Computing the Fast Fourier Transforms on the Connection Machine

The Fourier Transform

- **A Fourier Transform**
  converts shapes to sums of sines & cosines.

- **The DFT (Discrete Fourier Transform)**
  represents shapes and trig. sums by vectors.

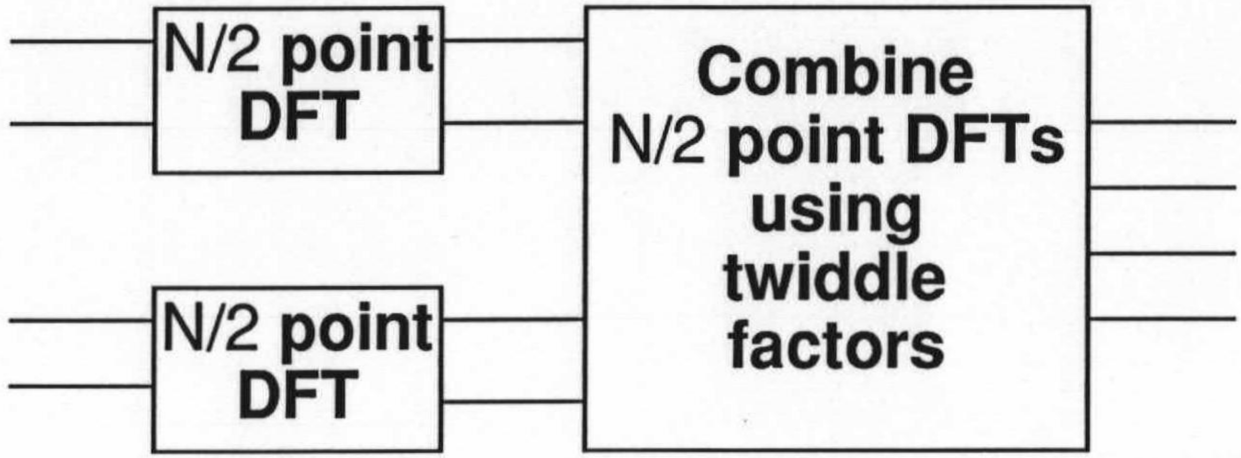
- **The FFT (Fast Fourier Transform)**
  computes the DFT in $n \cdot \log n$ **time.**

- **Other uses include**
  - **filtering of signals and patterns**
  - **solution of partial differential equations**
  - **multiplication of large integers.**

