VLSI-II
2nd version which passed 2nd review
Figs: Renewed (artwork done
~~Accepted~~ for publication in VLSI journal (sent in March 92)

MULTIDIMENSIONAL FOURIER TRANSFORMS BY SYSTOLIC ARCHITECTURES

Roziner, T.D. and Karpovsky, M.G.

Research Laboratory for Design and Testing of Computers and Communication Systems

Department of Electrical,Computer, and Systems Engineering

Boston University

44 Cummington Street, Boston, Massachusetts 02215

(617) 353-4840
(617) 353-9592

Abstract

In this paper, time-efficient systolic and semi-systolic architectures for the implementation of multidimensional DFTs are proposed that allow modularity and easy expansibility while keeping throughput independent of the dimension of the DFT. We shall call an array semi-systolic if the input data is to be preloaded into every cell of the array or if the output data can be calculated not only in the boundary cells of the array. DFT algorithms are represented by FORTRAN-like code, in order to explicitly display the suggested rotational transformations in the index space. After the transformation, multidimensional DFT algorithm can be mapped onto a systolic structure. The area-time^2 complexity of the proposed design is within log k factor of the lower bound for a k-point DFT ($AT^2=\Omega(k^2\log^2 k)$), i.e. equals $O(k^2\log^3 k) = O(N^{2M}\log^3 N)$ for $k = N^{2M}$-point DFT; A is the area complexity and T is the system throughput.

1. INTRODUCTION.

Fast Fourier Transforms (FFT) are well known to play an important role in many application fields such as spectral analysis, digital filtering, image processing, video transmission, etc. Increasing demands for high speed in many real-time applications have stimulated, in recent years, the development of a number of new very fast FT algorithms with the reduced number of multiplications [17,18,20,33-36]. However, the decrease in the number of multiplications in a very fast FT algorithm involves in most cases an increase in the number of additions and/or preprocessing operations. From the viewpoint of a further advance as to speed increase of FFTs, and with a progress in VLSI technology, multiprocessor highly parallel systems (in particular, systolic array architectures) become an attractive alternative compared to uniprocessor systems.

In recent time, there has been increasing interest in implementing different types of algorithms with systolic and semi-systolic array structures since these architectures are extremely suitable for VLSI technology (e.g.[2-8,10,14-17,24,25,27-28,30,50-54]). With increasing demands for high speed computations, the development of 3-D VLSI chips has created a challenge to speed up algorithm computations by going beyond 1-D and 2-D systolic networks. In some cases, 3-D systolic arrays turn out to be faster, and the 3-D VLSI technology presents additional advantages like shorter and more systematic wire routing as well as the higher density of the circuit. To derive maximal benefits from the 3-D VLSI technology, the existing algorithms must be transformed, to allow easy mapping to multidimensional systolic architectures.

Many authors suggested various formal transformation methods for mapping algorithms onto systolic structures[2-8,11-13,25,28]. A detailed classification of different approaches to the design of algorithmically specified systolic arrays can be found in [13].

The topic of our interest is the efficient systolic implementation of two- and multidimensional DFTs. It is well known that the 1-D FFT is faster than the 1-D DFT, and there are powerful 1-D FFT chips available in the market (e.g. [45,46]). However, as it was shown with convincing examples in [9], there are certain problems in signal processing when the 1-D DFT, if properly implemented, turns out to be more efficient than the 1-D FFT within a wide range of transform sizes. In [9], the authors compare the computation of the DFT algorithm on an MISD machine (multiple instruction single data) to the computation of the FFT algorithms on an SIMD machine (single instruction multiple data); in certain cases, the comparison is in favor of the DFT.

The 2-D (N x N) FFT can be computed as a sequence of two 1-D FFT transforms (well-known approach "FFT by rows and after that FFT by columns"; all rows are processed in parallel as well as all columns). However, between the two steps, the matrix of the intermediate spectrum is to be transposed, and this step significantly reduces the efficiency of FFT that can be reached in each stage. For a large transform size, the matrix can exceed the computer memory which presents another difficulty. Various methods of matrix partitioning[37-39] were proposed as well as other approaches [15,40-44] to maintain the efficiency of FFT in case of two or more dimensions. As shown e.g. in [23,15], the lower bound on a complexity for an M-dimensional k-point FFT is

$$AT^2 = \Omega(k^2 \log^2 k) \quad , \qquad (1)$$

where A is the computation area (the number of processing cells) and T is the computation time for one spectrum set. It should be emphasized [23,15] that T is the circuit's throughput, not its delay, that is taken as time performance measure T. Formula (1) allows us to obtain the following lower bound for the AT^2 complexity [15,23,26] for M-dimensional FFT:

$$AT^2 = \Omega(k^2 \log^2 k) = \Omega(N^{2M} \log^2 N), \text{ for } k = N^M \qquad (2)$$

The design proposed in [15] attains the lower bound on a complexity for the multidimensional FT transform while the designs of [24,25] are within log N factor of the lower bound. In all designs of [15,24,25], the computation area is independent of the number of transform dimensions; however, all three designs require repeated passes of the array of processing cells in different directions (pipelining in time). The latency of the performance in this case increases linearly with the number of dimensions; consequently, T increases as the squared number of FT dimensions). Also, the number of pins for each processing cell increases; if a cell is passed three times (for 3-D FFT) (each pass in the direction of one of the coordinate axes [25]), six input and six output multibit data lines are required for each cell.

An attractive area-efficient approach for the implementation of convolution is suggested in [53]. It would at first appear that with minor modification, this approach can be used to implement 2-D DFT and the design will attain the lower bound on complexity. However, the implementation of a 2-D DFT would require a routing network with complexity dominating over the complexity of the other (area-efficient) parts. The design of [53] in relation to the 2-D DFT is discussed in more detail in Section 3 of this paper.

The design proposed in this paper can be defined as pipelining in space; it is a macropipeline of two or more cascaded systolic and semi-systolic arrays with slightly different cells. We shall call an array semi-systolic [1] if it requires data injection into each cell of the array or if it requires output not only from the boundary cells of the array.

At the expense of an increase in the area complexity, the throughput can be made independent of the number of dimensions. For N^M-point DFT, the throughput equals N clocks. In other words, one set of N^M spectrum values is completed every N clocks of the array pipeline (after the initial delay while the pipeline is being filled).

It should be noted that some designs using macropipelining of systolic arrays were proposed at the recent time (e.g. [27,28]) for different purposes. In [28], macropipelines of systolic arrays are described where the computation arrays are alternated with buffer systolic arrays that perform data format conversion (restructuring) only. In [27], a detailed analysis is performed (resulting in the exhaustive list of all possible efficient systolic arrays), for a specific problem (solution of Toeplitz system of linear equations).

2. THE TRANSFORMATION OF THE MULTIDIMENSIONAL DFT ALGORITHM TO A COMPLETELY SYSTOLIZED FORM.

Consider k-point DFT where $k = N^M$. By definition, an M-dimensional DFT is calculated as follows:

$$X(i_{M-1},\dots, i_0) = \sum_{j_{M-1}=0}^{N-1} \dots \sum_{j_0=0}^{N-1} x(j_{M-1},\dots,j_0)\, W_N^{i_0 j_0 + \dots + i_{M-1} j_{M-1}} \qquad (3)$$

where $W_N = \exp(-2\pi j/N)$, $j^2 = -1$, and $0 \leq i_p, j_p \leq N-1$, $0 \leq p \leq M-1$.

Algorithm (3) can be rewritten in the form corresponding to M steps of spectrum computation (for M=2, DFT by rows, then DFT by columns of the intermediate spectrum). Each step is a 1-D DFT by one of the variables j_{p-1}, $p = 1,2,\dots,M$:

$$P_0(j_{M-1},\dots,j_0) \overset{\Delta}{=} x(j_{M-1},\dots,j_0);$$

$$P_p(j_{M-1},\dots, j_p, i_{p-1}, i_{p-2},\dots,i_0) =$$

$$= \sum_{j_{p-1}=0}^{N-1} P_{p-1}(j_{M-1},\dots,j_p,j_{p-1},i_{p-2},\dots,i_0)\, W_N^{i_{p-1} j_{p-1}},$$

$p = 1,\dots, M;$

$$P_M(i_{M-1},\dots,i_0) \overset{\Delta}{=} X(i_{M-1},\dots,i_0); \qquad (4)$$

$P_p(*,\dots,*)$, $p = 1,\dots, M-1$, are the intermediate spectra.

