FAST FOURIER TRANSFORM ALGORITHM FORMULATION

Tran Thong
Electronics Laboratory
General Electric Company
Syracuse, New York 13221

ABSTRACT

A new unified formulation of the fast Fourier transform based on the unwrapping of a multi-dimensional array is presented. The decimation in time FFT algorithms is treated in detail. The decimation in frequency algorithms is then discussed.

INTRODUCTION

The fast Fourier transform (FFT) is an efficient algorithm for computing the discrete Fourier coefficients of a time series.

The decimation in time FFT algorithm was discovered by Cooley and Tukey [1] . Subsequently five other algorithms [2-6] were derived by rearranging the positions of the nodes of the FFT flowgraph.

Gentleman and Sande [3] have proposed a different type of FFT - the decimation in frequency algorithms. Five similar algorithms [3-6] were subsequently derived.

A systematic method for deriving all FFT algorithms was proposed by Sloate [6] Using matrix Kronecker products, Sloate showed how, with the proper factorization of the permutation matrix, any of the known algorithms can be derived. These matrix manipulations are however not intuitively obvious.

In this paper we will present a new unified derivation of the FFT algorithms using a simpler algebraic formulation.

DECIMATION IN TIME ALGORITHMS

The discrete Fourier coefficients of a finite complex sequence $\{x(n)\}_{n=0}^{N-1}$ are defined to be

$$A(k) = \sum_{n=0}^{N-1} x(n) \exp\left[-j \frac{2\pi}{N} nk\right] \tag{1}$$

for k = 0, 1,...,N-1 , where exp is the exponential function and $j = (-1)^{1/2}$.

In this paper we treat only the radix 2 case when N is given by

$$N = 2^M \ . \tag{2}$$

The arbitrary radix cases can be similarly derived.

The integers k and n in Eq. (1) can be expressed as

$$k = \sum_{i=1}^{M} k_i \, 2^{i-1} \tag{3}$$

$$n = \sum_{i=1}^{M} n_i \, 2^{i-1} \tag{4}$$

where $n_i, k_i \in \{0, 1\}$.

Equation (1) can be rewritten as

$$A(k) = \sum_{n_1=0}^{1} \cdots \sum_{n_M=0}^{1} X_0(n_1, \ldots, n_M) \exp\left[-j \frac{2\pi}{N} \sum_{i=1}^{M} \sum_{\ell=1} n_i k_\ell \, 2^{i+\ell-2}\right] \tag{5}$$

where

$$X_0(n_1, \ldots, n_M) = x(n) \tag{6}$$

It can be shown [7] that if we compute the M -dimensional arrays $X_1, X_2,...,X_m$ as follows

$$\begin{aligned} X_m(n_1, \ldots, n_{M-m}, k_m, \ldots, k_1) = \\ X_{m-1}(n_1, \ldots, n_{M-m}, 0, k_{m-1}, \ldots, k_1) \\ + (-1)^{k_m} X_{m-1}(n_1, \ldots, n_{M-m}, 1, k_{m-1}, \ldots, k_1) \\ \exp\left[-j \frac{2\pi}{2^m} \sum_{i=1}^{m-1} k_i \, 2^{i-1}\right] \end{aligned} \tag{7}$$

for all values of the n_i's and k_i's, then

$$A(k) = X_M(k_M, \ldots, k_1) \tag{8}$$

Up until now we have assumed that the arrays X_m's are M dimensional arrays of N elements. While this is convenient in the derivation of Eq. (7), in the actual computation it is more practical to represent them as one dimensional arrays. The various decimation in time algorithms are obtained by using different unwrapping schemes to convert these M-dimensional arrays into one-dimensional arrays.

