PIPELINE IMPLEMENTATIONS OF THE FAST FOURIER TRANSFORM (FFT)

- Consulted work:

  Chiueh, T.D. and P.Y. Tsai, *OFDM Baseband Receiver Design for Wireless Communications*, John Wiley and Sons Asia, (2007).
  Second edition of 2012 available as e-book via UT Library.

TOPICS

- Discrete Fourier Transform
- Fast Fourier Transform
  - Decimation in time
  - Decimation in frequency
- FFT pipelines:
  - Radix-2 multi-path delay commutator
  - Radix-2 single-path delay feedback
  - Delay buffer implementation
  - Radix-4 algorithms

DISCRETE FOURIER TRANSFORM (DFT)

- Consider a block of *N* samples of a (possibly complex-valued) data stream:

$$x[n], n = 0, 1, \ldots N-1$$

- The *discrete Fourier transform* of this block is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi k}{N}n}, k = 0, 1, \ldots N-1$$

INVERSE DFT (IDFT)

- The inverse DFT is almost the same computation as the DFT:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi n}{N}k}, n = 0, 1, \ldots N-1$$

TWIDDLE FACTORS

- Define: $W_N = e^{-j\frac{2\pi}{N}}$
- Then DFT becomes: $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$
- W_N^{kn} is called a *twiddle factor* (it is a number on the unit circle in the complex plane).
- Multiplying with a twiddle factor is a *vector rotation*. How to implement?

MATRIX REPRESENTATION OF DFT

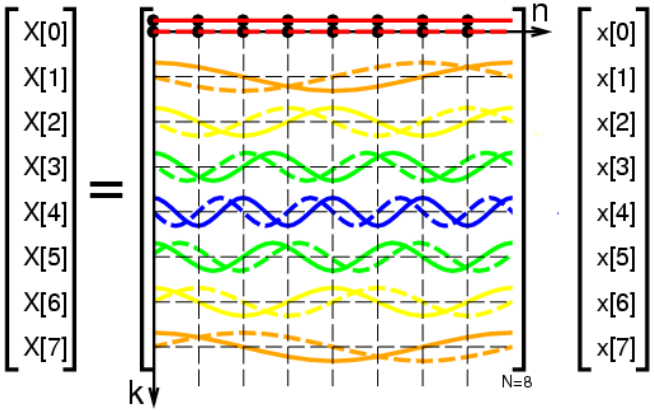
- The DFT can be expressed in matrix form:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W_N & W_N^2 & \dots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \dots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix} \mathbf{x}$$

- Number of *complex multiplications* involved (including trivial ones with 1, *j*, etc.): N^2

DFT VISUALIZATION

- The DFT basically matches frequencies.
- The following picture originates from *Wikipedia* (entry: "DFT matrix"):



FAST FOURIER TRANSFORM

- Reduces the number of calculations in a DFT from $O(N^2)$ to $O(N \log N)$.
- First published by Cooley and Tukey in 1965.
- Check e.g. Wikipedia page for historical background dating back to Gauss in 1805 (earlier than Fourier!).
- Two variants:
  - Decimation in time
  - Decimation in frequency.

DECIMATION-IN-TIME FFT (1)

- Split odd and even terms in DFT definition:

$$X[k] = \sum_{m=0}^{N/2-1} x[2m] W_N^{2mk} + \sum_{m=0}^{N/2-1} x[2m+1] W_N^{(2m+1)k}$$
$$k = 0, 1, \ldots N-1$$

DECIMATION-IN-TIME FFT (2)

- Consider first half of outputs: $k = 0, 1, \ldots \frac{N}{2} - 1$
- Rewrite, using $W_N^{2mk} = W_{N/2}^{mk}$:

$$X[k] = \sum_{m=0}^{N/2-1} x[2m] W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} x[2m+1] W_{N/2}^{mk}$$

DECIMATION-IN-TIME FFT (3)

- DFT has been expressed in terms of half-size DFTs:

$$X[k] = \sum_{m=0}^{N/2-1} x[2m] W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} x[2m+1] W_{N/2}^{mk}$$

DFT of even samples

DFT of odd samples

DECIMATION-IN-TIME FFT (4)

- Now consider second half of outputs:

$$X\left[k + \frac{N}{2}\right] = \sum_{m=0}^{N/2-1} x[2m] W_{N/2}^{m(k+N/2)} + W_N^{k+N/2} \sum_{m=0}^{N/2-1} x[2m+1] W_{N/2}^{m(k+N/2)}$$
$$k = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-TIME FFT (5)

