(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 902 375 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.03.1999 Bulletin 1999/11

(51) Int Cl.[6]: **G06F 17/14**

(21) Application number: **98307395.8**

(22) Date of filing: **11.09.1998**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **12.09.1997 US 58592 P**

(71) Applicants:
- **Sharp Kabushiki Kaisha**
  Osaka 545-8522 (JP)
- **SHARP MICROELECTRONICS TECHNOLOGY, INC.**
  Camas, WA 98607 (US)

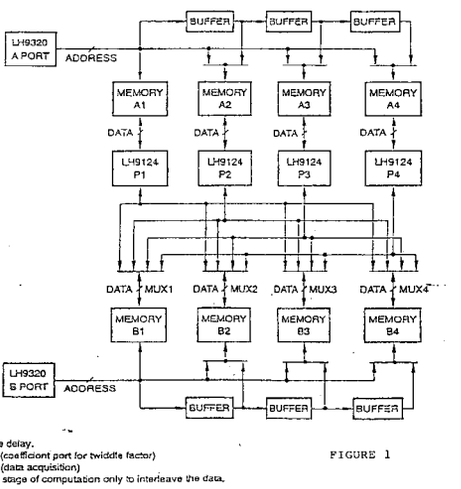
(72) Inventors:
- **Ju, Chwen-Jye**
  Vancouver, WA 98683 (US)
- **Sidman, Steven B.**
  Vancouver, WA 98683 (US)

(74) Representative: **Brown, Kenneth Richard et al**
R.G.C. Jenkins & Co.
26 Caxton Street
London SW1H 0RJ (GB)

(54) Apparatus for fast Fourier transform

(57) A data processing system for use in digital signal processing applications for processing N data points through Y processing stages using Z execution units, each execution unit having a plurality of I/O ports including A and B ports. The data processing system comprises an addressable memory unit for each A and B port on each execution unit, including memory units A(1) through A(Z) operatively connected to the A ports of execution units P(1) through P(Z), respectively, and including memory units B(1) through B(Z) operatively connected to the B ports of execution units P(1) through P(Z), respectively. Data is processed through the Y processing stages by moving data in selected sequences through each execution unit P(q) between memory units A(q) and B(q). The system further includes:

a data router which distributes the N data points between memory units A(1) through A(Z), each memory unit receiving N/Z data points, each execution unit P(1) through P(Z) processing the block of N/Z data points through the Y processing stages, the processing occurring in parallel;
a first address generator AG-A operatively connected to memory units A(1) through A(Z) for supplying the address sequences used in the Y processing stages to memory units A(1) through A(Z);
a second address generator AG-B operatively connected to memory units B(1) through B(Z) for supplying the address sequences used in the Y processing stages to memory units B(1) through B(Z); and
address generator AG-A supplies the same address sequence to all A(1) through A(Z) memory units, and address generator AG-B supplies the same address sequence to all B(1) through B(Z) memory units, whereby the N data points are processed in Z parallel streams through the Z execution units.



NOTES:
1. Buffer = 1 clock time delay.
2. Not shown: C-Port (coefficient port for twiddle factor)
   Q-Port (data acquisition)
3. MUX is used at first stage of computation only to interleave the data.

FIGURE 1

EP 0 902 375 A2

Description

Background and Summary of the Invention

I. INTRODUCTION

[0001] The Sharp LH9124 FFT processor has been widely used in array signal processing applications such as radar, medical imaging, telecommunications, etc. The paper will discuss the design consideration of FFT-based parallel systems under the strict data latency and throughput requirements in high data rate applications.

[0002] With the inherent pipelined structure of FFT algorithms, it is natural to build pipelined systems (such as the Butterfly DSP DSPMAX-V4 and Catalina Research Inc. CRCV IM40/50) to enhance the data throughput for high-end DSP applications. However, data latency can only be improved by the parallel architecture. Data management is usually the performance killer in parallel systems. The lack of algorithms in handling data addressing sequences for multi-dimensional, multichannel, or multi-rate DSP applications also causes the problems in defining FFT-based parallel systems. Therefore, the present invention aims to alleviate such problems. Some novel approaches and algorithms will be proposed for handling data partitioning and management in parallel and multi-dimensional problems. In addition, we will discuss the practical parallel system design problems by using the Sharp LH9320 Address Generator to control the parallel data sequences as an example.

[0003] The invention is focused on controlling the data partitioning and data communication issues for efficient parallel FFT implementation. The beauty of the FFT can be seen from twiddle factor matrix factorization. In Section II, how to partition the matrix to get a good structure for implementation is discussed. It is shown that both 1-D and 2-D can be represented by similar vector-matrix form. In addition, the butterfly computations and the I/O data transfer can be separately represented by matrices. Section III discusses the by-pass form FFT implementation based on the LH9124/LH9320 chip set. In section IV, we will derive the parallel FFT addressing sequences from the normal FFT addressing for a single processor. Section V will introduce two methods of building by-pass form parallel FFT systems. The final conclusions are given in Section VI.

[0004] In general, digital signal processing (DSP) tasks are performed either in the time domain or in the frequency domain. Time domain refers to the situation where data are continuously sampled and, for subsequent digital processing, quantized. Frequency domain refers to the situation where the original time-ordered sequence of data is tansformed into a set of spectra, with specific frequencies and amplitudes. These data are also usually quantized for further processing. For many DSP tasks, dealing with spectra in the frequency domain yields more insight and provides a more tractable way of solving problems than dealing with sequentially ordered data points in the time domain. The issue then becomes how to most efficiently transform time domain data into the frequency domain.

[0005] The most common algorithm used to do this transformation is called the Fast Fourier Transform, or FFT.

[0006] The FFT is obtained by starting with a more general form called the Discrete Fourier Transform, and then subdividing the calculations until one is left with an irreducible kernel operation, most commonly operating on two input data points only. This operation is called a butterfly operation because of the shape of the signal flow graph used to describe it. In the simplest butterfly operation ("radix-2"), the two input data points have their sum and difference multiplied by a coefficient, or "twiddle" factor. The sequence in the order of data for the two data points is then interchanged at the output. This interchange, or crossover gives rise to the term "butterfly" when the calculation is drawn as a signal flow graph.

[0007] Time domain data is transformed into the frequency domain by successively applying the butterfly operation to the entire time domain data set two data points at a time, and repeating this process multiple times, or "passes" using the partially transformed results of the prior pass of calculation. For Radix-2 calculation, the number of required passes of pairwise calculation (butterfly operations) through the data set is equal to the log base radix of the number of data points in the set. For Radix-2 butterfly calculation, the factor log base radix would be the log base 2. For a data set of 1024 points, the log base 2 of 1024 is 10 (2^{10} = 1024), so 10 passes through the data and its intermediately calculated results are required in order to calculate a 1024 point ("1K") FFT. Transforming 1024 time domain data points is then seen to require 10 x 1024 = 10240 Radix-2 butterfly operations.