1. Algorithm T1 [4]

Performing a row -wise expansion, i.e.,

$$(d_1, d_2, \ldots, d_M) \longrightarrow \ell = \sum_{i=1}^{M} d_i \, 2^{M-i}$$

we map the multidimensional array X_m into the one-dimensional array Y_m as follows:

$$X_m(n_1, \ldots, n_{M-m}, k_m, \ldots, k_1) \longrightarrow Y_m(\ell_m)$$

where

$$\ell_m = \sum_{i=1}^{m} k_i \, 2^{i-1} + \sum_{i=m+1}^{M} n_{M-i+1} \, 2^{i-1} \tag{9}$$

For convenience we define the following one-dimensional array

$$X_m^1(n_1, \ldots, n_{M-m}, k_m, \ldots, k_1)_r = Y_m(\ell_m)$$

where the subscript r indicates a row-wise expansion.

Equations (7) and (9) define algorithm T1 (T for time). An example of T1 is illustrated in Figure 1.

From Eq. (9) we obtain for ℓ_o

$$\ell_o = \sum_{i=1}^{M} n_{M-i+1} \, 2^{i-1}$$

indicating that the one -dimensional input array X_o^1 is stored in reserve bit order.

The one -dimensional output array X_M^1 can be similarly shown to be identical to the array A.

It is also clear that the computation is in place since to the input pair

$X_{m-1}^{1}(n_1,\ldots,n_{M-m},{}^{0}_{1},k_{m-1},\ldots,k_1)_r$, is associated the output pair,

$X_m^{1}(n_1,\ldots,n_{M-m},{}^{0}_{1},k_{m-1},\ldots,k_1)_r$, which occupies the same position

in the one-dimensional array.

2. Algorithm T2 [1]

Performing a columnwise expansion i.e.

$$(d_1,d_2,\ldots,d_M) \longrightarrow \ell = \sum_{i=1}^{M} d_i\, 2^{i-1}$$

we can store the M-dimensional array X_m as a one-dimensional array $X_m^2(\ldots)_c$.

This algorithm is characterized by a sequential input and a reverse bit ordered output. It can also be shown that the computation is in place.

An example of algirithm T2 is illustrated in Figure 2.

Most computer programs to perform the FFT in core are based on either algorithm T1 or T2 or their decimation in frequency equivalent. The in-place property makes these algorithms attractive in core-bound computers.

3. Algorithms T3 - T8

Removing the restriction that the computation be in place, it is no longer necessary to place the index k_m in Eq. (7) in the same position as that of the index n_{M-m+1}.

Permutations of the indices n_i and k_j had the following six algorithms:

T3. $X_m^3(n_1,\ldots,n_{M-m},k_1,\ldots,k_m)_r$

T4.[4] $X_m^4(n_1,\ldots,n_{M-m},k_1,\ldots,k_m)_c$

T5. $X_m^5(k_1,\ldots,k_m,n_1,\ldots,n_{M-m})_r$

T6.[3] $X_m^6(k_1,\ldots,k_m,n_1,\ldots,n_{M-m})_c$

T7.[2] $X_m^7(k_m,\ldots,k_1,n_1,\ldots,n_{M-m})_r$

T8.[2] $X_m^8(k_m,\ldots,k_1,n_1,\ldots,n_{M-m})_c$

Algorithms T3 and T5 are characterized by reverse bit input and output streams and hence are of little interest to users. T4 and T6 are sequential input and output algorithms. T7 and T8 are isogeometric algorithms. Algorithms T3-T8 are illustrated in Figures 3-8. Table I summarizes the properties of algorithms T1-T8.

4. Other Algorithms

The eight algorithms described in Table I are by no means the only ones. There are $(M!)^M$ possible algorithms.

Relaxing the condition that all memory accesses and all computations for Eq. (7) be performed in a single step, we can derive another set of algorithms.

Two of these algorithms were described by Corinthios [8] . They are characterized by sequential input, sequential output and identical computational geometry. They are two step versions of algorithms T4 and T6.

