MPEG decoding in FPGAs

Why compress?

- Typical HDTV standard: 1920 x 1080 at 30 frames/sec
- 8 bits for each of three primary colors
- 1.5 Gb/sec!
- Bandwidth limitation => 18 Mb/sec per channel for video
- Require 83:1 compression
- With high quality video to end user

MPEG format

- Transmission in form of series of frames
- Use YCbCr color-space instead of RGB
- Each frame comprises a set of macroblocks
  - Macroblocks comprise set of blocks
  - e.g. formats 4:4:4 4:2:2 4:2:0

MPEG format

- High correlation between neighboring pixels
- Use Discrete Cosine Transform
  - Near optimal for energy concentration and decorrelation
  - Transform 8x8 block to produce 8x8 DCT coeffs
  - Most higher-order coeffs are 0
  - Drastically reduces bandwidth requirement

Discrete Cosine Transform

22	19	13	13	15
18	13	7	6	9
14	11	5	5	7
12	8	1	2	3
12	14	9	3	5
10	12	2	2	5

8	4	2	0	0
4	2	0	0	0
2	0	0	0	0
0	0	0	0	0
0	1	0	0	0
0	0	0	0	0

Discrete Cosine Transform

$$F(\mu,\nu) =$$

$$\frac{1}{4}C(\mu)C(\nu)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)\mu\pi}{16}\right]\cos\left[\frac{(2y+1)\nu\pi}{16}\right]$$

$$C(\mu) = \frac{1}{\sqrt{2}} \ for\ \mu = 0 \qquad C(\mu) = 1\ for\ \mu = 1,2\ldots7$$

$$f(x,y) =$$

$$\frac{1}{4}\sum_{\mu=0}^{7}\sum_{\nu=0}^{7} C(\mu)C(\nu)F(\mu,\nu)\cos\left[\frac{(2x+1)\mu\pi}{16}\right]\cos\left[\frac{(2y+1)\nu\pi}{16}\right]$$

Discrete Cosine Transform

- Implementing 2D DCT with two 1D DCTs

$$f(x,y) =$$

$$\frac{1}{4}\sum_{\mu=0}^{7}\sum_{\nu=0}^{7} C(\mu)C(\nu)F(\mu,\nu)\cos\left[\frac{(2x+1)\mu\pi}{16}\right]\cos\left[\frac{(2y+1)\nu\pi}{16}\right]$$

$$G(\mu,y) = \frac{1}{2}\sum_{\nu=0}^{7} C(\nu)F(\mu,\nu)\cos\left[\frac{(2y+1)\nu\pi}{16}\right]$$

$$f(x,y) = \frac{1}{2}\sum_{\mu=0}^{7} C(\mu)G(\mu,y)\cos\left[\frac{(2x+1)\mu\pi}{16}\right]$$

1D-DCT using DA

$$G(\mu, y) = \frac{1}{2} \sum_{v=0}^{7} C(v) F(\mu, v) \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Simplified form

$$G(y) = \sum_{v=0}^{7} F(v) \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

1D-DCT using DA

$$G(y)=\sum_{v=0}^{7}F(v)\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

F(0) F(1)

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	→ G(0)
C_0	C_3	C_6	C_9	C_{12}	C_{15}	C_{18}	C_{21}	
C_0	C_5	C_{10}	C_{15}	C_{20}	C_{25}	C_{30}	C_{35}	
C_0	C_7	C_{14}	C_{21}	C_{28}	C_{35}	C_{42}	C_{49}	
C_0	C_9	C_{18}	C_{27}	C_{36}	C_{45}	C_{54}	C_{63}	
C_0	C_{11}	C_{22}	C_{33}	C_{44}	C_{55}	C_{66}	C_{77}	
C_0	C_{13}	C_{26}	C_{39}	C_{52}	C_{65}	C_{78}	C_{91}	
C_0	C_{15}	C_{30}	C_{45}	C_{60}	C_{75}	C_{90}	C_{105}	→ G(7)

