



<b>Title</b>	<b>A novel approach to fast discrete Hartley transform</b>
<b>Author(s)</b>	<b>Liu, JG; Chan, FHY; Lam, FK; Li, HF</b>
<b>Citation</b>	<b>The 4th International Symposium on Parallel Architectures, Algorithms, and Networks, Perth/Fremantle, WA, Australia, 23-25 June 1999, p. 178-183</b>
<b>Issued Date</b>	<b>1999</b>
<b>URL</b>	<b><a href="http://hdl.handle.net/10722/46134">http://hdl.handle.net/10722/46134</a></b>
<b>Rights</b>	<b>Creative Commons: Attribution 3.0 Hong Kong License</b>

# A Novel Approach to Fast Discrete Hartley Transform

J. G. LIU\*, F. H. Y. CHAN\*\*, F. K. LAM\*\*, and H. F. Li\*\*\*

\*Institute for Pattern Recognition & Artificial Intelligence

Huazhong University of Science & Technology

State Education Ministry Laboratory for

Image Processing & Intelligent Control, Wuhan, P. R. China

\*\*Department of Electrical and Electronic Engineering

The University of Hong Kong, Hong Kong

\*\*\*Department of Computer Science

University of Concordia, Montreal, Canada

## ABSTRACT

**Discrete Hartley transform (DHT) is an important tool in digital signal processing. In this present paper, we propose a novel approach to perform DHT. We transform DHT into a form expressed in discrete moments via a modular mapping and truncating Taylor series expansion and present a completely new formula for computing DHT. We extend the use of our systolic array for fast computation of moments without any multiplications to one that computes DHT with only a few multiplications and without any evaluations of triangular functions. The multiplication number used in our method is  $O(N \log_2 N / \log_2 \log_2 N)$  superior to  $O(N \log_2 N)$  in the conventional FDT. The execution time of the systolic array is only  $O(N \log_2 N / \log_2 \log_2 N)$  for 1-D DHT and  $O(N^k)$  for k-D DHT ( $k \geq 2$ ). The systolic array consists of very simple processing elements and hence it implies an easy and potential hardware/VLSI implementation. The approach is also applicable to DHT inverses.**

## 1: Introduction

Discrete Hartley transform (DHT) is widely used in signal processing [1-3]. DHT maps a real-valued sequence to a real-valued spectrum while preserving some of the useful properties of the DFT and is a powerful tool as a substitute to the FFT for computing cyclic convolutions of real data [4-6]. There have been many papers reporting the development with the fast DHT [7-9].

On the other hand, computer vision and image analysis have propelled the advancement of fast computation of discrete moments (DM) [10-13]. These two research threads have been developed independently. Up to now, no fast algorithms proposed for DHT have some connections with moments.

In this paper, we first construct the bridge between DHT and discrete moments (DM) by a modular mapping and making use of Taylor expansions and hence we can transform DHT into computation involving moments. We also discuss the problems of convergence and errors. Based on our approach to the fast calculation of moments [10], a new systolic array to perform 1-D DHT is presented, followed by a complexity analysis. Finally, we give our conclusions.

## 2: Using moments to computing

The discrete Hartley transform (DHT) is defined for a real valued length-N sequence  $x(n)$  ( $0 \leq n \leq N-1$ ), by the following equations:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) \text{cas}(2\pi nk / N) \\ &= \sum_{n=0}^{N-1} x(n) (\cos(2\pi nk / N) + \sin(2\pi nk / N)) \end{aligned} \quad 0 \leq k \leq N-1 \quad (2.1)$$

Let us transform Equation (2.1) to look for a new approach to computing it. For every pair of  $k$  and  $i$  ( $i, k = 0, 1, 2, \dots, N-1$ ), and defining  $S(k, i)$  by

