International Journal of Hybrid Information Technology
Vol.8, No.7 (2015), pp.149-160
http://dx.doi.org/10.14257/ijhit.2015.8.7.13

High-Precision Correlation Algorithm Based On Moments

Zhenbing Liu and Qijia He

Guangxi Key Laboratory of Automatic Detecting Technology and Instruments, Guilin University of Electronic Technology, Guilin, China
zbliu@guet.edu.cn

Abstract

Correlation is an important and useful operation in the fields of digital signal processing. In this paper, based on the previous work of performing discrete Fourier transform (DFT) via linear sums of discrete moments, we have made development to eliminate multiplications in the DFTs by performing appropriate bit operations and shift operations in binary system, which can be implemented by integer additions of fixed points; then using the Correlation theorem with the DFT, we compute the Correlation with two DFTs, a point-by-point product, and an inverse DFT. Since our algorithm involves fewer multiplications, an efficient and regular systolic array is designed to implement it which is a demonstration of the locality of dataflow in the algorithms. The approach is also applicable to multi-dimensional DFT .

Keywords: *multiplierless discrete Fourier transforms, moments, Correlation theorem, systolic array*

1. Introduction

Calculation of finite digital Correlation is frequently encountered in digital signal processing applications [1-3]. A variety of algorithms for computing Correlation have been proposed, for example, the Cook-Toom algorithm and Winograd Short Convolution Algorithm [1,3,5,10], which can be used to compute correlation. On the other hand, many researchers focus on realizing convolution with efficient VLSI systems. Amongst the existing VLSI systems, systolic architectures have been extensively popular owing not only to the simplicity of their design and development; but also for the potential of using high level of pipelining in a small chip-area. Several different systolic architectures are, therefore, suggested for VLSI implementation of digital Correlation [10, 16].

Meanwhile, computer vision and image analysis have propelled the advancement of fast computation of discrete moments (DM) [14, 4, 11, 24]. Liu have constructed the bridge between DFT and discrete moments (DM) by modular mapping and making use of Taylor expansions and hence can transform DFT into computation involving moments [4, 20]. This method is more efficient and flexible, because we can get only a portion of the frequency coefficients without computing all N frequency values of the DFT.

In this paper, we propose an algorithm to compute correlation using Correlation theorem based on a novel DFT. First, the DFT algorithm without multiplications is introduced. Then, we can complete Correlation by computing DFT using Correlation theorem Based on the approach to the fast calculation of moments [4], new systolic arrays to perform 1-D DFT and Correlation are presented, followed by a complexity analysis.

The rest of the paper is organized as follows. First we introduce an improved algorithm for DFT in section 2, and design the systolic arrays and analyze the complexity of our method in section 3.Then we give the systolic arrays designed to compute Correlation in section 4. Finally, we include our paper in section 5.

ISSN: 1738-9968 IJHIT

2. DFT Without Multiplications

The DFT of a length-N sampled sequence $x(0), x(1), \text{L}, x(N-1)$ is defined by:

$$X(k) = \sum_{r=0}^{N-1} x(r) e^{-j2\pi rk/N} \qquad k = 0,1,2,\text{L}, N-1 \tag{1}$$

We give a brief introduction of the previous results below[20].

The first step partitions the set $\{0,1,2,\text{L}, N-1\}$ into N disjoint subsets, depending on k and N .Specifically,

$$S_{k,i} = \left\{ r \middle| kr \equiv i (\text{mod}\, N), r \in \{0,1,2,\text{L}, N-1\} \right\} \quad i,k = 0,1,2,\text{L}, N-1. \tag{2}$$

The sum of each of these subsets of sampled values is denoted by

$$y_k(i) = \begin{cases} \sum_{r \in S_{k,i}} x(r) & \text{if } S_{k,i} \neq \phi \\ 0 & otherwise \end{cases} \qquad k,i = 0,1,2,\text{L}, N-1 \ . \tag{3}$$

So for $N = 4$,

$$\begin{aligned} &y_0(0) = x(0) + x(1) + x(2) + x(3), y_0(1) = 0, y_0(2) = 0, y_0(3) = 0; \\ &y_1(0) = x(0), y_1(1) = x(1), y_1(2) = x(2), y_1(3) = x(3); \\ &y_2(0) = x(0) + x(2), y_2(1) = 0, y_2(2) = x(1) + x(3), y_2(3) = 0; \\ &y_3(0) = x(0), y_3(1) = x(3), y_3(2) = x(2), y_3(3) = x(1). \end{aligned} \tag{4}$$

In other words, using (3) and (4), the sampled sequence $x(r)$ is mapped into a new sequence $y_k(i)$ for each value of k by summing the terms of the sampled sequence that have the same multiplier of complex exponential function. We are then ready to rewrite (1) in terms of $y_k(i)$ instead of $x(r)$ as follows:

$$X(k) = \sum_{r=0}^{N-1} x(r) e^{-j2\pi rk/N} = \sum_{i=0}^{N-1} y(i) e^{-j2\pi i/N} \qquad k = 0,1,2,\text{L}, N-1 \tag{5}$$

In deriving (6), we have used the periodic property that

$$e^{-j2\pi rk/N} = e^{-j2\pi (rk \bmod N)/N} = e^{-j2\pi i/N}.$$

Applying the theorem of extended law of the mean to $e^{-j2\pi i/N}$:

$$e^{-j2\pi i/N} = \sum_{r=0}^{p} (-j2\pi i / N)^r / r! + R_i, \qquad i = 0,1,2,\text{L}, N-1.$$