$$C_k=\cos\left(\frac{k\pi}{16}\right)$$

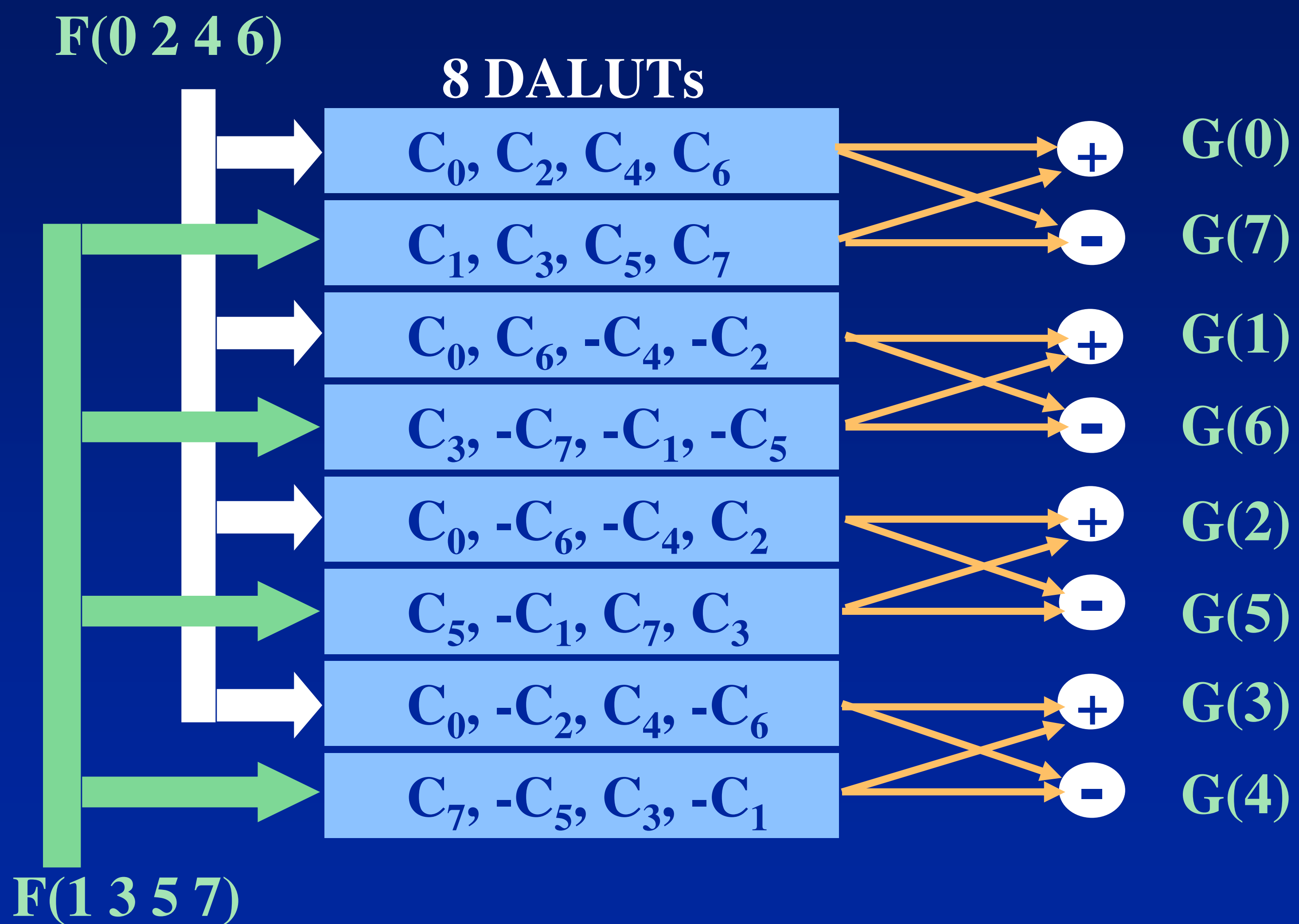
1D-DCT : simplified coeffs

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_0	C_3	C_6	$-C_7$	$-C_4$	$-C_1$	$-C_2$	$-C_5$
C_0	C_5	$-C_6$	$-C_1$	$-C_4$	C_7	C_2	C_3
C_0	C_7	$-C_2$	$-C_5$	C_4	C_3	$-C_6$	$-C_1$
C_0	$-C_7$	$-C_2$	C_5	C_4	$-C_3$	$-C_6$	C_1
C_0	$-C_5$	$-C_6$	C_1	$-C_4$	$-C_7$	C_2	$-C_3$
C_0	$-C_3$	C_6	C_7	$-C_4$	C_1	$-C_2$	C_5
C_0	$-C_1$	C_2	$-C_3$	C_4	$-C_5$	C_6	$-C_7$

1D-DCT : simplified coeffs

F(0)							F(7)	
C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	G(0)
C_0	C_3	C_6	$-C_7$	$-C_4$	$-C_1$	$-C_2$	$-C_5$	
C_0	C_5	$-C_6$	$-C_1$	$-C_4$	C_7	C_2	C_3	
C_0	C_7	$-C_2$	$-C_5$	C_4	C_3	$-C_6$	$-C_1$	
C_0	$-C_7$	$-C_2$	C_5	C_4	$-C_3$	$-C_6$	C_1	
C_0	$-C_5$	$-C_6$	C_1	$-C_4$	$-C_7$	C_2	$-C_3$	
C_0	$-C_3$	C_6	C_7	$-C_4$	C_1	$-C_2$	C_5	
C_0	$-C_1$	C_2	$-C_3$	C_4	$-C_5$	C_6	$-C_7$	G(7)

