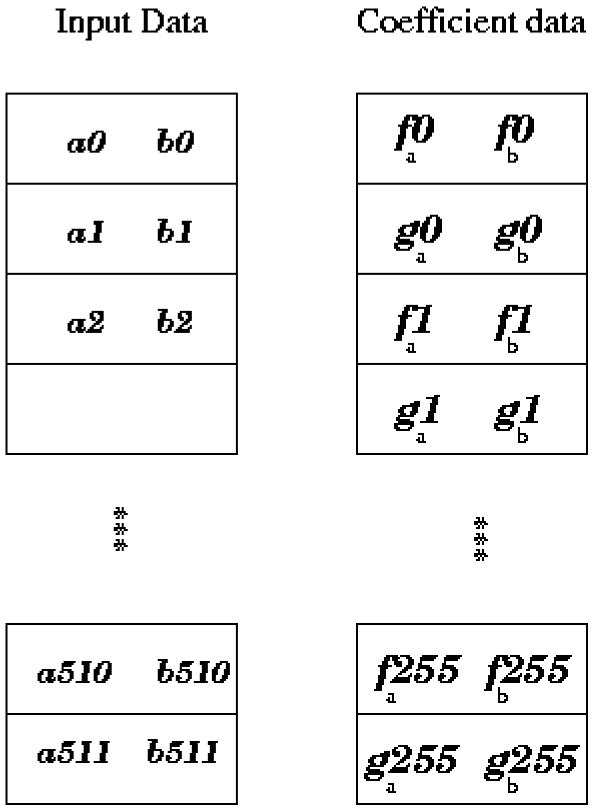
Stage 1 - DWT coefficients

- 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

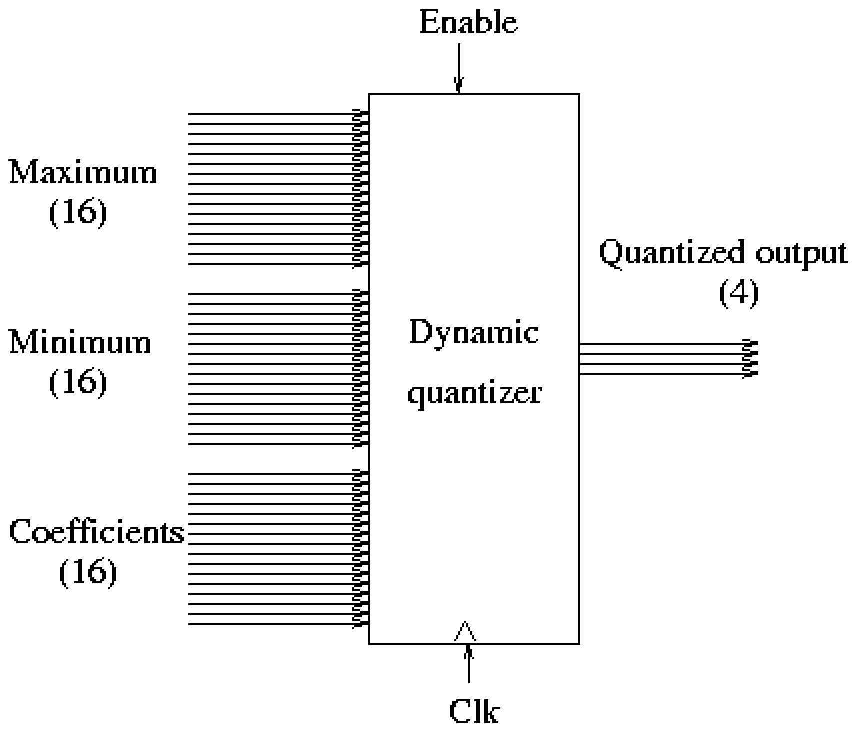




Design and Implementation

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.

