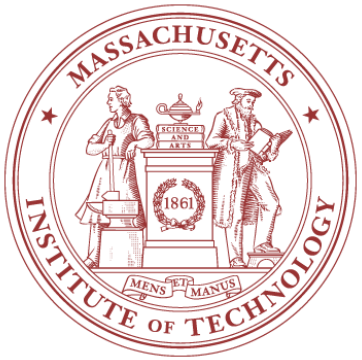
Fast Fourier Transform: VLSI Architectures

Lecture 10

Vladimir Stojanović



6.973 Communication System Design – Spring 2006

Massachusetts Institute of Technology

Pipelined FFT architectures

- Examples
- Radix-2
  - multi-path delay commutator
  - single-path delay feedback
- Radix-4
  - single-path delay feedback
  - multi-path delay commutator
  - single-path delay commutator

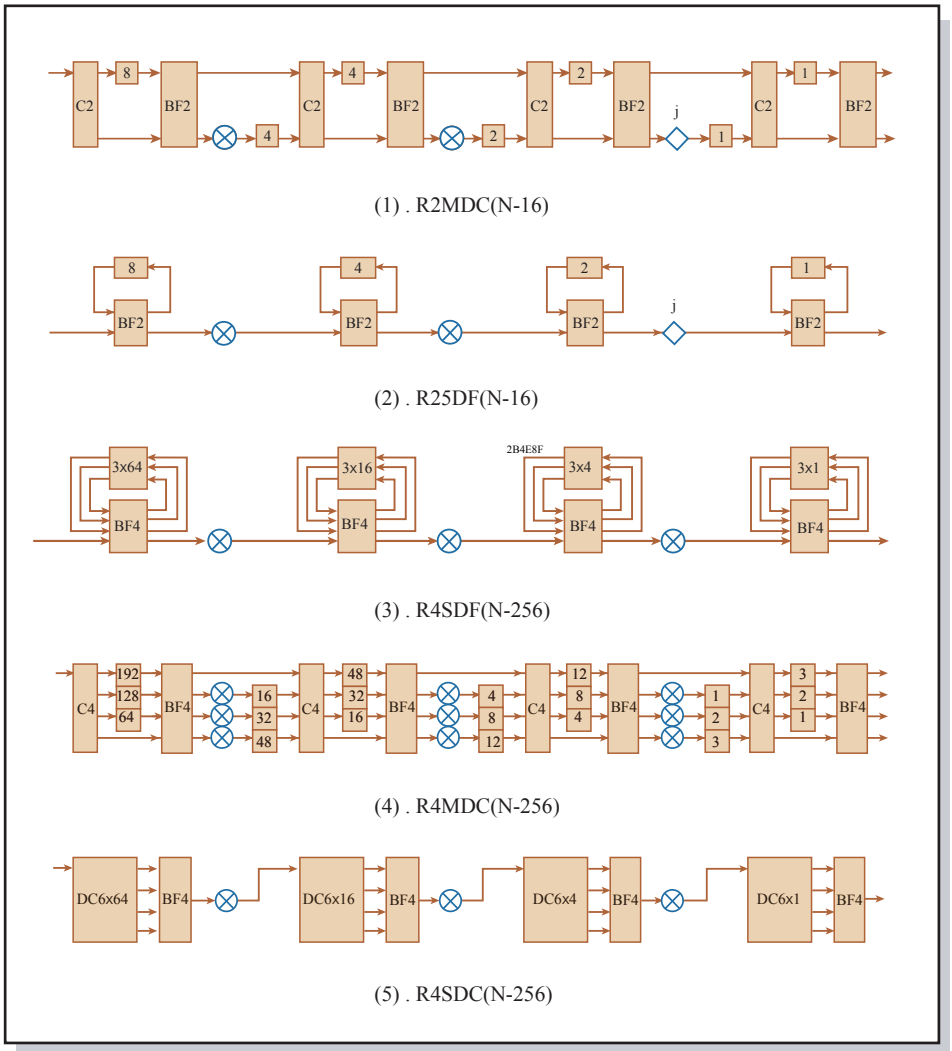


Figure by MIT OpenCourseWare.

Radix-2 Multi-path Delay Commutator

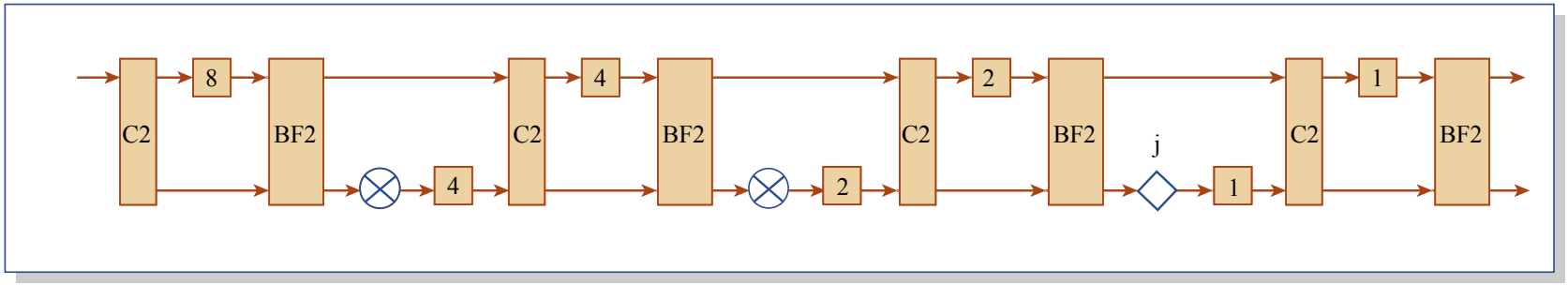


Figure by MIT OpenCourseWare.

- The most classical approach for pipeline implementation of radix-2 FFT
- Input sequence broken into two parallel data streams flowing forward with correct “distance” between data elements entering the butterfly scheduled by proper delays
- Both butterflies and multipliers are in 50% utilization

Radix-2 Single-path Delay Feedback

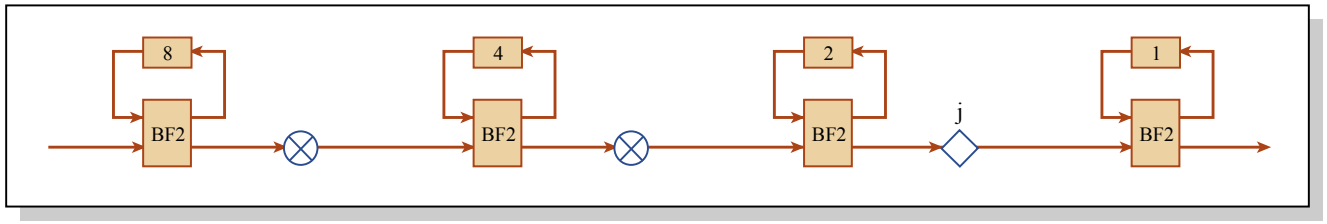


Figure by MIT OpenCourseWare.

[Wold & Despain '84]

- Uses registers more efficiently
  - Both as input and the output of the butterfly
- A single data stream goes through the multiplier at every stage
- Multiplier utilization is also 50%

Radix-4 Single-path Delay Feedback

[Despain74]

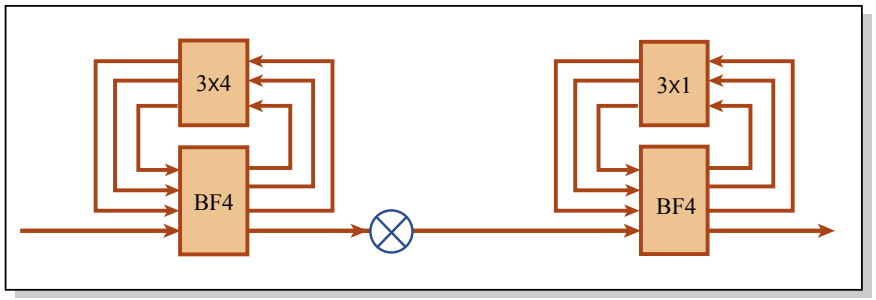


Figure by MIT OpenCourseWare.