[0008] Other radices are possible, besides the most basic one of radix two. The physical interpretation is as follows. Radix-2 operates on two numbers at a time, Radix-4 is a higher order butterfly calculation that operates on four numbers at a time, Radix-16 is a still higher order butterfly calculation that operates on 16 numbers at a time, and so forth. The advantage of higher radix computation is that fewer passes through the data are required, because the pass-defining term (log base radix), is a smaller number. For example, a Radix-2 FFT of 256 points requires 8 passes through the data (2^8=256), while a Radix-16 evaluation of the FFT requires only two passes through the data (16^2=256). Therefore, if sufficient computational resource is present to permit evaluation of a Radix-16 butterfly in the equivalent time per clock per data point of a Radix-2 butterfly, the FFT calculation using the Radix-16 butterfly will be faster.

[0009] A general style of FFT calculation is called by-pass form. In this form of calculation, there is a fixed datapath

that performs the butterfly calculation, but the data are supplied through external memory. The datapath performs a fixed set of calculations (the butterfly operation) on data that is presented to it. It is therefore necessary to properly sequence the data in these external memories for appropriate pass by pass FFT calculation by the datapath.

[0010] The LH9124 is an example of a bypass form processor that does FFTs. It has four data ports: Q, A, B, and C. Data are input into the Q or acquisition port. The results of the initial pass of butterfly calculation through the input data are output to port B. On the next pass of butterfly calculation, partially transformed data are read from the memory connected to Port B, processed, and output to the memory connected to Port A. Subsequent passes "ping-pong" the data between ports A and B. Port C is used to input coefficient data.

[0011] The time taken for any calculation may be described either as latency or as throughput. Latency is the time it takes from an initial condition until the first valid output data is obtained. Throughput is the time between subsequent data outputs, once a system is up and running.

[0012] There are three main ways of improving the FFT speed of bypass based systems.

1. Increase the order of the radix, thus decreasing the number of required passes through the data. This approach improves both latency and throughput, but is costly in terms of computational resource required for each processing element.
2. Cascade (pipeline) the datapath processors, such that each processor is responsible for the calculation of one pass of the FFT only, and calculates that pass repetitively on different blocks of data. Cascading improves throughput, but not latency. The very first FFT will still have to be processed by the appropriate number of passes, or cascaded stages. Every FFT after the first one will be output in 1/N the time of the first, where N is the number of stages.
3. Parallel the datapath processors, such that a single large FFT is divided into N smaller FFTs. Each datapath processor is then dedicated to the calculation of an FFT that is 1/N the size of the original, although the number of passes appropriate for the original, larger data set is still required. Both latency and throughput are improved with this arrangement. Latency is improved because there are 1/N the number of original points on which any individual datapath processor has to operate. Throughput is improved for the same reason - there are fewer points for any individual datapath processor.

[0013] Note that cascading and parallel operation can be combined.

[0014] The objects of this invention are:

1. To minimize latency in FFT computation by parallel connection of datapath processors.
2. To simplify system connection by permitting one address sequencer to be used for like data ports of all paralleled datapath processors.
3. To simplify system design by permitting the use of ordinary single-port SRAMs in a parallel datapath connection.

Brief Description of the Drawing

[0015] Fig. 1 is a partial schematic drawing of a digital signal processing system showing a parallel system in accordance with the present invention with four LH9124 execution units using multiplexer and using one addressing sequence.

Detailed Description of the Preferred Embodiment

II. FFT PARTITION AND DERIVATION

[0016] The Discrete Fourier Transform (DFT) of an *N*-point sequence {x(n)} is defined by

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

where $W_N^J = w^{-2\pi j/N}$. Thus, the vector-matrix form of an DFT can be expressed as

$$\underline{X} = TW_N * \underline{x} \qquad (2)$$

when $\underline{X}$ and $\underline{x}$ are N-point vectors and TW_N is an N by N twiddle factor matrix. The twiddle factor matrix itself cannot be recursively decomposed. Actually, the bit-reverse matrix or digit-reverse matrix plays the central role in the DFT to FFT decomposition.

[0017] The bit-reverse input and linear output (BI/LO) FFT algorithm can be obtained by factorizing the twiddle factor matrix post-multiplied by an N by N bit-reverse matrix P_{br} as

$$T_N = TW_N * P_{br} \tag{3}$$

[0018] The matrix T_N can be factorized by the following recursive equation:

$$\begin{aligned} T_N &= \begin{bmatrix} I_{N/2} & A_{N/2} \\ I_{N/2} & -A_{N/2} \end{bmatrix} \begin{bmatrix} T_{N/2} & 0_{N/2} \\ 0_{N/2} & T_{N/2} \end{bmatrix} \\ &= \begin{bmatrix} I_{N/2} & A_{N/2} \\ I_{N/2} & -A_{N/2} \end{bmatrix} \begin{bmatrix} I_{N/4} & A_{N/4} & 0_{N/4} & 0_{N/4} \\ I_{N/4} & -A_{N/4} & 0_{N/4} & 0_{N/4} \\ 0_{N/4} & 0_{N/4} & I_{N/4} & A_{N/4} \\ 0_{N/4} & 0_{N/4} & I_{N/4} & -A_{N/4} \end{bmatrix} \\ &\quad \begin{bmatrix} T_{N/4} & 0_{N/4} & 0_{N/4} & 0_{N/4} \\ 0_{N/4} & T_{N/4} & 0_{N/4} & 0_{N/4} \\ 0_{N/4} & 0_{N/4} & T_{N/4} & 0_{N/4} \\ 0_{N/4} & 0_{N/4} & 0_{N/4} & T_{N/4} \end{bmatrix} \end{aligned} \tag{4}$$

where I_N, A_N, T_N and 0 are N by N matrices. I_N is an identity matrix and 0_N is a zero matrix. The matrix $A_{N/2}$ is a diagonal matrix with

$$A_{N/2} = diag\left(w_N^0 \; w_N^1 \cdots w_N^{N/2-1}\right) \tag{5}$$

[0019] The elements of T_N and $T_{N/2}$ have the following relation.

$$T_{N/2}(i,j) = T_N(i,j) \text{ for } 0 \le i,j < N/2-1 \tag{6}$$

1-D FFT Implementation

[0020] This section will show that the twiddle factor matrix factorization (4) for BI/LO FFT can be further partitioned. Thus, each butterfly stage can be represented by the input interconnection matrix followed by the butterfly computation matrix and the output interconnection matrix. Some essential physical meaning in FFT algorithm implementation can be found through the three-matrix representation of the butterfly stage.