Substituting the above equation into (5), yields

$$\begin{aligned} X(k) &= y_k(0) + \sum_{i=1}^{N-1} y_k(i) \left[\sum_{r=0}^{p} (-j2\pi i / N)^r / r! \right] + R_p \\ &= y_k(0) + \sum_{r=0}^{p} \left[(-j2\pi)^r / N^r r! \right] \sum_{i=1}^{N-1} y_k(i) i^r + R_p \\ &= y_k(0) + \sum_{r=0}^{p} a_r m_k(r) + R_p, \qquad k = 0,1,2,\text{L}, N-1 \end{aligned} \tag{6}$$

where

$$a_r = (-j2\pi)^r / N^r r! \tag{7}$$

$$m_k(r) = \sum_{i=1}^{N-1} y_k(i) i^r \tag{8}$$

and

$$R_p = \begin{cases} \sum_{i=1}^{N-1} y_k(i)(\cos(\xi_{i1} + (p+1)\pi/2)(2\pi i/N)^{p+1}/(p+1)! \\ \qquad - j\sin(\xi_{i2} + p\pi/2)(2\pi i/N)^p/p!) \\ \quad 0 < \xi_{i1}, \xi_{i2} < 2\pi i/N, \quad \text{if } p \text{ is even} \\ \sum_{i=1}^{N-1} y_k(i)(\cos(\xi_{i1} + p\pi/2)(2\pi i/N)^p/p!) \\ \qquad - j\sin(\xi_{i2} + (p+1)\pi/2)(2\pi i/N)^{p+1}/(p+1)!) \\ \quad 0 < \xi_{i1}, \xi_{i2} < 2\pi i/N, \quad \text{if } p \text{ is odd.} \end{cases} \tag{9}$$

If R_p is ignored, $X(k)$ can be computed as follows:

$$X(k) = y_k(0) + \sum_{r=0}^{p} a_r m_k(r), \qquad 0 \le k \le N-1. \tag{10}$$

Suppose $p > 2\pi$ and p is even, thus the error introduced by ignoring R_p is bounded by

$$|R_p| \le \max_r |x(r)| 2^{1/2}(N-1)(2\pi)^P / p!. \tag{11}$$

When $p > 2\pi$ and p is odd, the same results can be proved. For example, when $\max_r |x(r)| \le 256(0 \le r \le N-1)$, $N \le 2K, P = 34$,

$$|R_p| \le 3.45 \times 10^{-6}.$$

Increasing p to 35, we get

$$|R_p| \le 6.20 \times 10^{-7}.$$

Thus the error converges to zero very rapidly and uniformly and the approximation (10) can satisfy the accuracy requirement of most applications without computing too many terms. It is obvious that p is a slow-growing function of N . Suppose $\max_r |x(r)| \le 256(0 \le r \le N-1)$;

Some cases of N , p and R_p are listed in Table 1.

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 infinity. In Table 1, p can be expressed approximately in the firm of $[2\log_2 N / \log_2 \log_2 N] + 32$.

The computation of $X(k)$ using the approximation of (10) establishes the relationship between DFT and moments which involves the generation of the r th order moments of the transformed data sequence $y_k(i)$ and then performing a dot product of these moments

with a constant vector (a_r) and an addition with $y_k(0)$.

Table 1. Some Cases Of N, p, R_p And $R_p^{'}$

N	2^{10}	2^{15}	2^{20}	2^{25}	2^{30}	2^{35}	2^{40}	2^{50}
$2\log_2 N$	20	30	40	50	60	70	80	100
p	35	37	39	41	43	45	47	50
R_p	3.1×10^{-7}	2.9×10^{-7}	2.5×10^{-7}	1.9×10^{-7}	3.1×10^{-7}	3.1×10^{-7}	3.1×10^{-7}	3.1×10^{-7}
$R_p^{'}$	1.3×10^{-2}	4.2×10^{-4}	1.5×10^{-5}	4.7×10^{-7}	1.5×10^{-8}	5×10^{-10}	1.7×10^{-11}	1.7×10^{-14}

In (10), there is a dot product of the moments with a constant vector (a_r) to compute. When N is large, (a_r) is too small to compute. We can resolve this problem and transform the product of floating-point into additions of integers by the following steps.

When $N=2^k$, multiplying a_r by $2^{g_r}=2^{\log N^{r+2}+10}$ ($r=0,1,\mathrm{L},p$), we get: $[a_r\times 2^{g_r}]=[(-j2\pi)^r/N^r r!\times 2^{\log N^{r+2}+10}]\le(-j2\pi)^r/r!\times 2^{\log N^2+10}<(-j)^r 86\times 2^{\log N^2+1}$ which are integers and can be represented as sums of distinct powers of 2:

$$[a_1\times 2^{g_1}]=\sum_{i=0}^{t}-jn_{1,i}2^{t-i}=-j(n_{1,0}2^t+n_{1,1}2^{t-1}+n_{1,2}2^{t-2}+\mathrm{L}+n_{1,t-1}2+n_{1,t})$$

$$[a_2\times 2^{g_2}]=\sum_{i=0}^{t}(-j)^2 n_{2,i}2^{t-i}=(-j)^2(n_{2,0}2^t+n_{2,1}2^{t-1}+n_{2,2}2^{t-2}+\mathrm{L}+n_{2,t-1}2+n_{2,t})$$

L

$$[a_p\times 2^{g_p}]=\sum_{i=0}^{t}(-j)^p n_{p,i}2^{t-i}=(-j)^p(n_{p,0}2^t+n_{p,1}2^{t-1}+n_{p,2}2^{t-2}+\mathrm{L}+n_{p,t-1}2+n_{p,t})$$

(12)

where $n_{m,i}=0or1$ ($m=1,\mathrm{L},p,\ i=1,\mathrm{L},t.$). Since $[a_r\times 2^{g_r}]<(-j)^r 86\times 2^{\log N^2+1}$, we can know $2^t<86\times 2^{\log N^2+1}$, i.e. $t<\log N^2+17=2\log N+17$. For convenient, we let $t=2\log N+17$ in the below equations.

