Course Notes 11 – Computation of the DFT

11.0 Introduction

The FFT is a class of fast computational algorithms for the DFT. Many types of FFT algorithms exist.

Why is a fast DFT needed?

Consider the complexity of the DFT (*i.e.*, the number of multiplies and additions required)

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

Recognize that $x(n)$ and $X(k)$ are complex in general, then

$$X(k) = \sum_{n=0}^{N-1} \left[\mathrm{Re}\{x(n)\} + j\,\mathrm{Im}\{x(n)\} \right] \cdot \left[\mathrm{Re}\left\{W_N^{kn}\right\} + j\,\mathrm{Im}\left\{W_N^{kn}\right\} \right] \qquad 0 \le k \le N-1$$

$$= \sum_{n=0}^{N-1} \left[\mathrm{Re}\{x(n)\} \cdot \mathrm{Re}\left\{W_N^{kn}\right\} - \mathrm{Im}\{x(n)\} \cdot \mathrm{Im}\left\{W_N^{kn}\right\} + j\,\mathrm{Re}\{x(n)\}\,\mathrm{Im}\left\{W_N^{kn}\right\} + j\,\mathrm{Im}\{x(n)\}\,\mathrm{Re}\left\{W_N^{kn}\right\} \right] \qquad 0 \le k \le N-1$$

The significance: A measure of the number of complex arithmetic operations

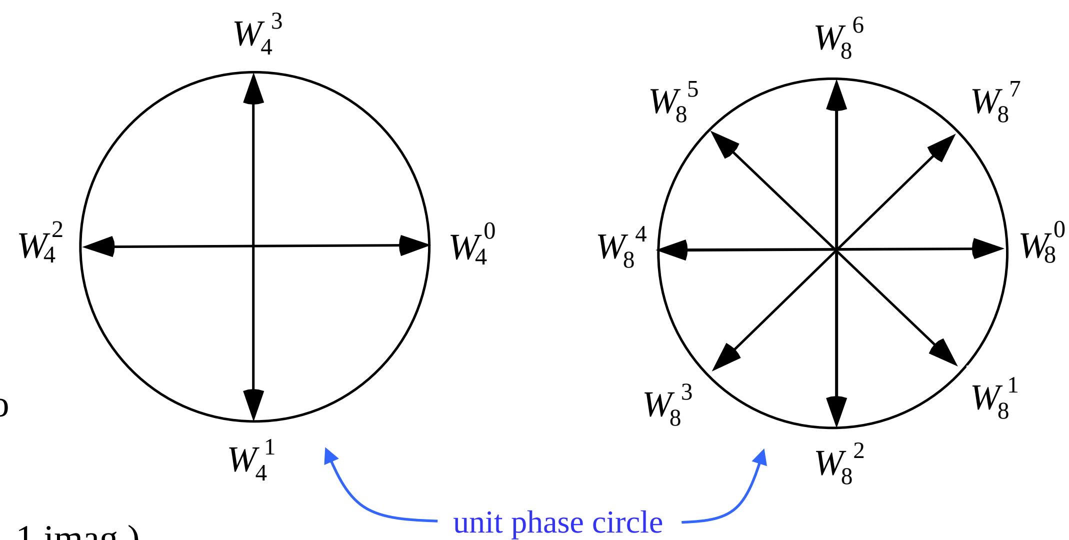
N	N^2 complex mult.
16	256
128	16,384
1,024	1,048,576
4,096	16,777,216

Therefore, we need a more efficient implementation.

11.1 Efficient Computation of the DFT

Most FFT algorithms exploit one or both of the following properties:

- $W_N^{k(N-n)} = \left(W_N^{kn}\right)^*$ conjugate symmetry $\left(\text{i.e, } W_N^{k+N/2} = -W_N^{k}\right)$ half-cycle
- $W_N^{((kn))_N} = W_N^{kn}$ periodicity $\left(\text{i.e, } W_N^{k+N} = W_N^{k}\right)$ full-cycle



For each k and n we have:

- 4 – multiplies (2 real, 2 imag.)
- 2 - adds (1 real, 1 imag.)

So for each k there are N values of n, so

- $4N$ multiplies
- $4N - 2$ adds ($2N - 1$ real, $2N - 1$ imag.)

and for N values of k, there are

- $4N^2$ - multiplies (approx N^2)
- $N(4N - 2)$ - adds (approx N^2)

← This is a measure of the complexity of the DFT

Example: Consider a 4-point DFT

$$X(k) = \sum_{n=0}^{3} x(n)\, W_4^{kn}$$

$$X(0) = x(0)W_4^0 + x(1)W_4^0 + x(2)W_4^0 + x(3)W_4^0$$

$$X(1) = x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3$$

$$X(2) = x(0)W_4^0 + x(1)W_4^2 + x(2)W_4^4 + x(3)W_4^6$$

$$X(3) = x(0)W_4^0 + x(1)W_4^3 + x(2)W_4^6 + x(3)W_4^9$$

12 adds &

16 multiplies

in DFT form

Recognizing that (by symmetry):

$$W_4^0 = -W_4^2$$

$$W_4^1 = -W_4^3$$

and also that (by periodicity):

$$W_4^4 = W_4^0$$

$$W_4^6 = W_4^2 = -W_4^0$$

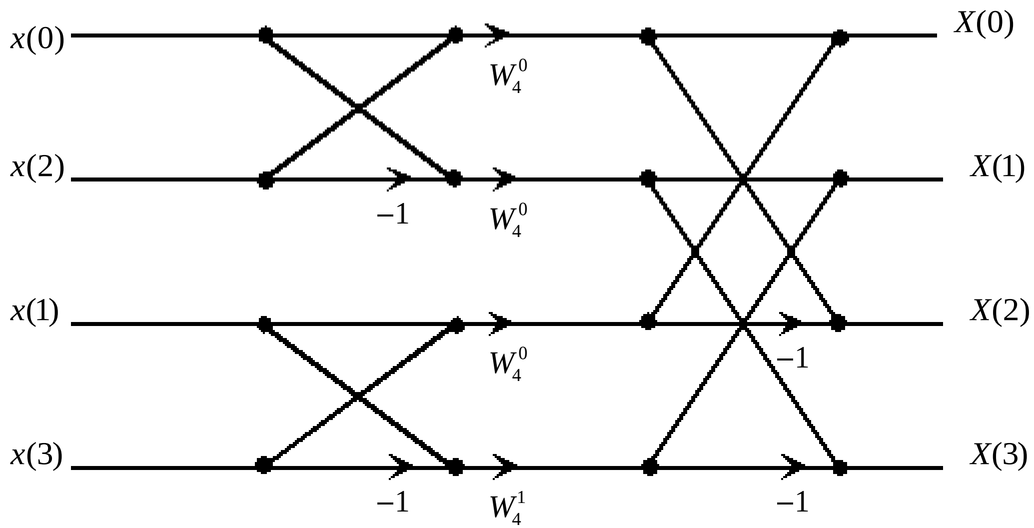
$$W_4^9 = W_4^1$$

express all W_4^r in terms of W_4^0 and W_4^1

$$X(0) = x(0)W_4^0 + x(1)W_4^0 + x(2)W_4^0 + x(3)W_4^0$$
$$X(1) = x(0)W_4^0 + x(1)W_4^1 - x(2)W_4^0 - x(3)W_4^1$$
$$X(2) = x(0)W_4^0 - x(1)W_4^0 + x(2)W_4^0 - x(3)W_4^0$$
$$X(3) = x(0)W_4^0 - x(1)W_4^1 - x(2)W_4^0 + x(3)W_4^1$$

$$X(0) = [x(0) + x(2)]W_4^0 + [x(1) + x(3)]W_4^0$$
$$X(1) = [x(0) - x(2)]W_4^0 + [x(1) - x(3)]W_4^1$$
$$X(2) = [x(0) + x(2)]W_4^0 - [x(1) + x(3)]W_4^0$$
$$X(3) = [x(0) - x(2)]W_4^0 - [x(1) - x(3)]W_4^1$$

Now draw the network flowgraph that implements this set of operations:



This implementation requires <u>8 adds</u> and <u>4 multiplies</u> instead of <u>12 adds</u> and <u>16 multiplies</u> in DFT form.

