COBRA: A 100-MOPS Single-Chip Programmable and Expandable FFT

Tom Chen, Glen Sunada, and Jian Jin

Abstract—This paper presents an optimized column fast Fourier transform (FFT) architecture, which utilizes bit-serial arithmetic and dynamic reconfiguration to achieve a complete overlap between computation and communication. As a result, for a clock rate of 40 MHz, the system can compute a 24-bit precision 1K point complex FFT transform in 9.25 μs, far surpassing the performance of any existing FFT systems.

Index Terms—Bit parallel arithmetic, bit serial arithmetic, column FFT, DFT, FFT, VLSI.

I. INTRODUCTION

The fast Fourier transform (FFT) transforms a set of data from the time domain to the frequency domain or vice versa. Since it saves a substantial amount of computation time over the conventional design for testability (DFT) methods, the FFT is widely used in a number of diverse areas such as power-spectrum analysis, speech and image processing, and digital filter design. Since the introduction of the Cooley–Tukey FFT algorithm [1] and its variants in 1960’s, early implementations of the FFT algorithms have been mainly in software running on general-purpose computers. With increasing demand for high-speed signal processing, the processing speed of the FFT has become one of the major focuses of research and development of FFT systems. Since the late 1980’s, with the development of VLSI technology, several specialized VLSI FFT chips [5]–[7] have been introduced. Several FFT systems using wafer scale integration (WSI) technology have also been reported [8]–[10].

More recently, the multipurpose signal processor, the MVP chip, has been introduced [14], [12] to perform a variety of DSP functions, including the FFT. The MVP employs a massive parallel architecture incorporating a switch matrix, several DSP modules, and a general-purpose reduced instruction set computer (RISC) module to achieve real-time processing for a large number of DSP tasks. Another popular integrated circuit (IC), LH9124/BDSP9124 from Sharp and Butterfly digital signal processing (DSP), for FFT applications supports multipoort dataflow concept to avoid external multiplexing of data often necessary in other existing IC’s. Almost all these high-performance FFT chips employ parallel arithmetic units and cascaded structures with processing times ranging from 8.8 μs for a 256-point FFT to 112.8 μs for a 1024 point FFT. Although the cascaded structure provides a good cost–performance tradeoff, it has two major problems which affect its performance. The first problem is memory access conflicts during processing because the data output from one cascaded stage should be reordered before being passed to the next cascaded stage. To solve this problem, several methods such as phase rotation [11], commutation [15], and special memory structures [6] have been introduced. The memory access conflict problem can be minimized using these methods at the cost of extra hardware or extra delay in the data path.

The second problem with the cascaded structure is that it does not fully exploit the parallelism inherent in the FFT algorithms. It uses one butterfly processor to perform the entire column of FFT processing by repeatedly performing the butterfly operation \(N/R\) times where \(N\) is the transform length and \(R\) is the radix of the butterfly operation.

To further improve the performance of the existing FFT systems without relying on further improvement in silicon technology, architectural changes become necessary. The column FFT structure dramatically increases the exploitation of parallelism over the existing cascaded FFT structures. However, to implement a column FFT, hardware cost is increased dramatically and the cost of using conventional bit-parallel arithmetic will become prohibitive. To implement a 1024-point radix-4 FFT system, 256 butterfly processors are required. Taking the state-of-the-art DASP/PAC chipset [6] from Array MicroSystems Inc. as an example, which has an area of about 2.4 cm² using 1.2-μm CMOS technology, about 61 cm² of silicon area is required. Even with WSI technology, only short transform-length FFT’s can be implemented using bit-parallel arithmetic units.

An alternative approach to implementing column FFT is to use a bit-serial structure. For a word length of \(M\), the bit-parallel structure uses \(M\) times the silicon area of the bit-serial structure. Although the individual bit-serial arithmetic unit is slower than the bit-parallel unit, its simplicity in both control and computational logic makes it possible to exploit the maximum parallelism in the FFT. From the computational point-of-view, the bit-serial structure can sometimes be organized as a very tight pipeline structure to increase efficiency. From the control and communications point-of-view, signals are transmitted on single wires in bit-serial structure instead of on busses, as in bit-parallel structures, leading to a significant savings on communications and control requirements.

