THE DESIGN OF A VECTOR-RADIX 2DFFT CHIP

Wentai Liu* and J. C. Duh
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC. 27695-7911

Daniel E. Atkins
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI. 48109

ABSTRACT

Architectures based on the vector-radix 2DFFT algorithm and hence can avoid the matrix transpose problem have been proposed. The unique feature of the proposed architectures is that the data can be driven into the arithmetic processors in a pipeline fashion. This paper presents a propotype chip, which has been designed in 2 μm NMOS technology, for the generalized butterfly unit. The chip is a two-stage pipelined processor. The design experience, timing information, and the chip features including four multipliers, one adder/subtracter and PLA controllers are presented.

1. INTRODUCTION

The two dimensional **Discrete Fourier Transform** is defined as follows:

$$X(k_1,k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x(n_1,n_2) W_N^{n_1k_1} W_N^{n_2k_2} \qquad (1.1)$$

This transform is generally evaluated either by the classical row-column algorithm [1] or by the "vector radix algorithm" [2, 3].

Many applications of digital signal processing require the evaluation of discrete Fourier transforms (DFT's) of multidimensional sequences. The case of two-dimensional discrete Fourier transform has been widely applied in the area of filtering, image enhancement, image coding, image compression and restoration, radar detection, and computerized tomography, nuclear magnetic resonance (NMR) tomography and seismic analysis [4].

Since the appearance of the original Cooley-Tukey algorithm in 1965 [5], the standard methods of computing the two dimensional discrete Fourier transform have been governed by the **separability** of two dimensional DFT. Two dimensional Fourier transform has been decomposed into two one dimensional DFT by using the one dimensional fast Fourier Transform (FFT) to execute the transform either in row-column-wise or in column-row-wise format. The case of one dimensional FFT has been carefully studied and implemented in either software or hardware since 1965 [6, 7].

Matrix transposition and high I/O bandwidth are two major problems associated with the use of a row-column (or column-row) algorithm for the two dimensional DFT. Therefore techniques to efficiently store the data in the secondary storage device such that it can avoid the matrix transposition or minimize the traffic between the main memory and secondary memory are very crucial in the execution of the FFT.

In many applications the matrix is stored on a mass storage device, *e.g.* a disk or a tape, where the smallest record that can be easily accessed is either an entire row or column. The way in which the data is stored facilitates data access for one dimension but impedes data access of another dimension. The decomposition approach severely degrades the performance of the implementation of a two dimensional FFT either in conventional machines (especially a virtual memory machine) or SIMD machines. The efficient matrix transposition algorithm as shown in [8] is a solution for this problem. However as pointed out by Moorhead [9], matrix transposition is still a serious bottleneck even for most super vector machines in performing vector gathering operations. One way to ease the matrix transposition problem is to employ a **staging memory** proposed by Batcher [10] between the memory and the processor. The staging memory can reformat or corner-turn the array of data.

The naive or "direct" algorithm for computing the one dimensional DFT in Equation 1.1 required N^2 multiplications. However the 1-D FFT is an algorithm to reduce the requirement to (N log N) multiplications. Algorithms for the 1-D FFT can be classified into full parallel, iterative parallel, cascade, and scalar structure. Many different architectures for the 1-D FFT have been proposed or implemented. For example, discrete implementations for the 1-D FFT have been mentioned in [6]. Despain et al. use the CORDIC technique and VLSI technology to implement an 1-D FFT processor in a cascade structure [11]. Computational tasks involving two dimensional discrete Fourier transforms are computationally intensive and need very high I/O bandwidth. Although a two dimensional DFT has wide applications and has been proposed to map row-column-wise decomposition into a SIMD machine [12], so far there doesn't exist a highly efficient architecture for it simply because of the inherent complicated data communication patterns in the row-column decomposition approach. This has motivated us to search for new algorithms and to propose highly parallel and pipelined architectures for the two dimensional FFT [13, 14]. The architectures fully utilize the ideas of **pipelining** and **parallelism** which are the important characteristics in VLSI design. This paper reportes an attempt to implement the 2-D FFT in VLSI other than simply using the 1-D FFT algorithm repeatedly.

