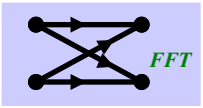
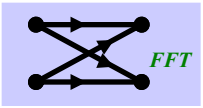


Transform Domain Representation of Discrete Time Signals

Fast Fourier Transform (FFT) Algorithm

Yogananda Isukapalli



FFT - Decimation in Time Algorithm

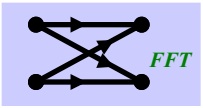
In an attempt to reduce the number of calculations needed for the DFT, let us try breaking up the sequence x[n] into two smaller subsequences. In particular, let us divide x[n] into its odd and even indices

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

$$X(k) = \sum_{n=even} x[n] W_N^{kn} + \sum_{n=odd} x[n] W_N^{kn} \qquad (1)$$

Let us perform a change of variables, let
n = 2m for n even and n = 2m+1 for n odd
for m = 0, 1, ……(N/2 - 1) (2)

Both sequences are half of the length of the original sequence. The first change of variables will give us the indices n = 0, 2, 4 and the other change will give us the odd indices n = 1, 3, 5….



Equation (1) now becomes :

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_N^{k(2m)} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] W_N^{k(2m+1)} \quad (3)$$

Let us look at each sequence in equation (3) independently

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_N^{k(2m)} = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_{N/2}^{mk}$$

$$= \sum_{n=0}^{N_G-1} g[n] W_{N_G}^{nk} \quad (4)$$

Which is the length $N_G = N/2$ DFT of the sequence g[n], which is nothing but x[2m]

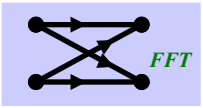
We can do the same to the other sequence after first factoring out the additional W_N^k :

$$\sum_{m=0}^{\frac{N}{2}-1} x[2m+1]W_N^{k(2m+1)} = \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]W_N^{k\,2m}\,W_N^k$$

$$= W_N^k\left[\sum_{m=0}^{\frac{N}{2}-1} x[2m+1]W_{N/2}^{km}\right]$$

$$= W_N^k\left[\sum_{n=0}^{N_H-1} h[n]W_{N_H}^{kn}\right] \quad (5)$$

Which is the length $N_H = N/2$ DFT of the sequence h[n], which is nothing but x[2m+1]



Combining equations (4) and (5) back into (3), we get :

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_{N/2}^{mk} + W_N^k \left[\sum_{m=0}^{\frac{N}{2}-1} x[2m+1] W_{N/2}^{km} \right] \quad (6a)$$

$$X(k) = \sum_{n=0}^{N_G-1} g[n] W_{N_G}^{nk} + W_N^k \left[\sum_{n=0}^{N_H-1} h[n] W_{N_H}^{kn} \right] \quad (6b)$$

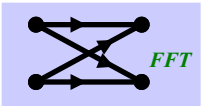
$$X(k) = G(k) + W_N^k H(k) \quad (6c)$$

$$DFT[N.Samples] = DFT\{even.samples\} + W_N^k [DFT\{odd.samples\}]$$

Thus, we can compute the length N DFT by computing two N/2 length DFT's. But is there a saving in doing so ?

In other words :

N - point Transform = N/2 point transform + W_N^k [N/2 point transform]



Total MADS :

G(k) requires $(N/2)^2$ MADS

H(k) requires $(N/2)^2$ MADS

W_N^k H(k) requires N MADS

Computing X(k) :