A column FFT architecture using bit-serial structures has been proposed in the past [13]. The similarities between the
COBRA architecture and the column FFT architecture in [13] include the use of a crossbar switch matrix as a central circuit-switching network and bit-serial arithmetic throughout computation. However, the COBRA chip has the following distinctive features:

1) An on-chip interface logic provides a much more efficient inter-chip data-exchange mechanism when multiple COBRA chips are used to form a longer transform length. As a result, the COBRA’s processing time for larger transform lengths is significantly reduced.

2) The external glue logic required for expansion in [13] has been totally eliminated, making it easier to expand the COBRA chip’s transform length.

3) A great deal of DFT features are incorporated into the COBRA chip.

This paper presents a column FFT system based on the principles of serial communication and computation. To provide the efficient data reordering required for the FFT, a crossbar switch matrix is used for dynamic reconfiguration. Furthermore, the switch matrix provides a mechanism for easy expandability. The COBRA chip uses a 24-b data word length and is capable of expanding to a 47-b internal word length. At a clock speed of 40 MHz, the chip can calculate a 64-point complex FFT in 5.5 $\mu$s, a 256-point complex FFT in 7.4 $\mu$s, a 1K point complex FFT in 9.25 $\mu$s, and a 4K point complex FFT in 11.1 $\mu$s. One of the main reasons the COBRA chip can achieve such a speed is that the inter-column communications time for the COBRA chip is completely overlapped with the processing time so that the processors have no idle time.

In Section II, the details of the COBRA chip architecture are presented. On-chip test features are discussed in Section III. The COBRA chip statistics and the performance comparison is presented in Section IV. In Section V, a system programming environment is illustrated to assist system designers in using and programming the COBRA chip. Finally, conclusions are presented in Section VI.

II. THE COBRA ARCHITECTURE

The idea of the column FFT structure is to use a column of butterfly processors ($N/R$ butterfly processors) repeatedly by feeding the reordered outputs back to the inputs of the butterfly processors. The details of the column FFT processing can be found in [13]. The number of column iterations for the transformation length $N$ is $\log_R N$, where $R$ is the radix of the butterfly operation. Table I compares the hardware cost and processing time of column FFT with other FFT implementations. The advantages of the COBRA chip in both performance and system complexity by using bit-serial processing structures are mainly due to two reasons. Firstly, bit-serial butterfly processors require much smaller silicon area so that it is possible to integrate an entire column of butterfly processors on to a single chip. Secondly, the use of switch matrix provides memory access for every datum output from the butterfly processors. By programming and reprogramming the switch matrix, the data can be easily reordered between data input and internal processing steps to most efficiently implement the constant geometry FFT algorithm [3], [13].

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Hardware Cost</th>
<th>Throughput</th>
<th>Area x Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive</td>
<td>1</td>
<td>$\frac{N}{R} \log_R N$</td>
<td>$\frac{N}{R} \log_R N$</td>
</tr>
<tr>
<td>Pipeline</td>
<td>$\log_R N$</td>
<td>$\log_R N$</td>
<td>$\log_R N$</td>
</tr>
<tr>
<td>Column</td>
<td>$\frac{R}{N} \log_R N$</td>
<td>$\log_R N$</td>
<td>$\log_R N$</td>
</tr>
<tr>
<td>Array</td>
<td>$B \log_R N$</td>
<td>1</td>
<td>$\frac{B}{N} \log_R N$</td>
</tr>
</tbody>
</table>