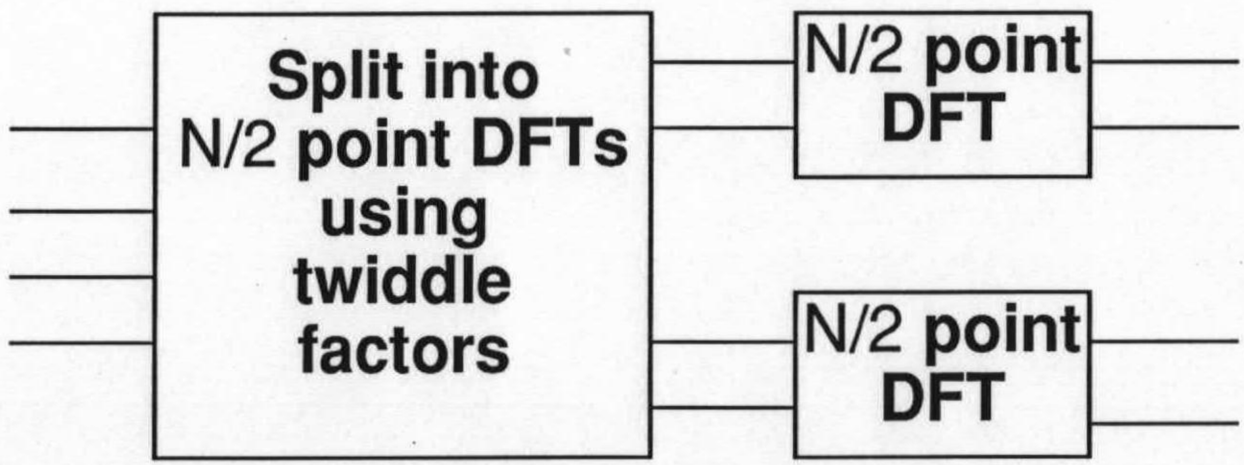
J. W. Cooley and J. W. Tukey, 1965

FFT Butterfly Circuits

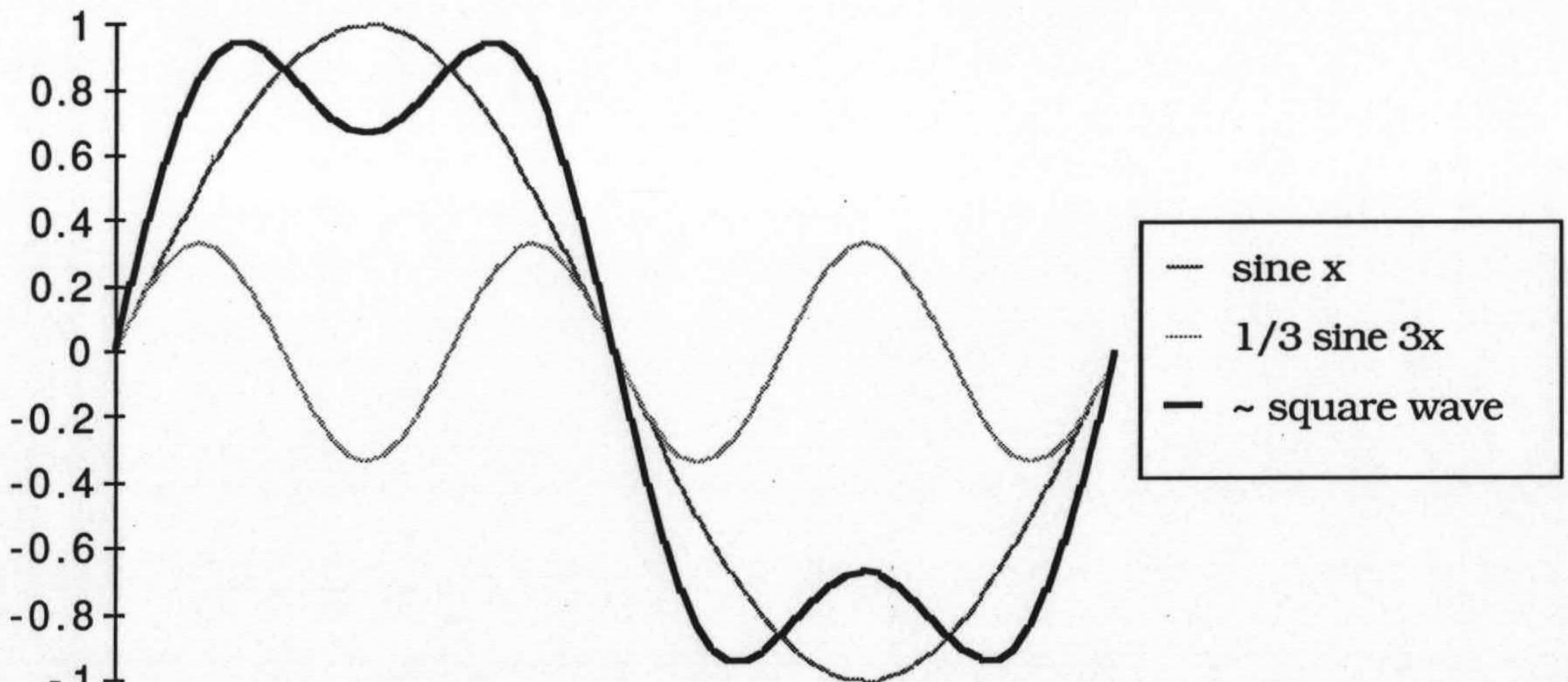
Decimation in time recursion



Decimation in frequency recursion

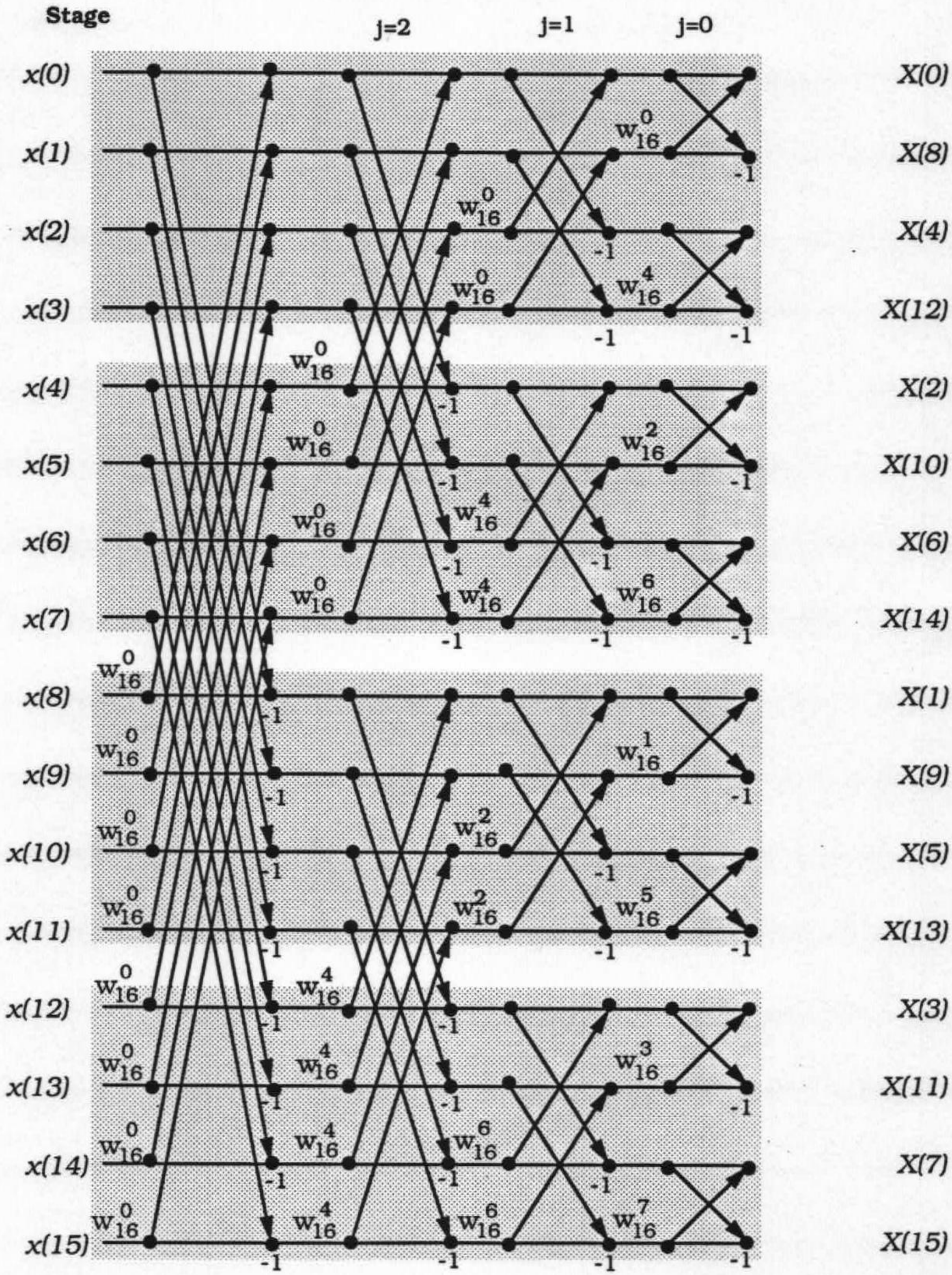