Example: If one multiply takes 10 μsec on a digital machine, the time to compute a 1024 point DFT is (neglecting add time)

$$4N^2 = 4(1024)^2 \cdot 10\mu\text{sec} = 42\,\text{sec.}$$

210-fold reduction

The FFT requires only $2N \cdot \log_2(N)$ multiplies

$$2N \cdot \log_2(N) = 2(1024)(10) \cdot 10\mu\text{sec} = 0.2\,\text{sec.}$$

This savings is dramatic and has been partly responsible for the growth and popularity of DSP.

11.2 The Goertzel Algorithm

- Allows computation of the DFT as a linear filtering operation
- Evaluates only a single frequency (a "tone" detector)

Begin with:

$$X(k) = \sum_{m=0}^{N-1} x(m) W_N^{km} \qquad 0 \le k \le N-1$$

Multiply by W_N^{-kN}

$$W_N^{-kN} X(k) = W_N^{-kN} \sum_{m=0}^{N-1} x(m) W_N^{km} \qquad 0 \le k \le N-1$$

noting that

$$W_N^{-kN} = e^{\frac{+j2\pi kN}{N}} = e^{+j2\pi k} = 1$$

so =

$$X(k) = \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)}$$

which <u>looks like a convolution</u> operation. Indeed if we express this as:

$$y_k(n) = \sum_{m=0}^{N-1} x(m) W_N^{-k(n-m)}$$

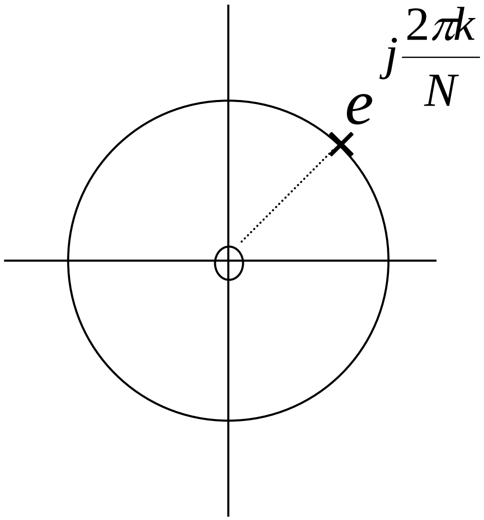
$$y_k(n) = x(n) * W_N^{-kn} u(n)$$

Output sequence — Input sequence — Impulse response, $h_k(n)$

Then, $X(k) = y_k(n)\big|_{n=N}$

with, $h_k(n) = W_N^{-kn} u(n)$

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{Y_k(z)}{X_k(z)}$$



From $H_k(z)$ we can obtain an algorithm:

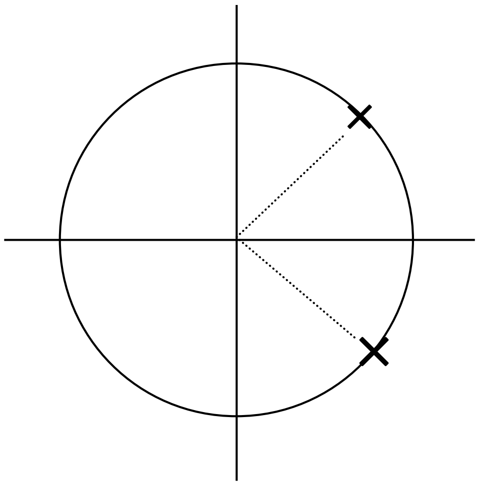
$$y_k(n) = W_N^{-k} y_k(n-1) + x(n), \qquad y_k(-1) = 0$$

and the desired output is: @ $n = N$

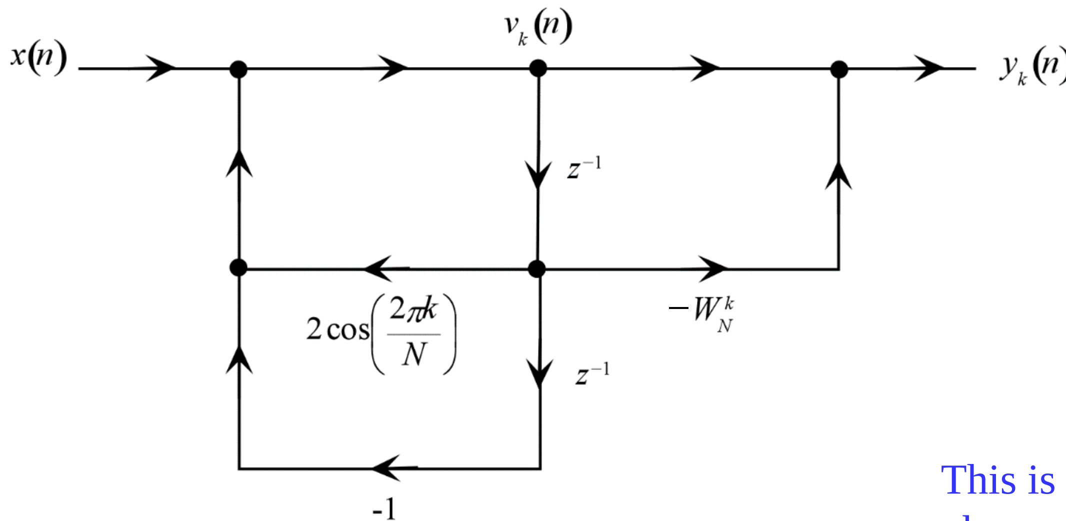
$$X(k) = y_k(N), \qquad \text{for } k = 0, 1, \cdots, N-1$$

Complex multiplication is not desirable. The same results can be obtained by allowing a conjugate pole:

$$H_k(k) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}}$$



Which is implemented as:



Computed N+1 times

$$v_k(n) = 2\cos\left(\frac{2\pi k}{N}\right) v_k(n-1) - v_k(n-2) + x(n)$$

Computed only once @ $n = N$

$$y_k(N) = v_k(N) - W_N^k v_k(N-1)$$

This is how the phone system determines which number you dial, since each digit corresponds to a pair of freqs. and the rest don't matter

Since finite and defined over $x(0)$ to $x(N$-1), the $x(N)$ term is 0

11.3 The Decimation-in-Time FFT Algorithm

Remember, the standard to which we compare any FFT algorithm is the so-called "direct-form" (*i.e.*, the DFT, which can also be viewed as a matrix multiply).

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

There are many FFT techniques. The most popular approach (from Cooley and Tukey) is based on decomposing the DFT into smaller transforms and then combining them to give the total transform.

