A versatile VLSI fast Fourier transform processor

by KUANG-CHENG TING and CHUAN-LIN WU
University of Texas at Austin
Austin, Texas

ABSTRACT

A versatile special-purpose VLSI fast Fourier transform (FFT) processor is presented. It can process variant data sizes of FFT and cooperate with other identical FFT processors to accomplish cascade and parallel FFT processing schemes. The operations of the single processor FFT processing scheme, the multiprocessor cascade FFT processing scheme, and the multiprocessor parallel FFT processing scheme are described. The results of performance analysis show that the combination of adaptive architecture capability and VLSI technology can provide a practical solution for meeting the goal of advanced real-time FFT processing.
INTRODUCTION

The fast Fourier transform (FFT) algorithm is one of the most widely used tools in digital signal processing systems. A large body of knowledge has been generated on the subject of the FFT algorithm, and its parallelism has been studied extensively. Recently, the VLSI FFT computational networks were proposed for constructing the special-purpose FFT processor. However, these studies of VLSI FFT computational networks do not consider the flexibility of processing different data sizes. The VLSI technology is constrained by the chip density, packaging area, and pin number. These constraints also cause the problem of I/O bound and computation bound. If one processes a user's FFT task in a special-purpose hardware FFT processor, the I/O operations of the source and result data may easily impose the performance limitation. In addition, for the distributed processing system, the distributed source data might be stored in a computer unit with several I/O ports or be arranged (or mapped) in multiple-port memories. Processing a user's FFT task with the arrangement of source data and available resources can improve the resource utilization and can prevent the performance limitation imposed by the I/O operations.

This paper presents a versatile VLSI FFT processor for the Star local network, which not only can process variant data sizes of FFT but also can cooperate with other identical FFT processors to accomplish the cascade and parallel FFT processing schemes. Star is a local computer network designed to integrate image database management and image analysis into a system. Figure 1 is the block diagram of the Star communication subnet. The system components are attached to the interface unit, which in turn connects to multiple ports of the interconnection network. The communication path is established via the destination tag-routing technique, and a path establishment is less than one microsecond. Star is flexible and can be configured into various topology to provide better performance level than other rigid special architecture.

In Section 2, the various parts of the versatile VLSI FFT processor are described. A detailed description of the processing user's FFT task on Star is given in Section 3. The operations of the single processor FFT processing scheme, the multiprocessor cascade FFT processing scheme, and the multiprocessor parallel FFT processing scheme are discussed separately. The performance analysis is done in Section 4. Section 5 is the conclusion.

A VERSATILE VLSI FFT PROCESSOR

Figure 2 is the block diagram of a versatile VLSI FFT processor. The processor communicates with other processors and data units through four interface units (IUs), denoted as IUo, IUn, IUS, and IU1, that connect to the Stamet. The processor control unit (PCU) accepts the FFT task description from the user (or other processor) and decides the sequence of actions to be taken; it coordinates and controls the activities of the whole processor. The MCSW switches between the memory bank unit and the computation unit (CU) serve the function of switching the input and output ports of the CU with two memory bank units MB0 and MB1. Such config-

![Figure 1—The block diagram of the Star communication subnet.](From the collection of the Computer History Museum (www.computerhistory.org))

![Figure 2—The block diagram of a versatile VLSI FFT processor](From the collection of the Computer History Museum (www.computerhistory.org))
from processor control unit (PCU)

Figure 3—The circuit diagram of memory bank unit

Figure 4—The block diagram of a VLSI FFT circuit

uation and bidirectional IUs eliminate the restriction of fixed I/O ports and allow the FFT processor to act as a bidirectional FFT processing processor. The memory bank control unit (MBCU) generates the memory address sequences and controls the read/write operation of four data storages M₀₋₃ in the memory bank unit. The switch control signals MSWC and Cₓ₋₃ set up the paths among data storages, IUs, and CU. The memory enable (ME), memory read (MR), and memory write (MW) signals control the operation of individual data storage. The circuit diagram of one of the memory bank units, is shown in Figure 3.

