Fast Fourier Transform

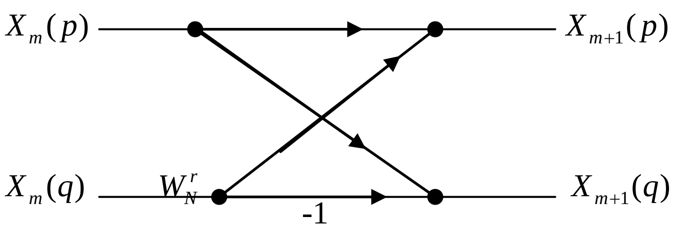
The decomposition procedure is called the fast Fourier transform *(FFT)* algorithm. The reduction in proportionality from N^2 to $N \log_2 N$ multiply/add operations represents a significant saving in computation effort as shown in the following table.

N	***N^2 (DFT)***	***$N \log_2 N$ (FFT)***
2	4	2
4	16	8
8	64	24
16	256	64
32	1024	160
64	4096	384
128	16384	896
256	65536	2048

In the FFT algorithm the number of samples is equal to 2^n where n is a positive integer and N is assumed to be of the form $N = 2^n$

2^n = N = *number of inputs*
2^{n-1} = *butterfly / level*
n = *number of levels*

<u>Butterfly Computation</u>



$$X_{m+1}(p) = X_m(p) + X_m(q) * W_N^r$$
$$X_{m+1}(q) = X_m(p) - X_m(q) * W_N^r$$

where:

m: represent stage (level) number
p,q: two different pixel in the same stage
W_N^r N: represent number of pixel(samples)
r: is computed as
for a=1 to $2^{level-1}$ *do*
r= (a-1) * $2^{n-level}$

$$W_N^r = e^{-j2\pi/N}$$

Bit Reverse:

x_0 = 000	$\longrightarrow$	000 = x_0	even		
x_1 = 001	$\longrightarrow$	100 = x_4			
x_2 = 010	$\longrightarrow$	010 = x_2			
x_3 = 011	$\longrightarrow$	110 = x_6			
x_4 = 100	$\longrightarrow$	001 = x_1	odd		
x_5 = 101	$\longrightarrow$	101 = x_5			
x_6 = 110	$\longrightarrow$	011 = x_3			
x_7 = 111	$\longrightarrow$	111 = x_7			

Example: ***Use FFT to find the Fourier transform of the following points***

8 3 1 4 ?

Solution: *number of points = N = 4 =* $2^2 = 2^n$

Butterfly / level = $2^{n-1} = 2^{2-1} = 2$

Number of level = n = 2

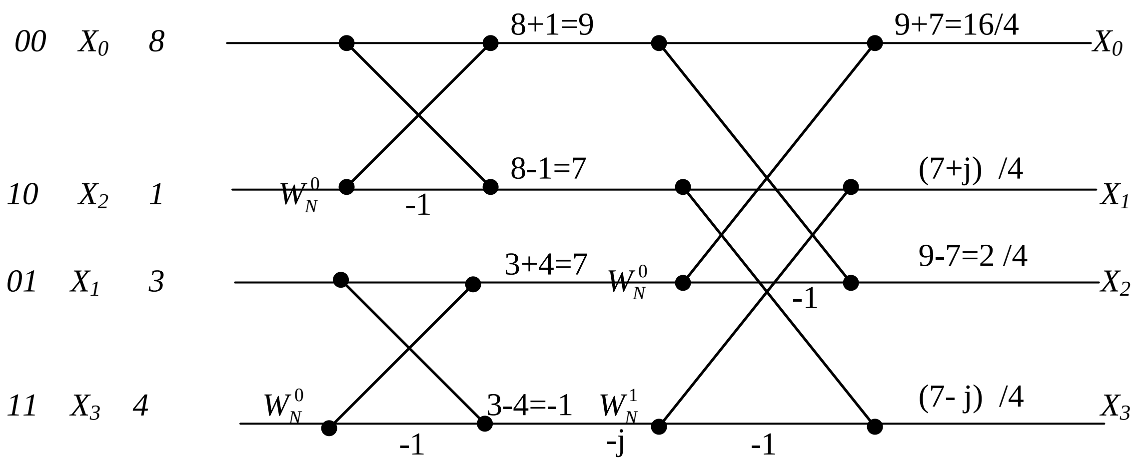
Bit Reverse

$X_0 = 00$		$X_0 = 00$	$= X_0$	**8**
$X_1 = 01$	Bit $\Rightarrow$ Reverse	$X_1 = 10$	$= X_2$	1
$X_2 = 10$		$X_2 = 01$	$= X_1$	**3**
$X_3 = 11$		$X_3 = 11$	$= X_3$	**4**

