Why needs Compression?

- Typical NTSC video
  460x340=148000 pels/frame, 30 frames/sec, 16 bits/pel (YIQ)
  =71 Mbps
- Bandwidth are limited resource and expensive
- Storage—enormous for uncompressed video
- Compression processing—relative cheap
- Video Camera, VCR, TV Receiver
- Comparison of NTSC and HDTV

Reference:

"Discrete Cosine Transform Algorithms, Advantages, Application", Rao and Yip, Academic Press, 1990.

Video Compression Applications

- Cable TV distribution
- Interactive communications—videophone, videoconferencing, videotex.
- Digital storage media—CD-ROM, CD-V, digital VTR.
- Network database services video library, video information provider.
- Video-on-demand
- Broadcasting
- Video surveillance

Applications vs. Bit Rates

Application	Standard	Bit Rate Range
Fax Still Image	CCITT G3/G4/JBIG JPEG	4.8-14.4 kbps 56/64 kbps
Videophone (analog) Videophone (digital)	 CCITT H.261 px64	9.6-19.2 kbps 64-128 kbps
Videoconferencing Entertainment video	CCITT H.261 px64 MPEG	320-1920 kbps 1-3 Mbps
Broadcast video Intraframe JPEG	MPEG II JPEG	4-10 Mbps 10-20 Mbps
Digital VCR NTSC (very high quality)	MPEG II CCIR/ANSI	8-20 Mbps 34/45 Mbps
HDTV (broadcasting) HDTV (studio)	FCC(US only)	20-25 Mbps 30-45 Mbps

Why can video data be compressed?

Redundancy

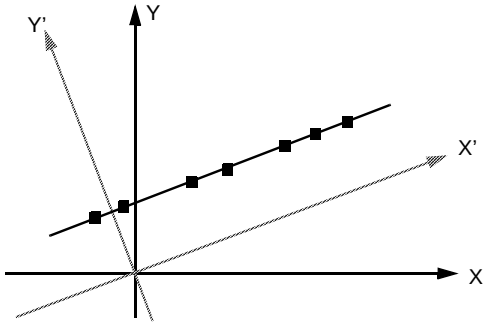
- Intraframe
- Interframe

Compression Techniques

- Spatial domain—DPCM, Transform, Subband, Vector quantization
- Temporal domain—frame difference, motion compensation

Transform Coding Concept

- Why can data be compressed?—Redundancy
- Orthogonal transform:
  —decorrelate data into a more compact form
  so that redundancy can be discarded more easily.



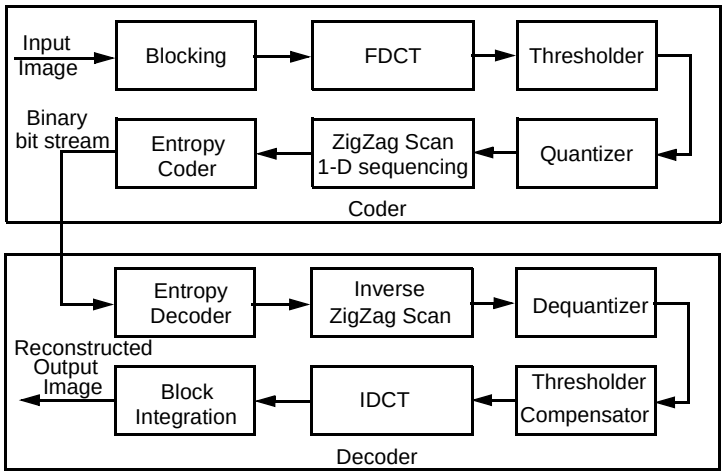
Orthogonal Transforms

- There are many of them: Which is better?

  Karhunen-Loeve Transform (KLT)
  optimal, complete decorrelates the random functions in its transform domain.
  Discrete Fourier Transform (DFT)
  Discrete Slant Transform
  Discrete Cosine Transform (DCT) 1974
  Discrete Sine Transform (DST)
  Discrete Hadamard/Walsh Transform (DHT)
  Discrete Harr Transform

- DCT emerges as the choice of transform for
  —fast algorithms
  —performance (quality)
  —data-independent

A typical Transform Coder/Decoder



2D-Discrete Cosine Transform (DCT)

- 8x8 FDCT/IDCT formula

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

where $C(k) = \frac{1}{\sqrt{2}}$ if $k = 0$, otherwise $C(k) = 1$

- Separability property

$$F(u,v) = \frac{1}{2}C(u)\sum_{x=0}^{7}\left\{\frac{1}{2}C(v)\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\right\}\cos\frac{(2y+1)v\pi}{16}$$

The 2-D FDCT can be implemented by 1-D 8-point FDCT along the rows of $f(x,y)$ followed by 1-D 8-point FDCT along the columns of the matrix obtained after the row transformation.