2. NEW ALGORITHM AND ARCHITECTURES FOR 2DFFT

2.1. Vector Radix Algorithm

In this section, we briefly derive the vector radix algorithm for two dimensional FFT. This class of algorithms is superior to the decomposition approach in the merits of both the number of multiplication as well as the error performance due to finite word length. For detail derivation, we refer the readers to the paper [14].

The two dimensional **Discrete Fourier Transform** is defined as follows:

* This work is partially supported by a Faculty and Professional Development Fund at North Carolina State University

$$X(k_1,k_2) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} x(n_1,n_2)W_N^{n_1k_1}W_N^{n_2k_2} \qquad (2.1)$$

The matrix $[x(n_1,n_2)]$ can be decomposed into four submatrices $[a(n_1,n_2)]$, $[b(n_1,n_2)]$, $[c(n_1,n_2)]$, and $[d(n_1,n_2)]$. i.e.

$$[x(n_1,n_2)] = \begin{matrix} a(n_1,n_2) & b(n_1,n_2) \\ c(n_1,n_2) & d(n_1,n_2) \end{matrix} \qquad (2.2)$$

Thus we have

$$X(k_1,k_2) = \sum_{n_1=0}^{\frac{N}{2}-1}\sum_{n_2=0}^{\frac{N}{2}-1} [a(n_1,n_2)+b(n_1,n_2)e^{-j\pi k_2}+c(n_1,n_2)e^{-j\pi k_1}$$

$$+d(n_1,n_2)e^{-j\pi(k_1+k_2)}]W_N^{n_1k_1}W_N^{n_2k_2} \qquad (2.3)$$

The transformed space is divided into four submatrices, namely $X[2k_1,2k_2]$, $X[2k_1,2k_2+1]$, $X[2k_1+1,2k_2]$, and $X[2k_1+1,2k_2+1]$ and the transformations are performed on the four submatrices as follows:

$$A= X(2k_1,2k_2) \qquad \text{(Even-Even)}$$

$$\frac{N}{2}-1\sum_{n_2=0}^{\frac{N}{2}-1} [a(n_1,n_2)+b(n_1,n_2)+c(n_1,n_2)+d(n_1,n_2)]W_{\frac{N}{2}}^{n_1k_1}W_{\frac{N}{2}}^{n_2k_2}$$

$$B= X(2k_1,2k_2+1) \qquad \text{(Even-Odd)}$$

$$\frac{N}{2}-1\sum_{n_2=0}^{\frac{N}{2}-1} \left\{[a(n_1,n_2)-b(n_1,n_2)+c(n_1,n_2)-d(n_1,n_2)]W_N^{n_2}\right\}W_{\frac{N}{2}}^{n_1k_1}W_{\frac{N}{2}}^{n_2k_2}$$

$$C= X(2k_1+1,2k_2) \qquad \text{(Odd-Even)}$$

$$= \sum_{n_1=0}^{\frac{N}{2}-1}\sum_{n_2=0}^{\frac{N}{2}-1} \left\{[a(n_1,n_2)+b(n_1,n_2)-c(n_1,n_2)-d(n_1,n_2)]W_N^{n_1}\right\}W_{\frac{N}{2}}^{n_1k_1}W_{\frac{N}{2}}^{n_2k_2}$$

$$D= X(2k_1+1,2k_2+1) \qquad \text{(Odd-Odd)}$$

$$= \sum_{n_1=0}^{\frac{N}{2}-1}\sum_{n_2=0}^{\frac{N}{2}-1} \left\{[a(n_1,n_2)-b(n_1,n_2)-c(n_1,n_2)+d(n_1,n_2)]W_N^{n_1-n_2}\right\}W_{\frac{N}{2}}^{n_1k_1}W_{\frac{N}{2}}^{n_2k_2}$$

The formulas are recursively applied on the submatrix until we reach the transformation of a 2x2 submatrix.

