VI. SUMMARY AND CONCLUSION

The MAC module is essentially a programmable asynchronous sequencer designed to operate in the fundamental mode. Its use permits one defect-free fundamental mode state machine to be quickly converted to another radically different defect-free fundamental mode state machine simply by replacing one programmable logic device (PLD) with another to drive the MAC module. The MAC module could, for example, be externally microprogrammed to quickly switch from one state machine to another through a set of multiplexed PLD's. If machine size and complexity ever becomes a problem, MAC modules can be bit-sliced to produce modules of virtually any size without compromising speed or reliability.

Any asynchronous state machine that is implemented with the MAC module will not have critical races, essential hazards, d-trios, output race glitches or static hazards in the state variables. Provided that MAC module controller designs are protected by placing an arbiter (e.g., the Signetics bus arbiter) between the driving PLD and the MAC module, no restrictions on the external inputs to the system are needed for complex system reliability.

The versatility, speed and reliability of the MAC module make it an attractive alternative to synchronous controller design. Microprocessors and CPUs can now be conveniently and advantageously designed for multi-controller, clock-independent applications at speeds limited only by the logic circuitry and logic family technology that is used. Rapid switching between radically different MAC module designed controllers and the use of cascaded MAC modules permit time shared and parallel controller action. With increasing demands for greater speeds and versatility in digital systems of ever increasing complexity, there is a need for new, perhaps radical, approaches to design offered by the MAC module concept.

VII. ACKNOWLEDGMENTS

We gratefully acknowledge the support of Washington State University, the Boeing Aerospace Company, and the Patent and Copyright Office of Washington State University.

REFERENCES


A Pipeline Design for the Realization of the Prime Factor Algorithm Using the Extended Diagonal Structure

Daniel Pak-Kong Lun and Wan-Chi Siu

Abstract—In this brief contribution, an efficient pipeline architecture is proposed for the realization of the Prime Factor Algorithm (PFA) for digital signal processing. By using the extended diagonal feature of the Chinese Remainder Theorem (CRT) mapping, we show that the input data sequence can be directly loaded into a multidimensional array for the PFA computation without any permutation. Short length modules are modified such that an in-place and in-order computation is allowed. The computed results can then be directly restored back to the memory array without the need for further reordering. More importantly, the CRT mapping can also be used to represent the output data, hence we can utilize the extended diagonal feature of the CRT mapping to directly send the computed results to the outside world. As compared to the previous approaches, the present approach requires no shifting or rotation during the data loading and retrieval processes. In the case of multidimensional PFA computation, it does not require the computation to be split up into a number of two-dimensional computations. Hence, the overhead required for data loading and retrieval in each two-dimensional stage can be saved.

Index Terms—Prime factor algorithm, prime factor mapping, in-place and in-order computation, extended diagonal structure, Chinese Remainder Theorem mapping.

I. INTRODUCTION

The Prime Factor Algorithm (PFA) [1] is one of the efficient techniques for the computation of the discrete Fourier transform (DFT). The PFA makes use of the prime factor mapping (PFM) [2] technique to decompose the computation of the original problem into some short length computations. By using the efficient short length algorithms such as the Winograd short length algorithms [3], the PFA can give an extremely good computational performance. In many cases [4], the PFA can outperform the radix-2 type FFT algorithms [5] due to its low multiplicative complexity.

However, in the case of parallel realization, the radix-2 type algorithms seem to give better results compared to the PFA due to their highly regular structures. It has been shown that parallel computer architectures which are used for the radix-2 type algorithms, such as the hypercube structure [6], are not suitable for the realization of the PFA. Different architectures were then suggested [7]–[9]. The major concern which they put forward was the data loading and retrieval problems of the PFA, that is, how the data are loaded into the computing machine and how the results are retrieved from the machine. Previously, the PFM was considered to be a complicated process for either a software or a hardware realization because of the tedious modulo arithmetic computations. In [8], [9], the cyclic-shift algorithm is adopted to realize the PFM. Serial input data are loaded with a number of switches and rotations into a two-dimensional array for the subsequent short length computations.

Manuscript received August 26, 1991; revised September 15, 1992 and September 22, 1993. This work was supported by the Joint Edward Sai Kim Ho Hung Fund and Hong Kong Polytechnic Research Grant 34/93P8.

The authors are with the Department of Electronic Engineering, Hong Kong Polytechnic, Hung Hom, Hong Kong.