- Make use of the following identities:

$$W_{N/2}^{m(k+N/2)} = W_{N/2}^{mN/2} W_{N/2}^{mk} = W_{N/2}^{mk}$$

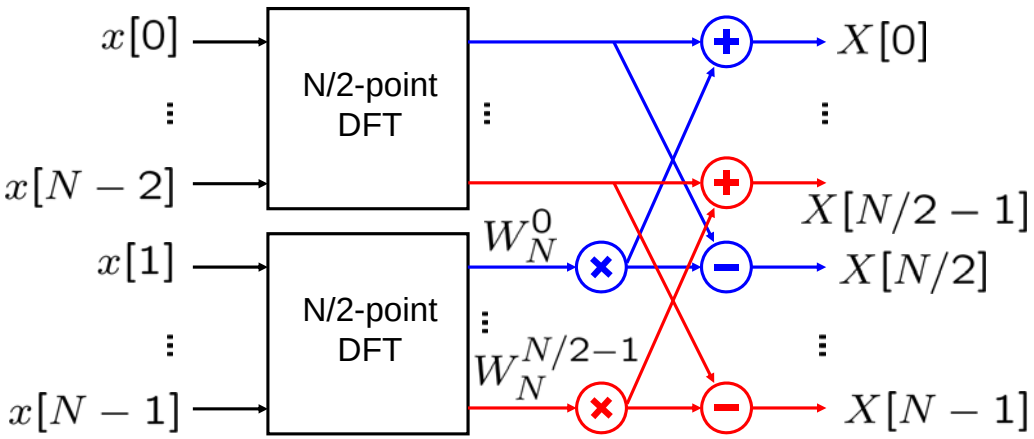
$$W_N^{k+N/2} = W_N^{N/2} W_N^k = -W_N^k$$

DECIMATION-IN-TIME FFT (6)

- The expression for the second half becomes the same as for the first except for a minus sign:

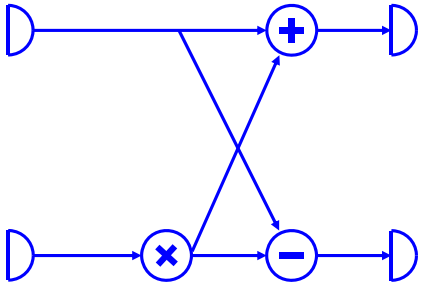
$$X[k+\frac{N}{2}] = \sum_{m=0}^{N/2-1} x[2m] W_{N/2}^{mk} + -W_N^k \sum_{m=0}^{N/2-1} x[2m+1] W_{N/2}^{mk}$$

DECIMATION-IN-TIME FFT (7)



DECIMATION-IN-TIME BUTTERFLY

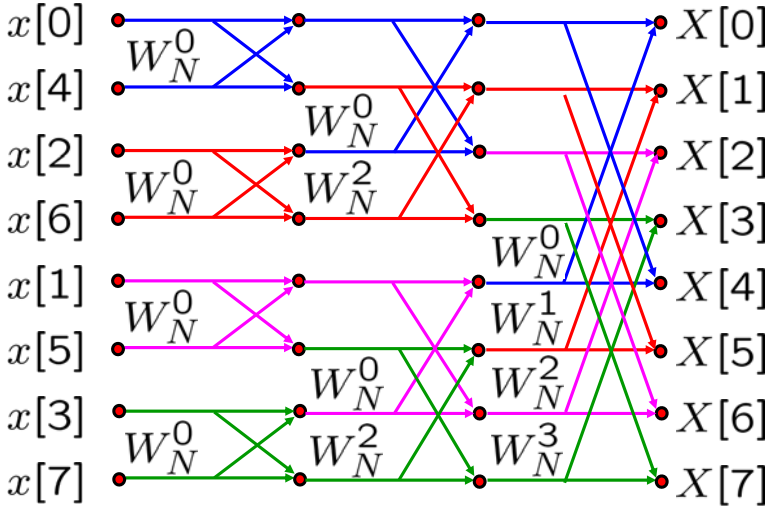
- elementary FFT operation with two inputs and two outputs
- consisting of one complex *multiplication*, one complex *addition* and one complex *subtraction*:



DECIMATION-IN-TIME FFT (8)

- One *decimation-in-time* step defines an *N-point* DFT in terms of two *N/2-point* DFTs. They are combined by means of butterflies.
- Applying the principle *recursively*, results in a computation that consists of butterflies only.