[0021] Inserting the recursive factorization equation (4) into (2) and setting the data length $N = 2^s$, we can represent the DFT by cascaded butterfly stage matrices as follows:

$$\underline{X} = FG_s(BI(s)) * FG_{s-1}(BI(s-1)) * \ldots * FG_1(BI(1)) * P_{br} * \underline{x} \tag{7}$$

where $\underline{X}$ and $\underline{x}$ respectively, denote the N-point linear output vector and N-point linear input vector. $FG_k(BS(k))$ denotes the k-th stage of the FFT algorithm. Each stage of the FFT can be further partitioned into three matrices: the right permutation matrix indicating the input interconnection, the central matrix indicating the butterfly operation, and the left permutation matrix indicating the output interconnection as follows:

$$FG_k(BI(k)) = P_{lk} * BI(k) * P_{rk} \tag{8}$$

P_{rk} specifies the interconnection between inputs and butterfly modules and is a permutation matrix. P_{lk} specifies the interconnection between butterfly modules and outputs and is also a permutation matrix. If the in-place algorithm is employed, the left permutation matrix P_{lk} is the transpose of the right permutation matrix P_{rk}BI(k) is the k-th stage butterfly operation matrix and it is a block diagonal matrix.

2-D FFT Implementation

[0022] Signal Flow Graph (SFG) is usually used to represent the FFT algorithms. In the above description, we showed that the multi-dimensional (M-D) FFT can be mapped into 1-D SFG. All the M-D FFT algorithms can also be represented by vector-matrix form. It is derived that all the FFT algorithms with the same number of data regardless of dimensions can have the same permutation matrices. This means they share the same SFG structure. In addition, the butterfly permutation (interconnection) matrices for the k-th stage of 1-D to M-D FFTs are the same. These results imply that the addressing for the 1-D FFT can be employed to that for the M-D FFT. The only difference between 1-D and M-D FFT is the twiddle factor coefficients in SFG.

[0023] Given a 2-D array with N_1XN_2 = N, in the row-column approach, we may compute N_2 1-D row FFT (*rfft*) and N_1 1-D column FFT *(cfft).* It can be derived that the N_2 *rffts* can be implemented by the first s_1 stages of the *N*-point 1-D FFT and the N_1 *cffts* by the first s_2 stages of the *N*-point 1-D FFT, where $N_1 = 2^{s1}$ and $N_2 = 2^{s2}$. If the *rfft*s and *cfft*s are implemented by (7) and the transpose matrix was combined with the *cfft*s, we can get the vector-matrix form for 2-D FFT as follows:

$$\begin{aligned} \underline{X}_r^T = {} & FG_{s1+s2}(BI(s_2)) * FG_{s1+s2-1}(BI(s_2-1)) * \cdots \\ & * FG_{s1+1}(BI(1)) * FG_{s1}(BI(s_1)) \\ & * \cdots * FG_1(BI(1)) * \underline{x}_{br}^T \end{aligned} \tag{9}$$

where $\underline{X}^T_r$ is an *N*-point vector in the row-major order of the linear output array and $\underline{x}^T_{br}$ is an N-point vector in the row-major order of the bit-reverse input array. Comparing (9) with (7), the N_1 by N_2 2-D FFT has the same inter-connection structure as the *N*-point 1-D FFT. Moreover, the butterfly operation matrices are the same for the k-th stage of the row FFT, column FFT, and 1-D FFT.

[0024] In practical implementation, the unified vector-matrix representation for 1-D to M-D FFT implies that the 1-D addressing sequences, including data, twiddle factor, and bit-reverse, can be employed to all the FFTs regardless of its dimension. Therefore, the parallel FFT systems designed for 1-D FFT can be automatically implemented to FFTs with higher dimension.

III. FFT IMPLEMENTED BY BY-PASS FORM

[0025] In the by-pass form FFT implementation, there is no on-chip RAM and FFT butterfly modules are built in the data path. The by-pass form implementation is more straightforward mapped from the signal-flow or data-flow of FFT structures. Sharp LH9124/LH9320 FFT processors are a typical by-pass form structure to implement the FFT signal-flow. In the following, we will the use Sharp LH9124/LH9320 FFT chip set to discuss the parallel FFT implementation to reduce the latency.

[0026] Section II shows that both 1-D and 2-D FFT can be represented by the vector-matrix form. Each butterfly operation matrix can be implemented by one LH9124 instruction. In addition, each input or output permutation matrix can be implemented by one LH9320 instruction. Therefore, only a few instructions are required to run these applications based upon the proposed algorithms. Thus, system and application design time cost can be greatly reduced by the by-pass form implementation.

LH9124 Execution Unit

[0027] The LH9124 is an execution unit with four I/O ports: Q, A, B, and C. Each port has 48 bits (24-bit real bus, 24-bit imaginary bus) for transferring complex data and twiddle factor coefficients. It has a built-in radix-4 butterfly data path to implement the butterfly operation matrix BI(k) as shown in Eq. (8). There is no limitation on the size of FFT implementation as long as the right data and twiddle factor sequences are provided. In the LH9124, each butterfly operation matrix is mapped and implemented by one LH9124 instruction. For example, if the 4 k-point FFT is imple-

mented by three radix-16 butterfly stages, only three instructions are required to compute it.

[0028] Unlike general-purpose DSPs, the 9124 does not have to consider data partitioning for large size of FFT computations using a single processor. The FFT will be computed one butterfly-stage by one butterfly-stage. From Eqs. (7) and (8), we can see that the FFT is inherently a pipelined structure. The advantage of the by-pass form FFT processors is that parallel pipelined architecture scan be easily built to enhance the input data throughput. If the ping-pong approach is used to data memories and the number of processors is equal to that of butterfly stages N, the throughput will be improved by a factor of N. For example, Catalina Research CRCV1M40/50 can have one to six cascaded LH9124 stages. For the 4 K-point FFT implemented by 3 radix-16 butterfly stages, the throughput can be enhanced by a factor of 3 with a three-deep pipelined LH9124 architecture.

LH9320 Memory Management Unit

[0029] The LH9320 memory management unit provides the required input and output data address sequences for the LH9124 execution unit From Eq. (8), the permutation matrices P_{lk} and P_{rk} are mapped into and implemented by the instruction set of the LH9320. Each instruction will generate an address sequence and the size of the sequence can be defined by the parameter N that will be equivalent to the size of FFT.