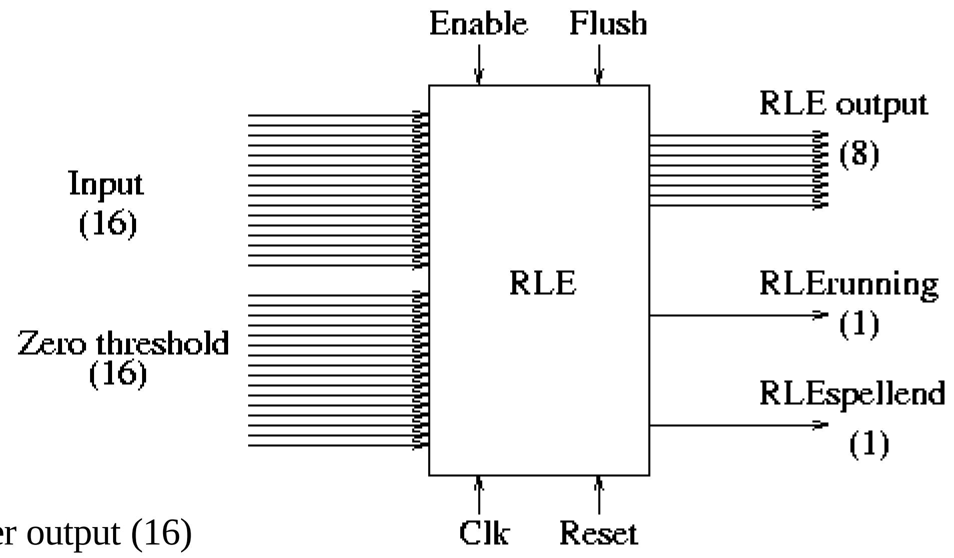


Design and Implementation

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

- *RLErunning* is asserted when in the middle of a run
- *RLEspellend* is asserted when a run ends - read the count now
- *Flush* is used to disable counting across subbands

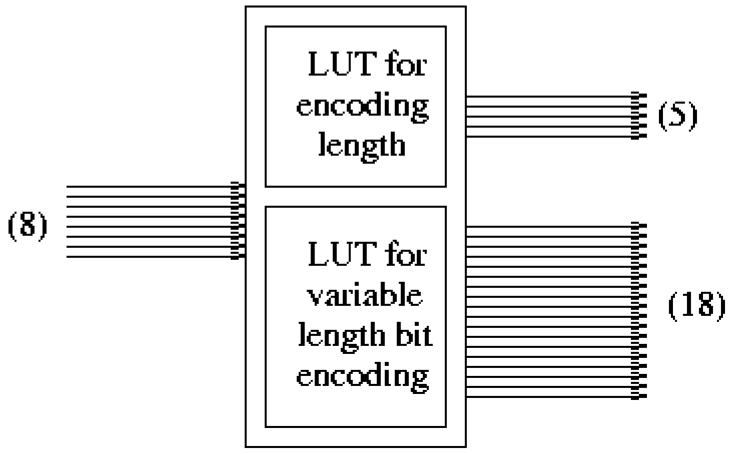


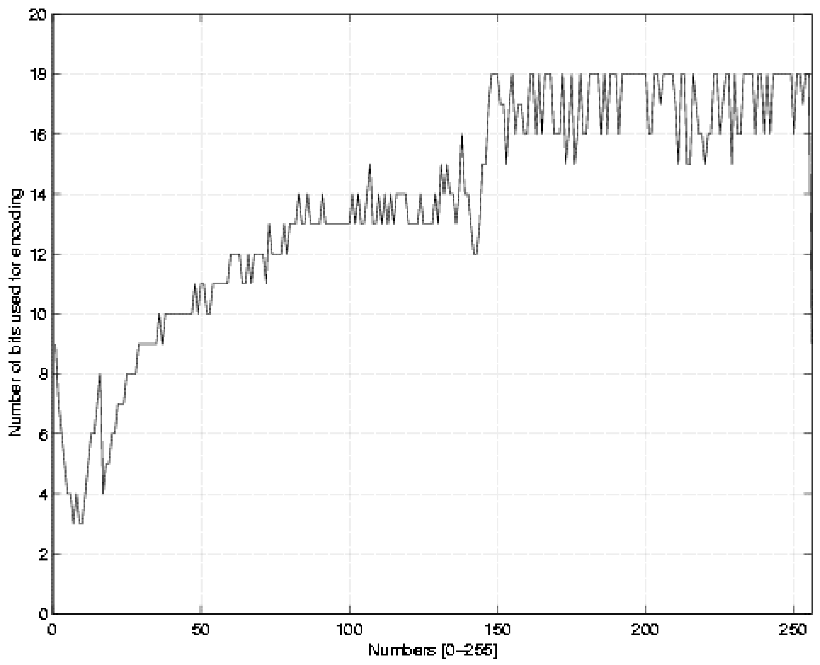
00000000	to	00001111	Quantizer output (16)
00011111	to	11111111	RLE output (240)

