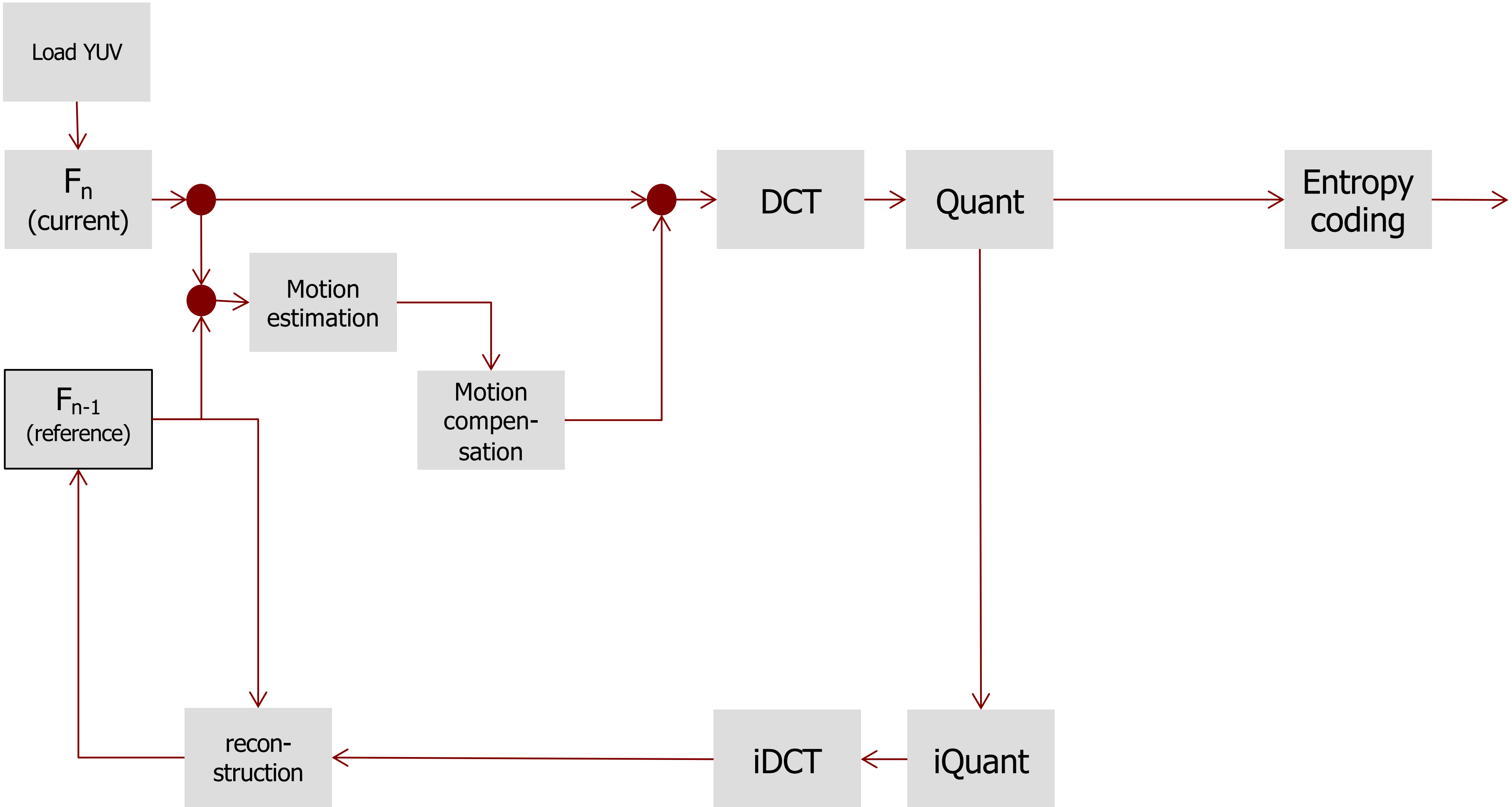
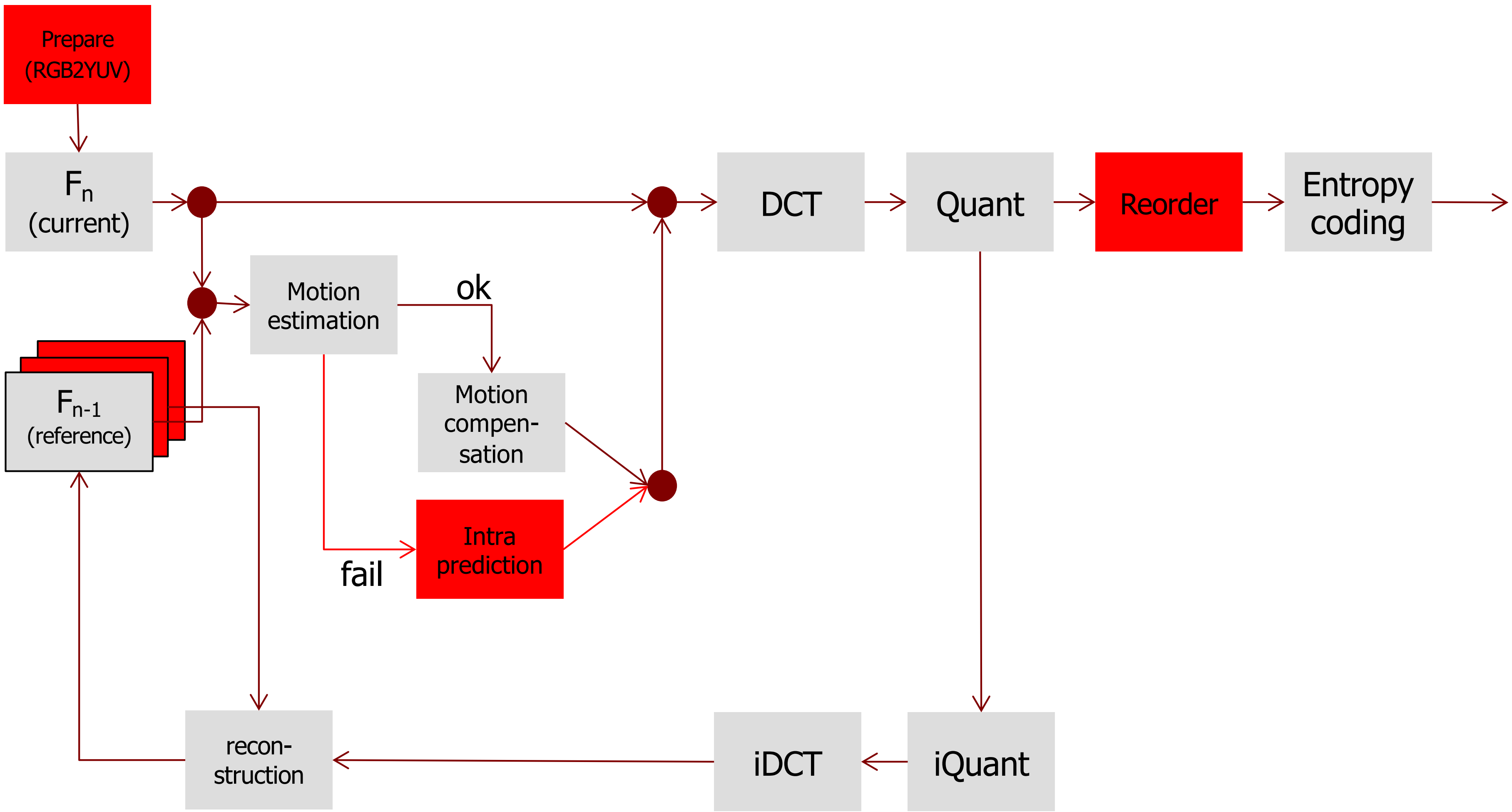
Codec 63



