Image Compression with DCT on FPGA

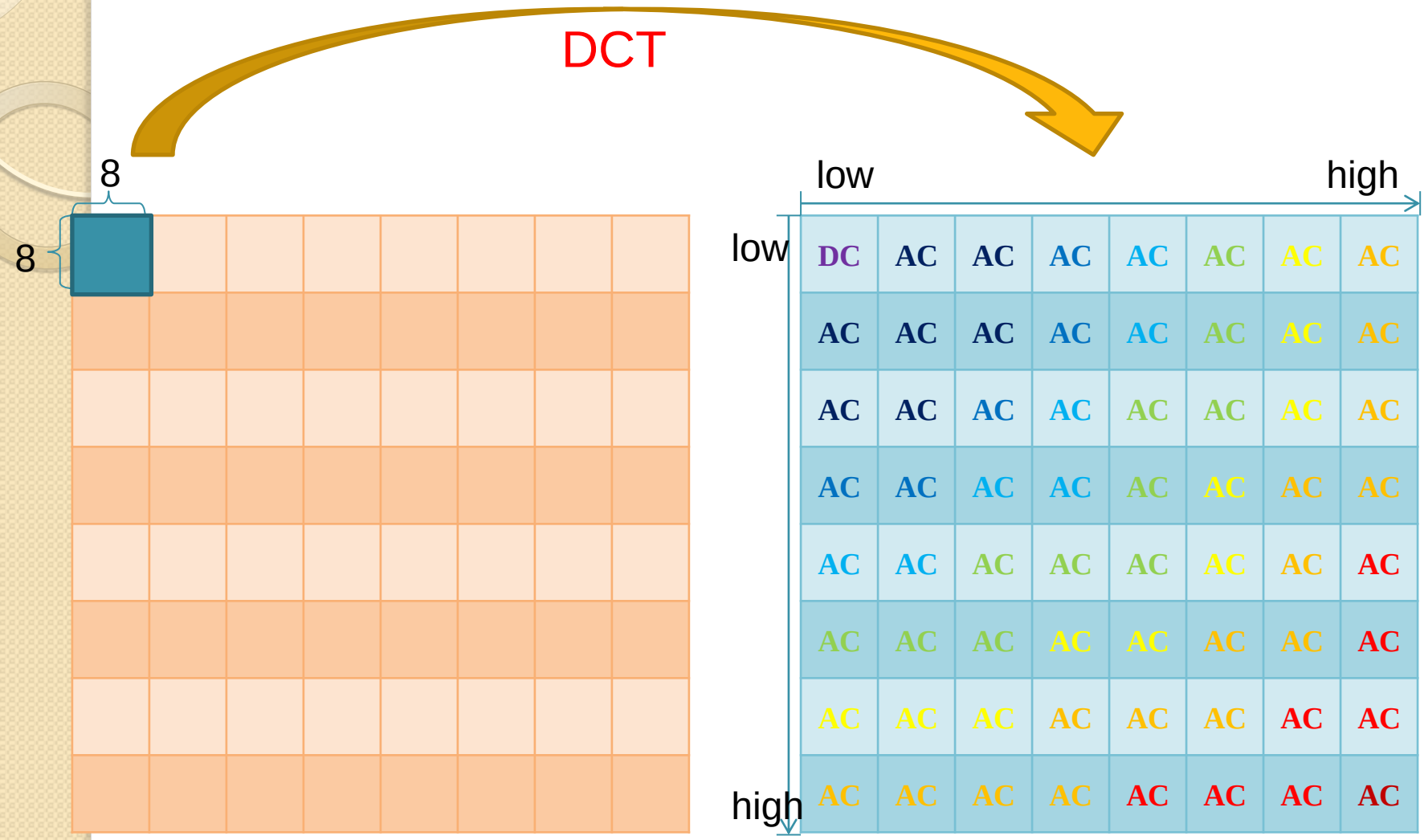
Dianfan Pan

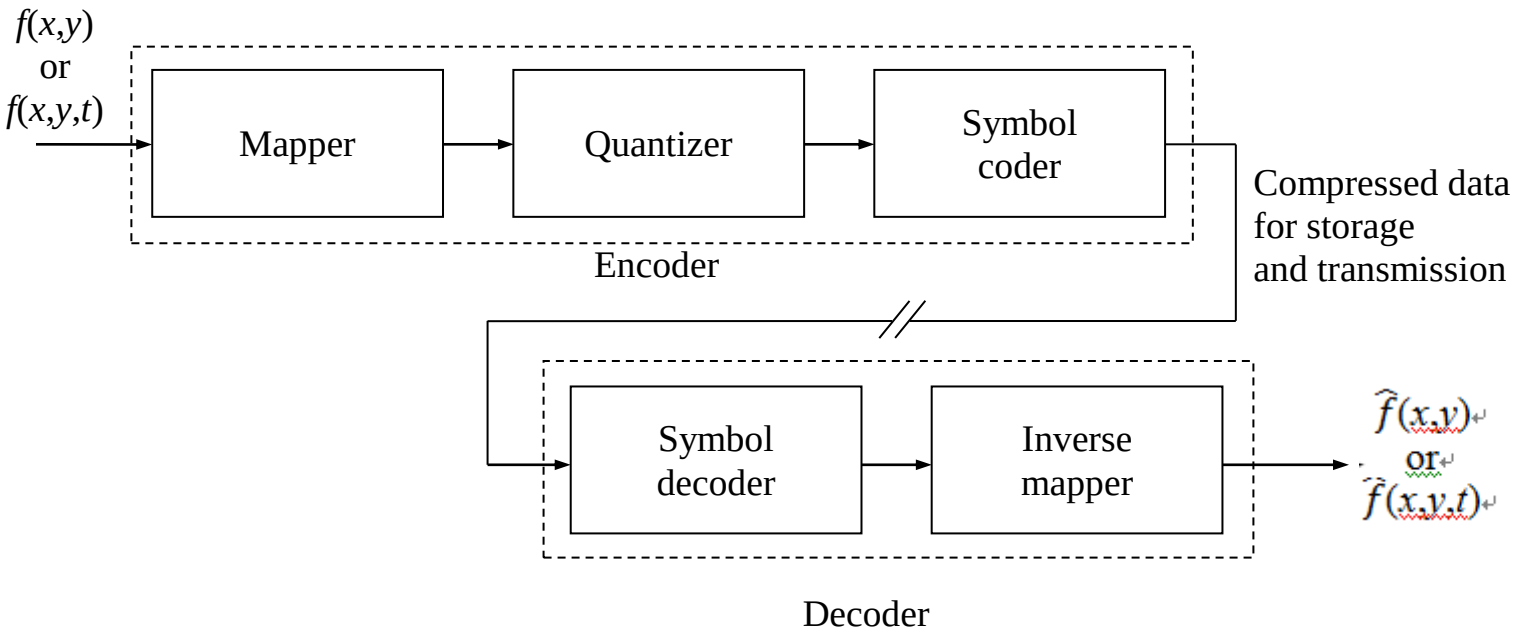
31080151

Instructor: Mustafa Torun

What is DCT?

- The DCT, is often used in signal and image processing, especially for lossy data compression, because it has a strong "energy compaction" property.
- Most of the signal information tends to be concentrated in a few low-frequency components of the DCT.


DCT
8
8
low
high
low
high
DC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC
AC AC AC AC AC AC AC AC


f(x,y)
or
f(x,y,t)
Mapper
Quantizer
Symbol coder
Encoder
Compressed data for storage and transmission
Symbol decoder
Inverse mapper
f̂(x,y)
or
f̂(x,y,t)
Decoder

1-D DCT

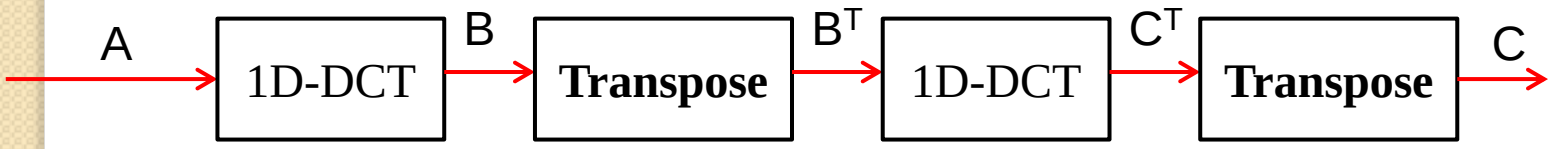
$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right]$$