Block diagram for 1D-DCT



XILINX

Original Picture block

22	19	13	13	15
18	13	7	6	9
14	11	5	5	7
12	8	1	2	3
12	14	9	3	5
10	12	2	2	5

DC-transformed block

8	4	2	0	0
4	2	0	0	0
2	0	0	0	0
0	0	0	0	0
0	1	0	0	0
0	0	0	0	0

8 4 2 0 0 4 2 0 0 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

Zigzag scan ordering

8	4	2	0	0
4	2	0	0	0
2	0	0	0	0
0	0	0	0	0
0	1	0	0	0
0	0	0	0	0

8 4 4 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

Variable-length code table

Zero run-length	Amplitude	MPEG code value
N/A	8	110 1000
0	4	0000 1100
0	4	0000 1100
0	2	0100 0
0	2	0100 0
0	2	0100 0
12	1	0010 0010 0
EOB	EOB	10

Huffman codes

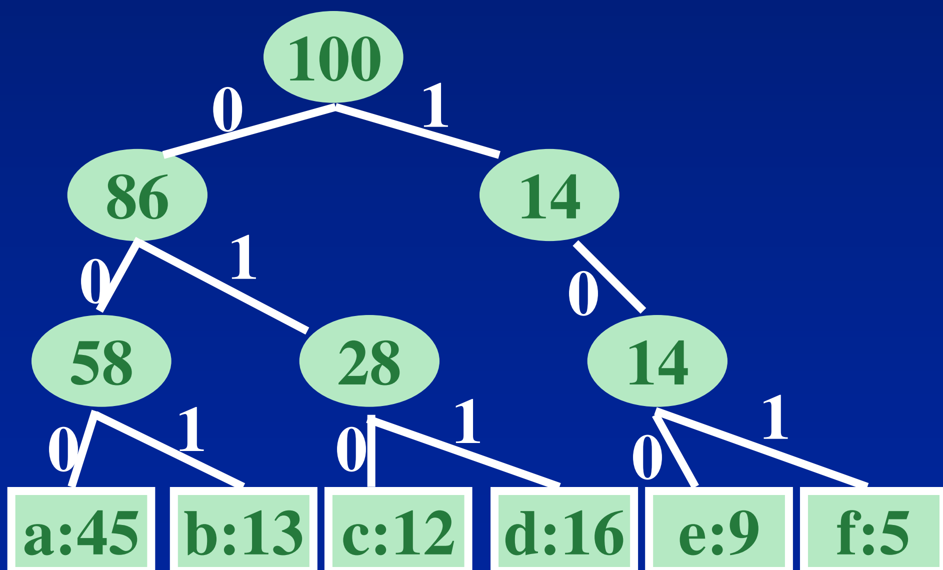
- Typically 20% - 90% compression
- Coding based on frequency of occurrence

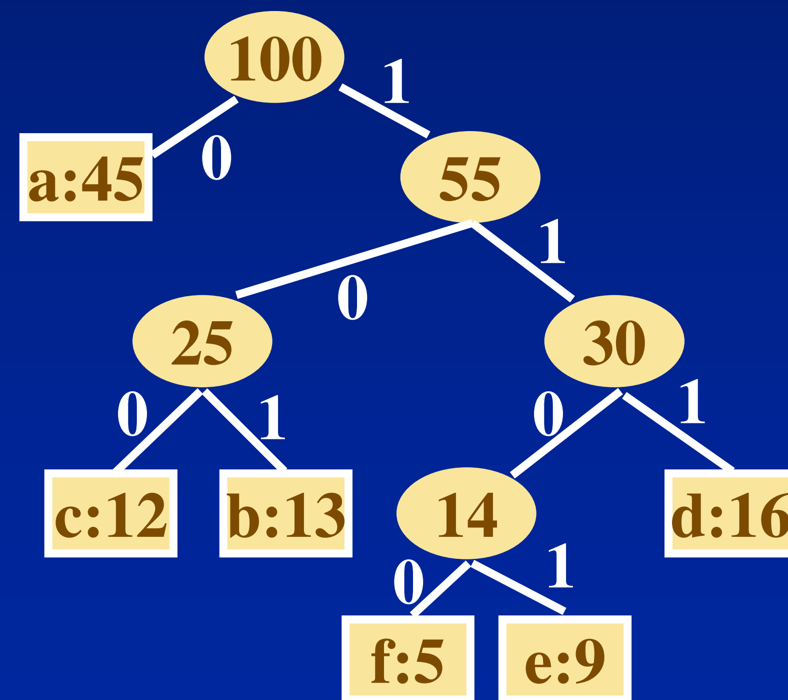
	a	b	c	d	e	f
Frequency (%)	60	11	12	15	1	1
Fixed-length code	000	001	010	011	100	101
Huffman code	0	101	100	111	1101	1100