Algorithm (4) can be written in the form of a high-level language program that consists of M parts and each part has (M+1) nested DO-loops, since one variable is introduced as the accumulation step number. For example, for M = 2 there are two parts of the algorithm, each containing three nested DO-loops. For each participating variable, the leftmost M arguments can be considered "coordinates" of a point in the M-dimensional array of data, and the rightmost argument stands for the accumulation step number.
It will be more convenient to denote the loop indices in each part by the same index set m_i, $i = 0,1,\dots,M-1$. Using a loose notation, we assume that P_0 is initialized as input data and that P_p, $1 \leq p \leq M$, is initialized as 0, for the accumulation step number "-1." The algorithm of (4) can be written as follows:

```
Part p, 1≤p≤M :                      Comments

DO 10 m_0     = 0, N-1               $m_0 = j_{M-1}$
.....................                .............
DO 10 m_{M-p-1} = 0, N-1             $m_{M-p-1} = j_p$
DO 10 m_{M-p}   = 0, N-1             $m_{M-p} = i_{p-1}$
                                   (no-change index for $P_{p-1}$)
```

```
DO 10 $m_{M-p+1}$ = 0, N-1            $m_{M-p+1} = i_{p-2}$
........................            ......................
DO 10 $m_{M-1}$   = 0, N-1            $m_{M-1} = i_0$
DO 10 $m_M$       = 0, N-1            $m_M = j_{p-1}$ (accumulation index
                                                  for $P_p$)
```

$$P_{p-1}(m_0,\dots,m_{M-p-1},m_M,m_{M-p+1},\dots,m_{M-1},m_{M-p}) =$$
$$= P_{p-1}(m_0,\dots,m_{M-p-1},m_M,m_{M-p+1},\dots,m_{M-1},m_{M-p}-1) ;$$

$$P_p(m_0,\dots,m_M) = P\,(m_0,\dots,m_M-1) +$$
$$+ P_{p-1}(m_0,\dots,m_{M-p-1},m_M,m_{M-p+1},\dots,m_{M-1},m_{M-p})\, W_N^{m_{M-p}m_M} \qquad (5)$$

Note that the algorithm in its original form (5) does not allow application of the well-known mapping technique (e.g. of [5-7]) onto a systolic structure since the algorithm cannot be characterized by a single dependence matrix (the same one for all steps). We suggest the formal transformation of the algorithm index space (equivalent to different index space rotations in different steps of the algorithm, cf. geometric transformations in [12,27]) which results in the "completely systolized" form of the algorithm for any number of dimensions. The transformed index space allows application of the mapping technique and mapping the algorithm onto a pipeline of systolic/semi-systolic arrays.

The method of the original algorithm transformation has been developed for any number of dimensions. To obtain in general case the form of the "completely systolized" DFT algorithm, certain permutations of loop index order are needed. As a result, the algoritm will be transformed to the form that could be described by a single (M+1)x(M+1) dependence matrix for the N^M DFT. The rules of transformation are as follows.

Let (i,j) denote the permutation of m_i and m_j in the index set:

$$(i,j)(m_0,\dots,m_M) = (m_0,m_1,\dots,m_{i-1},m_j,m_{i+1},\dots,m_{j-1},m_i,m_{j+1},\dots,m_M).$$

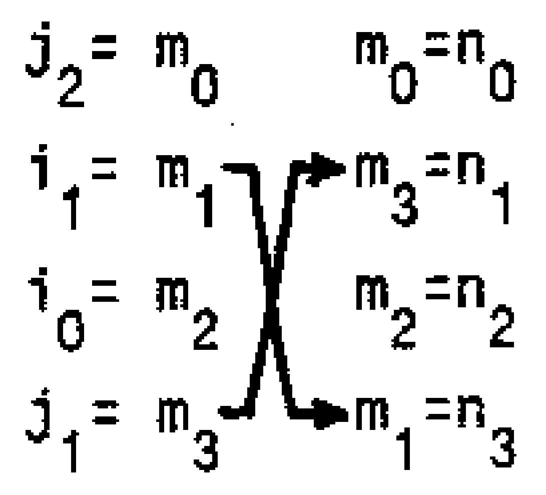
To systolize the algorithm (5), the following permutations are needed.
In Part 1: no permutations.
In Part 2: apply (M,M-2) to the index set of Part 2.
In Part p, $3 \leq p \leq M$: apply (M-3,M-2)...(M-p,M-p+1)(M,M-p) (starting from the rightmost permutation) to the index set of Part p.

Example 1. For 3-D DFT (M=3), the permutations to be applied are as shown:

Part 1

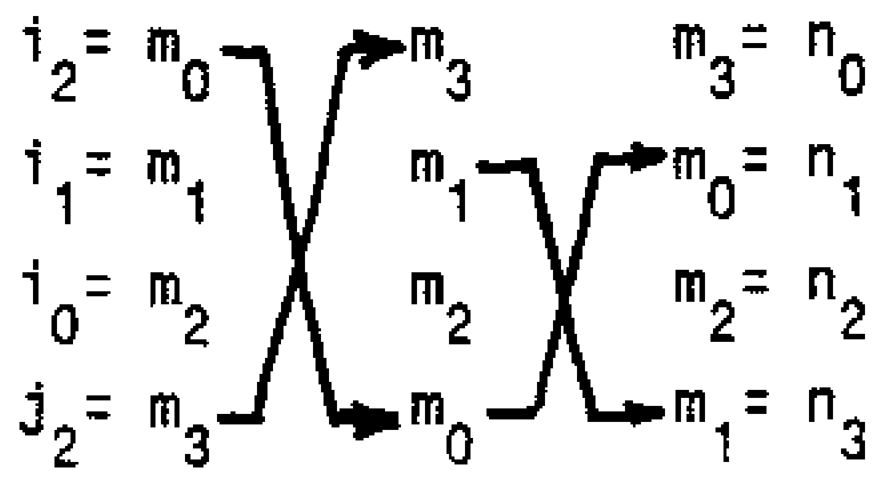
$j_2 = m_0 = n_0$
$j_1 = m_1 = n_1$
$i_0 = m_2 = n_2$
$j_0 = m_3 = n_3$

Part 2 (p=2)



(3,1)

Part 3 (p=3)



(3,0) (0,1)

We will describe now the implementation of the N^M DFTs with cascaded systolic and semi-systolic arrays for M = 2, 3. The approach can be extended to any M.

Consider 2-D k-point DFT ($k = N^2$) .

$$X(i_1,i_0) = \sum_{j_1=0}^{N-1} \sum_{j_0=0}^{N-1} x(j_1,j_0)\, W_N^{i_1 j_1 + i_0 j_0} \tag{6}$$

where $W_N = \exp(-2\pi j/N)$; $j^2 = -1$. Formula (6) can be rewritten in the form corresponding to the two steps of the spectrum computation (DFT by rows, then DFT by columns of the intermediate spectrum $P(j_1,i_0)$):

$$P(j_1,i_0) = \sum_{j_0=0}^{N-1} x(j_1,j_0)\, W_N^{i_0 j_0} \ ; \qquad X(i_1,i_0) = \sum_{j_1=0}^{N-1} P(j_1,i_0)\, W_N^{i_1 j_1} \tag{7}$$

Algorithm (7) can be presented in the form of a FORTRAN-like program. (For each participating variable, the leftmost argument is the row number, the middle one is the column number, and the rightmost argument stands for the accumulation step number). Note that the algorithm in its original form does not allow us to apply the mapping technique (e.g of [5,6]) onto a systolic structure (as in the case of 2-D FFT, the matrix of the intermediate spectrum is to be transposed). In other words, the algorithm of (4) (and consequently of (7)) cannot be described by a single dependence matrix for all steps.

For 2-D DFT, the transformation needed is a single permutation of indices (2,0) in the second part of the algorithm. The algorithm of completely systolized 2-D DFT (NxN points) can be written as follows:

```
                                                              Comments
DO 10 $n_0$ = 0,N - 1                                           $n_0 = m_0$
DO 10 $n_1$ = 0,N -1                                            $n_1 = m_1$
     $P(n_0,n_1,-1) = 0$  (initialize)
DO 10 $n_2$ = 0,N-1                                             $n_2 = m_2$
     $x(n_0,n_1,n_2) = x(n_0,n_1-1,n_2)$
     $P(n_0,n_1,n_2) = P(n_0,n_1,n_2-1) + x(n_0,n_1,n_2)W_N^{n_1 n_2}$
10 CONTINUE

DO 20 $n_0$ = 0,N-1                                             $n_0 = m_2$
DO 20 $n_1$ = 0,N-1                                             $n_1 = m_1$
DO 20 $n_2$ = N,2N-1                                            $n_2 = m_0$
     $P(n_2,n_1,n_0) = P(n_2,n_1,n_0-1)$
     $X(n_0-N,n_1,n_2) = X(n_0-N,n_1,n_2-1) + P(n_2,n_1,n_0)W_N^{n_0 n_2}$
20 CONTINUE
```

(8)

The DFT algorithm in the form (8) can be characterized by a single dependence matrix (variable P appears in the loops labeled 10 and 20 in the same form). The form (8) does not follow from (7) in a straightforward manner; we have applied an intermediate transformation of the loop index set, to make the approach of [5,6] applicable. Note that in (8) one loop index has the shifted range (N,2N-1); this is done to obtain the correct timing in the final version of the algorithm, after the mapping onto a systolic structure. The general rule found by us is that the range of the index used as an accumulation index in one of the parts of the completely systolized algorithm must become (N,2N-1) in the following parts.

The dependence matrix D for (8) is

$$D = \begin{array}{ccc} \left[\begin{array}{ccc} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right] & \begin{array}{c} n_0 \\ n_1 \\ n_2 \end{array} \\ \begin{array}{ccc} x & P & X \end{array} & \end{array}$$

(x formally varies by n_1, P varies by n_2, X varies by n_0).

