Chapter 15

FFT Algorithms

FFT Algorithms

- Highly efficient algorithms for computing the DFT were first developed in the 1960s.

- All Fast Fourier Transforms (FFTs) rely upon the fact that the standard DFT involves redundant calculation.

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp(-j2\pi kn / N)$$

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

.

where

$$W_N = \exp(-j2\pi / N)$$

and X[k] are evaluated for

$$0 \le k \le (N-1)$$

Periodic Nature of the W_N^{nk} function

- Before getting down to detail, you may find it helpful if we demonstrate the periodic nature of W_N^{nk} For simplicity let us consider a short DFT - say N=8.

Number of terms

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

In this case, the complete computation involves varying both k and n over the range 0 to 7 inclusive, giving a product (kn) which varies between 0 and 49.

Values obtained

Value of k \ Value of n	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
1	1	$\frac{(1-j)}{\sqrt{2}}$	$-j$	$\frac{-(1+j)}{\sqrt{2}}$	-1	$\frac{-(1-j)}{\sqrt{2}}$	j	$\frac{(1+j)}{\sqrt{2}}$
2	1	$-j$	-1	j	1	$-j$	-1	j
3	1	$\frac{-(1+j)}{\sqrt{2}}$	j	$\frac{(1-j)}{\sqrt{2}}$	-1	$\frac{(1+j)}{\sqrt{2}}$	$-j$	$\frac{-(1-j)}{\sqrt{2}}$
4	1	-1	1	-1	1	-1	1	-1
5	1	$\frac{-(1-j)}{\sqrt{2}}$	$-j$	$\frac{(1+j)}{\sqrt{2}}$	-1	$\frac{(1-j)}{\sqrt{2}}$	j	$\frac{-(1+j)}{\sqrt{2}}$
6	1	j	-1	$-j$	1	j	-1	$-j$
7	1	$\frac{(1+j)}{\sqrt{2}}$	j	$\frac{-(1-j)}{\sqrt{2}}$	-1	$\frac{-(1+j)}{\sqrt{2}}$	$-j$	$\frac{(1-j)}{\sqrt{2}}$

Observations

- There are only eight distinct values.
- Half the values are the negative of the other half, we might say that there are only four distinct values.
- The table forms a matrix with diagonal symmetry. Furthermore, many of the values are +1 or -1, implying simple additions and subtractions.

2 Algorithms will be presented

- The Conventional Decomposition Method
- The index-Mapping Method

It is important to understand that conventional decomposition and index-mapping are two ways of looking at the same problem. There is no essential difference between them.

In Both Methods, we require $N=2^{i,}$ where i is an integer

Conventional decomposition

- Conventional decomposition may be introduced by considering a signal x[n] broken down into shorter, interleaved, subsequences. The process is referred to as decimation-in-time, and gives rise to one of the major classes of FFT algorithm.
- Suppose we have a signal with N sample values, where N is an integer power of 2. We first separate x[n] into two subsequences, each with N/2 samples. The first subsequence consists of the even-numbered points in x[n], and the second consists of the odd-numbered points. Writing n=2r when n is even, and n=2r+l when n is odd, the DFT may be recast as:

Procedure

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \qquad 0 \le k \le N-1$$

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

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

$$W_N^2 = \exp(-2j2\pi / N) = W_{N/2}$$

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

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

$$= G[k] + W_N^k H[k]$$

Conclusion

- If we assume that the transform length N is an integer power of 2, it follows that N/2 is even. Therefore we can take the decomposition further, by breaking each N/2-point subsequence down into two shorter, N/4-point sub-sequences. The process can continue until, in the limit, we are left with a series of 2-point subsequences, each of which requires a very simple 2-point DFT. A complete decomposition of this type gives rise to one of the commonly-used radix-2, decimation-in-time, FFT algorithms.