- 100char: Fixed-length: 300 bits Huffman-code: 182 bits

Huffman codes

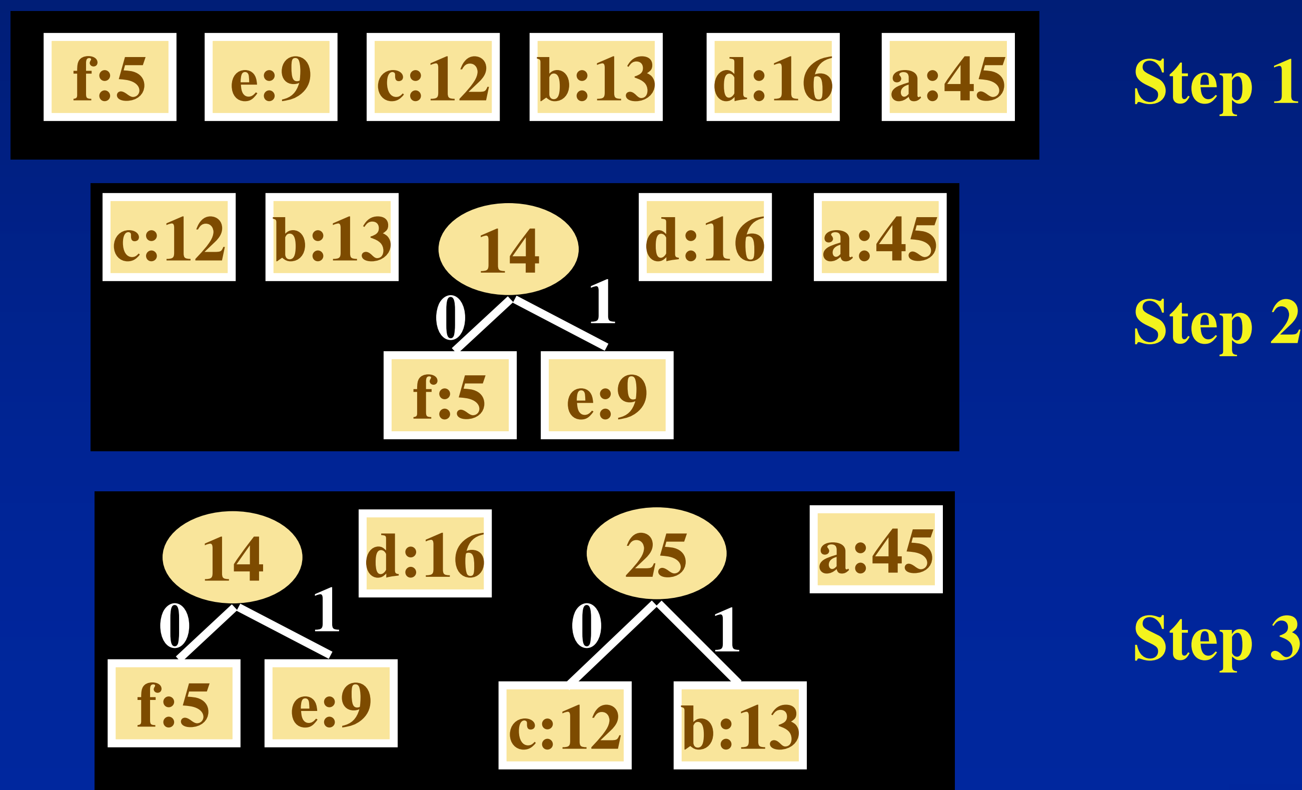
	a	b	c	d	e	f
Frequency (%)	60	11	12	15	1	1
Fixed-length code	000	001	010	011	100	101
Huffman code	0	101	100	111	1101	1100