Then:

$$y_k(0)+(\sum_{r=0}^{p}a_r m_k(r))=y_k(0)+m_k(0)+(\sum_{r=1}^{p}a_r m_k(r))$$

$$= y_k(0) + m_k(0) + \sum_{r=1}^{p}(a_r \times 2^{\log N^{r+2}+10})(m_k(r) / 2^{\log N^{r-1}-10}) / 2^{\log N^3+20}$$

$$= y_k(0) + m_k(0) + (\sum_{r=1}^{p}[a_r \times 2^{g_r}][m_k(r) / 2^{\log N^{r-1}-10}]) / 2^{3\log N+20} + R_p^{'}$$

$$= y_k(0) + m_k(0) + (\sum_{r=1}^{p}\sum_{i=0}^{t}(-j)^r n_{r,i} 2^{t-i} m_k^{'}(r)) / 2^{3\log N+20} + R_p^{'}$$

$$= y_k(0) + m_k(0) + \sum_{i=0}^{t}(\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)) \times 2^{t-i} / 2^{3\log N+20} + R_p^{'} \quad (13)$$

$$(m_k^{'}(r) = [m_k(r) / 2^{\log N^{r-1}-10}]),$$

where

$$\left|R_p^{'}\right| = \left|\sum_{r=1}^{p}((a_r \times 2^{g_r})(m_k(r) / 2^{\log N^{r-1}-10})) / 2^{3\log N+20} - (\sum_{r=1}^{p}[a_r \times 2^{g_r}][m_k(r) / 2^{\log N^{r-1}-10}]) / 2^{3\log N+20}\right|$$

$$= \left|\begin{array}{l}\sum_{r=1}^{p}((a_r \times 2^{g_r})(m_k(r) / 2^{\log N^{r-1}-10})) / 2^{3\log N+20} - \sum_{r=1}^{p}([a_r \times 2^{g_r}](m_k(r) / 2^{\log N^{r-1}-10})) / 2^{3\log N+20} \\ + \sum_{r=1}^{p}([a_r \times 2^{g_r}](m_k(r) / 2^{\log N^{r-1}-10})) / 2^{3\log N+20} - \sum_{r=1}^{p}([a_r \times 2^{g_r}][m_k(r) / 2^{\log N^{r-1}-10}]) / 2^{3\log N+20}\end{array}\right|$$

$$\le \left|\sum_{r=1}^{p}(a_r \times 2^{g_r} - [a_r \times 2^{m_r}])(m_k(r) / 2^{\log N^{r-1}-10}) / 2^{3\log N+20}\right|$$

$$+ \left|\sum_{r=1}^{p}[a_r \times 2^{g_r}](m_k(r) / 2^{\log N^{r-1}-10} - [m_k(r) / 2^{\log N^{r-1}-10}]) / 2^{3\log N+20}\right|$$

$$\le \left|\sum_{r=1}^{p}(m_k(r) / 2^{\log N^{r-1}-10}) / 2^{3\log N+20}\right| + \left|\sum_{r=1}^{p}[a_r \times 2^{g_r}] / 2^{3\log N+20}\right|$$

$$< \sum_{r=1}^{p}(256 \times N^{r+1} / 2^{\log N^{r-1}-10}) / 2^{3\log N+20} + \sum_{r=1}^{p}84 \times 2^{2\log N+10} / 2^{3\log N+20}$$

$$\le \sum_{r=1}^{p}256 / 2^{\log N+10} + \sum_{r=1}^{p}84 / 2^{\log N+10}$$

$$= p / 2^{\log N+2} + 84p / 2^{\log N+10}$$

$$< p / (4N) + p / (8N) = 3p / (8N)$$

(supposing $\max_r \left|x(r)\right| \le 256$,we can get that $m_k(r) = \sum_{i=1}^{N-1} y_k(i) i^r \le 256N \times N^r$).

It is obvious that the larger the N , the smaller the error $R_p^{'}$, which can satisfy the accuracy requirement of most applications. Some cases of N , p and $R_p^{'}$ are listed in Table 1.

Thus, $X(k)$ can be approximately computed as follows:

$$X(k) = y_k(0) + m_k(0) + \sum_{i=0}^{t}(\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)) \times 2^{t-i} / 2^{3\log N+20} \quad (14)$$

where $n_{r,i} = 0 or 1$ and $j^2 = -1$.

By doing so, all the products of floating-point constants with moments in (10) are eliminated, replaced by shifting the digits and accumulations of integers. Since $n_{r,i}$ can be computed advance and reused in a real-time system, less than $p(t+1)+2$ additions of integers and $t+2+p$ shifts are required to implement $X(k)$ with (14) once all moments are produced. $p(t+1)N+2N < p(2\log N+17)N+2N$ additions and $(t+2+p)N$ shifts are required to perform $X(0), X(1), \mathrm{L}, X(N-1)$ ($t = 2\log N+17$).