In the following development, we assume that the number of points to be transformed is a power of 2. FFTs designed with this assumption are called <u>radix-2 FFTs</u>. That is,

$$N = 2^{\nu}$$

The approach is to break the N-point transform into…

- two $N/2$ - point sequences, then into…
  - four $N/4$ - point sequences, then into…
    - eight $N/8$ - point sequences, then into…
      - etc., until only 2-point sequences are obtained ($N/2$ of them).

Begin with the DFT:

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

Break into two parts, one with <u>odd index values</u> and one with <u>even index values</u>, as

$$X(k) = \sum_{\substack{n=0 \\ (n \text{ even})}}^{N-2} x(n) W_N^{kn} + \sum_{\substack{n=1 \\ (n \text{ odd})}}^{N-1} x(n) W_N^{kn}$$

Let $n = 2r$ (even sum) Let $n = 2r + 1$ (odd sum)

$n = 2r$ $\quad\quad$ $n = 2r + 1$

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

Note that: $$W_N^{2kr} = e^{-j\frac{2\pi rk}{N}(2)} = e^{-j\frac{2\pi rk}{N/2}} = W_{N/2}^{rk}$$

Now <u>half</u> the number of points around the unit phase circle

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

N/2 - point DFT of even indexed sequence, *G*(*k*)

N/2 - point DFT of odd indexed sequence, *H*(*k*)

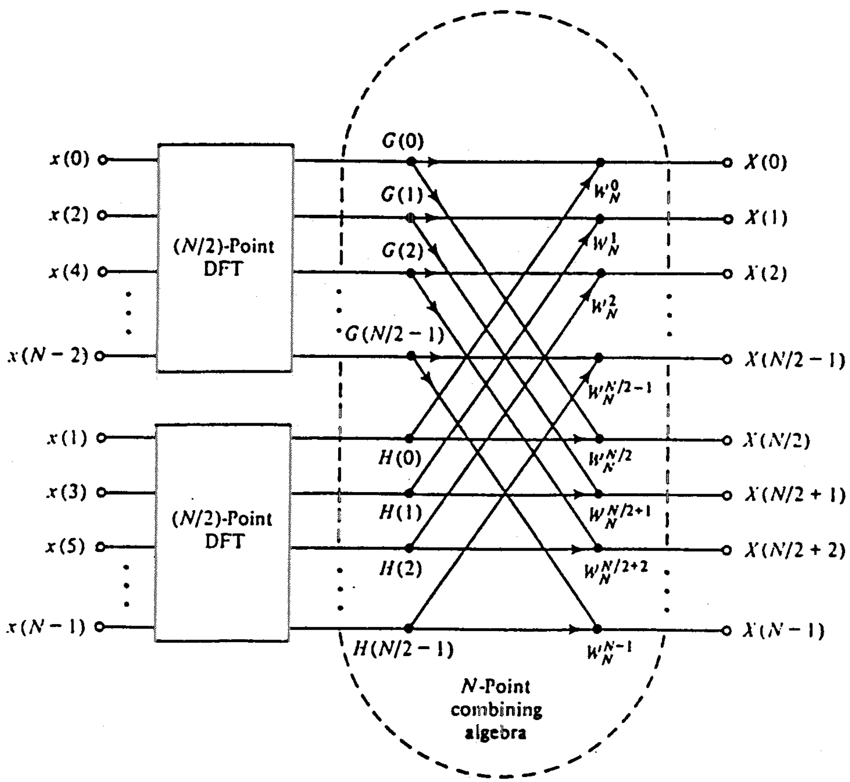
$$X(k) = G(k) + W_N^k\, H(k) \qquad 0 \le k \le N-1$$

But note that *G*(*k*) and *H*(*k*) are <u>*N*/2 - point DFTs</u>

Due to the periodic nature of the DFT, an equivalent expression for $X(k)$ is:

$$X(k)=\begin{cases} G(k)+W_N^k H(k) & 0\le k\le N/2-1 \\ G(k-N/2)+W_N^k H(k-N/2) & N/2\le k\le N-1 \end{cases}$$

The first stage decomposition can be implemented as:



At this point, with

$$X(k) = G(k) + \underline{W_N^k}\ H(k)$$

$G(k)$: $\left(\frac{N}{2}\right)^2$ complex multiplies

$\underline{W_N^k}$: N complex multiplies

$H(k)$: $\left(\frac{N}{2}\right)^2$ complex multiplies

the # of complex multiplies after first decomposition is

$$\eta_1 = \left(\frac{N}{2}\right)^2 + \left(\frac{N}{2}\right)^2 + N = N + \frac{N^2}{2}$$

Next, we can compute each <u>N/2 - point DFT</u> by breaking it into the sum into <u>two N/4 - point DFTs</u> as:

$$G(k) = \sum_{r=0}^{N/2-1} g(r)\ W_{N/2}^{rk} \qquad \text{where } g(r) = x(2r)$$

$$= \sum_{m=0}^{N/4-1} g(2m)\ W_{N/2}^{2mk} + \sum_{m=0}^{N/4-1} g(2m+1)\ W_{N/2}^{(2m+1)k}$$

$$= \sum_{m=0}^{N/4-1} g(2m)\ W_{N/4}^{mk} + W_{N/2}^{k} \sum_{m=0}^{N/4-1} g(2m+1)\ W_{N/4}^{mk} \qquad \text{for } 0 \le k \le N/2 - 1$$

So $$G(k) = \tilde{G}_1(k) + W_N^{2k}\tilde{H}_1(k)$$

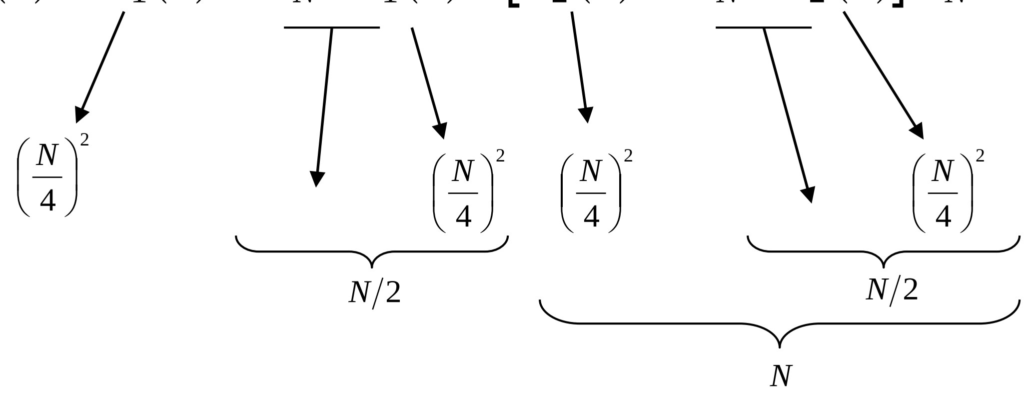
and similarly $$H(k) = \sum_{m=0}^{N/4-1} h(2m)\, W_{N/4}^{mk} + W_N^{2k} \sum_{m=0}^{N/4-1} h(2m+1)\, W_{N/4}^{nk}$$