The COBRA chip shown in Fig. 1 consists of an array of 16 bit-serial radix-4 butterfly processors, a $128 \times 128$ crossbar switch matrix, a 128-element data-exchange block, a 128-element input/output (I/O) memory, and a controller. The I/O memory is divided into two sub-blocks: the input memory block and the output memory block, each of which is 128 bits × 24 bits in size. The memory sub-blocks are further divided into two sections: one for the real part of a 64-point complex input vector and one for the imaginary part. The input memory has two operational modes: parallel input mode and serial output mode. In the parallel input mode, the 24-b data bus is used to load elements of the input vector in sign-magnitude format into either the real or the imaginary portion of the input memory. Typically, the data will be loaded by alternating real and imaginary values. When all 64 complex input vector elements are loaded into the input memory and the controller is told to start processing data, all 64 complex numbers are serially shifted into the butterfly processors through the switch matrix. Similarly, the output memory also has two operational modes: serial input mode and parallel output mode. During the serial input mode, the serial format data output from the butterfly processors is shifted into the output memory through the switch matrix rows. Once all the data has been shifted into the output memory, the parallel output mode is used to output the data one 24-b word in two’s complement form at a time. Once again, the typical transfer order will be alternating real and imaginary parts.

Fig. 2 shows the logic structure inside a bit-serial radix-4 butterfly processor. The butterfly processor has three hybrid bit-serial/parallel complex number multipliers, 16 adders and subtractors, and two 47-b shift registers. There is a 49-clock-cycle output latency in a butterfly processor of which 24 clock cycles are used to input the data. Therefore, there are 25 clock cycles to reprogram the switch matrix between the input column and the processing columns. The $128 \times 128$ switch
matrix is divided into eight matrix slices, each of which is 16 × 128 in size. All eight slices can be programmed in parallel, giving a 17-clock-cycle programming time for the switch matrix.

The data-exchange block, shown in Fig. 3, provides temporary data storage that allows simultaneous transfer of the butterfly processor output data between any number of COBRA chips and is the key to the overlapped processing and communications that allows 100% processor usage. The data-exchange block is structured as a 128-element array of 25-b shift registers and 128 two's complement to sign-magnitude numeric format converters. Once the output of the butterfly processors is loaded into the data-exchange block, it can be routed back to the inputs with no further delays.

The twiddle values for each butterfly processor is stored in a register file local to the butterfly processor. A local register file is also used to store two interconnection patterns for each switch matrix slice. These register files are loaded after each power-up of the COBRA chip.

The control block generates the control signals to drive the other blocks according to the external control signals. The external control signals include signals for transform length control, global testing, switch matrix pattern loading, twiddle factor loading, data loading, transform start, and data output. The transform length control signals (CTL1, CTL0) indicate transform length of 64, 256, 1024, or 4096. The global test signal (G_test) allows testing different functional units of the chip before the beginning of the FFT computation. The switch matrix pattern load signal (WPN_load) loads the switch matrix connection patterns from the data bus into the per slice register files. The switch matrix connection patterns are generated by a routing software based on the transform length. The twiddle factor load signal (WFT_load) loads the twiddle values into the processor register files through the data bus. Data parallel input signals (DTP_in1, DTP_in2) control the loading of data into imaginary and real input data memories. The transform start signal (T_start) marks the beginning of the FFT transform and is the main reference point for other control signals. The data output enable signals (DOUT_enb1, DOUT_enb2) allow the system to output the imaginary and real data, respectively, after the FFT transform is completed.

The signals generated by the control block are as follows:

1) switch matrix reset signal (SW_reset), which clears all connections of the switch matrix;
2) the switch matrix programming signal (SW_prgm) during which is active, the switch matrix is reprogrammed;
3) butterfly processor reset signal (BFT_reset), which initializes the internal registers of the butterfly processors;
4) butterfly processor data input enable signal (BIN_enb), which allows the data to be shifted from the input data memory or data-exchange block into the butterfly processor to begin a new column computation;
5) most significant bit signal (MSB), which indicates the sign bit of the data;
6) butterfly processor output enable signal (BOUT_enb), which controls the time during which the butterfly processors output valid data to the switch matrix;
7) data-exchange-block clear signal (DEB_clr), which clears the sign-bit registers in the data-exchange block;
8) data-exchange-block input direction select signal (DEB_dir), which selects the input data either from the switch matrix or from the its input bus;
9) output memory input data shift signal (OMIN_sft), which allows data to be shifted into the output memory from switch matrix.