$S(k, i) = \{n \mid kn \equiv i \pmod{N}, 0 \leq n \leq N-1\}$   
then defining  $x_{k,i}$  ( $i, k = 0, 1, 2, \dots, N-1$ ) by

$$x_{k,i} = \begin{cases} \sum_{j \in S(k,i)} x(j) & S(k,i) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

i, k=0, 1, 2, ..., N-1  
 For example, for N=4, For example,  
 $S_{00}=\{0, 1, 2, 3\}$ ,  $S_{01}=S_{02}=S_{03}=\Phi$ ;  
 $S_{10}=\{0\}$ ,  $S_{11}=\{1\}$ ,  $S_{12}=\{2\}$ ,  $S_{13}=\{3\}$ ;  
 $S_{20}=\{0, 2\}$ ,  $S_{21}=\Phi$ ,  $S_{22}=\{1, 3\}$ ,  $S_{23}=\Phi$ ;  
 $S_{30}=\{0\}$ ,  $S_{31}=\{3\}$ ,  $S_{32}=\{2\}$ ,  $S_{33}=\{1\}$ .  
 $x_{00}=x(0)+x(1)+x(2)+x(3)$ ,  $x_{01}=0$ ,  $x_{02}=0$ ,  $x_{03}=0$ ;  
 $x_{10}=x(0)$ ,  $x_{11}=x(1)$ ,  $x_{12}=x(2)$ ,  $x_{13}=x(3)$ ;  
 $x_{20}=x(0)+x(2)$ ,  $x_{21}=0$ ,  $x_{22}=x(1)+x(3)$ ,  $x_{23}=0$ ;  
 $x_{30}=x(0)$ ,  $x_{31}=x(3)$ ,  $x_{32}=x(2)$ ,  $x_{33}=x(1)$ .

Thus, by using the periodic properties of sine functions and cosine functions, Equation (2.1) can be rewritten as follows:

$$X(k) = \sum_{i=0}^{N-1} x_{k,i} (\cos(2\pi i/N) + \sin(2\pi i/N)) \quad 0 \leq k \leq N-1 \quad (2.3)$$

For  $0 \leq i \leq N-1$

$$\begin{aligned} \cos(2\pi i/N) &= 1 - (2\pi i/N)^2/2! + \dots + (-1)^p (2\pi i/N)^{2p}/(2p)! + R_{cos} \\ R_{cos} &= \cos \xi_i + (2p+1)\pi/2 (2\pi i/N)^{2p+1}/(2p+1)! \quad 0 < \xi_i < 2\pi i/N \\ \sin(2\pi i/N) &= (2\pi i/N)/1! - (2\pi i/N)^3/3! + \dots + (-1)^p (2\pi i/N)^{2p+1}/(2p+1)! \\ &\quad + R_{sin} \\ R_{sin} &= \sin \xi_i + (p+1)\pi (2\pi i/N)^{2p+2}/(2p+2)! \quad 0 < \xi_i < 2\pi i/N \end{aligned} \quad (2.4)$$

This follows immediately by applying the theorem of extended law of the mean [14] to  $\cos(2\pi i/N)$  and  $\sin(2\pi i/N)$ , where  $R_i$  is Taylor remainder term. Substituting them into Equation (2.3), yields

$$\begin{aligned} X(k) &= x_{k,0} + \sum_{i=1}^{N-1} x_{k,i} \sum_{r=0}^p [(-1)^r (2\pi i/N)^{2r}/(2r)! + \\ &\quad (-1)^r (2\pi i/N)^{2r+1}/(2r+1)!] + R_p \\ &= x_{k,0} + \sum_{r=0}^p (-1)^r (2\pi)^{2r}/N^{2r} (2r)! \sum_{i=1}^{N-1} x_{k,i} i^{2r} + \\ &\quad \sum_{r=0}^p (-1)^r (2\pi)^{2r+1}/N^{2r+1} (2r+1)! \sum_{i=1}^{N-1} x_{k,i} i^{2r+1} + R_p \\ &= x_{k,0} + \sum_{r=0}^{2p+1} a_r m_{k,r} + R_p \\ &\quad 0 \leq k \leq N-1 \end{aligned} \quad (2.5)$$

Where

$$a_r = \begin{cases} (-1)^q (2\pi)^{2q}/N^{2q} (2q)! & r=2q \\ (-1)^q (2\pi)^{2q+1}/N^{2q+1} (2q+1)! & r=2q+1 \end{cases} \quad 0 \leq q \leq p \quad (2.6)$$

$$m_{k,r} = \sum_{i=1}^{N-1} x_{k,i} i^r \quad (2.7)$$

$$\begin{aligned} R_p &= \sum_{i=1}^{N-1} x_{k,i} [\cos \xi_{cos,i} + (2p+1)\pi/2 (2\pi i/N)^{2p+1}/(2p+1)! + \\ &\quad \sin \xi_{sin,i} + (p+1)\pi (2\pi i/N)^{2p+2}/(2p+2)!] \\ &\quad 0 < \xi_{cos,i}, \xi_{sin,i} < 2\pi i/N \end{aligned} \quad (2.8)$$