The computation unit is an FFT VLSI chip that contains a pipeline butterfly computation element (PIPECE) and a parallel FFT quotient network (PARQUO) as shown in Figure 4. The PIPECE offers the capabilities of a fast butterfly computation rate and the overlapping of I/O operations with the computation. The PARQUO offers the capability of parallel processing the FFT within certain data size ranges. The twiddle factors of the PIPECE come from the outside of the VLSI chip, while the twiddle factors of the PARQUO come from the presorted Read Only Memory (ROM) associated with each computation element (CE). Considering the pins limitation, the I/O ports of the VLSI FFT circuit are denoted as IN0, IN1, OUT0, and OUT1. The hand-shaking mechanism of the VLSI FFT circuit with the external world is done by the control unit with four hand-shaking signals: input available (INAVL), input acknowledge (INACK), OUTPUT available (OUTAVL), and output acknowledge (OUTACK). The control unit performs the function of accepting the operation command from the external world, coordinating the data input/output operations, and controlling the operations of CEs. The external world issue command to the control unit by activating the Command Strob (CMSB) signal and putting the command work into the IN0 and/or IN1 ports. The command word contains parameters to specify the active PIPECE or the active PARQUO operation mode.

The construction of the PIPECE is straightforward. With three pipeline real adders, three pipeline real subtracters, four pipeline real multipliers, and delays, one can form a pipeline butterfly computation element as shown in Figure 5. Considering the PIPECE as L concatenated computation stations, each station performs a portion of the butterfly computation. For computation station i, 1 < i < L, it can accept data from station i - 1 only if its intermediate result was accepted by station i + 1. Therefore, the last computation station accepts data from its previous station only after the external world has received its output. The hand-shaking mechanism can be incorporated between computation stations and implemented by means of simple hand-shaking protocol.

The transformation from the complete parallel FFT computational network such as the Shuffle-Exchange network to the equivalent quotient network can be found in Fishburn and Finkel's paper. Figure 6 is the circuit diagram of the PARQUO. Since each CE in the quotient network emulates the actions for several CEs in the large network, buffers are required to hold data, and this is accomplished by two parallel double queues (DEQs) denoted as DEQ0 and DEQ1. Two DEQs share two common pointers and an INQUE signal that controls one of the DEQs in accessing data from the IN0 or
A Versatile VLSI Fast Fourier Transform Processor

Figure 5—A pipeline butterfly computation element

IN1 port. Assume that the PARQUO is designed with 2^a CEs, which are addressed as C_{a-1}...C_0, and its maximum processing capability is 2^q-point FFT, where a < q. The input sequence of source data A(k), where k = 0 to 2^b - 1 and a < b <= q, is defined as

\[ \text{INO} := A(0_{b-2}...i_b); \]
\[ \text{INI} := A(i_{b-2}...i_b); \quad i_{b-2}...i_0 = 0 \text{ to } 2^{b-1} - 1. \]

The control unit enables the CE(0C_{a-2}...C_0) and CE(1C_{b-2}...C_0) to access 2^b-a data points from the INO and IN1 ports, respectively, by activating the INQUE signal and the CE addressing signals. After completing the external data input operation, each CE(C_{a-1}...C_0) holds the source data A(C_{a-1}...C_0i_{b-a-1}...i_b) in DEQO and A(C_{a-1}...C_0i_{b-a-1}...i_b) in DEQ1. The control unit starts activating all CEs to process the FFT. At the end of FFT computation, each CE(C_{a-1}...C_0) holds the final Fourier coefficients X(0_{i_1}...i_{b-a-1}C_0...C_{a-1}) in DEQO and X(1_{i_1}...i_{b-a-1}C_0...C_{a-1}) in DEQ1 according to the bit-reversal output order of the DIF isogeometry algorithm with perfect shuffle permutation. The output operation is then accomplished by sequentially accessing 2^{b-a-1} pairs of data from the DEQs of each CE, and it is expressed as

\[ X(0_{i_1}...i_{b-a-1}) = \text{OUT0}; \]
\[ X(1_{i_1}...i_{b-a-1}) = \text{OUT1}; \quad i_1...i_{b-1} = 0 \text{ to } 2^{b-1} - 1. \]

The PARQUO accepts the next group of data only if its DEQs are empty. This nonpipeline restriction simplifies the design of the control unit, but a price is paid for increasing the processing time.