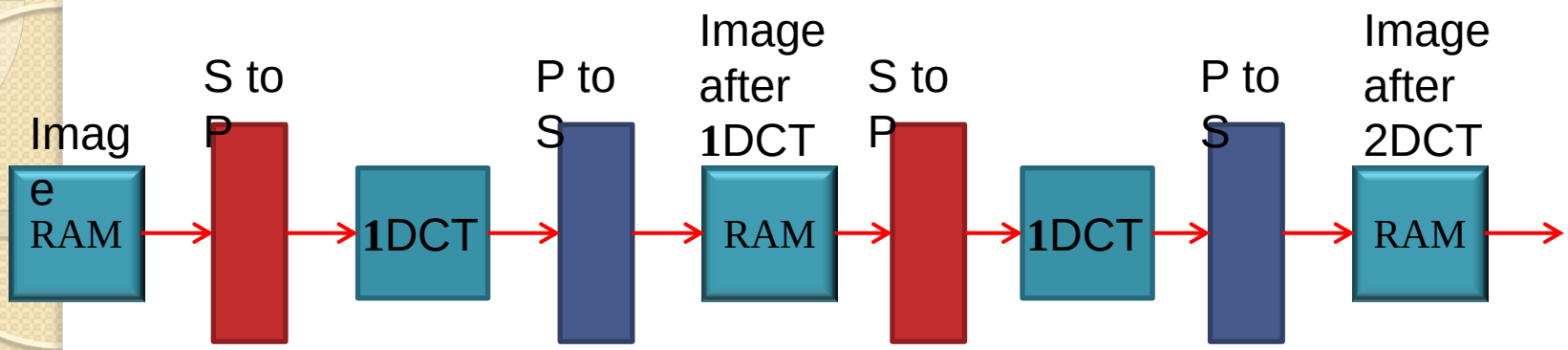
2-D DCT

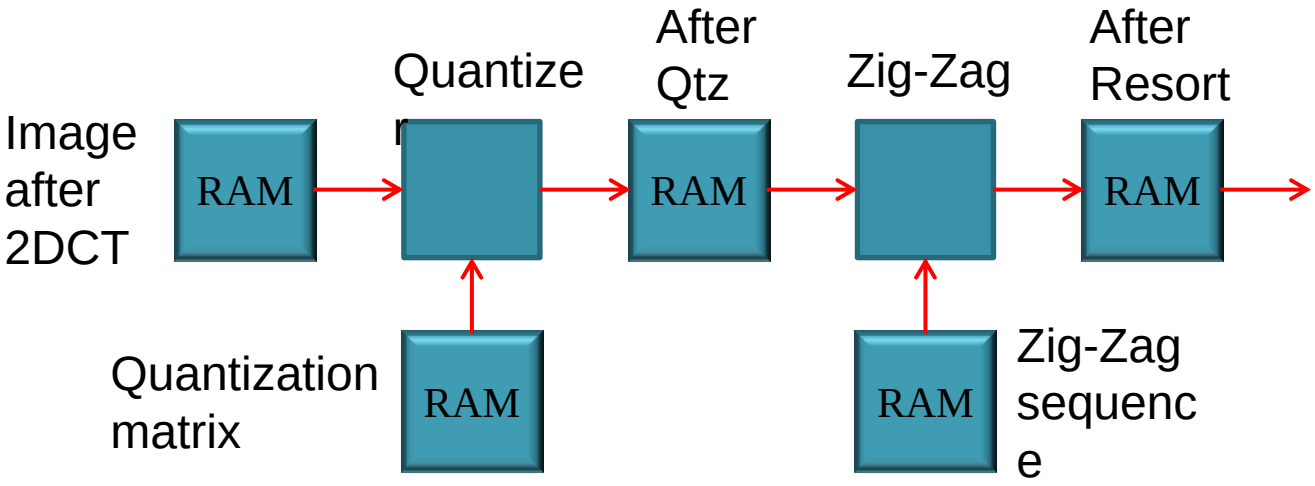
$$T_{u,v} = \sum_{x=0}^{7}\sum_{y=0}^{7} \alpha(u)\alpha(v) g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

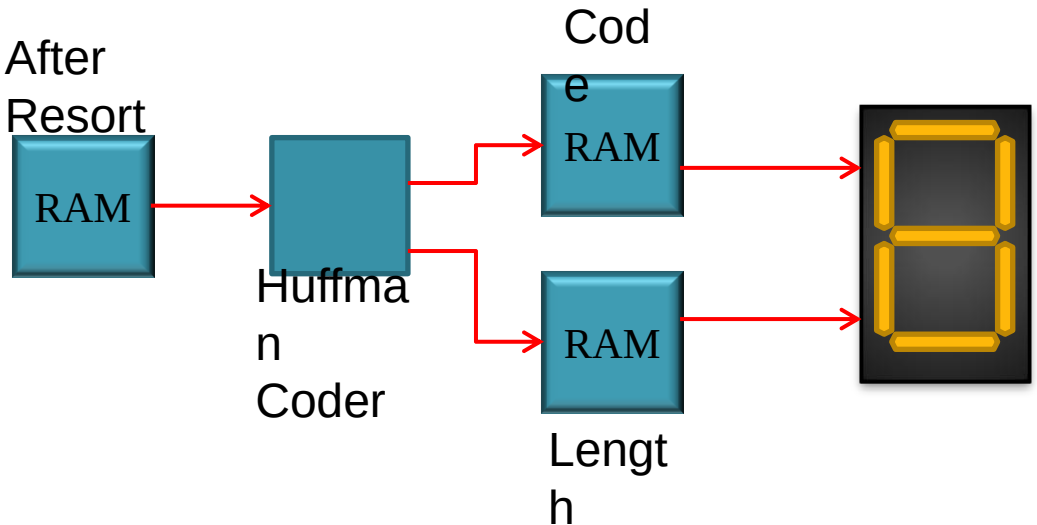
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } u = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$$