Example

- We can demonstrate the decimation-in-time process quite easily in the case of a short transform - for example, N=8. The signal x[n] is defined for:

$$n = \{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\}$$

- We start by forming two subsequences of the even-numbered and odd-numbered points:

$$n = \{0\ 2\ 4\ 6\} \text{ and } \{1\ 3\ 5\ 7\}$$

- Each subsequence is further decimated, giving: n = {0 4} and (2 6} and {1 5}, and {3 7}

Conclusion

- The 8-point DFT is now realized as four 2-point DFTs, performed on pairs of sample values separated by half the transform length. Thus x[0] is paired with x[4], x[2] is paired with x[6], and so on. Although the original DFT has been recast in terms of a series of 2-point DFTs, we must be careful not to assume that the resulting computation is trivial.

Index-Mapping Method

Procedures

- The index-mapping approach may be introduced as follows. We start by expressing the transform length N as the product of two factors N1 and N2: .

$$N = N1\ N2$$

- We also define two new indices:

$$n_2 = 0, 1, 2, ..., N_2 - 1$$

$$n_1 = 0, 1, 2, ..., N_1 - 1$$

Computation of n

- The following linear equation may now be used to map the values of and into the time-index n of the DFT:

$$n = (M_1 n_1 + M_2 n_2)_N$$

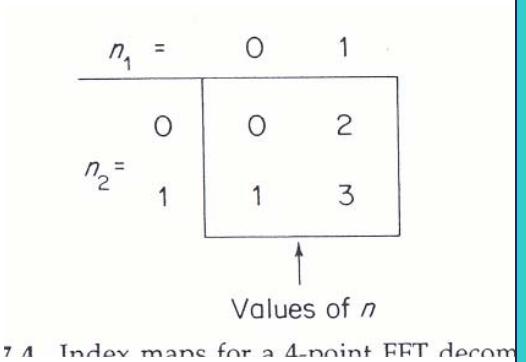
- where M1 and M2 are constants, and the brackets and subscript N denote modulo-N.

Example

- Let us begin with a simple example - a 4-point DFT which we will decompose into 2-point DFTs. We can use it to illustrate many important features of FFT algorithms, and to introduce further terminology. In this case N=4 and N1=N2=2. For reasons which will become clear as we proceed, we choose M1=2, M2=1.
- Hence:

$$n = 2n1 + n2$$

The mappings of n



7.4 Index maps for a 4-point FFT decom

$$n = 2n1 + n2$$

The DFT for N=4 is given by:

$$X[k]= \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{n=0}^{3} x[n] W_4^{nk}$$

$$X[k] = \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x[n_1, n_2] W_4^{k(2n_1+n_2)}$$

$$X[k] = \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x[n_1, n_2] W_4^{2kn_1} W_4^{kn_2}$$

$$X[k] = \sum_{n_2=0}^{1} W_4^{kn_2} \sum_{n_1=0}^{1} x[n_1, n_2] W_4^{2kn_1} = \sum_{n_1=0}^{1} x[n_1,0] W_4^{2kn_1} + W_4^{k} \sum_{n_1=0}^{1} x[n_1,1] W_4^{2kn_1}$$

$$X[k] = \sum_{n_1=0}^{1} x[n_1,0]W_4^{2kn_1} + W_4^k \sum_{n_1=0}^{1} x[n_1,1]W_4^{2kn_1}$$

- Let us examine the last equation a little more carefully. Referring to the index map for n, we see that if n2=0, then n=0, 2 when nl=0, 1; and if n2=1 then n=l, 3 when nl=0, 1. Hence:

$$X[k] = x[0] + x[2]W_4^{2k} + W_4^k \left[x[1] + x[3]W_4^{2k} \right]$$

- Therefore one of the 2-point DFTs involves pairing x[0] and x[2]; the other involves pairing x[1] and x[3]. Once again, we see that the decimation creates pairs of sample values separated by half the transform length.