A non-singular transform (given by a 3x3 matrix T) of the index set (n_0,n_1,n_2) to the time-space "new" index set (t,s_0,s_1)can be found, to map the algorithm onto a systolic structure. In the new index set, t is the integer-valued time, and s_0, s_1 are the space coordinates of the processing element in the 2-D systolic array. The transformation T can be found if one selects a priori the new dependence matrix $\hat{D}$ = TD. We accept certain limitations as to the form of $\hat{D}$ that follow the requirements listed below.

Our criteria for the choice of $\hat{D}$ were as follows:

- to have no idle processors at any macropipeline clock (which implies all 1's in the upper row of $\hat{D}$);
- to have communications only between adjacent cells in the array(s);
- to minimize the number of input/output pins (that is, the data are to move through constituent arrays in two directions only);
- to minimize the computation time (i.e., to have the maximum throughput dependent on the size of one dimension only).

In 2-D case, there are three possibilities for the choice of $\hat{D}$ which corresponds to three different efficient systolic designs for k-point DFT, $k = N^2$. Since there are only two space coordinates (s_0,s_1) in the transformed ("new") index set, and we have three variables x, P, and X to move along the new coordinate axes, each of the three possible designs corresponds to the case when one of the variables x,P, or X does not move in the array(s) of the processing elements (PEs).This restriction implies semi-systolic features of the chosen design options:

a) Variable x (input data) does not move: this means data preload into the first array of the macropipeline that consists of two cascaded N x N arrays of processing cells .(Note that data preload does not need all the data set simultaneously; on each array clock, one data value is fed into one of the cells in the first array so that one input bus for each column is needed).

b) Variable X (output spectrum) does not move, i.e. X is accumulated in the cells of the second array. Therefore, each processing cell of the second array must have communication with the outside world. However, at any pipeline clock only one cell in each array column generates an output value so that for N x N array N output bus lines can be used .

c) Variable P (intermediate spectrum) does not move, i.e. P is being accumulated inside the processing cells. In this case, the 2-D DFT can be implemented with a single array of N x N processing cells.

Example 2. To illustrate the results, consider the case of a two-dimensional 4-point DFT ($N=M=2$, $k = N^M = 4$).
Figure 1 shows the three designs ("a","b",and "c") :

a) x is preloaded (does not move); P moves by s_0, and X moves by s_1; two cascaded arrays of the size N x N are needed .

b) x moves by s_0, P moves by s_1, and X is accumulated in the PEs); two cascaded arrays of the size N x N are needed.

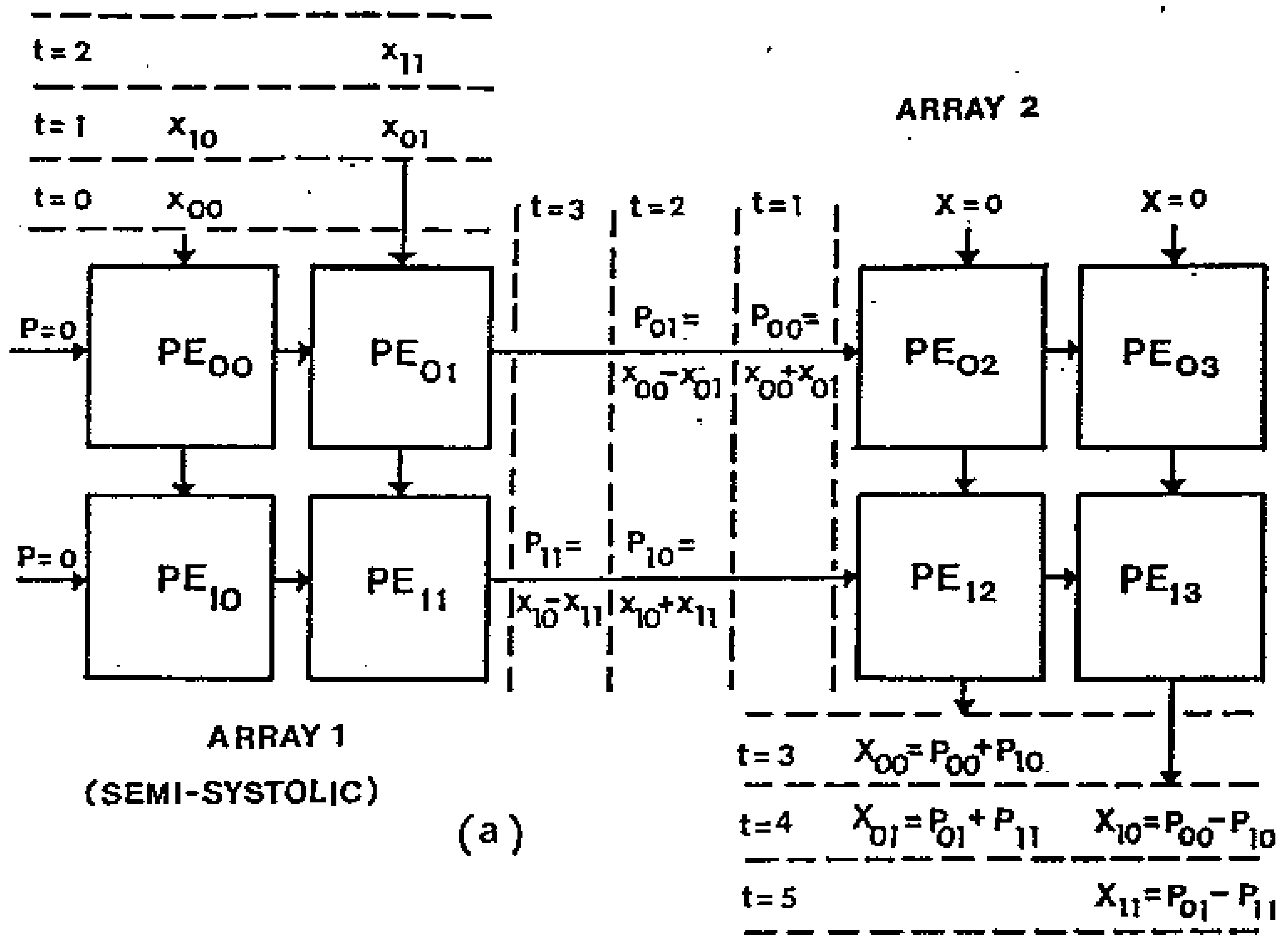
c) x moves by s_0, P accumulates in the PEs, and X moves by s_1. One N x N array (with cells of double complexity) is needed.

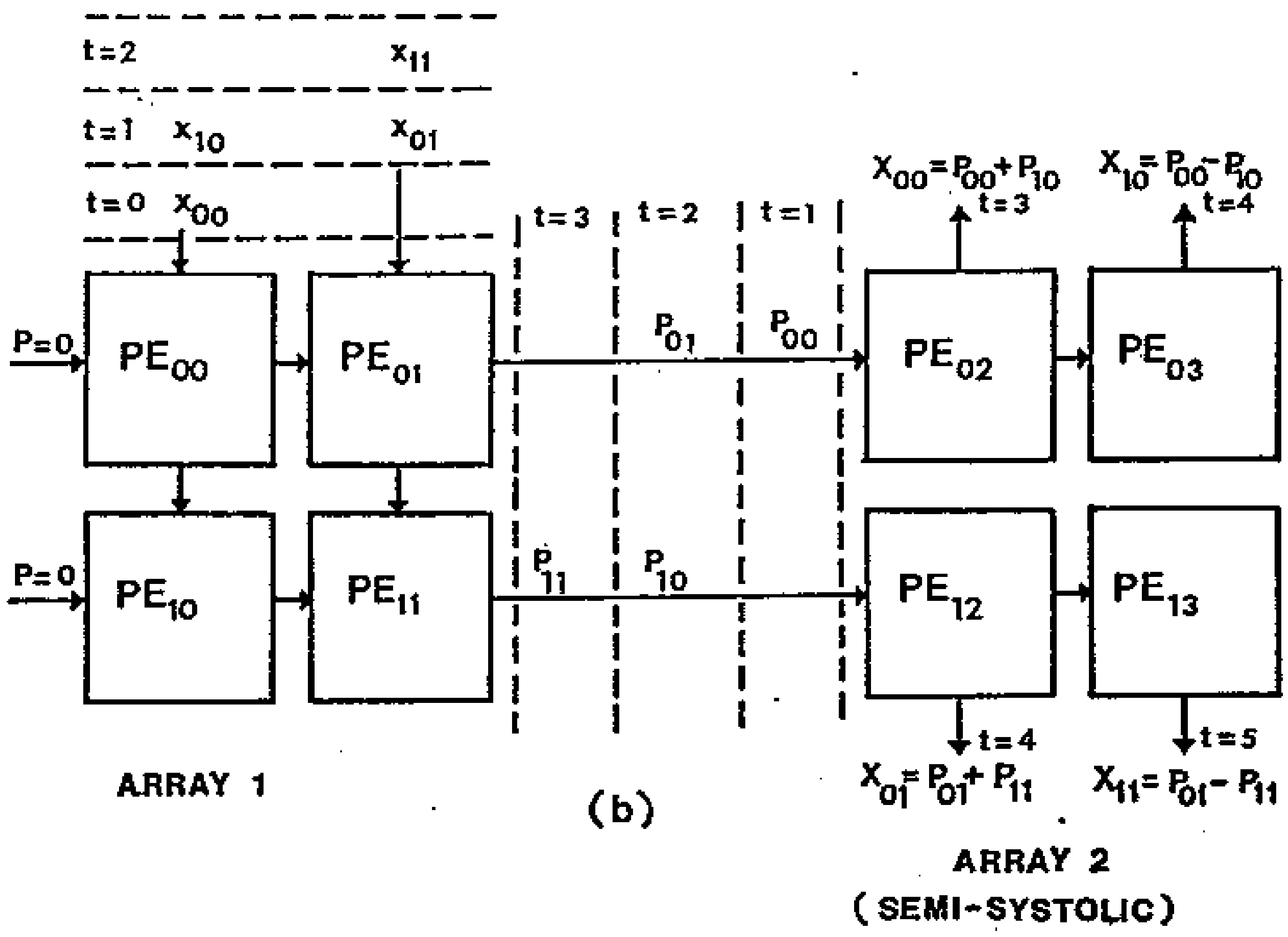
Basically, the processing elements (processing cells) in the arrays of the macropipeline are multiply-add cells. However, at least in one of the arrays the twiddle factors (powers of W^N) depend on time. Note that for the 2-D DFT, one of the variables in the transformed index set (t, s_0, s_1) is the time (in macropipeline clocks), and s_0, s_1 are respectively the row and column number of a processing cell . For example, for the design "a", $t = n_0 + n_1 + n_2$, $s_0 = n_0$, and $s_1 = n_2$.