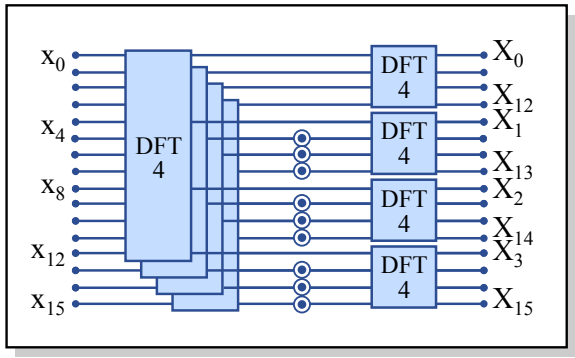


Figure by MIT OpenCourseWare.

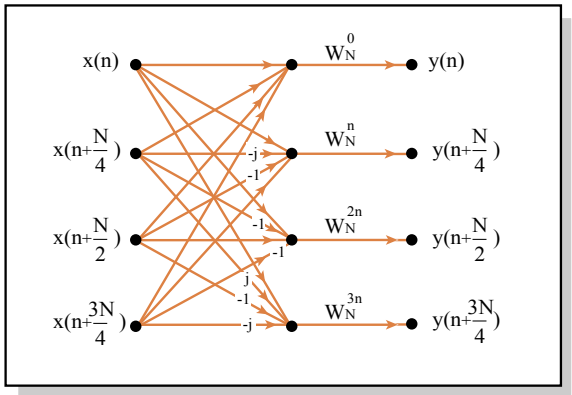


Figure by MIT OpenCourseWare.

- Utilization of multipliers 75%
  - By storing 3 BF4 outputs
- Radix-4 butterfly utilization only 25%
  - Butterfly fairly complicated
    - At least 8 complex adders

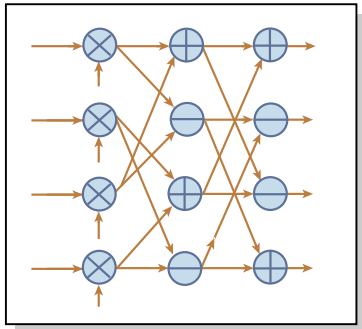


Figure by MIT OpenCourseWare.

Radix-4 Multi-path Delay Commutator

[Swartzlander84]

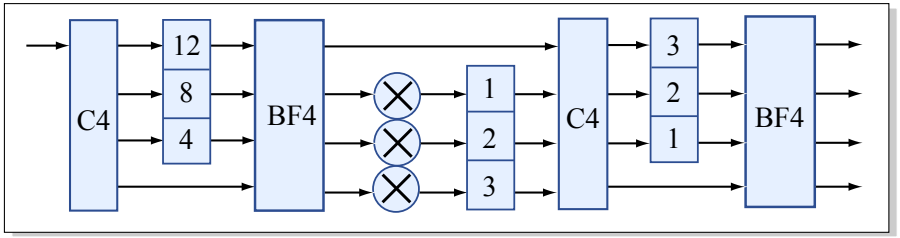


Figure by MIT OpenCourseWare.

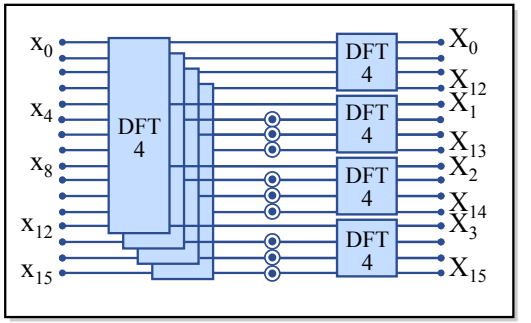


Figure by MIT OpenCourseWare.

- What is the utilization of
  - Butterflies?
  - Multipliers?

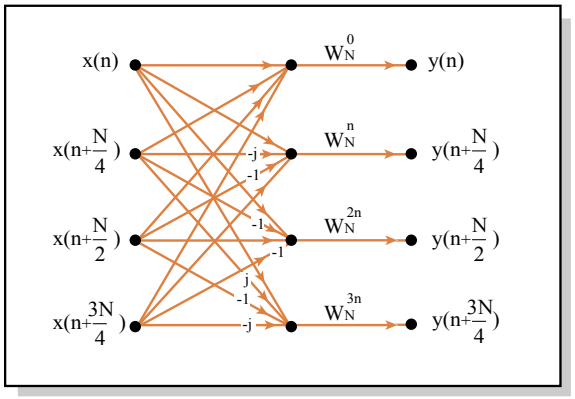


Figure by MIT OpenCourseWare.

Radix-4 Single-path Delay Commutator

[Bi & Jones '89]

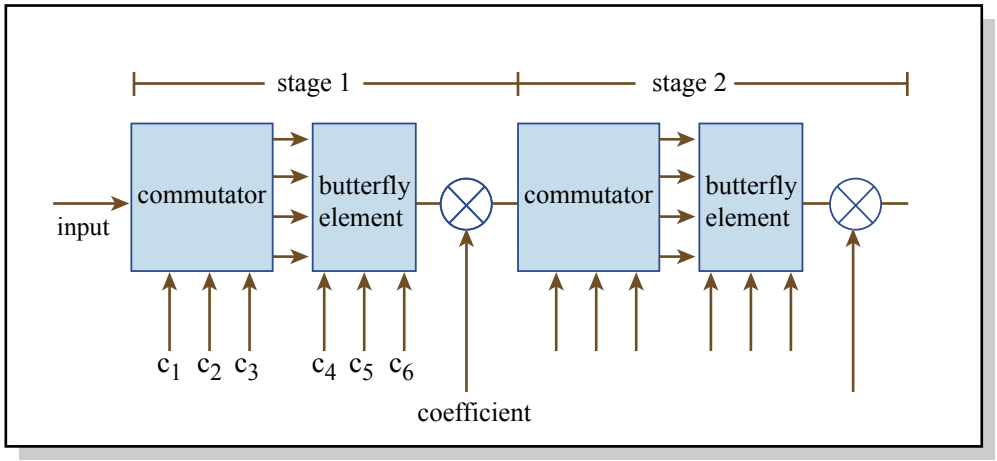


Figure by MIT OpenCourseWare.

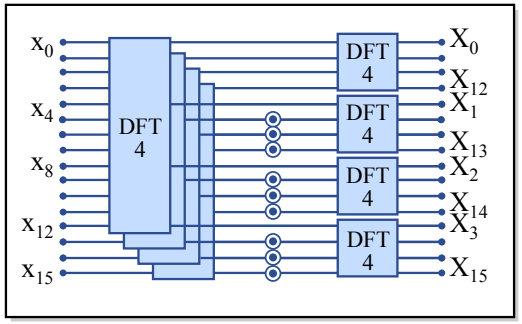


Figure by MIT OpenCourseWare.

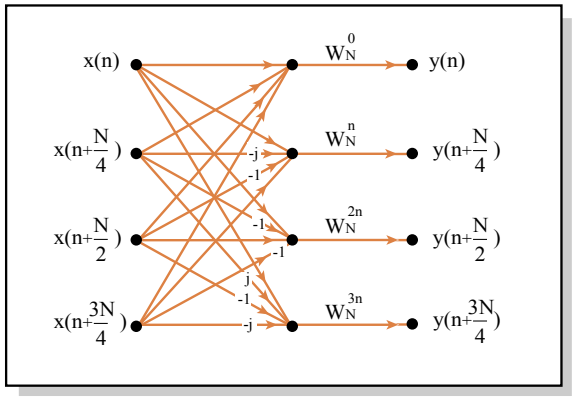


Figure by MIT OpenCourseWare.