[0030] The LH9320 provides the data sequences for mixed-radix FFT operations. Since in-place algorithms are used, the input and output data sequences will be the same and the permutation matrix P_{lk} will be the transpose of P_{rk}. For radix-2 FFT, the algorithm for generating the data sequence of i-th stage will be

for (k = 0; k <2^i; k++)
  for (j = 0; k < $N/2^i$; j++)
    Output {j * 2^i + k} For the radix-4 FFT, the algorithm for the data sequence of i-th stage will be

for (k = 0; k < 4^i; k++)
  for (j = 0; k < $\frac{N}{4^i}$; j++)
    Output {j*4^i+k} For the radix-16 FFI, the algorithm for the data sequence of i-th stage will be

for (k = 0; k <16^i; k++)
  for (j=0; k < $\frac{N}{16^i}$;j++)
    Output {j * 16^i + k}

IV.FFT PARALLEL PROCESSING

[0031] From Eqs. (7) and (8), we can see that the FFT algorithm is very suited to be implemented by the by-pass form pipelined architecture. The output data of the current butterfly stage can be directly used as the input data for the next stage. Data throughput can be improved by pipelined or parallel architectures. However, the latency of FFT computations cannot be enhanced by pipelined structures. Parallel architecture is the only way to enhance the latency. It is not obviously shown from Eqs. (7) and (8) how to directly get a parallel structure to enhance the latency. The data is differently shuffled from stage to stage. Thus, the data scramble has to be taken care of if the FFT is to be implemented by the parallel architecture.

[0032] In this section, we will discuss how to use the LH9320 to design the parallel architecture for FFT computations to reduce the latency. Data partitioning will be the key issue for efficient FFT parallel processing. It is derived that the butterfly computation matrix BI(k) is a block diagonal matrix. In addition, the input and output permutation matrices P_{lk} and P_{rk} have the structure of repeated regular patterns. Thus, if we carefully partition the data in each stage, we can get parallel computing structures. From the LH9320 instruction set, we can also partition the data address sequence into parallel data address sequences. Assume we have a parallel FFT architecture with LH9124s where P is a power of 2. We may derive P parallel addressing sequences. For the radix-2 FFT, the parallel addressing sequences for the i-th stage can be derived from the i-th stage LH9320 radix-2 instruction BF2i as follows:

for (k = 0; k < $2^{i-\log_2 P}$; k++)
  for (j = 0; k < $N/2^i$; j++)
    Parallel Outputs:
    P1: {j*2^i + k * P}
    P2: {j*2^i + (k*P+1)}
      ...
    PP: {j*2^i + (k*P+P-1)}

[0033] For the radix-4 FFT, the parallel addressing sequences for the i-th stage can be derived from the i-th stage LH9320 radix-4 instruction BF4i as follows:

for (k = 0; k <$4^{i-\log_4 P}$; k++)
  for (j = 0; k < $N/4^i$; j++)
    Parallel Outputs:

P1: {j * 4^i + k * P}
P2: {j * 4^i + (k * P + 1)}
PP: {j * 4^i + (k * P + P - 1)}

[0034] For the radix-16 FFT, the parallel addressing sequences for the i-th stage can be derived from the i-th stage LH9320 radix-16 instruction BF16i as follows:

for (k = 0; k < $16^{i-\log_{16}P}$; k++)
for (j = 0; k < N/16^i; j++)
Parallel Outputs:
P1: {j * 16^i + k * P}
P2: {j * 16^i + (k * P + 1)}
...
PP: (j * 16^i + (k * P + P - 1)}

[0035] Similarly, we can derive parallel digit-reverse addressing sequences from the LH9320 digit-reverse sequence instruction. Assume we have a digit-reverse sequence for the FFT with M stages as
[$n_1 n_2$ ... $n_{M-1} n_M$]
if n_M has radix-P, we can partition the above sequence into P parallel sub-sequences as follows:
P1: [$n_1 n_2$ ... $n_{M-1} n_M$]X P
P2: [$n_1 n_2$ ... $n_{M-1} n_M$]X P + 1
PP: [$n_1 n_2$ ... $n_{M-1} n_M$]X P + (P - 1)

V.PARALLEL FFT IMPLEMENTATION

[0036] Until recently, there has been relatively little work in parallel FFT implementation, especially in by-pass form structures. The advantage of by-pass form is that it is easy to meet the strict requirements of high throughput and low latency by building parallel and pipelined system architectures. This section will discuss how to use multiplexer or dual-port memory to build FFT parallel systems with the LH9124/LH9320 chip set. The essential issue is how to place the data in the right place and right time for execution.

Parallel LH9124s With Multiplexer

[0037] This sub-section will discuss how to use multiplexer to control the data communication required for parallel FFT computations. To simplify the discussion, the number of parallel processors is set to be 4 and the size of the FFT is 4K. The system configuration with 4 LH9124s is shown in Figure 1. The local memory shown in the figure for each processor in each port stores 1k points of complex data. In the beginning, input data are stored in A port memories with 1st data in Al, 2nd data in A2, and so on.

[0038] From Section IV by setting P = 4, we can get digit-reverse sub-sequences from the LH9320 digit-reverse instruction. We can use just one LH9320 to control the four A-port local memories because the input addressing sequence is the same for all the processors. We have to arrange the data stored in the B-port memories such that each processor in the later butterfly stages only uses its own local memories. In addition, the parallel instruction for radix-2, radix-4, or radix-16 butterfly discussed in Section IV can be employed. Therefore, the multiplexers are put at the B-ports of LH9124s and they will be used in the first stage butterfly operations. The multiplexers are scheduled such that the (4i + k)-th data in the output of each processor will be loaded in the k-th local memory, where i = 0, 1, 2, ..., 1023, and k = 0, 1, 2, 3. Figure 1 shows some buffers connected to LH9320s. These buffers are used to control the interleave of the data sequences.

[0039] Table 1 shows the instructions used by LH9124 and LH9320 for executing the 4K-point FFT by 4 LH9124s. Referring to the instructions used by LH9230 in Table 1, we can see that if the single processor uses the instruction BF4i, then the four parallel processors will use the instructions BF4(i-1) as derived in Section VII. For 4 parallel LH9124s, each processor will compute 1K points of data instead of 4K points of data in each stage. Therefore, the latency for 4 parallel processors will be enhanced by a factor of 4 when compared with that of single processor. At 50 MHz system clock, the 4K-point FFT can be finished in 250 μs by a single LH9124 and in 66 μs by 4 parallel LH9124s.