IEEE Log Number 9402550.
the computations, the results which are stored in the 2-D array are retrieved back into a serial form by either again using the cyclic-shift algorithm for non-in-order output [8] or using a dedicated selector for in-order output [7]. The cyclic-shift algorithm is complicated in terms of hardware realization. Moreover, if the sequence length N consists of more than two relatively prime factors that the PFA appears in a multidimensional form, the previous approaches require the computation to be carried out in a number of two-dimensional forms. This implies the overhead required for each two-dimensional stage will be accumulated and hence magnified.

In this brief contribution, we propose an enhanced pipeline structure for the parallel realization of the PFA. The effort required for data loading and retrieval is further simplified. It differs from the previous approaches [8], [9] in that no shifting or rotation hardware is required for the loading of serial input data into the two-dimensional array. Furthermore, no dedicated hardware is required to generate the in-order output. It is possible mainly because of the utilization of the extended diagonal structure [3], [10] in the realization of the prime factor mapping. The approach can be extended into a multidimensional form and is shown to be equally efficient as for the two-dimensional case.

II. IN-PLACE AND IN-ORDER MAPPING SCHEME

The PFA can be realized in an in-place and in-order form [11]. The term “in-place” computation is defined as the computation where the memory requirement is the same as the problem size. However, a very small amount of temporary buffers are usually allowed for the storage of the intermediate results in actual computation. The term “in-order” computation refers to the computation which has its output sequence in a natural order. These two properties are important for the hardware realization of the PFA because it reduces the memory requirement during the computation. Also, the extra hardware required to unscramble the output sequence into a proper order is eliminated.

A length-N DFT can be converted into an h-dimensional form as follows [11]:

\[
X(k_1, \ldots, k_h, k_h) = \sum_{n_1=0}^{N_1-1} \cdots \sum_{n_h=0}^{N_h-1} x(n_1, \ldots, n_h, n_{h+1}) \\
\times W_{N_1}^{M_1^{-1}n_1k_1} W_{N_2}^{M_2^{-1}n_2k_2} \cdots W_{N_h}^{M_h^{-1}n_hk_h}
\]

where

\[
N = \prod_{i=1}^{h} N_i; \text{GCD}(N_i, N_j) = 1; a, b = 1, 2, \ldots, h; a \neq b;
\]

\[
M_i = N_i / N_1; \text{ and } M_i > N_i = 1.
\]

The term GCD(a, b) refers to the greatest common factor of the integers “a” and “b” and the term (C_i) refers to the residue of the integer “C” modulo an integer “b.” X(k_1, \ldots, k_h, k_h) and x(n_1, \ldots, n_h, n_{h+1}) are the resulting h-dimensional representations of the input and output data, respectively, after the prime factor mapping. Although there are various mapping schemes which can be used to achieve one-to-multidimensional indices mapping [12], an in-place and in-order one was suggested in [11], namely

\[
x(n) = x(n_1, \cdots, n_2, n_1) = \left( \sum_{i=1}^{h} M_i M_i^{-1} n_i \right)_N
\]

where \( n_i = 0, 1, \cdots, N_i - 1 \)

\[
X(k) = X(k_1, \cdots, k_2, k_1) = \left( \sum_{i=1}^{h} M_i M_i^{-1} k_i \right)_N
\]

where \( k_i = 0, 1, \cdots, N_i - 1 \).

These mappings are in-place and in-order because the coefficients, namely \( M_i M_i^{-1} \) in (2) for the input data mapping are exactly the same as in (3) for the output data mapping. It implies that the results after the computation can be restored back to the original addresses where the input data are loaded from. No unscrambling is required to obtain the results in a correct order. Nevertheless, some modifications are required to make in (1) to achieve this in-place and in-order mapping. Note that for each \( W_{N_i} \) in (1), there exists a parameter \( M_i^{-1} \) which has its value totally dependent on the value of \( N \). As each \( W_{N_i} \) represents a stage of short length computation, the existence of \( M_i^{-1} \) implies that for different sequence length \( N \), different program code modules may be required for a single short length module. This is undesirable for software realization of the PFA since a FFT program should be able to handle various sequence lengths without changing the program. However, in hardware realization, it is common to have some applications which require only a fixed length FFT hardware module. Hence, the in-place and in-order mapping scheme in (2) and (3) can be applied. In fact, the existence of the term \( M_i^{-1} \) does not affect the arithmetic complexity of the short length modules. It only implies the use of a different set of multiplicative constants [13]. Thus, the use of this in-place and in-order mapping scheme will not affect the hardware realization of the PFA. Note that in (2) and (3), the mapping equations are selected to be the well-known Chinese Remainder Theorem (CRT) mapping due to its special extended diagonal property. This property is particularly useful for hardware realization.