- Modified radix-4 algorithm
- Programmable ¼ radix-4 BF
- 75% utilization
- Used to build one of the largest single-chip FFTs (8192pts) [Bidet'95]

R4SDC commutator and butterfly details

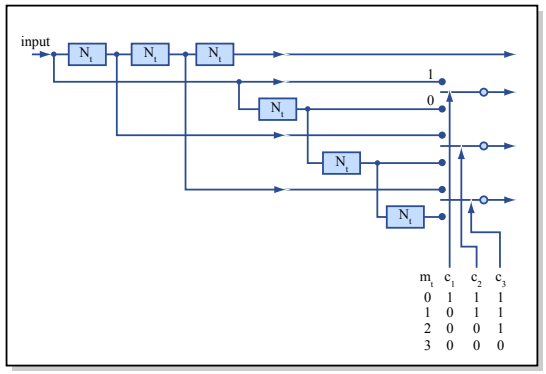


Figure by MIT OpenCourseWare.

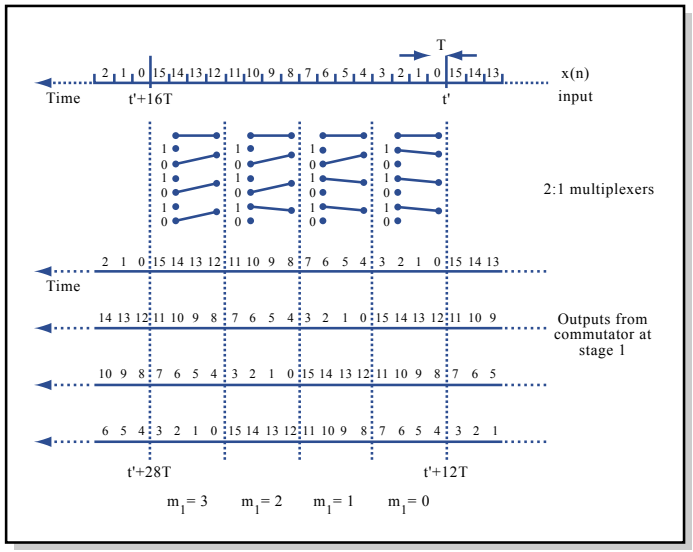


Figure by MIT OpenCourseWare.

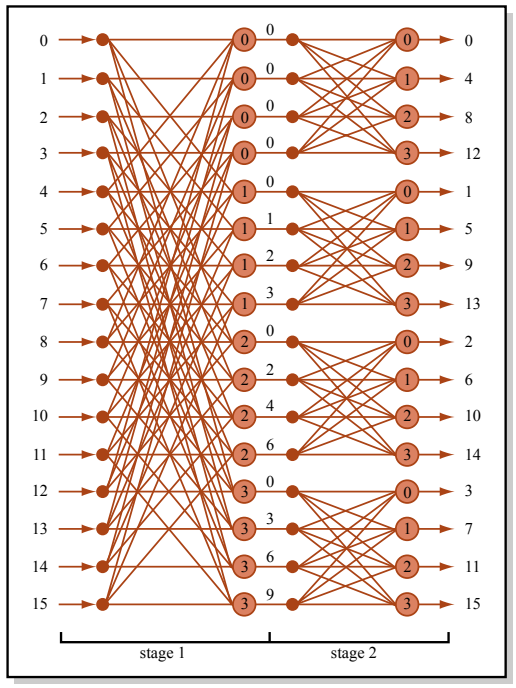


Figure by MIT OpenCourseWare.

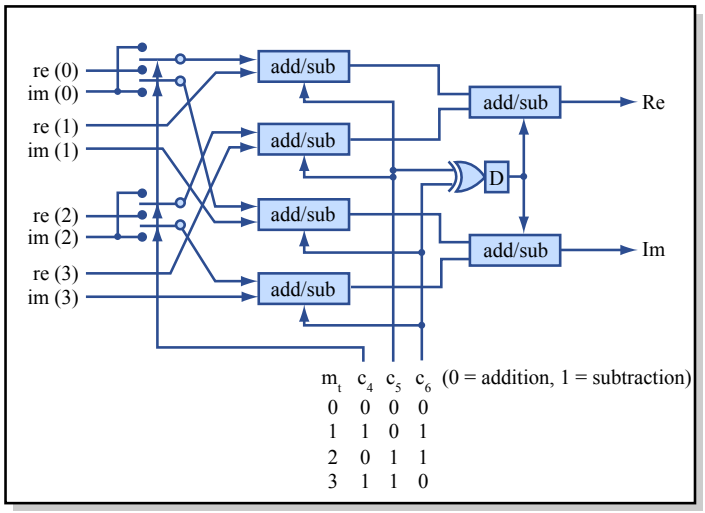


Figure by MIT OpenCourseWare.

Some conclusions

- Delay feedback approaches are always more efficient than corresponding delay-commutator approaches
  - In terms of memory utilization
    - Since butterfly outputs share same storage with its inputs
- Pipeline architectures require FFT algorithms to be formulated in a “hardware-oriented” form
  - Where spatial regularity is preserved in a signal-flow graph (SFG)
  - So that arithmetic operations can be tightly scheduled for efficient hardware utilization

Decomposition – a review

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad 0 \le k < N$$

- Twiddle factor is Nth primitive root of unity
  - With exponent evaluated modulo N
- Most fast algorithms share same general strategy
  - Map one-dimensional transform int a two or multi-dimensional representation
    - Exploit congruence property of coefficients to simplify computation
- Unlike traditional step-by-step decomposition of twiddle factors
  - Cascading the twiddle factor decomposition leads to new forms of FFT with high-spatial regularity

Radix 2^2 approach

- Start by classical divide-and-conquer radix-2 DIF indexing

$$n = < \tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3 >_N$$
$$k = < k_1 + 2k_2 + 4k_3 >_N$$

  - But, consider the first two steps of decomposition together

[Shouseng and Torkelson 1996]

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\tfrac{N}{2} n_1 + \tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)}$$

$$= \sum_{n_3=0}^{\frac{N}{4}-1} \sum_{n_2=0}^{1} B_{\frac{N}{2}}^{k_1}(\tfrac{N}{4} n_2 + n_3) W_N^{(\frac{N}{4} n_2 + n_3) k_1} W_N^{(\frac{N}{4} n_2 + n_3)(2k_2 + 4k_3)}$$

Compute directly in standard radix-2 approach

New idea is to proceed to shorter DFTs cascading the twiddle factor $W_N^{(N/4n2+n3)k1}$

$$W_N^{(\frac{N}{4} n_2 + n_3)(k_1 + 2k_2 + 4k_3)} = W_N^{N n_2 k_3} W_N^{\frac{N}{4} n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3}$$

$$= (-j)^{n_2 (k_1 + 2k_2)} W_N^{n_3 (k_1 + 2k_2)} W_N^{4 n_3 k_3}$$

$$B_{\frac{N}{2}}^{k_1}(\tfrac{N}{4} n_2 + n_3) = x(\tfrac{N}{4} n_2 + n_3) + (-1)^{k_1} x(\tfrac{N}{4} n_2 + n_3 + \tfrac{N}{2})$$

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left[H(k_1, k_2, n_3) W_N^{n_3 (k_1 + 2k_2)} \right] W_{\frac{N}{4}}^{n_3 k_3}$$

$$H(k_1, k_2, n_3) = \overbrace{\left[x(n_3) + (-1)^{k_1} x(n_3 + \tfrac{N}{2}) \right]}^{\text{BF I}} + (-j)^{(k_1 + 2k_2)} \overbrace{\left[x(n_3 + \tfrac{N}{4}) + (-1)^{k_1} x(n_3 + \tfrac{3}{4} N) \right]}^{\text{BF I}}$$

BF II

A 16pt example

- Get radix-4-like mulitplier complexity with radix-2 butterfly structures (radix-2^2)