Sums of sines & cosines can model arbitrary shapes



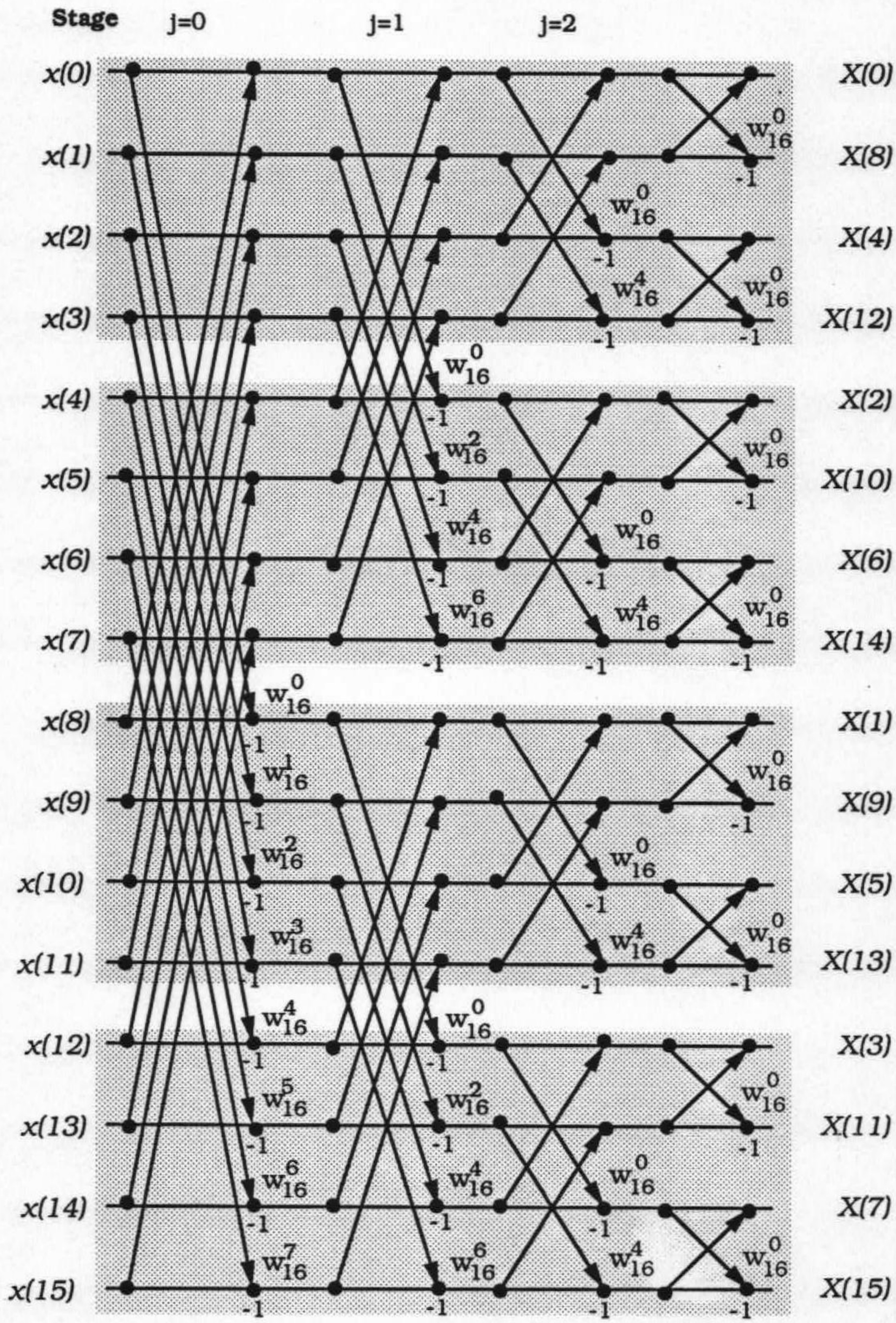
Decimation-in-time

Inputs in normal order



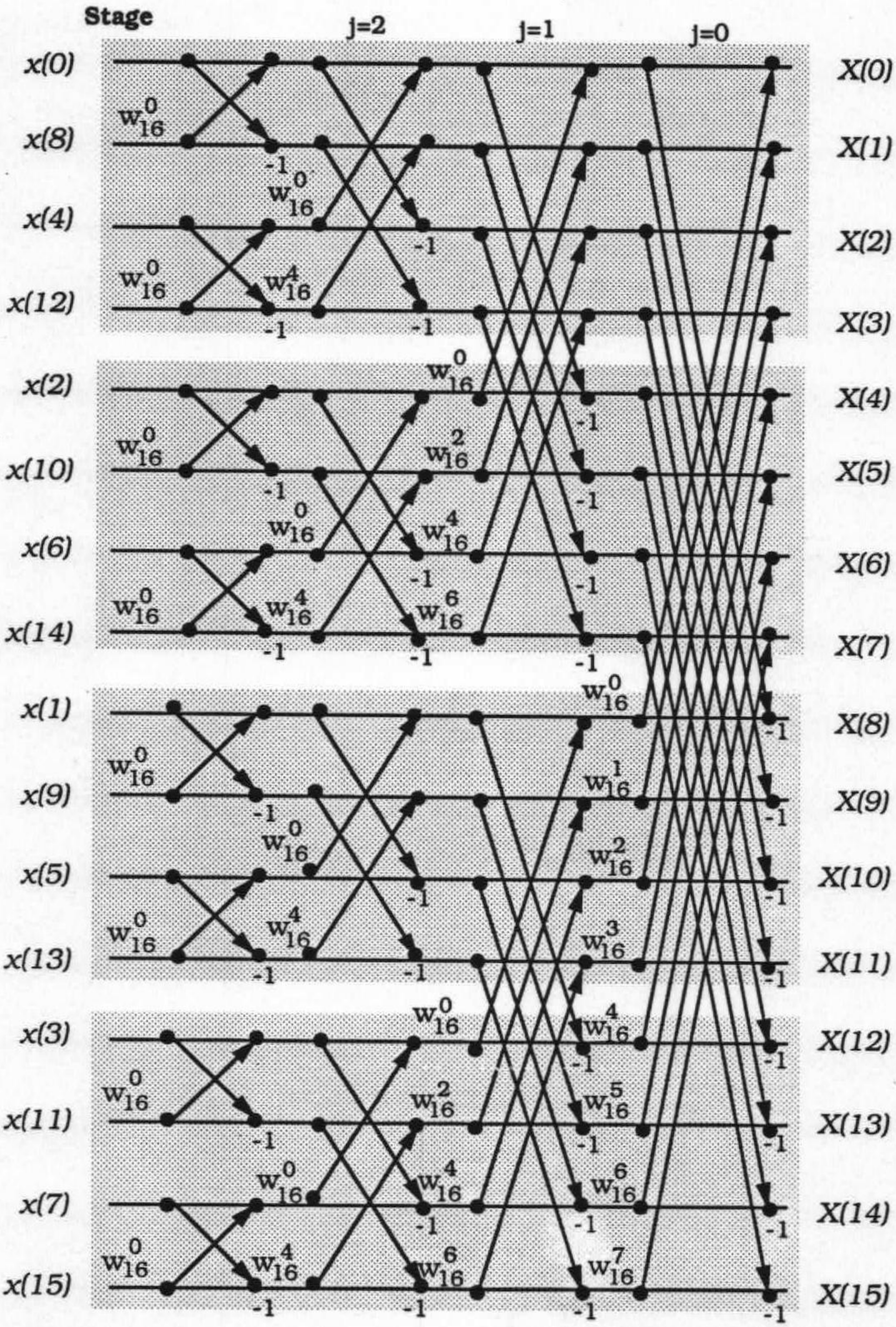
Decimation-in-frequency