Fig. 4 shows the timing relationship of the required control signals for the 64-point FFT. There are two phases of the operation: the power_up phase and the processing phase.
During the power_up period, the system performs the testing, loads the switch matrix patterns and twiddle values, loads input data into the input memory, and, at the same time, the on-chip controller programs the switch matrix for the first computation column. After all these steps are completed, the system enters the processing phase marked by the T_start signal. The butterfly processors are initialized before the input data is shifted in. The butterfly processing begins when data are shifted into the butterfly processors. It takes 24 clock cycles for data to be shifted into the butterfly processors and then the switch matrix is reprogrammed for reordering the data for the next processing column. This period is labeled ‘1’ in Fig. 4. At the end of the switch matrix reprogramming cycle, the DEB_clr signal is used to flush the data-exchange block and clear the sign bit registers. Eight clock cycles after the switch matrix is reprogrammed, the first bit of the data is available and the BOUT_enb is set to high to allow it to be shifted into the shift registers in the data-exchange block. This section of the column processing, labeled ‘2’ in Fig. 4, is the overlap of processing and communication. The BOUT_enb stays high for 24 clock cycles for the butterfly processors to finish the computation and to shift data into the shift registers in the data-exchange block. At the end of the butterfly output cycle, the DEB_msb signal is used to indicate the sign bit of the output. To start the second column computation, the BFT_reset signal is used to initialize the butterfly processors. The second-column computation takes the same number of clock cycles as the first column. Since the constant geometry algorithm is used, data output from the second to the third column have the same transfer routes as the data from the first to the second column so that the switch matrix does not need to be reprogrammed. During the third-column computation, the switch matrix is reprogrammed for the butterfly processors to input data from the input data memory for the next FFT transform. This period is labeled ‘1’ in Fig. 4. When the reprogramming is completed, the BOUT_enb and OMIN_sft signals become active for a period of 24 clock cycles to ensure that data are shifted to the output memory instead of the data-exchange block. Fig. 4 shows that the total number of clock cycles for completing a 64-point FFT is 222, which is 5.55 \mu s at a clock frequency of 40 MHz. One COBRA chip is capable of performing a 64-point complex-number FFT.

In addition to using the on-chip controller to perform the FFT operation, the COBRA chip is also capable of being used under external control for such operations as complex multiplication. To perform the complex multiplication, the following three processing steps are carried out:

1) convert the two’s complement input data into sign magnitude for multiplier input;
2) perform the multiplication;
3) convert the two’s complement multiplier output into sign-magnitude output.

The data-exchange block is used to perform steps 1 and 3 and the butterfly processing units are used to perform step 2. These steps require the use of the G_test signal to disconnect the internal controller. We also use the op_mode signal to output the multiplier result directly instead of the result of the butterfly post-add array. All the normal control signals, except SW_prgm and SW_reset, are also used during multiplication. Since there is only one switch matrix connection pattern for...
the multiply operation, the SW_reset and SW_prgm signals are issued during the power-up phase only. The butterfly processing units have been optimized for area by using only three complex multipliers. Therefore, a COBRA chip is capable of performing 48 complex multiplications simultaneously. Since the multiplier operation does not use the post-add array of the butterfly processor, the multiply time is shortened by two clock cycles. To perform the two format-conversion steps, 25 clock cycles are added before and after the multiply to allow the data to pass through the data-exchange block. The twiddle-factor memories on the COBRA chip are capable of holding six sets of multiplication coefficients. Fig. 5 shows the timing diagram for the multiply operation.

A. Expanding the Single-Chip 64-Point FFT System

Practically, the transform length for an FFT system is often more than 64 points. To perform FFT’s with longer transform lengths, it is widely desired that FFT systems with short transform lengths be expandable so that they can be used easily to form new FFT systems with longer transform lengths. The COBRA chip discussed above is expandable and can be used as a basic building block for constructing larger column FFT systems.