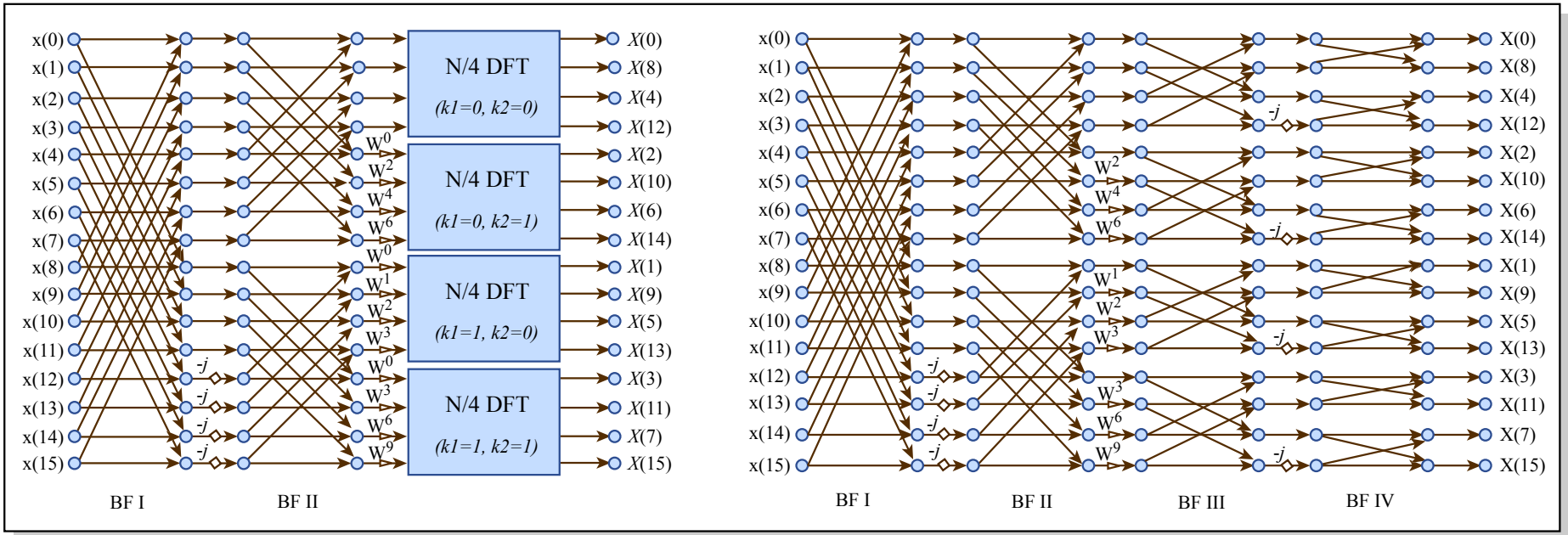


Figure by MIT OpenCourseWare.

$$X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{\frac{N}{4}-1} \left[H(k_1, k_2, n_3) W_N^{n_3(k_1+2k_2)} \right] W_{\frac{N}{4}}^{n_3 k_3}$$

$$H(k_1, k_2, n_3) = \overbrace{\left[x(n_3) + (-1)^{k_1} x(n_3 + \frac{N}{2}) \right]}^{\text{BF I}} + (-j)^{(k_1+2k_2)} \overbrace{\left[x(n_3 + \frac{N}{4}) + (-1)^{k_1} x(n_3 + \frac{3}{4}N) \right]}^{\text{BF I}}$$

$$\text{BF II}$$

A 64pt radix-2^2 example

Image removed due to copyright restrictions.

Radix-2^2 (R2^2SDF) architecture

N=256

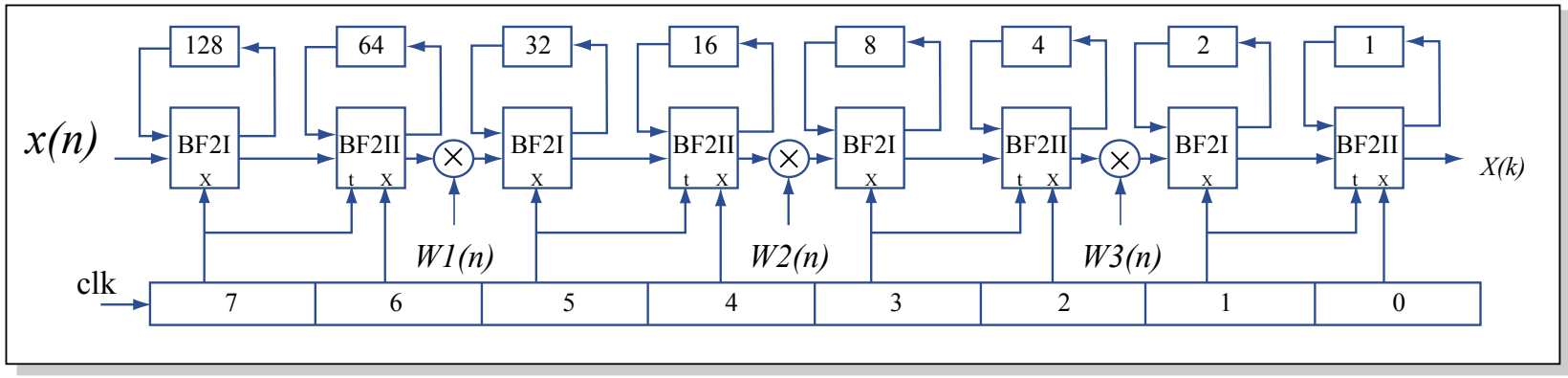


Figure by MIT OpenCourseWare.

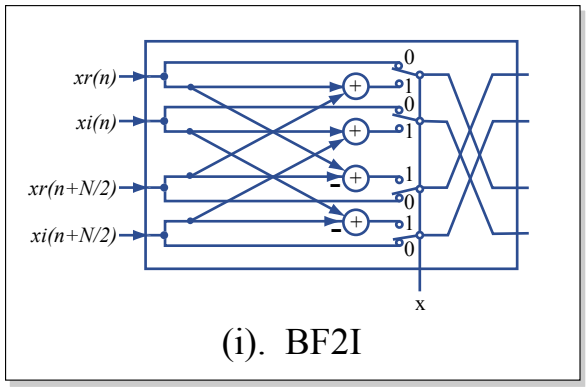


Figure by MIT OpenCourseWare.

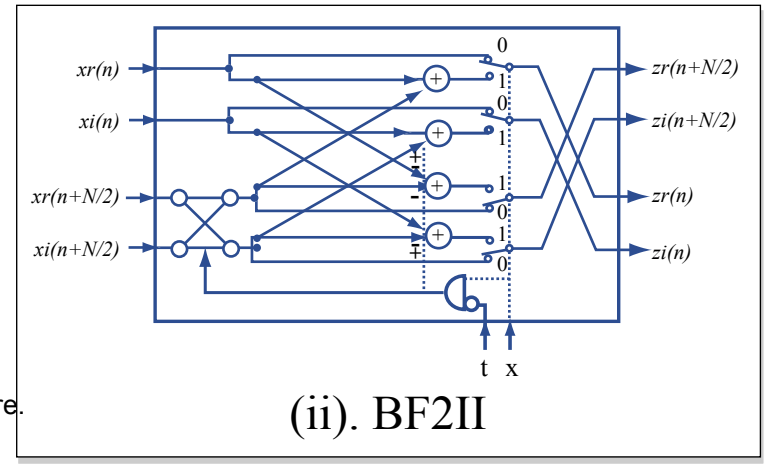


Figure by MIT OpenCourseWare.