Inputs in normal order



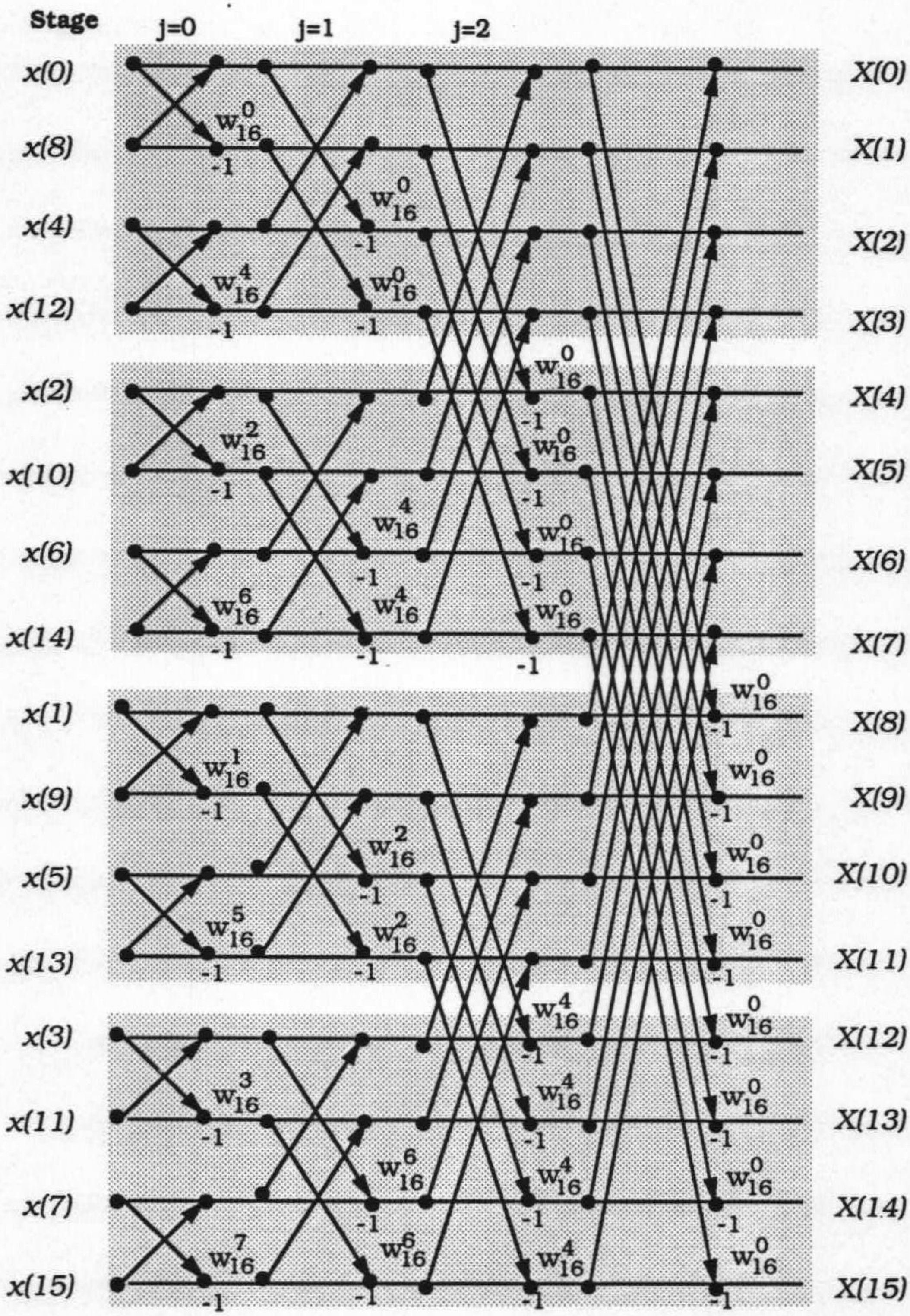
Decimation-in-time

Inputs in bit-reversed order



Decimation-in-frequency

Inputs in bit-reversed order



Twiddle Factors

Proper Choice of FFT and Twiddle storage:

N/P-1+logP per processor, Total = N+P(logP-1)

Twiddles: processor address $(a_{n-1} \ldots a_0)$; stage j

bit-reversed

DIT, normal order: if $a_j=1$, then index = $2^j(a_{j+1} \cdots a_{n-1})$

DIF, normal order: if $a_{n-1-j}=1$, then index = $2^j(a_{n-2-j} \cdots a_0)$

DIT, bit-reversed order: if $a_{n-1-j}=1$, then index = $2^j(a_{n-2-j} \cdots a_0)$

DIT, bit-reversed order: if $a_j=1$, then index = $2^j(a_{j+1} \cdots a_{n-1})$

bit-reversed

Exploiting 90-degree rotations: Total = N/2 +P(logP-1)

Poor choice of FFT and storage: (N-P)logP

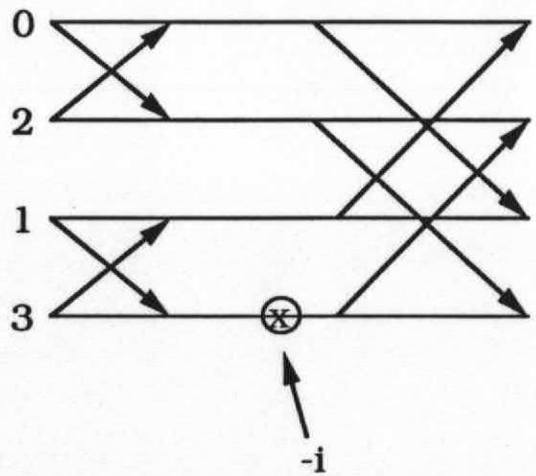
Example: N=16 million, P=2048

Proper choice 64 Mbytes
Poor choice 2 Gbytes

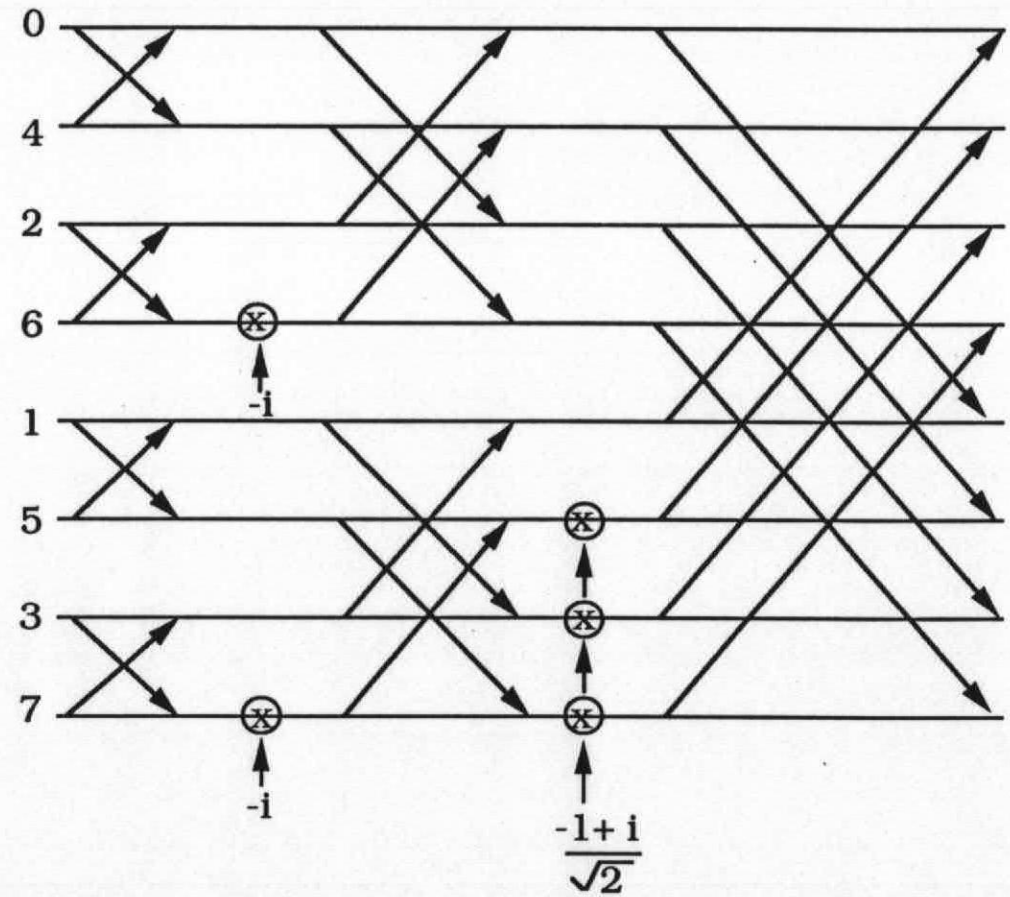
Bit-reversal

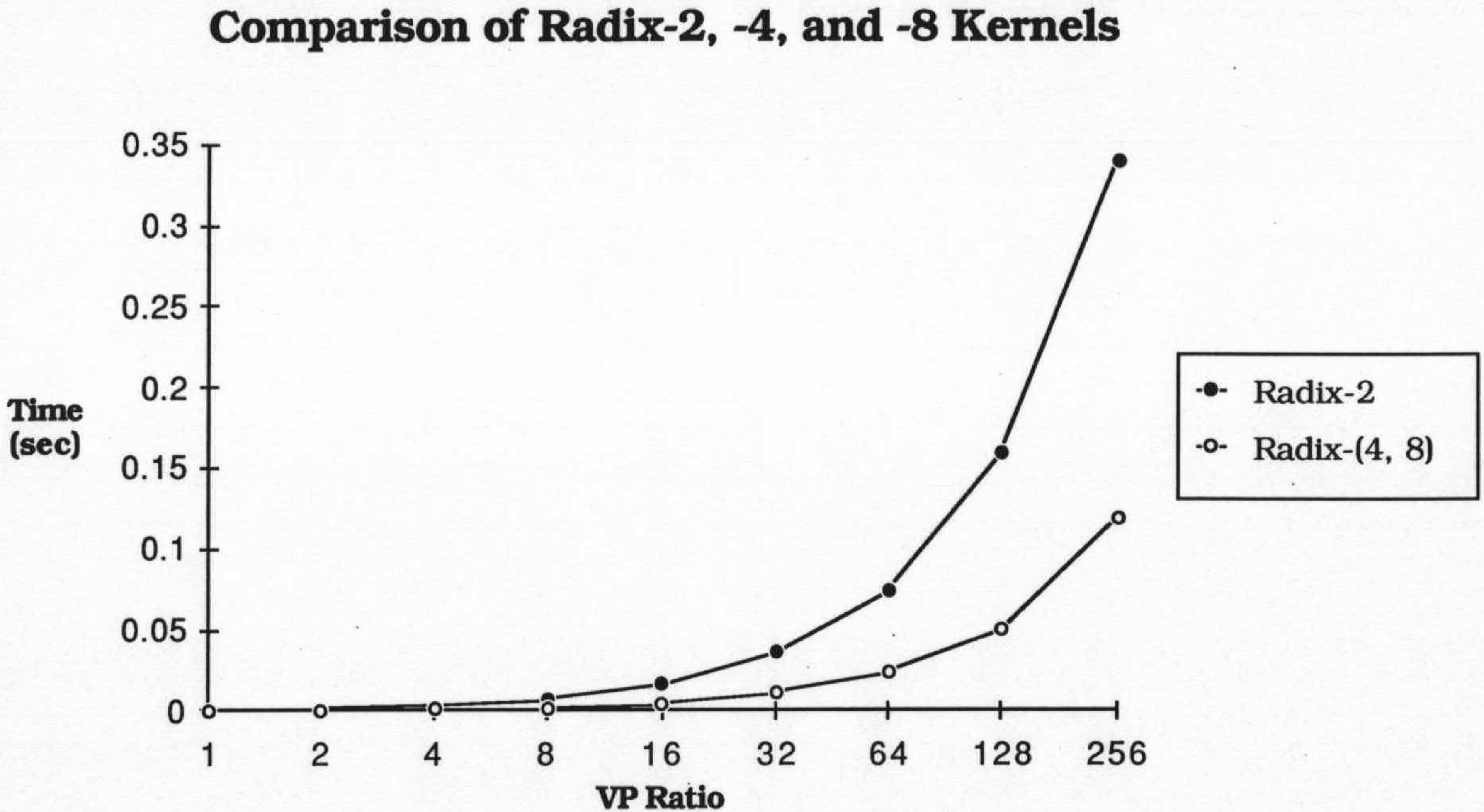
Integer	Binary	Binary	Integer
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15