When $N \neq 2^k$, $\log N$ is not an integer, but substituting $2^{3\log N+20}$ by $2^{[3\log N+1]+20}$, $2^{\log N^{r-1}-10}$ by $2^{[\log N^{r-1}+1]-10}$, and $2^{g_r} = 2^{\log N^{r+2}+10}$ by $2^{[3\log N+1]+[\log N^{r-1}+1]+10}$ in (14), we can get the same result and prove $p(t+1)N+2N$ (here $t=[2\log N+18]$) additions are required and $\left|R_p^{'}\right| < 3p/(8N)$.

For convenient, we let $t=[2\log N+18]$ either when $N=2^k$ or when $N \neq 2^k$. In effect, the additions and error $R_p^{'}$ are much less than the above result because there are so many 0s of $n_{r,i}$.

3. Complexity Analysis And Systolic Implementation Of DFT

According to section 2, our 1-D DFT can be implemented by the following three procedures:

(Input initial $N, p, t, x(0), x(1), \mathrm{L}, x(N-1)$)

1. Compute $y_k(i)$ and $n_{r,i}$ for $k, i = 0,1,2,\mathrm{L}, N-1$ and $r = 1,\mathrm{L}, p,$;

2. Compute $m_k(r)$ for $k, r = 0,1,2,\mathrm{L}, N-1$;

3. Compute $X(k)$ with (14) for $k = 0,1,2,\mathrm{L}, N-1$.

Next we analyze the complexity of the algorithm. Step 1 constitutes a preprocessing step which involves three equations, (3), (4), (8) and (12). The parameters a_r and the index set S_{ki} can be computed in advance, so these computations are not part of the real-time system. It is noteworthy that $[a_r \times 2^{g_r}]$ is a real number or a pure imaginary one only with relation to N and can be obtained and expressed as (12) in advance. It remains to consider the generation of $y_k(i)$. In next section, we introduce a method to compute all $y_k(i)$ with $N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N)$ additions.

To compute $y_k(i)$ ($k, i = 0,1,2,\mathrm{L}, N-1$), we use the p-network method presented in [3], which require less than $(p+1)(p+2)(N-2)N/2$ additions because many $y_k(i)$ are 0's. In fact, the number of $y_k(i) \neq 0$ is $\sum_{k=0}^{N-1} N/\gcd(k,N)$ (see Appendix 1). In another words, only $\frac{(\sum_{k=0}^{N-1} 1/\gcd(k,N))(p+1)(p+2)(N-2)}{2}$ additions are required to complete all the moments. From section 3, the computation of (14) involves $p[2\log N+18]N+2N$ additions and $([2\log N+18]+2+p)N$ shifts

So there are all together about

$$N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N) + \frac{(\sum_{k=0}^{N-1} 1/\gcd(k,N))(p+1)(p+2)(N-2)}{2} + p[2\log N+18]N + 2N$$

Additions in the algorithm. If N is a prime number, then these are $(p+1)(p+2)(N-2)N/2 + p[2\log N+18]N + 2N$ additions in the algorithm. All the

p's mentioned above can be expressed in the form of $[2\log_2 N/\log_2\log_2 N]+32$ in the case of $N<2^{100}$ and $\max_r|x(r)|\le 256(0\le r\le N-1)$. In any case, $p=O(\log_2 N/\log_2\log_2 N)$.

Compared with the direct method having a computational complexity $O(N^2)$ and the conventional FFT having $O(N\log_2 N)$, our method is superior to the direct method and seems to be inferior to FFT if the calculations are executed in a sequential machine (for example given in Section 2, $N\le 2K, \log_2 N\le 11, P=34$ or 35). However, if the sampled data are real numbers, since a_r ($r=1,\text{L}, p,$) is either a real number or a pure imaginary one, then additions involved in our method are all real operations. Even though sampled data are complex numbers. Since one complex addition is equivalent to two real additions, our method has an advantage over a FFT which involves complex operations. The traditional FFT requires $O(N\log_2 N)$ complex multiplications and additions, for example, radix-2 FFT require $N\log_2 N/2$ multiplications that are equivalent to $2N\log_2 N$ real number multiplications and $N\log_2 N$ real additions. When N is large enough ($N\ge 2^{20}$), p can get a value much lower than $2\log_2 N$ and still satisfy the accuracy requirement as demonstrated in Table 1 and $p=O(\log_2 N/\log_2\log_2 N)$.

Our proposed algorithm seems to require a larger number of additions than many fast Fourier transforms, but it is easily implemented by systolic arrays because it only involves integer additions and shifts.

The general network for implementing DFT is shown in Figure 1. It consists of the moment generator [4], the preprocessing arrays, and the shift and accumulation array. The moments generator formed of $N-2$ sections of Pascal triangles with a row of adder-latch could be used to generate the 1-D moments. It receives $y_k(i)$ ($i=0,1,2,\text{L},N-1$) that the processing arrays produce. The shift and accumulation array (Figure 1) receives $m_k'(r)$ which can be performed by shifting digits of $m_k(r)$ to the left $\log N^{r-1}-10$ places and computes (14) . The numbers in square brackets that are below the horizontal line denote the amount of time delay to keep synchronous pace in Figure 1. The scheduling of the dataflow is

$\sum_{i=0}^{t}(\sum_{r=1}^{p}(-j)^r n_{r,i}m_k'(r))\times 2^{t-i}/2^{3\log N+20}$ such that produced by the shift and accumulation array moves from left to right, and accumulates with $y_k(0)+m_k(0)$ to produce the $X(k)$. The total execution time of the systolic arrays is

$N+(p+1)(N-2)+2+\log(p([2\log N+18]+2))$.