We first consider the modified T4 algorithm. Starting with

$$X_{m-1}^4(n_1,\ldots,n_{M-m+1},k_1,\ldots,k_{m-1})_c$$

the data is resampled to become (skip this step for m = 1)

$$X_m^{4\,*}(n_1,\ldots,n_{M-m},k_;,\ldots,k_{m-1},n_{M-m+1})_c$$

This array is then processed according to Eq. (7) to yield

$$X_m^4(n_1,\ldots,n_{M-m},k_1,\ldots,k_m)_c$$

With the modified T6 algorithm, we start with

$$X^{6}_{m-1}\,(k_1, \ldots, k_{m-1}, n_1, \ldots, n_{M-m+1})_c$$

and compute the following quantity according to Eq. (7)

$$X^{6\,*}_{m}\,(k_1, \ldots, k_{m-1}, n_1, \ldots, n_{M-m}, k_m)_c$$

It is then reshuffled to become

$$X^{6}_{m}\,(k_1, \ldots, k_m, n_1, \ldots, n_{M-m})_c$$

A hardware implementation of these algorithms is discussed in [8].

DECIMATION IN FREQUENCY

It can be shown [7] that the discrete Fourier transform coefficients can be computed by the following M-step process

$$X_m(n_1, \ldots, n_{M-m}, \ldots, k_1) =$$

$$\Big[X_{m-1}(n_1, \ldots, n_{M-m}, 0, k_{m-1}, \ldots, k_1)$$

$$+ (-1)^{km}\; X_{m-1}(n_1, \ldots, n_{M-m}, 1, k_{m-1}, \ldots, k_1)\Big]$$

$$\exp -j\,\frac{2\pi}{2^{M-m+1}}\;\; k_m \sum_{i=1}^{M-m} n_i\, 2^{i-1} \tag{10}$$

where

$$X_o(n_1, \ldots, n_M) = x(n) \tag{11}$$

$$A(k) = X_m(k_M, \ldots, k_1) \tag{12}$$

As with the decimation in time algorithms, Eq. (10) is best evaluated using one dimensional arrays. Hence all the unwrapping algorithms discussed in the previous section applies. Illustrations of algorithms F1-F8 (F for frequency) are shown in Figures 9-18.

CONCLUSIONS

In this paper we have presented a new simple method for deriving all FFT algorithms. Our purpose here is not to present an exhaustive list of all possible algorithms (> (M !)M) but to point out that all FFT algorithms can be decomposed into two parts.

1. Computation : Eqs. (7) and (10)

2. Data shuffling

The various algorithms differ only in the way data are shuffled.

With this paper, we hope that hardware signal processors now have a tool to develop FFT algorithms which are tailored to their hardware requirements instead of the opposite situation which has existed from the early days of digital signal processing.

REFERENCES

1. J.W. Cooley and J.W. Tukey, “An algorithm for the machine calculation of complex Fourier series”, Math. Comput., Vol. 19, pp 297-301, April 1965.

2. R.C. Singleton, “A method for computing the fast Fourier transform with auxiliary memory and limited high speed storage”, IEEE Trans. Audio Electroacoust., Vol. AU-15, pp 91-97, June 1967.

3. W.M. Gentleman and G. Sande, “Fast Fourier transforms -- for fun and profit”, 1966 Fall Joint Comput. Conf ., AFIPS Conf. Proc., Vol. 29, pp 563-578, 1966.

4. W. T. Cochran et al., “What is the fast Fourier transform?”, IEEE Trans. Audio Electroacoust., Vol. AU-15, pp 45-55, June 1967.

5. M.C. Pease, “An adaptation of the fast Fourier transform for parallel processing”, J. Ass. Comput. Mach., Vol. 15, pp 252-264, April 1968.

6. H. Sloate, “Matrix representation for sorting and the fast Fourier transform”, IEEE Trans. Circuits and Systems, Vol. CAS-21, pp 109-116, January 1974.

7. E.O. Brigham, The Fast Fourier Transform, Englewood Cliffs, N.J.: Prentice-Hall, 1974.

8. M.J. Corinthios, “The design of a class of fast Fourier transform computers”, IEEE Trans. Computers, Vol. C-20, pp 617-623, June 1971.