To calculate the factor $W_N^{n_1 n_2}$ for the first part of the algorithm (10) (i.e for the cells of the first array), one must calculate $W_N^{(t-s_0-s_1)s_1}$. However,with the increase of the local time $t - (s_0+s_1)$ by 1, the previous multiplication factor is to be multiplied by $W_N^{s_1}$, which is a constant value for all cells in column s_1. However, that would require an additional multiplier to generate the needed powers of W_N inside the cells of the first array; another option is to generate the twiddle factors outside the array cells. In the last case, extra pins would be required for each cell of the first array.

For the second array of the design "a", the factor $W_N^{s_0 s_1} = W_N^{n_0 n_2}$ depends on row and column number of the processing cell. Therefore, the twiddle factors can be prestored in the cells of the second array.

Example 3. Fig. 2 shows the time/space performance of a 3-D 2^3-point DFT (N=2,





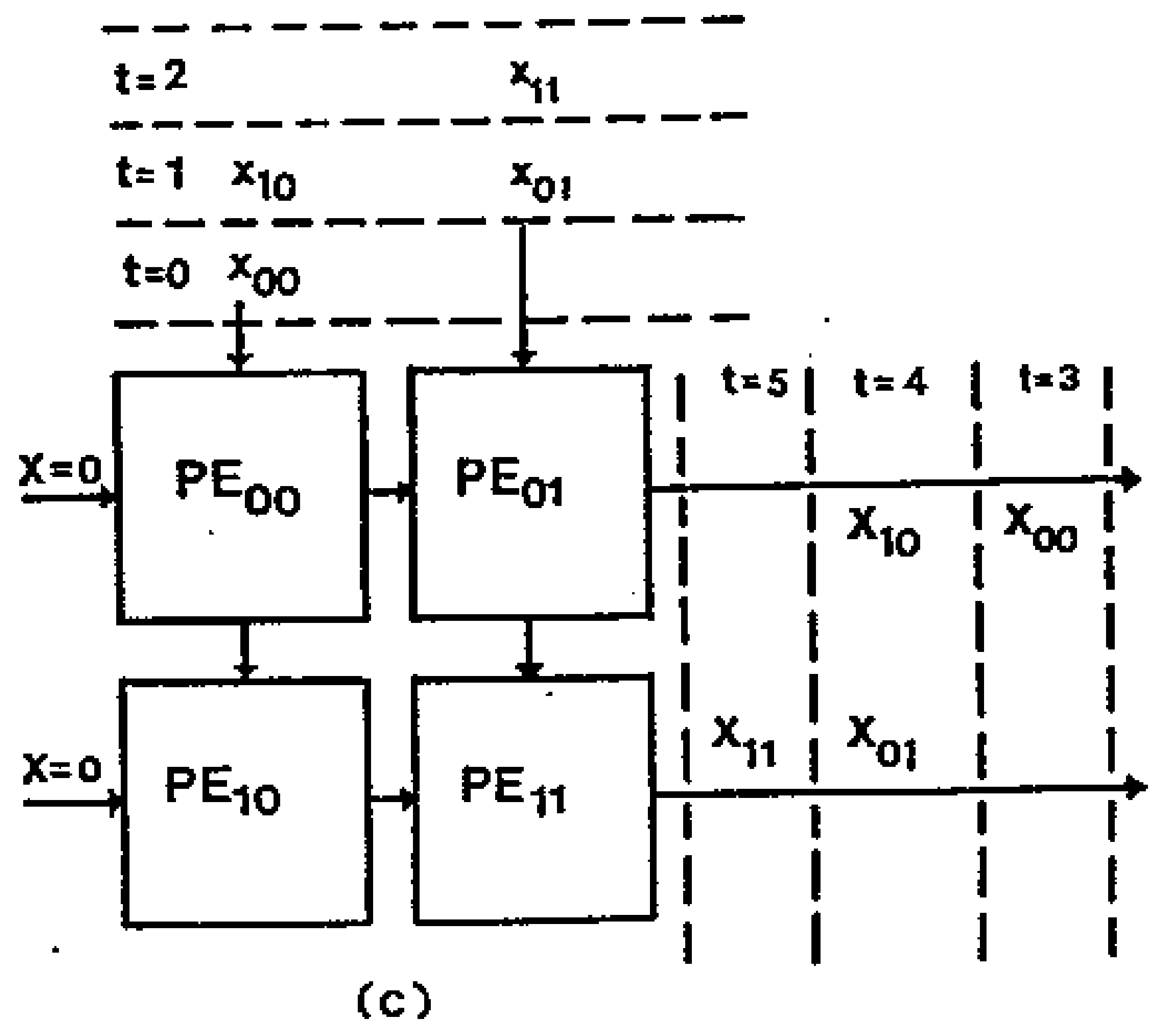


Fig.1. Three designs of systolic/semi-systolic arrays for the implementation of 2-D DFT for N=2
(a) - x is preloaded, P and X move in the arrays
(b) - x and P move, X is accumulated in Array 1;
(c) - x and X move, P is accumulated in the arra

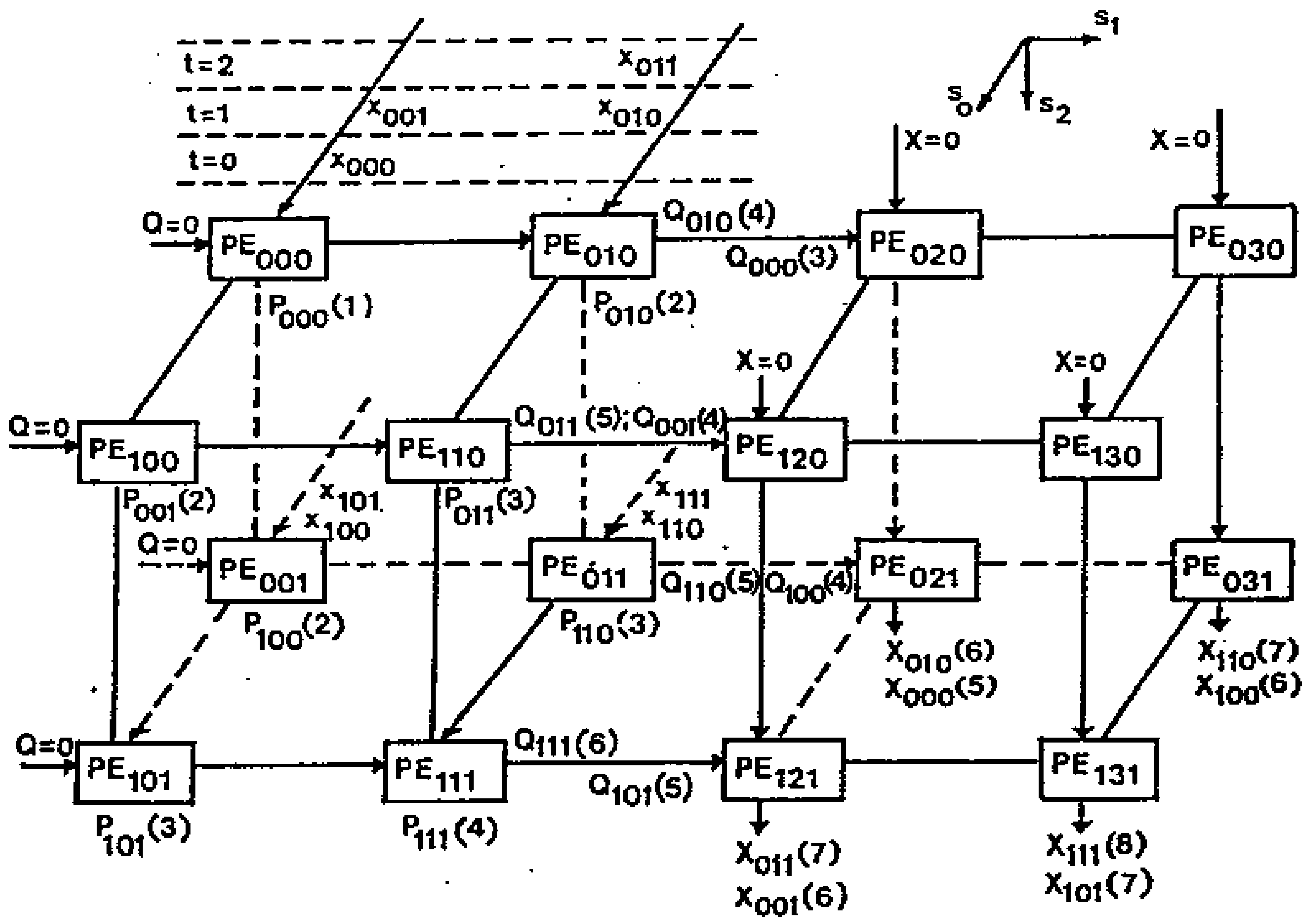


Fig.2. Time-space performance for 2^3 DFT by two cascaded systolic arrays (total 16 PEs); x moves by s_0,P accumulates in the PEs of Array 1, Q moves by s_1, and X moves by s_2. The numbers in the parentheses denote the time of computation in array clocks.