1) Expanding to a 256 FFT System To construct a 256-point column FFT system, 64 radix-4 butterfly processors are needed. Fig. 6 shows the structure of the 256-point FFT system. A 256-point column FFT uses four COBRA chips without any extra glue logic. The data-exchange block along with the 128 × 128 switch matrix in each chip takes care of all the intra-chip as well as the inter-chip communications. The 128-b data-exchange bus, shown in Fig. 1, is divided into four groups labeled as $G_{i0}$, $G_{i1}$, $G_{i2}$, and $G_{i3}$ in Fig. 6. $G_{i0}$ is a group of the first output of each butterfly in chip $i$, $G_{i1}$ is a group of the second output of each butterfly in chip $i$, $G_{i2}$ is a group of the third output of each butterfly in chip $i$, and $G_{i3}$ is a group of the fourth output of each butterfly in chip $i$. To start the 256-point FFT, data are loaded into the input memories through the 24-b input bus. After all the data are loaded, the butterfly processors begin processing. After the first column computation, the output data is shifted into the data-exchange block and are then routed to other COBRA chips by external busses. Since the constant geometry algorithm is used, the inter-chip data flow pattern is unchanged for every computational column. According to the constant geometry algorithm, the 64 outputs from the first chip is routed to the first input of every butterfly processor and the first output of each butterfly is routed to the first chip, the second output of each butterfly is routed to the second chip, and so on. The group index is given by $G_{ij}$, where $i = 0, 1, 2, 3$ and $j = 0, 1, 2, 3$.

Note that one of the four groups of each 64-point FFT chip are routed within the same chip. Therefore, each chip needs to transfer 48 data and receive 48 data from the other three chips. Fig. 7 shows the time overlap of executing a 256-point FFT using the constant geometry algorithm.

2) Expanding to a 1024 FFT System The same principle can also be applied to constructing a 1024-point FFT system by using 16 COBRA chips without any glue logic. Fig. 8 shows the organization of the 1024-point FFT system. Again, the data-exchange block and the switch matrix in each chip facilitate simultaneous data transfer after each computational column. Five column computations are needed to complete the 1K FFT transform. Each column needs 74 clock cycles. Therefore, 370 clock cycles are needed for the 1K FFT transform.

III. Test Features in the COBRA Chip

The COBRA chip incorporates several DFT features, which include functional block isolation, added control and observation points, functional unit scan, and internal controller override capabilities. The functional block isolation is used to allow testing inputs to be applied only to a single functional unit without interaction from other functional units. This is primarily used to perform a full functional test of the data-exchange block. Once the data-exchange block has been tested, the data-exchange input pins are used to apply stimulus to the other functional units while the output of those units is monitored at the data-exchange block output pins. To facilitate testing the switch matrix, we included a functional unit scan path around it. This scan path allows us to directly observe the values on every column of the switch matrix. After the switch matrix is tested, it is used to transfer data while testing the input and output memories. Finally, the butterfly processors have multiplexors that allow us to directly observe...
the multiplier outputs and apply a stimulus directly to the post
addition array.

To test the COBRA chip, the on-chip main controller is
bypassed using the G_test signal and testing is conducted
according to the following order of steps.

1) **Data-exchange-block testing**: The data-exchange block
is isolated from the rest of the system and a sequence
of two two-dimensional (2-D) checkerboard patterns and
appropriate control signals are applied. This does a full
functional test of the data-exchange block, and is shown
as route 1 in Fig. 9.

2) **Switch matrix testing**: The switch matrix is tested in two
passes, shown as route 2 in Fig. 9.

   Firstly, appropriate control signals to disconnect all
switch cells are generated. Next a “1” is applied to all
data-exchange-block input pins and passed on to the
switch matrix inputs. The data around the switch matrix
is then captured by the scan latches between the switch
matrix and the butterfly processors, and scanned out into
a buffer. After the scan out is completed, a “0” is applied
in the switch matrix inputs in the same manner, and the
data from the switch matrix is once again captured by
the scan latches and scanned out. The results of this
scan are compared with the buffer and any “0” → “1”
transitions indicate a stuck on cell.

   Secondly, appropriate control signals to sequentially
connect all switch matrix columns to one row at a
time are generated. The application of “0” to the switch
matrix row followed by scan, and the application of “1”
followed by scan, the same steps as in the first pass, are
performed. The absence of a “0” → “1” transition in a
switch matrix column indicates a stuck open or stuck at
fault in the switch cell intersecting the column and the
row.