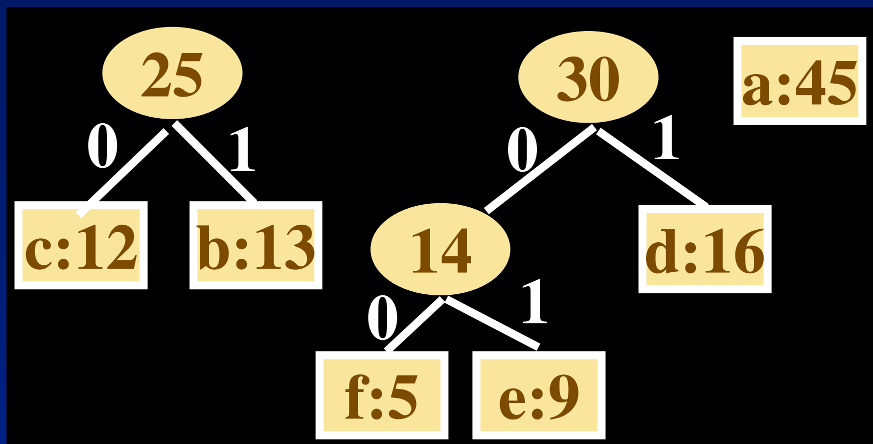


Construction of Huffman codes

- Sort objects
- Merge 2 objects with least frequencies

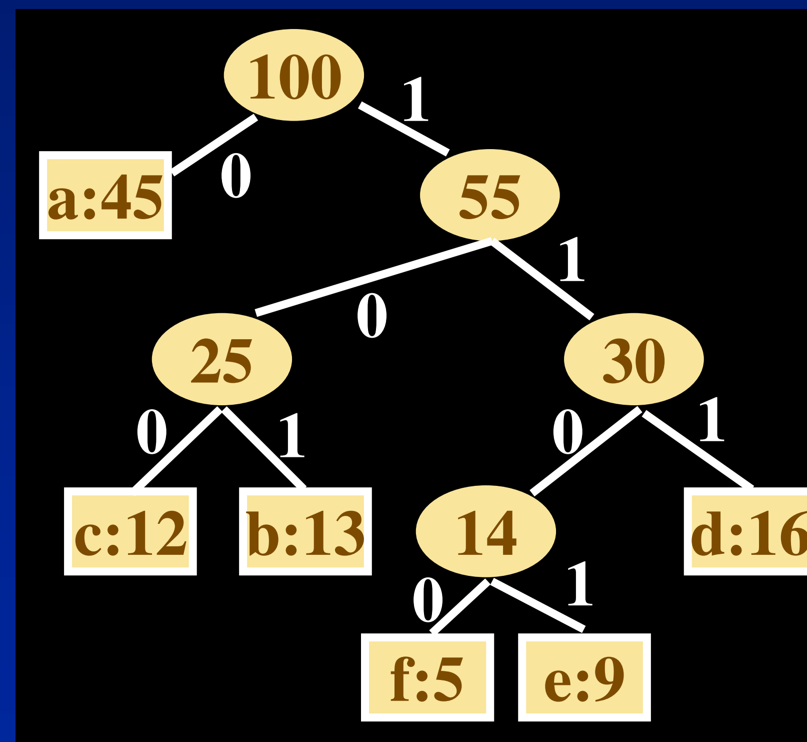


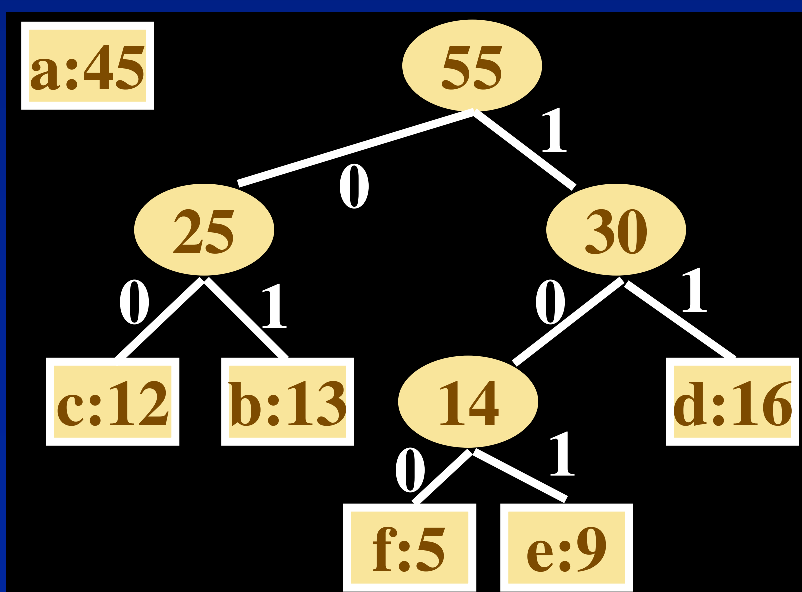
Huffman codes



Step 4

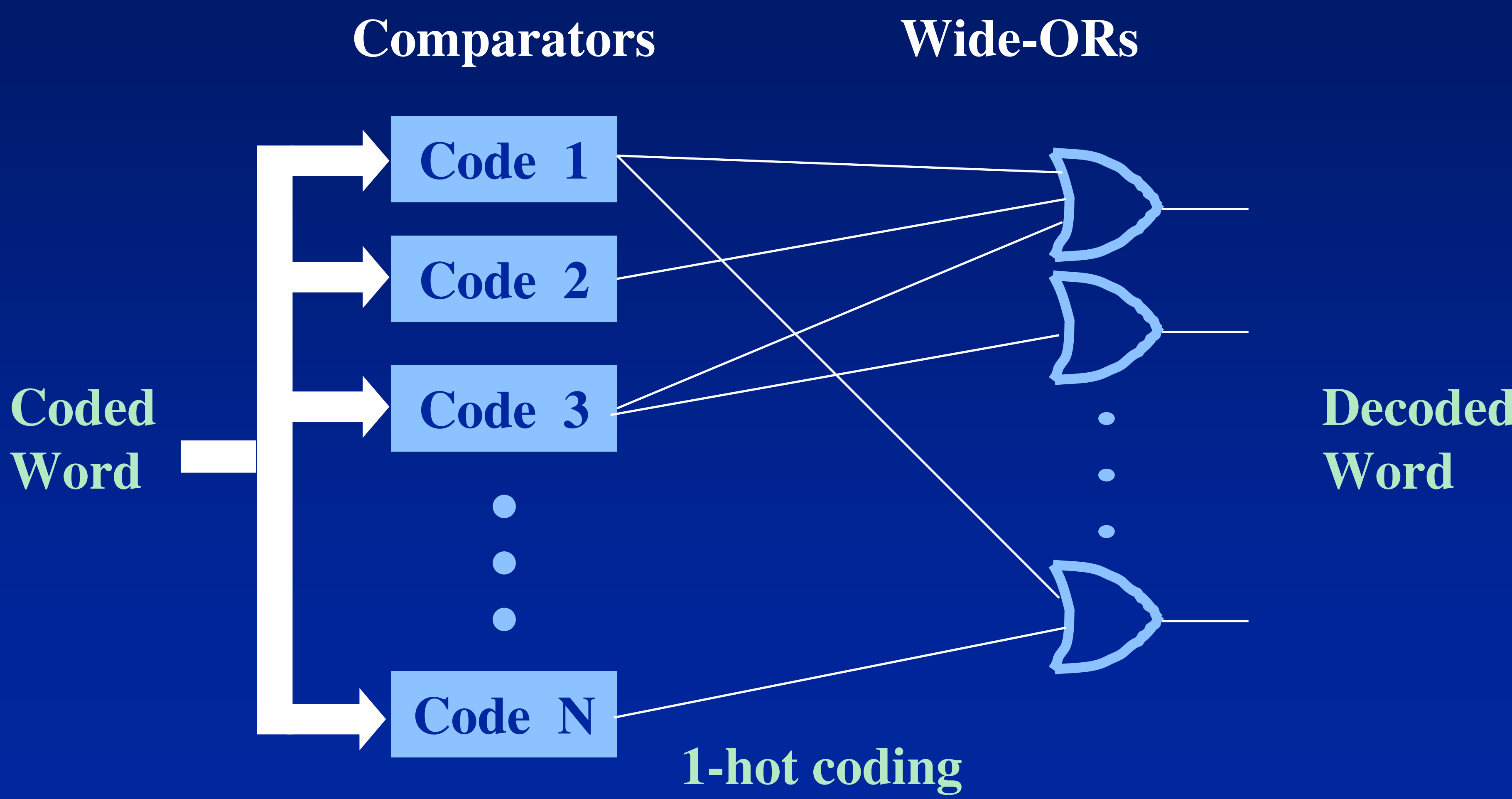
Step 6





Step 5

Huffman decoder in FPGA



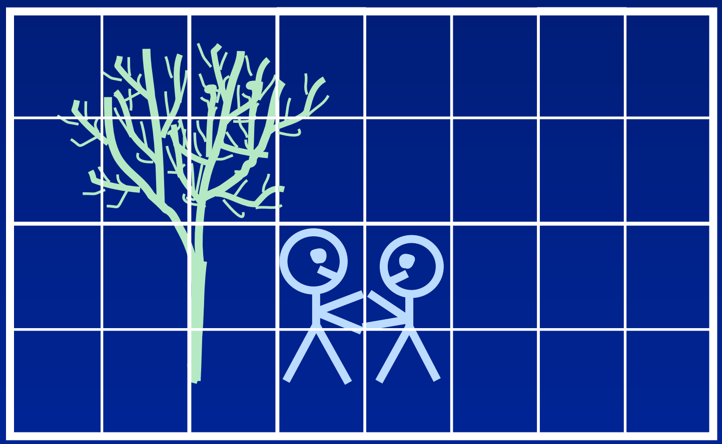
MPEG decoding in FPGA

- Variable-length decoding