M=3) by $2 \times 2^3 = 16$ processing elements in two 3-D systolic arrays. The design objective is similar to that of Design "c" for the 2-D DFT where the intermediate spectrum values are accumulated in the PEs of the first (semi-systolic) array.

A question of interest is the number of the communication links (for M>2) among the processing cells inside the systolic stages. Since the processing in the systolic array of stage k occurs using two variables only (DFT by k-th index, $0 \leq k \leq M-1$), obviously not all the edges of the M-dimensional cube (with the PEs as vertices) are needed for the processor communication.

3. COMPLEXITY ESTIMATION OF THE SYSTOLIC MACROPIPELINE

For the complexity estimation, we shall use the methods of [23,51,19]. To avoid misunderstanding, we repeat below Thompson's derivation of the lower bound on area x (squared throughput) complexity for one-dimensional FT. (The lower bound on complexity is attained for FFT). One possible layout for k-point FFT is shown in Fig.4 (taken from [23]). The array has k logk multiply-add cells (k/2 rows of logk cells each of the O(log k) area as shown in Fig. 4 for k = 8). A vertical chord (say between two leftmost columns) crosses O(N) wire tracks which means [51,19] that the area complexity is at least of the order $O(k^2)$ (and not of the order $O(k \log^2 k)$ which might be concluded if the wires' complexity is not taken into account). Note that <u>each wire track is counted as one wire</u> (Thompson [23]) though evidently the lines are the multibit ones. The array (FFT-butterfly) of Fig.4 generates one set of k spectrum values each array clock. Assume that the usual multipliers with area O(logk) and calculation time T = O(logk) are used for the cells.
The FFT network of Fig.4 can be considered composed of the cells which are O(logk) units wide and O(1) units tall [23]. The entire network will fit into a rectangular area that is O(k) units wide and O(k) units tall. The width of the logk columns is asymptotically negligible [23]. The FFT network generates one set of k spectrum values every T = O(logk) units of time. Then the AT^2-complexity of the FFT-network is

$$AT^2 = \Omega(k^2 \log^2 k) \qquad (9)$$

which is the well-known lower bound on the AT^2- complexity of FT attained for the FFT. The estimate $O(k^2)$ for the area complexity dominates the estimate $O(\log k) \times (\text{number of cells}) = O(k \log^2 k)$.

Similarly, we can estimate the complexity of the 2-D systolic arrays for $k = N^2$ DFT described in Section 2 . A vertical or horizontal chord crosses N wires (again, we count a multibit line as one wire following [23]). The area complexity due to wires is at least of the order N^2 but the area complexity estimated as the area taken by N^2 multiply-add cells each of O(logN) complexity is $O(N^2 \log N)$; therefore, the entire DFT array needs the area $O(N^2 \log N)$. (Wire connection for the DFT systolic array is simple compared to FFT network). One

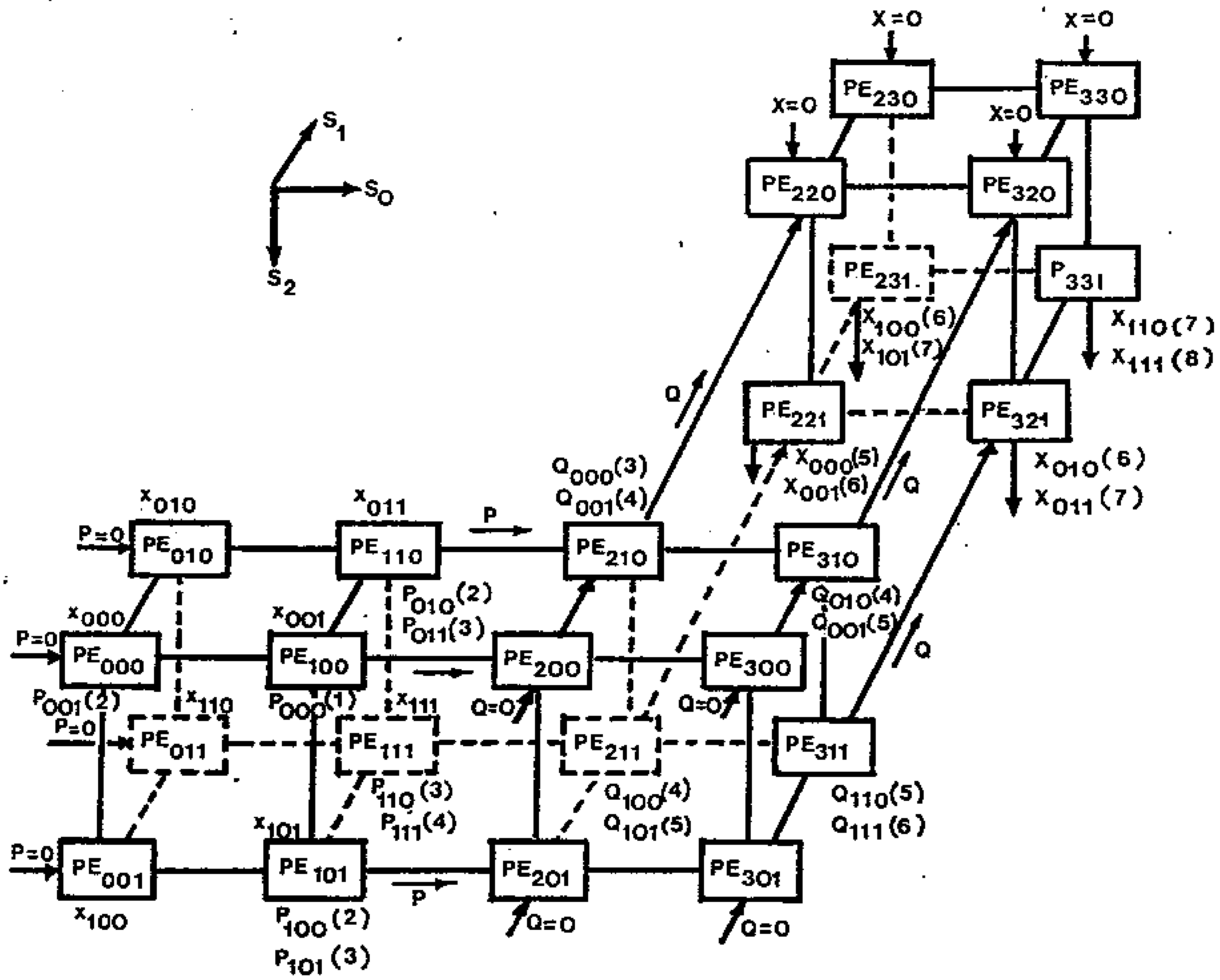


Fig.3. The 3-D 2^3-point DFT by three cascaded systolic arrays; x is preloaded, P moves by s_0, Q moves by s_1, and X moves by s_2. The numbers in the parentheses denote the time of computation in array clocks.

can also consider multiply-add cells as rectangular areas, O(1) units tall and O(logN) units wide; the total array area estimated by the method of [23] gives the same result $O(N^2\log N)$. The systolic arrays presented in Section 2 generate one spectrum set (N^2 values) every N array clocks, therefore T = O(NlogN) and $T^2 = O(N^2\log^2 N)$. Thus the AT^2 - complexity of the proposed systolic macropipeline for N^2 DFT is

$$AT = O(N^4\log^3 N) \qquad (10)$$

which is within a logN factor of the lower bound $\Omega(k^2\log^2 k) = \Omega(N^4\log^2 N)$, for $k = N^2$ (2-D N^2-point FT).