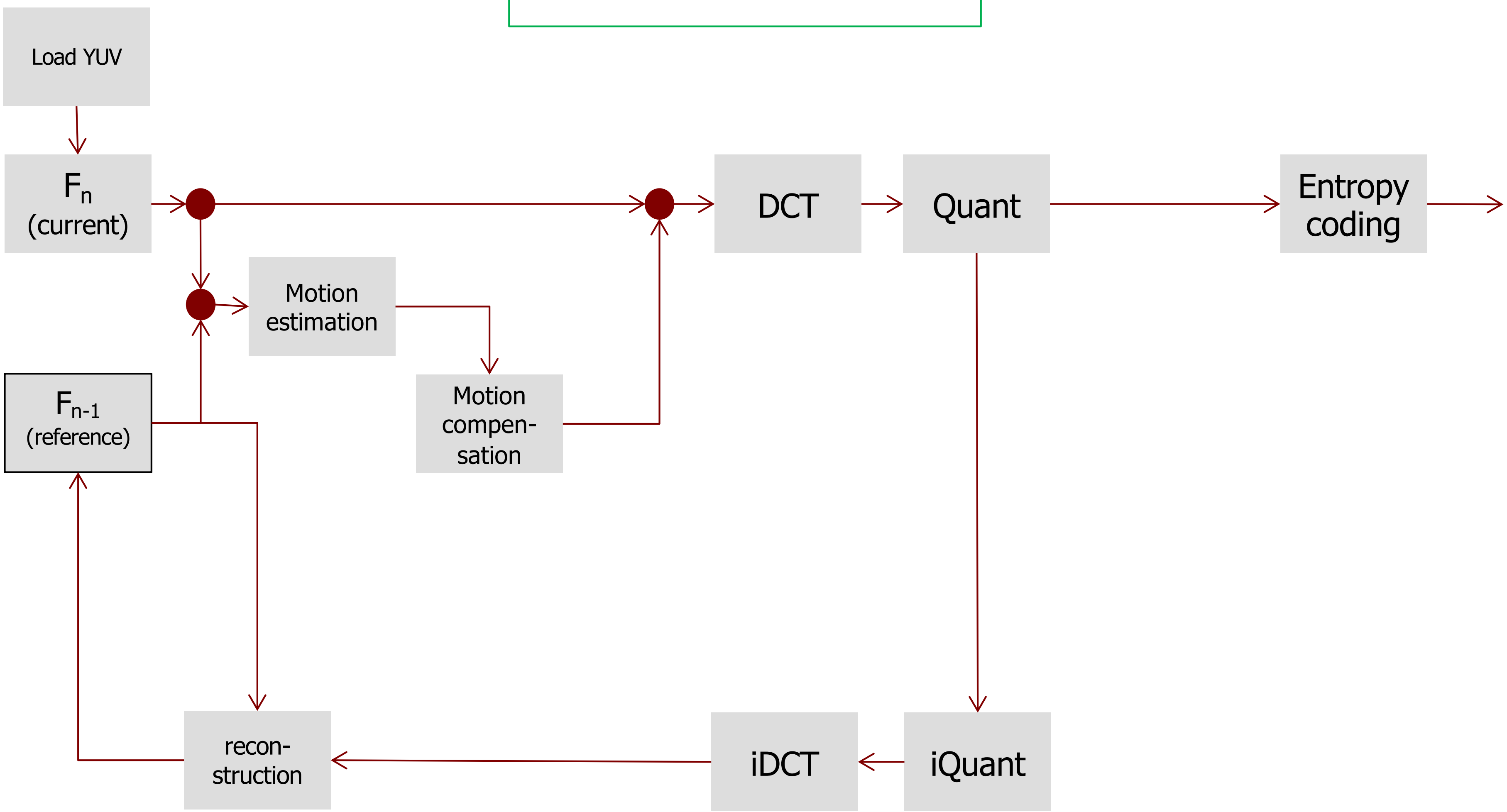
Load YUV
F
n
(current)
DCT
Quant
Entropy coding
Motion estimation
F
n-1
(reference)
Motion compen-sation
recon-struction
iDCT
iQuant