If  $R_p$  is ignored, we have

$$X(k) = x_{k,0} + \sum_{r=0}^{2p+1} a_r m_{k,r} \quad 0 \leq k \leq N-1 \quad (2.9)$$

The absolute value of the error introduced by overlooking  $R_p$  is bounded by

$$\begin{aligned} |R_p| &\leq \left| \sum_{i=1}^{N-1} x_{k,i} \cos(\xi_{cos,i} + (2p+1)\pi/2) (\pi i/2N)^{2p+1}/(2p+1)! \right| \\ &\quad + \left| \sum_{i=1}^{N-1} x_{k,i} \sin(\xi_{sin,i} + (p+1)\pi) (2\pi i/N)^{2p+2}/(2p+2)! \right| \\ &\leq \max_n |x(n)| (N-1) [(2\pi)^{2p+1}/(2p+1)! + (2\pi)^{2p+2}/(2p+2)!] \end{aligned} \quad (2.10)$$

From the inequality above, it is clearly shown that  $R_p$  converges to zero very rapidly and uniformly. For example, for  $\max |x(n)| \leq 256$ ,  $(0 \leq n \leq N-1)$ ,  $N \leq 2048$ ,  $p=17$ ,

$$|R_p| \leq 5.13 \times 10^{-7}$$

This error can satisfy accuracy requirements of most applications by computing only thirty-six terms.

Furthermore, we can prove that the least upper bound of  $p$  is not more than  $O(\log_2 N / \log_2 \log_2 N)$  as  $N$  tends to infinite [15, 16].

Let

$$f(N, p) = \max_r |x(r)| \sqrt{2} (N-1) (2\pi)^p / p! = A(N-1) (2\pi)^p / p! \quad (2.11)$$

Substituting  $[2 \log_2 N / \log_2 \log_2 N]$  for  $p$  in Equation (2.11) ( $[x]$  denotes an integer closest to  $x$ ), we get

$$f(N, p) = F(N) = A(N-1) (2\pi)^{[2 \log_2 N / \log_2 \log_2 N]} / [2 \log_2 N / \log_2 \log_2 N]!$$

Let

$$2 \log_2 N / \log_2 \log_2 N = t,$$

then

$$\log_2 N = t (\log_2 \log_2 N) / 2,$$

and

$$F(N) \leq A N (2\pi)^{t+1} / [t]!$$

so

$$N = 2^{(t \log_2 \log_2 N) / 2} = (\log_2 N)^{t/2}$$

and

$$\lim_{N \rightarrow \infty} t = \infty$$

$$\begin{aligned} \lim_{N \rightarrow \infty} (\log_2 N)^{t/2} / t &= \lim_{N \rightarrow \infty} (\log_2 N)^{1/2} \log_2 \log_2 N / 2 \log_2 N \\ &= \lim_{N \rightarrow \infty} (\log_2 \log_2 N) / 2 (\log_2 N)^{1/2} \\ &= \lim_{M \rightarrow \infty} (\log_2 M) / 2 M^{1/2} = 0 \end{aligned}$$

then

$$\begin{aligned} \lim_{N \rightarrow \infty} F(N) &\leq \lim_{t \rightarrow \infty} A (2\pi)^{t+1} (\log_2 N)^{t/2} / [t]! \\ &= \lim_{t \rightarrow \infty} A (2\pi)^{t+1} (\log_2 N)^{t/2} / t! e^{-t} \sqrt{2\pi} \\ &= \lim_{t \rightarrow \infty} A (2\pi) (2\pi e (\log_2 N)^{1/2} / t)^t / \sqrt{2\pi} = 0 \end{aligned}$$

The final result is obtained by using the well known Stirling formula [17]. The computation of  $X(k)$  using the approximation of Equation (2.9) involves the generation of the  $r$ -th order moments of the transformed data sequence  $x_{k,i}$  and then performing a dot product of these moments with a constant vector ( $a_r$ ) and an addition with  $x_{k,0}$ . Thus if the moments can be computed very quickly and efficiently, so can be the above procedure.