8-POINT DECIMATION-IN-TIME FFT



COMPLEXITY REDUCTION

- Number of computations has been reduced:
  - From $\mathcal{O}(N^2)$ for the DFT
  - To $\mathcal{O}(N \log N)$ for the FFT

IMPLEMENTATION ISSUES

- *Buffering* of input is needed because of block-based nature.
- One could e.g. use a *ping-pong memory:*
  - While the input is filling one memory, the FFT could consume samples of the other memory.
  - After processing one block of samples, the memories change roles.
- Note the *bit reversal* of the addresses:
  - Address order can be found from increasing binary addresses read in reverse order $4 = 100_2$ e.g. becomes $1 = 001_2$.

DECIMATION-IN-FREQUENCY FFT (1)

- Split input block in first and second half and consider the outputs with even index:

$$X[2m] = \sum_{n=0}^{N/2-1} x[n]W_N^{2mn} + \sum_{n=N/2}^{N-1} x[n]W_N^{2mn}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-FREQUENCY FFT (2)

- Shift index in second sum:

$$X[2m] = \sum_{n=0}^{N/2-1} x[n]W_N^{2mn} + \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{2m(n+N/2)}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-FREQUENCY FFT (3)

- Using simplification rules mentioned earlier:

$$X[2m] = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2])W_{N/2}^{mn}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-FREQUENCY FFT (4)

- This is a half-size DFT applied to an input stream consisting of pairs of the original input stream:

$$X[2m] = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/2])W_{N/2}^{mn}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-FREQUENCY FFT (5)

- Now consider the outputs with odd index:

$$X[2m+1] = \sum_{n=0}^{N/2-1} x[n] W_N^{n(2m+1)} + \sum_{n=0}^{N/2-1} x[n+N/2] W_N^{(2m+1)(n+N/2)}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

DECIMATION-IN-FREQUENCY FFT (6)

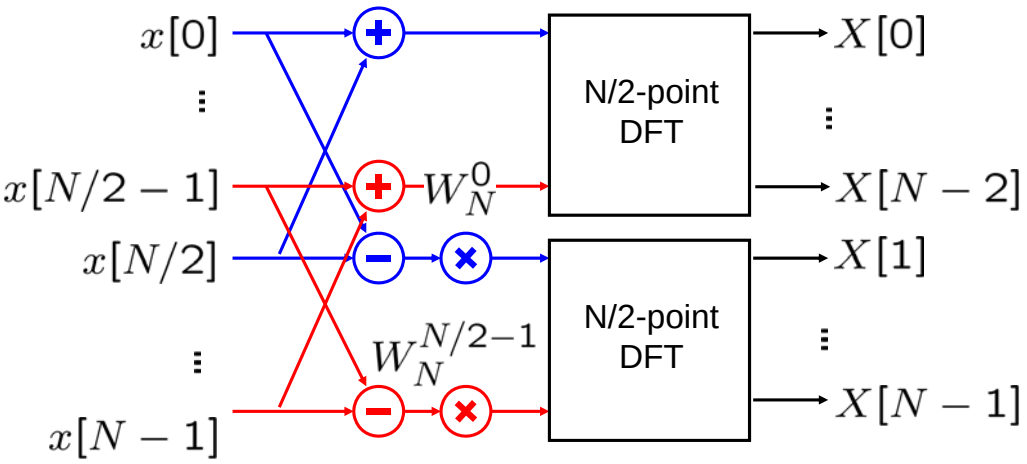
- With the usual type of simplifications:

$$X[2m+1] = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2])\, W_N^n W_{N/2}^{mn}$$

$$m = 0, 1, \ldots \frac{N}{2} - 1$$

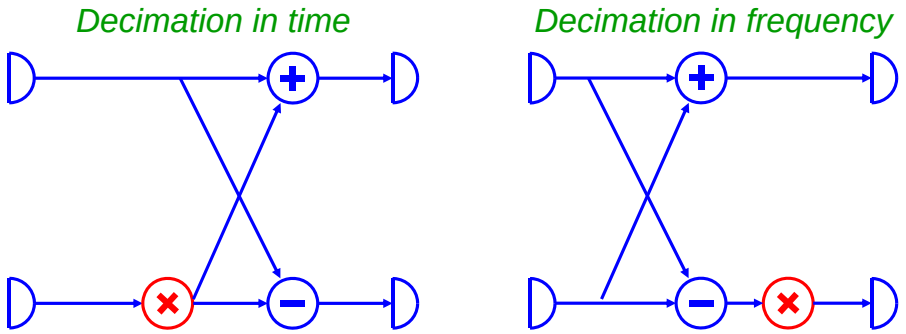
- This is a half-size DFT applied on a sequence obtained by taking the difference of pairs of the input and multiplying them with factor W_N^n .