3) **Input memory testing**: A 2-D checkerboard pattern is
loaded into the input memory and appropriate control
signals are generated to shift the pattern out through the
data-exchange block. The complementary 2-D checker-
board pattern is then loaded and shifted out. This is
shown as route 3 in Fig. 9.

4) **Output memory testing**: A 2-D checkerboard pattern is
shifted into the output memory from the data-exchange
block and is subsequently read out from the output
memory. The complementary 2-D checkerboard pattern
is also shifted into and read out from the output memory.
Route 4 in Fig. 9 shows the data flow for this test step.

5) **Butterfly processor testing**: This test is performed one
butterfly at a time and is a combination of a stuck at
fault test and functional test. The test patterns for the
stuck-at test have a 98% stuck-at fault coverage. We
believe that a stuck-at test with a high fault coverage,
together with an at-speed functional test, should be
sufficient enough to achieve a low defect level after test.
Due to the bit-serial structure in the butterfly processor,
the amount of logic between the latches is very small.
This makes a timing-related test, such as a delay test,
unnecessary. The stimulus is applied through the data-exchange-block inputs and the results are monitored at the data-exchange-block outputs, as shown by route 5 in Fig. 9.

IV. IMPLEMENTATION AND PERFORMANCE COMPARISON

Assuming a clock rate of 40 MHz, Table II gives the performance comparison between the COBRA column FFT system presented in this paper and several recent commercial FFT products.

It follows from Table II that for long transform lengths, the performance of the COBRA chip is much faster than the existing commercial FFT systems. This is because the column FFT systems are able to use additional hardware more efficiently.

We have implemented the COBRA chip on silicon using a 0.75-μm triple-layer metal CMOS technology. The chip has about 1.2 million transistors and the die size is $13.5 \times 11.5 \text{ mm}^2$ or $155.25 \text{ mm}^2$. Fig. 10 is the microphotograph of the chip. Comparing our system’s cost (chip area) with the most recent commercial FFT chipset (DASP/PAC) from Array Microsystems Company, the DASP has about 160,000 logic gates and a die size of $40,922 \times 11,684 = 127,61 \text{ mm}^2$ in a 1.2-μm CMOS technology. The PAC chip has about 100,000 logic gates and a die size $11,836 \times 11,938 = 141,30 \text{ mm}^2$ in a 1.2-μm CMOS technology. A 64-point cascaded FFT

### Table II

<table>
<thead>
<tr>
<th>Systems</th>
<th>64-point (μs)</th>
<th>256-point (μs)</th>
<th>1024-point (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plessey PSDP16510 FFT Processor</td>
<td>20</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>Array Microsystems DASP/PAC (recursive)</td>
<td>5.28</td>
<td>—</td>
<td>131</td>
</tr>
<tr>
<td>Array Microsystems DASP/PAC (cascaded)</td>
<td>2.2</td>
<td>8.8</td>
<td>26</td>
</tr>
<tr>
<td>E-Systems Systolic FFT (recursive)</td>
<td>5.04</td>
<td>22.08</td>
<td>104.4</td>
</tr>
<tr>
<td>E-Systems Systolic FFT (cascaded)</td>
<td>1.28</td>
<td>5.12</td>
<td>20.48</td>
</tr>
<tr>
<td>200 MIPS WSI System</td>
<td>—</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>MVP</td>
<td>7.05</td>
<td>28.2</td>
<td>112.8</td>
</tr>
<tr>
<td>Sharp/Butterfly LH9124 (recursive)</td>
<td>—</td>
<td>—</td>
<td>80.7</td>
</tr>
<tr>
<td>Sharp/Butterfly LH9124 (cascaded)</td>
<td>—</td>
<td>—</td>
<td>25.6</td>
</tr>
<tr>
<td>The COBRA chip</td>
<td>5.55</td>
<td>7.4</td>
<td>9.25</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th></th>
<th>64-point</th>
<th>256-point</th>
<th>1024-point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area mm$^2$</td>
<td>DASP/PAC</td>
<td>453.77</td>
<td>605.0</td>
</tr>
<tr>
<td></td>
<td>COBRA chip</td>
<td>155.25</td>
<td>621.0</td>
</tr>
<tr>
<td>Area mm$^2 \times 10^9$</td>
<td>DASP/PAC</td>
<td>2196.2</td>
<td>46853</td>
</tr>
<tr>
<td></td>
<td>Column FFT</td>
<td>1701.8</td>
<td>34006</td>
</tr>
</tbody>
</table>
system requires three DASP/PAC sets. The total silicon area is \(3 \times (127.6 + 141.3) = 806.7\) mm\(^2\). Table III gives Area \(\times\) Time\(^2\) comparison of COBRA chip with the DASP/PAC chip set with its area scaled to the equivalent of using the 0.75-\(\mu\)m technology.