$$T_{u,v} = \sum_{y=0}^{7} \alpha(u) \underbrace{\sum_{x=0}^{7} \alpha(v) g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]}_{\text{1D-DCT}} \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$



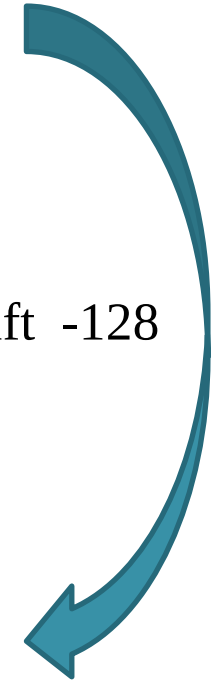

Imag
e
RAM
S to
P
1DCT
P to
S
Image
after
1DCT
RAM
S to
P
1DCT
P to
S
Image
after
2DCT
RAM


Image
after
2DCT
RAM
Quantize
r
After
Qtz
RAM
Zig-Zag
After
Resort
RAM
Quantization
matrix
RAM
RAM
Zig-Zag
sequenc
e


After
Resort
RAM
Huffma
n
Coder
Cod
e
RAM
RAM
Lengt
h

The original image

$$A=\begin{bmatrix}52 & 55 & 61 & 66 & 70 & 61 & 64 & 73\\ 63 & 59 & 66 & 90 & 109 & 85 & 69 & 72\\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73\\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69\\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70\\ 79 & 65 & 60 & 70 & 77 & 63 & 58 & 75\\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83\\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94\end{bmatrix}$$



shift -128

$$B=\begin{bmatrix}-76 & -73 & -67 & -62 & -58 & -67 & -64 & -55\\ -65 & -69 & -62 & -38 & -19 & -43 & -59 & -56\\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55\\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59\\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58\\ -49 & -63 & -68 & -58 & -51 & -65 & -70 & -53\\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45\\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34\end{bmatrix}$$

After DCT

$$C=\begin{bmatrix} -400 & -25 & -68 & 13 & 43 & -18 & -10 & -6 \\ -2 & -14 & -57 & 5 & 7 & -2 & -2 & 2 \\ -44 & 5 & 80 & -23 & -28 & 7 & 5 & -2 \\ -45 & 7 & 33 & -16 & -5 & 3 & -1 & 1 \\ 14 & -7 & -10 & 3 & -7 & 0 & -2 & 0 \\ -15 & -2 & 0 & -2 & -5 & -3 & 1 & 0 \\ -5 & 0 & 1 & 0 & 3 & 0 & 1 & -2 \\ 1 & -3 & 0 & -1 & -2 & -2 & -2 & 0 \end{bmatrix}$$

We can see that the high value components are concentrated in the upper left part

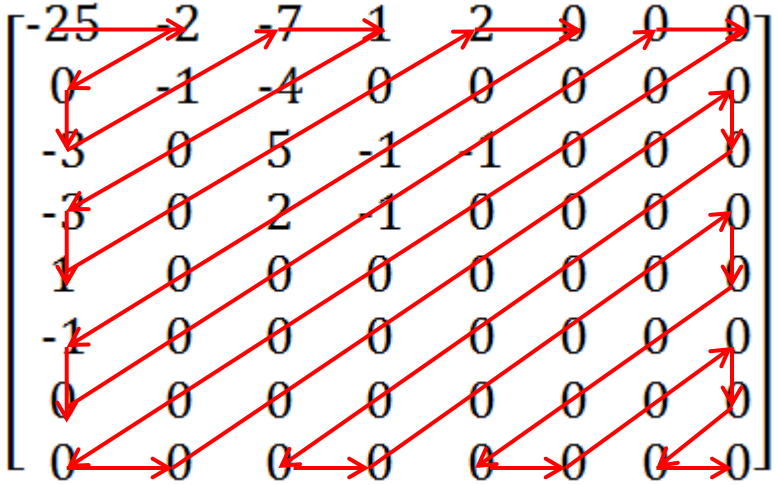
Quantization Matrix

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

After Quantization

$$D = \begin{bmatrix} -25 & -2 & -7 & 1 & 2 & 0 & 0 & 0 \\ 0 & -1 & -4 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Sequence of coding