The vector radix algorithm is an "in place" algorithm. It executes $\frac{N^2}{4}$ generalized butterfly operations in each of logN passes. Each butterfly operation fetches four suitable data inputs from four corners of a window on the data array, does transformation, and sends the results back to the four original window corner positions. The patterns of data positions vary in each pass, and so do the sizes of the windows. Each window provides four data inputs from the positions that have the same label.

The window operations can be carried out in parallel, or in any particular order, with the only real limitation being that all of the input data required for butterfly operations in any one pass must generally be completed prior to initiation of those butterfly operations.

The size of the window varies from $\frac{N}{2}x\frac{N}{2}$, $\frac{N}{2^2}x\frac{N}{2^2}$,..., $\frac{N}{2^i}x\frac{N}{2^i}$,..., to 1x1. (i.e. it is $\frac{N}{2^i}x\frac{N}{2^i}$ in ith pass). Since the size is shrunk simultaneously in two directions, different schemes are used to emulate the shrinking window operations in each direction. In the X-direction, the flow of the data is accomplished by a set of synchronous switches such that the set of generalized butterfly units get the desired data from the data pipes. In the Y-direction, the data flow is accomplished by a perfect shuffle scheme. The cascade of these two schemes exactly emulate the required window operations.

The vector radix algorithm executes $\frac{N^2}{4}$ generalized butterfly operations in each of logN passes. Each butterfly operation fetches four suitable data inputs from four corners of a window on the data array, does transformation, and sends the results back to the four original window corner positions.

2.2. PIPELINED ARCHITECTURES FOR 2DFFT

2.2.1. A VLSI Parallel/Pipelined Architecture

The NxN two dimensional FFT can be advantageously implemented by a highly modular architecture of Figure 1, with a cascade of the stages consisting of a perfect shuffle switch, a synchronous switch and a generalized butterfly unit.

A synchronous switch at the ith pass can be in either upward or downward position. At the upward position, the synchronous switch simply feeds the data into the pipe while the generalized butterfly unit is idle. At the downward position, the synchronous switch feeds the data into the active generalized butterfly unit. Recently, a VLSI implementation of the synchronous switch has been reported by Dr. Swartzlander of TRW [15].

Figure 2 shows the generalized butterfly unit which takes four inputs a, b, c, d, executes the transformations of equations (Even-Even), (Even-Odd), (Odd-Even), (Odd-Odd) in Section 2.1, and then generates four outputs A, B, C, D.

The architecture can be implemented in either a fully parallel pipelined architecture or an iteratively parallel architecture.

Each architecture variation can be implemented with bit parallel arithmetic or bit serial arithmetic in which a CORDIC technique [16] can be applied.

2.2.2. A Serial Pipelined Architecture

The architecture in the previous section, it requires to have three kinds of components, namely perfect shuffle switch, data commutator, generalized butterfly unit, to implement a parallel/pipelined two-dimensional Fast Fourier Transform. It offers enormous speed for NxN 2-D FFT with the cost of great number of hardwares which contain $\frac{N}{2}$ generalized butterfly units in each stage. In the paper [14], we propose another pipelined architecture for 2-D FFT with much less hardware than the one in the previous section. For NxN 2-D FFT, this architecture has logN stages and needs only one generalized butterfly unit in each stage. Each stage, as shown in Figure 3, contains a multiplexer, a demultiplexer, eight delay lines and one generalized butterfly unit. Both the multiplexer and demultiplexer are controlled by two signal lines which are a boolean function of the index sequences in the data flow. The architecture is especially useful for the real-time application with a raster scan as its input device. Through the multiplexer, the data in raster-scan format are folded and fed into four delay lines. With the appropriate delay time enforced in every delay line, the generalized butterfly unit are assured to correctly have four piece of data ready for doing butterfly transformation at every time interval. When the generalized butterfly unit finishes the transformation, it sends out four piece of data to the output delay lines. Again by enforcing appropriate delay time in every output delay line, the demultiplexer is able to rè-assemble the data into a raster-scan format. In summary, the input data from a raster-scan device are able to be continuously fed into the pipelined processor by three principal operations, namely *disassembly, transformation, re-assembly.*