Butterfly Computation

$$X_{m+1}(p) = X_m(p) + X_m(q) * W_N^r$$

$$X_{m+1}(q) = X_m(p) - X_m(q) * W_N^r$$



$$W_N^0 = W_4^0 = 1$$

$$W_N^1 = W_4^1 = e^{-j2\pi/4} = e^{-j\pi/2} = \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = 0 - j = -j$$

Inverse of Fast Fourier Transform

FFT Inverse

The inverse algorithm is same as *FFT* algorithm but the difference is taking the conjugate of the complex number.

Bit Reverse

$X_0 = 00$		$X_0 = 00$	$= X_0$	*4*
$X_1 = 01$	Bit Reverse $\Longrightarrow$	$X_1 = 10$	$= X_2$	0.5
$X_2 = 10$		$X_2 = 01$	$= X_1$	*1.75+0.25j*
$X_3 = 11$		$X_3 = 11$	$= X_3$	*1.75-0.25j*

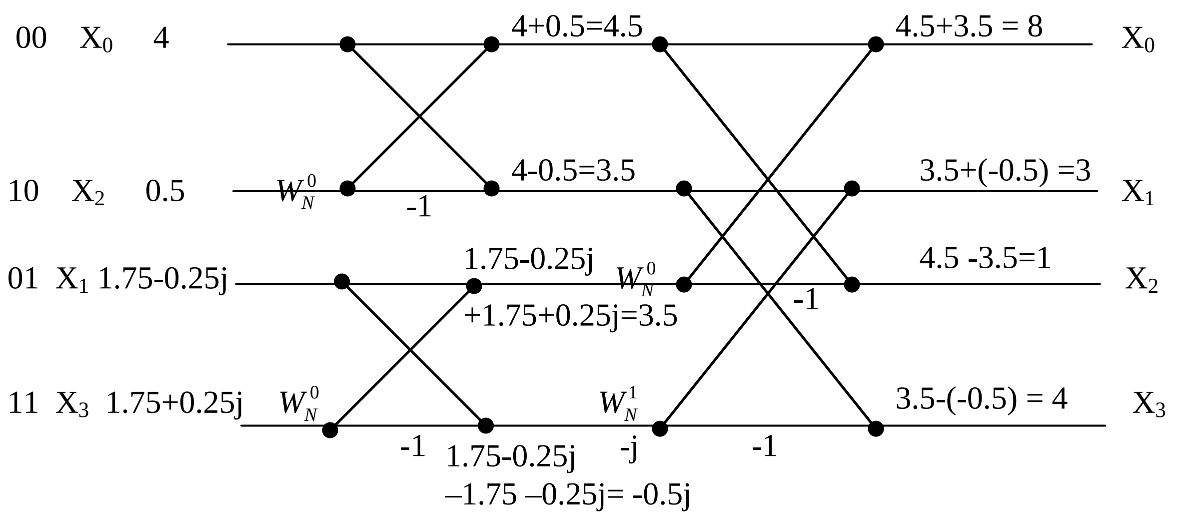
Butterfly Computation

$$X_{m+1}(p) = X_m(p) + X_m(q) * W_N^r$$

$$X_{m+1}(q) = X_m(p) - X_m(q) * W_N^r$$

Conjugate

4	***1.75+0.25j***	***0.5***	***1.75-0.25j***
4	***1.75-0.25 j***	***0.5***	***1.75+0.25j***



$$W_N^0 = W_4^0 = 1$$

$$W_N^1 = W_4^1 = e^{-j2\pi/4} = e^{-j\pi/2} = \cos\frac{\pi}{2} - j\sin\frac{\pi}{2} = 0 - j = -j$$