### 3: An algorithm for computing 1-D DHT

According to Section 2, fast DHT includes the following three steps:

1. Computing  $x_{k,i}$  ( $i=0, 1, 2, \dots, N-1$ ;  $k=0, 1, 2, \dots, N-1$ ) and  $a_r$  ( $r=0, 1, 2, \dots, 2p+1$ );
2. Computing  $m_{k,r}$  ( $r=0, 1, 2, \dots, 2p+1$ ;  $k=0, 1, 2, \dots, N-1$ );
3. Computing  $X(k)$  ( $k=0, 1, 2, \dots, N-1$ ).

The procedure of computing  $x_{k,i}$  and  $a_r$  is also called the Preprocessing. The algorithm of computing  $X(k)$  can be described below.

```

Input initial p, N, x(0), x(1), ..., x(N-1)
perform Preprocessing
for k=0 to N-1
  compute Moment2p+1( $x_{k,N-1}, x_{k,N-2}, \dots, x_{k,1}$ )
  X(k)=0
  for r=0 to 2p+1
    X(k)=X(k)+ $a_r m_{k,r}$ 
  end for
  X(k)=X(k)+  $x_{k,0}$ 
end for

```

Moment<sub>2p+1</sub>( $x_{k,N-1}, x_{k,N-2}, \dots, x_{k,1}$ ) is a subroutine that produces the moments up to  $2p+1$  order [10]. Now the complexity of the algorithm can be given. In the Preprocessing, computing  $a_r$  and index set  $S(k, i)$  can be completed upon condition that  $N$  and  $p$  are given. In real-time systems, these computations can be preprocessed and are not included in execution procedure. The computation of  $x_{k,i}$  is  $N^2 - \sum_{k=0}^{N-1} N / \gcd(k, N)$  additions (greatest common divisor). Computing Moment<sub>2p+1</sub>  $N$  times needs  $(2p+2)(2p+3)(N-2)N/2$  additions. The computation of  $X(k)$  ( $k=0, 1, 2, \dots, N-1$ ) involves  $(2p+2)N$  additions and  $(2p+1)N$  multiplications. So there are altogether  $N^2 - \sum_{k=0}^{N-1} N / \gcd(k, N) + (2p+2)(2p+3)(N-2)N/2 + (2p+2)N$  additions and  $(2p+1)N$  multiplications in the algorithms. If  $N$  is a prime number,  $(N^2 - \sum_{k=0}^{N-1} N / \gcd(k, N))$  disappears from the above formula.

#### 4: Pre-processing array for computing $X_{k,i}$

In the general case, we propose to use a special linear array to implement  $x_{k,i}$  ( $i, k=0, 1, \dots, N-1$ ). Each element  $x(r)$  is tagged with a rank= $kr \pmod N$  before it is sent into the array. For example, for  $k=2$  and  $N=4$ , the samples become  $(x(0), 0), (x(1), 2), (x(2), 0), (x(3), 2)$ . These are sent into the linear array one element at a time. An

element is percolated from left to right until it reaches a cell in the array whose position is identical to its rank. When this happens, the value is accumulated in that cell in case it receives multiple values (i.e.  $|S_{ki}| > 1$ ). Figure 1 illustrates an example. A cell in a linear array contains an adder only if the corresponding partition  $S_{ki}$  has a size greater than 1. In 8 clocks,  $x_{2,0}$  will emerge at the right hand side, as shown in Figure 1. In general, it takes  $2N$  clocks for the vector to appear on the right. It is obvious that number of additions required in the preprocessing array for each  $k$  is equal to  $N - N / \gcd(k, N)$ . So the total number of addition required in the preprocessing array is equal to  $\sum_{k=0}^{N-1} (N - N / \gcd(k, N)) = N^2 - \sum_{k=0}^{N-1} N / \gcd(k, N)$ .

$x(r)$  to  $x_{k,i}$  is noteworthy: If  $(k, N)=1$ , (i.e.,  $k$  and  $N$  are coprime) then  $\{ki \mid i=0, 1, 2, \dots, N-1\}$  is a complete system of residues modulo  $N$  [18] and  $|S_{ki}|=1$ . It means  $x_{k,i}$  is a permutation of  $x(r)$ . It remains to consider the generation of  $x_{k,i}$ . One interesting property of the mapping of  $x(r)$ . Particularly, if  $N$  is prime, then  $S_{00}=\{0, 1, \dots, N-1\}$  and  $S_{0j}=\emptyset$  for  $j \neq 0$ , but  $|S_{ki}|=1$  for all  $k \neq 0$ . In other words,  $x_{k,i}$  is a permutation of  $x(r)$  whenever  $k \neq 0$ . In the remaining case,  $y_0(0) = \sum_{r=0}^{N-1} x(r)$  and  $x_{0,j}=0$  for  $j \neq 0$ . So in the case where the sample size is a prime number, a preprocessor can be used to produce the  $x_{k,i}$  via permutations of the sampled sequence  $x(r)$ , after producing  $y_0(0) = \sum_{r=0}^{N-1} x(r)$  in the first pass.