Table 1.

Instructions for running FFT by 4 LH9124s with multiplexer structure			
	STAGE 1	STAGE 2	STAGE 3
LH9124 Data Flow	BFLY16 RAWB	BFLY16 RBWA	BFLY16 RAWB
LH9320 A	RBF0	RF41	BF43

Table 1. (continued)

Instructions for running FFT by 4 LH9124s with multiplexer structure			
	STAGE 1	STAGE 2	STAGE 3
LH9320 B	BF44	BF41	BF43

[0040] In the description herein the terms "execution unit," "processor," "LH9124," "data path processor," and LH9124 FFT processor" are used interchangeably. The terms refer to computation units for performing digital signal processing (DSP). Digital signal processors formed on integrated circuit chips are referred to as DSP chips. The terms "address generator" and LH9320" are also used interchangeably. Memory units A(1) through A(4) are shown in Fig. 1 as boxes "MEMORY A1" through "MEMORY A4." Memory units B(1) through B(4) are shown in Fig. 1 as boxes "MEMORY B1" through "MEMORY B4." Execution units P(1) through P(4) are shown in Fig. 1 as boxes "LH9124 P1" through "LH9124 P4." And address generators AG-A and AG-B are shown in Fig. 1 as boxes "LH9320 A PORT" and "LH9320 B PORT," respectively.

[0041] What follows are examples of how parallel DSP operations are carried out in accordance with the present invention using the processor configuration shown in Fig. 1. In the examples, each address generator can be used to supply addresses to four memories, each of which is connected to a port, for example the A or the B port, on one of the four parallel execution units. The example processes 1024 data points using four execution units to carry out up to five stages of operations in four parallel paths, each execution unit handling 256 data points.

[0042] One form of parallel system implementation is given below. For the purpose of this discussion, the desired transform size is 1K, and the number of parallel datapath processors is set to 4. The system configuration is shown in the accompanying figure. There are two memory ports, A and B, to consider.

[0043] First we discuss how the 1K-point FFT can be implemented by a single LH9124. Then we describe how a 1K FFT can be implemented by four parallel LH9124s. Please refer to Figure 1 of the paper about the system diagram.

[0044] We may view the LH9124 as a black box with 4 I/O ports (A, B, C, and Q). C port is confined to be used for FFT twiddle factor coefficients. The data can be input from any of three data ports (A, B, and Q) and output to any of the remaining two ports.

Overview of operation, single vs parallel datapath processors

[0045] This example is for a 1K FFT.

Single Datapath Processor Overview

[0046] For a single processor, the data sequence in each stage will be:

Stage 0: 0, 1, 2, 3,... , 1021, 1022, 1023

Stage 1: 0, 4, 8, 12, ..., 1012, 1016, 1020
1, 5, 9, 13, ..., 1013, 1017, 1021
2, 6, 10, 14, ..., 1014, 1018, 1022
3, 7, 11, 15, ..., 1015, 1019, 1023

Stage 2: 0, 16, 32, 48, ..., 976, 992, 1008
1, 17, 33, 49, ..., 977, 993, 1009
2, 18, 34, 50, ..., 978, 994, 1010
14, 30, 46, 62, ..., 990, 1006, 1022
15, 31, 47, 63, ..., 991, 1007, 1023

Stage 3: 0, 64, 128, 192, ..., 832, 896, 960
1, 65, 129, 193, ..., 833, 897, 961
2, 66, 130, 194, ..., 834, 898, 962
62, 126, 190, 254, ..., 894, 958, 1022
63, 127, 191, 255, ..., 895, 959, 1023

Stage 4: 0, 256, 512, 768
1,257,513,769

2,258, 514, 770
...
254, 510, 766, 1022
255, 511, 767, 1023

Parallel Datapath Implementation Overview

[0047] For parallel system implementation with 4 LH9124s we may store the complex data in the memory banks in the locations from 0 to 255 as follows:

Bank 1:0, 1, 2, 3, ..., 253, 254, 255
Bank 2: 256, 257, 258, 259,..., 549, 550, 551
Bank 3: 552, 553, 554, 555, ..., 765, 766, 767
Bank 4: 768, 769, 770, 771,..., 1021, 1022, 1023

[0048] Thus, the i-th location in bank 1 will be the i-th data in bank 2 will be (i+256)-th data, in bank 3 will be (i+512)-th data, and in bank 4 will be (i+768)-th data.

[0049] From the stage 4 discussed in 9, each bank has to provide one data for each radix-4 butterfly module computations. In addition, for the first module computations with data. 0, 256, 512, and 768, we can see the data must from the first location of each bank. Since the LH9124 only receive one data in each cycle, the data must come in the memory bank interleave. Similarly, for any other module computations, the data are also from the same location of each bank and come in the interleave way.

[0050] The architecture shown in Figure 1 of the paper is an implemention of the idea discussed in the paragraphs above. The buffers shown in the figure are to implement the memory bank interleave. Note that ordinary single-port SRAMs are used. It can be seen that only one address generator is required because the data for each module computations are from the same location in each bank.

Detailed Explanation, 1K FFT, One Datapath Processor (9124)

[0051] In this example, we will use the radix-4 butterfly instruction to implement the 1K-point FFT. Thus, we will have 5 passes of radix-4 butterfly calculation. The LH9124 can be seen as a radix-4 butterfly execution engine. Each radix-4 butterfly computation needs 4 data points. Therefore, there will be 256 (1024/4) radix-4 butterfly computations for each pass. The engine will perform the radix-4 butterfly operations without interruption until the last data of that pass. The data flow instruction is to define the input and output ports for each pass.

[0052] We assume the data are stored in the B port in the sequence as b(0), b(1), b(2), b(3), ... , b(1021), b(1022), b(1023)

Stage 0:
Input Port: B
output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
B Port LH9320 Addressing Instruction: BF40 (Assumed data in digital-reverse sequence)
A Port LH9320 Addressing Instruction: BF40
(Input and Output can use the same addressing instruction because in-place FFT algorithm is employed)

[0053] The addressing sequence will be
0, 1, 2, 3, 4, 5, ..., 1019, 1020, 1021, 1022, 1023

[0054] The 1st radix-4 butterfly execution will use the first four input data 0, 1, 2, 3, the 2nd butterfly will use the following four data 4, 5, 6, 7, and so on. We can collect the output results after 18 cycles from the first input data entering the LH9124. The first 4 output data will be the execution result of the 1st radix-4 butterfly and the following 4 data will be the execution result of the 2nd radix-4 butterfly. The output data will be stored in the A-port memory with an address sequence as 0, 1, 2, 3, 4, ... , 1021, 1022, 1023