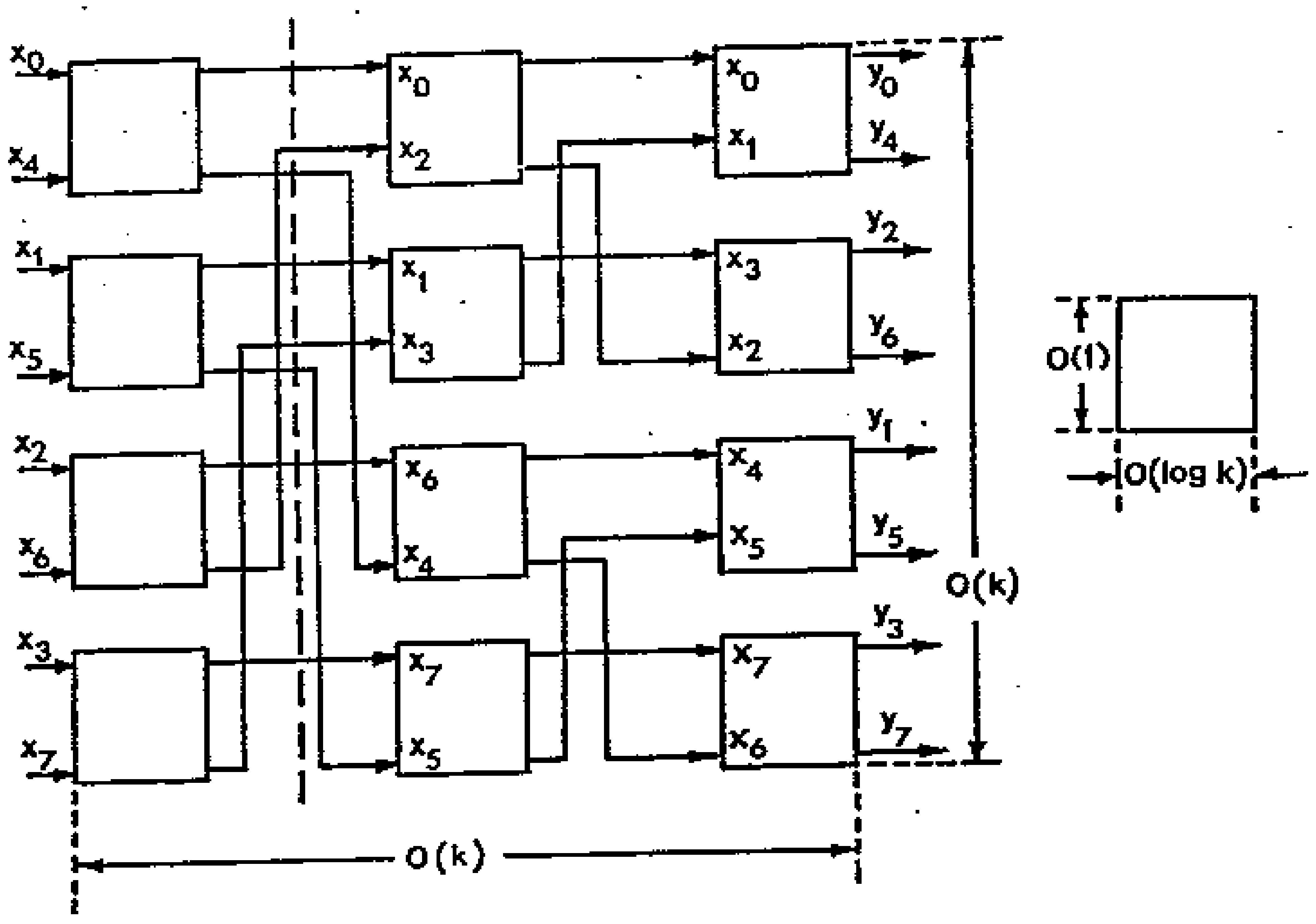


Fig. 4. The FFT network for k = 8 [23] .

For the 3-dimensional systolic macropipeline (for k-point DFT where $k = N^3$), one can consider a planar layout of a 3-D array. It is rather evident that the wire complexity of the cells' connection dominates the complexity of $N^3\log N$ - area taken by N^3 multiply-add cells. The chord 1 shown in Fig. 6 (for the planar layout of a 3-D 64-cell array) crosses 28 wires and chord 2 crosses 16 wires ($O(N^2)$ wires in general) and the chip area must be at least of the order of N^4. A more accurate estimation (again, with the multiply-add cells taken as rectangles O(1) units tall and O(logN) units wide) results in the chip area of the order $O(N^4\log N)$. The throughput T equals O(NlogN) and the AT^2- complexity for N^3-point DFT is

$$AT^2 = O(N^6\log^3 N) = O(k^2\log^3 k) \qquad (11)$$

which is again a factor of logN of the lower bound $\Omega(k^2\log^2 k) = \Omega(N^6\log^2 N)$ for any Fourier transform of the size k, $k = N^3$.

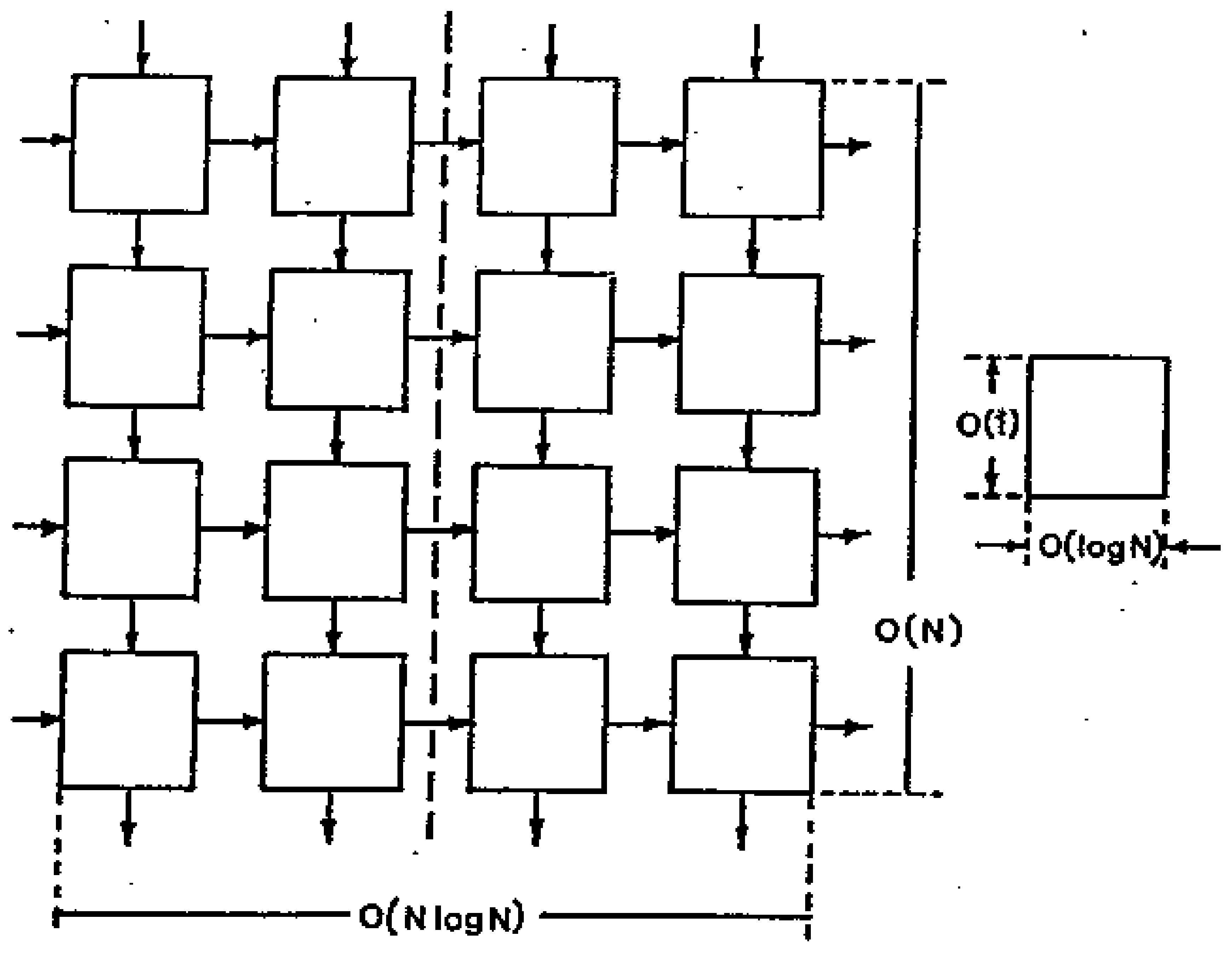


Fig. 5. One of the 2-D systolic arrays of the macropipeline for N^2 DFT; N = 4.