  - Comparators
  - Wide-ORs
- IDCT
  - Distributed Arithmetic
- All these functions can be implemented very efficiently in VIRTEX

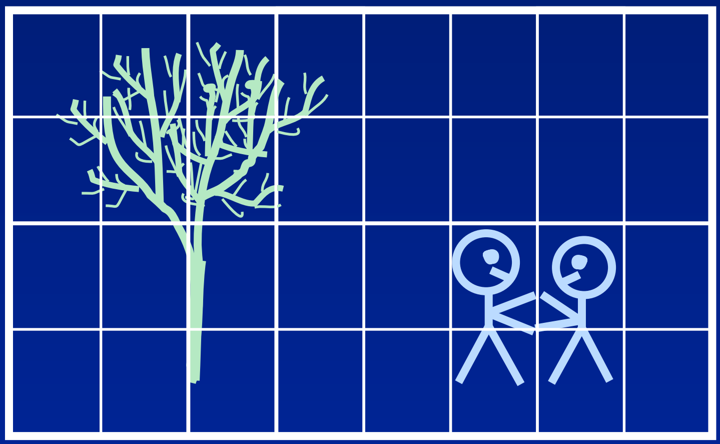
Motion compensation

- Consecutive video frames similar
- Small movement => block matches very likely
  - search space based on encoder
- Use motion vectors
- Four motion vectors for bidirectional prediction
- Residual error frame encoded efficiently