DECIMATION-IN-FREQUENCY FFT (7)

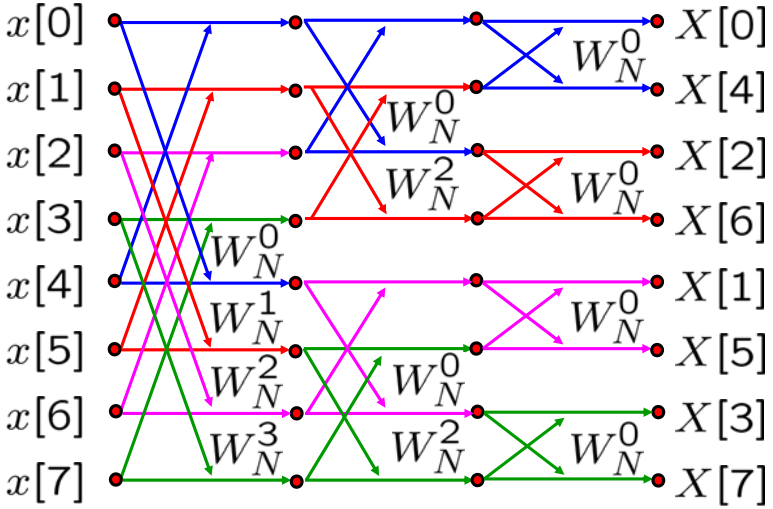


DECIMATION-IN-FREQ. BUTTERFLY

- Similar to the *decimation-in-time* butterfly, but location of multiplication is now at the output side.

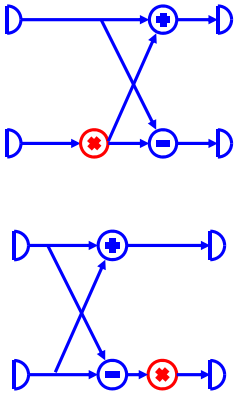


8-POINT DECIMATION-IN-FREQ. FFT



SUMMARY FFT BASICS

- *Decimation-in-time* (DIT) FFT:
  - "Divide and conquer" approach to DFT, based on grouping even and odd *inputs* in DFT definition
  - Butterfly: multiply before add/subtract
- *Decimation-in-frequency* (DIF) FFT:
  - "Divide and conquer" approach based on grouping even and odd outputs
  - Butterfly: multiply after add/subtract
- Both are of *radix-2* type: problem size is reduced by 2 at each stage



FFT IMPLEMENTATIONS

- FFTs can be realized on all kind of platforms:
  - Programmable processors
  - Dedicated hardware
- Here attention is paid to one type of implementation, viz. the *FFT pipelines*. An FFT pipeline transforms a computation with a *block processing* nature into one of a *streaming* nature.
- Issues of concern:
  - Area
  - Speed
  - Power
  - Memory access

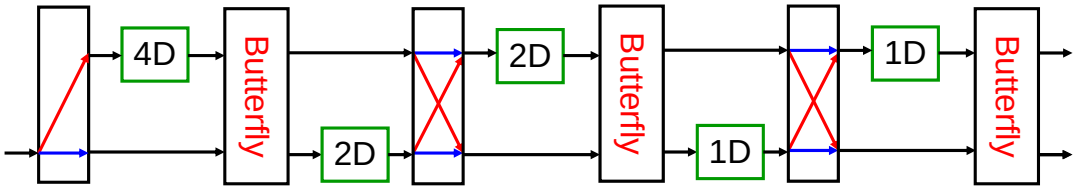
RADIX-2 MULTI-PATH DELAY COMMUTATOR (R2MDC)

- Pipeline solution:
  - "Stream-like" processing of block-based algorithm.
- Examples based on 8-point FFT:
  - solutions scale for higher powers of 2.
- Originally proposed in:

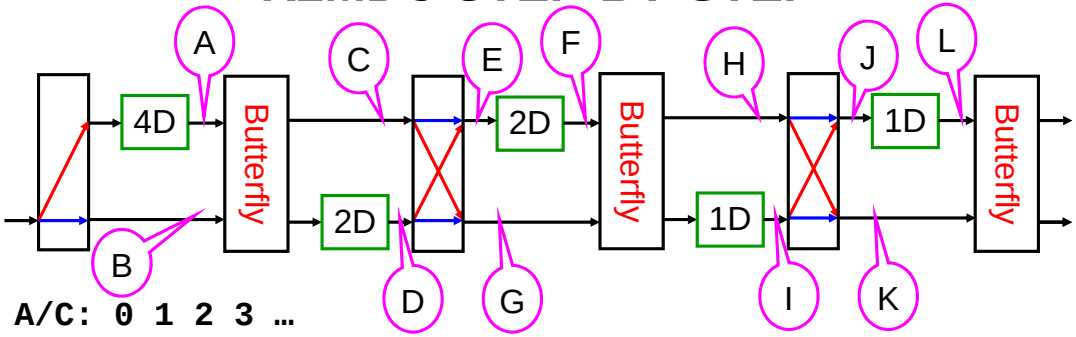
  Rabiner, L.R. and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, (1975).