- Similar to R2SDF
  - Reduced number of multipliers
- Need two types of butterflies
  - One identical to that in R2SDF
  - The other contains the logic for trivial twiddle factor multiplication (with j)
- Synchronization control very simple due to spatial regularity
  - Just a $\log_2$N binary counter

Radix-2^2 architecture – Sync control

- log2N-bit binary counter
  - Synchronization controller
  - Address counter for twiddle factor reading in each stage
- On first N/2 cycles, 2-to-1 mux in BF1 switch to 0
  - Butterfly is idle (input data directed to shift registers)
- On next N/2 cycles, muxes in BF1 switch to 1
  - Butterfly computes a 2pt DFT with incoming data and data stored in the shift registers
  - Output Z1(n) sent to twiddle multiplier
  - Output Z1(n+N/2) sent back to the shift register to be "multiplied" in next N/2 cycles, when the first half of the next frame is loaded in

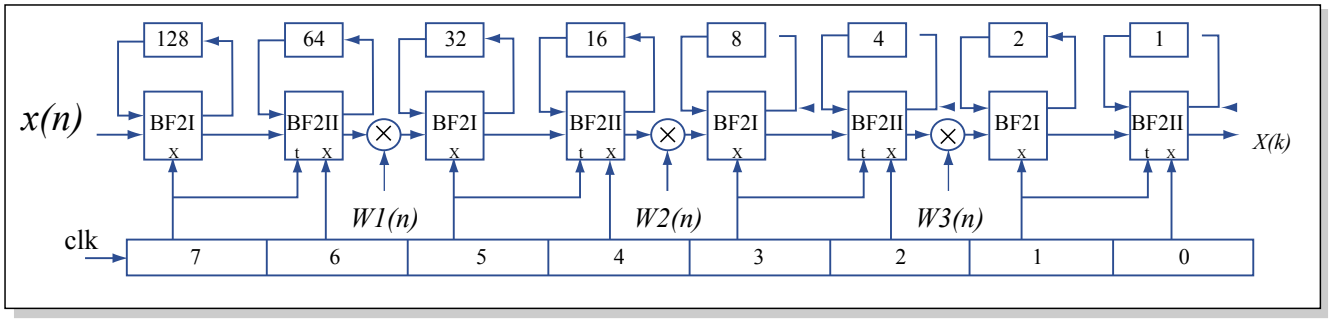


Figure by MIT OpenCourseWare.

$$Z1(n) = x(n) + x(n+N/2)$$
$$Z1(n+N/2) = x(n) - x(n+N/2) \quad ,0 \le n < N/2$$

- Operation of BF2 is similar, except the "distance" of butterfly input sequence is just N/4 and the trivial multiply logic
- Utilization of the multiplier is 75%
- Next frame can be computed w/o pausing due to the pipelined processing in each stage
- Pipeline register can be inserted between each multiplier and BF stage to improve the performance

Arithmetic complexity

	multiplier #	adder #	memory size	control
R2MDC	$2(\log_4 N - 1)$	$4 \log_4 N$	$3N/2 - 2$	simple
R2SDF	$2(\log_4 N - 1)$	$4 \log_4 N$	$N - 1$	simple
R4SDF	$\log_4 N - 1$	$8 \log_4 N$	$N - 1$	medium
R4MDC	$3(\log_4 N - 1)$	$8 \log_4 N$	$5N/2 - 4$	simple
R4SDC	$\log_4 N - 1$	$3 \log_4 N$	$2N - 2$	complex
R2^{2}SDF	$\log_4 N - 1$	$4 \log_4 N$	$N - 1$	simple

Figure by MIT OpenCourseWare.

- R2^{2}SDF has reached minimum requirement for both multiplier and storage
- Only R4SDC better in terms of adder usage
- R2^{2}SDF well suited for VLSI implementations of pipeline FFT processors

Memory issues

- The area/power consumption in the pipeline architectures dominated by the
  - FIFO register files at each stage
  - Complex multipliers at each (or every other stage)
- To diminish the unnecessary data moving in the FIFO need to reconstruct the storage
  - A known approach is to use FIFO with 2-port RAM
    - With read and write addresses displaced by a constant
    - 2-port RAM cells 33% more area of the 1-port RAM cell

  - Use two N/2 1-port RAMs
    - Read and write interleaved
    - Each active every other cycle

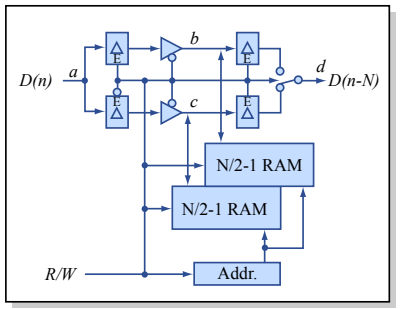


Figure by MIT OpenCourseWare.

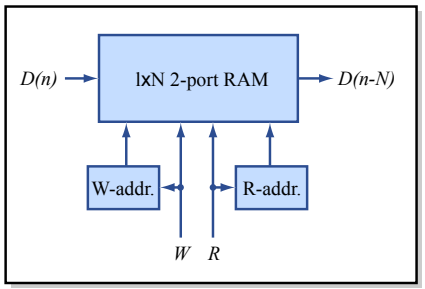


Figure by MIT OpenCourseWare.

Single stage hardware example

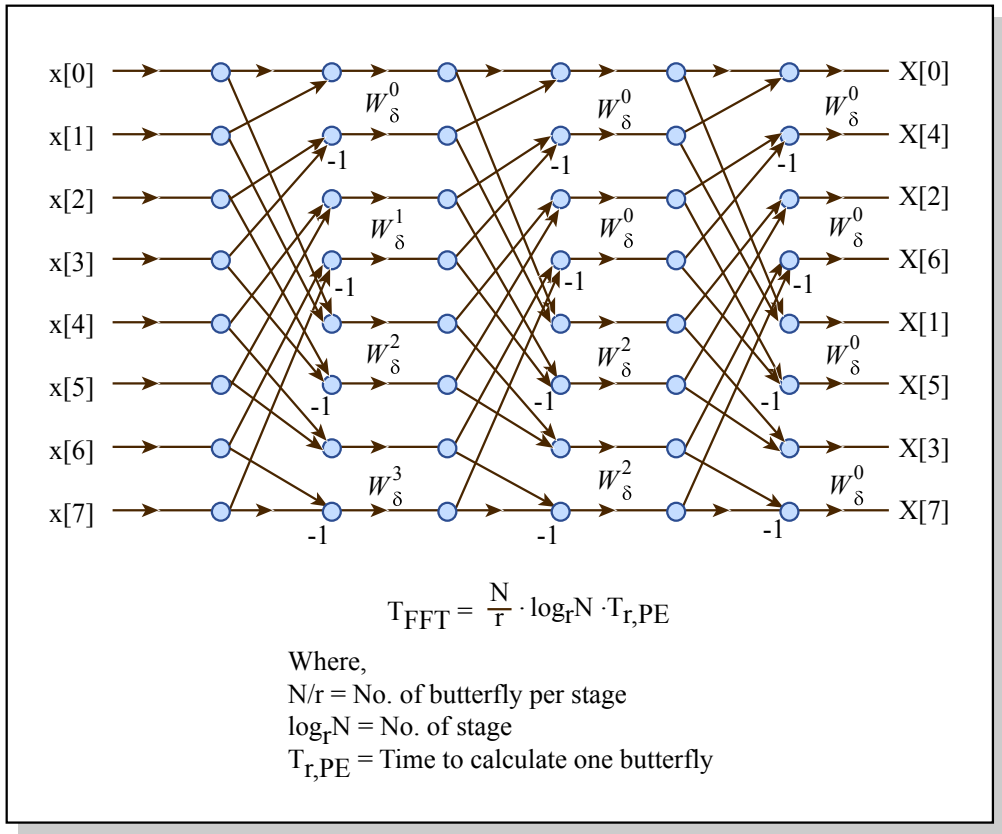


Figure by MIT OpenCourseWare.