V. SYSTEM PROGRAMMING USER INTERFACE

To simplify the programming of a COBRA-based FFT system a Motif/X-windows-based development system has been developed that will generate the required switch matrix connection patterns and twiddle values for forward FFT, inverse FFT (IFFT), and complex number multiplication. The IFFT is performed by prescaling the output of the multiplier and changing the connection and twiddle values while using the on-chip controller. Electrically programmable read-only memory (EPROM’s) can be used to transfer the connection patterns and twiddle values from the development station to the FFT system.

Fig. 11 shows the Motif interface on the development station. The major vertical areas of the screen from the top are the menu bar, individual block control areas, global control area, and simulation plot area. The individual block control area has the following three separate sections:

1) FFT control;
2) multiplier control;
3) IFFT control.

Each area has the required control information push buttons to specify the various files used by each module. The download file for each section is used to store the data that will be burned into the EPROM, and the simulation file is a stimulus file to perform a logic simulation of the system. The FFT and IFFT control information is the size of the operation, and the multiplier control information is the size of the operation and the coefficient set to use. The simulation output area consists of a real and imaginary plot of the input, a real and imaginary plot of the FFT output, a real and imaginary plot of the multiplier output, and a real and imaginary plot of the IFFT output.

VI. CONCLUSION

We have presented a high-speed column FFT system that can reach a speed which is far superior than any of the existing systems. This improvement can be attributed to exploiting the maximum parallelism in the FFT system and, more importantly, the efficient computation and communication arrangement that results from the complete overlap of butterfly processing and data transfer between columns. The easy expandability of the architecture allows longer transform lengths by simply connecting multiple COBRA chips together. The performance data indicates that at the transform length of 64-points, our system is not as fast as other cascaded systems, but better than the recursive systems. However, since those
cascaded systems need three chip sets to perform the 64-point processing while our system only uses one chip, the cost performance is still in favor of our system. For the transform length of 256 or longer, the FFT system based on COBRA chips is much better than any of the existing systems both in speed and costs.

ACKNOWLEDGMENT

The authors thank M. Berzins for his contribution to some of the cells used in the layout. The authors would also like to thank many people at Symbios Logic Inc., Fort Collins, CO, for giving them a lot of help during the design and testing of the COBRA chip.

REFERENCES


Tom Chen received the B.Sc. degree in electronic engineering from Shanghai Jiao-Tong University, Shanghai, China, and the Ph.D. degree in electrical engineering from the University of Edinburgh, Edinburgh, Scotland.

He was with Philips Semiconductors, Hamburg, Germany, and the New Jersey Institute of Technology, Newark. He is currently an Associate Professor in the Department of Electrical Engineering, Colorado State University, Fort Collins. His research interests include novel computation and signal-processing architectures, VLSI design and testing, and real-time computer vision systems.

Glen Sunada received the B.S.E.E. degree from Colorado State University, Fort Collins.

He was a Research Associate in the Department of Electrical Engineering, University of Colorado at Boulder. He is currently a Research Associate in the Department of Electrical Engineering, Colorado State University. His research interests include VLSI design and testing and computer-aided design (CAD) techniques for VLSI design.

Jian Jin received the M.S.E.E. degree from Colorado State University, Fort Collins.

He is currently with Trident Microsystems Inc, Mountain View, CA. His research interests include VLSI architectures for signal processing and graphics applications.