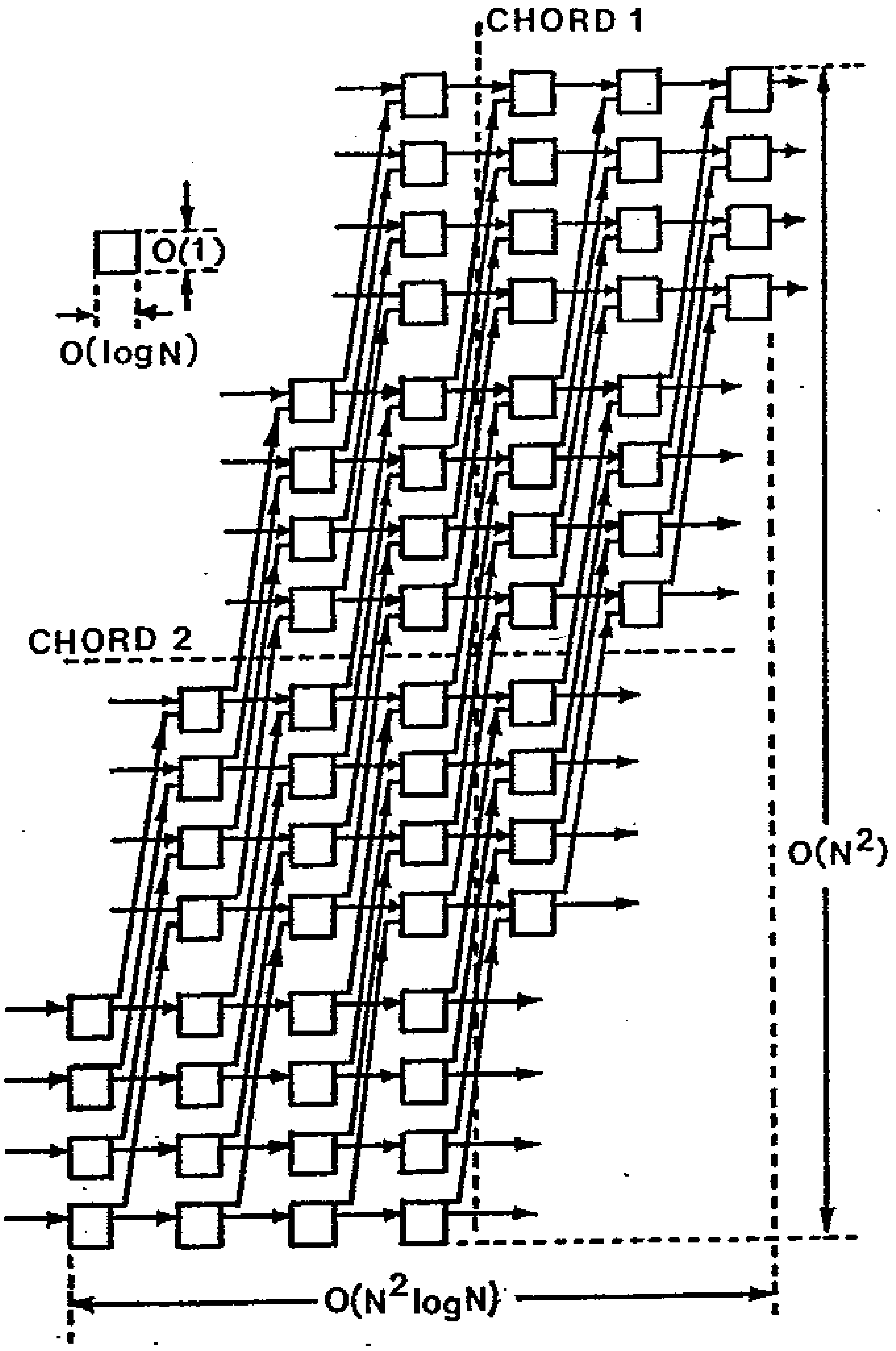


Fig. 6. The planar layout for a 3-dimensional systolic array for N^3 DFT; N = 4.

It can be expected that for the higher number of dimensions, the complexity of the systolic macropipeline with multiply-add cells is of the order $O(N^{2M}\log^3 N)$ where M is the number of dimensions (N^M-point DFT).

In conclusion, return to the area-efficient "stingy" architecture suggested in [53] for the implementation of convolution. The design of [53] (the pipelined version) uses O(k) adders and O(k) multipliers for 1-D k-point convolution; with minor modifications, it can be used to implement 1-D DFT. It seems at first that the design can implement a 2-D DFT in two stages (DFT by rows and DFT by columns of intermediate spectrum), and that the architecture would outperform our 2-D macropipeline in AT^2-complexity. However, between the two stages (each performing 1-D DFT), the matrix of the intermediate spectrum is to be transposed, and the area complexity of the routing network will dominate over the area of the rest of the circuit. A crossbar network or demultiplexing/multiplexing circuit would be needed to feed the intermediate spectrum in the proper order into the second stage. In any case, the complexity of the routing network is at least of the order of N^4 , therefore there is no gain in total area.

4. DISCUSSION

To summarize, the proposed design has the following advantages:

1) The smaller number of pins needed for one processing cell compared to the cell of [25]. For example, for M = 3 (3-D N^3 - DFT) and r-bit data lines, the macropipeline consists of three 3-D arrays. A cell needs no more than six input and output data lines (6r pins for the data). Note that in the case of data preload, only 5r pins are needed for data lines (the control lines are not taken into account).

For the comparison, the cell of the systolic array of [25] requires more than 12r pins for r-bit data lines.

2) Simple processing cells, basically of multiply-add type.

3) Simple connections between the arrays of the macropipeline; no intermediate networks for data reshuffling or routing between the constituent one-dimensional transforms are needed.

4) The time x $(\text{area})^2$ complexity of the macropipeline is within log k factor of the lower bound if the usual multipliers with area O(log k) and time complexity O(log k) are used in the pipeline cells.

5) The input of data is "word-serial" (that is, the macropipeline does not require the parallel input of the complete set of $k = N^M$ data values).

6) At the expense of the increase in area complexity, the throughput of the macropipeline is held constant for any number of dimensions M; for k-point DFT ($k = N^M$), one spectrum set of N^M values is completed every N pipeline clocks; N is the size of one dimension. Compared to the throughput of 6N+1 for the 3-D N x N x N DFT obtained in [25], our design presents an essential gain in speed. (Note that the throughput is to be squared for the complexity evaluation [51]).

7) The large area complexity of the proposed design (Mk cells for k-point M-dimensional DFT) seems to be a deficiency of the proposed method. (The design of [25] requires k cells only but it is about 2M times slower than the proposed design). The linear (as to the number of the dimensions M) loss in area complexity results in the linear gain in speed (throughput) T_p, and since T_p^2 - - the squared value of the throughput - participates in the total complexity estimation, the systolic macropipeline has an advantage for real-time applications where the high speed is the main criteria.

Note that we implement k-point DFT with O(k) multiply-add cells (both for $k = N^2$ and $k = N^3$). For our design, $AT^2 = O(k^2 \log^3 k)$ while the DFT of [54] mentioned in [23] also uses O(k) multiply-add cells for k-point transform but results in complexity $AT^2 = O(k^3 \log^3 k)$, with area A = O(klogk) and T = O(klogk). For 2-D DFT, i.e for $k = N^2$, our design needs same area $A = O(k \log k) = O(N^2 \log N)$ while the time performance (throughput) is $T = O(\sqrt{k} \log k) = O(N \log N)$, i.e. our design is faster compared to the k-point DFT with O(k) cells of [23,54] .

In general, the k-point DFT (for $k = N^M$) can be implemented by a sequence of cascaded M-dimensional cubes of PEs (Mk cells as in Designs "a" and "b" for M=2, or (M-1)k PEs as in Design "c" for M=2, instead of k PEs in the implementation of [25] when the 1-D DFT of the size k is computed using a two- or three-dimensional array. For 2-D systolic design, the macropipeline proposed by us needs twice as large an area (2k PEs) compared to the k-cell array of [25] ; however, for M=2 the throughput of our design is four times higher than that of [25]. It should be kept in mind that in [25] the 1-D DFT is calculated while we consider multidimensional DFT .

It is assumed that the multipliers inside the array cells have area complexity O(logk) = O(logN) and time complexity O(logk) = O(logN) for $k = N^M$-point DFT (like the multipliers typically used in processor design, [54,23]). As a result, the AT^2 complexity of our design is within a logN factor of the lower bound. Theoretically, it is possible to realize a mesh multiplier [52] with area O(logk) and time complexity $O(\sqrt{\log k})$, to multiply two binary numbers of O(logk) bits each; such a multiplier attains the lower bound on multiplier complexity. With the use of minimal complexity multipliers, our design would attain the lower bound on the complexity. Another multiplier with area x $(\text{time})^2$ complexity of the order $O(\log^2 k \log^3(\log k))$ (in our notations) is developed in [54]. However, both multipliers of [52] and [54] are too complicated for the practical use in systolic array cells; in these multipliers, the FFT and the convolution theorem are used.

5. ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewer of the journal for his useful comments and suggestions. The authors are also indebted to Dr. L.B.Levitin for valuable discussions.