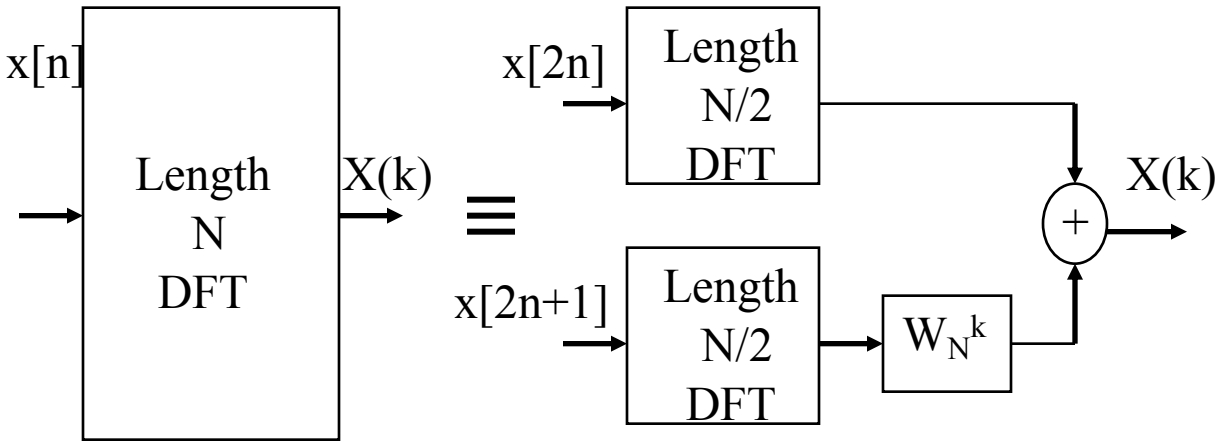
Directly => N^2

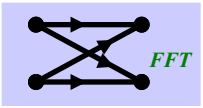
By Decomposition =>$2(N/2)^2 + N$

Now if we do the above exercise for N = 256

Direct => 65,536 MADS
Decomposition => 33,024 MADS

As a result, we have seen that the breaking up a large DFT into two smaller DFT's will allow savings in computations. This process is demonstrated by the block diagram below :





Lets look at equation 6c more closely, since the sequences G(k) and H(k) are also DFT's they have properties described earlier. However since they are of length N/2, we can say that :

$$G(k+\frac{N}{2}) = G(k) \text{ and } H(k+\frac{N}{2}) = H(k) \quad (7)$$

If we compute (6c) explicitly for N=8, like we did in the direct computation of the DFT, and simplify using (7) we get:

$$X(0) = G(0) + W_8^0 H(0) \quad (8\text{-}1)$$

$$X(1) = G(1) + W_8^1 H(1) \quad (8\text{-}2)$$

$$X(2) = G(2) + W_8^2 H(2) \quad (8\text{-}3)$$

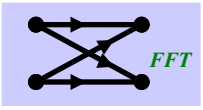
$$X(3) = G(3) + W_8^3 H(3) \quad (8\text{-}4)$$

$$X(4) = G(4) + W_8^4 H(4) = G(0) + W_8^4 H(0) \quad (8\text{-}5)$$

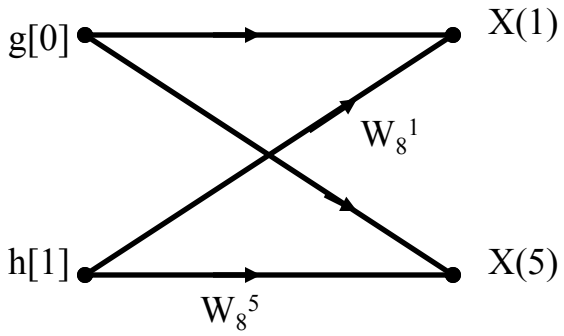
$$X(5) = G(5) + W_8^5 H(5) = G(1) + W_8^5 H(1) \quad (8\text{-}6)$$

$$X(6) = G(6) + W_8^6 H(6) = G(2) + W_8^6 H(2) \quad (8\text{-}7)$$

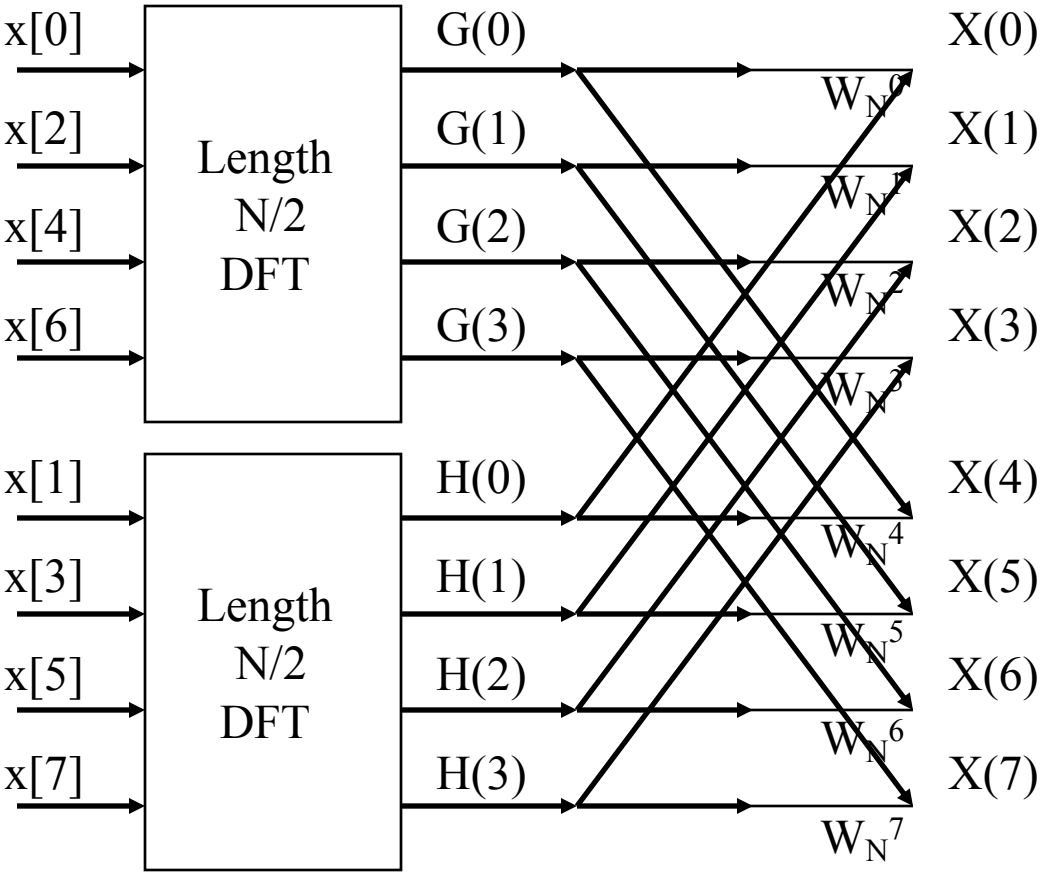
$$X(7) = G(7) + W_8^7 H(7) = G(3) + W_8^7 H(3) \quad (8\text{-}8)$$