Figure 6—The circuit diagram of the parallel FFT quotient network (PARQUO) with four computation elements

Figure 7, one can see that a 2^m-point FFT can be processed as s stages of butterfly computation and then 2^s groups 2^{m-s}-point FFT, where 0 <= s <= m - 1. Since the PARQUO has the maximum processing capability of 2^q-point FFT and the minimum processing capability of 2^{m-1}-point FFT without zero padding, the decision in decomposition is based on the
Figures 7 and 8 illustrate the signal flow graph of a 16-point radix-2 Decimation-In-Frequency FFT algorithm with the in-place property. The data size and the processing capability of the PARQUO. To avoid the side effect of zero padding, when the data size is smaller than $2^m$, the given FFT task is processed by activating the PIPECE. If the data size is larger than $2^m$ and smaller than $2^{m+1}$, then it is processed by the PARQUO. As the data size $2^m$ exceeds the maximum processing capability of the PARQUO, the FFT computation will first be performed by processing $m-q$ stages of butterflies in the PIPECE, and then the intermediate results of the $(m-q)$th stage are treated as $2^{m-q}$ groups $2^q$-point FFT, which can be processed by the PARQUO. When the PIPECE is activated, the intermediate results of one iteration are arranged in the internal data storages properly to be ready for the next iteration. After each iteration, the processor will change the processing direction by controlling the MCSW switches. Following the above decomposition rules, variant sizes of FFT can be processed in a single FFT processor.

Multiprocessor Cascade FFT Processing Scheme

In Figure 7, after the first half of the butterflies in stage 1 are done, the successive output of stage 1 can be processed in stage 2, and so on. Hence, for a given FFT task with G groups of $2^m$ data points, one can linearly connect an m number of FFT processors, and according to the sequence order of the linear connection, each processor is then assigned a Pseudo Number (PSN) to charge one stage of butterfly computation. The Linear($P$, $j$, $i$) defines the linear connection pattern such that the IU$_{j0}$ and IU$_{j1}$ of the FFT processor with PSN = $k$ connect to the IU$_{00}$ and IU$_{01}$ of the FFT processor with PSN = $k+1$, where $P$ is the number of processors and $1 \leq k \leq P$ and $i,j$ represent the two IU groups. The FFT processor with PSN = 1 accepts pairs of source data from its IU$_{j0}$ and IU$_{j1}$, whereas the processor with PSN = $P$ produces the Fourier coefficients from its IU$_{00}$ and IU$_{01}$, which are connected to the destination unit through the StarNet. Figure 8 shows the connection pattern of Linear($4$, $1$, $0$).

The data movement operation is divided into three phases and is shown in Figure 9. Suppose a 24-point 1-D FFT task, as shown in Figure 7, is processed by four linearly connected processors. At phase 1, the first processor queues the OUT1 data of the first four butterflies in M$_i$ and sends the OUT0 data through the IU$_{j0}$ to the next processor, which will queue the received data in M$_o$. At phase 2, the first processor queues the OUT1 data of the next four butterflies in M$_3$ and sends the OUT0 and queued M$_1$ data through the IU$_{j1}$ and IU$_{j0}$ to the next processor. The second processor stores the incoming data from the IU$_{j0}$ and M$_1$ data storage and processes the queues M$_0$ data and the incoming data from the IU$_{j1}$ as a pair of IN0 and IN1 data. Finally, at phase 3, the first processor...
A Versatile VLSI Fast Fourier Transform Processor

**Figure 9**—The data movement of cascade 1-D FFT processing scheme

```
(*COMMENT:
TASK_TYPE: CAScade Multiple One Dimensional FFTs.
TOPOLOGY: Linear.
DATA_SIZE: G groups of M-point data, M = 2^m,
REQUEST: log_M processors.
ASSIGN: Pseudo number PSN, v = 0 and u = 1, where PSN is an integer and 1 ≤ PSN ≤ m.
SOURCE: Source data are sent to the processor with PSN = 1 as the sequence of

For i = 0 to G - 1
IF i = 0 and c = 0
THEN MB_j(C_l, k); M_j(k) => IU_j;
ELSE MB_j(C_l, k); M_j(k) => IU_j;
END IF.
END FOR

* )
Begin
Linear(m, 1,0); (* establish linear connection *)
MCSW: = pass; i = v; j = u; (* set direction *)
s = PSN; (* specify computation stage *)
(* performing the FFT computation *)
For I = 0 to G - 1 do
For c = 0 to 2^s - 1 do
For k = 0 to 2^m - s - 1 do
Begin
[INPUT DATA FEEDING PROCESS]
Case of s
s = 1: MB_j(C_l, k); C_l, C_l, C_l, C_l;
IN0 <= IU_j;
IN1 <= IU_j;
Then
If k < 2^m-1:
(phase 1) Then MB_j(C_l, k);
M_j(k) => IU_j;
Else MB_j(C_l, k); M_j(k) => IU_j;
End IF.
(phrase 2)
Else MB_j(C_l, k); M_j(k) => IU_j;
MB_j(C_l, k); MCSW = pass;
IN0 <= M_j(k);
IN1 <= IU_j;
M_j(k) => IU_j;
End.
End IF.
End FOR
For k = 0 to 2^m-1 do
IF s < m
THEN MB_j(C_l, k);
End.
End.

Figure 10—CASMOD algorithm
```
sends its $M_i$ data through the $IU_{j}$ to the next processor, which processes the queued $M_i$ data and the incoming data from the $IU_{k}$ and a pair of IN0 and IN1 data. Similarly, the data movement operation is available for the second and third processor, and so on. In general, the processor with $PSN = k$, $1 \leq k \leq m$, repeats $2^{k-1}$ times of phase 1, 2, and 3 operations, and each processor overlaps the phase 3 operation with the next repetitive phase 1 operation. In the case of processing $G$ groups of 1-D FFT, the first processor (i.e., $PSN = 1$) overlaps the phase 3 operation with the next group's phase 1 operation, and the rest of the processors (except the last processor) repeat $G \times 2^{k-1}$ times of phase 1 to 3 operations. Note that the cascade FFT processing scheme only involves the active PIPECE operation mode.

