PARALLEL IMPLEMENTATION OF 2D-DFT USING WAVEFRONT ARRAYS

*KR Santha, **V Vaidehi
*Department of Electrical and Electronics Engg., Sri Venkateswara College of Engg., Pennalur.
**Department of Information Technology, Madras Institute of Technology, Chennai.
E-mail : santha@svce.ac.in

ABSTRACT

Two-dimensional Discrete Fourier Transform is the fundamental operation in many signal processing applications for which several hardware solutions have been provided in the literature. However they are systolic array based architectures. In this paper an attempt is made to develop a wavefront array based architecture for implementing the 2D-DFT. The algorithm is implemented as an $N \times N \times N$ matrix multiplication in two phases. In phase 1, the processors are data driven and each processor is independent of its operation. In phase 2, a set of four processing elements (stages) form a micropipeline and four such micropipelines operate in parallel. The stages operate by exchanging data via a two-phase handshaking protocol. The proposed scheme has been verified analytically.

1. INTRODUCTION

The DFT has a wide variety of applications such as radar, sonar, image, power spectrum analysis, general filtering, convolution etc. Recent applications include Asymmetrical Digital Subscriber Lines (ADSL) for ordinary phone services, Orthogonal Frequency Division Multiplexing (OFDM)[1] for broadcast and wireless communications and Advanced Cellular Internet Services (ACIS)[2] for mobile communications and computing. The processing of OFDM signals[1] is effected using inverse Fast Fourier Transform and Fast Fourier Transform for modulation and demodulation respectively. As the transform length $N$ increases, the increase in size of the FFT leads to communication overheads between the processors performing FFT/IFFT. Hence for large transforms multidimensional DFT's are widely used. They are better suited for parallel implementation using VLSI arrays and require no interprocessor communication. Several parallel systolic arrays have been presented for FFTs and one-dimensional or multidimensional DFT's [3],[4],[5]. In such synchronous designs, the clock distribution becomes an increasingly costly and complicated issue and power consumption rapidly emerges as a major concern.

Asynchronous design is a topic of latest research[6] as they take the advantage of not requiring a global clock. The wavefront arrays [7],[8],[9],[10] combine the asynchronous data driven properties of data flow machines with regularity, modularity and local communication properties of systolic arrays [7]. So an attempt is made in this paper to implement the 2D-DFT using wavefront arrays. The rest of the paper is organised as follows. Section 2 describes the 2D-DFT algorithm. Section 3 briefs the array architectures namely systolic and wavefront arrays and their role in VLSI implementation. Section 4 describes the implementation of wavefront architecture for 2D-DFT. Section 5 presents results and conclusion.

2. MATRIX-MATRIX FORMULATION FOR 2D-DFT

An $N$-point one-dimensional Discrete Fourier Transform is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk},$$

where $W_N = e^{-j(2\pi k/N)}$. Let $x(r,s); 0 \leq r, s < N$ be an $N \times N$ input matrix. The two-dimensional DFT is the output matrix $Y(p,q); 0 \leq p, q < N$. $Y(p,q)$ is defined as

$$Y(p,q) = \sum_{r=0}^{N-1} \sum_{s=0}^{N-1} x(r,s) W_N^{pr} W_N^{qs};$$

where $0 \leq ij < N$ is a symmetric $N \times N$ matrix.

3. PARALLEL ARCHITECTURES

Systolic and Wavefront Arrays are the popular parallel VLSI array architectures with massive concurrency derived from pipeline...
processing or parallel processing or both. Extensive literature on systolic/wavefront array processing is presented in [7]. The burden of synchronising an entire systolic computing network becomes heavy for very large arrays. A simple solution is to take advantage of the data flow computing principle which is natural to signal processing algorithms and which leads the designer to wavefront array processing [8],[9],[10].

One of the important features that distinguish a wavefront array processor from other processors is the data driven operation of each of its processing elements (PEs). To ensure correct sequencing and data transfers between adjacent PEs, handshaking protocols must be adopted to synchronize the operations. Completion signal generation is a necessary requirement in this approach. Completion signal generation circuits can be implemented using differential cascade voltage switch (DCVSL) logic [11] or a replica delay module or using current sensing. The handshaking can be done using two-phase or four-phase signaling protocols.

An essential component of virtually any handshaking module is the muller-C element. Figure(1) shows how to use this element to enforce the two-phase handshaking protocol for the example of sender-receiver. Assume that Request(req), Acknowledge(ack) and Data Ready(DR) are initially 0. When the sender wants to transmit the next data, the DR signal is set to 1, which triggers the C-element. The req line goes high and data is sent to the receiver. Subsequently the data is processed in the receiver and ack goes high blocking the C-element, no new data is sent to the data bus as long as the transmitted data is not processed by the receiver.

A wave front array processor for matrix multiplication is suggested by S.Y.Kung. Sayfe [12] has suggested an Independent Data Flow Wavefront Array Processor architecture for Kalman algorithm. In this paper similar attempt has been made to design a wavefront array architecture for 2D-DFT.