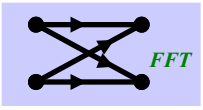
For Example :



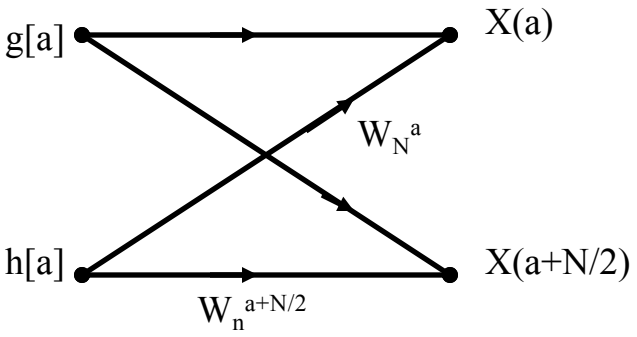
Putting everything together leads to the following flow diagram for N = 8:



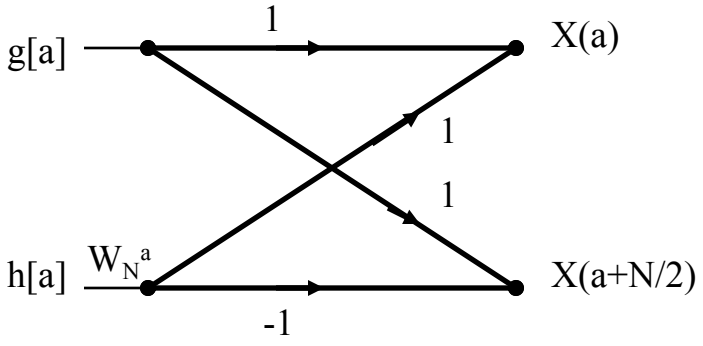
The value to be multiplied on each branch is the one closest to the arrow head. No value indicates that G(k) is taken as is (multiplied by 1)



The general form of each structure can be expressed as :



By using the properties of the basis functions W_N^{nk}, we can simplify this by factoring out W_n^a. This leads to the following structure :



This is known as the 2-point butterfly, since the lattice structure resembles a butterfly on its side. Butterflies are the elementary computations, or building blocks of many FFT structures.

Thus the earlier shown block diagram for N=8 can be now simplified to :