R2MDC FOR 8-POINT FFT

- Consists of
  - Commutators, switches that send the data either straight ahead or crisscross
  - Butterfly blocks (either for DIT or DIF)
  - Delay buffers



R2MDC STEP-BY-STEP



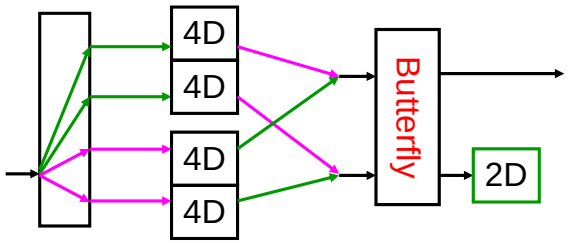

```
A/C: 0 1 2 3 …
  B: 4 5 6 7
  D: . . 4 5 6 7
  E: 0 1 4 5
  F: . . 0 1 4 5
  G: . . 2 3 6 7
```

```
  H: . . 0 1 4 5 …
  I: . . . 2 3 6 7
  J: . . 0 2 4 6
  L: . . . 0 2 4 6
  K: . . . 1 3 5 7
```


R2MDC EFFICIENCY

- If implemented as presented, the pipeline has a 50% hardware utilization:
  - The first butterfly is idle half of the time until it can process new pairs of inputs.
  - All hardware operates at input sample frequency
- Situation can be easily improved by duplicating input buffer and operating it in ping-pong fashion:
  - Hardware utilization jumps to 100%.
  - All hardware operates at half the sample frequency.
- As there are no feedback paths, hardware can be further pipelined to cope with possibly long computational delays.

R2MDC WITH PING-PONG INPUT BUFFER



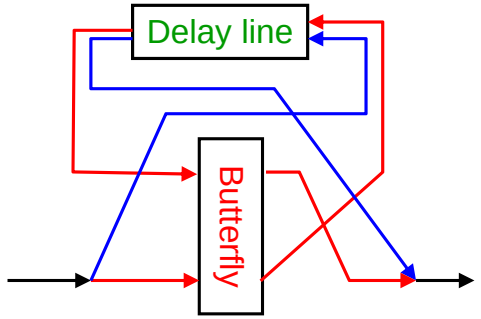
RADIX-2 SINGLE-PATH DELAY FEEDBACK (R2SDF)

- Alternative pipeline solution, with optimized memory size
- Originally proposed in:

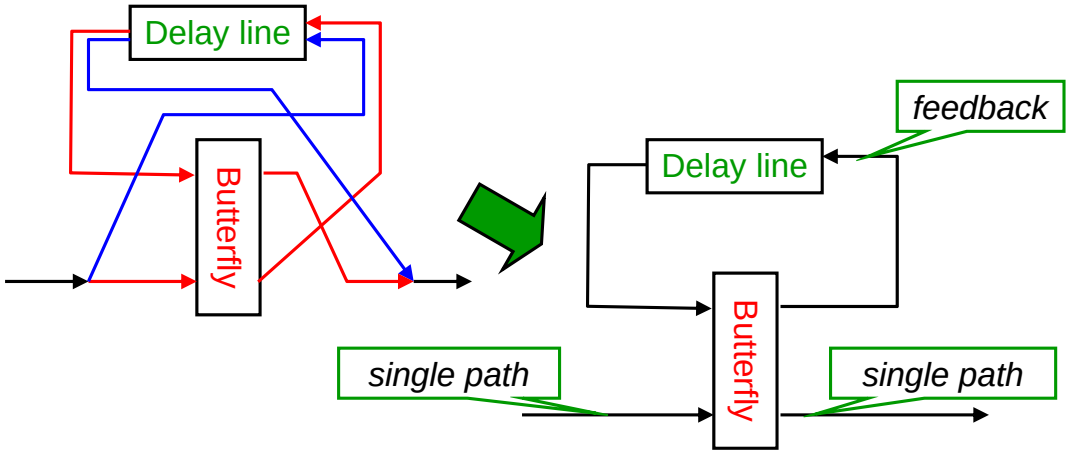
  Groginsky, H.L. and G.A. Works, *A Pipeline Fast Fourier Transform*, IEEE Transactions on Computers, Vol.C-19(11), pp. 1015-1019, (November 1970).