Stage 1:
Input Port: A (The input port will be the output port of previous stage or pass of calculation)
Output Port: B
Data Flow Instruction: RAWB

LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF41
B Port LH9320 Addressing Instruction: BF41
(Input and Output can use the same addressing instruction because in-place FFT algorithm is employed)
The addressing sequence will be
0, 4, 8, 12, ... , 1012, 1016, 1020
1, 5, 9, 13, ... , 1013, 1017, 1021
2, 6, 10, 14, ... , 1014, 1018, 1022
3, 7, 11, 15, ..., 1015, 1019, 1023

[0055] The 1st radix-4 butterfly execution will use the first four input data at the memory addresses 0, 4, 8, and 12, the 2nd butterfly will use the following four data at memory addresses 16, 20, 24, and 28, and so on. We can collect the output results after 18 cycles from the first input data entering the LH9124. The first 4 output data will be the execution result of the 1st radix-4 butterfly and the following 4 data will be the execution result of the 2nd radix-4 butterfly. The output data will be stored in the B-port memory with the address sequence as the input address sequence.

Stage 2:
Input Port: B (The input port will be the output port of previous stage)
Output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF42
B Port LH9320 Addressing Instruction: BF42
(Input and Output can use the same addressing instruction because in-place FFT algorithm is employed)
The addressing sequence will be
0, 16, 32, 48,..., 976, 992, 1008
1, 17, 33, 49,..., 977, 993, 1009
14, 30, 46, 62, ..., 990, 1006, 1022
15, 31, 47, 63, ..., 991, 1007, 1023

[0056] The 1st radix-4 butterfly execution will use the first four input data at the memory addresses 0, 16, 32, and 48, the 2nd butterfly will use the following four data at memory addresses 64, 80, 96, and 112, and so on. We can collect the output results after 18 cycles from the first input data entering the LH9124. The first 4 output data will be the execution result of the 1st radix-4 butterfly and the following 4 data will be the execution result of the 2nd radix-4 butterfly. The output data will be stored in the A-port memory with the address sequence as the input address sequence.

Stage 3:
Input Port: A (The input port will be the output port of previous stage)
Output Port: B
Data Flow Instuction: RAWB
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instuction: BF43
B Port LH9320 Addressing Instruction: BF43
(Input and Output can use the same addressing instruction because in-place FFT algorithm is employed)
The addressing sequence will be
0, 64, 128, 192, ..., 832, 896, 960
1, 65, 129, 193, ... , 833, 897, 961
...
62, 126, 190, 254,... , 894, 958, 1022
63, 127, 191, 255, ... , 895, 959, 1023

[0057] The 1st radix-4 butterfly execution will use the first four input data at the memory addresses 0, 64, 128, and 192, the 2nd butterfly will use the following four data at memory addresses 256, 320, 384, and 448, and so on. We can collect the output results after 18 cycles from the first input data entering the LH9124. The first 4 output data will be the execution result of the 1st radix-4 butterfly and the following 4 data will be the execution result of the 2nd radix-4 butterfly. The output data will be stored in the B-port memory with the address sequence as the input address sequence.

Stage 4:

Input Port: B (The input port will be the output port of previous stage)
Output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF44
B Port LH9320 Addressing Instruction: BF44
(Input and Output can use the same addressing instruction because in-place FFT algorithm is employed)
The addressing sequence will be
0, 256, 512, 768
1, 257, 513, 769
...
254, 510, 766, 1022
255, 511, 767, 1023

[0058] The 1st radix-4 butterfly execution will use the first four input data at the memory addresses 0, 64, 128, and 192, the 2nd butterfly will use the following four data at memory addresses 256, 320, 384, and 448, and so on. We can collect the output results after 18 cycles from the first input data entering the LH9124. The first 4 output data will be the execution result of the 1st radix-4 butterfly and the following 4 data will be the execution result of the 2nd radix-4 butterfly. The output data will be stored in the A-port memory with the address sequence as the input address sequence.

Detailed Explanation, 1K-point FFT Implemented by 4 Parallel LH9124s

[0059] For simplicity, we make an assumption that the input data are already in the digit-reverse order. Actually, there is no such limitation for our implementation. We may arrange the data stored in the B port of 4 LH9124 processor as follows:

[0060] Memory Bank 1 at the B port of processor 1:
b(0), b(1), b(2), b(3), b(4), ... , b(253), b(254), b(255)

[0061] Memory Bank 2 at the B port of processor 2:
b(256), b(257), b(258), b(259),b(260),... , b(509), b(510), b(511)

[0062] Memory Bank 3 at the B port of processor 3:
b(512), b(513), b(514), b(515), b(516), ... , b(765), b(766), b(767)

[0063] Memory Bank 4 at the B port of processor 4:
b(768), b(769), b(770), b(771), b(772), ... , b(1021), b(1022), b(1023)

[0064] We will choose the radix-4 butterfly instruction to implement 1K-point parallel FFT. We also need 5 radix-4 butterfly stages. However, in each stage, each LH9124 will computes 256 data instead of 1024 data.

[0065] The computation for each stage is shown as follows:

Stage 0:
Input Port: B
output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF40
B Port LH9320 Addressing Instruction: BF40
The addressing sequence for each bank will be
0, 1, 2, 3, 4, 5, ..., 251, 252, 253, 254, 255

[0066] For datapath processor 1, the 1st four input data will be 0, 1, 2, 3 and the 2nd four input data, will be 4, 5, 6, 7, and so on.

[0067] For datapath processor 2, the 1st four input data will be 256, 257, 258, 259, and the 2nd four input data will be 260, 261, 262, 263, and so on.

[0068] For datapath processor 3, the 1st four input data will be 512, 513, 514, 515, and the 2nd four input data will be 516, 517, 518, 519, and so on.

[0069] For datapath processor 4, the 1st four input data will be 768, 769, 770, 771, and the 2nd four input data will be 772, 773, 774, 775, and so on.

[0070] After 18 cycles later, we begin to get the output results from the A port. The results will be stored in the A port memory with the same address sequence as the input port.

Stage 1:
Input Port: A
output Port: B
Data Flow Instruction: RAWB
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF41
B Port LH9320 Addressing Instruction: BF41
The addressing sequence for each bank will be
0, 4, 8, 12, ... , 244, 248, 252
1, 5, 9, 13, ... , 245, 249, 253
2, 6, 10, 14, ... , 246, 250, 254
3, 7, 11, 15, ... , 247, 251, 255

[0071] For datapath processor 1, the 1st four input data will be 0, 4, 8, 12 and the 2nd four input data, will be 16, 20, 24, 28, and so on.