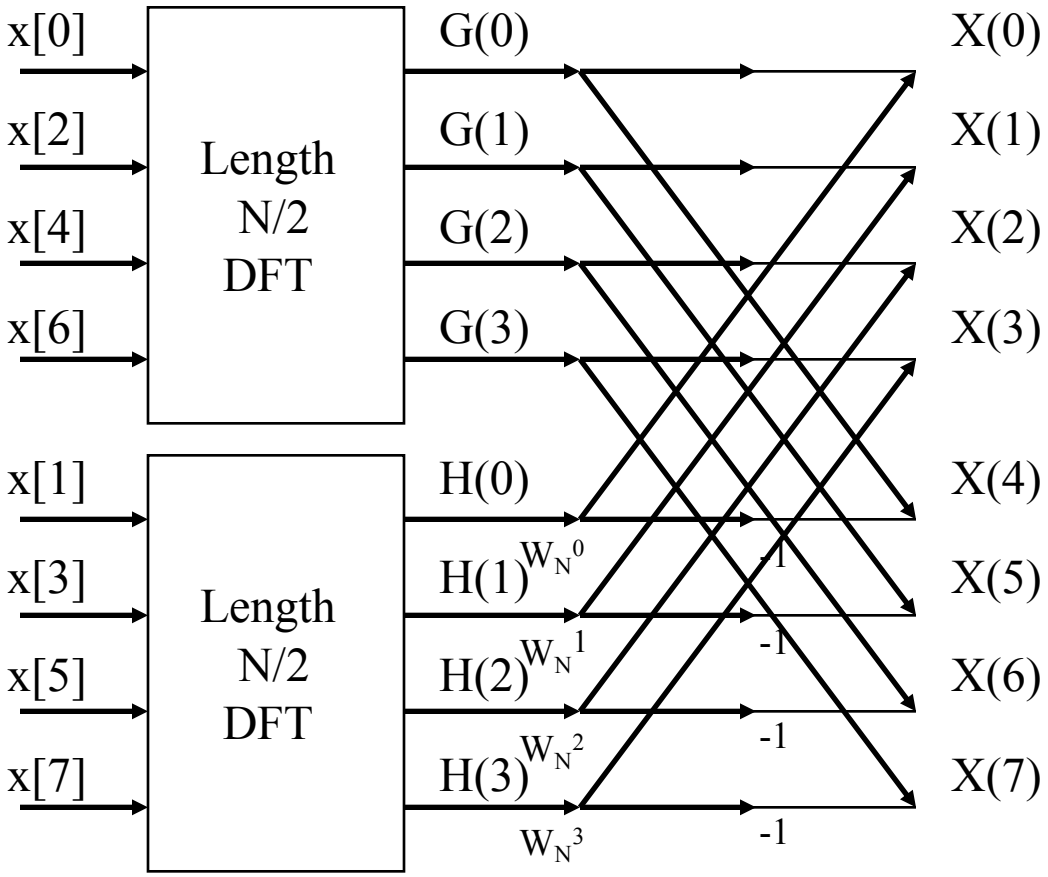
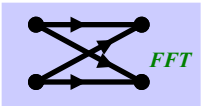


Figure 2

The terms W_N^a for a = 0, 1, …..N/2 -1 are known as twiddle factors.



Continue the Decomposition in Time

We can also break G(k) and H(k) into smaller sequences for even more savings. By further dividing (6) into its new even and odd sequences, we get equations similar to the ones we got when we initially divided the sequences.

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m] W_{N/2}^{mk} + W_N^k \left[\sum_{m=0}^{\frac{N}{2}-1} x[2m+1] W_{N/2}^{km} \right]$$

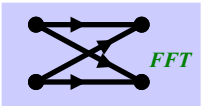
For each sequence, split into :

m = 2p and m = 2p + 1 for p = 0, 1, ..,(N/4 - 1)

Which yields:

$$X(k) = \sum_{p=0}^{\frac{N}{4}-1} g[2p] W_{N/4}^{pk} + W_{N/2}^k \left[\sum_{p=0}^{\frac{N}{4}-1} g[2p+1] W_{N/4}^{kp} \right] +$$

$$W_N^k \left\{ \sum_{p=0}^{\frac{N}{4}-1} h[2p] W_{N/4}^{kp} + W_{N/2}^k \left[\sum_{p=0}^{\frac{N}{4}-1} h[2p+1] W_{N/4}^{kp} \right] \right\}$$



$$X(k) = \sum_{p=0}^{\frac{N}{4}-1} x[4p] W_{N/4}^{pk} + W_{N/2}^{k} \left[\sum_{p=0}^{\frac{N}{4}-1} x[4p+2] W_{N/4}^{kp} \right] +$$

$$W_N^k \left\{ \sum_{p=0}^{\frac{N}{4}-1} x[4p+1] W_{N/4}^{kp} + W_{N/2}^{k} \left[\sum_{p=0}^{\frac{N}{4}-1} x[4p+3] W_{N/4}^{kp} \right] \right\} \quad (9)$$

Looking at equation (9), we see that x[n] has now been divided into four sequences. Substituting for p yields the indices n = 0, 4, 8… for the first sequence, n = 2, 6, 10, .. For the second, n = 1, 5, 9… for the third, and n = 3, 7, .. For the last, thus including all the indices of the sequence x[n].