R2SDF ELEMENT

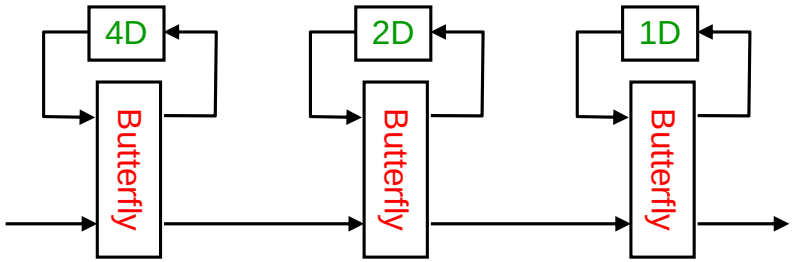
- Element:
  - Either shifts first half of input in delay buffer and second half of output out of delay buffer (blue);
  - Or shifts second half of input into butterfly together with first half from delay buffer, first half of output to next stage and second half output into delay buffer (red).



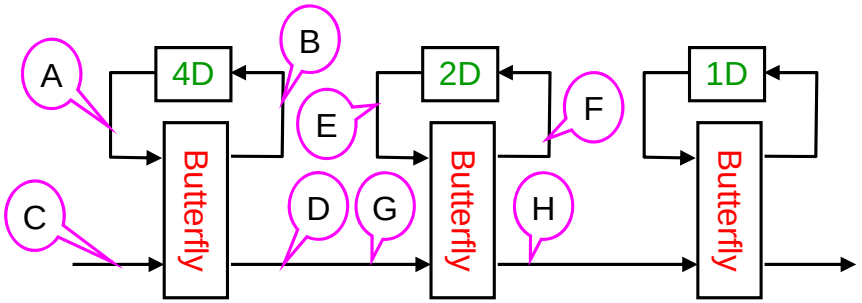
SYMBOLILC REPRESENTATION OF R2SDF ELEMENT



R2SDF FOR 8-POINT FFT



R2SDF STEP-BY-STEP




```
A: . . . . 0 1 2 3 4
B: 0 1 2 3 4 5 6 7
C: 0 1 2 3 4 5 6 7
D: . . . . 0 1 2 3 4 5 6 7
```

```
E: . . . . . . . 0 1 2 3 4 5 6 7
F: . . . . 0 1 2 3 4 5 6 7
G: . . . . 0 1 2 3 4 5 6 7
H: . . . . . . . 0 1 2 3 4 5 6 7
```


R2SDF EFFICIENCY

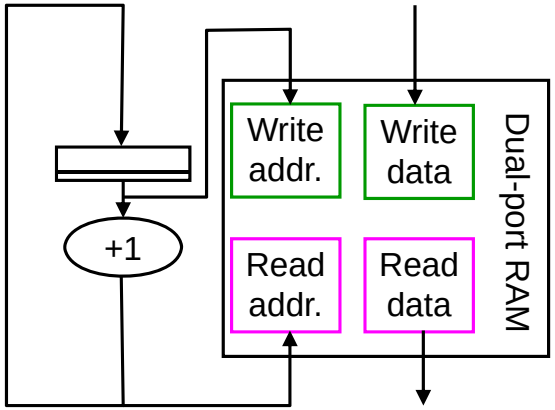
- It needs to operate at same speed as input sample rate.
- Hardware utilization is 50%.
- Number of delay elements for N-point FFT:
  - R2SDF: $N-1$
  - R2MDC: $\frac{3}{2}N-2$
- R2SDF and R2MDC have the same number of (complex) adders and multipliers.

DELAY-BUFFER IMPLEMENTATION

- The straightforward implementation of a delay buffer in hardware would be a shift register.
- Such an implementation is not convenient (think of e.g. a 1024-point FFT):
  - Memory elements implemented by D-flipflops are large!
  - Shifting all data at every clock cycle consumes a lot of energy!
- Much better idea to keep the data in RAM and shift the address pointer(s) instead.