$$X[k] = x[0] + x[2]W_4^{2k} + W_4^{k}\left[x[1] + x[3]W_4^{2k}\right]$$

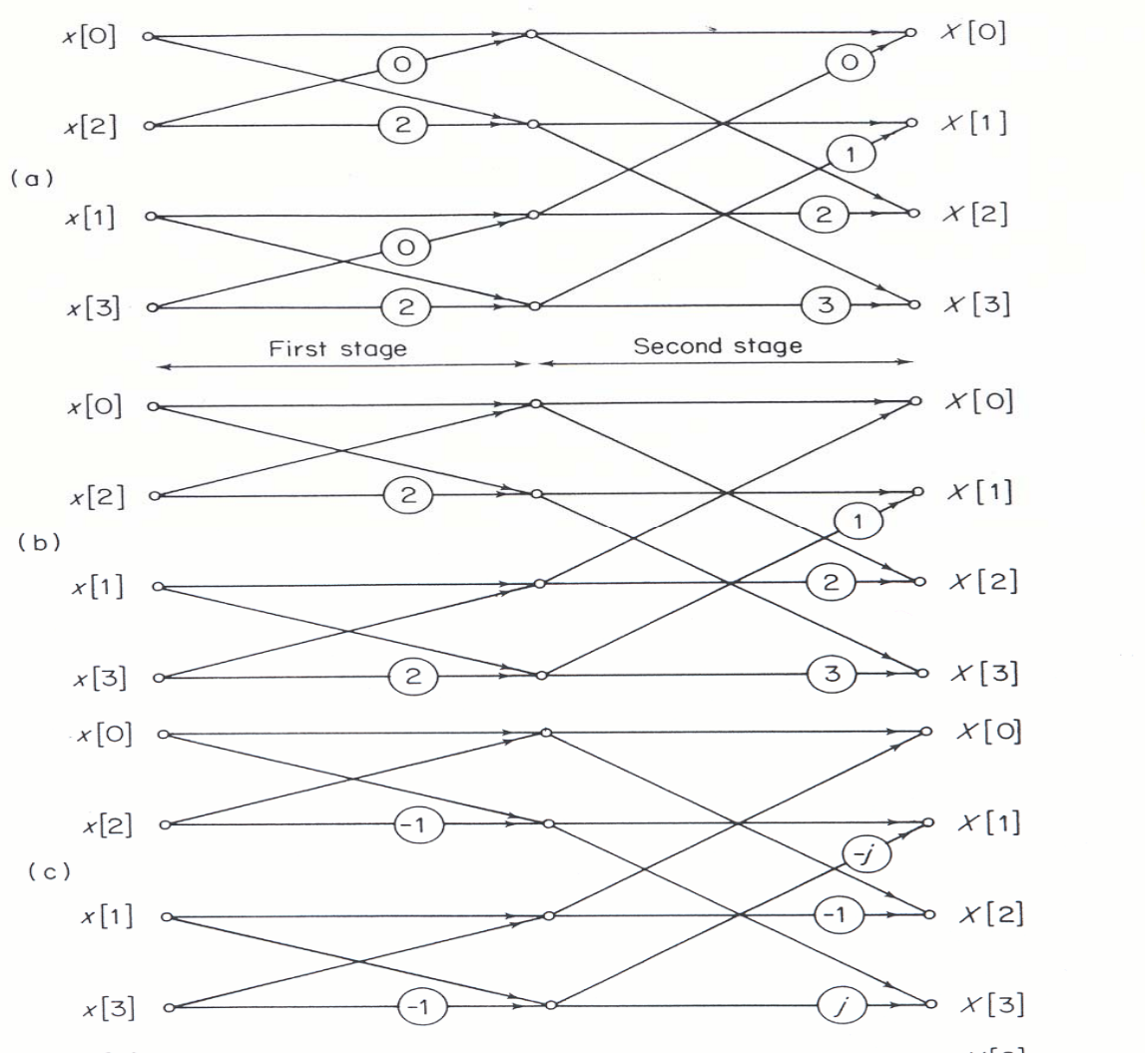
$$X[0] = \{x[0] + x[2]W_4^{0}\} + W_4^{0}\left[x[1] + x[3]W_4^{0}\right]$$

$$X[1] = \{x[0] + x[2]W_4^{2}\} + W_4^{1}\left[x[1] + x[3]W_4^{2}\right]$$

$$X[2] = \{x[0] + x[2]W_4^{0}\} + W_4^{2}\left[x[1] + x[3]W_4^{0}\right]$$

$$X[3] = \{x[0] + x[2]W_4^{2}\} + W_4^{3}\left[x[1] + x[3]W_4^{2}\right]$$

This way of representing a 4-point DFT may be visualized the help of the signal flow graph in the Figure below.



