Fast Fourier Transform

CLRS

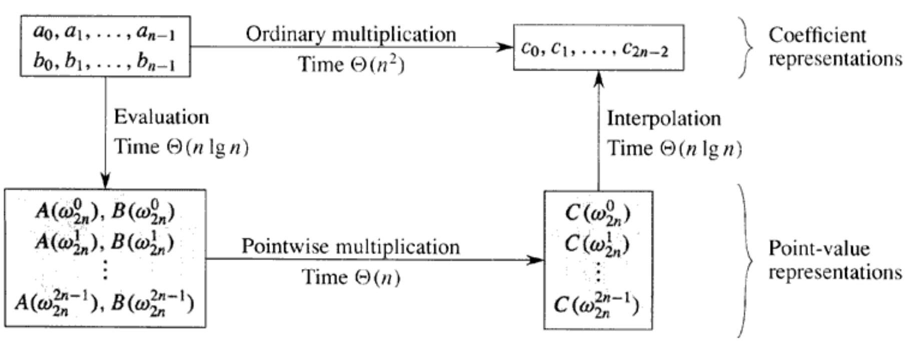

Figure 30.1 A graphical outline of an efficient polynomial-multiplication process. Representations on the top are in coefficient form, while those on the bottom are in point-value form. The arrows from left to right correspond to the multiplication operation. The ω_{2n} terms are complex $(2n)$th roots of unity.

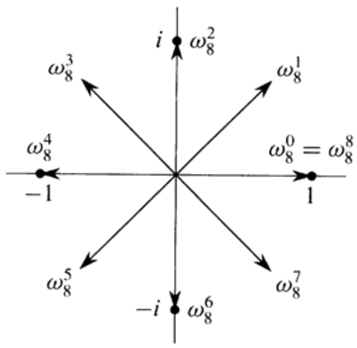

Figure 30.2 The values of $\omega_8^0, \omega_8^1, \ldots, \omega_8^7$ in the complex plane, where $\omega_8 = e^{2\pi i/8}$ is the principal 8th root of unity.

```
RECURSIVE-FFT(a)
n ← length[a]                      ▷ n is a power of 2.
if n = 1
   then return a
ω_n ← e^{2πi/n}
ω ← 1
a^[0] ← (a_0, a_2, ..., a_{n-2})
a^[1] ← (a_1, a_3, ..., a_{n-1})
y^[0] ← RECURSIVE-FFT(a^[0])
y^[1] ← RECURSIVE-FFT(a^[1])
for k ← 0 to n/2 - 1
     do y_k ← y_k^[0] + ω y_k^[1]
        y_{k+(n/2)} ← y_k^[0] - ω y_k^[1]
        ω ← ω ω_n
return y                           ▷ y is assumed to be a column vector.
```

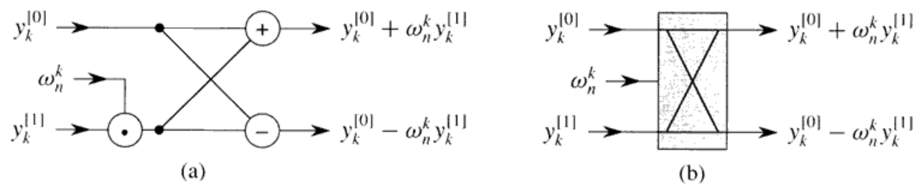

Figure 30.3 A butterfly operation. (a) The two input values enter from the left, the twiddle factor ω_n^k is multiplied by $y_k^{[1]}$, and the sum and difference are output on the right. (b) A simplified drawing of a butterfly operation. We will use this representation in a parallel FFT circuit.

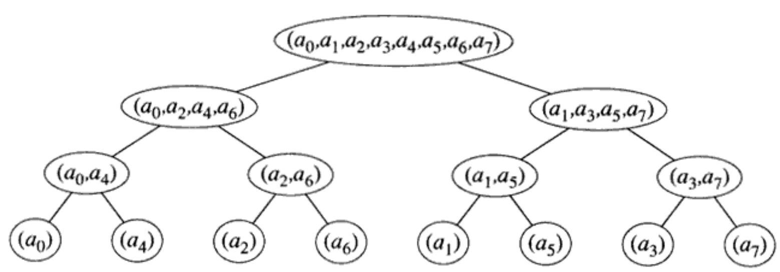

Figure 30.4 The tree of input vectors to the recursive calls of the RECURSIVE-FFT procedure. The initial invocation is for $n = 8$.

```
ITERATIVE-FFT(a)
BIT-REVERSE-COPY(a, A)
n ← length[a]                    ▷ n is a power of 2.
for s ← 1 to lg n
     do m ← 2^s
        ω_m ← e^{2πi/m}
        for k ← 0 to n − 1 by m
             do ω ← 1
                for j ← 0 to m/2 − 1
                     do t ← ω A[k + j + m/2]
                        u ← A[k + j]
                        A[k + j] ← u + t
                        A[k + j + m/2] ← u − t
                        ω ← ω ω_m
```

```
BIT-REVERSE-COPY(a, A)
n ← length[a]
for k ← 0 to n − 1
     do A[rev(k)] ← a_k
```

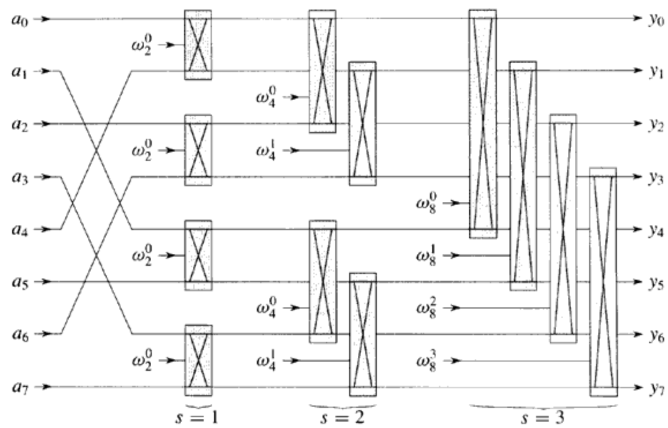

Figure 30.5 A circuit PARALLEL-FFT that computes the FFT, here shown on $n = 8$ inputs. Each butterfly operation takes as input the values on two wires, along with a twiddle factor, and it produces as outputs the values on two wires. The stages of butterflies are labeled to correspond to iterations of the outermost loop of the ITERATIVE-FFT procedure. Only the top and bottom wires passing through a butterfly interact with it; wires that pass through the middle of a butterfly do not affect that butterfly, nor are their values changed by that butterfly. For example, the top butterfly in stage 2 has nothing to do with wire 1 (the wire whose output is labeled y_1); its inputs and outputs are only on wires 0 and 2 (labeled y_0 and y_2, respectively). An FFT on n inputs can be computed in $\Theta(\lg n)$ depth with $\Theta(n \lg n)$ butterfly operations.