[0072] For datapath processor 2, the 1st four input data will be 256, 260, 264, 268, and the 2nd four input data will be 272,276,280, 284, and so on

[0073] For datapath processor 3, the 1st four input data will be 512, 516, 520, 524, and the 2nd four input data will be 528, 532, 536, 540, and so on.

[0074] For datapath processor 4, the 1st four input data will be 768, 772, 776, 780, and the 2nd four input data will be 784, 788, 792, 796, and so on

[0075] After 18 cycles later, we begin to get the output results from the B port. The results will be stored in the B port memory with the same address sequence as the input port.

Stage 2:
Input Port: B
Output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF42
B Port LH9320 Addressing Instruction: BF42
The addressing sequence for each bank will be
0, 16, 32, 48, ... , 208, 224, 240
1, 17, 33, 49, ..., 209, 225, 241
...
14, 30, 46, 62, ... , 222, 238, 254
15, 31, 47, 63, ... ,223, 239, 255

[0076] For datapath processor 1, the 1st four input data will be 0, 16, 32, 48 and the 2nd four input data, will be 64, 80, 96, 112, and so on.

[0077] For datapath processor 2, the 1st four input data will be 256, 272, 288, 304, and the 2nd four input data will be 320, 336, 352, 368, and so on.

[0078] For datapath processor 3, the 1st four input data will be 512, 528, 544, 560, and the 2nd four input data will be 576, 592, 608, 624, and so on.

[0079] For datapath processor 4, the 1st four input data will be 768, 784, 800, 816, and the 2nd four input data will be 832, 848, 864, 880, and so on.

[0080] After 18 clock cycles, we begin to get the output results from the A port. The results will be stored in the A port memory with the same address sequence as the input port.

Stage 3:
Input Port: A
output Port: B
Data Flow Instruction: RAWB (Read A Write B)
LH9124 Execution Instruction: BFLY4
A Port LH9320 Addressing Instruction: BF43
B Port LH9320 Addressing Instruction: BF43
The addressing sequence for each bank will be
0, 64, 128, 192,

1, 65, 129, 193,
...
62, 126, 190, 254,
63, 127, 191, 255

[0081] For datapath processor 1, the 1st four input data will be 0, 64, 128, 192 and the 2nd four input data, will be 1, 65, 129, 193, and so on.

[0082] For datapath processor 2, the 1st four input data will be 256, 320, 384, 448, and the 2nd four input data will be 257, 321, 384, 448, and so on.

[0083] For datapath processor 3, the 1st four input data will be 512, 576, 640, 704, and the 2nd four input data will be 513, 577, 641, 705, and so on.

[0084] For datapath processor 4, the 1st four input data will be 768, 832, 896, 960, and the 2nd four input data will be 769, 833, 897, 961, and so on.

[0085] After 18 cycles later, we begin to get the output results from the B port. The results will be stored in the B port memory with the same address sequence as the input port.

Stage 4:
Input Port: B
output Port: A
Data Flow Instruction: RBWA
LH9124 Execution Instruction: BFLY4
B Port LH9320 Addressing Instruction: BF40
A Port LH9320 Addressing Instruction: BF43
Only this stage uses the delay buffers and switching circuits.
(In the paper, we use the delay buffers and switching circuits in the first stage. The result is the same. Here, the order is reversed because it might be easier to explain and understand the approach.)

[0086] The input addressing sequence will be: 0, 1, 2, 3, 4, 5, ... , 252, 253, 254, 255

[0087] The datapath processor 1 will use the sequence 0, 4, 8, 12, ..., 244, 248, 252

[0088] Thus, the 1st radix-4 butterfly will get the four data from address 0 of all the four banks and the 2nd radix-4 butterfly will get the four data from address 4 of all the four banks, and so on.

[0089] The datapath processor 2 will use the sequence 1, 5, 9, 13, ... , 245, 249, 253 Thus, the 1st radix-4 butterfly will get the four data from address 1 of all the four banks and the 2nd radix-4 butterfly will get the four data from address 5 of all the four banks, and so on.

[0090] The datapath processor 3 will use the sequence 2, 6, 10, 14, ...,246,250,254 Thus, the 1st radix-4 butterfly will get the four data from address 2 of all the four banks and the 2nd radix-4 butterfly will get the four data from address 6 of all the four banks, and so on.

[0091] The datapath processor 4 will use the sequence 3, 7, 11, 15, ... , 247, 251, 255 Thus, the 1st radix-4 butterfly will get the four data from address 3 of all the four banks and the 2nd radix-4 butterfly will get the four data from address 7 of all the four banks, and so on.

[0092] The output sequence will be different from the input sequence. Each output port use the same BF43 instruction with time interleave.

[0093] The output sequence for each port will be:
0, 64, 128, 192, 1, 65, 129, 193, ..., 63, 127, 191, 255

[0094] The address for Processor i is one cycle ahead of that for processor i+1. Thus, the final data will be stored in A port.

[0095] The output results stored in the Bank 1 of A port will be:
X(0), X(4), X(8), X(12), ... , X(1016), X(1020)

[0096] The output results stored in the Bank 2 of A port will be:
X(1), X(5), X(9), X(13), ... , X(1017), X(1021)

[0097] The output results stored in the Bank 3 of A port will be:
X(2), X(6), X(10), X(14), ... ,X(1018), X(1022)

[0098] The output results stored in the Bank 4 of A port will be:
X(3), X(7), X(11), X(15), ... , X(1019), X(1023

Claims

1. In digital signal processing applications for processing N data points through Y processing stages using Z execution units, each execution unit having a plurality of I/O ports including A and B ports, a system for processing the data comprising:

an addressable memory unit for each A and B port on each execution unit, including memory units A(1) through A(Z) operatively connected to the A ports of execution units P(1) through P(Z), respectively, and including memory units B(1) through B(Z) operatively connected to the B ports of execution units P(1) through P(Z), respectively, and wherein data is processed through the Y processing stages by moving data in selected sequences through each execution unit P(q) between memory units A(q) and B(q);
a data router which distributes the N data points between memory units A(1) through A(Z), each memory unit receiving N/Z data points, each execution unit P(1) through P(Z) processing the a block of N/Z data points through the Y processing stages, the processing occurring in parallel;
a first address generator AG-A operatively connected to memory units A(1) through A(Z) for supplying the address sequences used in the Y processing stages to memory units A(1) through A(Z);
a second address generator AG-B operatively connected to memory units B(1) through B(Z) for supplying the address sequences used in the Y processing stages to memory units B(1) through B(Z); and
address generator AG-A supplies the same address sequence to all A(1) through A(Z) memory units, and address generator AG-B supplies the same address sequence to all B(1) through B(Z) memory units, whereby the N data points are processed in Z parallel streams through the Z execution units.