Example of 2D-DCT

f(x,y)

f(0,0) 0.125	f(1,0) 0.125	f(2,0) 0.125	f(3,0) 0.125	f(4,0) 0.125	f(5,0) 0.125	f(6,0) 0.125	f(7,0) 0.125
f(0,1) 0.125	f(1,1) 0.125	f(2,1) 0.125	f(3,1) 0.125	f(4,1) 0.125	f(5,1) 0.125	f(6,1) 0.125	f(7,1)f 0.125
f(0,2) 0.125	f(1,2) 0.125	f(2,2) 0.125	f(3,2) 0.125	f(4,2) 0.125	f(5,2) 0.125	f(6,2) 0.125	f(7,2)f 0.125
f(0,3) 0.125	f(1,3) 0.125	f(2,3) 0.125	f(3,3) 0.125	f(4,3) 0.125	f(5,3) 0.125	f(6,3) 0.125	f(7,3)f 0.125
f(0,4) 0.125	f(1,4) 0.125	f(2,4) 0.125	f(3,4) 0.125	f(4,4) 0.125	f(5,4) 0.125	f(6,4) 0.125	f(7,4)f 0.125
f(0,5) 0.125	f(1,5) 0.125	f(2,5) 0.125	f(3,5) 0.125	f(4,5) 0.125	f(5,5) 0.125	f(6,5) 0.125	f(7,5)f 0.125
f(0,6) 0.125	f(1,6) 0.125	f(2,6) 0.125	f(3,6) 0.125	f(4,6) 0.125	f(5,6) 0.125	f(6,6) 0.125	f(7,6)f 0.125
f(0,7) 0.125	f(1,7) 0.125	f(2,7) 0.125	f(3,7) 0.125	f(4,7) 0.125	f(5,7) 0.125	f(6,7) 0.125	f(7,7)f 0.125

F(u,v)

F(0,0) 1	F(1,0) 0	F(2,0) 0	F(3,0) 0	F(4,0) 0	F(5,0) 0	F(6,0) 0	F(7,0) 0
F(0,1) 0	F(1,1) 0	F(2,1) 0	F(3,1) 0	F(4,1) 0	F(5,1) 0	F(6,1) 0	F(7,1) 0
F(0,2) 0	F(1,2) 0	F(2,2) 0	F(3,2) 0	F(4,2) 0	F(5,2) 0	F(6,2) 0	F(7,2) 0
F(0,3) 0	F(1,3) 0	F(2,3) 0	F(3,3) 0	F(4,3) 0	F(5,3) 0	F(6,3) 0	F(7,3) 0
F(0,4) 0	F(1,4) 0	F(2,4) 0	F(3,4) 0	F(4,4) 0	F(5,4) 0	F(6,4) 0	F(7,4) 0
F(0,5) 0	F(1,5) 0	F(2,5) 0	F(3,5) 0	F(4,5) 0	F(5,5) 0	F(6,5) 0	F(7,5) 0
F(0,6) 0	F(1,6) 0	F(2,6) 0	F(3,6) 0	F(4,6) 0	F(5,6) 0	F(6,6) 0	F(7,6) 0
F(0,7) 0	F(1,7) 0	F(2,7) 0	F(3,7) 0	F(4,7) 0	F(5,7) 0	F(6,7) 0	F(7,7) 0

$$F(0,0) = \frac{1}{4}C(0)C(0)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)0\pi}{16}\cos\frac{(2y+1)0\pi}{16}$$

F(0,0)=1/4*1/2*(64*0.125)=1 $C(0) = 1/\sqrt{2}$

1D-DCT

- 8-point DCT definition $F(u) = \frac{1}{2}C(u)\sum_{x=0}^{7} f(x)\cos\frac{(2x+1)u\pi}{16}$
- Consider **f** as a vector of n point data and **F** as a vector of transformed values the above transformation can be represent by matrix operations F=cAf
- There are four different 1D-DCTs, the DCT-II was the first to be studied.

Properties of 1D-DCTs

- Only DCT-I and DCT-IV are involutary ($F=F^{-1}$)
- Inverse DCT-II≠DCT-II
- Since Matrix multiplication is a linear operation, all four DCTs retain linearity properties. F[*a***x** + *b***y**] = *a*F**x** + *b*F**y**.
- For properties related to scaling in time, shift in time, differentiation in time, and the convolution, see [DCT9x] by Rao and Yip.