The algorithm CASMOD (Figure 10) describing the cascade FFT processing scheme is given as follows. The notation "destination ← source" stands for the data transfer operation, and the transfer operations in each PROCESS occur concurrently except when they are separated by the conditional statement If-Then-Else. The INPUT DATA FEEDING PROCESS, COMPUTATION PROCESS, and OUTPUT DATA HANDLING PROCESS are pipelined. The states of switch control signals $C_{0-3}$ are represented with "↑" or "↓" to stand for the upper or lower link. The data path set by the MCSW or MSWC is either "pass" or "cross."

The multiprocessor cascade FFT processing scheme becomes attractive when the $G$ value is greater than one, because it reduces the external data transferring time by overlapping the receiving of source data and the transmitting of the results with the butterfly computation.

**Parallel Processing Multiple One-Dimensional FFTs**

Representing $2^x$ FFT processors in binary form as $PSN = P_{x-1} \ldots P_0$, the $Cube(P, c, i)$ defines the connection pattern of the $IU_{0}$ of processor $P_{x-1} \ldots P_0$ connecting to the $IU_{0}$ of processor $P_{x-1} \ldots P_0$, and the $IU_{0}$ of processor $P_{x-1} \ldots P_0$ connecting to the $IU_{0}$ of processor $P_{x-1} \ldots P_0$, where $P = 2^x$, and $0 \leq c < x - 1$ and $i$ represent one of two IU groups. Figure 11 shows the connection pattern of $Cube(4,1,1)$ and $Cube(4,0,0)$. Suppose one requests four FFT processors with $PSN = P_{0}P_{0}$ to process a 16-point 1-D FFT task as shown in Figure 7, then each processor will charge two-butterfly computations in each stage according to the order of $P$. Assume that the source data input ports will the $IU_{00}$ and $IU_{01}$, before starting the computation, and that each processor establishes $Cube(4,1,1)$ as shown in Figure 11. Those processors with $P_i = 0$ queue the $OUTO$ data and send the $OUT1$ data through $IU_{00}$, and processors with $P_i = 1$ queue the $OUT1$ data and send $OUT0$ data through $IU_{11}$. This data exchange operation is shown in Figure 12. When the last pair of incoming data arrive, each processor establishes $Cube(4,0,0)$ as shown in Figure 11 and stage 2 computation can start after finishing the exchange of intermediate results and switching the processing direction. It allows each processor to have two-butterfly computation time to establish the next connection pattern $Cube(4,0,0)$. In Starinet, a path establishment time is less than one microsecond.

This procedure is then continued until there is no need to exchange data; i.e., $c = 0$. In general, processing $2^m$-point FFT with $2^x$ processors, where $0 < x < m$, requires $x$ times of exchange steps and each step takes $2^{m-x-1}$ data transfer operations. After the $x$th data exchange step, each processor processes $2^{m-x}$-point FFT independently. The operation of parallel-processing multiple one-dimensional FFT is described in the PARMOD algorithm (Figure 13).

When $x = 0$, the above parallel processing scheme becomes a single processor FFT processing scheme. If $x = m-1$, i.e., each processor executes only one butterfly computation in each stage, one obtains the maximum parallelism in processing one-dimensional FFT.

**PERFORMANCE ANALYSIS**

The following parameters are defined.