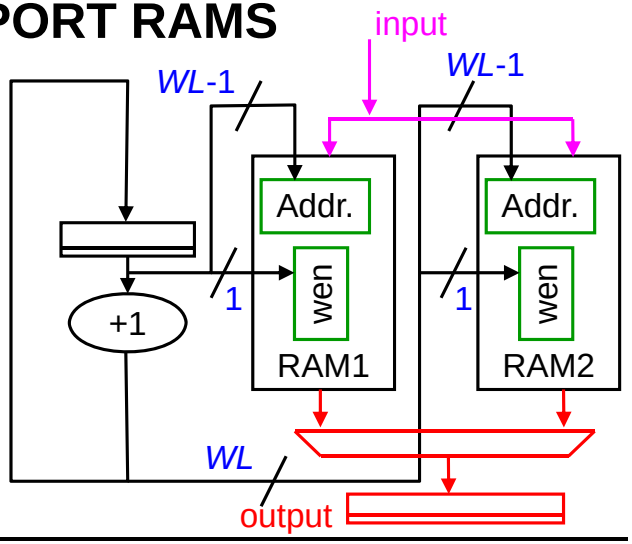
DUAL-PORT RAM DELAY BUFFER

- Dual-port RAM:
  - One *read port* with its own data and address
  - One *write port* with its own data and address
- Cyclic buffer: *L+1* locations to store *L* items.



DELAY BUFFER WITH TWO SINGLE-PORT RAMS

- Idea is to alternatively read from and write to the two RAMs.
- Two single-port RAMs are cheaper than one dual-port RAM.
- Use LSB of address to connect to “write enable” (wen).



SPECIAL-PURPOSE RAMS

- Replace address decoder of RAM by single-bit shift register.
- Only one D-flipflop has output 1.
- Not much switching activity in shift-register (no waste of power).

RADIX-4 FFT

- Has both “decimation in time” as “decimation in frequency” variants.
- Idea is to express DFT as the combination of 4 DFTs whose sizes are one fourth of the original DFT.
- Takes advantage of the following symmetries:

$$W_N^{nk+N/4} = -W_N^{nk+3N/4} = -jW_N^{nk}$$

- This leads to a reduction of the number of multipliers (multiplication by *j* can be performed without multiplier).

RADIX-4 DIF (1)

$$X[4m] = \sum_{n=0}^{N/2-1} (x[n] + x[n+N/4] + x[n+N/2] + x[n+3N/4])W_{N/4}^{mn}$$

$$X[4m+1] = \sum_{n=0}^{N/2-1} (x[n] - jx[n+N/4] - x[n+N/2] + jx[n+3N/4])W_{N/4}^{mn}W_N^n$$

$$m = 0, 1, \ldots \frac{N}{4} - 1$$

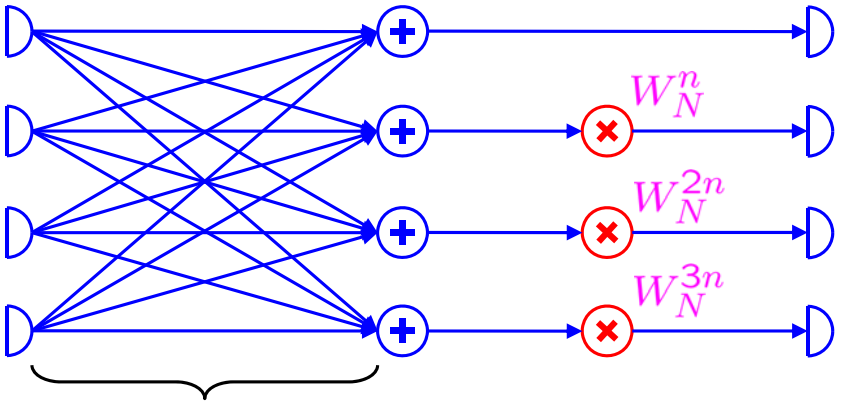
RADIX-4 DIF (2)

$$X[4m+2] = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/4] + x[n+N/2] - x[n+3N/4]) W_{N/4}^{mn} W_N^{2n}$$

$$X[4m+3] = \sum_{n=0}^{N/2-1} (x[n] + jx[n+N/4] - x[n+N/2] - jx[n+3N/4]) W_{N/4}^{mn} W_N^{3n}$$

