ECE 4270
Fundamentals of Digital Signal Processing

Lecture 22:
The Fast Fourier Transform

School of Electrical and Computer Engineering
Georgia Institute of Technology
Summer, 2004

Computation of the DFT

- In order for the DFT to be useful for linear filtering, nonlinear filtering, spectrum analysis, *etc*., we need efficient computation algorithms for

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \qquad k = 0, 1, \cdots, N-1$$

- Using the above directly requires N complex multiplications and N-1 complex additions for each of the N DFT values $\Rightarrow$ $\beta(N) = N^2$ complex multiplications. For example,

$$N = 1024 \quad \Rightarrow \quad \beta(1024) \approx 10^6$$

Computation Required

- We'll count complex multiplies; the number of complex adds is about the same
- Let $\mu(N)$ be the number of multiplies needed to compute an N-point DFT using an FFT algorithm, and let $\beta(N)$ be the number of multiplies needed to compute an N-point DFT in brute-force fashion
- Thus we start with

$$\beta(N) = N^2$$

The Goal of "the" FFT Algorithm ...

- … is to compute the DFT of size N with significantly fewer than N^2 complex multiplications and additions
- To accomplish this, researchers have come up with entire families of fast algorithms; these are *collectively* referred to as "the" fast Fourier transform (FFT) algorithm
- The first (or at least most timely) FFT algorithm was published by Jim Cooley and John Tukey in 1965 and is now referred to as the radix-2 Cooley-Tukey FFT algorithm
  - this is the main version we will consider …

Cooley-Tukey Radix 2 Algorithm

- We assume that N is a power of two:

$$N = 2^{\nu}$$
$$= \underbrace{2 \cdot 2 \cdots 2 \cdot 2}_{\nu \text{ times}}$$

  - if not, and we want to use this algorithm, we zero-pad our data until it is the next highest power of 2 in length
- Note that $\nu = \log_2 N$

Decimation-in-Time Derivation - 1

- We want to compute $X[k]$ efficiently. Divide $x[n]$ into even- and odd-indexed subsequences; then

$$X[k] = \sum_{n \text{ even}} x[n] W_N^{nk} + \sum_{n \text{ odd}} x[n] W_N^{nk}$$

- Substituting $n=2r$ and $n=2r+1$ into above gives

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

$$= \sum_{r=0}^{(N/2)-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] (W_N^2)^{rk}$$

Decimation-in-Time Derivation - 2

- Using the fact

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2}$$

  it follows that

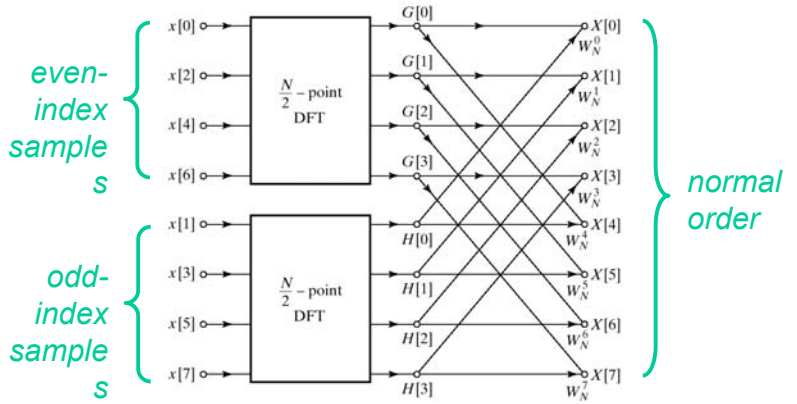
$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1.$$

- In other words, $G[k]$ and $H[k]$ are $N/2$-point DFTs!