1. $T_s$: one data item transfer operation time between the source/result data unit and the FFT processor.
2. $T_p$: one data item transfer operation time between FFT processors.
3. $T_i$: the input or output operation time of the PASQUO for one pair of data.
4. $T_b$: one butterfly operation time.
The Performance Measures of the Single FFT Processor

The total FFT processing time of G groups 2^m-point FFT is the single versatile FFT processor is formulated as follows:

\[ T_{\text{FFT}} = T_s \cdot G \cdot 2^m + T_s \cdot [m + G \cdot (m - 2)/L \cdot 2^{m-1}] \]  

\[ T_{\text{FFT}} = G \cdot T_s \cdot 2^m + G \cdot T_s \cdot m \cdot 2^{m-x-1} \]

---

Figure 12—The data exchange of parallel 1-D FFT processing scheme

Figure 13—PARMOD algorithm

Case 1. \( m \leq a \), active PIPECE.

Case 2. \( a < m \leq q \), active PARQUO.
Case 3. $q < m$, active PIPECE and PARQUO.

$$T_{\text{FFT}} = G \cdot T_e \cdot 2^m + G \cdot T_s \cdot 2^{m-1} + T_s \cdot [m - q + G \cdot (m - q - 1)] / L \cdot 2^{m-1} + G \cdot q \cdot 2^{m-1} - 1].$$ \hspace{1cm} (3)$$

To evaluate the speed performance of the designed versatile FFT processor, two conceptual machines are defined. The first one, named machine A, can always process any given size of FFT in maximum parallelism. The second machine, named machine B, is the sequential-type hardware FFT processor that sequentially executes the butterflies. If machine A and B each have two input ports and two output ports, then the processing time of $2^m$-point FFT in machine A is expressed as

$$T_{\text{MA}} = T_e \cdot 2^m + m \cdot T_b,$$ \hspace{1cm} (4)$$

and in machine B it is expressed as

$$T_{\text{MB}} = T_e \cdot 2^m + m \cdot 2^{m-1} \cdot T_b.$$ \hspace{1cm} (5)$$

One might note that the speed-up ratio of machine A is about $0(m/2^m L/T_b)$ over machine B.

Denote $R_b$ as $T_b/T_s$, $R_a$ as $T_a/T_s$, and let $a = 6$, $q = 10$, $L = 32$, and $R_a = 40$. Figure 14 shows the log$_2$ speed-up of machine A and a versatile VLSI FFT processor, compared with machine B, as a function of $m$ and $R_a$. As $R_b$ decreases, the I/O operation becomes a dominate term in evaluating the FFT processing time. In some applications, the source/result data might be stored in the medium-speed storage, which may cause the $R_b$ value to be small. In such a case, further improvement of the speed performance should be done by using either the multiprocessor cascade FFT processing scheme or the multiprocessor parallel FFT processing scheme. As semiconductor technology progresses, the increasing speed of hardware circuits reduces the $T_b$ value and results in the importance of the I/O consideration.

The Performance Measures of the Cascade Processing Scheme

Since the cascade FFT processing scheme only activates the PIPECEs and each processor overlaps its butterfly computation with the incoming data from the previous processor, its output handling rate is determined by the next processor. For easy illustration, assuming that $T_a = T_e$ and counting from the time the first pair of source data arrive at the first processor, the second processor can start its butterfly computation after $m/4 + T_e + T_b$ time units and the third processor can start its butterfly computation after $(m/4 + m/8) \cdot T_e + 2 \cdot T_b$ time units, and so on. This means that it takes about $m/2^2 T_e + m \cdot T_b$ time units to produce the first pair of Fourier coefficients. The total processing time of $G$ groups $2^m$-point FFT
with m linearly connected versatile VLSI FFT processors is then formulated as

\[ T_{\text{FFT}} = 2^{m-1} \cdot (G + 1) \cdot T_s + m \cdot T_b. \]  

(6)

Figure 15 shows the \( \log_2 \) speed-up ratio of the cascade FFT processing scheme, relative to machine B, as a function of m and G with \( R_b = 5 \) and \( R_b = 1 \) respectively. As G increases, the CASMOD FFT processing scheme gets better performance, and its throughput is twice as high as that of machine A for large G value. In applications where the source/result data storages only have several input/output ports, the designed cascade FFT processing scheme can achieve both high performance and high throughput.