Motion compensation

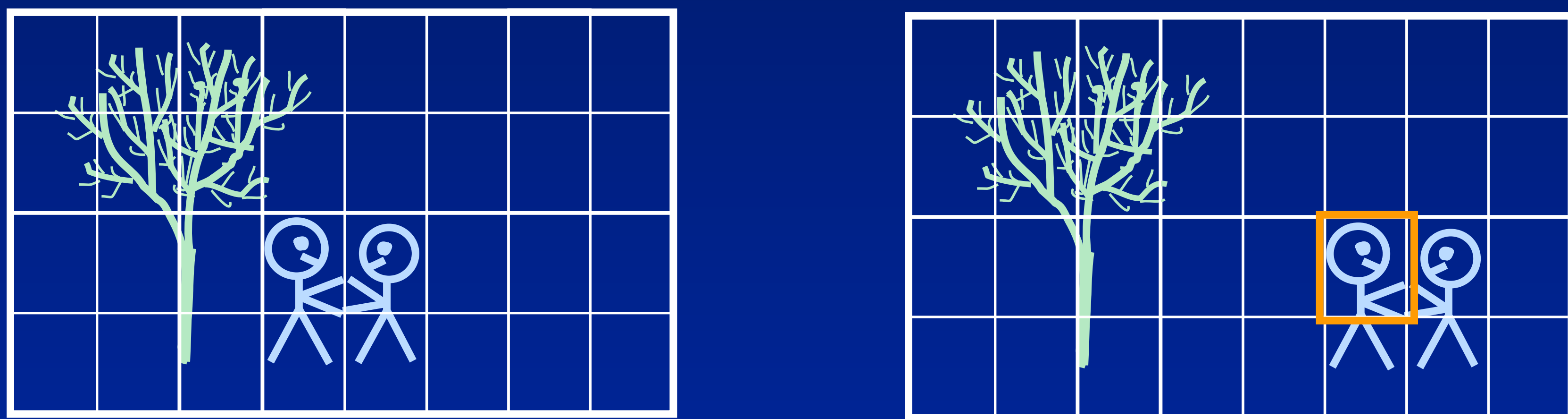


FRAME 1



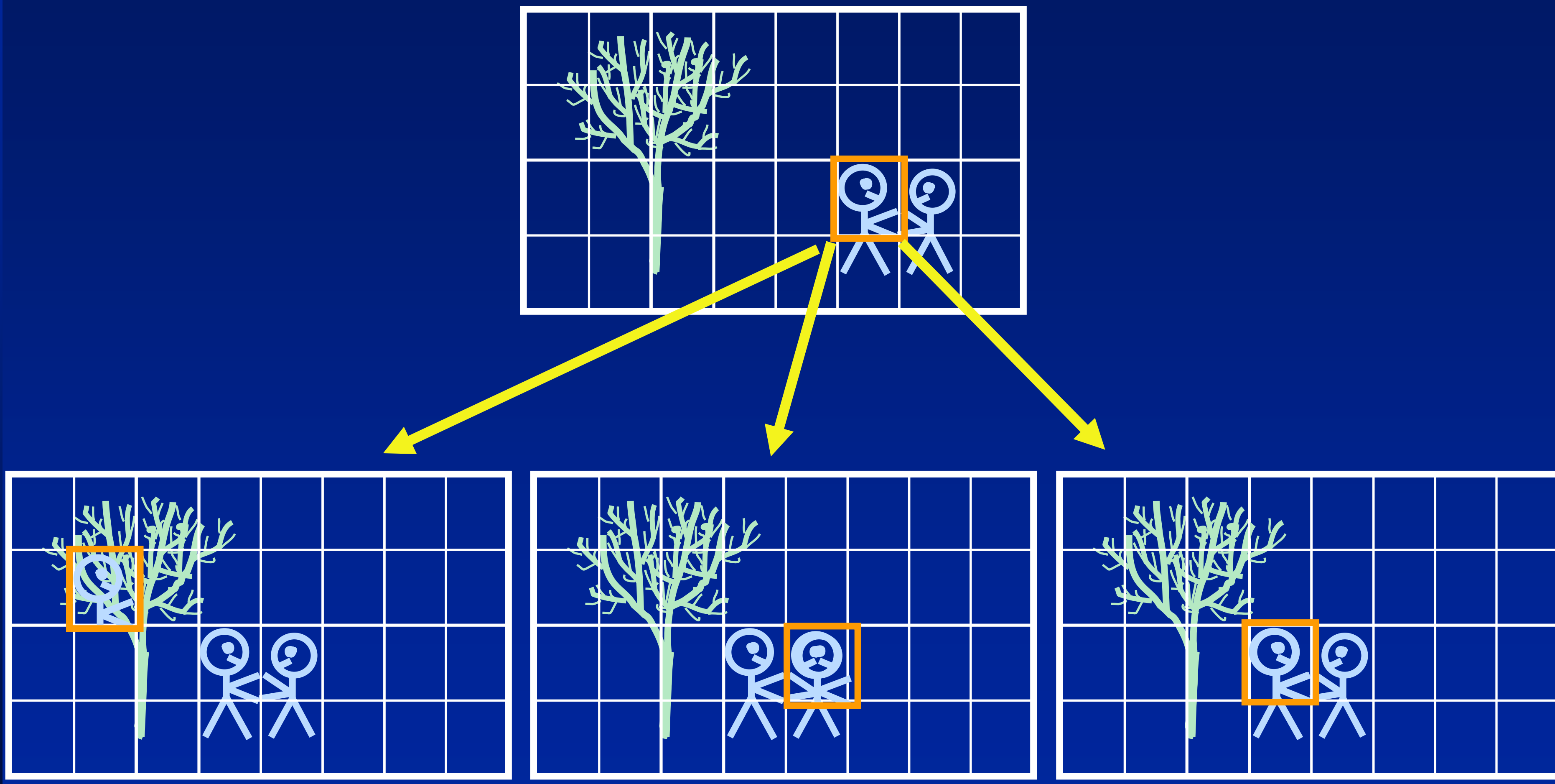
FRAME 2

Motion compensation



FRAME 1 FRAME 2

Motion compensation



BAD MATCH FAIR MATCH GOOD MATCH

Frames

- Transmission in form of series of frames
- Each frame comprises a set of macroblocks
- 3 types of frames
  - I frame
    - Exploits spatial redundancy
  - P/B frames
    - Exploit temporal redundancy

P Frames

- Predictor frames
- Macroblocks comprise motion-vectors relative to previous frame's macroblocks
- I P P P P P I P P P P P
  - P frame predicted from frame before
  - I frames coded spatially - no reference to any other frames

B Frames

- Bidirectional frames
- Macroblocks comprise motion-vectors relative to previous AND next frame's macroblocks
- I B P B P B I B P B P B
  - P frame predicted from frame before
  - I frames coded spatially - no reference to any other frames
  - B frame predicted from frame before and after

Conclusions

- Image compression
  - YCbCr color-space
  - DCT
  - Variable-length coding
  - Motion compensation
- Primary components for MPEG decoding (IDCT/Huffman decoding) can be implemented efficiently in FPGAs for speed