Radix-4



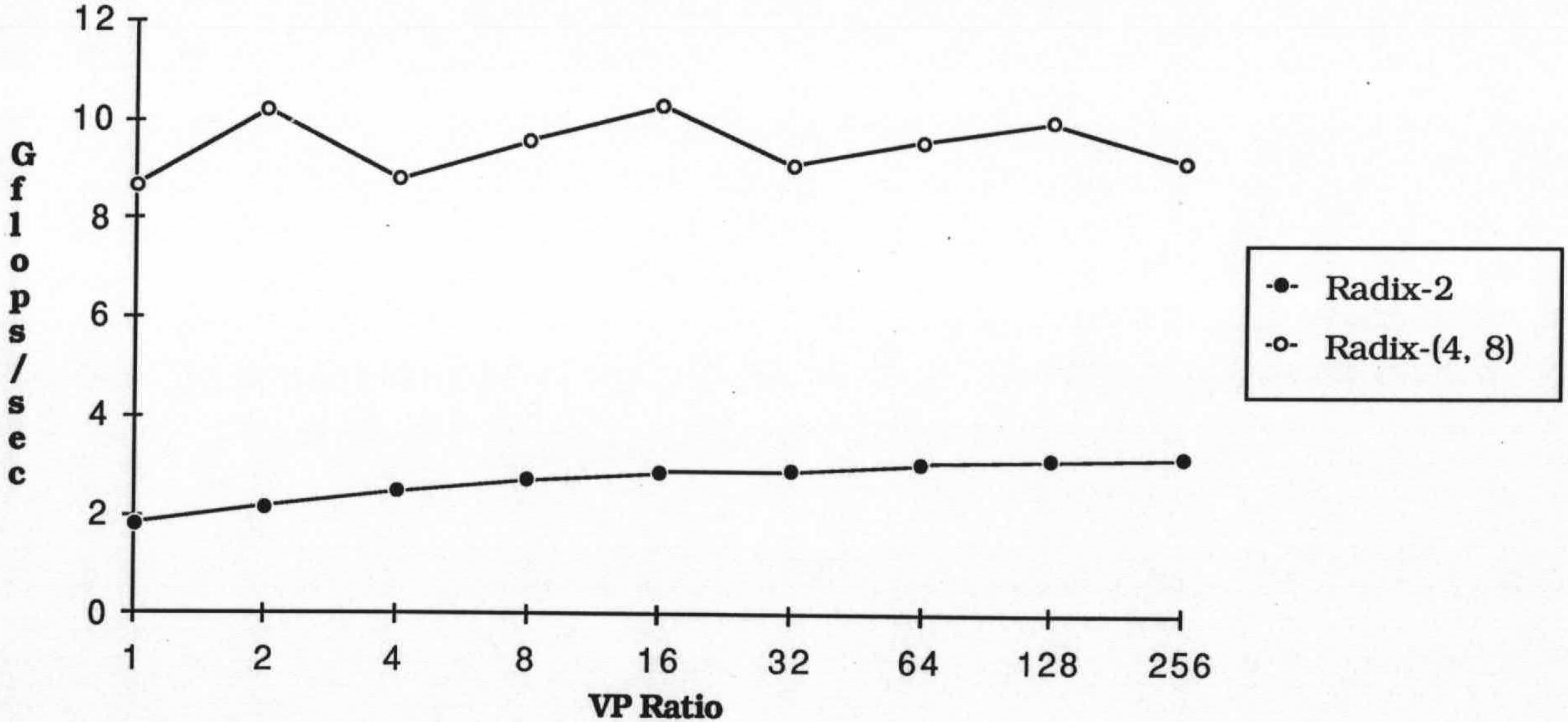
Radix-8

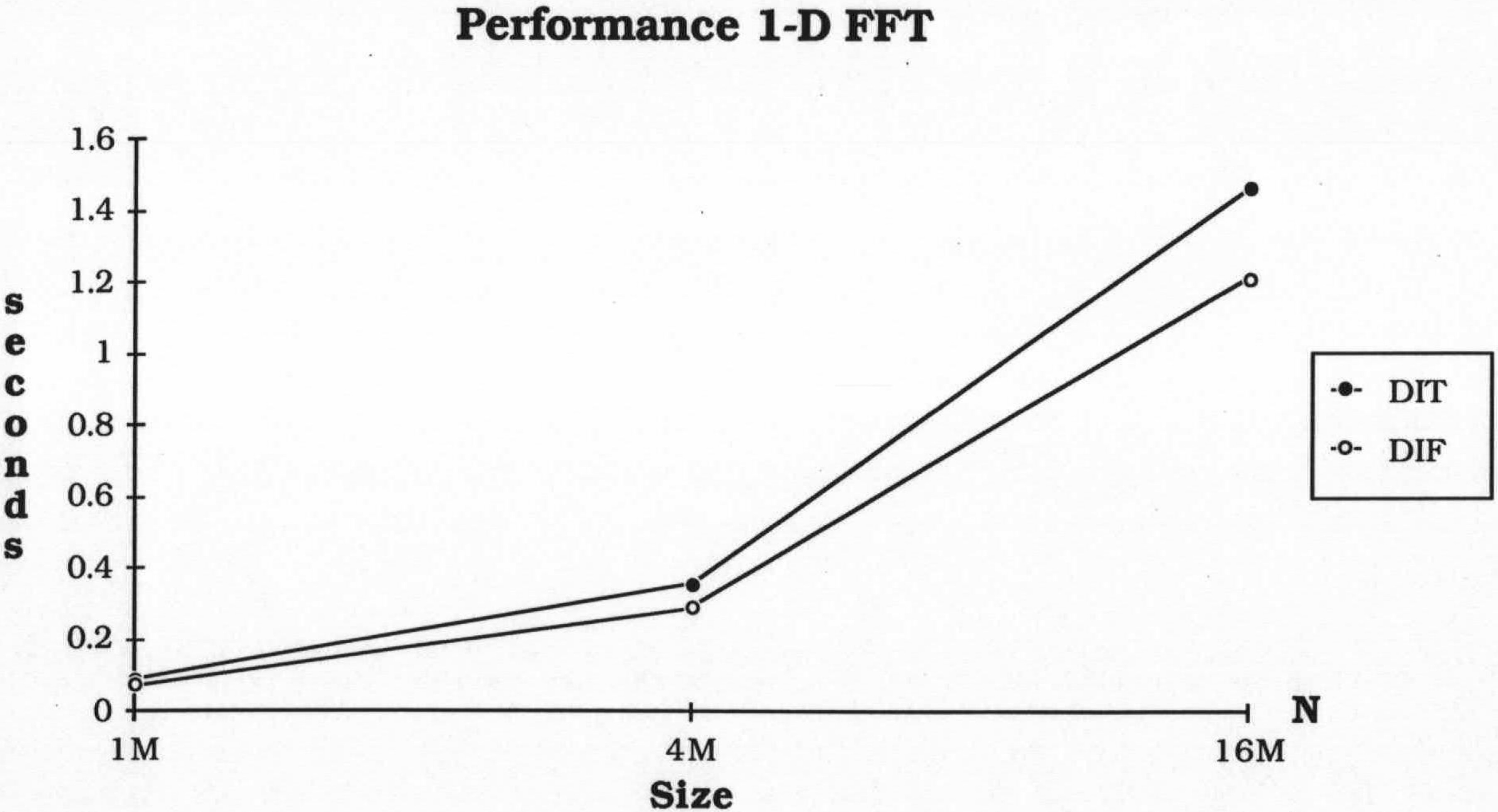




Comparison of Radix-2, -4, and -8 Kernels
Time (sec)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
4
8
16
32
64
128
256
VP Ratio
Radix-2
Radix-(4, 8)

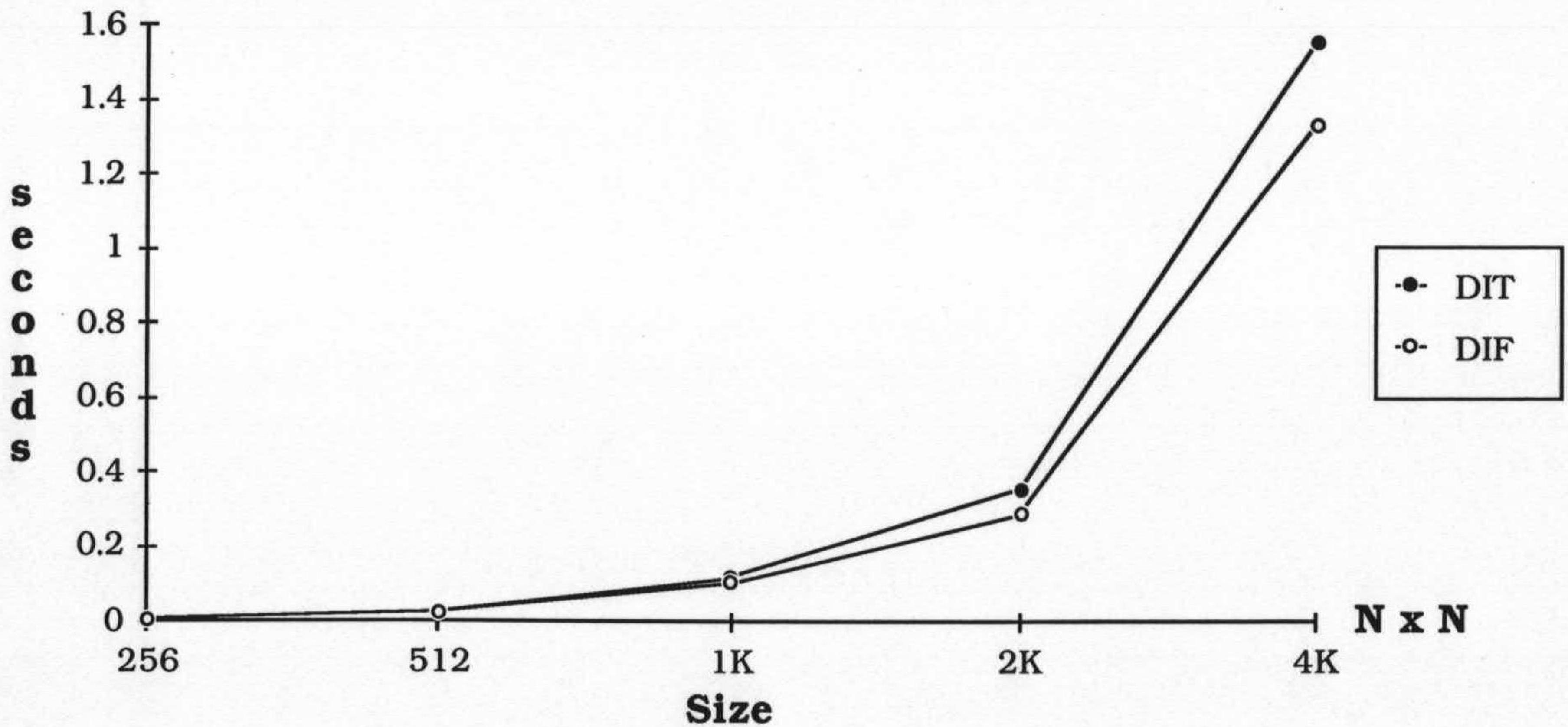
Comparison of Radix-2, -4, and -8 Kernels