The first term N is for preprocessing, the second for producing moments [3], and last for (14).

It remains to consider the generation of $y_k(i)$.In the general case, we propose using a special linear array to implement $y_k(i)$, $k=0,1,2,\text{L},N-1$. Each element $x(r)$ is tagged with a rank $=kr(\text{mod}\,N)$ 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 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 2 illustrates an example. A cell in a linear contains an adder only if the corresponding partition S_{ki} has a size greater than 1. In 8 clocks, $y_2(0)$ will emerge at the right-hand side, as shown in Figure 2. In general, it takes $2N$ clocks for the vector to appear on the right. It can be proved that the number of additions required in the preprocessing array for

each k is equal to $N - N/\gcd(k,N)$. So the total number of additions required in the preprocessing array is equal to $\sum_{k=0}^{N-1}(N - N/\gcd(k,N)) = N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N)$. we can prove that

$N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N) < N^2 - e^{-\gamma} N^2/\log_2\log_2 N$ (γ is the Eular's constant), when N tends to infinity. Exactly, $N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N) < [N^2/3]$ when $N = 2^k$, and $N^2 - \sum_{k=0}^{N-1} N/\gcd(k,N) < [3N^2/4]$ when $N < 6\times 2^{30}$.

The moments generator was formed by $N-2$ p-network which contains pN latches and $p(p-1)N/2 - pN$ adder-latches [4].

The shift and accumulation array (Figure 3) can compute (14) once the moments are produced. Since $n_{r,i} = 0 or 1$ and $m_k^{'}(r)$ which can be performed by shifting digits of $m_k(r)$ to the left $\log N^{r-1} - 10$ places in binary system are integers, the computation of $\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)$ only involves accumulations of $m_k(r)$ $i = 0, 1, ..., p$. By shifting digits of $\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)$ to the left $t-i$ places in binary system, the multiplication of 2^{t-i} can be performed, and shifting $\sum_{i=0}^{t}(\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)) \times 2^{t-i}$ to the right $3\log N + 20$ places in binary system, $\sum_{i=0}^{t}(\sum_{r=1}^{p}(-j)^r n_{r,i} m_k^{'}(r)) \times 2^{t-i} / 2^{3\log N+20}$ can be performed. This array was formed by about $pt\log(pt)$ adder-latches and some shifters and the execution time is about $\log(p([2\log N + 18] + 2))$.

Compared with the designs in [2, 6, 17, 22] (Table 2), our systolic arrays do not need multipliers though the adders are more (in most case, our p is nearly a constant no larger than 50 and $t = [2\log N + 18]$, so the adders are not much more). This means that our method is easily implemented by hardware and is very suited to a real-time processing. Meaning while, our method does not have limits on N, which is better than other FFTs.

Table 2. Comparison of Parameters for Linear Array, 2-D Systolic, Base-4, Pipelined FFT and Our Systolic Circuits

Architecture	multiplications	Multipliers	Adders	Limits on N
Linear array	N^2	N	N	none
2-D systolic [10],[21]	$N(N_1 + N_2 + 1)$	N	N	$N = N_1 N_2$
Base-4 [19]	$(N_1 + N_2)N/16$	$N_1/4$	$2N_2$	$N = 256n$
pipelined FFT	$(N(3n-8) - (-1)^n)/9 + 1$	$\log_4 N - 1$	$\log_4 N$	$N = 2^n$
Our method	none	none	$p(p-1)N/2 - pN + pt\log(pt)$	none