Not in Codec 63

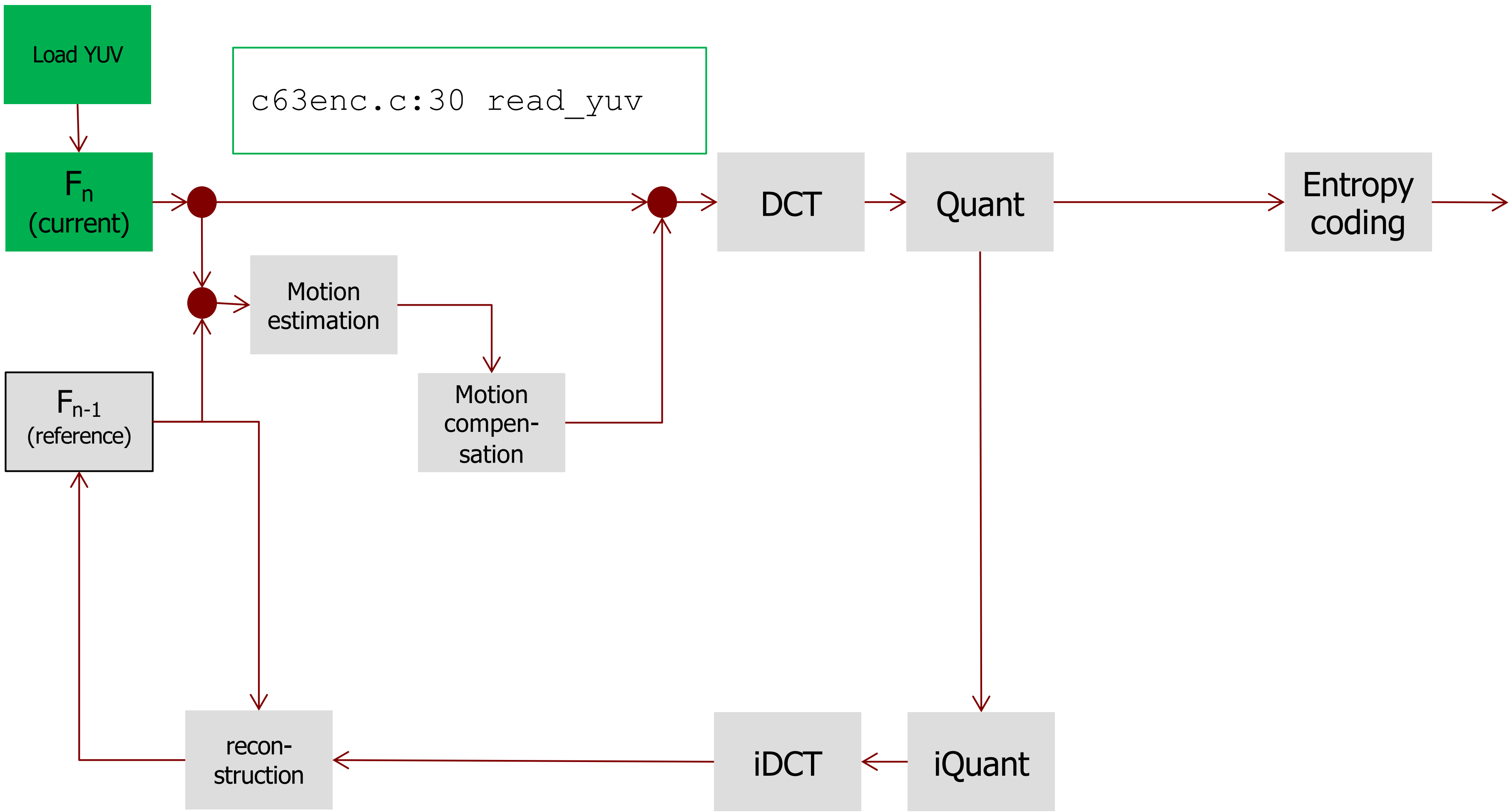


main

```
c63enc.c:193 main
```

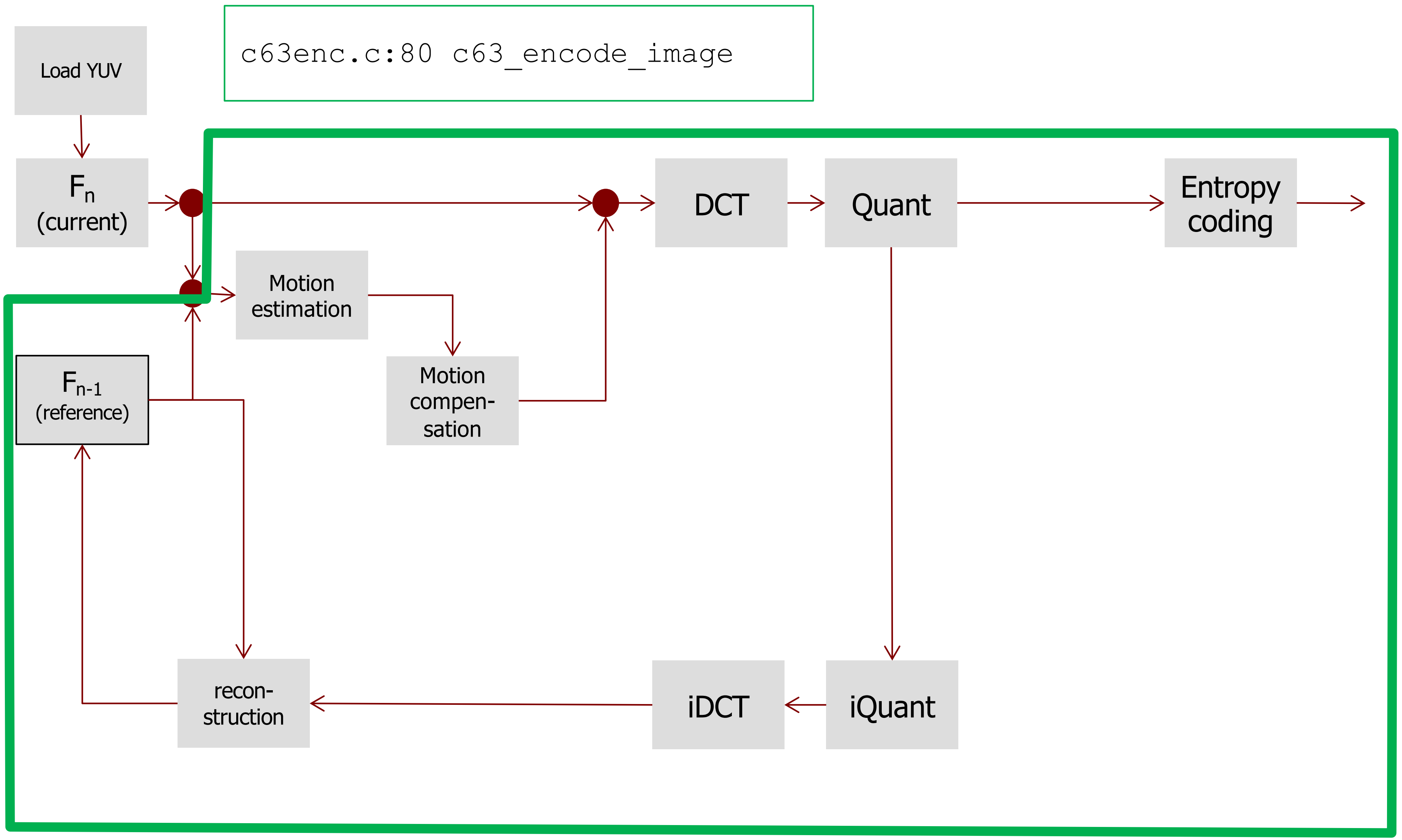


read_yuv



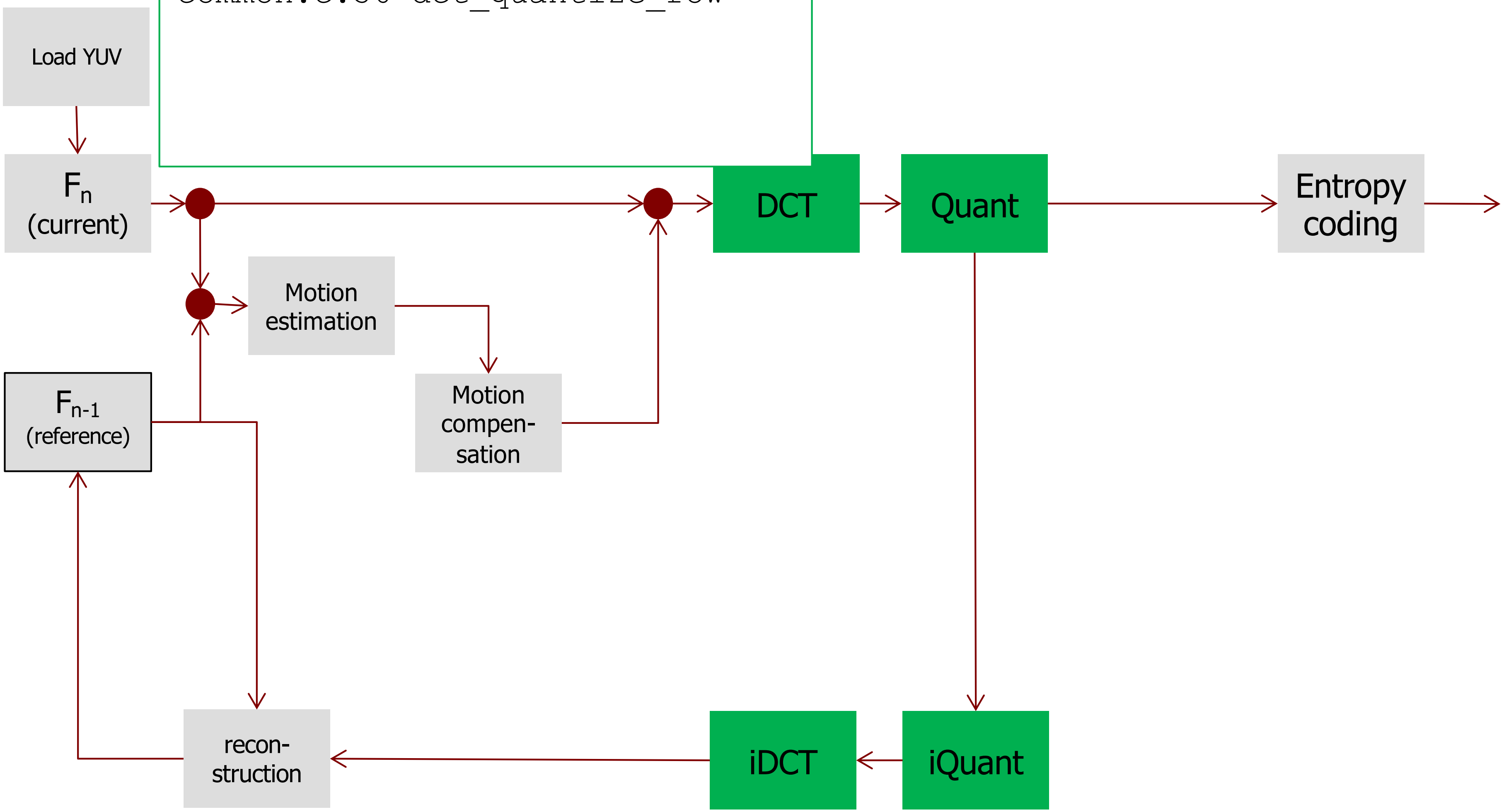
Load YUV
c63enc.c:30 read_yuv
Fn (current)
DCT
Quant
Entropy coding
Motion estimation
Motion compen-sation
Fn-1 (reference)
recon-struction
iDCT
iQuant

c63_encode_image

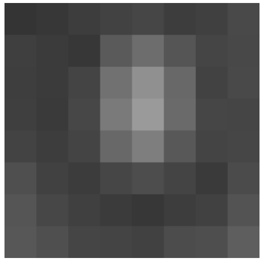


c63_encode_image

```
common.c:83 dct_quantize
common.c:56 dct_quantize_row
```



DCT

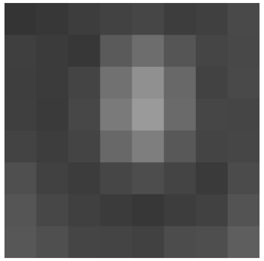


$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} - 128 = \quad g = \overset{x \longrightarrow}{\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}} \Big\downarrow y.$$

Each 8 × 8 block (Y, Cb, Cr) is converted to a frequency-domain representation, using a normalized, two-dimensional DCT

- two-dimensional DCT: $G_{u,v} = \alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$
  - $G_{u,v}$ is the DCT at output coordinates (u,v)
  - u and v are from $\{0, ..., 7\}$
  - $g_{x,y}$ is the pixel value at input coordinates (x,y)
  - α is a normalizing function: $\alpha(u) = \begin{cases} \sqrt{\frac{1}{8}}, & \text{if } u = 0 \\ \sqrt{\frac{2}{8}}, & \text{otherwise} \end{cases}$

DCT



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} - 128 = \quad g = \overset{\xrightarrow{x}}{\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}} \Big\downarrow y.$$