In this architecture, the delay lines can be implemented by either of RAMs or of dual-port memories.

3. NMOS IMPLEMENTATION OF THE GENERALIZED BUTTERFLY UNIT

3.1. Chip Implementation

The NMOS ($\lambda = 2\ \mu m$) implementation of a generalized butterfly unit with four-bit data path is 5400 μm high, 4400 μm wide, contains 6500 transistors, and was fabricated by MOSIS last fall. Figure 4 shows the layout of the design. Due to the pin limitation, we decide to implement the butterfly unit in the way of time multiplexing. Each data is a complex number and hence consists of a real and an imaginary number which is four-bit wide respectively. In order to accelerate the computation, a complex number multiplier is implemented by four real number multipliers.

The chip has been designed to be a two micro-stage pipelined processor as shown in Figure 6. The first stage consists of an adder/subtrater, while the second stage consists of those four multipliers operating in parallel. The carry ripple adder/subtrater has been responsible for the operations, namely a+b+c+d, a-b+c-d, a+b-c-d, and a-b+c+d. It has been implemented to do an operations in five cycles (in our case, each cycle takes 960ns). Each multiplier has been implemented in terms of four carry save adders, as shown in Figure 5, and takes three cycles. Two finite state machines implemented in PLAs (Programmable Logic Arrays) as shown in Figure 7, are used to generate control sequences for controlling the adder/subtrater and the multipliers. It is clear that a micro-stage in the chip has delay time of 5x960ns = 4.8μs. For a generalied butterfly operation, it takes 5x4.8μs = 24.0μs.

For the case of a 1024x1024 two dimensional FFT, we need to have ten macro-stages, as shown in Figure 3, cascading into a pipeline in order to implement a serial pipelined architecture as discussed in section 2.2.2. Each macro-stage include two micro-stages. The total time to execute an 1024x1024 2DFFT is approximately equal to $\frac{1024\text{x}1024}{4}$x24.0$\mu$s = 6.0sec. If we implement the fully parallel/pipelined architecture as discussed in section 2.2.1, we need to have ten macro-stages cascading into a pipeline. Each macro-stage consists of 512 chips. The total time to execute an 1024x1024 2DFFT is approximately equal to 6ms. Both architectures can be executed in a pipelining fashion. It would be highly efficient especially in the applications that the input data could be continuously fed into the architectures.

3.2. Chip Operations

The chip is operated by a two phase non-overlapped clock scheme. Each clock phase has 50% duty cycle with 960ns cycle time. First of all when the signal "LOAD" is activated, the chip latches the twiddle factors W_N^0, W_N^μ, W_N^ν, and $W_N^{\mu+\nu}$ into input registers. Then the chip starts to execute the transformations by activating the signal "START". When the "START" signal is received by FSM1, a reset signal will be generated. This will force FSM2 to generate the signals("INIT" and "ADD/SUB") to control the adder/subtrater such that the operations of a+b+c+d, a-b+c-d, a+b-c-d, and a-b+c+d can be computed sequentially. When each operation has been completed, FSM2 issues a "DONE" signal to latch the result and activates FSM1 to start the multiplications and latches the results into the output registers. It is our intention that the adder/subtrater (the first micro-stage) and the multipliers (the second stage) are operated simultaneously in the pipelining fashion. Then the chip can continuously accept the input data from the previous macro-stage and feeds the results to the next macro-stage.