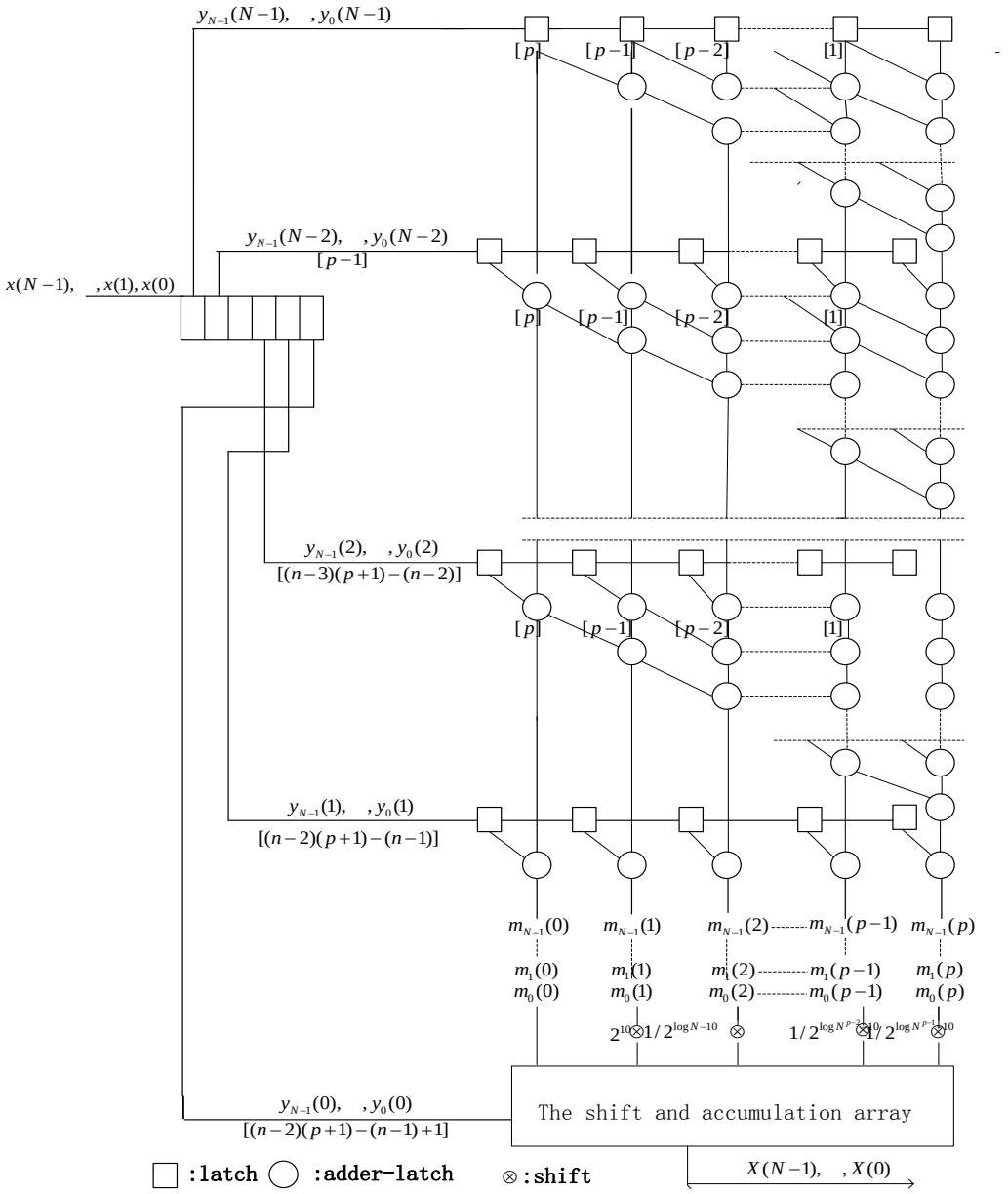

Figure 1. The Systolic Array for 1-D DFT

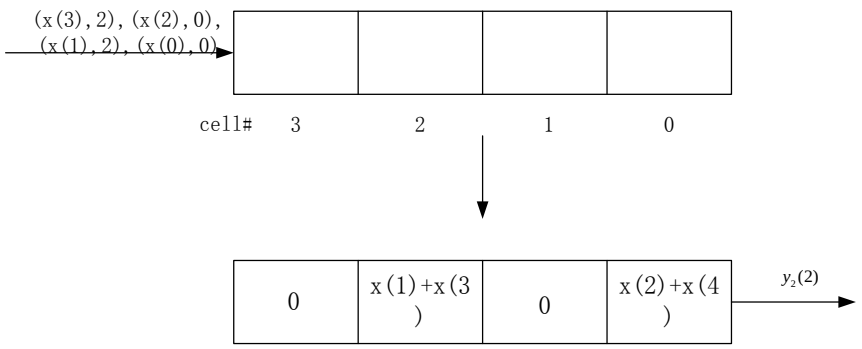

Figure 2. The Preprocessing Arrays

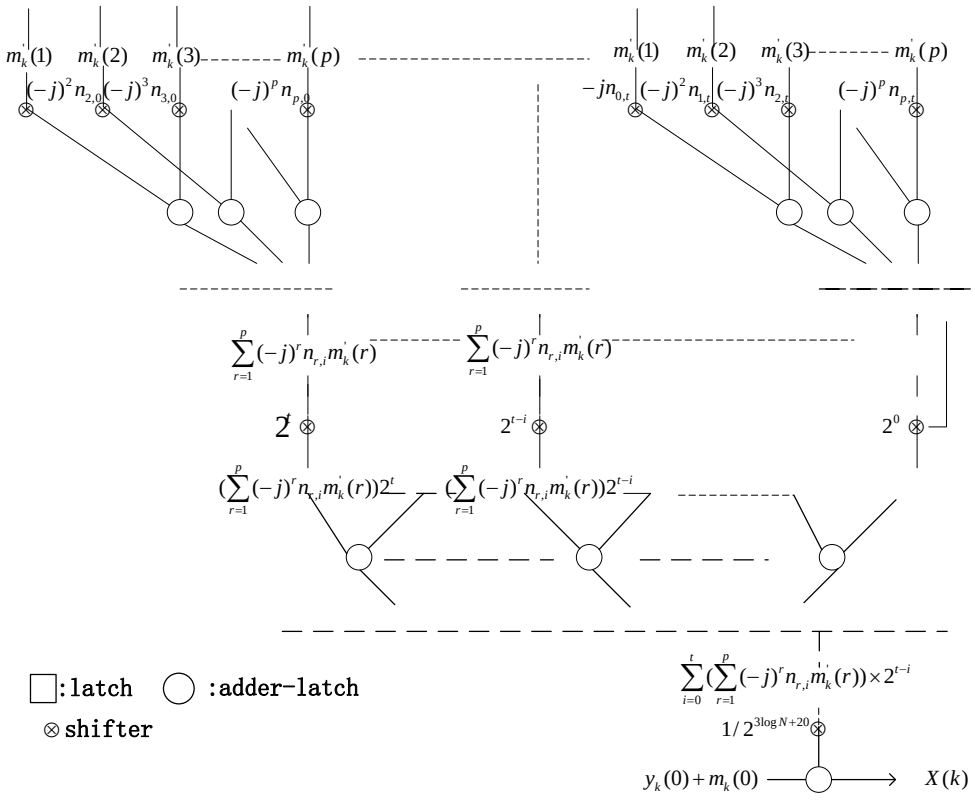

Figure 3. The Shifting and Accumulation

4. Computing Correlation By Correlation Theorem And Complexity Analyses

The Correlation of two N -point sequences $\{x(n)\}$ and $\{h(n)\}$ can be given by