III. A HARDWARE STRUCTURE USING EXTENDED DIAGONAL DATA LOADING AND RETRIEVAL

The CRT mapping has the property that the data are mapped sequentially on an extended diagonal. Its structure can be easily seen in a two-dimensional case. For example if the sequence length \( N = N_1 \times N_2 = 3 \times 5 \), then \( N_1^{-1} \) and \( N_2^{-1} \) are both equal to 2. By substituting these parameters into (2) and (3), we have the following equations. They can be shown graphically as in Fig. 1.

\[
n = \{5.2, n_1 + 3.2, n_2\}_{15} \quad k = \{5.2, k_1 + 3.2, k_2\}_{15}
\]
Fig. 4. A possible structure of each memory buffer.

Fig. 5. A possible connection scheme of the distributed memory system.

Fig. 1 shows that by means of the CRT mapping, the one-dimensional input data are mapped into a two-dimensional array. Note the special data arrangement in the two-dimensional array as it has been exemplified by an arrow in the figure. It shows that the input data are stored in the two-dimensional array starting from the upper left-hand corner, and listing the components down the "extended diagonals." This property of the CRT mapping has been stated in [3], [10], [13].

A multidimensional CRT mapping is similar to the 2-D case. Recall (2) and (3) which represent a general multidimensional input and output indexing using the CRT mapping:

\[
\begin{align*}
\nu_i &= \langle n \rangle_{N_i} \\
\kappa_i &= \langle k \rangle_{N_i}
\end{align*}
\]

By taking modulo with \( N_i \) to both sides of (4), we find that

\[
\begin{align*}
n_i &= \langle n \rangle_{N_i} \quad \text{and} \quad k_i = \langle k \rangle_{N_i}.
\end{align*}
\]

If the values of "\( n \)" and "\( k \)" are incremented by 1, let

\[
\begin{align*}
\nu_i &= \langle n+1 \rangle_{N_i} \\
\kappa_i &= \langle k+1 \rangle_{N_i}
\end{align*}
\]

We have

\[
\begin{align*}
n_i &= \langle n+1 \rangle_{N_i} = \langle (n)_{N_i} + 1 \rangle_{N_i} = \langle n+1 \rangle_{N_i} \\
\kappa_i &= \langle k+1 \rangle_{N_i} = \langle (k)_{N_i} + 1 \rangle_{N_i} = \langle k+1 \rangle_{N_i}
\end{align*}
\]

where \( \nu_i \) and \( \kappa_i \) are the two new indices. The above results show that the indices of all dimensions modulo with the respective dimension sizes.

By utilizing the extended diagonal feature of the CRT mapping, we design an efficient hardware structure for the computation of the PFA. Fig. 2 shows the desired hardware structure as it requires minimal I/O interface for general digital signal processing applications. In our design, the PFA processing module mainly comprises an h-dimensional distributed memory system. The number of dimensions \( h \) is equal to the number of the relatively prime factors that compose the sequence length. For example, if the sequence length \( N = 15 \), a two-dimensional distributed memory system is required as it is shown in Fig. 3. Each circle in the figure represents an one word memory buffer for the storage of temporary results.

Note that a long link, which is expressed in a form of arrows in Fig. 3, is used to chain up all memory buffers. The path of this data link is exactly the extended diagonal of this rectangular array. Here we assume that the input data under question are in a bit-serial form. This extra link, physically, is just a connection wire which connects one memory buffer to another. In general, the sequential data which are loaded through the extended diagonal of an h-dimensional array will realize an h-dimensional CRT mapping. We now further elaborate this point. Let us clarify our statement on "connecting the memory buffers through the extended diagonal of an h-dimensional array." We refer to the connections which obey the following rule,

\[
B(n_h, \cdots, n_2, n_1) \to B_{n, \cdots, n_2, n_1} \cdot (n_h + 1)_{N_h}, \ldots, (n_2 + 1)_{N_2}, (n_1 + 1)_{N_1}
\]