$$\Rightarrow \quad \mu(N) = N + 2\beta(N/2)$$
$$= N + N^2/2 < N^2 = \beta(N)$$

Decimation in Time (First Stage)



$$X[k] = \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}$$

$$= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1.$$

Iterate …

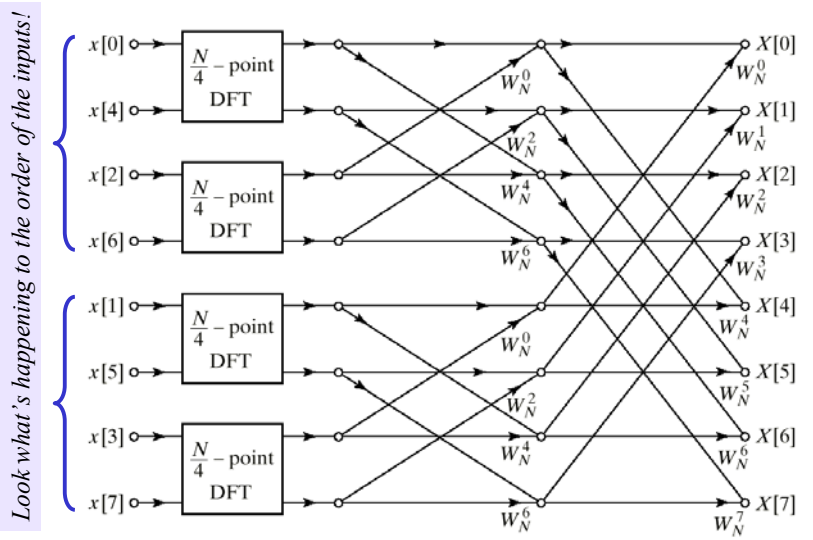
- We can repeat this trick by decomposing each of the $N/2$ point DFTs into two $N/4$ point DFTs
- The computation required to do an $N/2$ point DFT in this manner is

$$\mu(N/2) = N/2 + 2\beta(N/4)$$
$$= N/2 + 2\left(N^2/16\right) = N/2 + N^2/8$$

- Out total computation now has become

$$\mu(N) = N + 2\mu(N/2)$$
$$= N + 2\left(N/2 + N^2/8\right) = 2N + N^2/4$$

Decimation in Time (Second Stage)



How Many Times Can We Do This?

- The 1-point DFT is a "no-op":

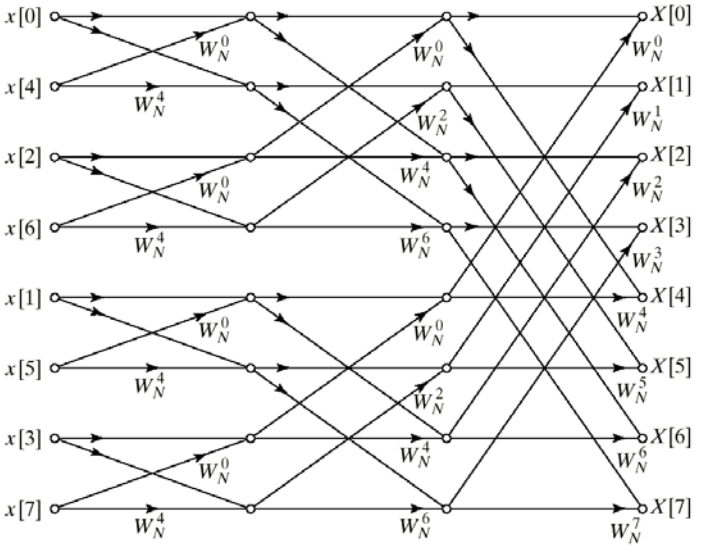
$$X[k] = \sum_{n=0}^{0} x[n] W_1^{nk}, \quad k = 0 \text{ only} \Rightarrow X[0] = x[0]$$

- So when we have it broken down to 2-point DFTs, we are done
- For $N = 2^{\upsilon}$, we do the decomposition υ-1 times
- The total computation turns out to be

$$\mu(N) = \underbrace{N + N + \cdots}_{\nu - 1 \text{ times}} + N^2/2^{\nu} = (\nu - 1)N + N = \nu N$$