TABLE 1. FFT ALGORITHMS

Algorithms	1-D Expansion	Input Order	Output Order	Characteristics
T1, F1	$X_m^1(n_1,\ldots,n_{M-m},k_m,\ldots,k_1)_r$	Reverse bit	Sequential	In Place
T2, F2	$X_m^2(n_1,\ldots,n_{M-m},k_m,\ldots,k_1)_c$	Sequential	Reverse bit	In Place
T3, F3	$X_m^3(n_1,\ldots,n_{M-m},k_1,\ldots,k_m)_r$	Reverse bit	Reverse bit	Same Output Geometry
T4, F4	$X_m^4(n_1,\ldots,n_{M-m},k_1,\ldots,k_m)_c$	Sequential	Sequential	Same Output Geometry
T5, F5	$X_m^5(k_1,\ldots,k_m,n_1,\ldots,n_{M-m})_r$	Reverse bit	Reverse bit	Same Input Geometry
T6, F6	$X_m^6(k_1,\ldots,k_m,n_1,\ldots,n_{M-m})_c$	Sequential	Sequential	Same Input Geometry
T7, F7	$X_m^7(k_m,\ldots,k_1,n_1,\ldots,n_{M-m})_r$	Reverse bit	Sequential	Isogeometric
T8, F8	$X_m^8(k_m,\ldots,k_1,n_1,\ldots,n_{M-m})_c$	Sequential	Reverse bit	Isogeometric