3.3. Implementation Experience

In this section, we focus on the design experience of the chip on the different level representations, which include functional specification level, register-transfer level, logic design level, mask layout level and circuit level. At the functional specification level, Figure 2 serves as a specification of the generalized butterfly unit. At the register transfer level, functional specification has been translated into a set of multipliers, adder/subtraters, registers, multiplexers and control sequences. The main result at this level is an APL based register-transfer simulation of the structure, as shown in Figure 6, to verify the functional correctness of the implementation.

At the mask layout level, we translate the register-transfer building blocks into the transistors through careful logic designs. With the capability of CALMA design stations at the Department of Electrical and Computer Engineering, North Carolina State University, the VLSI layout becomes more managable and enjoyable. The chip is composed of four kinds of cell elements (excluding I/O pads), namely register, multiplexers, adder, PLA. Among them, PLA has been automatically generated by the PLA generator - PLATT which has taken the output of the multiple output logic minimization program - MINI. With all cell elements designed, the final step is to assemble them. It is not suprised that the most difficult, tedious, and time consuming task is the wire routing process to interconnect the cell elements. A modium of cleverness for macro cell placement and routing program for building blocks design style such as BBL [17] is definitely required to facilitate the custom design in the future.

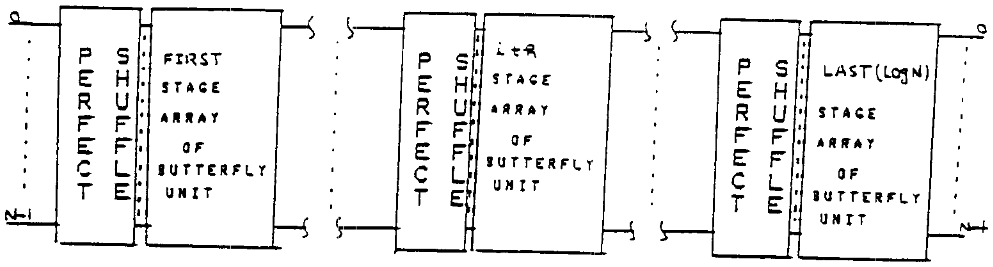
After the layout being finished, we make a conversion of the layout from CALMA's GDSII STREAM format to CIF format. Design Rule Checker, DRC, and Electrical Rule Checker, ERC, have been applied to ensure the geometric and electrical correctness of the layout (in CIF format). Then the logic correctness has been checked by a switch level simulation program - ESIM. The software programs of CRYSTAL and SPICE can provide us the worst case timing analysis. Finally the entire layout has been plotted by a HP7580 plotter. At North Carolina State University, we are gratefu for having accquired the design tools from MCNC, MIT, UCB and the tools developed by the first author in the Department. Those tools has been proved very useful and should be recommended to the VLSI designers.

3.4. Implementation Improvments

Several implementation improvments can be achieved. In order to speed up the multiplication, the carry ripple adder at the final stage of the multiplier can be replaced by a carry look ahead adder. This can reduce the cycle time from 960ns to 160ns even if the technology has not been changed. Instead of implementing the operations by only one adder/subtrater, we can dedicate more chip area to implement the operations (e.g. a+b+c+d etc.) to accelerate the operations dramatically. By the combination of both improving schemes, we are able to reduce the total execution time for an 1024x1024 2DFFT to 0.5sec if the serial archtecture is implemented. If we implementd the parallel/pipelined architecture, it takes approximately 0.5ms. Certainly further improvments can be achieved either by improving technology (e.g. λ=1μm) or by having more compact layout.

4. CONCLUSION

In this paper, we have presented two 2DFFT pipelined architectures, which are very efficient in the realtime applications if the input device is a raster scan device such as that is used in the satellite data transmissions. The unique feature of the proposed architectures is that the data can be driven into the arithmetic processors in a pipeline fashion. we have presented a prototype chip, which has been designed in 2 ' m NMOS technology, for the butterfly unit. By our current chip design, it is possible to have 1024x1024 two-dimensional Fast Fourier Transform in 6 sec by a serial pipelined architecture or in 6 ms by a parallel architecture.