4. VLSI IMPLEMENTATION

A wavefront array based architecture for implementing the 2D-DFT is shown in Figure(2) for size of the array N=4. The internal structure of PE is shown in Figure(3). The architecture contains 16 PEs in the mesh array. The processors transmit data between them through the two-phase handshaking protocol explained in Figure(1). The architecture performs the matrix multiplication of equation(3) in two phases namely phase 1 and phase 2. Here W and X are 4x4 matrices. The output Y is also a 4x4 matrix. In phase 1, the multiplication WxX is performed and the result is stored in the register Rij. Initially all acknowledge lines(ack) are low and the input data enter the PEs along each row and column through handshaking. The first wave of input data [x(0,0),x(0,1),

Figure(1) The Two-Phase handshake protocol
To begin with, in the pipeline all processors $[P_{10}, P_{11}, P_{12}, P_{13}]$ perform the multiplication $(W_{ij}X_{in})$. Now $P_{10}$ performs the addition with the horizontal input data and sends the output $H_{out}$ to the successive processor $P_{11}$ through handshaking. Similarly $P_{11}$ performs addition and sends the result to $P_{12}$ and so on. The results of 2D-DFT are obtained at the outputs of $P_{13}$ successively. The completion signal generation circuit generates a data ready (high) to the succeeding processor and an ack (low) to its own processor after completing the addition operation.

The Tables (2) & (3) show a detailed operation of PEs during each wavefront step. The timing diagram is shown in Figure(4). The total time required to compute one row of the output 2D-DFT matrix is given as $N(M+A)+(N(M+A)+(N-1)A=2N(M+A)+(N-1)A$.

Assuming $(M+A)$ takes one clock cycle and addition takes $1/\alpha$ times the clock the total time required is $2N+(N-1)/\alpha$. The systolic array architecture suggested by Swartzlander [4], [5] computes one row of the 2D-DFT output matrix in $[2N+(N-1)]\approx[3N-1]$ clock cycles, where one clock cycle is the time required to perform one multiply-add operation. Thus,

$$3N-1$$

The Speed up factor $S = \frac{2N+(N-1)/\alpha}{2N+(N-1)/\alpha}$

Efficiency $= S/N$

Moreover the time required to compute the entire 2D_DFT is $(4N-1)$ in the systolic architecture whereas it is $2N+(N-1)/\alpha$ in our proposed architecture.
A wavefront array architecture for 2D-DFT is proposed. This is a novel architecture as it is data driven and can be used to compute very large length transforms. The wavefront array is superior to the systolic array in the sense that (i) it needs no synchronization around very large arrays and (ii) it maximizes the pipelining efficiency and offers a speed achievable by multi-rate systolic arrays. Programming is simple as it is only the description of activities. The wavefront model incurs a fixed time delay overhead due to the handshaking process.
But the clock skew involved in systolic arrays changes the time delay dramatically as the size of the array \(N\) increases. Further work is being carried out to validate the proposed model with simulation studies.

## 6. References


### Table 2 Phase 1 Operation

<table>
<thead>
<tr>
<th>Wavefront Steps / PE</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_{00})</td>
<td>(R_{00} = R_{00} \cdot (W^{0} \cdot X(0,0))</td>
<td>(R_{00} = R_{00} \cdot (W^{0} \cdot X(1,0))</td>
<td>(R_{00} = R_{00} \cdot (W^{0} \cdot X(2,0))</td>
<td>(R_{00} = R_{00} \cdot (W^{0} \cdot X(3,0))</td>
</tr>
<tr>
<td>(P_{01})</td>
<td>(R_{01} = R_{01} \cdot (W^{0} \cdot X(0,1))</td>
<td>(R_{01} = R_{01} \cdot (W^{0} \cdot X(1,1))</td>
<td>(R_{01} = R_{01} \cdot (W^{0} \cdot X(2,1))</td>
<td>(R_{01} = R_{01} \cdot (W^{0} \cdot X(3,1))</td>
</tr>
<tr>
<td>(P_{02})</td>
<td>(R_{02} = R_{02} \cdot (W^{0} \cdot X(0,2))</td>
<td>(R_{02} = R_{02} \cdot (W^{0} \cdot X(1,2))</td>
<td>(R_{02} = R_{02} \cdot (W^{0} \cdot X(2,2))</td>
<td>(R_{02} = R_{02} \cdot (W^{0} \cdot X(3,2))</td>
</tr>
<tr>
<td>(P_{03})</td>
<td>(R_{03} = R_{03} \cdot (W^{0} \cdot X(0,3))</td>
<td>(R_{03} = R_{03} \cdot (W^{0} \cdot X(1,3))</td>
<td>(R_{03} = R_{03} \cdot (W^{0} \cdot X(2,3))</td>
<td>(R_{03} = R_{03} \cdot (W^{0} \cdot X(3,3))</td>
</tr>
</tbody>
</table>

### Table 3 Phase 2 Operation

<table>
<thead>
<tr>
<th>Wavefront Steps / PE</th>
<th>V</th>
<th>VI</th>
<th>VII</th>
<th>VIII</th>
<th>IX</th>
<th>X</th>
<th>XI</th>
<th>XII</th>
<th>XIII</th>
<th>XIV</th>
<th>XV</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_{00})</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td>(H_{00} = (W^{0} \cdot X(R_{00}) )</td>
<td></td>
</tr>
<tr>
<td>(P_{01})</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td>(H_{01} = (W^{0} \cdot X(R_{01}) )</td>
<td></td>
</tr>
<tr>
<td>(P_{02})</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td>(H_{02} = (W^{0} \cdot X(R_{02}) )</td>
<td></td>
</tr>
<tr>
<td>(P_{03})</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td>(H_{03} = (W^{0} \cdot X(R_{03}) )</td>
<td></td>
</tr>
</tbody>
</table>