Block Diagram for the decomposition :

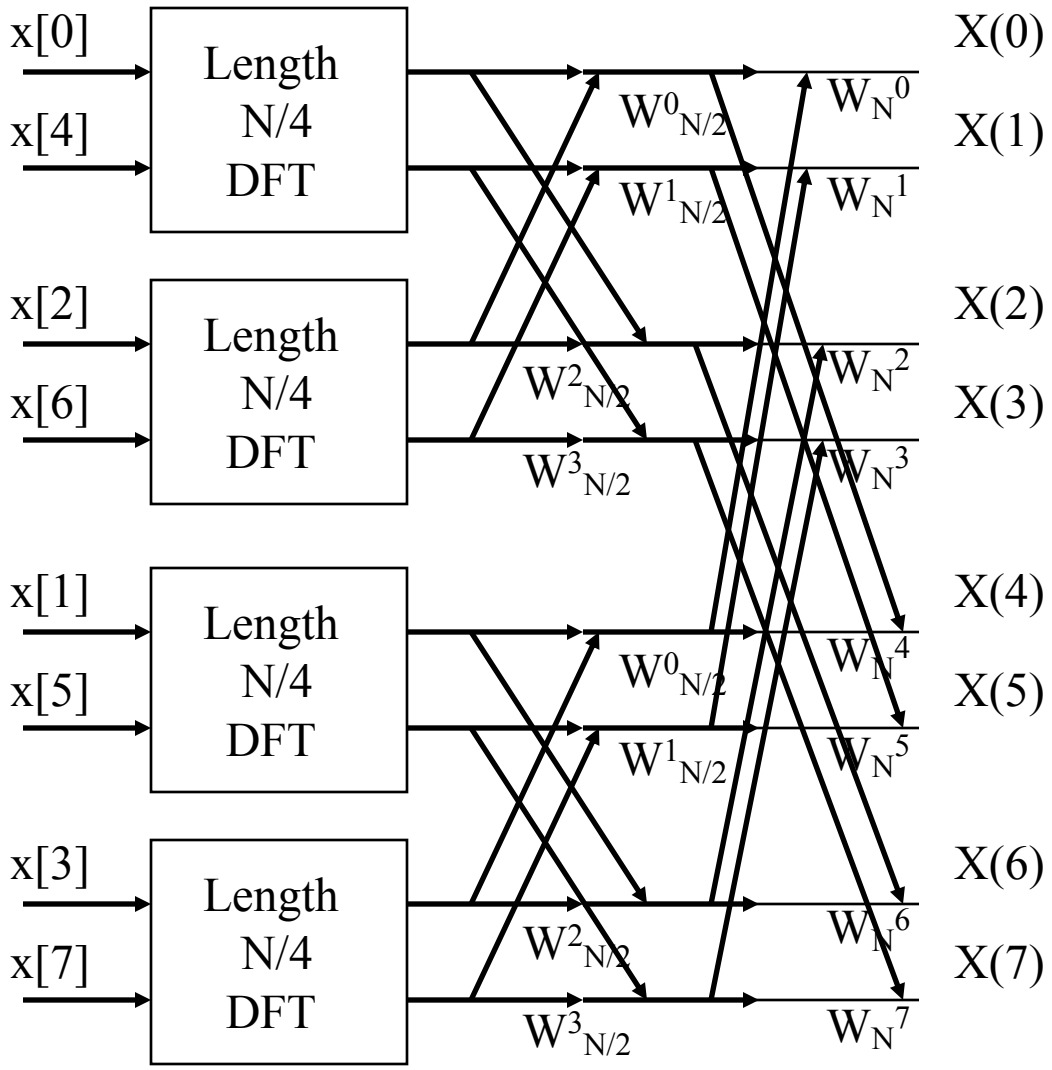
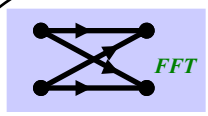