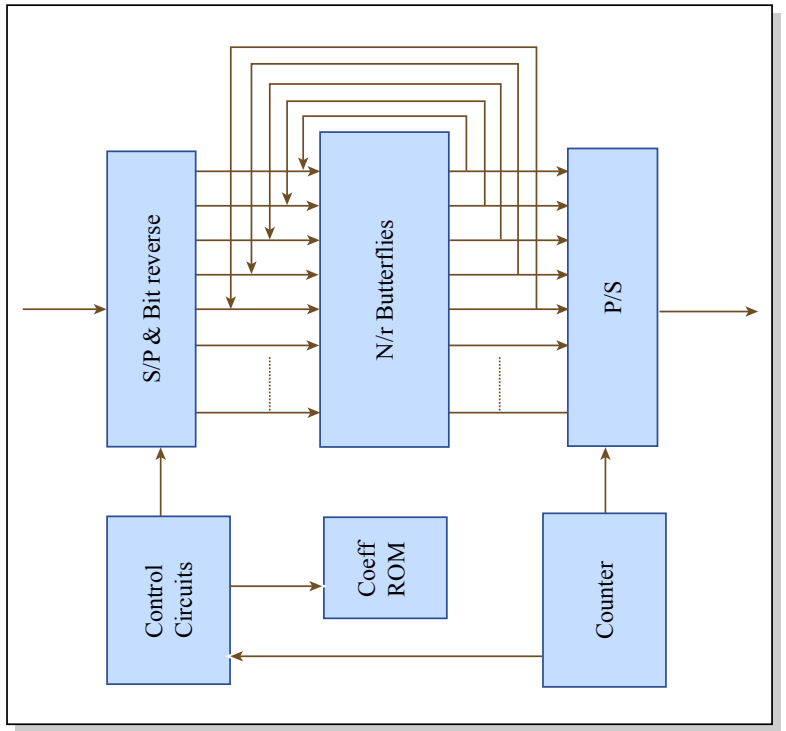


Figure by MIT OpenCourseWare.

[Sadat2001]

- Fold stages onto each other
  - Need constant geometry signal flow graph
  - Big price in area for parallelism (within each stage)

Radix-8 Pipelined/Parallel implementation

- A 64pt FFT example for 802.11a

[Excerpted from Maharatna et al 2004]

$$A(r) = \sum_{k=0}^{N-1} B(k) W_N^{rk}$$

$$A(s+Tt) = \sum_{l=0}^{M-1} W_M^{lt} \left[W_{MT}^{sl} \sum_{m=0}^{T-1} B(l+Mm) W_T^{sm} \right]$$

$$A(s+8t) = \sum_{l=0}^{7} \left[W_{64}^{sl} \sum_{m=0}^{7} B(l+8m) W_8^{sm} \right] W_8^{lt}.$$

- Two dimensional structure of 8pt FFTs
  - The number of nontrivial complex multiplications is 49 (7x7)
    - Since the first twiddle is always 1
  - The number of nontrivial complex multiplications for radix-2 FFT is 66
  - Radix-4 (or 2^2) FFTs need only 52 multiplies
- Important to note that for 8pt FFT (DIT) no need for multiplies

8pt DIT FFT

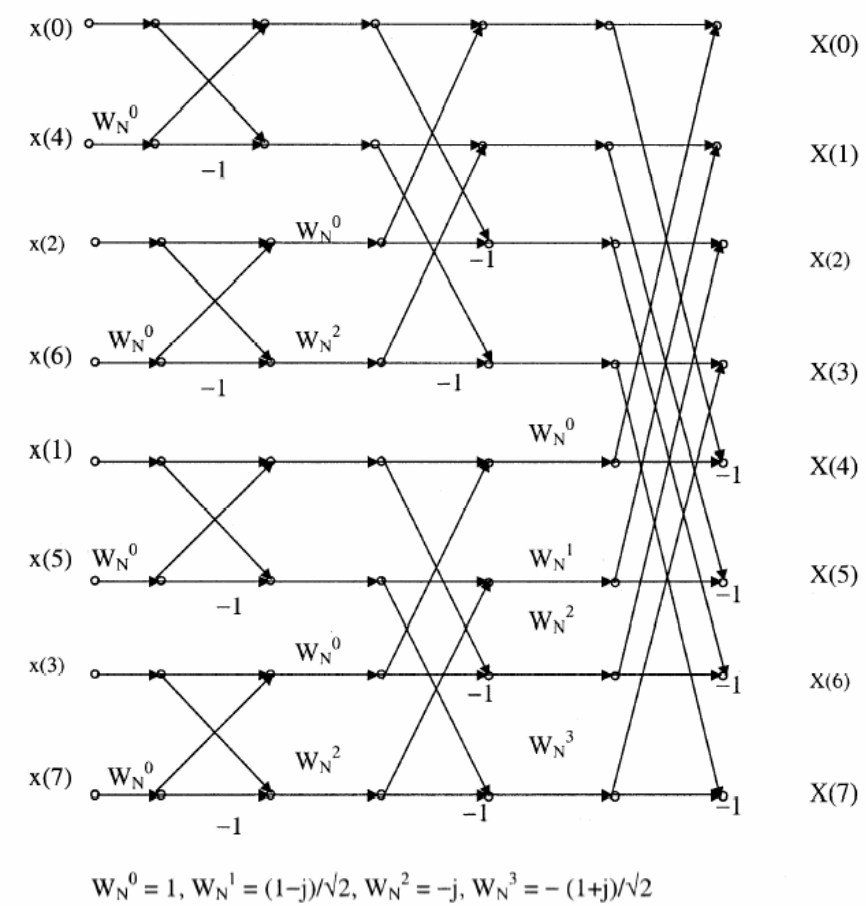


Figure from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

- The only nontrivial multiply is with 1/sqrt(2)
  - Easily realize using hardwired shift-and-add