$$x(n) \circ h(n) = \frac{1}{N}\sum_{i=0}^{N-1} x^*(i)h(n+i) \tag{15}$$

where $x^*(i)$ denotes Complex conjugation the of $x(n)$.

The Correlation theorem says that in the frequency domain,

$$S(k) = X^*(k) \cdot H(k) \qquad k = 0,\text{L},2N-1 \tag{16}$$

where $\{S(k)\}$ is the DFT of $\{x(n) \circ h(n)\}$, and $\{X(k)\}$ and $\{H(k)\}$ are the DFT of the sequences $\{x_1(k)\}$ and $\{h_1(k)\}$ which are augmented by $\{x(n)\}$ and $\{h(n)\}$ as follows ,

$$x_1(k) = \begin{cases} x(k) & k = 0,1,\text{L},N-1 \\ 0 & k = N,\text{L}\ 2N-1 \end{cases} \text{ and } h_1(k) = \begin{cases} h(k) & k = 0,1,\text{L},N-1 \\ 0 & k = N,\text{L}\ 2N-1 \end{cases}$$

So it is computationally efficient to implement the Correlation of $\{x(n)\}$ and $\{h(n)\}$ by the following procedures:

1) Compute the DFT of $\{X(k)\}$ and $\{H(k)\}$, $\{x_1(k)\}$ and $\{h_1(k)\}$ using method in section 4, respectively;

2) Compute the product of $\{X(k)\}$ and $\{H(k)\}$ for $0 \le k \le 2N-1$;

3) Compute the sequence $\{x(n) * h(n)\}$ as the inverse DFT of $\{X^*(k)H(k)\}$ using method in section 4.

In other words, to calculate Correlation of two length-N sequences, we need to compute two length-$2N$ DFT, an length-$2N$ inverse DFT, and a product of two length-$2N$ vector. From section 4, we can know that there are all together about

$$4N^2 - \sum_{k=0}^{N-1} 2N / \gcd(k, 2N)$$
$$+ \frac{(\sum_{k=0}^{N-1} 1/\gcd(k, 2N))(p+1)(p+2)(2N-2)}{2}$$
$$+2p[2\log 2N + 18]N + 4N$$

Integer additions and $2N$ multiplications to complete the computation of two length-N sequences.. Similarly, a systolic array can be designed to calculate the Correlation (Figure 4). It consists of four parts, two systolic arrays for DFT designed in section 4, a multiplier, and a systolic array for inverse DFT. So the whole arrays require $p(p-1)N - 2pN + pt\log(pt)$ adders and one multiplier. The total execution time of the systolic arrays is $2(2N + (p+1)(2N-2) + 2 + \log(p([2\log(2N)+18]+2)) + 1$. The first term is for DFT and inverse DFT, the second for product of $X(k)$ and $H(k)$. In this case, $p = O(\log_2(2N)/\log_2\log_2(2N))$ and $t = [2\log(2N)+18]$.

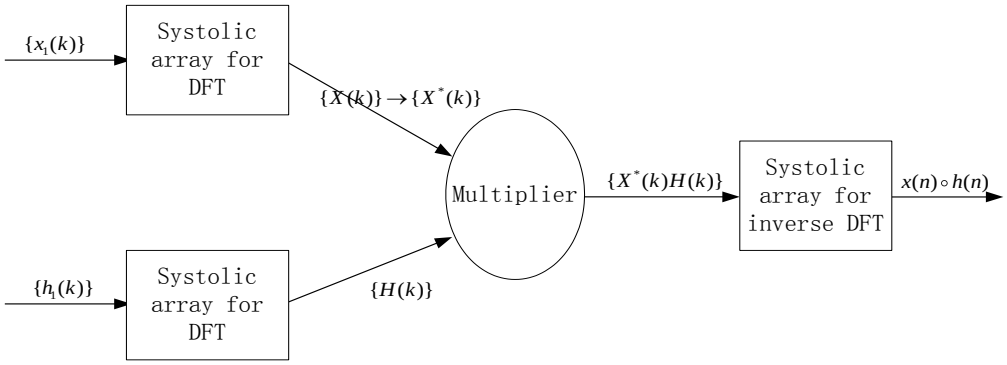

Figure 4. The Systolic Arrays for Correlation

5. Discussion and conclusion

By transforming the floating-point multiplication into operation of shifts and accumulation of integers, we introduce an improved algorithm and systolic arrays for 1-D DFT and have analyzed the computational complexity, then we extend our algorithm to calculate Correlation.