so $$H(k) = \tilde{G}_2(k) + W_N^{2k}\tilde{H}_2(k)$$

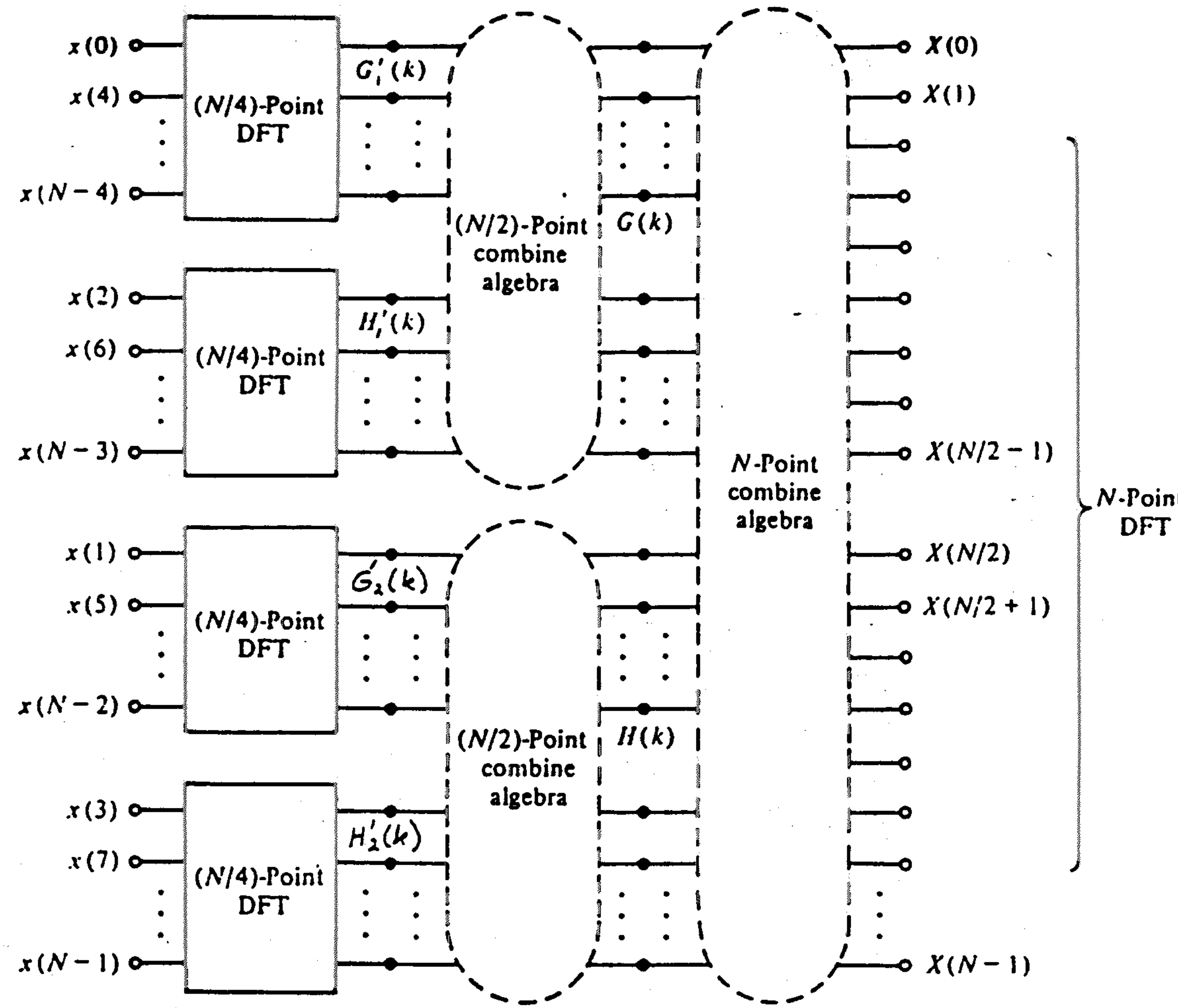
(The second stage decomposition is shown on the next page)

The number of complex multiplies after the second decomposition is therefore

$$X(k) = \tilde{G}_1(k) + W_N^{2k}\tilde{H}_1(k) + \left[\tilde{G}_2(k) + W_N^{2k}\tilde{H}_2(k)\right]W_N^k$$



yielding $$\eta_2 = 4\left(\frac{N}{4}\right)^2 + 2\left(\frac{N}{2}\right) + N = \frac{N^2}{4} + 2N$$

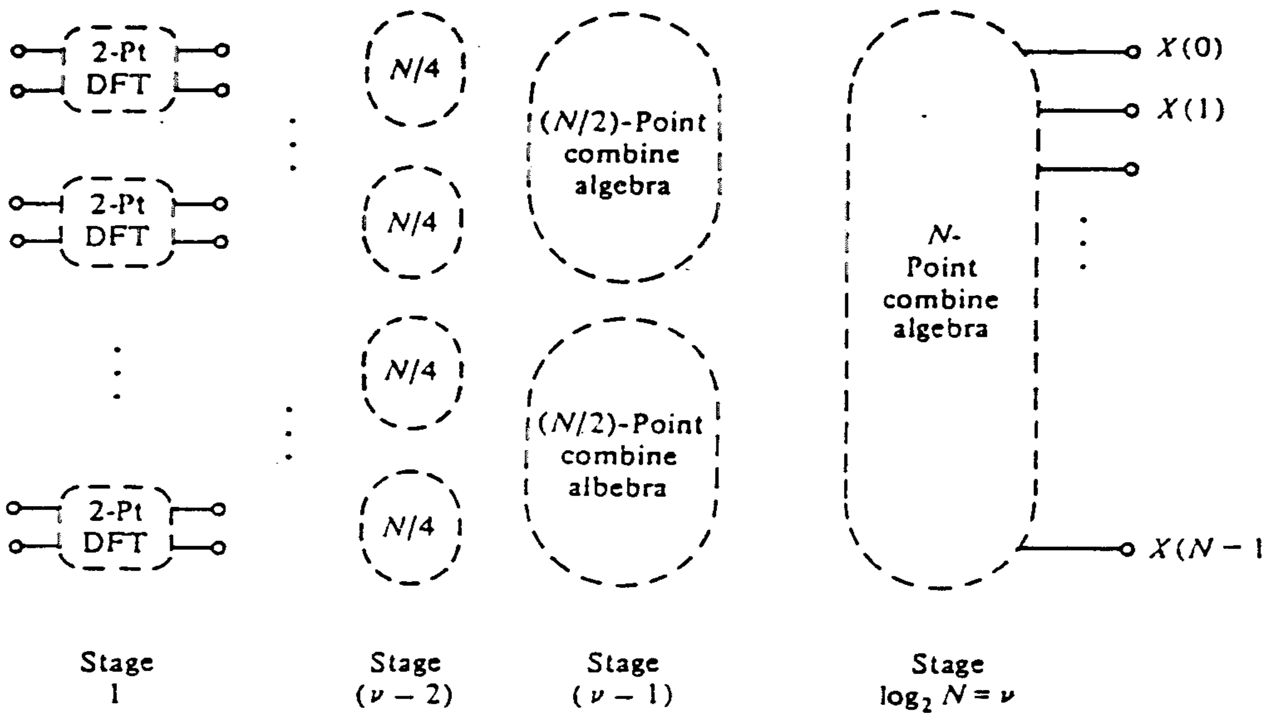


x(0)
x(4)
x(N − 4)
(N/4)-Point DFT
G'1(k)
x(2)
x(6)
x(N − 3)
(N/4)-Point DFT
H'1(k)
x(1)
x(5)
x(N − 2)
(N/4)-Point DFT
G'2(k)
x(3)
x(7)
x(N − 1)
(N/4)-Point DFT
H'2(k)
(N/2)-Point combine algebra
G(k)
(N/2)-Point combine algebra
H(k)
N-Point combine algebra
X(0)
X(1)
X(N/2 − 1)
X(N/2)
X(N/2 + 1)
X(N − 1)
N-Point DFT