Block diagram of the FFT unit

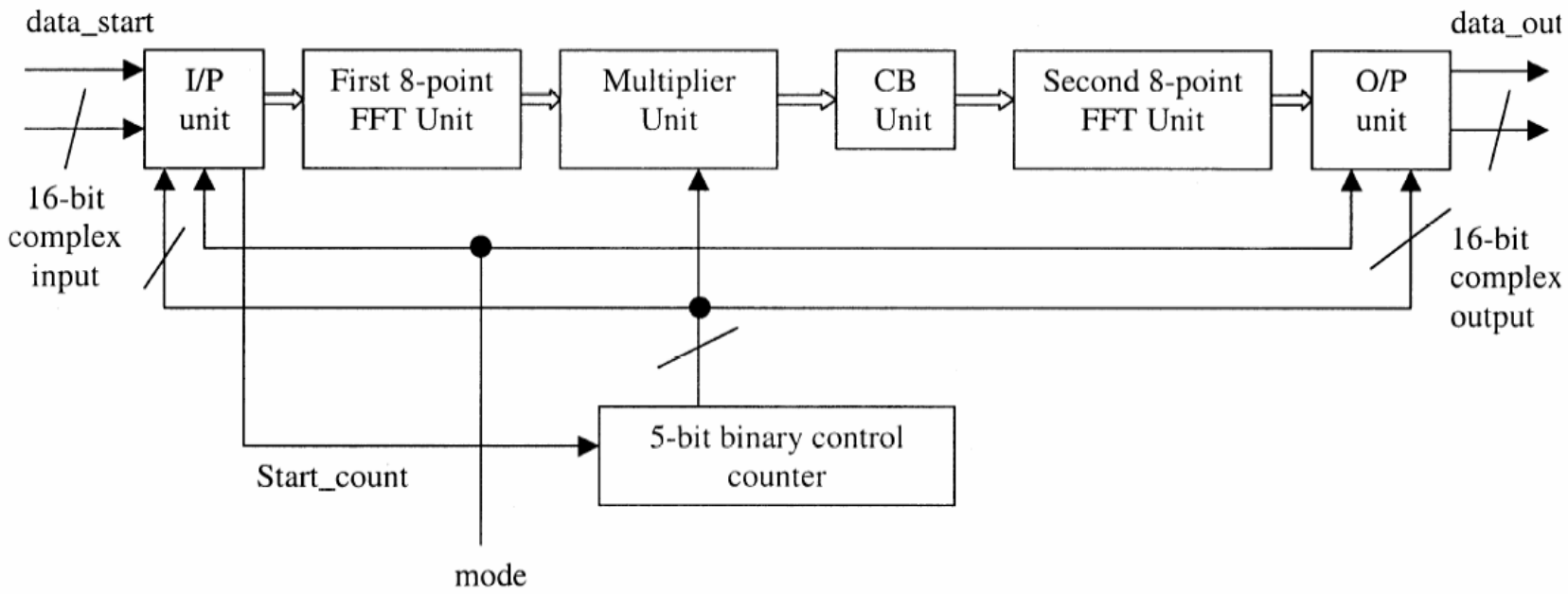


Figure from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

- Two-stages are pipelined
  - Fully parallel in each stage (radix-2 8pt FFT, single clk cycle)
- Two performance bottlenecks
  - Large number of global wires resulting from the multiplexing of complex data to the 8-point FFTs
  - Construction of the multiplier unit to attain the required speed with minimal silicon are is not trivial

Input unit

- Hard wired outputs and data shifting
  - To the 8pt FFT
  - Reduce de-muxing
  - Reduce global wires
- Cannot shift every clk
  - Multiplier cannot finish
  - Extend latency
    - Temporary registers 1,2,3

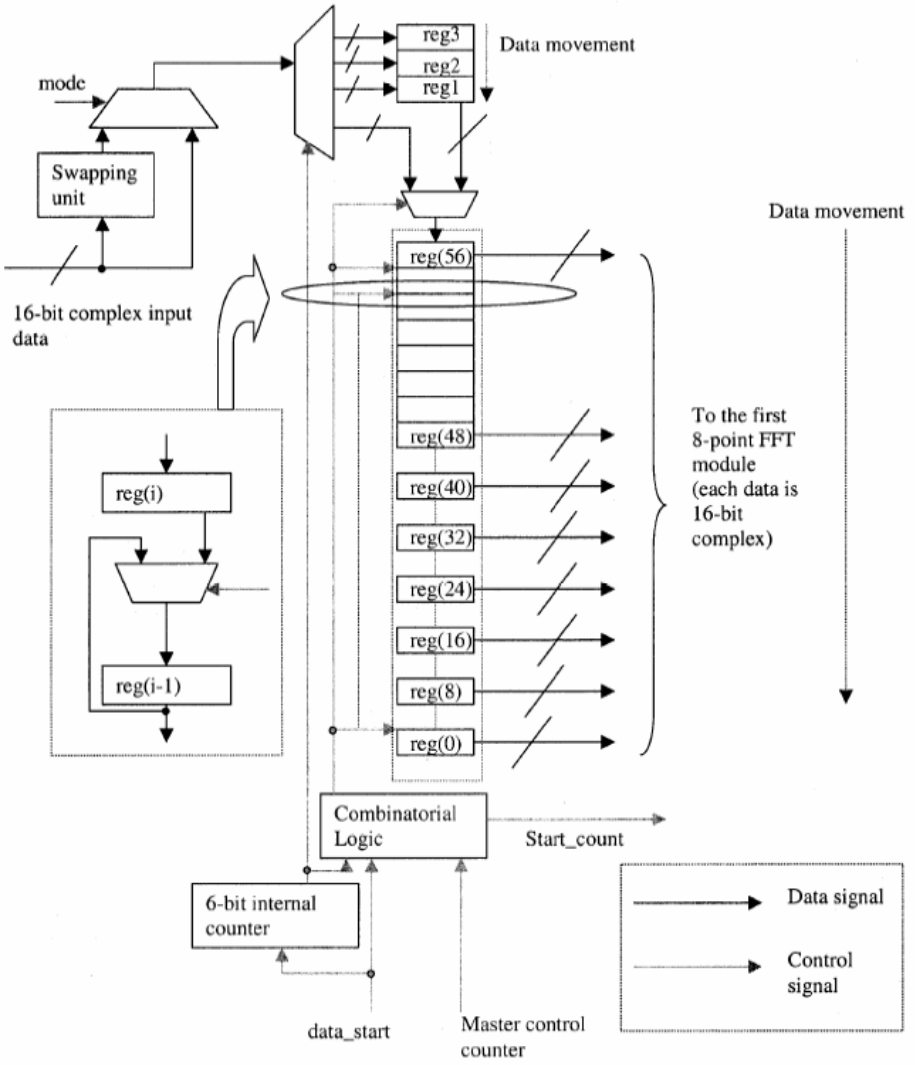


Figure from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

Multiplier unit

- 49 multiplies $(W_{64}^{sl}, s, l \in \{1, 2, \ldots, 7\})$
  - Only nine sets unique (cos,sin)

They are (1,0), (0.995178, 0.097961), (0.980773, 0.195068), (0.956909, 0.290283), (0.923828, 0.382629), (0.881896, 0.471374), (0.831420, 0.555541), (0.773010, 0.634338), (0.707092, 0.707092), where, in each set, the first entry cor-

  - Significantly less storage space
    - for coefficients
- Turn multiplies into shift&add

$$0.995178 \quad (1 - 2^{-8} - 2^{-10} + 2^{-14})$$

hard-wired constant

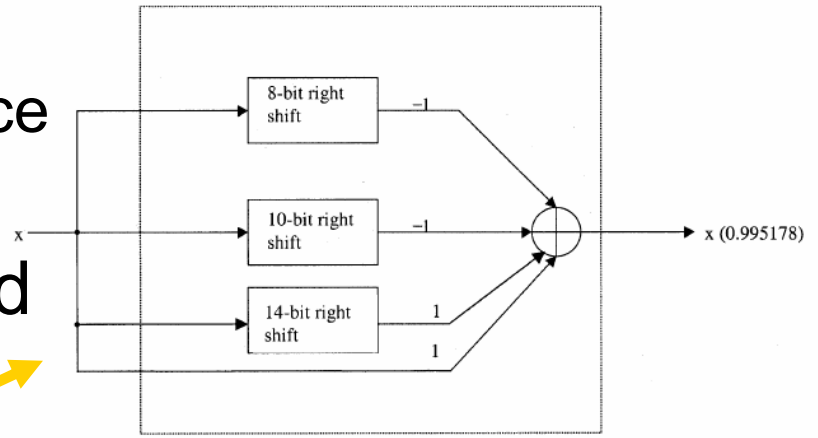
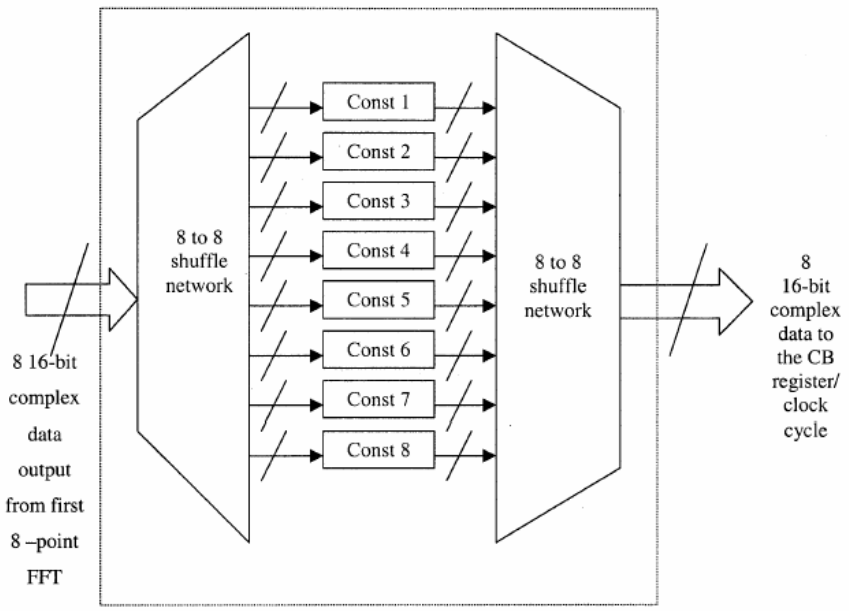