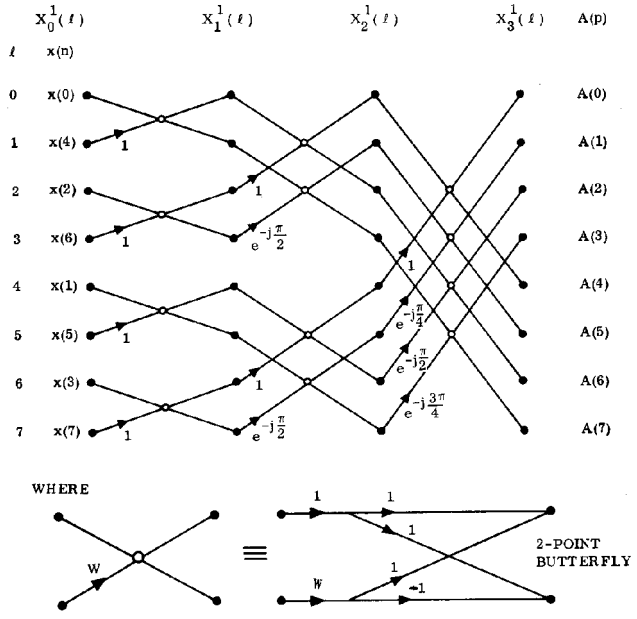

Figure 1. T1 Flowgraph for N = 8

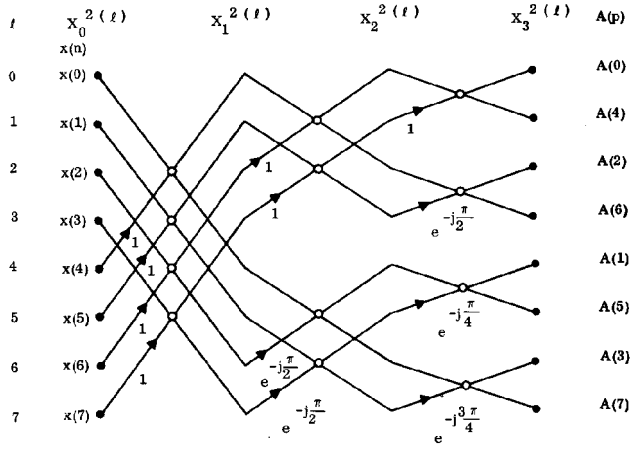

Figure 2. T2 Flowgraph for N = 8

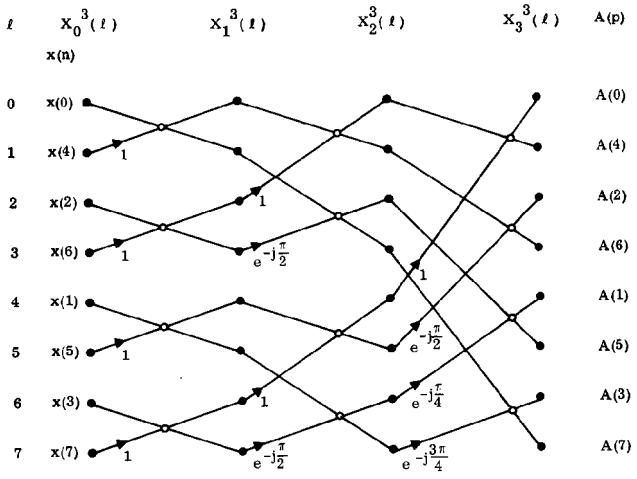

Figure 3. T3 Flowgraph for N = 8

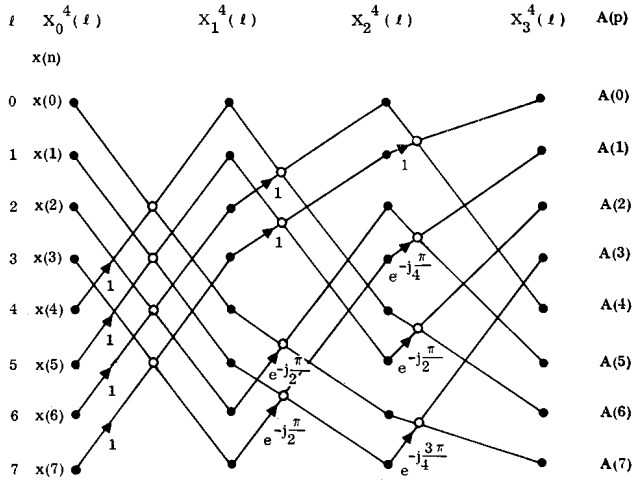

Figure 4. T4 Flowgraph for N = 8

