An extensible complex fast Fourier transform processor chip for real-time spectrum analysis and measurement.

Ediz Cetin
Richard Morling
Izzet Kale

School of Informatics

An Extensible Complex Fast Fourier Transform Processor Chip for Real-Time Spectrum Analysis and Measurement

Ediz Çetin, Richard C. S. Morling, *Member, IEEE,* and Izzet Kale, *Member, IEEE*

***Abstract*—This paper describes in detail the design of a CMOS custom fast Fourier transform (FFT) processor for computing a 256-point complex FFT. The FFT is well-suited for real-time spectrum analysis in instrumentation and measurement applications. The FFT butterfly processor reported here consists of one parallel–parallel multiplier and two adders. It is capable of computing one butterfly computation every 100 ns thus it can compute a 256-point complex FFT in 102.4 μs excluding data input and output processes.**

***Index Terms*—Digital signal processors, discrete Fourier transform (DFT), fast Fourier transform (FFT), FFT butterfly, integrated circuit, silicon implementation, spectrum analysis, very large scale integration.**

I. Introduction

THE discrete Fourier transform (DFT) is of considerable importance in instrumentation, measurement and digital signal-processing (DSP) applications. However, the computation burden of the DFT had prevented it from being widely implemented in real-time applications. A fast implementation of the DFT is the fast Fourier transform (FFT). With the development of high-speed processors, the FFT has found many real-time applications in the field of measurement and instrumentation. With users demanding higher processing speeds for real-time measurement applications, dedicated FFT processors are replacing general-purpose DSP's in some application areas [1]–[3]. A number of dedicated FFT processor implementations have been reported in the literature [4]–[8]. The FFT processor architecture presented in this paper differs from all these, in that a bit-parallel pipelined butterfly processor is used rather than the commonly used bit-serial counterpart. Also, instead of having a column of butterfly processors, a single butterfly processor is deployed. In addition to this, the processor is programmable in the sense that the basic architecture enables it to be used for different size FFT operations and is capable of other commonly used functions such as windowing, filtering and fast convolution. The FFT processor chip reported in this paper is intended as a demonstrator of the basic architecture and is restricted to 256-point transforms by virtue of the on-chip RAM size.

Manuscript received June 1, 1997; revised April 1, 1998. This work was supported by the Overseas Research Students Award Scheme (ORS).

The authors are with the Department of Electronic Systems, University of Westminster, London W1M 8JS, U.K. (e-mail: cetine@cmsa.westminster.ac.uk).

Publisher Item Identifier S 0018-9456(98)05451-5.

In the following sections we shall describe the architectural design, silicon implementation, and logic-level simulation of our FFT processor.

II. FFT Basics and Implementation Considerations

The N-point DFT of a sequence $x(k)$ is defined as [9]

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

and the Inverse DFT (IDFT) is defined as

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} X(n) W_N^{-nk} \qquad k = 0, 1, \cdots, N-1 \quad (2)$$

where

$$W_N = e^{-j2\pi/N} = \cos\left(\frac{2\Pi}{N}\right) - j \sin\left(\frac{2\Pi}{N}\right). \quad (3)$$

The IDFT is easily computed without any major change to the DFT algorithm. The only extra facility required is the conjugation of W_N. This is simply accomplished by negating the imaginary part of W_N.

The algorithm used in our FFT processor implementation is the modified version of the Cooley and Tukey's Decimation-In-Time (DIT) FFT algorithm with inputs in natural order and outputs in bit-reversed order, i.e., output scrambling. This input and output configuration is required if the processor is to be used for filtering applications. Fig. 1 shows the form of this scrambling.

As can be seen from Fig. 1, the modified 8-point DIT-FFT algorithm consists of three butterfly stages. To the left, we have eight input data samples. Input data is multiplied with the twiddle factor W_N^K. The solid dots represent addition/subtraction.

The outputs are in bit-reversed order. Generally, an N-point DIT-FFT algorithm consists of $\log_2 N$ stages, each stage containing $N/2$ butterfly operations [10].

