* * * * * *

DESIGN PROJECT FOR EE 524

A COMPLETE ALGORITHM FOR COMPUTING THE FFT ON 2D ARRAY PROCESSOR

Benmei Chen

P.O. Box 241, Gonzaga University

Spokane, WA 99258

May 5, 1987

A Complete Algorithm For Computing The FFT

On 2D Array Processor

I. INTRODUCTION

In this project, a complete algorithm for computing the fast Fourier transform (FFT) on 2D array processor has been introdued. The algorithm is including the computing procedure, data storage, data transferring and the weight assignment. The specific feature for this algorithm is to route the data simply according to a recursive pseudo transformation.

II. DATA STORAGE, DATA MOVEMENT AND WEIGHT ASSIGNMENT

Assuming the 2D array processor has Q*Q processing elements, where Q=2 and q is a integer. And the N point FFT is to be considered in this problem, where N=2 . The data for N points are stored on a 1D vector Vo. Also using V(i,j,k) to denote the data locations on the each processing element (PE), []c and []a to denote the intrger ceiling and floor fuction respectively.

1. DATA STORAGE:

The following equation defines the storage from 1D vector Vo to V(i,j,k) ,

$$V(i,j,k) = Vo(i + j*Q + k*Q*Q)$$

For example, Vo has 64 elements and Q=4. The data locations are shown in Figure 1 .

Figure 1. The Data Locations For 2D Array Processor

Or, we just simply redraw the figure as below,

```
48 49 50 51   52 53 54 55   56 57 58 59   60 61 62 63
32 33 34 35   36 37 38 39   40 41 42 43   44 45 46 47
16 17 18 19   20 21 22 23   24 25 26 27   28 29 30 31
00 01 02 03   04 05 06 07   08 09 10 11   12 13 14 15
_____ _____   _____ _____   _____ _____   _____ _____

_____________   _____________   _____________   _____________
PE PE PE PE   PE PE PE PE   PE PE PE PE   PE PE PE PE
```

Figure 2. The Redrawn Data Locations For 2D Array

2. DATA MOVEMENTS:

Define the recursive pseudo transformation PT as

$$PT\ \{\ A,\ V(i,j,k)\ \} = V(i',j',k')$$

where A is a constant and

$$s = [k*A]a+[(i+j*Q)*Q*Q/(A*N)]a*N/(Q*Q)c+A$$

$$i' = s + i \bmod[A]c - [s/Q]a * Q$$

$$j' = [s/Q]a + j \bmod[A/Q]c + j \bmod[Q/A]c$$

$$k' = [(i+j*Q) \bmod[A*N/(Q*Q)]c/A]a+k \bmod[1/A]c$$

Then the following simple algorithm will complete all data routing requirements in computing the FFT.

An algorithm for routing the data to the appreciate location:

```
A = Q^2/N
WHILE  A>1 DO
   BEGIN
   V(i,j,k) = PT { A, V(i,j,k) }
   A = A*Q^2/N
   END
V(i,j,k) = PT { A, V(i,j,k) }
```

Using the same example as that in section Data Storage, the data movements are shown in Figure 3. Also the other example is considered and shown in Figure 4.

```
48 49 50 51  52 53 54 55  56 57 58 59  60 61 62 63
32 33 34 35  36 37 38 39  40 41 42 43  44 45 46 47
16 17 18 19  20 21 22 23  24 25 26 27  28 29 30 31
00 01 02 03  04 05 06 07  08 09 10 11  12 13 14 15
```

(a)

```
12 13 14 15  28 29 30 31  44 45 46 47  60 61 62 63
08 09 10 11  24 25 26 27  40 41 42 43  56 57 58 59
04 05 06 07  20 21 22 23  36 37 38 39  52 53 54 55
00 01 02 03  16 17 18 19  32 33 34 35  48 49 50 51
```

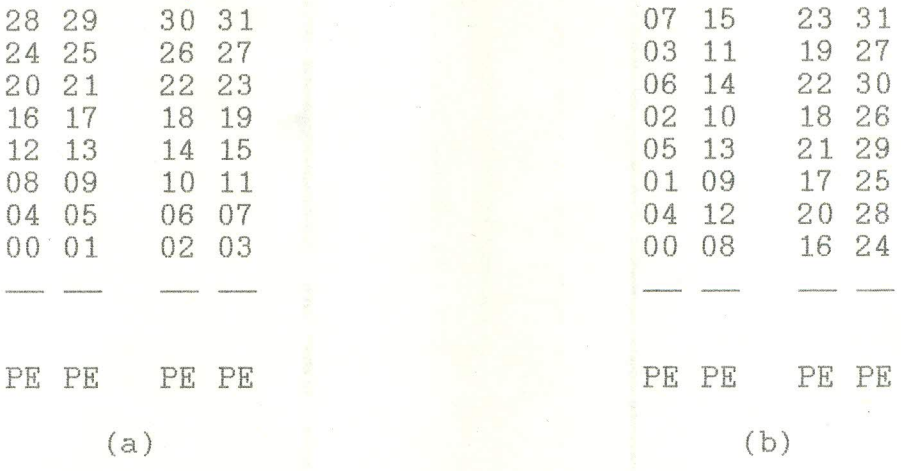
(b)

```
03 07 11 15  19 23 27 31  35 39 43 47  51 55 59 63
02 06 10 14  18 22 26 30  34 38 42 46  50 54 58 62
01 05 09 13  17 21 25 29  33 37 41 45  49 53 57 61
00 04 08 12  16 20 24 28  32 36 40 44  48 52 56 60
-------------------------------------------------
PE PE PE PE  PE PE PE PE  PE PE PE PE  PE PE PE PE
```

(c)

Figure 3. The data movements for N=64 on 4X4 array.