Continuing this process, each <u>$N/4$ - point transform</u> is broken into <u>two $N/8$ - point DFTs</u>, etc. Since $N = 2^n$, this process can be continued until there are

$$\nu = \log_2 N$$

stages as shown below;



The decomposition continues until the 2-point DFT stage is reached. At this point we compute the 2-point DFT from the direct formula.

$$X(k) = \sum_{n=0}^{1} x(n) W_2^{kn} \qquad k = 0, 1$$

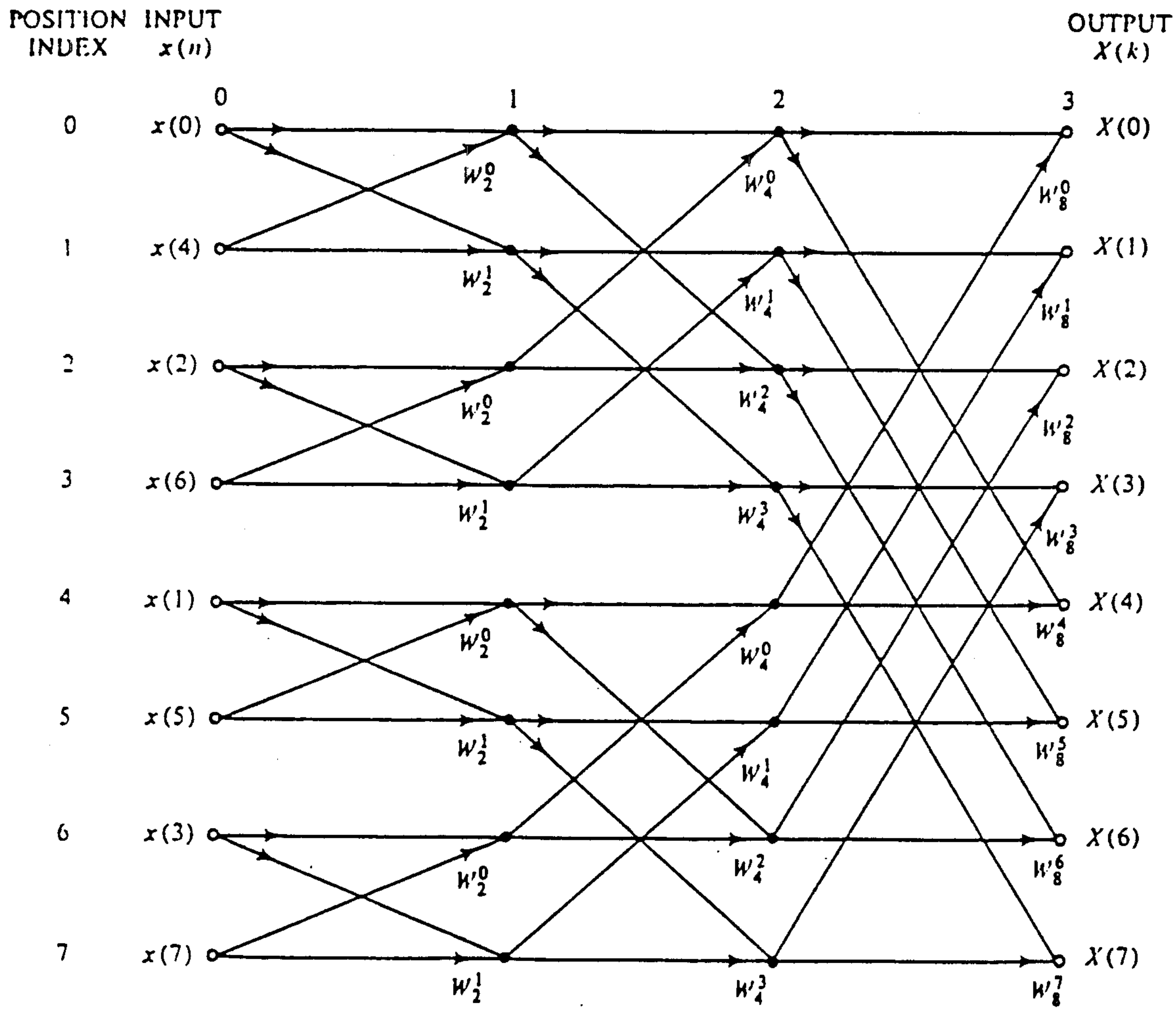
Or,

$$X(0) = x(0) + W_2^0 x(1) = x(0) + x(1)$$
$$X(1) = x(0) + W_2^1 x(1) = x(0) - x(1)$$

In general, the combining algebra at each stage takes N complex multiplies, and there are $\log_2 N$ stages. Therefore the approximate number of complex multiplies for the total decomposition becomes

$$\eta = N \log_2 N$$

Let's look at an example: an 8-point Decimation-in-Time FFT on the next page

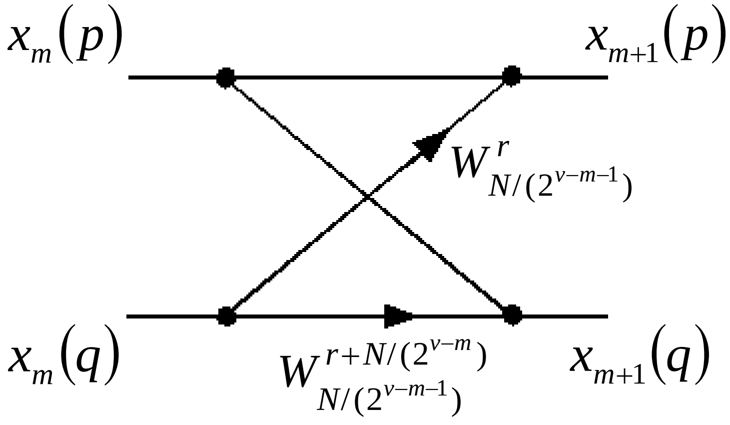

POSITION INDEX
INPUT x(n)
OUTPUT X(k)
0
1
2
3
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
4
5
6
7

From the flow diagram, several important observations can be made:

i) The input data has been shuffled. The input data appears in “bit reversed” order. For example:

Position	Binary	Bit-reversed	Sequence index
6	110	011	3
4	100	001	1
2	010	010	2

ii) The basic computational element is called a “butterfly”. If we use *m* to represent the stage, and *p* and *q* to represent the position numbers in each stage, we get



$$x_{m+1}(p) = x_m(p) + W^{r}_{N/(2^{v-m-1})} x_m(q)$$
$$x_{m+1}(q) = x_m(p) + W^{r+N/(2^{v-m})}_{N/(2^{v-m-1})} x_m(q)$$

$$r \in [0, \cdots, 2^m - 1]$$

r is variable and depends on the position of the butterfly.

Note that $x_{m+1}(p)$ and $x_{m+1}(q)$ can be calculated and placed back in the storage registers of $x_m(p)$ and $x_m(q)$. This kind of computation is referred to as an "in-place" computation.

iii) The frequency domain values, $X(k)$ are in normal order.