where \( B(i) \) represents a specific memory buffer in an h-dimensional memory array, and the symbol \( B(i) \to B(j) \) means memory buffer \( B(i) \) is connected to \( B(j) \). We allow only one system clock cycle time for a datum to transfer from \( B(i) \) to \( B(j) \). We assume that a data stream in descending order (in terms of the values of their indices) is sent to this h-dimensional memory array through the first memory buffer \( B(0, \cdots, 0, 0) \) one by one in every clock cycle. As the data are sent to the array sequentially, it takes \( "N-i-1" \) clock cycles for a particular datum, say \( "x_i" \), to be ready to the first buffer \( B(0, \cdots, 0, 0) \). Since totally \( N \) clock cycles are used to load
the data into the array, the datum "x," has only another "i + 1" clock cycles to reach the final position in the h-dimensional array. Now we can determine the final position of the datum "x," under this connection scheme. As it starts from \( B(0, \ldots, 0, 0) \) and has "i + 1" clock cycles to travel through the array, under the connection criteria defined in (7), the final position of the datum "x," should be \( B(i_{X_1}, \ldots, i_{X_h}, i_{X_1}) \) which is exactly the representation of a CRT mapping (Note that one clock cycle is required to latch the datum into \( B(0, \ldots, 0, 0) \) at the beginning).

IV. HARDWARE FOR FFT USING SHORT LENGTH MODULES

The structure of each memory buffer is shown in Fig. 4. Each input datum is a 32-bit complex number and is entered into the memory buffer in a bit-serial form. Two 32-bit First-In-First-Out (FIFO) shift registers are used for the storage of every complex input datum in each memory buffer. An \((h + 1)\)-to-1 multiplexer and an 1-to-\((h + 1)\) demultiplexer are also installed (where \( h \) is the number of dimensions) at the input and output of the FIFO shift registers, respectively. They provide different routes of data flow during 1) data loading or unloading and 2) short length computations. During data loading or unloading, the data are either loaded into or unloaded from the memory buffers through the extended diagonal. The ports "Loading" (as shown in Fig. 4), which should have been connected to the extended diagonal of the system, should be selected by the multiplexer and demultiplexer.

After the data are loaded into the memory buffer, they are then transferred to the short length FFT processors for computations. The data transfer between the processors and the memory buffers can be carried out in a number of ways. Nevertheless, in order to fully exploit the parallelism in the computation of the PFA, every independent short length computation would be evaluated with an individual processor. As shown in Fig. 5, five length-3 and three length-5 FFT processors are connected to all columns and rows, respectively, to perform parallel short length computations.

Based on the connection scheme in Fig. 5, the data in the memory buffers of each row are sent to the corresponding length-5 processors for the first stage computation. The computed results are retrieved back from each length-5 processor into the memory buffers of the corresponding row. Subsequently, the data in the memory buffers in each column are sent to the associated length-3 processors for the second stage computation. After the computed results are retrieved back from the length-3 processors to the memory, they can be unloaded through the extended diagonal to the outside world. One point which should be mentioned is that in order to achieve an in-place and in-order computation, the short length modules are the modified ones as described in Section II. The computational procedure can be summarized as follows:

1) Push the data sequentially into the \( h \)-dimensional array through the first memory buffer \( B(0, \ldots, 0, 0) \);
2) locate all the data in \( N \) clock cycles through the extended diagonal of the \( h \)-dimensional array;
3) perform the short length computations in each dimension, computed results are restored back to the \( h \)-dimensional array in the same locations where the input data were loaded from and
4) pull the results from the \( h \)-dimensional array from the last memory buffer \( B(N_h - 1, \ldots, N_2 - 1, N_1 - 1) \) through the extended diagonal.

Note that from loading the data into the array to obtaining the results back from the array, no shifting or rotation are required to be performed. The only effort is the use of an extra connection link.
Let us further illustrate the case for a multidimensional PFA computation by an example. Let the length $N = 2 \times 3 \times 5$, the PFA processing module in Fig. 3 should be constructed as a 3-D distributed memory system as shown in Fig. 6. Note carefully that the distributed memory system has been connected through an extra link according to (7). The data are fed into this 3-D array through the extended diagonal to all the memory buffers. Short length computations are then carried out in such a way as stated in Fig. 5. However, this time we need three different sets of short length FFT modules, i.e., the length-2, 3, and 5 modules, connected to different dimensions of the array. Computed results are restored back to the array and then pulled out, through the extended diagonal, to the outside world.

V. MULTI-STAGES PIPELINE STRUCTURE

For some applications in digital signal processing, the DFT's of some consecutive blocks of data have to be performed sequentially to obtain the spectral information of a real time signal. One of the examples is in the realization of the short time Fourier transform [14]. In this case, a hardware structure which is able to perform the DFT's of different blocks of data in a pipeline form can significantly increase the throughput of the computing machine. To serve this purpose, the design in Section IV should be modified as the pipeline structure shown in Fig. 7.