A Fast 1D-DCT Algorithm based on Sparse Matrix Factorization

The 1D-DCT can be represented as $\vec{X}^{C(2)} = \left(\frac{2}{N}\right)^{1/2}[A_N]\vec{x}$

Here we show how A_4 can be decomposed into three sparser matrices.

$C_k^i = \cos[i\pi/k]$

$$[A_4] = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ C_8^1 & C_8^3 & C_8^5 & C_8^7 \\ C_8^2 & C_8^6 & C_8^6 & C_8^2 \\ C_8^3 & C_8^7 & C_8^1 & C_8^5 \end{bmatrix} \xrightarrow{\text{matrix factorization}} [A_4] = \begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ C_4^1 & C_4^3 & C_4^3 & C_4^1 \\ C_8^3 & \text{Đ}C_8^1 & C_8^1 & \text{Đ}C_8^3 \\ C_8^1 & C_8^3 & \text{Đ}C_8^3 & \text{Đ}C_8^1 \end{bmatrix}$$

$$\begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ C_4^1 & C_4^3 & 0 & 0 \\ 0 & 0 & \text{Đ}C_8^1 & C_8^3 \\ 0 & 0 & C_8^3 & C_8^1 \end{bmatrix} \begin{bmatrix} 1&0&0&1 \\ 0&1&1&0 \\ 0&1&\text{Đ}1&0 \\ 1&0&0&\text{Đ}1 \end{bmatrix}$$

$[A_4]x$ requires 16 multiply and 12 add

Here $[A_4]x$ requires only 8 multiply and 8 add

Fast 1D-DCT based on Sparse Matrix Factorization

This matrix operation needs only 2 additions and 2 subtractions.

$$= \begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ C_4^1 & C_4^3 & 0 & 0 \\ 0 & 0 & \text{Đ}C_8^1 & C_8^3 \\ 0 & 0 & C_8^3 & C_8^1 \end{bmatrix} \begin{bmatrix} 1&0&0&1 \\ 0&1&1&0 \\ 0&1&\text{Đ}1&0 \\ 1&0&0&\text{Đ}1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ C_4^1 & C_4^3 & 0 & 0 \\ 0 & 0 & \text{Đ}C_8^1 & C_8^3 \\ 0 & 0 & C_8^3 & C_8^1 \end{bmatrix} \begin{bmatrix} x_1 + \\ x_2 + \\ x_2 \text{Đ} \\ x_1 \text{Đ} \end{bmatrix}$$

This matrix operation needs 8 multiplications, 3 additions, and 1 subtraction.

$$[A_4]x = \begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}}(x_1+x_4)+\frac{1}{\sqrt{2}}(x_2+x_3) \\ C_4^1(x_1+x_4)+C_4^3(x_2+x_3) \\ \text{Đ}C_8^1(x_2 \text{Đ} x_3)+C_8^3(x_1 \text{Đ} x_4) \\ C_8^3(x_2 \text{Đ} x_3)+C_8^1(x_1 \text{Đ} x_4) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}(x_1+x_4)+\frac{1}{\sqrt{2}}(x_2+x_3) \\ C_8^3(x_2 \text{Đ} x_3)+C_8^1(x_1 \text{Đ} x_4) \\ C_4^1(x_1+x_4)+C_4^3(x_2+x_3) \\ \text{Đ}C_8^1(x_2 \text{Đ} x_3)+C_8^3(x_1 \text{Đ} x_4) \end{bmatrix}$$

This matrix operation involves only the shuffling (moving) of data

In total, there are only 8 multiplications, 5 additions, and 3 subtractions.

Mapping Matrices to Signal Flowgraph

- Matrix operations can be mapped to the signal flowgraph representations.
- Signal flowgraphs can be mapped to the VLSI implementation very easily.
- Signal flowgraphs can be mapped to equivalent software implementations.
- Signal flowgraphs make it easy to capture redundant data movement.

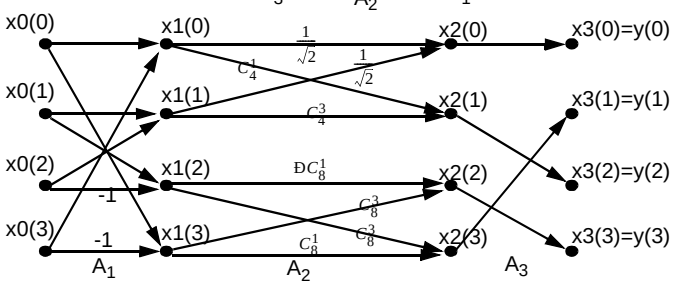
Steps: Map $\mathbf{y}=A_NA_2A_1\mathbf{x}$ to the equivalent signal flowgraph (A_i is a NxN matrix)