$$\mu(N) = N \log_2 N$$

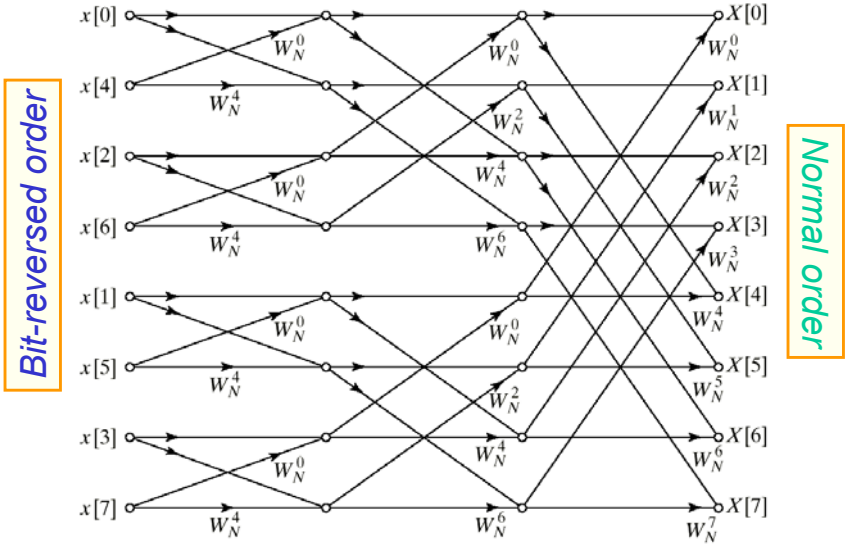
Decimation in Time (Third Stage)



Aside: Bit Reversed Order

Index	Binary	Bit-Reversed Binary	Bit-Reversed Index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Decimation in Time (Third Stage)



Simplifying the 2-Point DFT

- We can reduce the number of multiplies by an additional factor of 2 if we look at the details of a 2-point DFT:

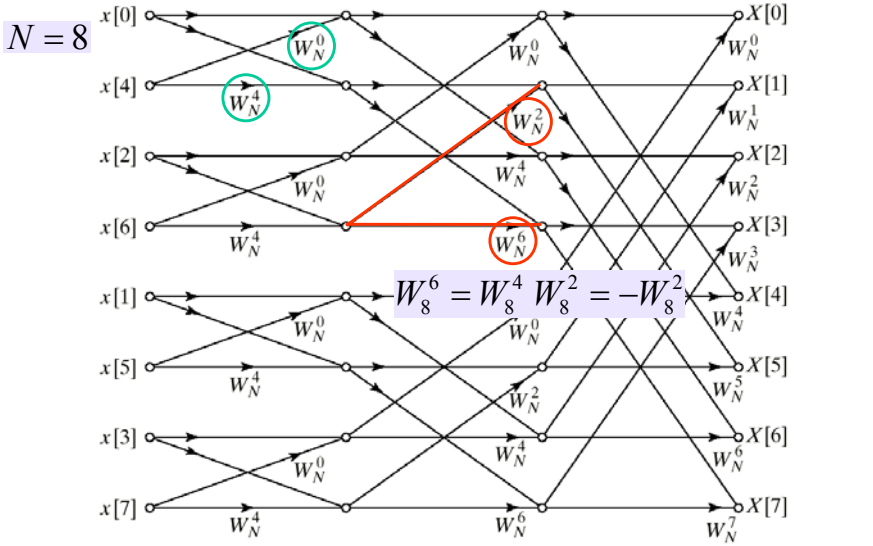
$$X[k]=\sum_{n=0}^{1}x[n]\,W_2^{nk}=\sum_{n=0}^{1}x[n]\,e^{-j\pi nk},\quad k=0,1\Rightarrow$$

$$X[0]=x[0]+x[1],\quad X[1]=x[0]-x[1]$$

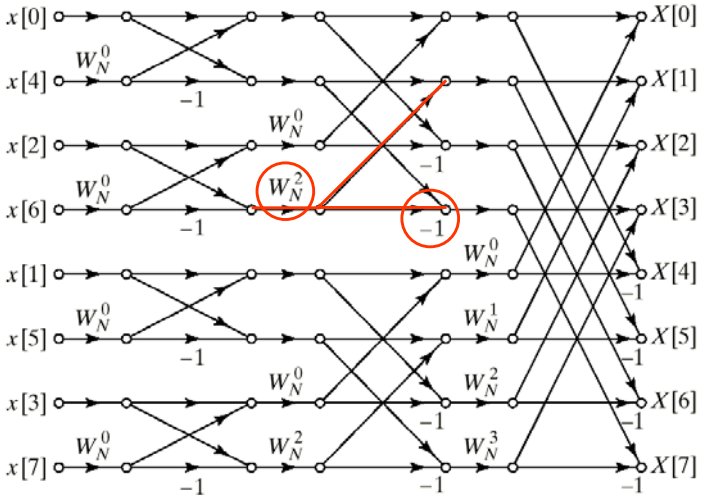
- In other words, the multiplies that appear in a two point DFT are actually trivial:

$$W_N^0=e^{-j0}=+1,\quad W_N^{N/2}=e^{-j\pi}=-1$$

Decimation in Time (Third Stage)