Figure 3



We can also modify this block diagram (3) as we did earlier : in Figure 2

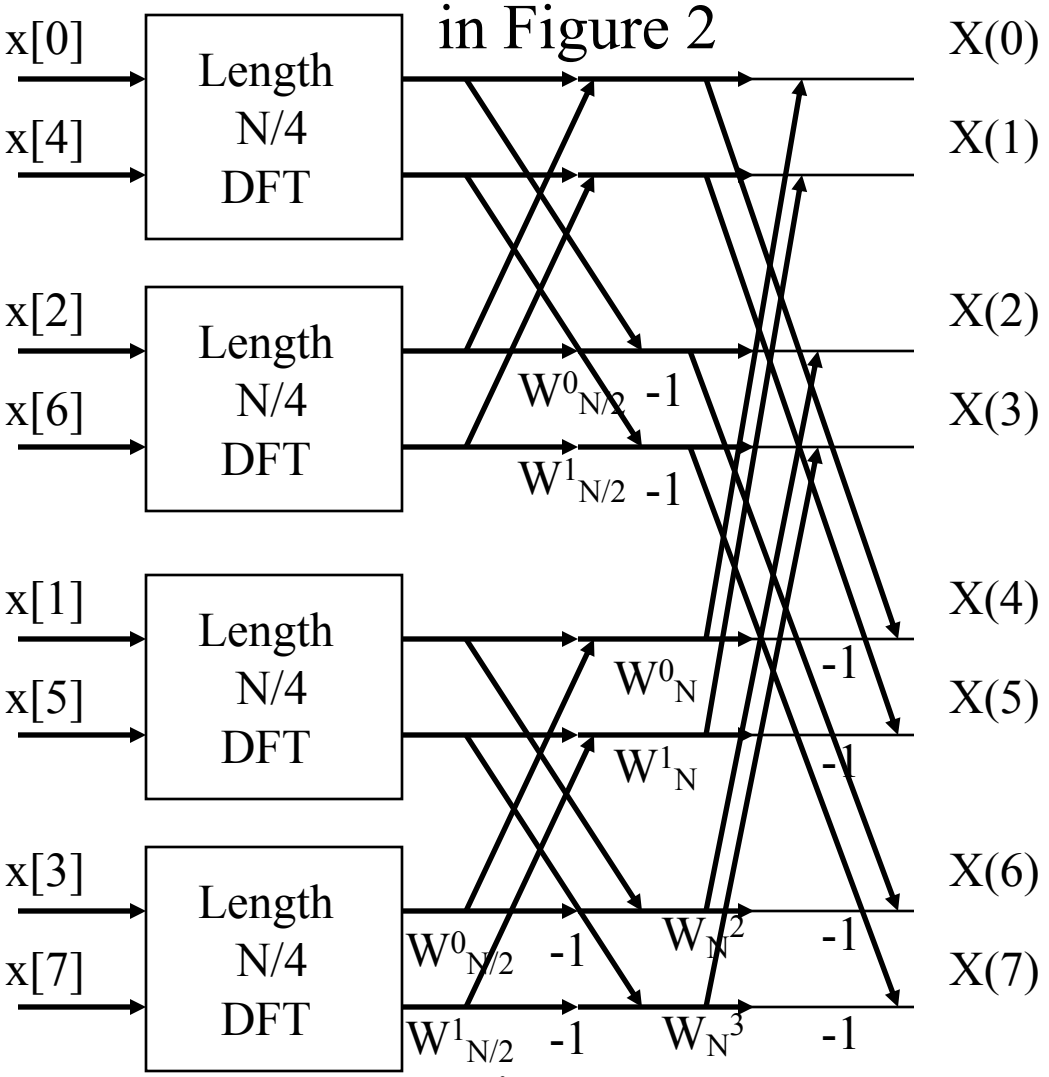


Figure 4

Note that the twiddle factors in the second stage (length N/2 DFT) can be changed as in property of the basis functions W_n^a: $W_{N/2}^0 = W_N^0 .. \& .. W_{N/2}^1 = W_N^2$

The Decomposition continues until we are left with N/2 2 point butterflies, resulting in $\log_2 N$ stages.