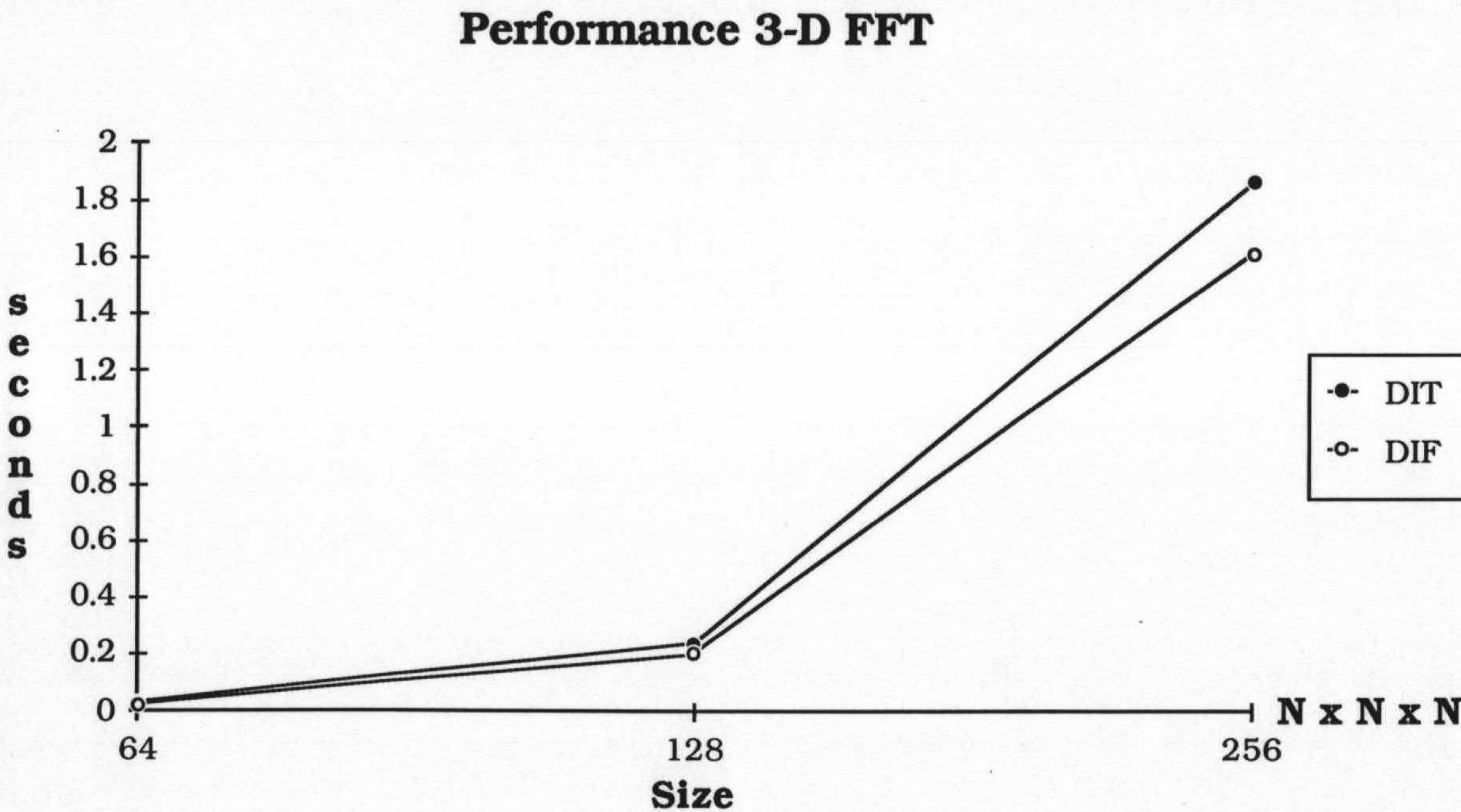




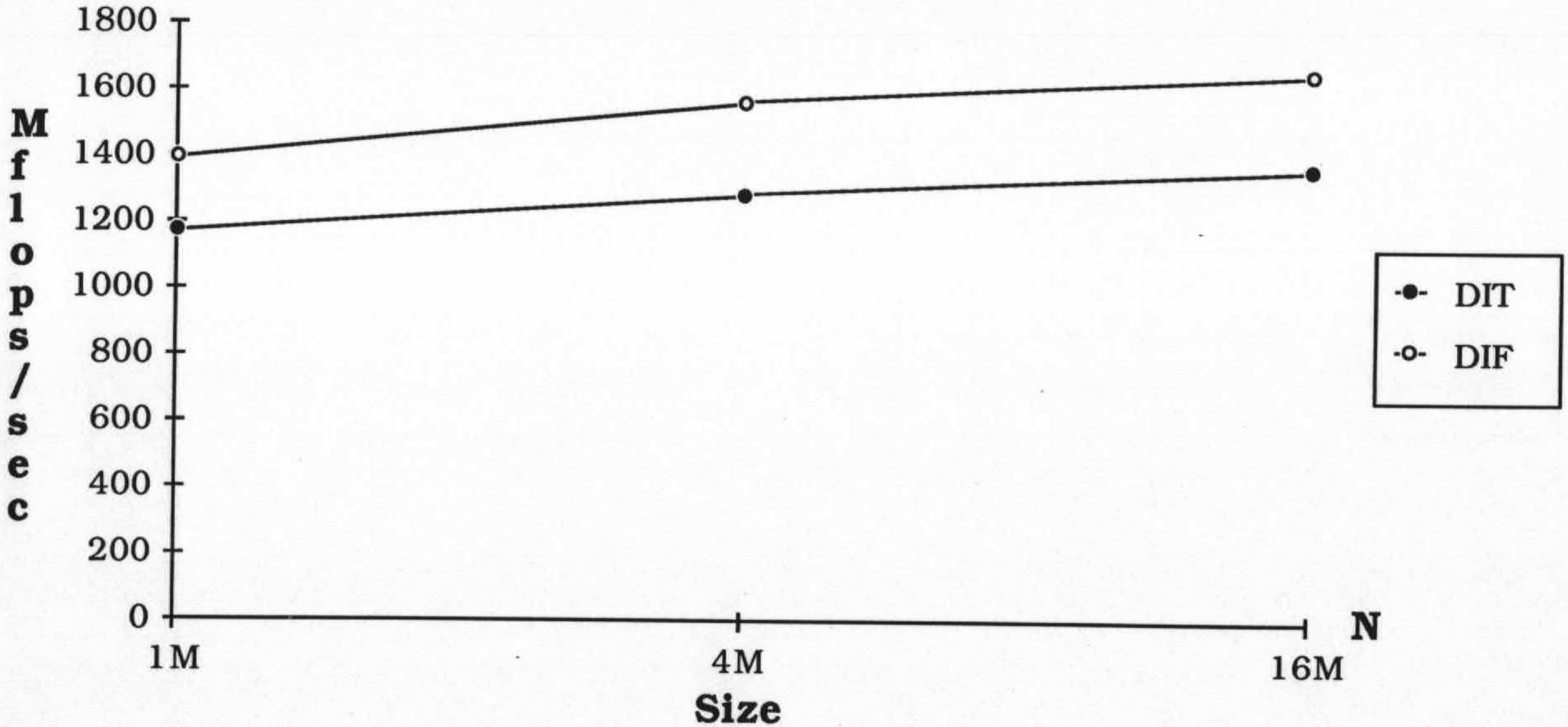
Performance 1-D FFT
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
seconds
1M
4M
16M
N
Size
DIT
DIF