#### 5: Systolic array for computing DHT

The general network for implementing DHT is drawn in Figure 2. It consists of the moment generator [10], the processing arrays and a dot product array. The moment generator formed of  $N-2$   $(2p+1)$ -networks with a row of adder-latch could be used to generate the 1-D moments. It receives  $x_{k,i}$  ( $i, k=0, 1, \dots, N-1$ ) that preprocessing arrays produce. The dot product array performs the dot product and produces  $X(k)$  at the only output. The numbers in square brackets denote the amount of time delay to keep synchronous pace in Figure 2. The scheduling of the dataflow is such that  $m_k(0) * a_0$  moves from left to right, accumulates with  $m_k(1) * a_1$  to produce the partial sum and this repeats until  $X(k)$  emerges at the far right of the output. As  $m_k(i)$  is produced earlier one cycle than  $m_k(i+1)$  ( $i=0, 1, \dots, 2p+1$ ), as shown in Figure 2, so the most right

term  $m_k(p)$  takes 2 cycles to merge into  $X(k)$ . It is just time cost for the dot product.

The total execution time of the computing procedure is:

$$N + (2p+2)(N-2) + 2 + N - 1 = 2(p+2)N - (4p+3)$$
 clock cycles. Where  $N$  is for preprocessing, and  $(2p+2)(N-2)$  for computing moments, and 2 for the dot product, and  $N-1$  for collecting all  $X(k)$ , ( $k=1, 2, \dots, N-1$ ).

Since the DHT is self-inverse, the approach is also applicable to DHT inverse. The method can also be extended to multi-dimensional DHT due to the separability of the DHT kernel. It can also be proved by mathematical induction that the execution time of the systolic array is  $O(N^k)$  for  $k$ -D DHT.

## 6: Conclusions

New fast algorithms and systolic arrays for one-dimensional DHT and their inverse are proposed and the results are extended to  $k$ -dimensional DHT ( $k \geq 2$ ), and their computation complexities are analyzed. The relationships between these discrete transforms and discrete moments are presented and the fast discrete transforms are converted into fast discrete moment transforms. It is a completely new approach to DHT.

The methods proposed have the advantages below:

1. Many multiplications have been converted into additions, and triangular functions have been replaced by simple polynomial functions. It decreases the computational cost and memory requirement. Comparing with the direct method with computation complexity  $O(N^2)$  and the conventional FHT which originates from FFT with  $O(N \log_2 N)$ , the methods proposed are superior to the direct method and but inferior to FHT) if the calculations are executed on a sequential machine (for the example given in Section 2:  $N \leq 2k, \log_2 N \leq 11, p=17$ ). When  $N$  is large enough,  $p$  can take a value much less than  $\log_2 N$  and can still satisfy the accuracy requirement. In other words, in the case of  $N$  large enough, the number of multiplication in our method is less than that in the conventional FHT though the number of addition in our method is still more than that in the conventional FHT;
2. The systolic arrays consist of only latches, adder-latches and a few multipliers. It results in easy hardware implementation and is very suitable for a real-time system;

3. It can cope with arbitrary data length and not limited to data length in the form of  $2^k$  or any other forms. It has a very good accuracy and convergence feature;

4. The multiplication number for 1-D DHT in our method is  $O(N \log_2 N / \log_2 \log_2 N)$  superior to  $O(N \log_2 N)$  in the conventional FHT;

5. The methods are very easily extended to multi-dimensional DHT and their inverses. The computational complexity is  $O(n^k)$  for  $k$ -dimensional DHT in the systolic arrays.