Since the butterfly used here is the DIT-FFT radix-2 butterfly with all wordlength reductions performed at the output of the butterfly, results from the butterfly are scaled and quantized back to 24-bits in order to prevent overflow due to multiply and add/subtract operations. Convergent rounding is used since it

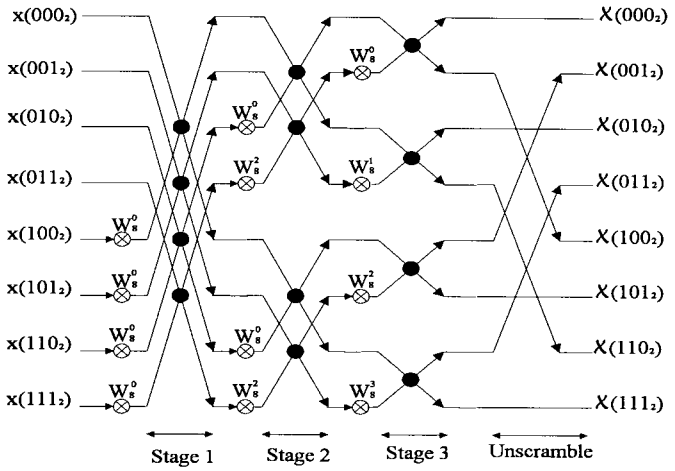

Fig. 1. Signal flowgraph for 8-point modified DIT-FFT with output scrambling.

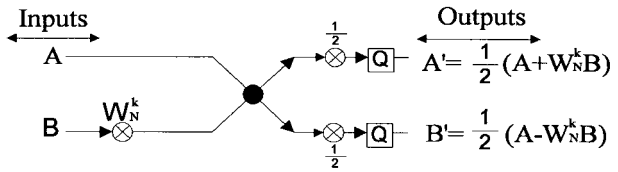

Fig. 2. Signal flowgraph for the DIT-FFT butterfly [10].

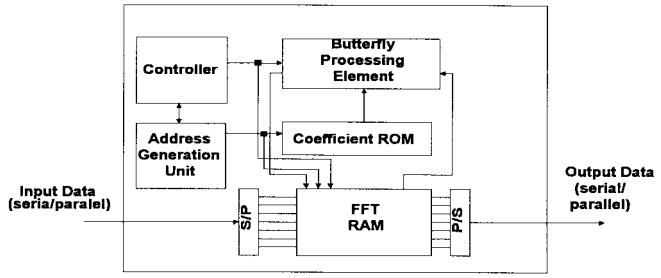

Fig. 3. Block diagram representation of the FFT processor.

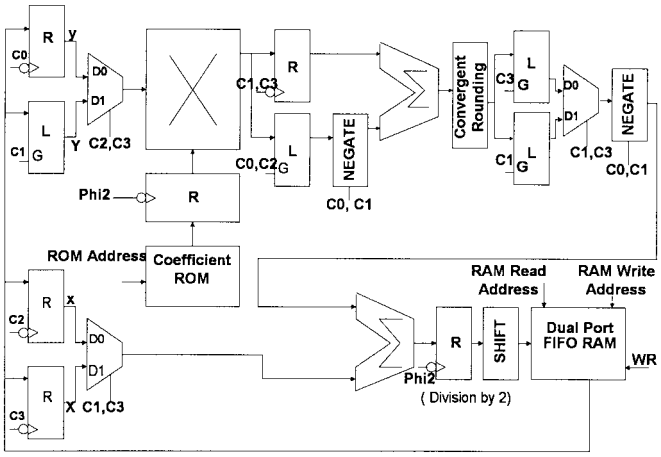

Fig. 4. Butterfly processing element architecture.

is bias free. The DIT-FFT radix-2 butterfly is shown in Fig. 2 [10].