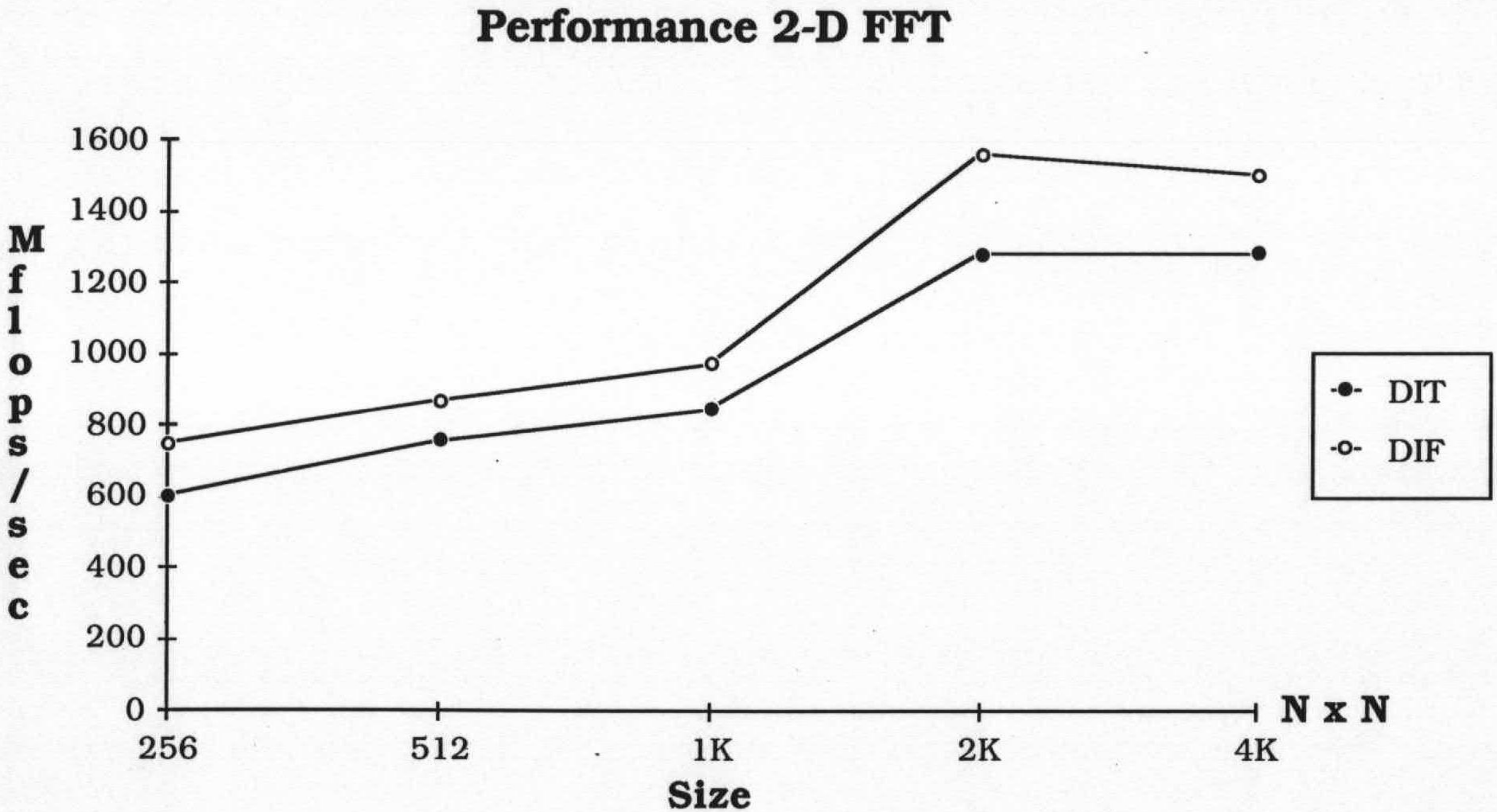
Performance 2-D FFT



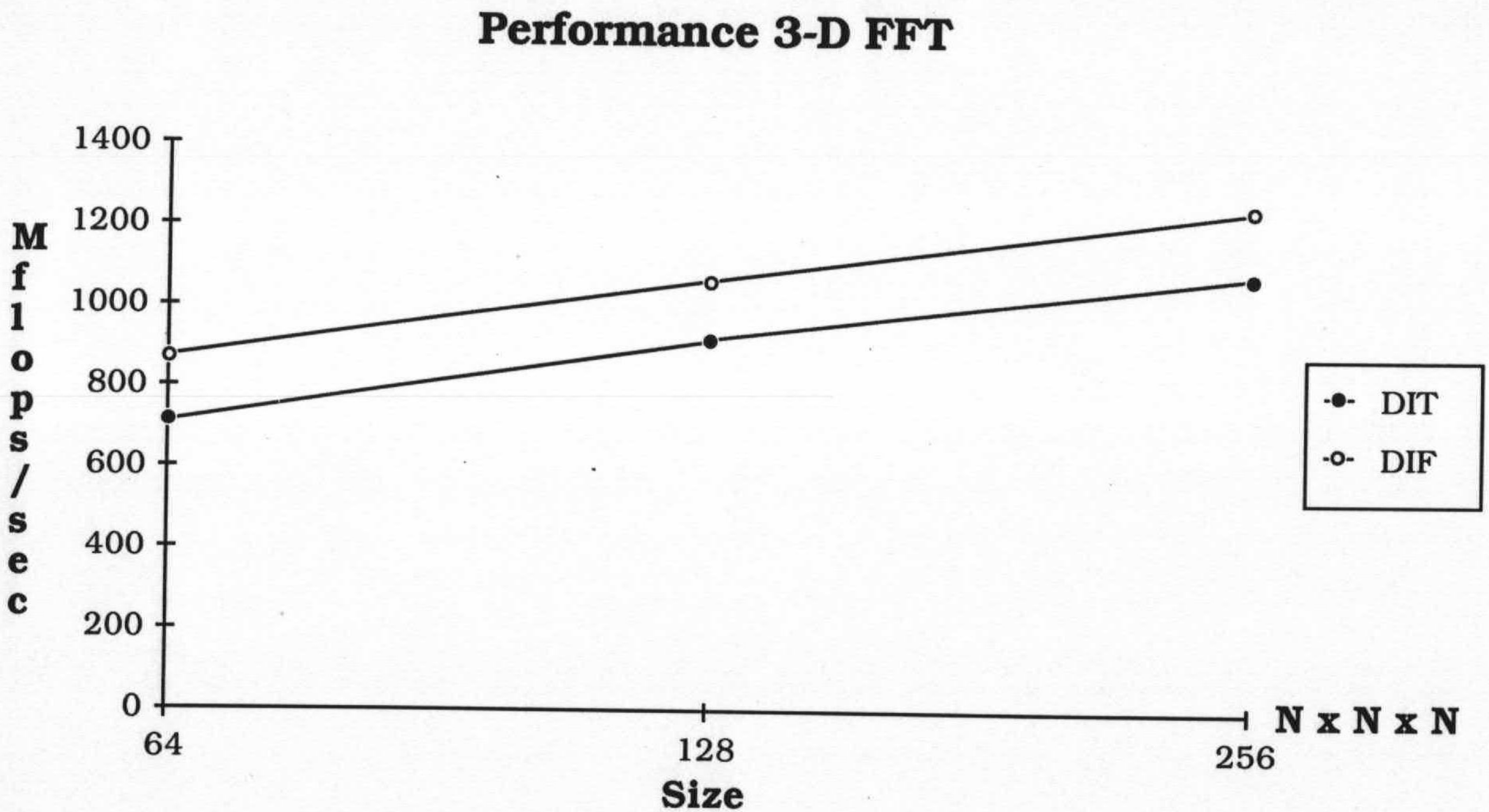

Performance 3-D FFT
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
seconds
DIT
DIF
N x N x N
64
128
256
Size

Performance 1-D FFT





Performance 2-D FFT
Mflops/sec
1600
1400
1200
1000
800
600
400
200
0
256
512
1K
2K
4K
N x N
Size
DIT
DIF


Performance 3-D FFT
Mflops/sec
1400
1200
1000
800
600
400
200
0
64
128
256
N x N x N
Size
DIT
DIF

Multi-dimensional Fourier Transforms

$$X(k_1,k_2) = \sum_{j_1=0}^{N_1-1} \sum_{j_2=0}^{N_2-1} \omega_{N_1}^{j_1 k_1} \omega_{N_2}^{j_2 k_2} x(j_1,j_2)$$

$$= \sum_{j_1=0}^{N_1-1} \omega_{N_1}^{j_1 k_1} \underbrace{\sum_{j_2=0}^{N_2-1} \omega_{N_2}^{j_2 k_2} x(j_1,j_2)}_{Z(j_1,k_2)}$$

$$X(k_1,k_2) = \sum_{j_1=0}^{N_1-1} \omega_{N_1}^{j_1 k_1} Z(j_1,j_2)$$