1. Draw N nodes top-down on the left hand side of a drawing area, label them x0(0) to x0(N-1). These are original values of input vector **x**.
2. To the left of x0, draw N nodes and label them x1(0) to x1(N-1). **x1** is the result of $A_1\mathbf{x}$.
3. Examine values in A_1. For each of the x0 elements, say x0(i), that contributes to x1(j) draw an arrow from x0(i) to x1(j) with its A1 coefficient as label. E.g., assume that $x1(i)=A_1(i,1)x0(1)+A_1(i,3)x0(3)$ where $A_1(i,1)$ and $A_1(i,3)$ are the only non-negative elements in row $A_1(i)$. Draw an arrow with label $A_1(i,1)$ from x0(1) to x1(i). Draw an arrow with label $A_1(i,3)$ from x0(3) to x1(i). These arrows represent the "flows" of signal from left to right. Coefficient of 1 is not labelled.
4. Repeat Steps 2 and 3 for (A_2, **x2**), (A_3, **x3**),..., and (A_N, **xN**).
5. **xN** is the same as **y**.

Example: Mapping matrices to signal flowgraph

$$[A]x = \begin{bmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ C_4^1 & C_4^3 & 0 & 0 \\ 0 & 0 & \text{Đ}C_8^1 & C_8^3 \\ 0 & 0 & C_8^3 & C_8^1 \end{bmatrix} \begin{bmatrix} 1&0&0&1 \\ 0&1&1&0 \\ 0&1&\text{Đ}1&0 \\ 1&0&0&\text{Đ}1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

A_3 A_2 A_1



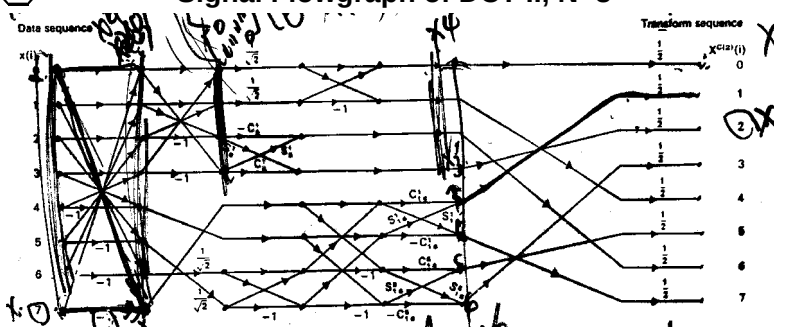
Algorithm for DCT-II, N=8 and its signal flowgraph

Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," IEEE Trans. Accoust., Speech, and Signal Process., vol. ASSP-32, pp.8x3-816, Aug. 1984.

$$C_8^{II} = \frac{1}{2}\begin{bmatrix} 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\times \begin{bmatrix} \begin{bmatrix} 1&0&0&0\\ 0&0&0&1\\ 0&1&0&0\\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}}\begin{bmatrix} 1&1\\ 1&-1 \end{bmatrix} & 0_2 \\ 0_2 & \begin{bmatrix} -C_8^1 & S_8^1\\ S_8^1 & C_8^1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} I_2 & \bar{I}_2\\ \bar{I}_2 & -I_2 \end{bmatrix} & 0_4 \\ 0_4 & \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{bmatrix} \begin{bmatrix} C_{16}^1 & S_{16}^1 & 0_2 \\ S_{16}^1 & -C_{16}^1 & \\ 0_2 & & \begin{matrix} C_{16}^3 & S_{16}^3\\ S_{16}^3 & -C_{16}^3 \end{matrix} \end{bmatrix} \begin{bmatrix} I_2 & I_2\\ I_2 & -I_2 \end{bmatrix} \begin{bmatrix} [I_2] & 0_2 \\ 0_2 & \frac{1}{\sqrt{2}}\begin{bmatrix} 1&1\\ 1&-1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0\\ 0&1&0&0 \end{bmatrix} \end{bmatrix} \times \begin{bmatrix} I_4 & \bar{I}_4\\ \bar{I}_4 & -I_4 \end{bmatrix} \quad \text{(A.2.3)}$$