The DIT-FFT butterfly takes a pair of complex input data values "A" and "B" and produces a pair of complex outputs "A'" and "B'" where

$$A' = \tfrac{1}{2}(A + BW_N^k) \quad (4)$$

$$B' = \tfrac{1}{2}(A - BW_N^k). \quad (5)$$

III. Architectural Design of the FFT Processor

The architecture of our FFT processor can best be understood, by tracing through its operation. The operation of the FFT processor is first partitioned into three main processes: data input, FFT computation, and data output. The operation cycle starts with the data input process, during which sampled data is read in and stored in memory. During the FFT Computation process, the FFT or inverse FFT (IFFT) is computed on the stored data. During the output process results of the FFT computation process are read out to the outside world.

The FFT processor architecture consists of a single DIT-FFT radix-2 butterfly (which is referred to as the butterfly processing element or butterfly PE), a dual-port FIFO RAM, a coefficient ROM, a controller and an address generation unit. Data pathways are in the form of 24-bit two's complement signed fractions. Coefficients are stored as 16-bit two's complement signed fractions. The block diagram representation of the FFT processor is depicted in Fig. 3.

A. Butterfly Processing Element

The butterfly is the core calculation of the FFT and computes a two point FFT. The entire FFT is performed by combining butterflies in patterns determined by the FFT algorithm. The butterfly PE takes two complex data words from memory and computes the FFT on them. Results are written back to the same memory locations of the inputs since an in-place algorithm is used. This makes efficient use of the available memory as the transformed data overwrites the input data. However, the indexing required to determine which location in memory to fetch the input data for each butterfly is quite complex.

The structure of the butterfly PE employing straightforward implementation of (4) and (5) using standard real arithmetic units requires four multipliers, four adders, and two subtractors. This level of complexity makes it unsuitable for silicon implementation. A novel silicon area/computation-time efficient architecture is depicted in Fig. 4.

The butterfly PE is capable of computing one butterfly operation every four cycles. It comprises one parallel–parallel multiplier and two adders. At each computation cycle the multiplier generates partial products of the complex multiplication $B.W_N^k$, i.e., $b\cos\Phi$, $B\cos\Phi$, $b\sin\Phi$, and $B\sin\Phi$. These results are in 40-bit two's complement signed fraction format.

TABLE I
BUTTERFLY PE SEQUENCE TABLE WHERE butterfly $= b6b5b4b3b2b1b0$

Cycle	RAM read	ROM Read	Multip O/P	First Adder O/P	2nd Adder O/P	RAM write
C0	y	cosΦ	previous YcosΦ	settling	prev y'	prev X'
C1	Y	sinΦ	ycosΦ	previous YcosΦ-ysinΦ	prev Y'	prev y'
C2	x	cosΦ	YsinΦ	settling	prev x'	prev Y'
C3	X	sinΦ	YcosΦ	ycosΦ+ YsinΦ	prev X'	prev x'
C0	next y	cosΦ	ysinΦ	settling	y'	prev X'
C1	next Y	sinΦ	next ycosΦ	Y cosΦ- ysinΦ	Y'	y'
C2	next x	cosΦ	next YsinΦ	settling	x'	Y'
C3	next X	sinΦ	next YcosΦ	next y cosΦ+ YsinΦ	X'	x'
C0	next y	cosΦ	ycosΦ	settling	next y'	X'

Since, computation of the twiddle factors is time consuming these are pre-calculated and stored in the coefficient ROM. Only half a cycle of $-\cos(2\pi p/256)$ is stored, i.e., range of [−1 1) with p varying from 0 to 127. These are stored as 16-bit two's complement-signed-fractions format. The first adder is 40 bits wide and sums the cross products of the complex multiplication to generate the sum/difference of cross products. The output of this adder is rounded to a 26-bit result using convergent rounding. However, the "retain" (a variable used in convergent rounding) word is not incremented at this stage. Instead it is put back as 1 LSB and propagated until the second adder. It is combined together with the "negate" signal of the following negator and evaluated at the second adder hence saving us a 26-bit full adder otherwise required to increment the "retain" word. The second adder produces the sum and difference outputs of the butterfly computation. These results are scaled by 1/2, i.e., 1-bit right shift and clipped if the results overflow.