REFERENCES

1. H.T.Kung. "Why systolic architectures?" Computer, vol.15, No.1, Jan.1982, pp. 37-46.
2. J.V.McCanny, J.G.McWhirter, and E.E.Swartzlander, Systolic Arrays. Prentice Hall, 1989.
3. H.T.Kung. "Let's design algorithms for VLSI systems," in Proc. Caltech Conf. VLSI, Jan. 1980, pp. 65-90.
4. H.T.Kung and C.E.Leiserson. "Systolic arrays for VLSI," in C.A.Mead and L.A.Conway, Introduction to VLSI systems. Reading, MA: Addison-Wesley, 1980.
5. D.I.Moldovan. "On the analysis and synthesis of VLSI algorithms," IEEE Trans. on Comp.,vol. C-31, 11, Nov. 1982, pp.1121 - 1126.
6. D.I Moldovan. "On the design of algorithms for VLSI systolic arrays," Proc. IEEE, vol. 71, 1, Jan.1983, pp. 113-120.
7. D.I.Moldovan and J.A.B.Fortes. "Partitioning and mapping algorithms into fixed size systolic structures". IEEE Trans. on Comp., vol. C-35,1, Jan. 1986, pp.1-12.
8. D.Moldovan. "Towards a computerized optimal design of VLSI systolic arrays." Design methodologies, Amsterdam, The Netherlands: North Holland, 1986, pp. 215-234.
9. V.Milutinovic, J.A.B.Fortes, L.H.Jamieson. "A multiprocessor architecture for real-time computation of a class of DFT algorithms," IEEE Trans. ASSP, vol. ASSP-34, 5, 1986, pp.1301-1309.
10. H.V.Jagadish, S.K.Rao, and T.Kailath. "Architectures fot iterative algorithms," Proc. of IEEE, vol. 75, 9, Sept. 1987, pp. 1304 - 1321.
11. W.L.Miranker and A.Winkler."Space-time representation of computational structures," Computing, vol.32, 1983, pp. 93 - 114.
12. P.R.Capello and K.Steiglitz. "Simplifying VLSI array design with geometric transformations," in Proc. IEEE Int. Conf. Parallel Processing, 1983, pp. 448 - 457.
13. J.A.B.Fortes, K.S.Fu, and B.W.Wah "Systematic approach to the design of algorithmically specified systolic arrays." Proc.IEEE Int. Conference on ASSP, vol.1, pp.300-303, March 26-29, 1985, Tampa, Florida.
14. C.J.Li and B.W.Wah, "The design of optimal systolic arrays," IEEE Trans. Comp., vol. C-34, 1985, pp. 66-77.
15. I.Gertner and M.Shamash. "VLSI architectures for multidimensional Fourier transform processing," IEEE Trans. Comp., vol.C-36, No.11, Nov. 1987, pp.1265-1274.
16. M.Marchesi, G.Orlandi, and F.Piazza. "A systolic circuit for Fast Hartley Transform," ISKAS'88, pp. 2685-2688, 1988.
17. R.Storn."A novel radix-2 pipeline architecture for the computation of the DFT," ISKAS'88, pp. 2685-2688, 1988.
18. S.Winograd, "On computing the discrete Fourier transform," Math. Comput., vol. 32, pp.175-199, 1978.
19. J.Vuillemin, "A combinatorial limit to the computing power of VLSI circuits," IEEE Trans. Comp.,vol. C-32, 3, 1983, pp.294-300.

20. Z.Wang, "Fast algorithms for the discrete W transform and fast DFT," IEEE Trans. Acoustics,Speech, and Signal Processing, vol. ASSP-32, pp. 803-816, 1984.
21. M.J.Corinthios, "3-D cellular arrays for parallel/cascade image/signal processing," in Spectral Tecniquies and Fault Detection, M.Karpovsky (editor) New York: Academic Press, 1985.
22. C.Moraga, "Design of a multiple-valued systolic system for the computation of Chrestenson spectrum," IEEE Trans. Comp., vol. C-35, pp.183-188, 1986.
23. C.D.Thompson. "Fourier Transforms in VLSI," IEEE Trans. Comp., vol. C-32, 11, pp. 1047-1057, 1983.
24. C.N.Zhang, D.Y.Y.Yun. "Multidimensional systolic network for DFT," Proc. 11th Int.Symp. Computer Architecture, Ann Arbor, Mich., pp.215-222, 1984.
25. N.Ling and M.A.Bayoumi. "An algorithm transformation technique for multidimensional DSP systolic arrays," ISCAS'88, pp. 2275-2278.
26. A.L.Gorin, A.Silberger, L.Auslander. "Computing the two-dimensional DFT on the ASPEN parallel computer architecture," Proc. Int. Conference on Computer Architecture, 1988, pp. 921-923.
27. J.M.Delosme, I.C.F.Ipsen. "Efficient systolic arrays for the solution of Toeplitz sytems: An illustration of a methodology for the construction of systolic architectures in VLSI," Proc. of the 1st Int. Workshop on Systolic Arrays, Oxford, UK, pp. 37-46, 1986.
28. B.W.Wah, M.Aboelaze, W.Shang. "Systematic designs of buffers in Macropipelines of systolic arrays," Journal of Parallel and Distributed Computing, vol.5, 1, pp.1-25, 1988.
29. T.D.Roziner, M.G.Karpovsky, and L.A.Trachtenberg. "Complexity analysis of generalized FFT in a multiprocessor environment," 12th IMACS Congress on Scientific Computation, July 18-22 1988, Paris, France.
30. M.G.Karpovsky, L.A.Trachtenberg, T.D.Roziner. "Computation of discrete Fourier transforms over finite Abelian groups using pipelined and systolic array architectures," International Symposium on the Mathematical Theory of Networks and Systems, June 19-23, 1989, Amsterdam, The Netherlands.
31. T.D.Roziner, M.G.Karpovsky, L.A.Trachtenberg. "Fast Fourier transforms over finite groups by multiprocessor systems," IEEE Trans. ASSP, vol. ASSP-38, 2, February 1990, pp. 226-240.
32. T.D.Roziner. "Systolic macropipelines for multidimensional Fourier transforms," Parallel Architectures, N.Rishe, S.Navathe, and D.Tal (editors), IEEE Computer Society Press, to appear in 1990.
33. K.R.Rao.O.Rath,,K.Yeng. "Recursive generation of the DIF FFT algorithm by 1-D DFT," IEEE Trans. ASSP, vol. ASSP-36, 9, pp.1534-1536, 1987.
34. K.R.Rao, S.Venkatamaran, V.R.Kanchan, and M.Mohanty. "Discrete transforms via the Walsh-Hadamard transform," Signal Processing (The Netherlands), vol. 14, 4, pp.371-382, 1988.
35. K.R.Rao, P.Yip. "The decimation-in-frequency algorithm for a family of discrete sine and cosine transforms," Circuits Syst. Signal Processing, vol. 7, 1, pp.3-19, 1988.

36. O.Rath, K.R.Rao, K.Yeung. "FFT via the Walsh-Hadamard transform," 20th Asilomar Conf. on Signals, Systems, and Computers. Pacific Grove, CA, Nov. 10-12, pp.373-377, 1986.

37. R.C.Singleton. "A method for computing the FFT with auxiliary memory and limited high speed storage," IEEE Trans. Audio and Electroacoustics, vol. AU-15, 3, 1967, pp.91-98.

38. N.M.Brenner. "Fast Fourier Transform of externally stored data," IEEE Trans. Audio and Electroacoustics, vol. AU-17, 3, 1969, pp.128-132.

39. H.L.Buijs. "Fast Fourier Transformation of large arrays of data," Applied Optics, vol.8,1, pp.211-212, 1969.

40. M.A.Onoe. "A method for computing large-scale 2-D transforms without transposing data matrix," Proc. IEEE, vol. 63, 1, pp.196-197, 1975.

41. R.E.Twogood, M.P.Ekstrom. "An extension of Eklund's matrix transposition algorithm and its application in digital image processing," IEEE Trans. Comp., vol.C-25, 9, pp. 950-952,1976.

42. G.L.Anderson. "A stepwise approach to computing the multidimensional Fourier Transform of large arrays," IEEE Trans. ASSP, vol.ASSP-28,3, pp. 280-284, 1970.

43. D.B.Harris, J.H.McClellan, D.Chan, H.S.Schuessler."Vector radix FFT," IEEE Int. Conf. ASSP Proc.,May 1977, pp. 548-551.

44. E.A.Hoyer, W.R.Berry. "An algorithm for the 2-D FFT," IEEE Int. Conf. Proc., May 1977, pp.552-555.

45. J.O'Brien, J.Mather, B.Holland. "A 200 MIPS single-chip 1K FFT processor," Proc. 1989 IEEE Int. Solid-State Circuits Conf., pp.166-167, 1989.

46. S.Magar,S.Shen, G.Luikuo, M.Fleming, R.Agular. "An application specific chipset for 100 MHz data rate," Proc. of ICASSP, pp. 1989-1996, 1988.

47. A.M.Despain. "Fourier transform computers using CORDIC iterations," IEEE Trans. on Comp., vol. C-23, 10, Oct. 1974, pp.993 - 1001.

48. Advanced Micro Devices Inc: Bipolar microprocessor and logic interface (AM 2900 Family) Data Book, Sunnyvale,Calif. 1985.

49. E.A.Trachtenberg. "Designing standard computer component using spectral techniques," Proc. of the 1987 IEEE Int. Conf. on Computer design : VLSI in Computers and Processors (ICCD'87), Rye Brook, NY, 5-8 Oct. 1987.

50. I.Delotto, D.Dotti. "Two-dimensional transform by minicomputers without matrix transposing," Computer graphics and Image Processing, vol.4, 3, pp.271-278, 1970.

51. R.P.Brent, H.T.Kung. "The area-time complexity of binary multipliers," Journal of ACM, vol.28, 3, 1981, pp.521-534.

52. F.P.Preparata."A mesh-connected area-time optimal VLSI multiplier of large integers," IEEE Trans. Comp., vol. C-32, 2, 1983, pp. 194-198.

53. R.M.Owens, M.J.Irwin."Being stingy with multipliers," IEEE Trans. Comp., vol. C-39, 6, 1990, pp.809-818.

54. H.T.Kung and C.E.Leiserson. "Systolic arrays for VLSI," Introduction to VLSI Systems, Reading, MA: Addison-Wesley, 1980, pp. 260-292.