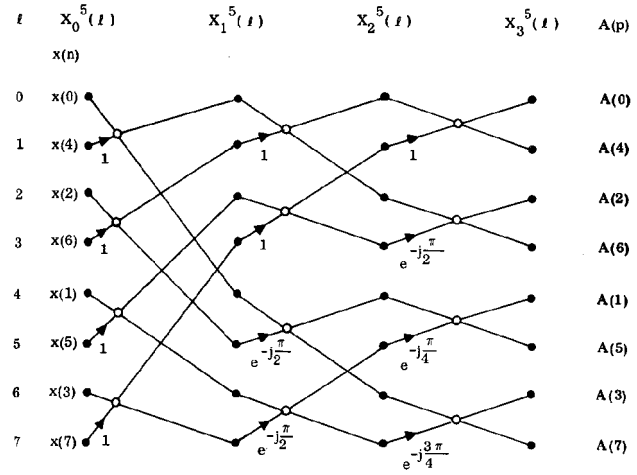

Figure 5. T5 Flowgraph for N = 8

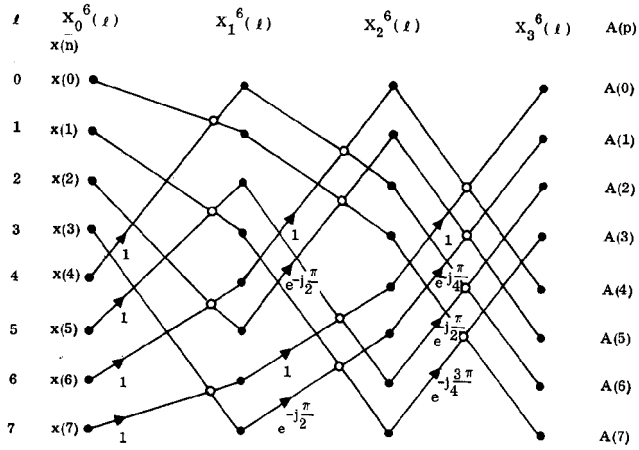

Figure 6. T6 Flowgraph for N = 8

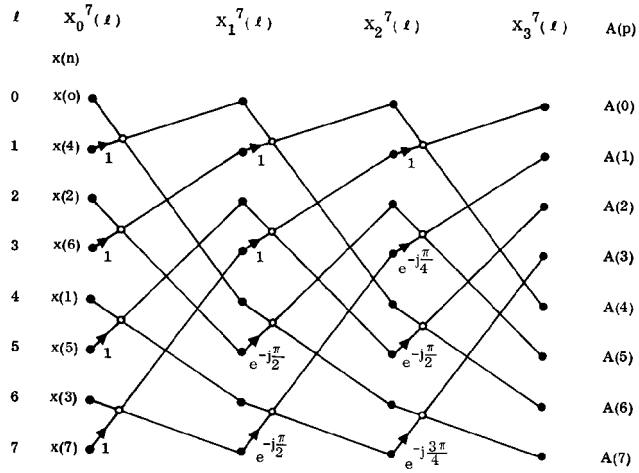

Figure 7. T7 Flowgraph for N = 8

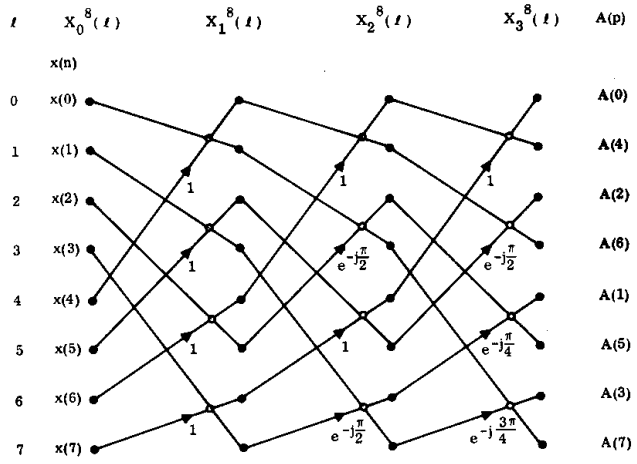

Figure 8. T8 Flowgraph for N = 8