Implementing the butterfly PE in this way leads to an increase in computational speed at a cost of increased silicon area relative to using a serial–parallel multiplier. However, bearing the length of the transform in mind, to achieve high throughput and high speed of operation this trade-off is cost effective. The butterfly PE takes four cycles to compute a two-point FFT, with a latency of five cycles. Three of these are associated with the fact that three input components (y, Y, and x) are required before an output can be computed and two are used to pipeline the RAM *read* and *write* operations. Thus, the write and access times of the RAM are not a critical path of the operation of the processor. The target speed for the processor is a clock frequency of 40 MHz which results in a butterfly computation of 100 ns. Allowing for the pipeline delay, the total computation time for a 256-point complex FFT is 102.4 μs excluding data input and output processes. The butterfly PE sequence is shown in Table I.

B. Address Generation Unit

The purpose of the address generation unit (AGU) is to provide the RAM and the coefficient ROM with the correct addresses. It also keeps track of which butterfly is being computed in which stage. In addition to this, since address generation during input, output, and FFT computation processes are different it keeps track of the mode of operation of the processor and generates the required addresses. A block level description of the AGU is shown in Fig. 5.

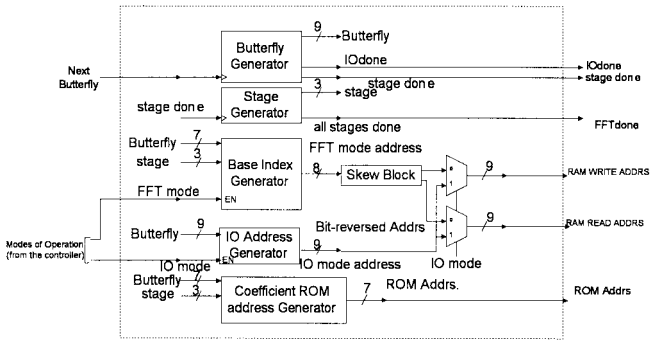

Fig. 5. Block diagram representation of AGU.

The butterfly generator keeps track of which butterfly is being computed in a particular stage. It is basically a nine-bit up counter, since for a 256-point complex FFT there are 128 butterflies per stage and four data words per butterfly (two real and two imaginary). The counter output is used for addressing the RAM during input and output processes and for providing the basic timing for the FFT process.

The stage generator keeps track of the current stage in the FFT computation, and supplies the base index generator with the number of the stage that is currently being computed. For a 256-point FFT, there are eight stages, and hence the stage generator is basically a three-bit counter which is incremented once every 128 butterfly counts.

The IO address generator is responsible for generating addresses for the RAM during the data input and output processes. During the data input process the output of the butterfly generator, "butterfly" can be used for addressing 512 locations in the RAM. However, during the data output process data should be bit-reversed while being written to the outside world. No extra hardware is required for implementing the bit-reversing in our hardware, as we simply reverse the wiring.

The base index generator is responsible for generating addresses during the FFT computation mode. FFT mode address generation is quite complex. The butterfly has two complex data inputs A and B. A is referred to as "index0" and B as "index1." "Index1" can be calculated from index0 by [8]

$$\text{index1} = \text{index0} + P \tag{6}$$

where P is the index spacing which can be expressed as $2^{N/s}$ where s is the stage number and N is the transform length. Also "index0" can be expressed as [8]

$$\text{index0} = 2^{*}P^{*}(\text{butterfly DIV } P) + \text{butterfly MOD } P \tag{7}$$

where "butterfly" consists of the first seven bits of the butterfly generator, i.e., butterfly $= b6b5b4b3b2b1b0$, and "index0" is

TABLE II
BINARY REPRESENTATION OF THE INDEX0 IN A 256 POINT DIT-FFT [8]

stage ←	index0 →							
1	0	b6	b5	b4	b3	b2	b1	b0
2	b6	0	b5	b4	b3	b2	b1	b0
3	b6	b5	0	b4	b3	b2	b1	b0
4	b6	b5	b4	0	b3	b2	b1	b0
5	b6	b5	b4	b3	0	b2	b1	b0
6	b6	b5	b4	b3	b2	0	b1	b0
7	b6	b5	b4	b3	b2	b1	0	b0
8	b6	b5	b4	b3	b2	b1	b0	0

the 8-bit wide RAM address. Table II shows the calculated "index0" for a 256-point FFT computation for each stage [8].