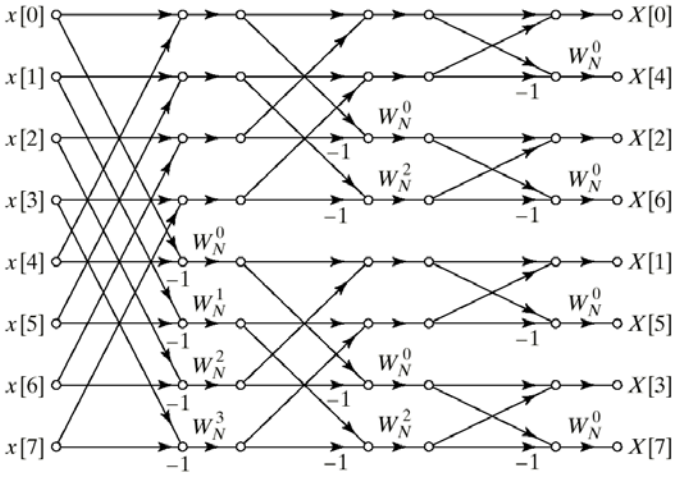
Decimation in Time (Restructured)



$\mu(N) = (N/2)\log_2 N$ complex multiplications

Decimation in Frequency

Signal in normal order

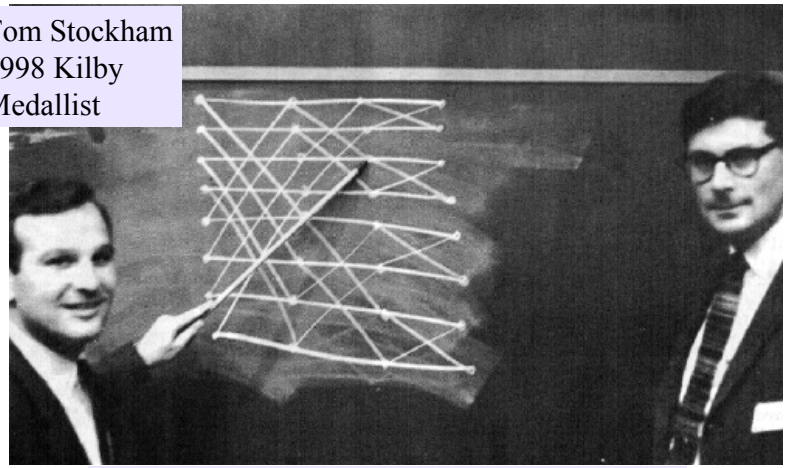


Trans-form in bit-reversed order

$\mu(N) = (N/2)\log_2 N$

Tom Stockham and Charlie Rader

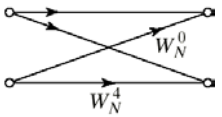
Tom Stockham
1998 Kilby
Medallist



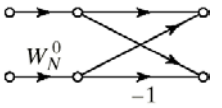
Inventors of the FFT flow graph representation

Modular Structure of the Cooley-Tukey FFT

- The entire FFT flowgraph is composed of repeated applications of the same 2-point DFT computation
  - There are $\upsilon = \log_2 N$ stages
  - Each has $N/2$ 2-point DFTs per stage
- The basic 2-point DFT is called a *butterfly*:

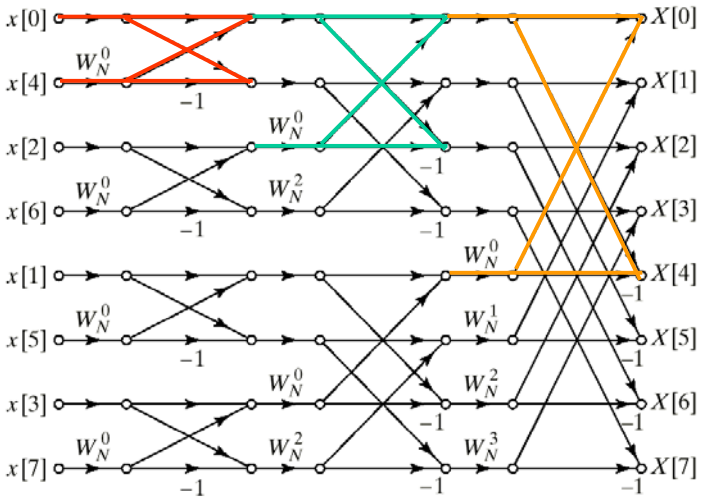


Original Form



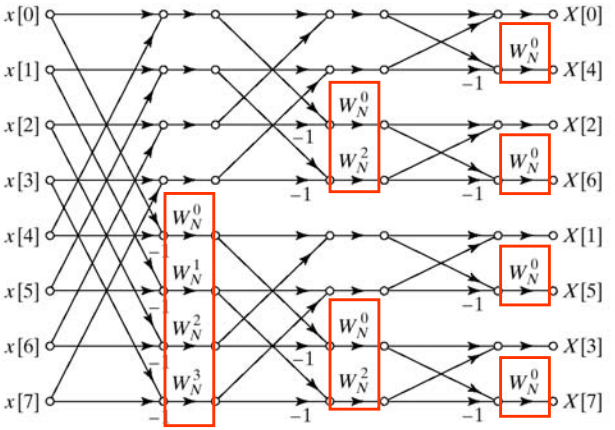
Reduced-multiplication Form

Butterflies in DIT FFT



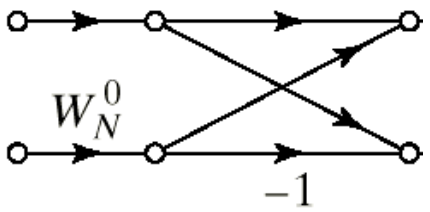
The Twiddle Factors

- The inter-stage multiplications, which are part of the butterfly, are called *twiddle factors*:



In-Place Computation

- "In-Place" means that, as each two values at the input to a butterfly are used,
  - they will not be used again, and
  - they can be overwritten with the outputs of the butterfly



FFT Generalizations

- Radix-R algorithms:

$$N = R^{\nu}$$

- Mixed-radix algorithms:

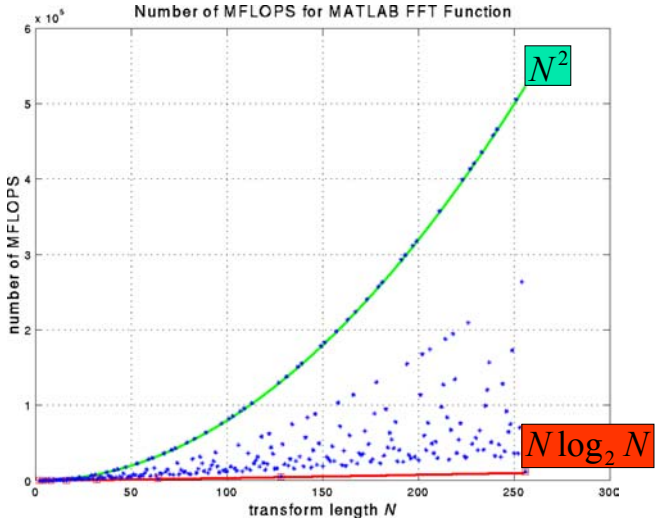
$$N = N_1 N_2 \ldots N_{\nu}$$

- Prime factor algorithms:

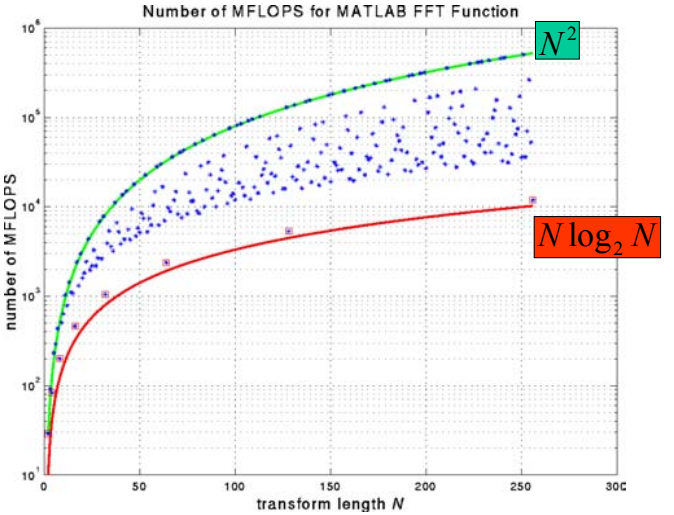
$$N = N_1 N_2 \ldots N_{\nu}, \quad N_i \text{ all prime}$$