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

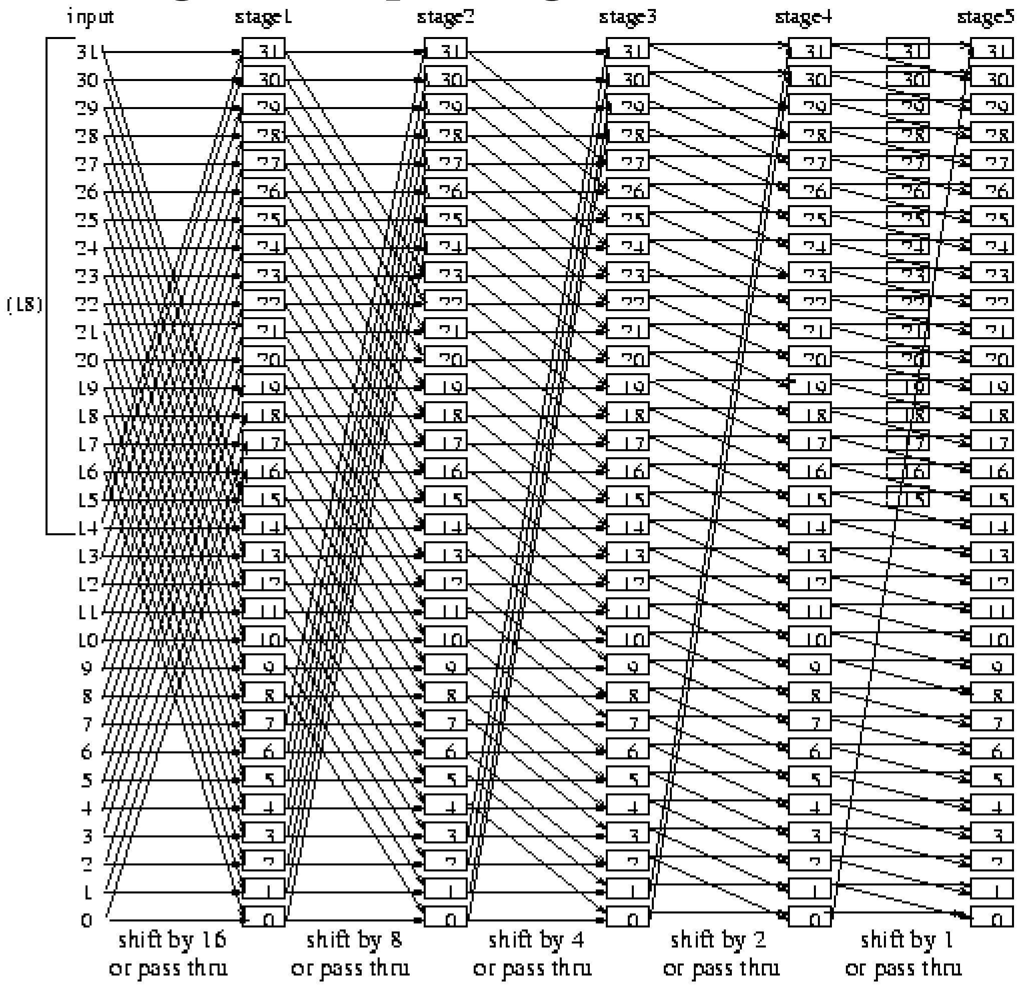




Bit allocation

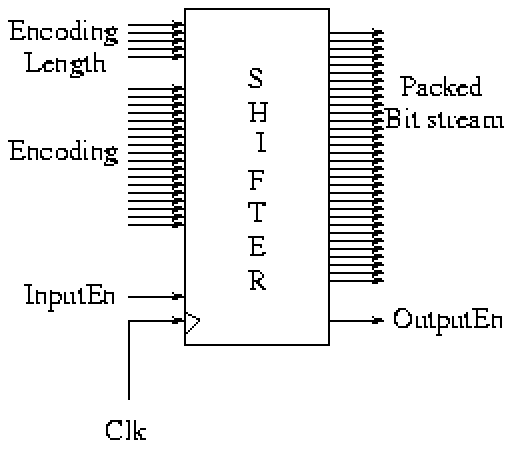
Design and Implementation

Stage 2 - Bit packing



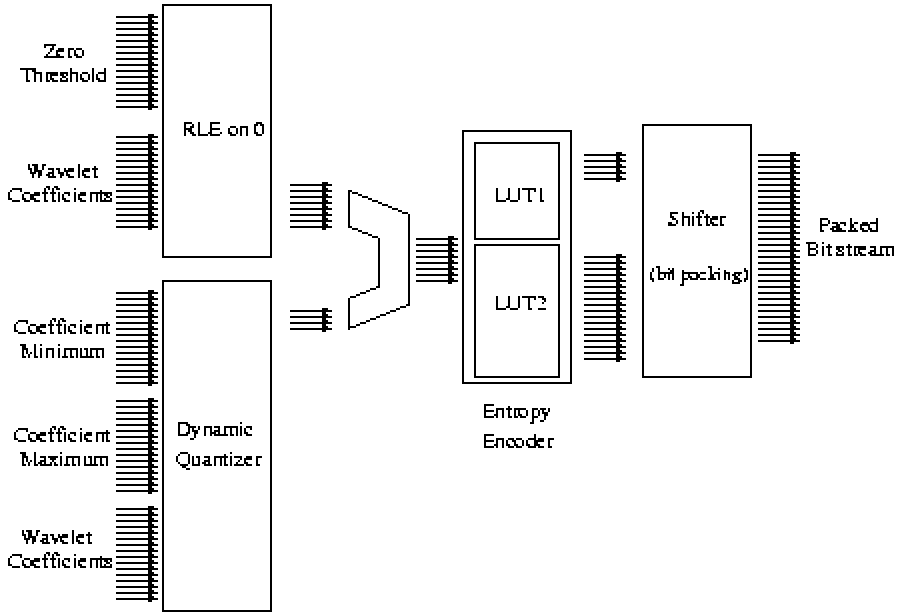
Binary Shifter

- 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

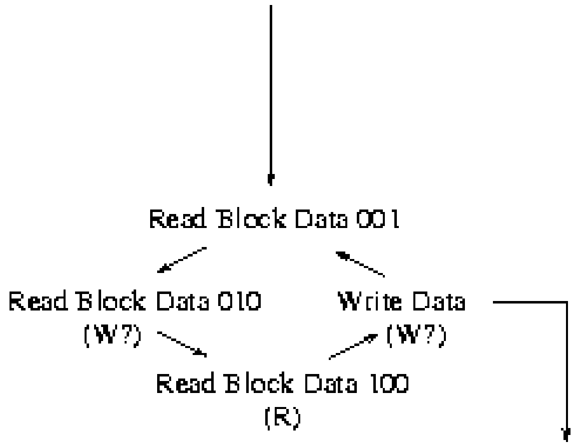


Design and Implementation

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

Memory write from host 0.5 MB	4.017 ms
PE1 running time @ 24MHz	57.402 ms
Memory read from host 1.0 MB	8.401 ms
Memory write from host 1.0 MB	7.948 ms
PE2 running time @ 24MHz	5.510 ms
Memory read from host 1.0 MB	8.394 ms