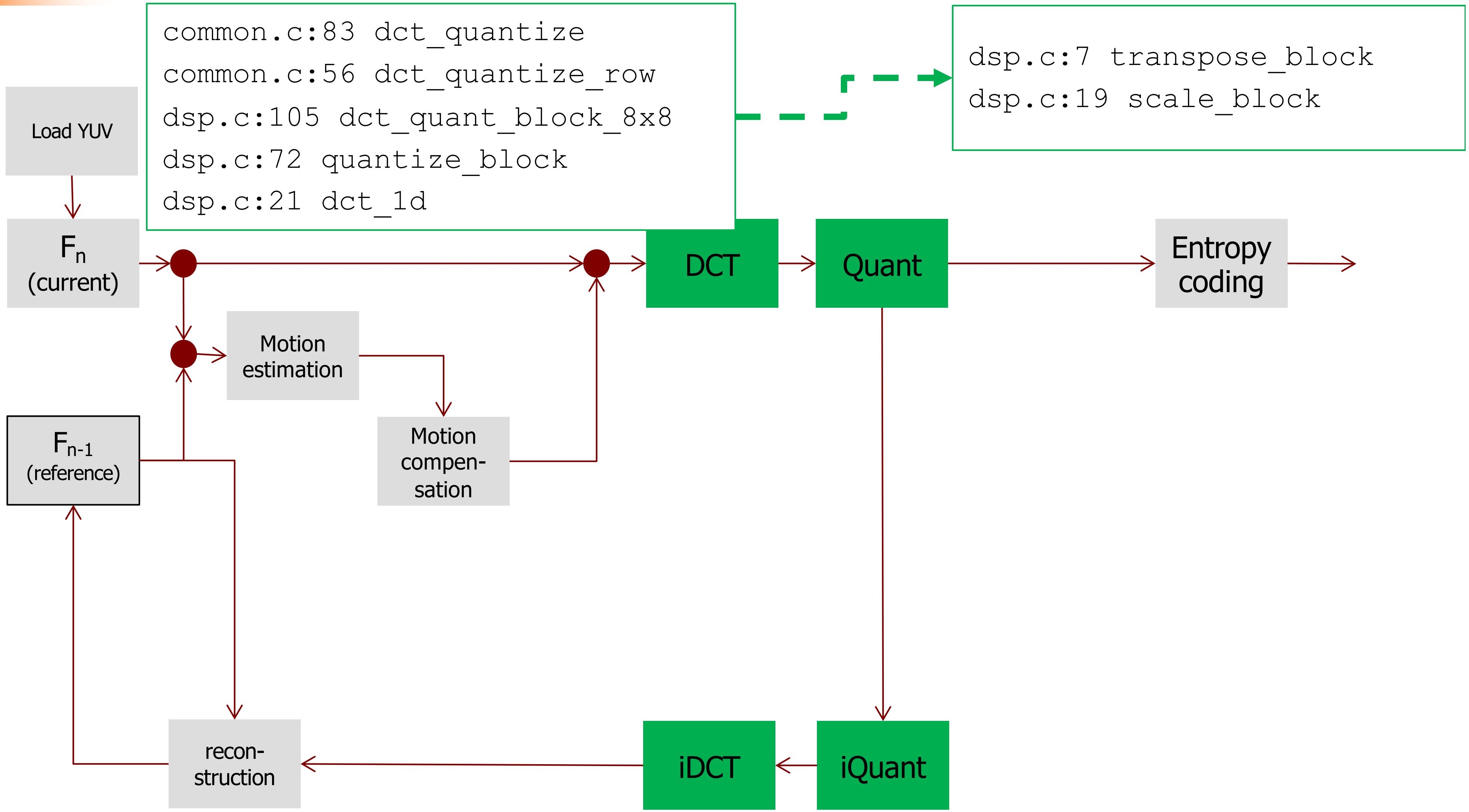
A 2D DCT can be replaced by applying a 1D DCT twice

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

can be replaced by

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

c63_encode_image



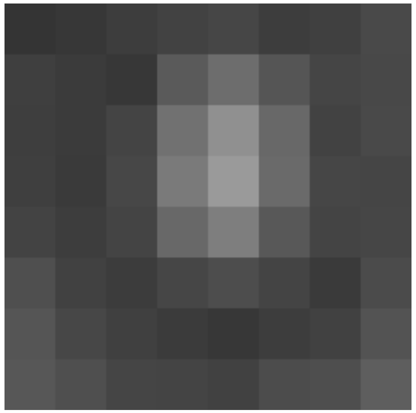
Quantization Example

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \overset{u}{\longrightarrow} \; \downarrow v.$$

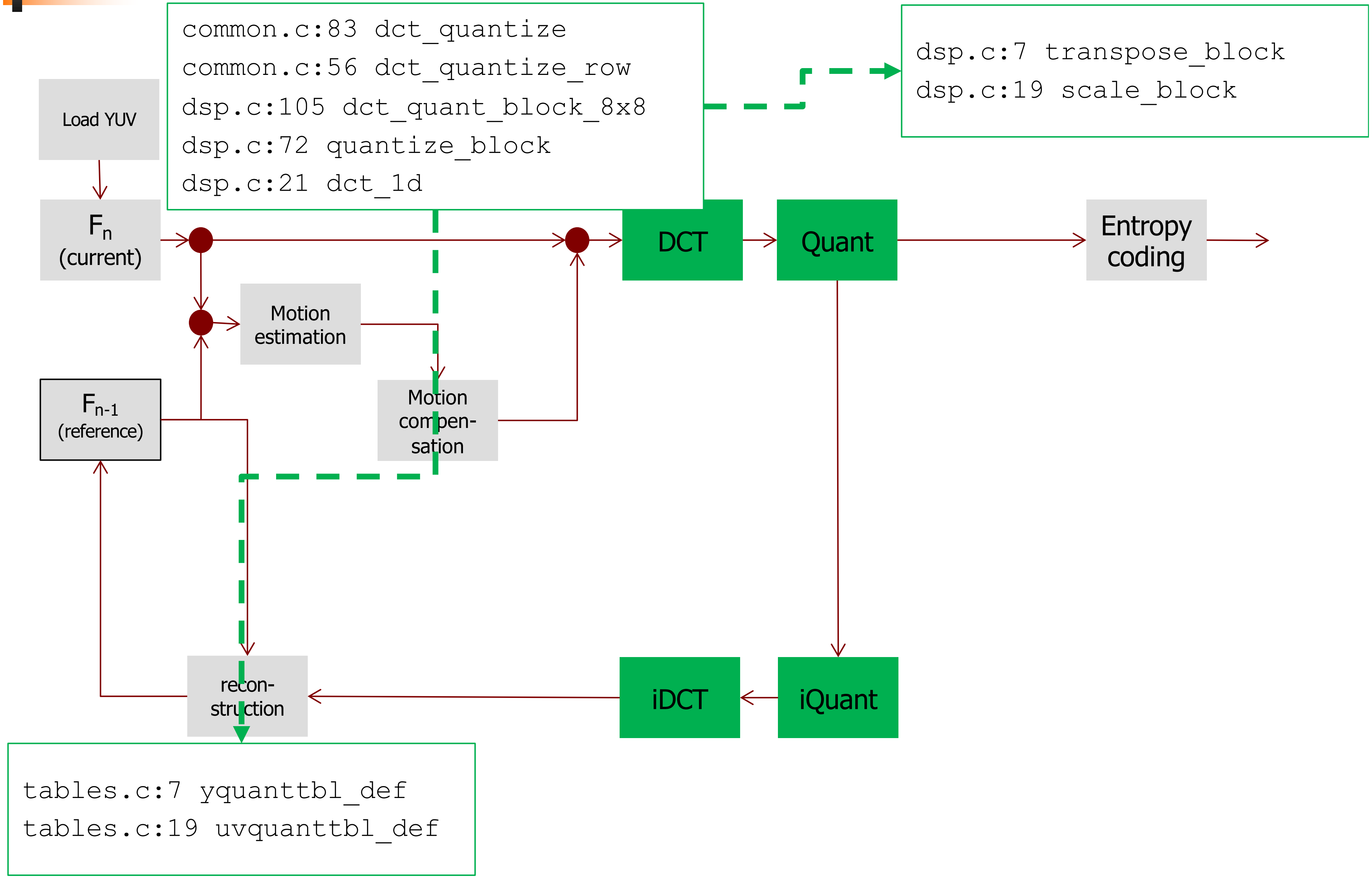
$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121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