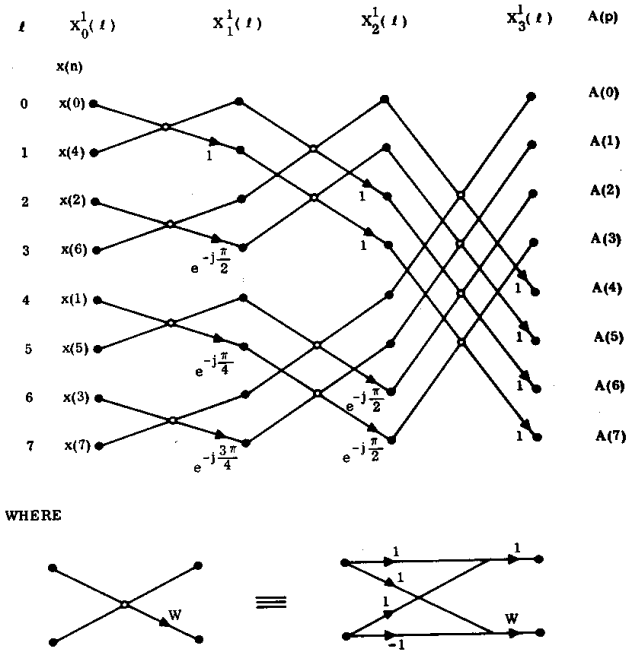

Figure 9. F1 Flowgraph for N = 8

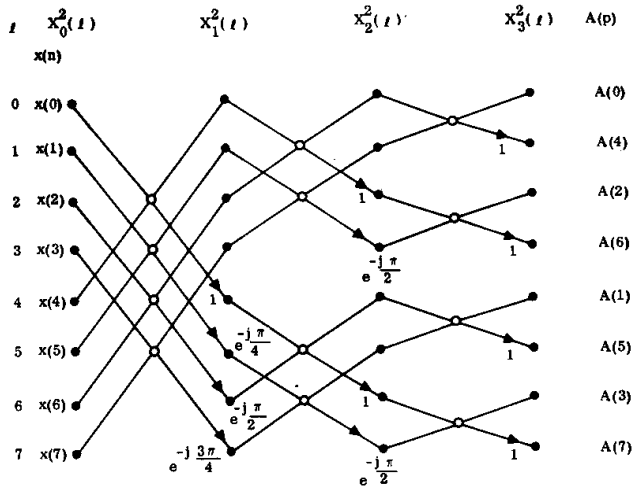

Figure 10. F2 Flowgraph for N = 8

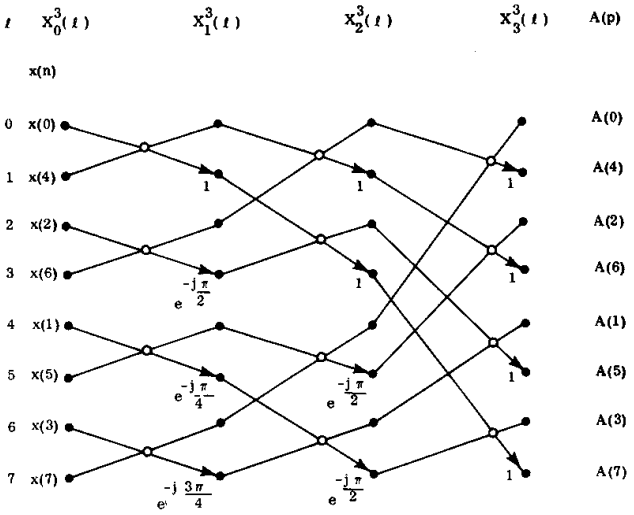

Figure 11. F3 Flowgraph for N = 8

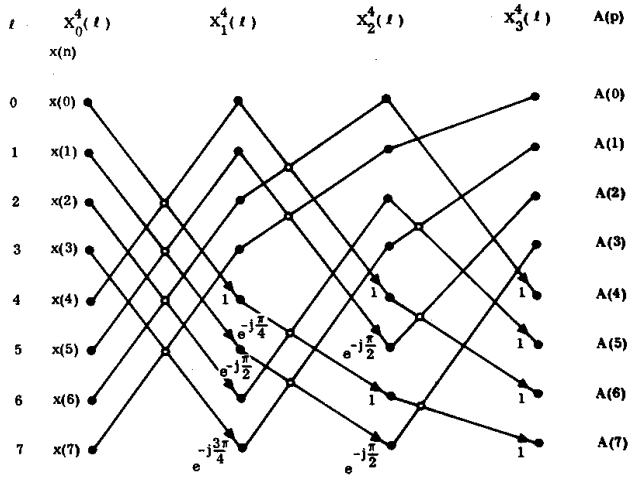

Figure 12. F4 Flowgraph for N = 8