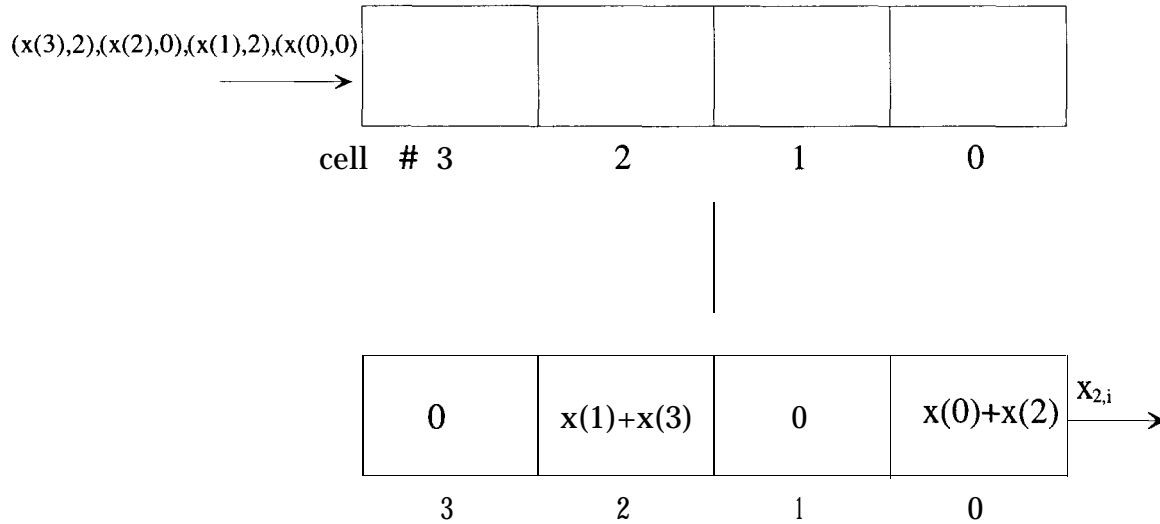
## Acknowledgement

This project has been supported by National Natural Science Foundation of China under grant 69775011 and by University of Hong Kong research grants.

## References

- [1] R. N. Bracewell, *The Hartley Transform*, New York: Oxford University Press, 1986.
- [2] S.-C. Pei and I.-I. Yang, "Computing pseudo-Wigner distribution by the fast Hartley transform," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2346-2349, September, 1992.
- [3] N.-C. Hu, H.-I. Chang, and O. K. Ersoy, "Generalized discrete Hartley transforms," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2931-2940, December, 1992.
- [4] R. N. Bracewell, "The fast Hartley transform," *Proc. IEEE*, vol. 72, pp. 1010-1018, August, 1984.
- [5] P. Duhamel and M. Vetterli, "Improved Fourier and Hartley transform algorithms: application to cyclic convolution of real data," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 642-644, 1986.
- [6] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete Hartley transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 1231-1238, 1985.
- [7] H. S. Hou, "The fast Hartley transform algorithm," *IEEE Trans. Comput.*, vol. 36, pp. 147-156, February, 1987.
- [8] O. Buneman, "In-situ bit-reversed ordering for Hartley transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 577-580, 1989.

- [9] O. Buneman, "Multidimensional Hartley transforms," *Proc. IEEE*, vol. 75, pp. 267, February, 1987.
- [10] Chan, F. H. Y., Lam, F. K., Li, H. F., and Liu, J. G. "An all adder systolic structure for fast computation of moments," *Journal of VLSI Signal Process.* Vol. 54, pp. 159-175, 1996.
- [11] Hatamian, M. "A real time two-dimensional moment generating algorithm and its single chip implementation," *IEEE Trans. Acoust. Speech Signal Process.* vol. 34, pp. 546-553, 1986.
- [12] Li, B.-C., and Shen, J. "Pascal triangle transform approach to the calculation of 3-D moments," *Computer Vision, Graphics, and Image Processing: Graphical Model and Image Processing*, vol. 54, pp. 301-307, 1992.
- [13] Zakaria, M. F., Vroomen, L. J., Zsombar-Murray P. J. A., and Van Kessel, J. M. H. M. "Fast algorithm for computation of moment invariants," *Pattern Recognition*, vol. 20, pp. 634-643, 1987.
- [14] Olmsted, J. M. H. *Real Variables*, Appleton-Century-Crofts, Inc. 1956.
- [15] J. G. Liu, H. F. Li, F. H. Y. Chan and F. K. Lam, "Fast discrete cosine transform via computation of moments," *Journal of VLSI Signal Processing* Vol. 19, No.3, 257-268, 1998.
- [16] J. G. Liu, H. F. Li, F. H. Y. Chan and F. K. Lam, "A novel approach to fast discrete Fourier transform," *Journal of Parallel and Distributed Computing*, vol. 54, pp. 48-58, 1998.
- [17] Beckenbach, E. F. and Bellman, B. *Inequalities*. Springer-Verlag, 1961.
- [18] Rose, H. E. *A Course in Number Theory*, Oxford University Press Inc., 1994.