The final Radix-2 decomposition in Time structure

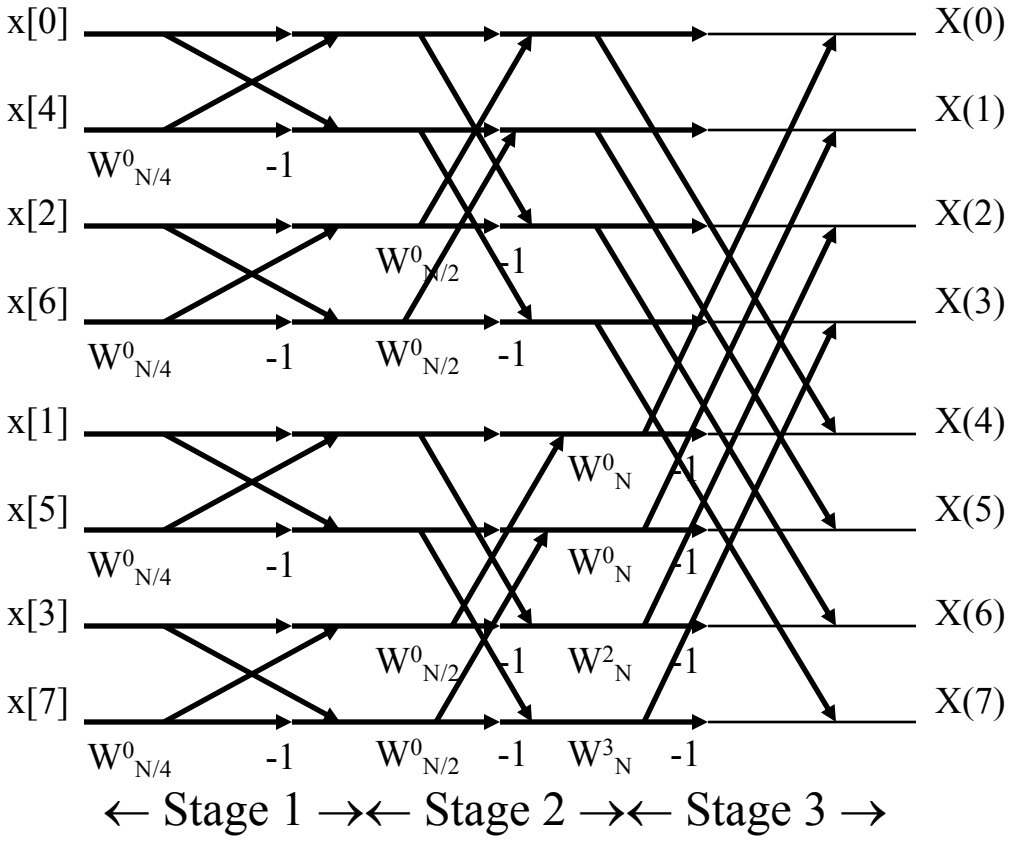
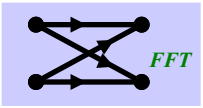


Figure 5

Figure 5 shows the lattice structure for computing a length N =8 DFT. Note that incorporated in this structure is the lattice for also computing the Length N = 2 & 4 DFTs. To compute N=16, we can combine two length 8 DFT's and add twiddle factors of the form W_{16}^{a}. Thus this could be extended to generate higher order DFT's of length 2^{v}, where v is an integer.

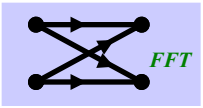


The structure shown in figure 5 is called the a radix-2 Fast Fourier Transform (FFT) because we are decomposing our signal into groups of two, which leads to two point butterflies.

Advantages of the Lattice Structure:

1. **In place computations**: Memory in hardware is an important consideration when computing DFT of a signal. The lattice implementation requires the least amount of memory to compute the DFT. Since each butterfly returns the same number of outputs as the inputs, we can load the input values into registers, perform the necessary calculations and return the output values to the same registers. Hence we can view each horizontal branch in the flow diagram as a memory location. This type of computation is commonly referred as in-place computation.

2) **Higher order Structures**: In Signal processing, lattice structure are popular because in providing an optimal M^{th} order solution to a problem, it incorporates the optimal m=0, 1, .. M-1 order solutions. Hence it is easy to compute the optimal $(M+1)^{th}$ order since only one additional stage need to be computed. The same is true for our FFT flow diagram too. All lower FFT's are incorporated into the N^{th} order FFT. In addition, to compute the length 2N FFT we simply combine two length N FFTs by adding on one stage.

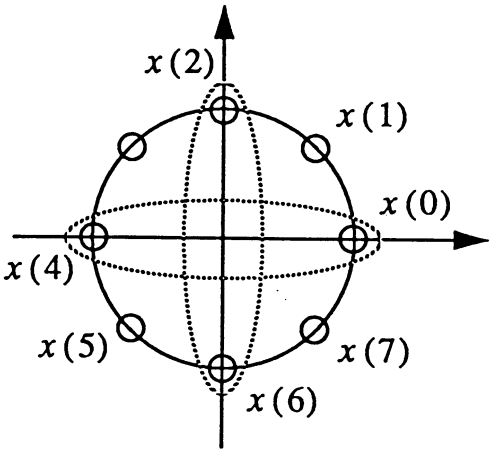


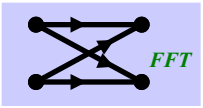
Decimation In Time (DIT) FFT Structure

1. **Re-ordering the input data**: By looking at the FFT in figure 5 we see that we must rearrange the order of the input samples to get the DFT samples I a sequential order, as well as perform the computations in place. A simple way to find how the inputs are arranged is to realize that the inputs in are bit reversed order. If we express 2^v output indices in terms of v bits of the form $X(n_0n_1\ldots n_v)$, then the input along the same horizontal line (same place in memory) is of the form $x(n_0n_1\ldots n_v)$. Another easy way to determine these pairings is to recognize the initial pairs are exactly 180^o apart on the unit circle