- Winograd and chirp algorithms: based on representing $X[k]$ as a convolution

MATLAB's fft( ) Function

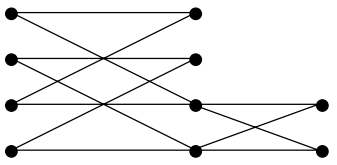


MATLAB's fft( ) Function

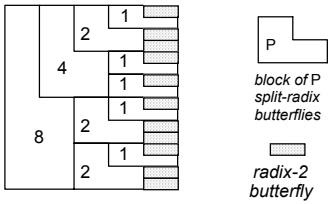


The Split-Radix Algorithm for Power-of-2 DFTs

- The split-radix algorithm butterfly is a cross between radix-4 and radix-2 designs:
- This results in an irregular structure which also requires radix-2 butterflies:



L-shaped split-radix butterfly



Structure of 32-point split-radix FFT

Multiply/Add Comparison of FFTs

- Split-radix has fewer multiplies than radix-2 or radix-4 Cooley-Tukey, and fewer combined adds and multiplies than minimum multiply algorithm
  - therefore best when multiply time dominates, or multiply and add times are similar and dominate

SIZE	COOLEY-TUKEY RADIX 2		COOLEY-TUKEY RADIX 4		SPLIT-RADIX		MINIMUM MULTIPLY	
	multiply	add	multiply	add	multiply	add	multiply	add
64	332	964	264	920	196	964	168	??
256	2316	5380	1800	5080	1284	5380	876	??
1024	13324	27652	10248	25944	7172	27652	3872	??

Note: these numbers are real multiplies and adds, and also take advantage of special cases such as multiply by +1, -1, +j, and -j

The Importance of Counting the Right Thing

- Suppose the radix-4 or split-radix butterflies is integrated onto a single VLSI chip or FPGA
  - butterfly time becomes the basic computational unit

SIZE	SPLIT-RADIX*	RADIX 2*	RADIX 4
64	68	80	48
256	356	512	256
1024	1764	2560	1280

** assumes butterfly unit also capable of dual radix-2 computation*

*** assumes butterfly unit performs two radix-2 butterflies simultaneously*

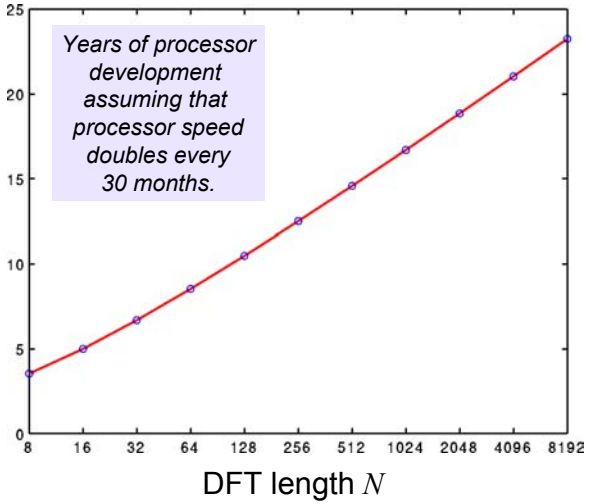
- Split radix now much less attractive than radix 4 Cooley-Tukey
  - in addition,irregular structure makes pipelining less efficient

Processor Speed *vs.* Algorithm Cleverness

- The FFT algorithm gives us a speedup of

$$\frac{N^2}{N\log_2 N}$$

- Hardware improvements provide a 2x speedup every ~30 months



All About FFT Algorithms

- DFT/FFT and Convolution Algorithms and Implementation
  - by C. S. Burrus, T. W. Parks (January 4, 1985)
  - John Wiley & Sons; ISBN: 0471819328

- Algorithms for Discrete Fourier Transform and Convolution (Signal Processing and Digital Filtering)
  - by Richard Tolimieri, Myoung An, Chao Lu, C. S. Burrus (Editor)
  - 2nd edition (January 15, 1997) Springer Verlag; ISBN: 0387982612