Example

An 8-pomt, radix-2, decimation in time, in-place, FFT may be defined by the following index-map equations

$$n = 4n_1 + 2n_2 + n_3$$

all six independent variables taken over the range 0 to 1.

- Construct index maps for n. Assuming, a natural-order output, specify the shuffled input sequence.
- Draw a signal flow graph for the FFT, expressing all branch transmittances as powers of W_8
- Redraw the flow graph in terms of basic (2-point) FFT butterflies and twiddle factors. How many complex multiplications, are required by the FFT?

Solution

n_1	n_2	n_3	n
0	0	0	0
1	0	0	4
0	1	0	2
1	1	0	6
0	0	1	1
1	0	1	5
0	1	1	3
1	1	1	7

b. The form of the DFT for N=8

$$X[k]= \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{n=0}^{7} x[n] W_8^{nk}$$

$$X[k] = \sum_{n_3=0}^{1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x[n_1, n_2, n_3] W_8^{k(4n_1+2n_2+n_3)}$$

$$X[k] = \sum_{n_3=0}^{1} W_8^{kn_3} \sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[n_1, n_2, n_3] W_8^{4kn_1}$$

$$X[k] = \sum_{n_3=0}^{1} W_8^{kn_3} \sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[n_1, n_2, n_3] W_8^{4kn_1}$$

$$X[k] = \sum_{n_3=0}^{1} W_8^{kn_3} \sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[4n_1 + 2n_2 + n_3] W_8^{4kn_1}$$

$$X[k] = W_8^0 \left(\sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[4n_1 + 2n_2 + 0] W_8^{4kn_1} \right) +$$

$$W_8^1 \left(\sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[4n_1 + 2n_2 + 1] W_8^{4kn_1} \right)$$

$$X[k] = W_8^0 \left(\sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[4n_1 + 2n_2 + 0] W_8^{4kn_1} \right) +$$

$$W_8^1 \left(\sum_{n_2=0}^{1} W_8^{2kn_2} \sum_{n_1=0}^{1} x[4n_1 + 2n_2 + 1] W_8^{4kn_1} \right)$$

$$X[k] = W_8^0 \left(W_8^0 \sum_{n_1=0}^{1} x[4n_1] W_8^{4kn_1} \right) + W_8^0 \left(W_8^{2k} \sum_{n_1=0}^{1} x[4n_1 + 2] W_8^{4kn_1} \right)$$