0
N-1
PERFECT
SHUFFLE
FIRST STAGE ARRAY OF BUTTERFLY UNIT
PERFECT
SHUFFLE
i-th STAGE ARRAY OF BUTTERFLY UNIT
PERFECT
SHUFFLE
LAST(LogN) STAGE ARRAY OF BUTTERFLY UNIT
0
N-1

Figure 1 The Parallel/Pipelined Architecture

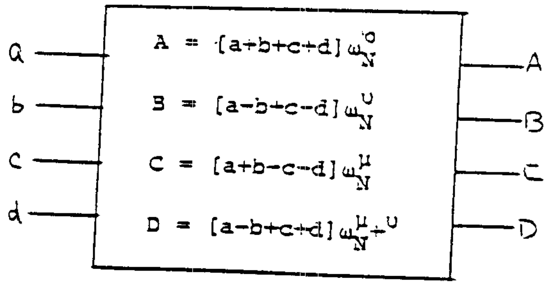


Figure 2 A Generalized Butterfly Unit

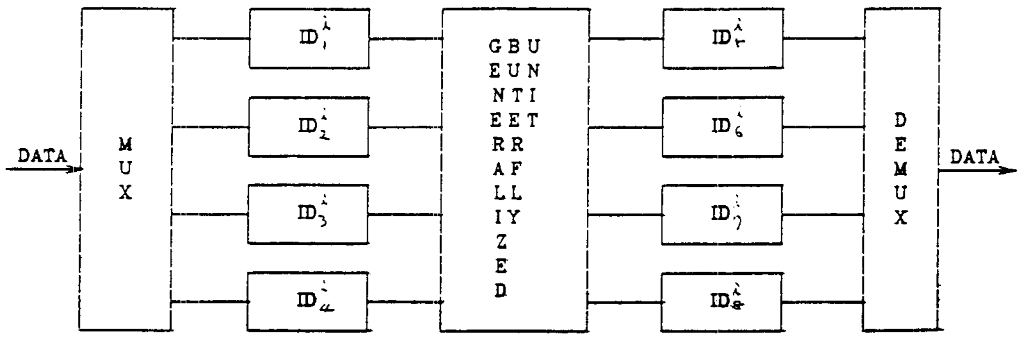


Figure 3 A Serial Pipelined Architectre

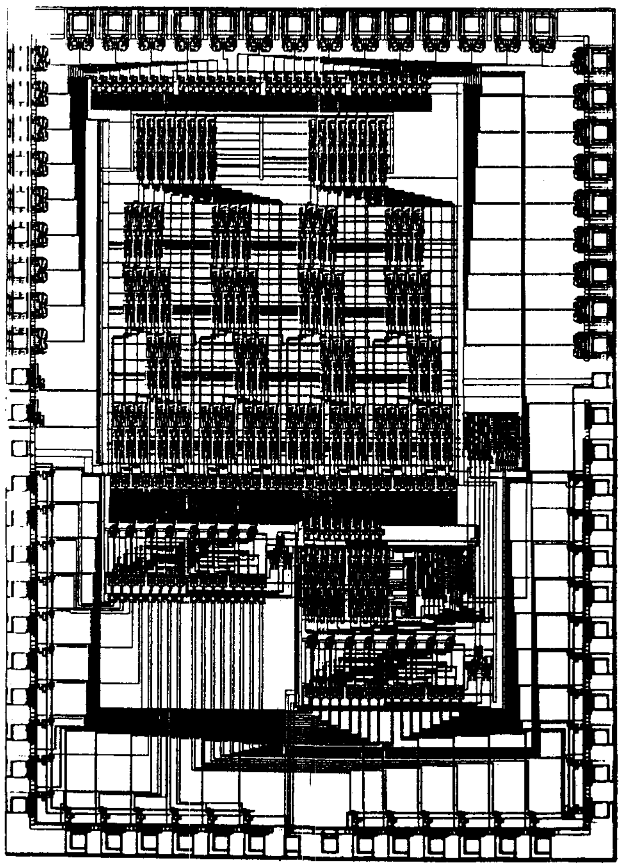


Figure 4 A Checkplot for the Chip

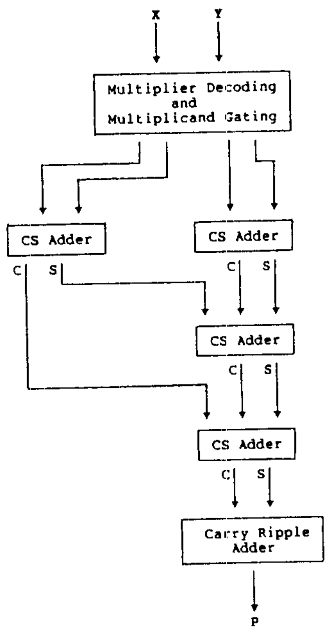


Figure 5 A CSA Implementation of a Multiplier

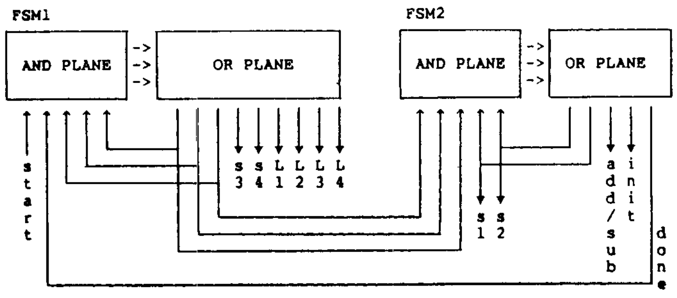


Figure 6 Signals Generated by Finite State Machines

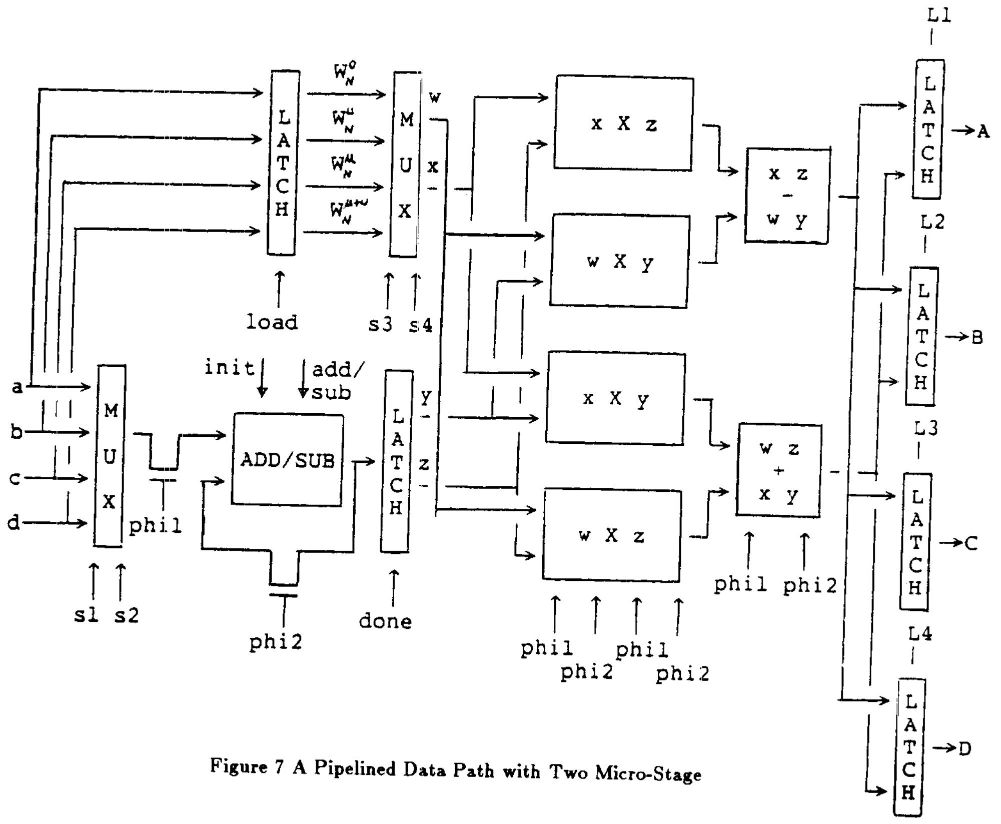


Figure 7 A Pipelined Data Path with Two Micro-Stage

5. REFERENCES

1. D. Dudgeon and R. Mersereau, *Multidimensional Digital Signal Processing,* Prentice-Hall Inc., New Jersey (1984).

2. B. Arambepola, "Fast computation of multi-dimensional discrete fourier transforms," *Proceed. IEE (London)* **127** pp. 49-52 (1980).

3. W. Liu, *Architecture for two dimensional Fast Fourier Transform,* US Patent pending No. 6-539540 (Oct. 5, 1983).