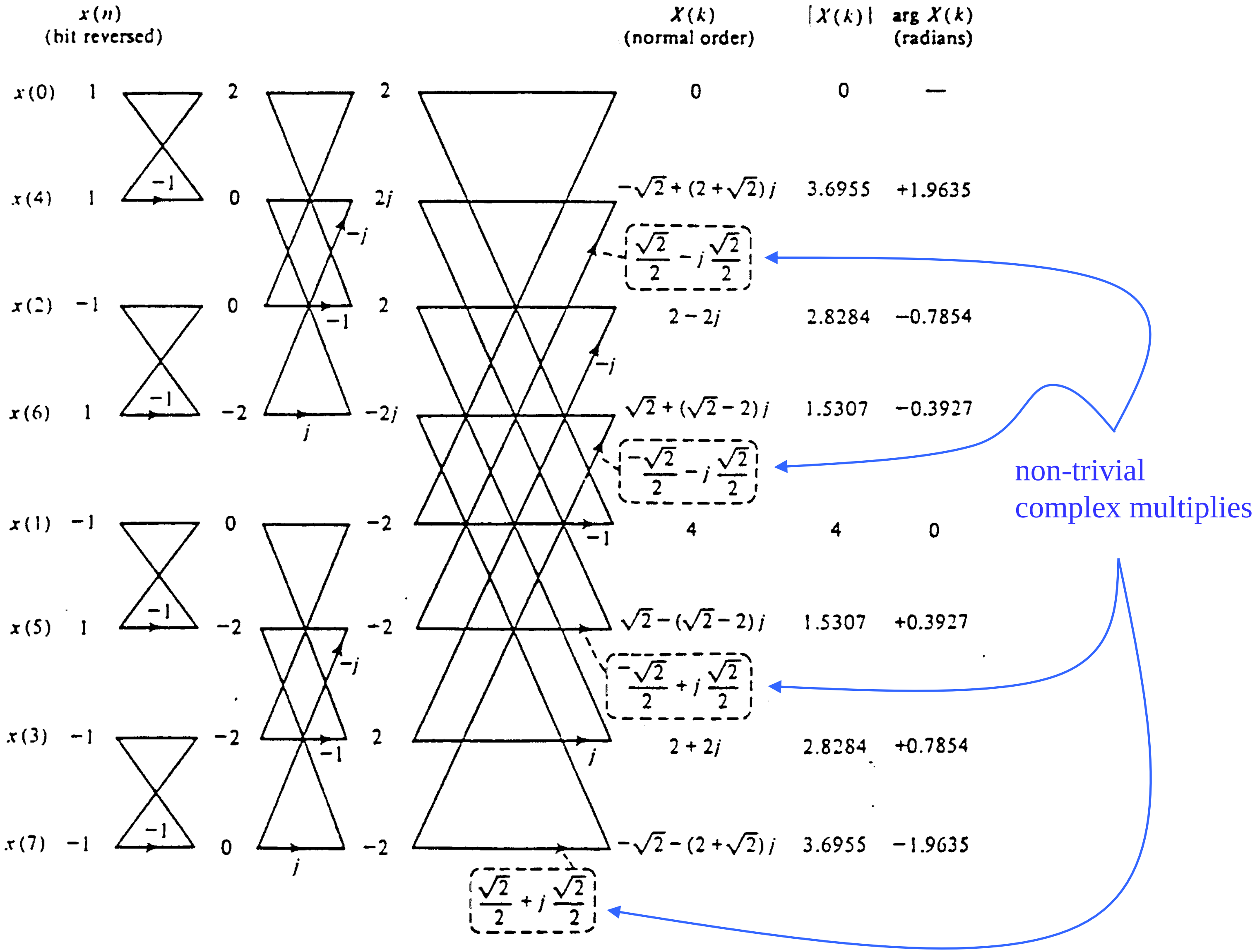
Also note that:

$$W_2^1 = W_4^2 = W_8^4 = -1 \qquad W_4^3 = W_8^6 = j$$

$$W_2^0 = W_4^0 = W_8^0 = 1 \qquad W_4^1 = W_8^2 = -j$$

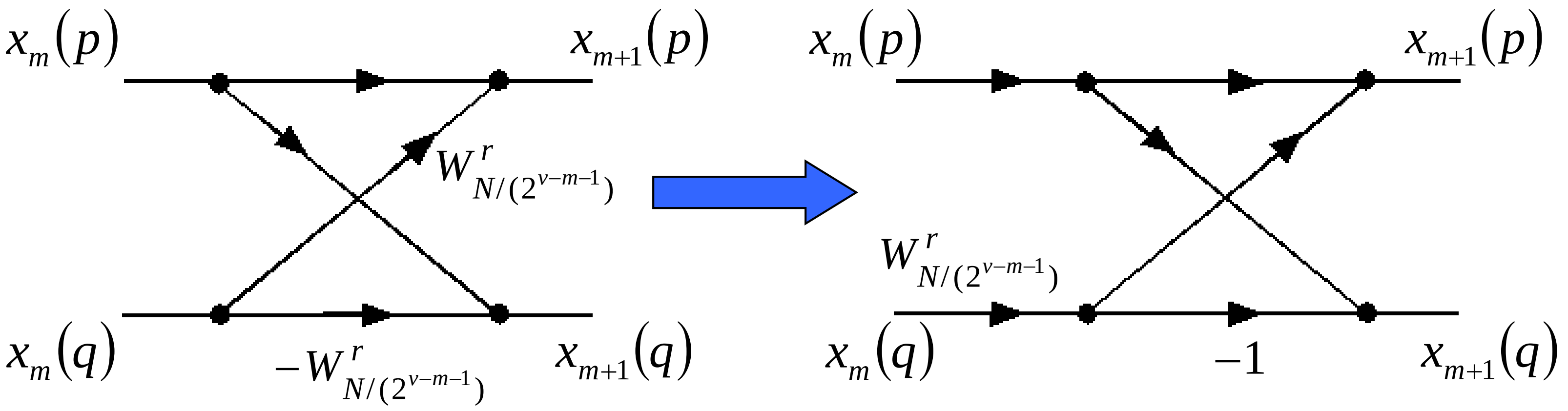
There are only 4 nontrivial complex multiplications in the entire 8-point FFT as shown below for the sequence

$$x(n) = (1, -1, -1, -1, 1, 1, 1, -1)$$



x(n)
(bit reversed)
X(k)
(normal order)
|X(k)|
arg X(k)
(radians)
x(0) 1
x(4) 1
x(2) −1
x(6) 1
x(1) −1
x(5) 1
x(3) −1
x(7) −1
−1
−1
−1
−1
2
0
0
−2
0
−2
−2
0
−j
−1
j
−j
−1
j
2
2j
2
−2j
−2
−2
2
−2
−j
−1
j
$\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$
$-\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$
$-\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
$\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
0 0 —
$-\sqrt{2} + (2+\sqrt{2})j$ 3.6955 +1.9635
$2 - 2j$ 2.8284 −0.7854
$\sqrt{2} + (\sqrt{2} - 2)j$ 1.5307 −0.3927
4 4 0
$\sqrt{2} - (\sqrt{2} - 2)j$ 1.5307 +0.3927
$2 + 2j$ 2.8284 +0.7854
$-\sqrt{2} - (2+\sqrt{2})j$ 3.6955 −1.9635
non-trivial
complex multiplies