Zig-Zag Pattern

[-25 2 0 -3 -1 -7 1 -4 0 -3 1 0 5 0 2 0 0
-1 2 0 -1 0 0 0 -1 -1 0 0 0 0 … 0]

38 zeros
EOB
(End of Block)

[-25 2 0 -3 -1 -7 1 -4 0 -3 1 0 5 0 2 0 0
-1 2 0 -1 0 0 0 -1 -1 EOB]



11000110 0101 11100100 000 100000
(-25) (2) (-3) (-1) (-7)
001 100011 11100100 001 1111001101
(1) (-4) (-3) (1) (5)
11100110 110110 0110 11000 1110100
(2) (-1) (2) (-1) (-1)
000 1010
(-1) (EOB)

11000110	0101	11100100	000	100000
8	4	8	3	6

001	100011	11100100	001	1111001101
3	6	8	3	10

11100110	110110	0110	11000	1110100
8	6	4	5	7

000	1010
3	4

Total length of code

=8+4+8+3+6+3+6+8+3

+10+8+6+4+5+7+3+4

=96 bits

Total length of the original image

=8×8×8 bits=512 bits

Compression ratio = 512 : 96 = 5.3 : 1

After inverse quantization

$$\begin{bmatrix} -400 & -22 & -70 & 16 & 48 & 0 & 0 & 0 \\ 0 & -12 & -56 & 0 & 0 & 0 & 0 & 0 \\ -42 & 0 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 0 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After inverse DCT

$$\begin{bmatrix} -76 & -66 & -58 & -62 & -69 & -68 & -58 & -48 \\ -68 & -66 & -51 & -31 & -27 & -40 & -54 & -59 \\ -65 & -73 & -52 & -6 & 9 & -22 & -59 & -75 \\ -59 & -74 & -52 & 1 & 17 & -23 & -62 & -70 \\ -49 & -66 & -52 & -12 & -6 & -40 & -59 & -50 \\ -52 & -67 & -63 & -45 & -46 & -65 & -64 & -45 \\ -51 & -62 & -67 & -66 & -66 & -68 & -60 & -47 \\ -35 & -43 & -53 & -59 & -54 & -46 & -40 & -39 \end{bmatrix}$$

Original image

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 66 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 63 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Output image

$$\begin{bmatrix} 52 & 62 & 70 & 66 & 59 & 60 & 70 & 80 \\ 60 & 62 & 77 & 97 & 101 & 88 & 74 & 69 \\ 63 & 55 & 76 & 122 & 137 & 106 & 69 & 53 \\ 69 & 54 & 76 & 129 & 145 & 105 & 66 & 58 \\ 79 & 62 & 76 & 116 & 122 & 88 & 69 & 78 \\ 76 & 61 & 65 & 83 & 82 & 63 & 64 & 83 \\ 77 & 66 & 61 & 62 & 62 & 60 & 68 & 81 \\ 93 & 85 & 75 & 69 & 74 & 82 & 88 & 89 \end{bmatrix}$$

Error

$$\begin{bmatrix} 0 & -7 & -9 & 0 & 11 & 1 & -6 & -7 \\ 3 & -3 & -11 & -7 & 8 & -3 & -5 & 3 \\ -1 & 4 & -8 & -9 & 7 & -2 & -3 & 20 \\ -6 & 4 & -5 & -7 & 9 & 1 & 4 & 11 \\ -12 & -1 & -8 & -12 & 4 & 0 & -1 & -8 \\ 3 & 4 & -5 & -13 & -5 & 0 & -6 & -8 \\ 8 & 5 & 3 & -3 & -7 & 1 & -3 & 2 \\ -6 & -6 & -6 & -1 & -9 & -6 & -10 & 5 \end{bmatrix}$$

$RMS_{error} = 6.7315$

Reference

- [1] Chen Pu-yue, “Fast 2-D DCT Algorithm and Its Implementation Based On FPGA”, Nanjing University of PLA of Technology.
- [2] Li Li, “FPGA Implementation of 2-D DCT-based Distributed Arithmetic”, Beijing University of Posts and Telecommunications.
- [3] JPEG, Wikipedia.
- [4] DCT, Wikipedia.

Thank you!