**Figure 1. The linear array to implement  $x_{2,i}$ .**

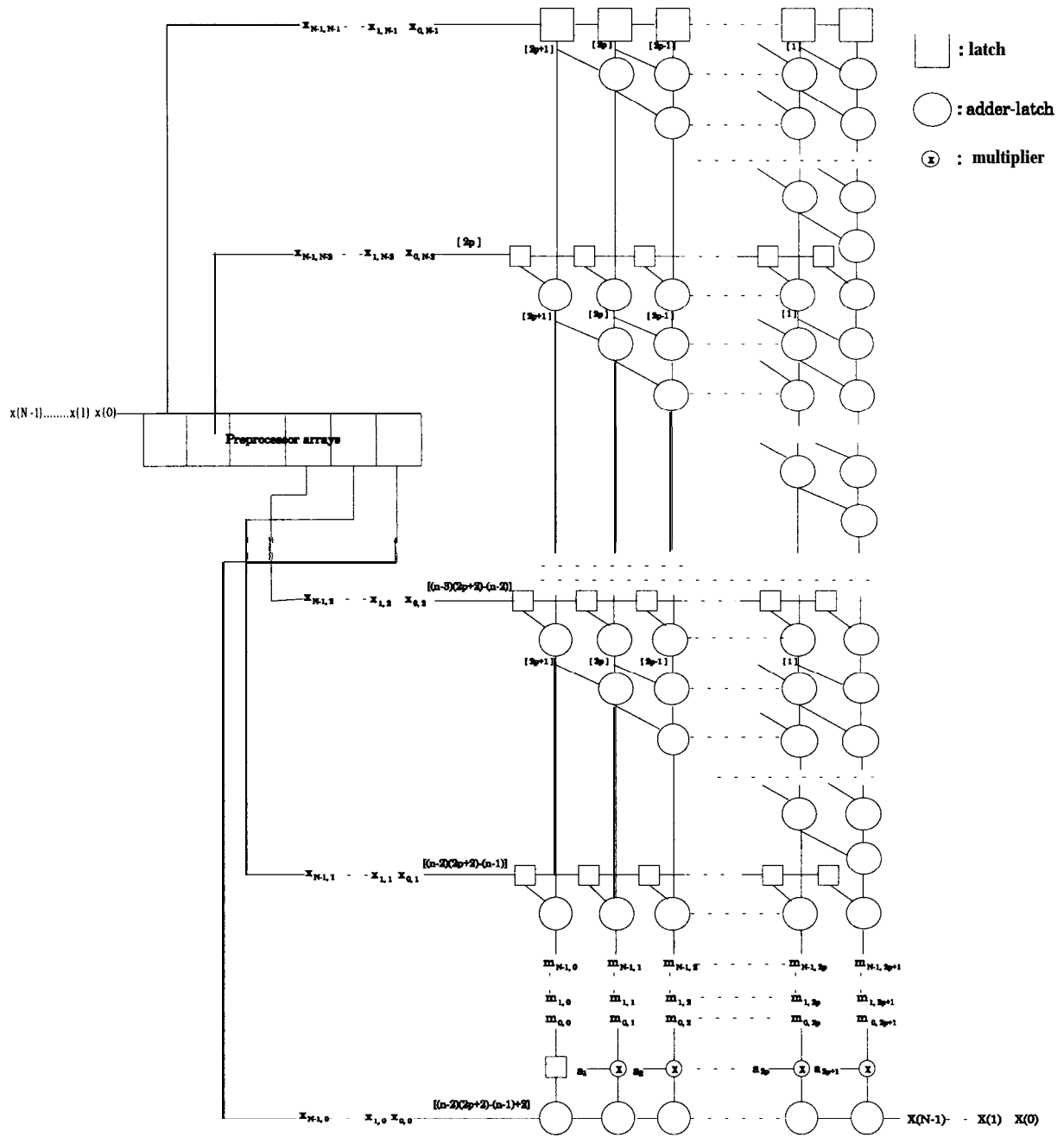


Figure 2. The systolic array for 1-D DHT.