Various delays along a single thread of execution

Lena- Original Image

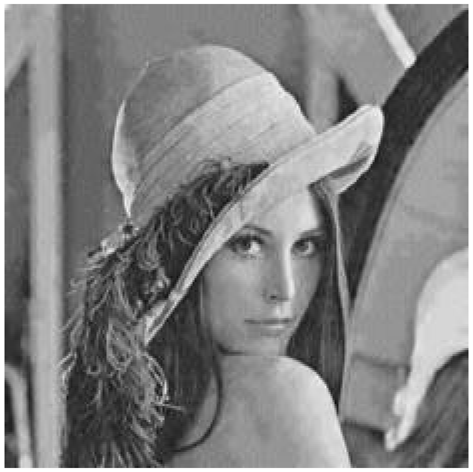
Results

PSNR and RMS computation

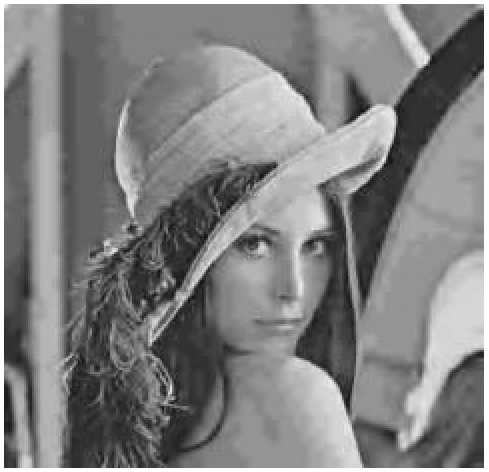
$$MSE = \frac{1}{512X512}\sum_{i=1}^{i=512}\sum_{j=1}^{j=512}[p(i,j) - p'(i,j)]2$$