The objective of this modification is to allow pipelining to be taken place within the stages of short length computations. In Fig. 7, each circle represents a memory cell and is connected to two different stages of short length processors. Note that each memory buffer as stated in section 4 (see Fig. 6) contains "h" memory cells for an $h$-dimensional PFA computation in this modified version. The memory cells have the similar structure as that in Fig. 4. Fig. 8 illustrates the details of such a design. Note that because of these extra memory cells, once the short length processors of a particular stage finish their computations, another set of data can be loaded in to accomplish a pipeline computation. Recall the example in Fig. 6 where a length-30 PFA is to be computed. As it is a three-dimensional computation, there will be three memory cells in each buffer. The computation procedure of this length-30 PFA with the proposed pipeline structure is illustrated by Fig. 9.

Here, we assume that the times taken by all short length processors and the time required for loading (retrieving) the data to (from) the array are the same. Hence, a 5-stage pipeline is designed. The input data are first loaded into this 3-D array through the extended diagonal. The memory cell MB0 in Fig. 7 will be used as a buffer for data transfers during the loading and retrieval processes. Note that at any particular instant during the pipeline operation, the first short length processor should read an input datum from MB0 for every $N$ clock cycles, and at the same time the last short length processor should write a result to MB0. The throughput of the system is one output for every clock cycle.

VI. AN ANALYSIS ON THE PERFORMANCE AND HARDWARE REQUIREMENT

Let us clearly define the model for which our analysis is based on. First, for the rest of the analysis, the term "previous approach" refers to the approach given in [9]. Our objective is to compare the performance and the hardware requirements of the previous and present approaches in the computations of the DFT's of a sequence of multiple blocks, single precision floating point and complex data. The sequence length of each block is assumed to be $60 = 3 \times 4 \times 5$ such that the present approach would use a 3-D PFA module. With the same sequence length, the previous approach would require a three stages pipeline structure. All complex input data are sent to the system in a bit-serial form. Each datum travels from one memory cell to another in one system clock cycle time. For the ease of analysis, we assume that the processing time required for the loading and retrieval of every data block and the computation time of all short length modules are the same.

Table I shows the result of comparisons. It is seen that, as compared to the previous approach, the present pipeline architecture has a 15 times improvement in efficiency and average throughput. Besides, the total computation time used by the present approach for each length-60 data block is reduced by 88%. The memory cells required by the present approach is also reduced by 37.5%. More importantly, the number of links required by the present approach for I/O interface is reduced by 33.5%. The reduction of links in the present approach implies that less effort for routing may be required. Finally, the requirement of data rotation are totally eliminated in the present approach. This implies that besides the processors and memory cells, no other hardware structure is required for the present approach. The cost required to trade for all the above improvements is the use of a multiprocessor pipeline structure as compared to the single processor pipeline structure used by the previous approach. Note that the present approach could be modified to become a single processor pipeline.
structure. It is achieved by making some rearrangements between the memory cells and the respective short length processor. Reader may check that the efficiency and system throughput of such a single processor pipeline structure would be similar to that of the previous approach. However, a significant saving on memory cells, data links and the hardware structures, such as shifters, could be achieved.

VII. CONCLUSION

In this brief contribution, we have proposed an efficient pipeline architecture for the realization of the PFA based on the extended diagonal feature of the CRT mapping. We have shown that the input data sequence can be directly loaded into a multidimensional array for the PFA computation without any permutation. As the short length modules are modified that an in-place and in-order computation is allowed, computed results can directly be restored back into the memory array without the need for further unscrambling. Furthermore, the CRT mapping can be used to represent the output data, hence we can again utilize the extended diagonal to send the computed results directly to the outside world. As compared to the previous approaches, the present pipeline structure has the following advantages.

1) As compared to the previous propositions, no data shifting, rotation or hardware delay is required.
2) Unlike the previous propositions which require the computation of an \( d \)-dimensional PFA to be split up into \( d-1 \) two-dimensional ones, the current architecture enables the data loading and data retrieval of all dimensions to be done at once. This special feature significantly reduces the overheads, such as the extra hardware and data transfer time, in data loading and data retrieval.
3) The throughput of the present approach is higher since the parallelism of the PFA computation is fully exploited. The pipeline structure is optimally designed that no delay exists in the computation.
4) The present approach localizes the hardware required for a multidimensional pipeline structure into each memory buffer. The hardware structure originally required to be developed into different modules due to the different stages of a multidimensional PFA computation can now be grouped together into a single module. This feature may benefit the VLSI realization of the PFA.

ACKNOWLEDGMENT

The authors would like to thank the Research Committee of the Hong Kong Polytechnic.

REFERENCES