2. A system for processing data as in claim 1 in which each said addressable memory unit is a single-port memory.

3. In digital signal processing applications for processing N data points through Y processing stages using 4 execution units P(1) through P(4), each execution unit having a plurality of I/O ports including A and B ports, a system for processing the data comprising:

an addressable memory unit for each A and B port on each execution unit, including memory units A(1) through A(4) operatively connected to the A ports of execution units P(1) through P(4), respectively, and including memory units B(1) through B(4) operatively connected to the B ports of execution units P(1) through P(4), respectively, and wherein data is processed through the Y processing stages by moving data in selected sequences through each execution unit P(q) between memory units A(q) and B(q);
a data router which distributes the N data points between memory units A(1) through A(4), each memory unit receiving N/4 data points, each execution unit P(1) through P(4) processing the a block of N/4 data points through the Y processing stages, the processing occurring in parallel;
a first address generator AG-A operatively connected to memory units A(1) through A(4) for supplying the address sequences used in the Y processing stages to memory units A(1) through A(4);
a second address generator AG-B operatively connected to memory units B(1) through B(4) for supplying the address sequences used in the Y processing stages to memory units B(1) through B(4); and
address generator AG-A supplies the same address sequence to all A(1) through A(4) memory units, and address generator AG-B supplies the same address sequence to all B(1) through B(4) memory units, whereby the N data points are processed in 4 parallel streams through the 4 execution units.

4. In digital signal processing applications for processing 1024 data points through Y processing stages using 4 execution units P(1) through P(4), each execution unit having a plurality of I/O ports including A and B ports, a system for processing the data comprising.

an addressable memory unit for each A and B port on each execution unit, including memory units A(1) through A(4) operatively connected to the A ports of execution units P(1) through P(4), respectively, and including memory units B(1) through B(4) operatively connected to the B ports of execution units P(1) through P(4), respectively, and wherein data is processed through the Y processing stages by moving data in selected sequences through each execution unit P(q) between memory units A(q) and B(q);
a data router which distributes the N data points between memory units A(1) through A(4), each memory unit receiving 256 data points, each execution unit P(1) through P(4) processing the a block of 256 data points through the Y processing stages, the processing occurring in parallel;
a first address generator AG-A operatively connected to memory units A(1) through A(4) for supplying the address sequences used in the Y processing stages to memory units A(1) through A(4);

a second address generator AG-B operatively connected to memory units B(1) through B(4) for supplying the address sequences used in the Y processing stages to memory units B(1) through B(4); and
address generator AG-A supplies the same address sequence to all A(1) through A(4) memory units, and address generator AG-B supplies the same address sequence to all B(1) through B(4) memory units, whereby the 1024 data points are processed in 4 parallel streams of 256 data points each through the 4 execution units.

5. A system for processing data as in claim 4 in which each said addressable memory unit A(1) through A(4) and B (1) through B(4) is a single-port memory.

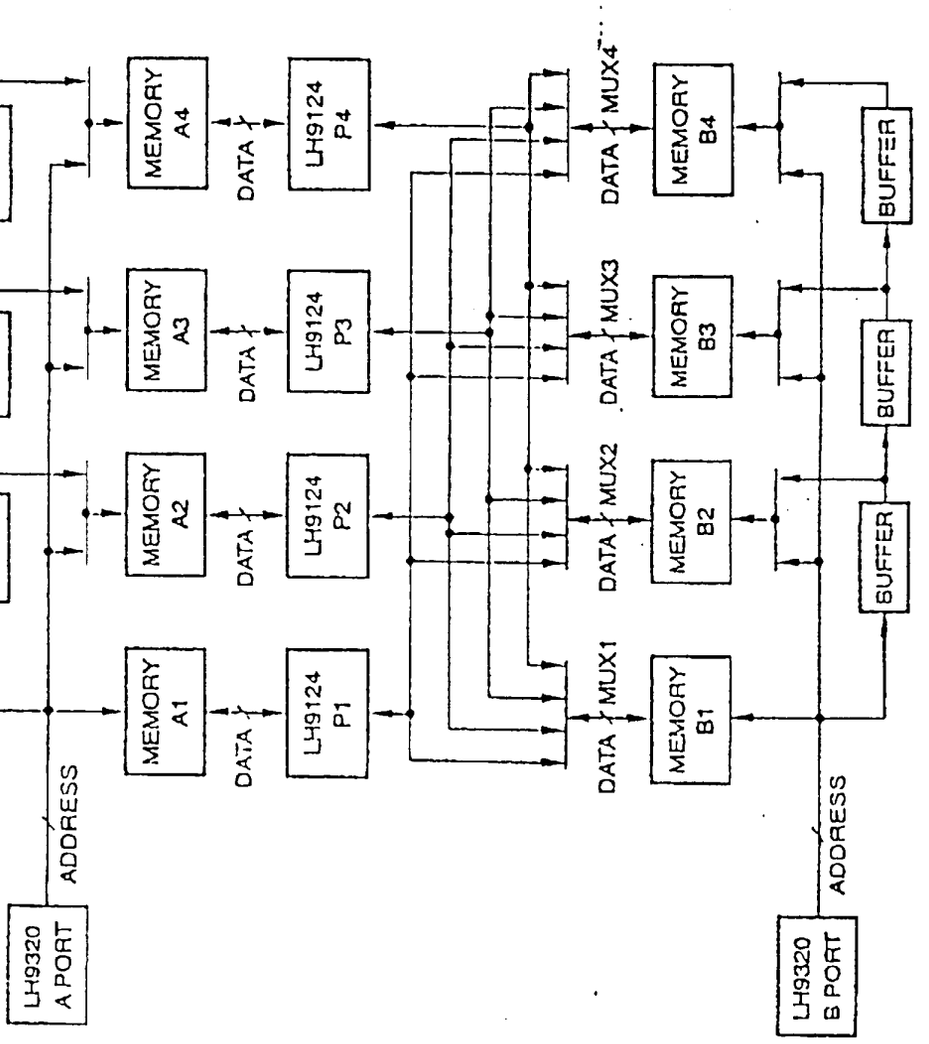


FIGURE 1

NOTES:

1. Buffer = 1 clock time delay.
2. Not shown: C-Port (coefficient port for twiddle factor)
   Q-Port (data acquisition)
3. MUX is used at first stage of computation only to interleave the data.