The Performance Measures of the Parallel Processing Scheme

The total processing time of \( 2^x \) processors, \( 0 < x < m \), parallel-processing G groups 2\(^m\)-point FFT is formulated as

Case 1. \( m - x \leq a \).

\[ T_{\text{FFT}} = G \cdot T_s \cdot 2^{m-x} + G \cdot T_e \cdot [1 + (x - 1) \cdot 2^{m-x-1}] + T_b \cdot [m + G \cdot (m - x - 1) / L \cdot 2^{m-x-1}] . \]  

(7)

Case 2. \( a < m - x \leq q \).

\[ T_{\text{FFT}} = G \cdot T_s \cdot 2^{m-x} + G \cdot T_e \cdot [1 + (x - 1) \cdot 2^{m-x-1}] + G \cdot T_i \cdot 2^{m-x-1} + T_b \cdot [G \cdot (m - x - 1) \cdot 2^{m-x-1} + x] . \]  

(8)

Case 3. \( q < m - x \).

\[ T_{\text{FFT}} = G \cdot T_s \cdot 2^{m-x} + G \cdot T_e \cdot [1 + (x - 1) \cdot 2^{m-x-1}] + G \cdot T_i \cdot 2^{m-x-1} + T_b \cdot [m - q + G \cdot (m - x - q) / L \cdot 2^{m-x-1} + G \cdot q \cdot 2^{m-x-1}] . \]  

(9)

Because the data exchange of the first stage is overlapped with the butterfly computation and the incoming source data, its actual data exchange operation time is the last produced intermediate result. Siegel\(^{6}\) has presented a parallel processing 1-D FFT algorithm for the SIMD machine. Performing 2\(^m\)-point FFT in an SIMD machine with 2\(^x\) processing elements, \( 0 < x < m \), takes \( m \cdot 2^{m-x-1} \) butterfly operations and \( x \cdot 2^{m-x-1} \) external data transfer operations. Due to the lack of information about the internal data transfer operations in the processing elements of an SIMD machine, which depends on the detail hardware circuit design, the processing time of 2\(^m\)-point 1-D FFT in an SIMD machine is approximately and optimistically expressed as

\[ T_{\text{SIMD}} = T_s \cdot 2^{m-x} + T_b \cdot m \cdot 2^{m-x-1} + T_e \cdot x \cdot 2^{m-x-1} . \]  

(10)

Assuming that \( T_e = T_b \), Figure 16 is the \( \log_2 \) speed-up of an idealized SIMD machine and versatile FFT processors in parallel 1-D FFT processing scheme as a function of x and \( R_b \).

Figure 16—The \( \log_2 \) speed-up of an idealized SIMD machine and versatile FFT processors in parallel 1-D FFT processing scheme as a function of x and \( R_b \). The result shows that the parallel FFT processing scheme of designed versatile FFT processors gains higher speed performance. Note that in Figure 16, with \( R_b = 10 \), 32 designed FFT processors can have the same performance as an SIMD machine with 1,024 processing elements. Such comparison gives only the approximation; in fact, the processing elements of an SIMD machine are usually not designed to process the FFT algorithm only. Hence, the \( T_b \) value of an SIMD machine will be larger than that of a special-purpose FFT processor.

CONCLUSION

As semiconductor technology progresses parallel FFT computing architecture becomes more and more attractive in real-time applications. However, the associated communication problem and the related I/O problem also become more and more important. Performance of a theoretical special-purpose hardware FFT processor that can process any given size of FFT with maximum parallelism can easily be limited by the I/O operation.

The versatile special-purpose VLSI FFT processor described in this paper facilitates single and multiple processors using cascade and parallel FFT processing schemes for various applications and source data arrangements. The design of the FFT VLSI computation unit takes a more practical approach by considering the pins limitation and the progress of VLSI technology. The flexible memory organization and bidirectional processing capability allow the processor to deal with a variety of source input and result output sequences. Furthermore, the flexibility of processing variant sizes of FFT in single FFT and multiple FFT processors will be suitable for a multiuser real-time processing environment.
The results of the performance analysis show that the combination of Star architecture capability with VLSI technology and related technology developments can provide a practical approach toward meeting the goal of advanced real-time FFT processing. The cascade FFT processing scheme offers the capability of meeting both the high performance and high throughput requirements with limited I/O ports. Such a scheme appears to be attractive for collecting and processing large amounts of data in real-time. It is concluded that the parallel FFT processing scheme with multiple versatile VLSI FFT processors in Star can achieve higher performance than can the SIMD machine. In addition, the achievement of high performance through an exploitation of parallelism using a distributed computing approach not only significantly improves fault tolerance but also allows maximum flexibility.

REFERENCES