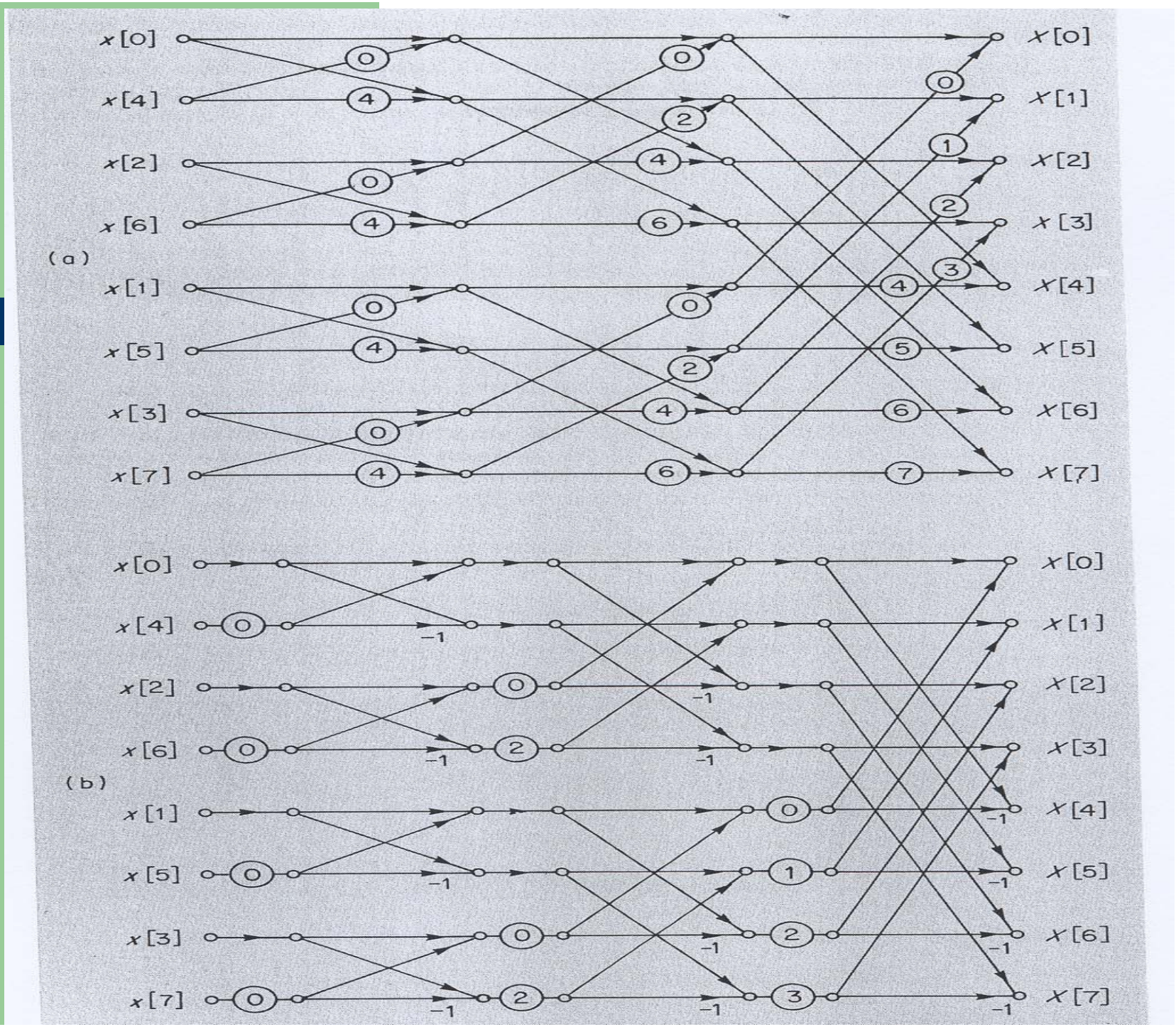
$$+ W_8^1 \left(W_8^0 \sum_{n_1=0}^{1} x[4n_1 + 1] W_8^{4kn_1} \right) + W_8^1 \left(W_8^{2k} \sum_{n_1=0}^{1} x[4n_1 + 2 + 1] W_8^{4kn_1} \right)$$

$$X[k] = W_8^0\left(W_8^0 \sum_{n_1=0}^{1} x[4n_1]W_8^{4kn_1} \right) + W_8^0\left(W_8^{2k} \sum_{n_1=0}^{1} x[4n_1+2]W_8^{4kn_1} \right)$$

$$+ W_8^1\left(W_8^0 \sum_{n_1=0}^{1} x[4n_1+1]W_8^{4kn_1} \right) + W_8^1\left(W_8^{2k} \sum_{n_1=0}^{1} x[4n_1+2+1]W_8^{4kn_1} \right)$$

$$X[k] = W_8^0\left(W_8^0\left[x[o]W_8^0 + x[4]W_8^{4k}\right]\right) + W_8^0\left(W_8^{2k}\left[x[2]W_8^0 + x[6]W_8^{4k}\right]\right)$$

$$+ W_8^1\left(W_8^0\left[x[1]W_8^0 + x[5]W_8^{4k}\right]\right) + W_8^1\left(W_8^{2k}\left[x[3]W_8^0 + x[7]W_8^{4k}\right]\right)$$


x[0]
x[4]
x[2]
x[6]
(a)
x[1]
x[5]
x[3]
x[7]
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]
(b)