$$RMSE = \sqrt{MSE}$$

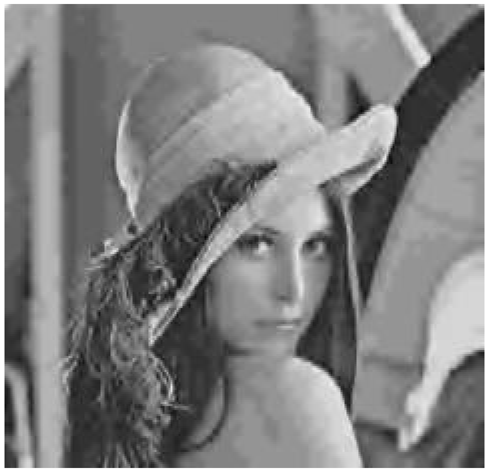
$$PSNR = 20\log_{10}(255/RMSE)$$



Minimum compression



Medium compression

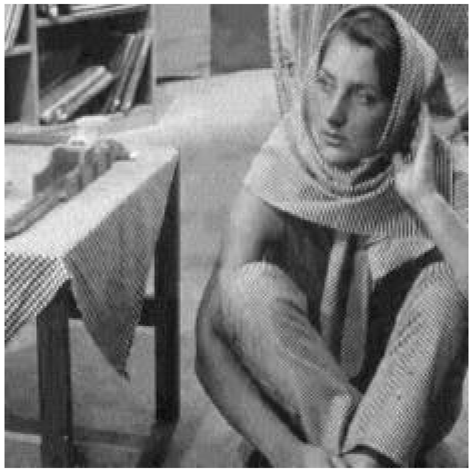


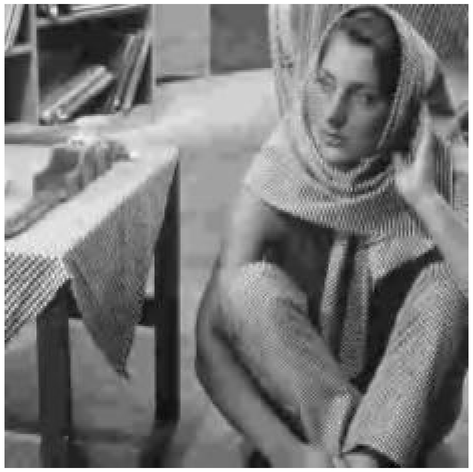
Maximum compression

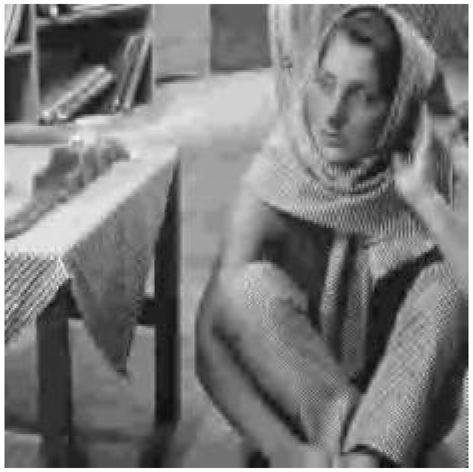
Results

Barbara - Original Image

Test images used and reconstructed images


Minimum compression

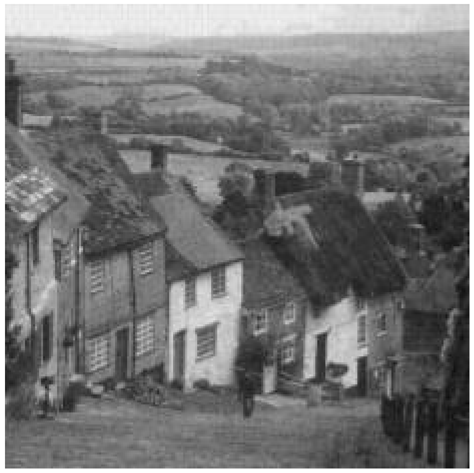

Medium compression


Maximum compression

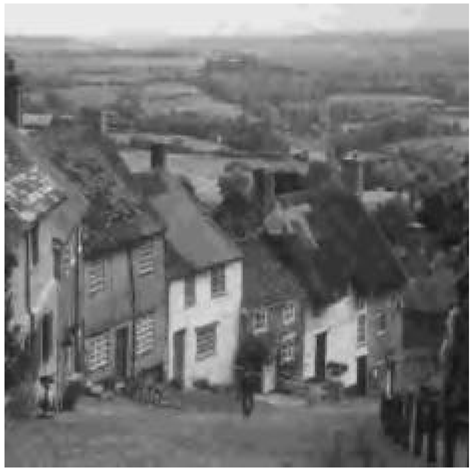
Goldhill- Original Image

Results

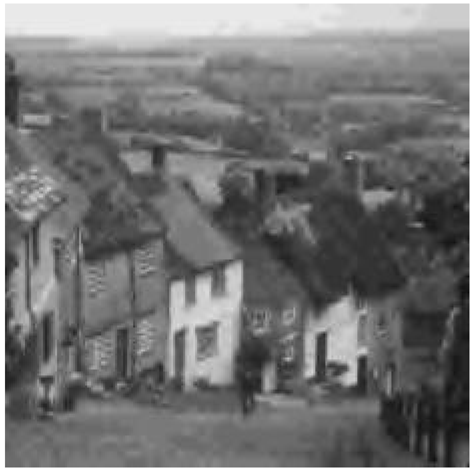
Test images used and reconstructed images



Minimum compression



Medium compression



Maximum compression

Results

Compression ratio, bpp, RMS, and PSNR

	LENA				BARBARA				GOLDHILL			
Configuration	Comp ratio	bpp	PSNR (dB)	RMS	Comp ratio	bpp	PSNR (dB)	RMS	Comp ratio	Bpp	PSNR (dB)	RMS
Config. 1 Minimum compression	9.11	0.878	30.783 (31.102)	7.368 (7.102)	8.82	0.906	24.891 (25.024)	14.520 (14.299)	8.7	0.916	29.733 (30.045)	8.314 (8.021)
Config. 2 Medium compression	47.18	0.169	29.630 (30.015)	8.414 (8.049)	32.01	0.249	24.412 (24.589)	15.343 (15.033)	43.18	0.185	28.001 (28.014)	10.150 (10.134)
Config 3 Maximum compression	69.58	0.114	28.040 (28.120)	10.104 (10.012)	53.33	0.149	23.525 (23.556)	16.992 (16.932)	72.09	0.110	26.355 (26.446)	12.267 (12.140)

Results

Implementation Costs - Device Utilization

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

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