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0,1,2,\ldots,7; k = 0,1,2,\ldots,7$$

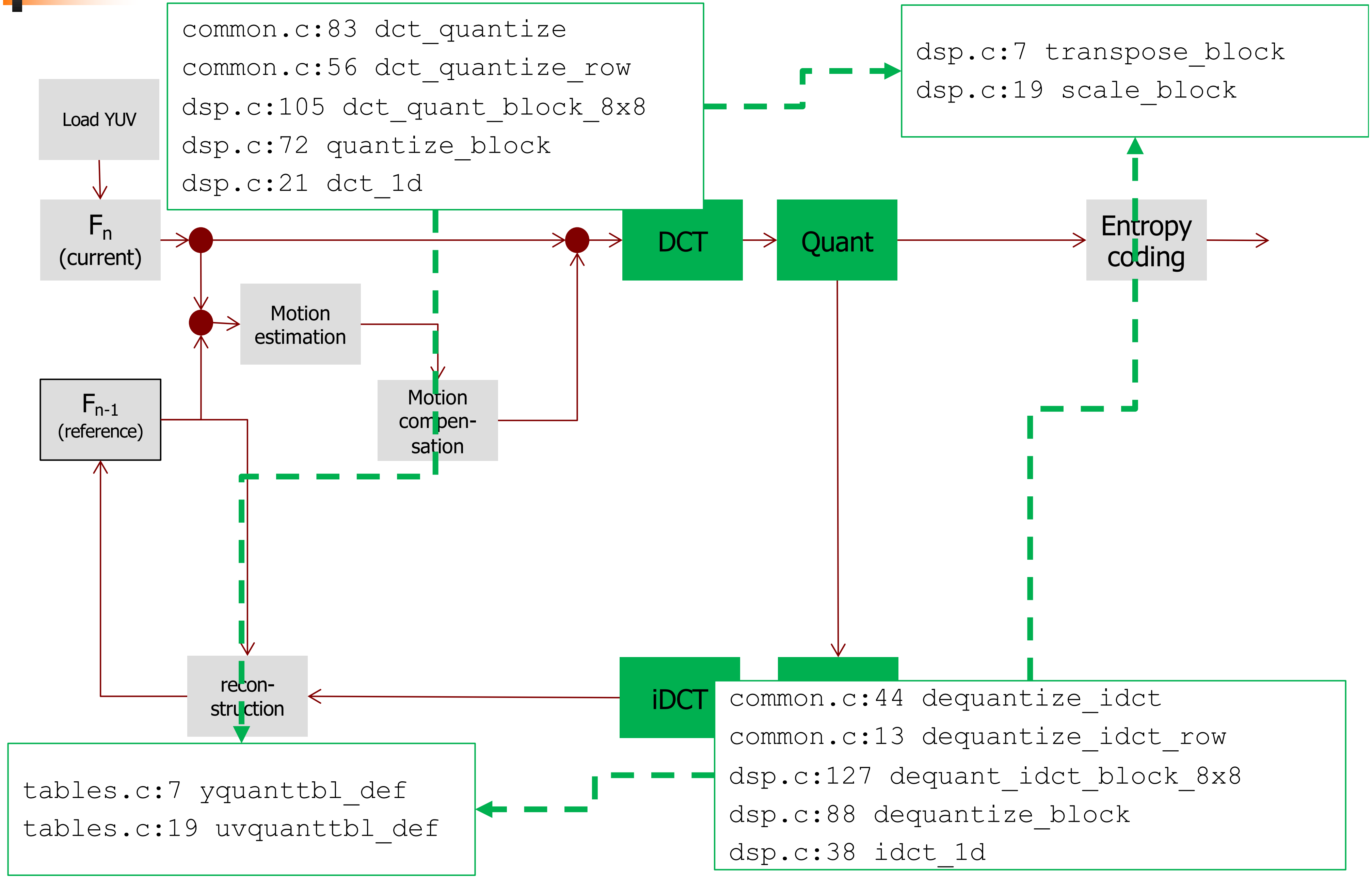
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$



c63_encode_image



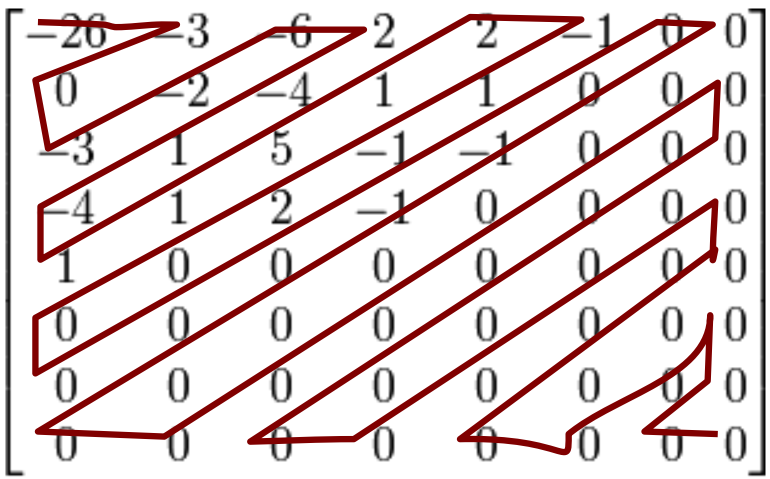
c63_encode_image

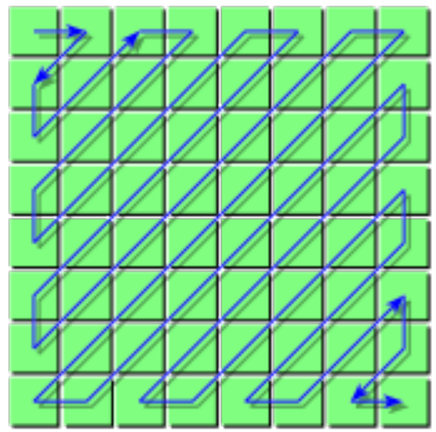


Lossless compression

The resulting data for all 8 × 8 blocks is further compressed with a loss-less algorithm