4. R. Gonzalez and P. Wintz, *Digital Image Processing,* Addison-Wesley, Reading, Mass. (1977).

5. J. Cooley and J. Tukey, "An algorithm for the machine computation of complex Fourier series," *Mathematical Computation* **19** pp. 297-301 (April 1965).

6. L. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* Prentice-Hall Inc., New Jersey (1975).

7. W. Liu and D. Atkins, "Towards a cost/performance analysis of application-directed implementations of the FFT and related transforms," SEL 125, System Engineering Laboratory, University of Michigan (Nov. 1978).

8. J. Eklundh, "Efficient matrix transposition," pp. 9-35 in *Two-Dimensional Digital Signal Processing II,* ed. T. Huang,Springer-Verlag (1981).

9. **W. Moorhead,** ***Private Communication*****Nov. 1983.**

10. **K. Batcher, "Architecture of the MPP,"** ***IEEE Computer Architecture for Pattern Analysis and Image Database Management,*** **pp. 170-174 (Oct. 1983).**

11. A. **Despain,** C. Sequin, C. Thompson, E. Wold, and D. Lioupis, "VLSI implementation of digital Fourier transform," UCB/CSD-82/111, EECS, University of California at Berkeley (Nov. 1982).

12. L. J. Siegel, "Image processing on a partitionable SIMD machine," pp. 293-300 in *Languages and Architectures for Image Processing,* ed. M. Duff and S. Levialdi,Academic Press (1981).

13. W. Liu, "A new pipelined/parallel architecture for two dimensional Fast Fourier Transform," *Proceed. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management,* pp. 214-219 (Oct. 1983).

14. W. Liu and D. Atkins, "VLSI pipelined architectures for two dimensional FFT with raster-scan input device," *Int'l Conference on Computer Design: VLSI in Computer (ICCD84),* pp. 370-375 (Oct. 1984).

15. E. Swartzlander Jr. et al., "A VLSI delay commutator for FFT implementation," *Int'l Solid-State Circuit Conference (ISSCC84),* pp. 266-267 (Feb. 1984).

16. J. Valder, "The CORDIC trigonometric computing technique," *IEEE Trans. Elec. Comp.,* **EC-9** pp. 227-231 (Sept. 1960).

17. N. Chen , C. Hsu, E. Kuh, C. Chen, and M Takahashi, "BBL: A building-block layout system for custom chip design," *1983 IEEE Int'l Conference on Computer-Aided Design,* pp. 40-41 (1983).