$$m = 0, 1, \ldots \frac{N}{4} - 1$$

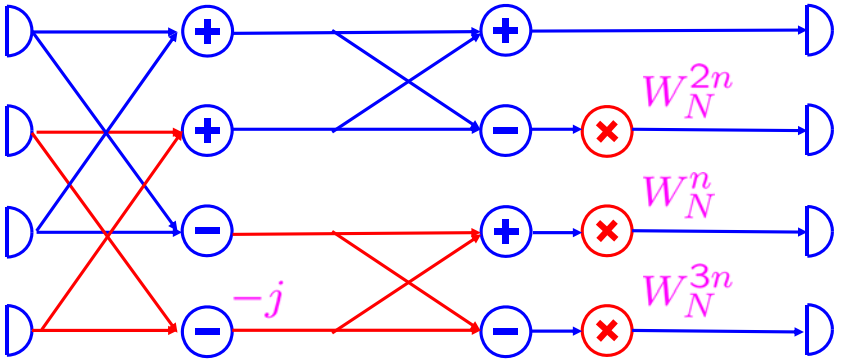
RADIX-4 DIF BUTTERFLY



MULTIPLIER & ADDER COMPLEXITY

- Radix-2 decomposition:
  - Number of stages: $\log_2 N$
  - *Complex* multiplications per stage: $N/2$ in total: $N/2 \log_2 N$
  - Complex additions per stage: N in total: $N \log_2 N$
- Radix-4 decomposition:
  - Number of stages: $\log_4 N = 1/2 \log_2 N$
  - Complex multiplications per stage: $3N/4$ in total: $3N/8 \log_2 N$
  - Complex additions per stage: $3N$ in total: $3N/2 \log_2 N$

RADIX-2^2 BUTTERFLY

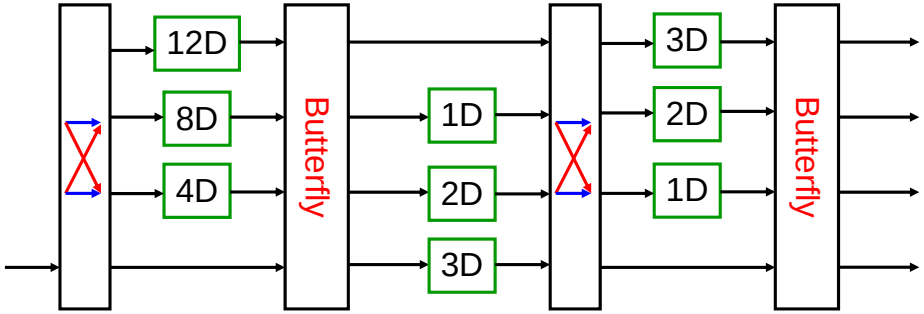


RADIX-2^2 BUTTERFLY EVALUATION

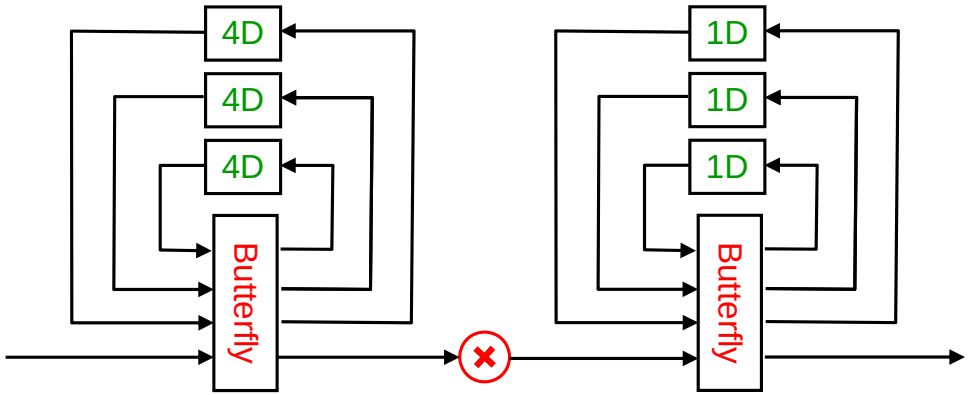
- Radix-2^2 has the same number of multipliers as radix-4.
- It has fewer additions: $N \log_2 N$

- In a similar way, a radix-8 butterfly can be decomposed in a radix-2^3 butterfly.

RADIX-4 MULTI-PATH DELAY COMMUTATOR (R4MDC) PIPELINE FOR 16-POINT FFT



RADIX-4 SINGLE-PATH DELAY FEEDBACK (R4SDF) PIPELINE FOR 16-POINT FFT



EVALUATION RADIX-4 PIPELINES

- The multi-path structure:
  - Has a high memory overhead;
  - Has 100% multiplier utilization;
  - Can operate at one quarter of sample frequency.

- The single-path structure:
  - Has minimal memory;
  - Has a 75% multiplier utilization (multiplier drawn outside butterfly!)
  - Operates at sample frequency.