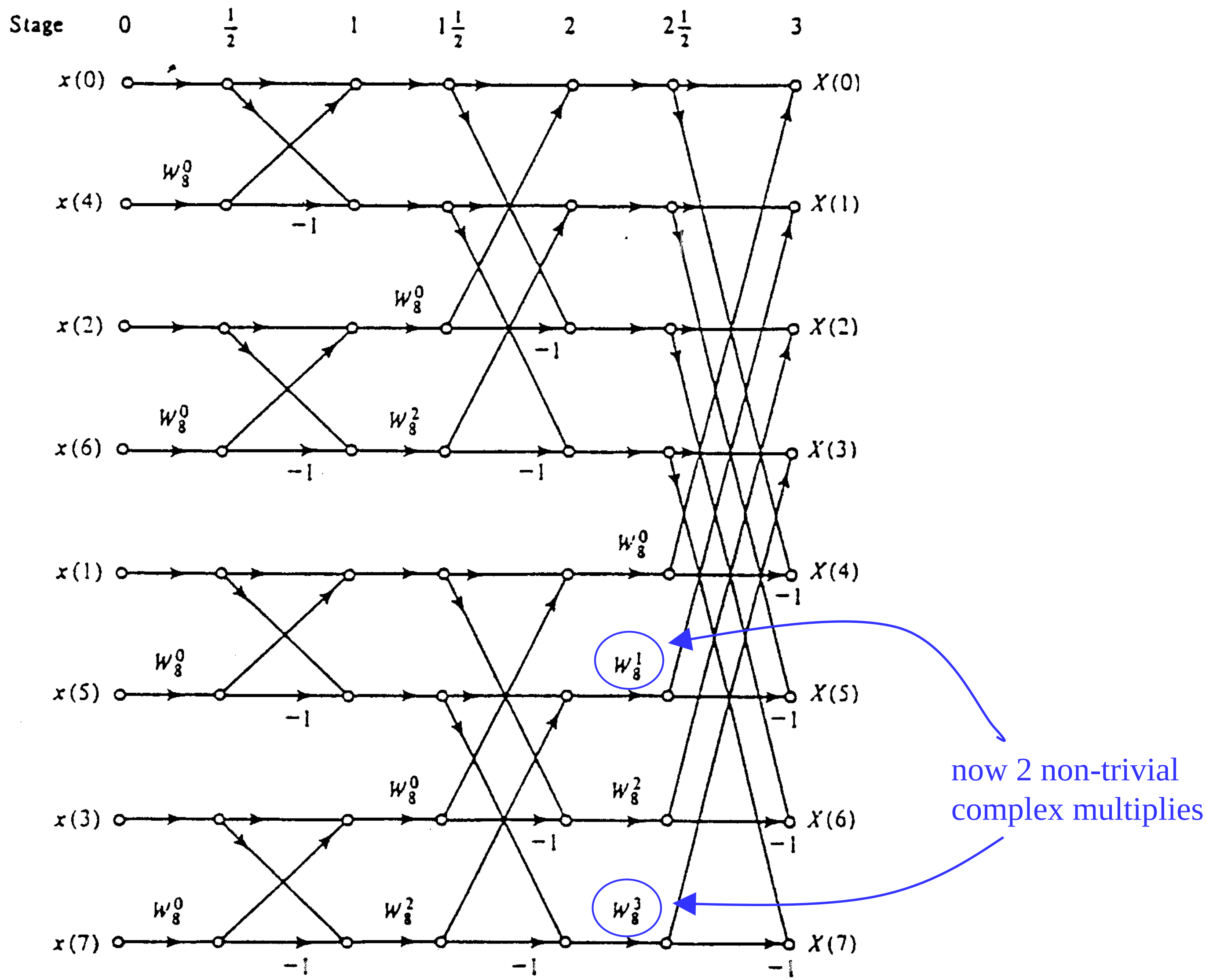
The basic butterfly can be further simplified to reduce the number of complex multiplications by one, as follows:



with this simplification, the number of complex multiplications required for calculating the FFT is

$$\eta = \frac{N}{2}\log_2 N$$

The reduced 8-point decimation in time FFT is shown on the next page


Stage
0
1/2
1
1 1/2
2
2 1/2
3
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
W_8^0
W_8^2
W_8^1
W_8^3
−1
now 2 non-trivial
complex multiplies

11.4 Decimation-in-Frequency FFT

Begin by breaking up $X(k)$ as follows:

1st half

2nd half

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

next let $r = n - \frac{N}{2}$

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

relabel r as n

And since

$$W_N^{kN/2} = e^{-j\frac{2\pi kN}{2N}} = e^{-j\pi k} = (-1)^k$$

then

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

The decimation is obtained by taking the odd and even terms of $X(k)$

even values $\Rightarrow$ $k = 2r$, for $r = 0, 1, \ldots, N/2 - 1$

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

$$= \sum_{n=0}^{N/2-1} \left[x(n) + x(n + N/2)\right] W_{N/2}^{nr}$$

odd values $\Rightarrow$ $k = 2r + 1$, for $r = 0, 1, \ldots, N/2 - 1$

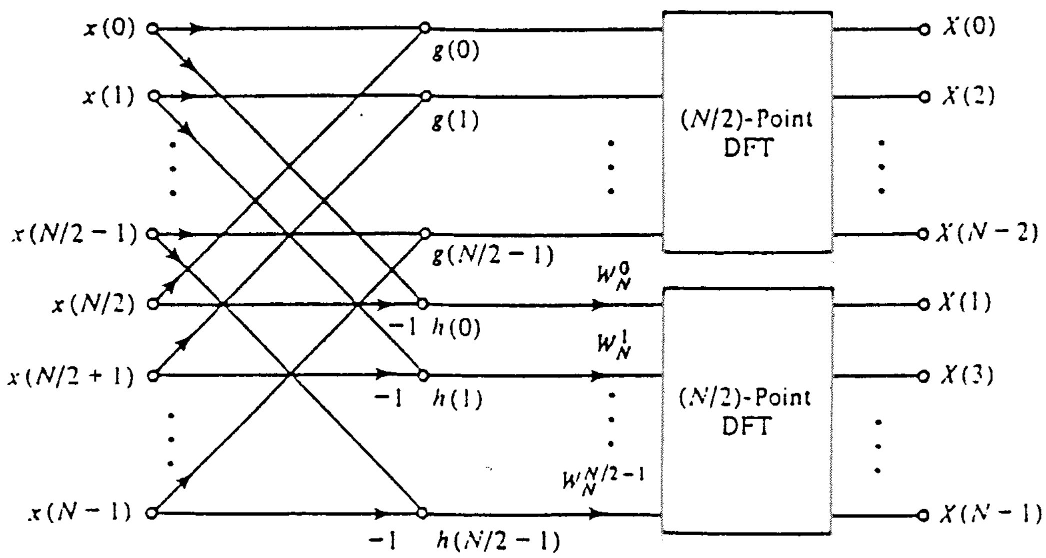
$$X(2r+1) = \sum_{n=0}^{N/2-1} \left[x(n) + (-1)^{2r+1} x(n + N/2)\right] W_N^{n(2r+1)}$$