1. organize numbers in **zigzag pattern**

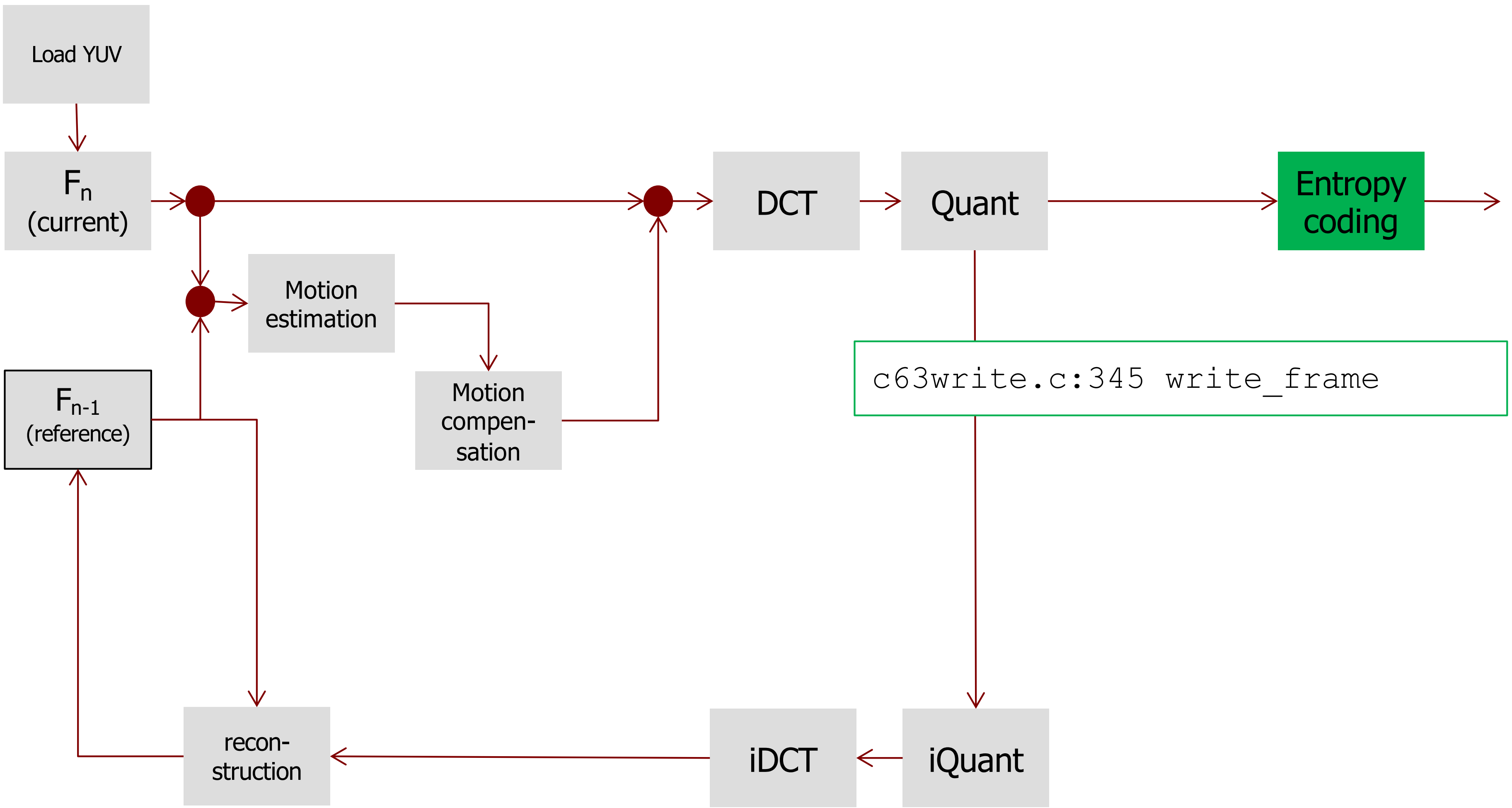




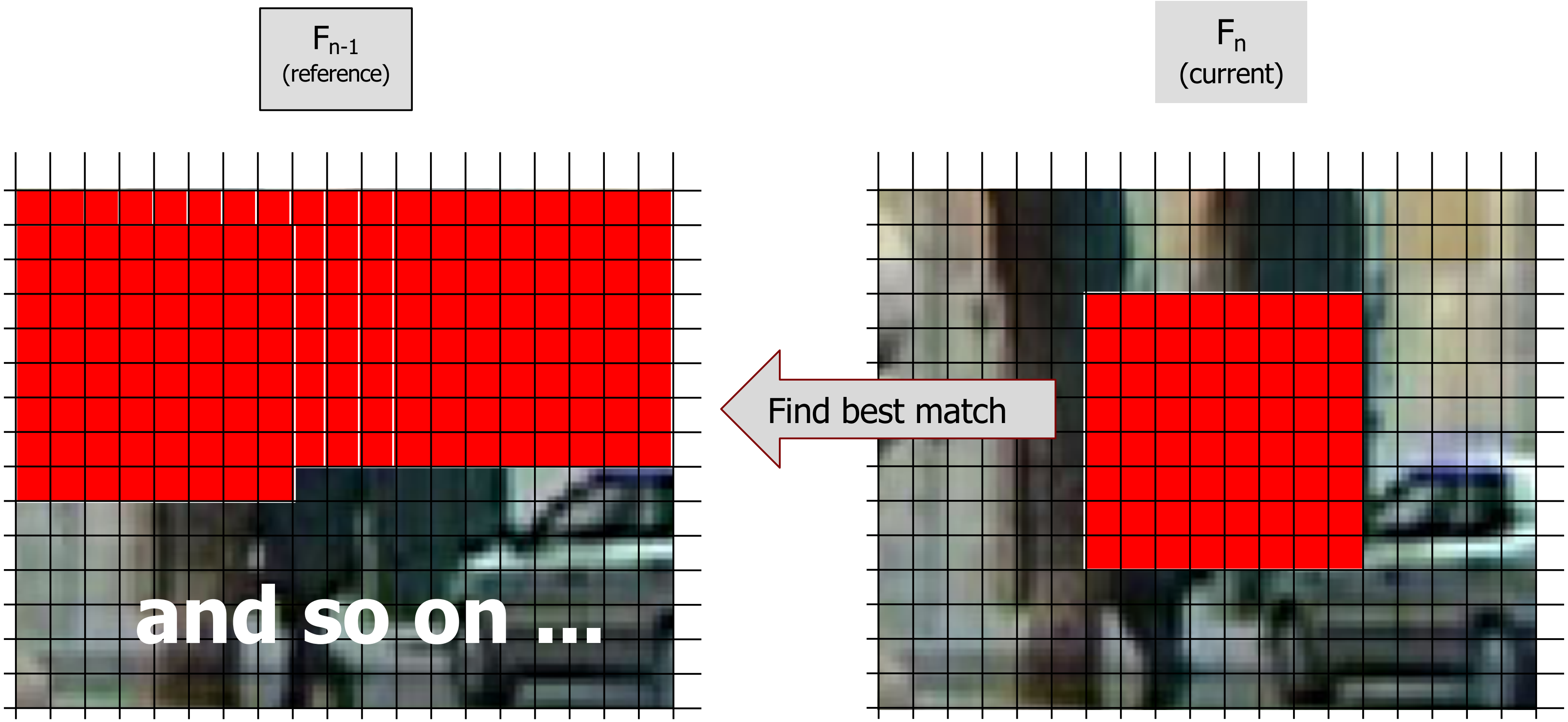
→ -26, -3, 0, -3, -2, -6, 2, -4, 1, -4, 1, 1, 5, 1, 2, -1, 1, -1, 2, 0, 0, 0, 0, 0, -1, -1, 0 , 0, 0, 0, 0, 0, 0, 0, 0, …., 0, 0