The example for Vo having 32 elements and Q=2,



(a) (b)

Figure 4. Data routings for N=32 on 2X2 array.

3. WEIGHT ASSIGNMENT:

Using the example shown in Figure 4 to explain the weight assignments in FFT oprations. Follows the weight assignments, there are the appreciate FFT calculations.

STEP 1: Assign W & W to V(i,j,k) for k>=4, respectively.

STEP 2: (a) Assign W & W to V(i,j,k) for 1<k<=3.

(b) Assign W & W to V(i,j,k) for 5<k<=7.

STEP 3: (a) Assign W & W to V(i,j,k) for k=1.

(b) Assign W & W to V(i,j,k) for k=3.

(c) Assign W & W to V(i.j.k) for k=5.

(d) Assign W & W to V(i,j,k) for k=7.

STEP 4: (a) Assign W & W to V(i,j,k) for k=4&6 and j=0.

(b) Assign W & W to V(i,j,k) for k=5&7 and j=0.

(c) Assign W & W to V(i,j,k) for k=4&6 and j=1.

... and so on.

III. THE COMPLETE ALGORITHM

Let V(k) denote V(*,*,k), r be the number of computing stages. And R = log N is the total number of computing stages for N point FFT. B represents the weights.

```
STEP 0:  STORE 1D VECTOR TO 2D ARRAY PROCESSOR V(i,j,k)
         V(i,j,k) = Vo(i+j*Q+k*Q*Q)
STEP 1:  m=log [N/(Q*Q)]
         r=0
         FOR  m'=0  TO  m-1   DO
            r=r+1
            B=[k/2   ]a * N/2
            V(k') = W*V(k) + V(k')
            V(k) = W   *V(k) + V(k')
         END OF m' LOOP
         A = Q /N

STEP 2:  WHILE  A>1  DO
          BEGIN
           V(i,j,k)= PT { A, V(i,j,k) }
           FOR  m'=0  TO  m-1  DO
              r=r+1
              B=[{i+j*Q+([k/2  ]a+k mod(2   ))*Q*Q}*2 /N]a*N/2
              V(k')=W *V(k) + V(k')
              V(k) =W  *V(k) + V(k')
           END OF m' LOOP
```

Parallel computing the complex number in each PE for all k mod(2)>=2 & k'mod( )< 2 .

The same parallel processing as that described above.

```
            A=A*Q*Q/N
          END
STEP 3: V(i,j,k) = PT { A, V(i,j,k) }
        m''= R-r-1
        FOR  m'=0  TO m''  DO
          B={(i+j*Q)/A+k mod(1/A)]*2 +[k/2  ]a}*N/2
          V(k')= W  *V(k) + V(k')        The same  parallel
                                         processing as that
          V(k) = W   *V(k) + V(k')       described above.
        END OF m' LOOP

    END OF THE ALGORITHM.
```

The complex calculations in the algorithm are defined as

```
V(k')=Re{V(k')} + J* Im{V(k')}
V(k) =Re{V(k)}  + J* Im{V(k)}
W  = cos(2* *B/N) - J*sin(2* *B/N)
W   = -cos(2* *B/N) + J*sin(2* *B/N)
W  *V(k)=[Re{V(k)}*cos(2* *B/N)+Im{V(k)}*sin(2* *B/N)]
      J* [Im{V(k)}*cos(2* *B/N)-Re{V(k)}*sin(2* *B/N)].
W *V(k)+V(k') = [ Re{W *V(k)}+Re{V(k')} ]
                + J* [ Im{W *V(k)}+Im{V(k')} ]
W  *V(k)+V(k')= [-Re{W *V(k)}+Re{V(k')} ]
                + J* [-Im{W *V(k)}+Im{V(k')} ].
```

where J= -1 .

IV. PERFORMANCE FOR N=512

Assume we have 8X8 array processor, that is Q=8 in this case. The performance for computing 512 point FFT opration is described in following.

STEP 0: STORE THE DATA

STEP 1: m=3 and SET r=0

Without data routing, we can parallel-compute 3 stage FFT oprations. In each PE , 4 complex multiplications and 4 adders are required for each stage.

A=8

STEP 2: For A=8>1, so, about 7 data locations in each PE have to be routed parallel with other PEs. Follows again 3 stage FFT oprations can be parallel computed, and required 4 complex multiplications and 4 adders in each PE for each stage.

A=1 , THEN GO TO STEP 3.

STEP 3: About 7 data locations in each PE have to be routed again.The other 3 stage FFT oprations will finish the performance. Once again, in each PE, 4 complex multiplications and 4 adders are needed for each stage.

The total performance is about

$$O(4*4 + 7 + 4*4 + 7 + 4*4) = O(64).$$