$$= \sum_{n=0}^{N/2-1} \left[x(n) - x(n + N/2)\right] W_N^{n} W_{N/2}^{nr}$$

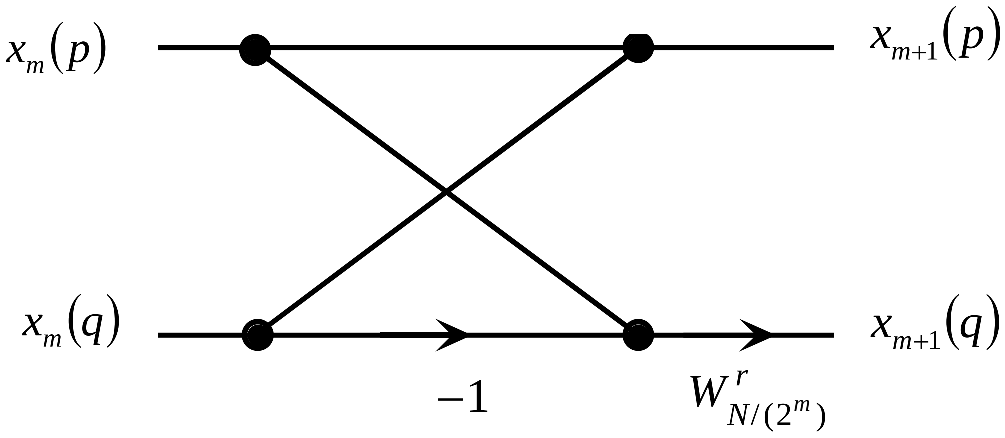
Now define $$g(n) = x(n) + x(n + N/2)$$
$$h(n) = x(n) - x(n + N/2)$$

So that $$X(k) = \begin{cases} \sum_{n=0}^{N/2-1} g(n) W_{N/2}^{nr}, & \text{for } k \text{ even} \\ \sum_{n=0}^{N/2-1} h(n) W_N^{n} W_{N/2}^{nr}, & \text{for } k \text{ odd} \end{cases}$$

The first stage is shown below:



The basic butterfly for the Decimation-in-Frequency FFT is



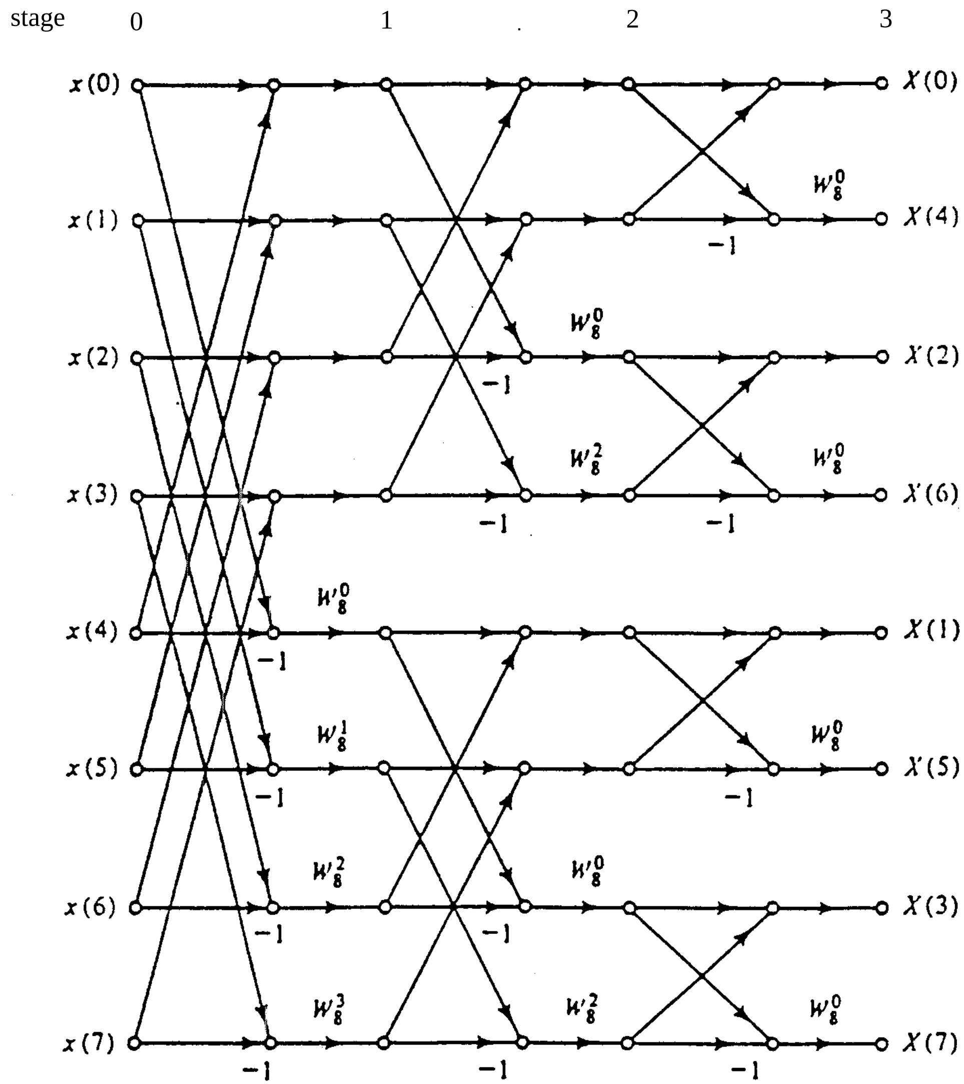
$$r \in [0, \cdots, 2^{\nu-m-1} - 1]$$

In-place computations can be accomplished with this structure in which

$$x_{m+1}(p) = x_m(p) + x_m(q)$$

$$x_{m+1}(q) = [x_m(p) - x_m(q)] W^r_{N/(2^m)}$$

Example: an 8-point Decimation-in-Frequency FFT



11.5 Practical Considerations

Computation of the Inverse DFT. The inverse DFT is given by,

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-kn}, \quad n = 0, 1, \ldots, N\text{-}1$$

*

*

* these are the only differences between the DFT and the IDFT

Therefore, there are two ways to implement the IDFT

(i) Alter the DFT (FFT) algorithm as follows:

- change $W_N^{kn} \rightarrow W_N^{-kn}$
- scale input data by $1/N$
- input $X(k)$ sequence values instead of $x(n)$

(ii) Use the DFT itself to compute the IDFT.

Consider the complex conjugate of the IDFT

$$x^*(n) = \left[\frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \right]^*$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} X^*(k) W_N^{kn} = \frac{1}{N} \mathrm{DFT}\left[X^*(k) \right]$$

Again, conjugate both sides to obtain

$$x(n) = \frac{1}{N} \left\{ \mathrm{DFT}\left[X^*(k) \right] \right\}^*$$

So really just need the one algorithm and can do both DFT/FFT and IDFT/IFFT with it

Efficient use of DFT for Two Real Sequences

Let $x_1(n)$ and $x_2(n)$ be real sequences. Form a complex valued sequence:

$$x(n) = x_1(n) + j\, x_2(n)$$

and from properties of symmetry we can write

$$x_1(n) = \frac{x(n) + x^*(n)}{2} \qquad x_2(n) = \frac{x(n) - x^*(n)}{2j}$$

Taking the DFT of both:

$$X_1(k) = \frac{1}{2}\left[X(k) + X^*(N-k)\right]$$
$$X_2(k) = \frac{1}{j2}\left[X(k) - X^*(N-k)\right]$$

Compute the DFT of complex $x(n)$ and then post-process to efficiently compute the separate DFTs of both $x_1(n)$ and $x_2(n)$