Signal Flowgraph of DCT-II, N=8

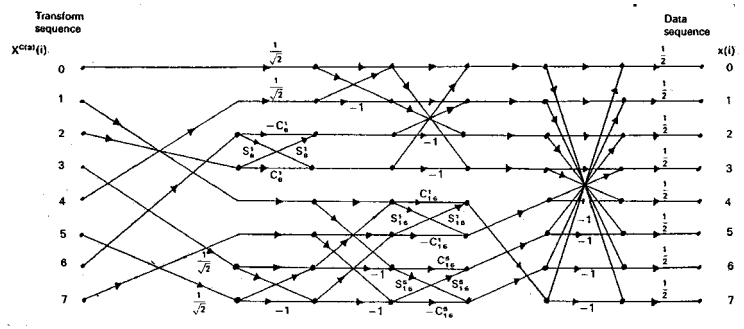


IDCT-II, N=8 and its signal flowgraph

$$C_{16}^{II} = \frac{1}{2\sqrt{2}}\begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0 \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} 1&0&0&0\\ 0&0&0&1\\ 0&1&0&0\\ 0&0&1&0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}}\begin{bmatrix} 1&1\\ 1&-1 \end{bmatrix} & 0_2 \\ 0_2 & \begin{bmatrix} -C_8^1 & S_8^1\\ S_8^1 & C_8^1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} I_2 & \bar{I}_2\\ \bar{I}_2 & -I_2 \end{bmatrix} & 0_4 \\ 0_4 & \begin{bmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&0&0&0 \end{bmatrix} \begin{bmatrix} C_{16}^1 & S_{16}^1 & 0_2 \\ S_{16}^1 & -C_{16}^1 & \\ 0_2 & & \begin{matrix} C_{16}^3 & S_{16}^3\\ S_{16}^3 & -C_{16}^3 \end{matrix} \end{bmatrix} \begin{bmatrix} I_2 & I_2\\ I_2 & -I_2 \end{bmatrix} \begin{bmatrix} [I_2] & 0_2 \\ 0_2 & \frac{1}{\sqrt{2}}\begin{bmatrix} 1&1\\ 1&-1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} 0&0&0&1\\ 1&0&0&0\\ 0&0&1&0\\ 0&1&0&0 \end{bmatrix} \end{bmatrix}$$

Signal Flowgraph of IDCT-II, N=8



Comparison of Fast DCT-II Algorithms

Algorithm	Complexity	Advantages	Disadvantages
Via N-point FFT [FF-1] (Section 4.2)	$\frac{1}{2}(N \log N - N + 2)$ complex multiplications	easy to implement using standard FFT routines	slow
Recur. FFT [FF-1, 2, 4] (Section 4.2, 4.6)	$\mu = \frac{1}{2}(N \log N)$ $\alpha = (3 \log N - N + 1)/2$	fast and recursive	complex index mapping
Sparse factors [FRA-1, 2] (Section 4.3)	$\mu = N \log N - 3N/2 + 4$ $\alpha = 3N(\log N - 1)/2 + 2$	reasonably fast	complex index mapping, nonrecursive
DIT [FRA-8] (Section 4.4)	$\mu = (N \log N)/2 + N/4$ $\alpha = (3N/2 - 1) \log N + N/4 - 1$	fast, recursive decimation in time	moderately complex index mapping
DIF [FRA-9] [FRA-14] (Section 4.4)	$\mu = (N \log N)/2$ $\alpha = (3N/2) \log N - N + 1$	fast, recursive decimation in frequency	moderately complex index mapping
Via WHT [DW-1] (Section 4.5)	dependent on WHT algorithm used	readily implementable, simple index mapping	conversion matrix needed, slow for $N \geq 16$
Via DHT [FRA-16] [FRA-19, 21] (Section 4.5)	same as using FFT	implementable using standard FFT routines	slow
PFA [FRA-10] (Section 4.6)	not applicable	not restricted to radix-2	very complex index mapping
Multiplexed [FRA-13] (Section 4.6)	$\mu = (N - 1)^*$ $\alpha = (N^2 + N - 7/3)^*$	fast, recursive simple index map	slower without multiplexing, shift required
Rotators [DP-22] (Section 4.6)	$(N \log N)/2$ rotations	single processor unit, good structure, simple index map	slow

μ = number of real multiplications, α = number of real additions; log is base 2.
*numbers without multiplexing are $\mu = (N \log N)/2$ and $\alpha = (3N/2)\log N + N^2/2 - N$.

Exercises

- Develop the computer program which implements the 1D-DCT-II in Page 1x with straight and compile the Fortran program given in the handout which implements Wang's Fast DCT-II and IDCT-II.
- Generate some random data to test and compare the performance of these two DCT-II implementations.
- Compare the original data with the reconstructed data generated by applying Wang's IDCT-II on the transformed data generated by applying Wang's DCT-II on the original data, and discuss the impact of the computation errors introduced by round-off and truncation).
- You can work as a group for this assignment.