Radix 2, Decimation-In-Time (DIT)

- Input order “decimated” — needs bit reversal
- Output in order
- Butterfly

\[ X = A + BW \]
\[ Y = A - BW \]

Radix 2, Decimation In Frequency (DIF)

- Input in order
- Output “decimated” — needs bit reversal
- Butterfly
  - Two CPAs
  - Wider multiplier

\[ X = A + B \]
\[ Y = (A - B) W \]
Radix 4, DIT Butterfly

- Decimation in Time (DIT) or Decimation in Frequency (DIF)

A B C D

V W X Y

Higher Radices

- Radix 2 and radix 4 are certainly the most popular
- Radix 4 is on the order of 20% more efficient than radix 2 for large transforms
- Radix 8 is sometimes used, but longer radix butterflies are not common because additional efficiencies are small and added complexity is non-trivial (especially for hardware implementations)
Common-Factor FFTs

• Key characteristics
  – Most common class of FFTs
  – Also called Cooley-Tukey FFTs
  – Factors of \( N \) used in decomposition have common factor(s)

• Radix-\( r \)
  – \( N = r^k \), where \( k \) is a positive integer
  – Butterflies used in each stage are the same
  – Radix-\( r \) butterflies are used
  – \( N/r \) butterflies per stage
  – \( k = \log_r N \) stages

Common-Factor FFTs

• Mixed-radix
  – Radices of component butterflies are not all equal
  – More complex than radix-\( r \)
  – Is necessary if \( N \neq r^k \)
  – Example
    • \( N = 32 \)
    • Could calculate with two radix-4 stages and one radix-2 stage
Prime-Factor FFTs

• The length of transforms must be the product of relatively prime numbers
• This can be limiting, though it is often possible to find lengths near popular power-of-2 lengths (e.g., $7 \times 11 \times 13 = 1003$)
• Their great advantage is that they have no $W_N$ twiddle factor multiplications
• Another large disadvantage is their irregular sorting of input and output data
• Irregular addressing for butterflies

Other FFTs

• Split-radix FFT
  – When $N = p^k$, where $p$ is a small prime number and $k$ is a positive integer, this method can be more efficient than standard radix-$p$ FFTs

• Winograd Fourier Transform Algorithm (WFTA)
  – Type of prime factor algorithm based on DFT building blocks using a highly efficient convolution algorithm
  – Requires many additions but only order $N$ multiplications
  – Has one of the most complex and irregular structures

• FFTW (www.fftw.org)
  – C subroutine libraries highly tuned for specific architectures
Other FFTs

• Goertzel DFT
  – Not a “normal” FFT in that its computational complexity is still order $N^2$
  – It allows a subset of the DFT’s $N$ output terms to be efficiently calculated

Bit-Reversed Addressing

• Normally:
  – DIT: bit-reverse inputs before processing
  – DIF: bit-reverse outputs after processing
• Reverse addressing bits for read/write of data
  – 000 (0) $\rightarrow$ 000 (0) # Word 0 does not move
  – 001 (1) $\rightarrow$ 100 (4) # Original word 1 goes to location 4
  – 010 (2) $\rightarrow$ 010 (2) # Word 2 does not move
  – 011 (3) $\rightarrow$ 110 (6) # Original word 3 goes to location 6
  – 100 (4) $\rightarrow$ 001 (1) # …
  – 101 (5) $\rightarrow$ 101 (5)
  – 110 (6) $\rightarrow$ 011 (3)
  – 111 (7) $\rightarrow$ 111 (7)
Addressing In Matlab
(Especially helpful for FFTs)

- Matlab
  - Matlab can not index arrays with index zero!
- In matlab, do address calculations normally
  - \( AddrA = 0, 2, 4, \ldots \)
  - \( AddrB = 1, 3, 5, \ldots \)
- then use pointers with an offset of one whenever indexing arrays
  - \( AddrA = \ldots; \)
  - \( AddrB = \ldots; \)
  - \( A = \text{data}(AddrA+1); \)
  - \( B = \text{data}(AddrB+1); \)
  - \( \text{data}(AddrA+1) = X; \)
  - \( \text{data}(AddrB+1) = Y; \)

Signal Growth

- Note in DFT equation signal can grow by \( N \) times
- This is also seen in the FFT in its growth by \( r \) times in a radix-\( r \) butterfly, and \( \log_r N \) stages in the entire transform: \( r^\left(\log_r N\right) = N \)
- Thus, the FFT processor requires careful scaling
  - Floating point number representation
    - Easiest conceptually, but expensive hardware. Typically not used in efficient DSP systems.
  - Fixed-point with scaling by \( 1/r \) every stage
    - First stage is a special case. Scaling must be done on the inputs before processing to avoid overflow with large magnitude complex inputs with certain phases.
  - Block floating point