$$X(n_0n_1n_2) \rightarrow x(n_2n_1n_0)$$

$$\begin{aligned}
x(0) &\rightarrow X(0) & X(000) &\rightarrow x(000)\\
x(4) &\rightarrow X(1) & X(001) &\rightarrow x(100)\\
x(2) &\rightarrow X(2) & X(010) &\rightarrow x(010)\\
x(6) &\rightarrow X(3) & X(011) &\rightarrow x(110)\\
x(1) &\rightarrow X(4) & X(100) &\rightarrow x(001)\\
x(5) &\rightarrow X(5) & X(101) &\rightarrow x(101)\\
x(3) &\rightarrow X(6) & X(110) &\rightarrow x(011)\\
x(7) &\rightarrow X(7) & X(111) &\rightarrow x(111)
\end{aligned}$$





2. **Twiddle factor patterns**: It is also easy to see the patterns of the twiddle factors in figure 5. For each stage, we will have a total of N/2 twiddle factors. They will appear on the lines leading to the bottom points of each butterfly in groups of 2^{v-1}. There will be $N/2^v$ of these groups. As was stated earlier, they will vary from a = 0 t a = 2^{v-1}-1. For example, in stage 2 of the length 8 FFT, the twiddle factors are in $N/2^2$=2 groups of 2^{2-1}=2 and vary from a = 0 to a=2^1-1=1.

Number of Computations :

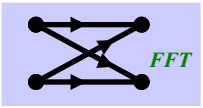
For each stage: Multiplications:

N/2 Twiddle factor multiplications +N/2 butterfly multiplications(by -1) = N

Additions : N butterfly additions

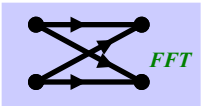
Total N MADS per stage. We have a total of $\log_2 N$ stages, giving us a total of :

$N\log_2 N$ MADS.



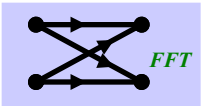
Example: N = 8
Total MADS = $8\log_2 8 = 24$ MADS for DIT
By direct method it would be 64 MADS
the DIT method gives us a saving of 62.5%

Computational savings increases as N increases:
for N = 1024: $N^2 = 1{,}048{,}576$ MADS
$N\log_2 N = 10{,}240$ MADS
Saving of 99.02%
This because $N\log_2 N$ increases linearly whereas N^2 non-linearly.



	DFT		FFT			
N	Number of complex multiplications	Number of complex additions	Number of complex multiplications	Number of complex additions	Ratio of DFT multiplications to FFT multiplications	Ratio of DFT additions to FFT additions
2	4	2	1	2	4	1
4	16	12	4	8	4	1.5
8	64	56	12	24	5.3	2.3
16	256	240	32	64	8.0	3.75
32	1 024	992	80	160	12.8	6.2
64	4 096	4 032	192	384	21.3	10.5
128	16 384	16 256	448	896	36.6	18.1
256	65 536	65 280	1 024	2 048	64.0	31.9
512	262 144	261 632	2 304	4 608	113.8	56.8
1024	1 048 576	1 047 552	5 120	10 240	204.8	102.3
2048	4 194 304	4 192 256	11 264	22 528	372.4	186.1
4096	16 777 216	16 773 120	24 576	49 152	682.7	341.3
8192	67 108 864	67 100 672	53 248	106 496	1 260.3	630.0

Figure 6: Savings in complex multiplications and additions when the FFT is used instead of the DFT



Inverse Fast Fourier Transform (IFFT)

• Let us consider the eqns for DFT and IDFT again

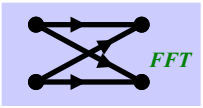
$$X(k) = F_D[x(n)] = \sum_{n=0}^{N-1} x[n] e^{-j(\frac{2\pi}{N})nk} \qquad k=0,.....N-1 \qquad \text{DFT}$$

$$\Updownarrow$$

$$x[n] = F_D^{-1}[X(k)] = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{j(\frac{2\pi}{N})nk} \qquad n=0,.....N-1 \qquad \text{IDFT}$$

We observe that the structure of x[n] differs from X(k) only in the factor 1/N and the sign of the exponent.

Thus with small modifications, we can use the same FFT algorithm to compute the IFFT also.



FFT - Decimation in Frequency (DIF) Algorithm

• Alternate approach involving separating the initial transform into two transforms, one containing the first half of the data and the other containing the second half of the data.

• Order of the input data is unaltered but the output FFT sequence is bit reversed.

• Overall, there is little to choose between DIF and DIT algorithms