Figure from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

Multiplier unit and scheduling



UTILIZATION OF THE DIFFERENT HARD-WIRED CONSTANTS DURING THE 49 COMPLEX MULTIPLICATION OPERATION

Time instant	Block data from 8-point FFT	Const1	Const2	Const3	Const4	Const5	Const6	Const7	Const8
T = 0	0th	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'0'
T = 1	1st	'1'	'1'	'1'	'1'	'1'	'1'	'1'	'0'
T = 2	2nd	'0'	'1'	'0'	'1'	'0'	'1'	'0'	'1'
T = 3		'0'	'1'	'0'	'1'	'0'	'1'	'0'	'0'
T = 4	3rd	'1'	'1'	'1'	'1'	'1'	'1'	'1'	'0'
T = 5	4th	'0'	'0'	'0'	'1'	'0'	'0'	'0'	'1'
T = 6		'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'
T = 7		'0'	'0'	'0'	'1'	'0'	'0'	'0'	'1'
T = 8		'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'
T = 9	5th	'1'	'1'	'1'	'1'	'1'	'1'	'1'	'0'
T = 10	6th	'0'	'1'	'0'	'1'	'0'	'1'	'0'	'1'
T = 11		'0'	'1'	'0'	'1'	'0'	'1'	'0'	'0'
T = 12	7th	'1'	'1'	'1'	'1'	'1'	'1'	'1'	'0'

Figures from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

- Some of the coefficients requested concurrently by different FFT outputs
  - Solve by adding temp registers in the input unit
- ~50% less power and area than 8 standard complex multipliers
- Buffer unit similar to input unit, just w/o temporary registers
  - Outputs also hardwired with distance of 8

Output unit

- A mirror of input unit
  - Just w/o temporary registers
- Control/sync is simple
  - 5-bit counter
    - Starts counting when input full
    - Local counters control
      - Input
      - Intermediate
      - Output units

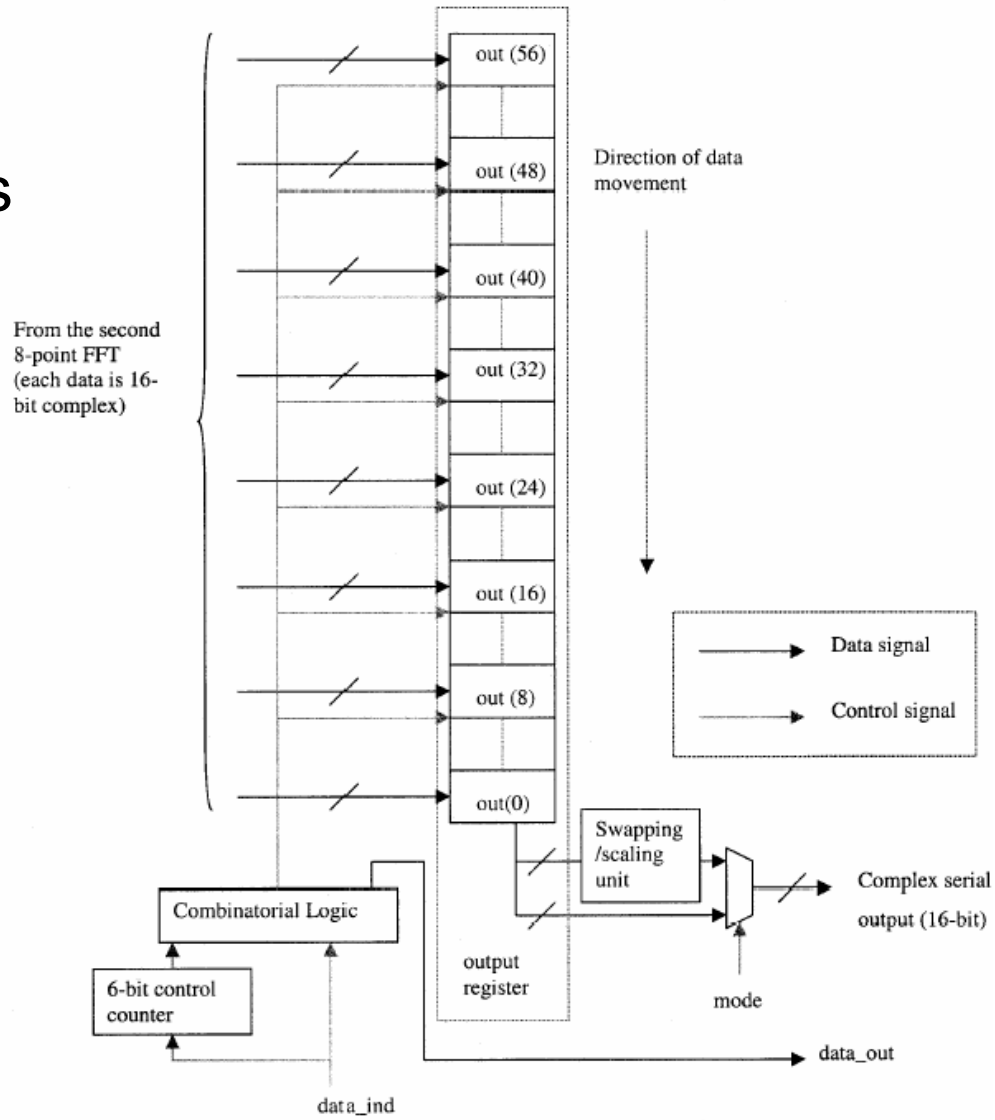


Figure from Maharatna, K., E. Grass, and U. Jagdhold. "A 64-point Fourier Transform Chip for High-speed Wireless LAN Application Using OFDM." *Solid-State Circuits* 39 (2004): 484-493. Copyright 2004 IEEE. Used with permission.

Readings

- [1] H.e. Shousheng and M. Torkelson "A new approach to pipeline FFT processor," *Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International* no. SN -, pp. 766-770, 1996.
  - [3] H.e. Shousheng and M. Torkelson "Designing pipeline FFT processor for OFDM (de)modulation," *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on* no. SN -, pp. 257-262, 1998.
- [2] E. Wold and Alvin M. Despain "Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations," *IEEE Trans. Computers* vol. 33, no. 5, pp. 414-426, 1984.
- [3] G. Bi and E.V. Jones "A pipelined FFT processor for word-sequential data," *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on* vol. 37, no. 12 SN - 0096-3518, pp. 1982-1985, 1989.
- [4] K. Maharatna, E. Grass and U. Jagdhold "A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM," *Solid-State Circuits, IEEE Journal of* vol. 39, no. 3 SN - 0018-9200, pp. 484-493, 2004.
- Interesting DIT&F algorithm
  - [4] C. Chiu, W. Hui, T.J. Ding and J.V. McCanny "A 64-point Fourier transform chip for video motion compensation using phase correlation," *Solid-State Circuits, IEEE Journal of* vol. 31, no. 11 SN - 0018-9200, pp. 1751-1761, 1996.
- Power-performance estimation
  - [2] S. Hong, S. Kim, M.C. Papaefthymiou and W.E. Stark "Power-complexity analysis of pipelined VLSI FFT architectures for low energy wireless communication applications," *Circuits and Systems, 1999. 42nd Midwest Symposium on* vol. 1, no. SN -, pp. 313-316 vol. 1, 1999.
  - [3] K. Pagiamtzis and P.G. Gulak "Empirical performance prediction for IFFT/FFT cores for OFDM systems-on-a-chip," *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on* vol. 1, no. SN -, pp. I-583-6 vol.1, 2002.