2. run-length coding

c63_encode_image



Full Search Motion Estimation



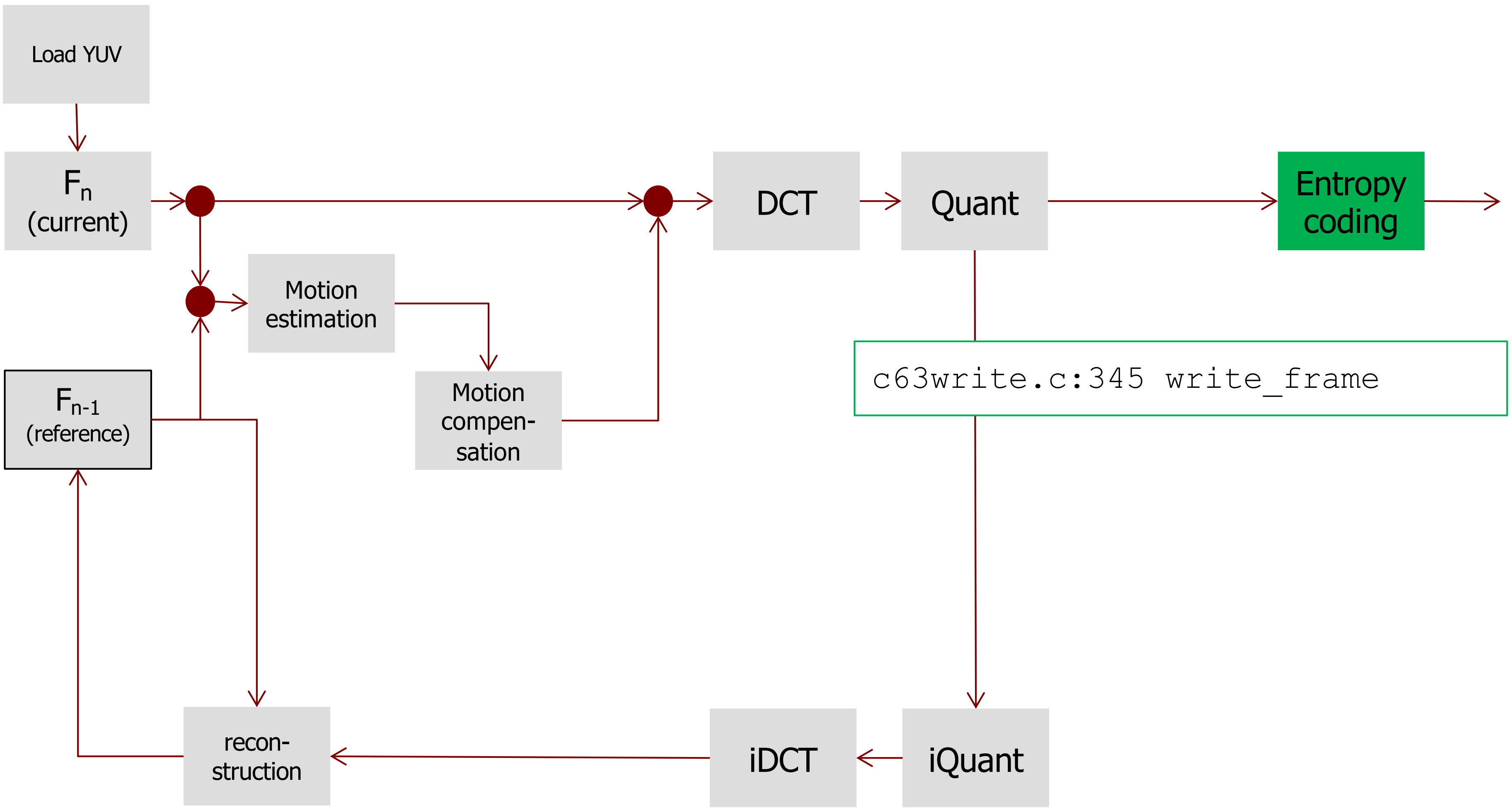
For comparing blocks:

- SAD - Sum of Absolute Differences

$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|$$

W: fixed set, but not only integers !

c63_encode_image



Motion Estimation

- The estimators often use a two-step process, with initial coarse evaluation and refinements

- Don’t do this for every frame, you must sometimes encode macroblocks in a “safe” mode that doesn’t rely on others

- This is called “Intra”-mode
  - When a complete frame is encoded in I-mode (always in MPEG-1 and MPEG-2), this is called an I-frame
  - x264 calls I-frames "keyframes". But the word keyframe has many, many other meanings as well. Avoid misunderstandings by writing I-frame.

- Refinements include trying every block in the area, and also using sub-pixel precision (interpolation)
  - quarter pixel in H.264

Motion Compensation

- When the best motion vector has been found and refined, a predicted image is generated using the motion vectors

- The reference frame can not be used directly as input to the motion compensator
  - The decoder never sees the original image. Instead, it sees a *reconstructed* image, i.e. an image that has been quantized (with loss)

- A reconstructed reference image must be used as input to motion compensation

Frame Reconstruction

- The motion compensator requires as input the same reference frame as the *decoder* will see

- De-quantize and inverse-transform the residuals and add them to our predicted frame
- The result is (roughly) the same *reconstructed* frame as the decoder will receive

Residual Transformation

- The pixel difference between the original frame and the reconstructed frame is called residuals

- Since the residuals only express the difference from the prediction, they are much more compact than full pixel values such as in JPEG

- Residuals are transformed using DCT and Quantization

- MPEG uses special Quantization tables for residuals
- in INF5063, we don't (so far)