Compared with the work in [17], our DFT does not have multiplications and the amount of additions to eliminate all the multiplications is very small. Our method has also the following advantages: there are no multiplications in our method, which is superior to the $O(N\log_2 N)$ in classical FFT; exponential functions have been replaced by simple polynomial functions and all multiplications have been changed into additions, which decreases the computational cost and memory requirement. One important advantage is that it can accommodate data samples of arbitrary length and compute any portion of frequency. In addition, since our method only involves integer operations, it produces nice accuracy and convergence property. Compared with the systolic arrays in [4], our systolic arrays require less multipliers and have the property of regularity. What is more important, our method can control the accuracy and hence can get high precision.

Acknowledgments

Our study was funded in part by the the National Natural Science Foundation of China (Grant No. 61105004), NSFC of Guangxi (No 2013GXNSFBA019279) and Guangxi Key Laboratory of Automatic Detecting Technology and Instruments (No. YQ14103). We would like to express our appreciation to all supporters above mentioned for their strongly financial support.

References

[1] R. Agarwal and J. Cooley, New algorithms for digital convolution, IEEE Trans Acoust., Speech, Signal Process., 25(Oct. **1977**), 392,
[2] J. A. Beraldin, T. Aboulnasr, W. Streenaart, Efficient 1-D systolic array realization for the discrete Fourier Transform, *IEEE Trans. Circuits Syst*., 36 (**1989**), .95-100,
[3] R. E. Blahut, Fast Algorithms for Digital Signal Processing, Addison-Wesley, Reading, MA, **1984**.
[4] F. H. Y. Chan, F. D. Lam, H. F. Li, J. G. Liu. An all adder systolic structure for fast computation of moments, *J.VLSI Signal Process*, 12 (**1996**), 159-175.
[5] H. -C. Chen , J. -I. Guo , C. -W. Jen and T. -S. Chang, Distributed arithmetic realisation of cyclic convolution and its DFT application, *Proc. IEE, Circuits, Devices, Syst.*, pp. 615, Dec. **2005**.
[6] C. Cheng, K. K. Parhi, Low-cost fast VLSI algorithm for discrete Fourier Transform, *IEEE Transactions on circuits and systems I*, 54(**2007**), 791-806.
[7] J. B. Cooley , J. W. Tukey, An algorithm for machine computation of complex Fourier series, *Math. Comput.* , 19 (**1965**), 297-301.
[8] P. Duhamel, Paper on the Fast Fourier Transform, IEEE Press, New York, **1995**.
[9] P. Duhamel, M. Vetterli. Fast Fourier transform: A tutorial review and a state of the art, *Signal Processing*. 19 (**1990**), 259-299.
[10] J. I. Guo, A new distributed arithmetic algorithm and its hardware architecture for the discrete hartley transform, Pattern Recong. Image Anal., 10(**2000**), 368-378.
[11] M. Hatamian. A real-time two-dimensional moment generating algorithm and its single chip implementation, IEEE Trans. Acoust, Speech Signal Process, 34(**1986**), 546-553.
[12] S. He, M. Torkelson, Designing pipeline FFT Processor for OFDM (de) Modulation, in Pro .IEEE URSI Int. symp. Signals, syst., Electron. (**1998**) 257-262.
[13] S. He, M. Torkelson, A systolic array implementation of common factor algorithm to compute DFT, in Proc. Int. Symp. Parallel Architectures, Algorithms and Networks, Kanazawa, Japan, (**1994**), 374-381.
[14] M. K. Hu, Visual pattern recognition by moment invariants, IRE Trans. Inform. Theory, 8(**1962**), 179-187.
[15] L. Jia, Y. GAO, J. Isoaho, H. Tenhunen, A new VLSI-oriented FFT algorithm and implementation, in Proc. Eleventh Annuo IEEE. Int. ASIC Conf., (**1998**), 337-341.
[16] K. J. Jones , Prime number DFT computation via parallel circular convolvers, Proc. IEE, Radar Signal Process., 137(Jun. **1990**), 205,.
[17] D. C. Kar , V. V. B. Rao, A new systolic realization for the discrete Fourier transform, IEEE Trans. Signal Process, 41(**1993**), 2008-2010.
[18] H. T. Kung, C. E. Leiserson, Systolic arrays (for VLSI), in Symp. Sparse Matrix Computations, (**1978**), 256-282.
[19] H. Lim, E. E. Swartzlander, Multidimensional systolic arrays for the implementation of the discrete Fourier transform, IEEE Trans. Signal process, 47(**1999**), 1359-1370.
[20] J. G. Liu, H. F. Li, F. H. Y. Chan, F. K. Lam, A novel approach to fast discrete Fourier transform, Journal of Parallel and Distributed Computing, 54(**1998**), 48-58.
[21] P. K. Meher, Highly concurrent reduced- complexity 2-D systolic array for discrete Fourier transform, IEEE signal processing letters,.13(**2006**),.481-484
[22] J. G. Nash, Computationally efficient systolic architecture for computing the discrete Fourier Transform, IEEE Trans. Signal process, 53(**2005**), 4640-4651.
[23] W. Yeh, and C. Jen, High-speed and low-power split-radix FFT, IEEE Trans. Signal process, 51(**2003**), 4640-4651.
[24] M. F. Zakaria, L. K. Vroomen, P. J. A. Zsombar-Murray, J. M. H. M. Van Kessel, Fast algorithm for computation of moment invariants, Pattern Recognition,.20(**1987**), 634-643.
[25] P. K. Meher, Hardware-Efficient Systolization of DA-Based Calculation of Finite Digital Convolution, IEEE Transactions on circuits and systems Ⅱ, 53(**2006**), 707-711.