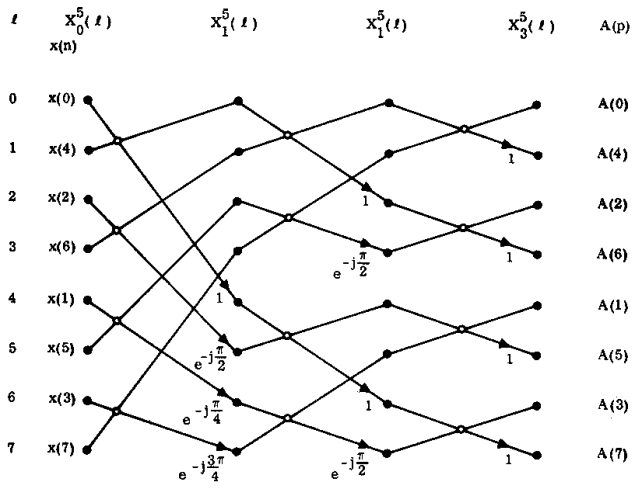

Figure 13. F5 Flowgraph for N = 8

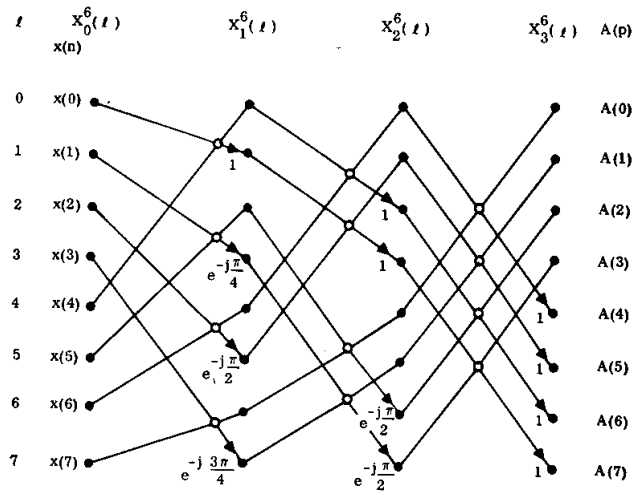

Figure 14 . F6 Flowgraph for N = 8

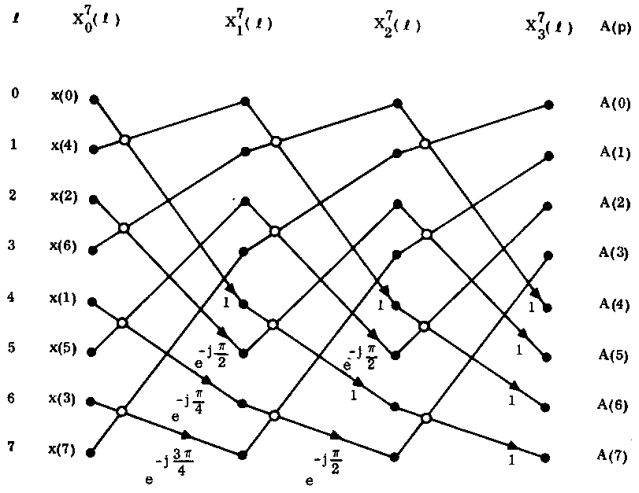

Figure 15. F7 Flowgraph for N = 8

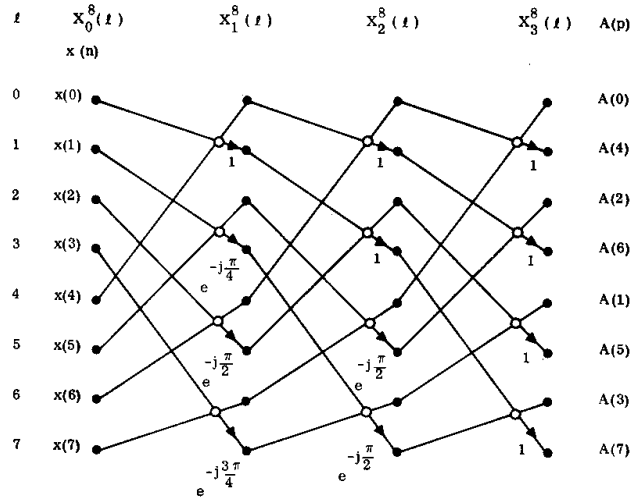

Figure 16. F8 Flowgraph for N = 8