As shown in Table II, "index0" can be derived by simply rearranging the bits in the "butterfly" and inserting zeros in the leading diagonal. "Index1" can simply be obtained from "index0" by replacing zeros on the leading diagonal with ones [8].

C. Controller

The sequence of events is determined by the controller depending on the signals it receives from the surrounding units and generates information about which mode the chip is in, i.e., input, output, or FFT computation. This is important since address generation in each mode is different. It also generates other control signals to take care of required house keeping duties, i.e., incrementing and clearing counters.

IV. SIMULATION

The whole architecture has been simulated at the logic level using simulation models generated with Cascade Design Automation's EPOCH silicon compilation tool. These models included extracted and back annotated capacitive trackload models. The MENTOR Quicksim package was used to carry out the extracted and back annotated simulations. During simulations a variety of test signals including cosine and sine waves at different frequencies were fed into the in-phase and quadreture channels of the FFT processor. All the logic simulation results in all modes of operation proved to be satisfactory, and complied to the expected outputs, giving us the green light to go for fabrication.

V. CONCLUSION

As the FFT processor was designed and optimized for performing high-speed sum-of-products operations, it is easily deployable in a variety of DSP based sum-of-products intensive instrumentation and measurement applications, such as correlation, convolution and digital filtering. The processor is implemented in silicon based 0.7 μm CMOS technology. The size of the chip (including pads) is 3.7 mm $\times$ 4.1 mm, i.e., 15.17 mm^2. The size without the pads is 2.7 mm $\times$ 3.2 mm, i.e., 8.64 mm^2. The overall FFT chip plot can be seen in Fig. 6.

The chip architecture consists of a bit-parallel radix-2 DIT-FFT butterfly, dual-port FIFO RAM, address generation unit and the controller. Separate memories are used for storing the data and the coefficients. Although the processor we have designed and reported here is quite small and fast, there are some improvements that can be made. Most of the cells used to build the FFT processor have been optimized for speed rather than area and power consumption. These blocks can be redesigned for reduced area and power consumption. Also, investigation into the use of more than one butterfly processing element is another possibility. The FFT processor is capable of computing a 256-point complex FFT in 102.4 μs excluding data input and output processes. The chip is operating with a clock frequency of 40 MHz.

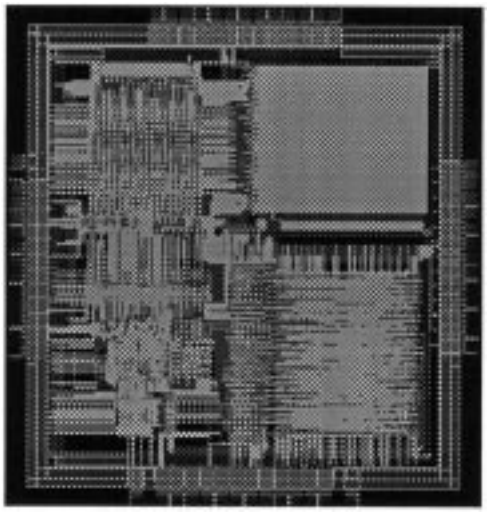

Fig. 6. FFT chip plot.

ACKNOWLEDGMENT

The authors would like to thank Cascade Design Automation for the EPOCH silicon compilation tool.

REFERENCES

[1] R. Blasco-Gimenez, G. M. Asher, M. Summer, and K. J. Bradley, "Performance of FFT-rotor slot harmonic speed detector for sensorless induction motor drivers," *IEE Proc.—Elect. Power Applicat.*, vol. 143, pp. 258–268, May 1996.

[2] P. G. Flikkema and S. G. Johnson, "Vehicle collision warning and avoidance system using real-time FFT," in *IEEE 46th Vehicular Technol. Conf., Mobile Technol. Human Race*, 1996, vol. 3, pp. 1820–1824.

[3] G. A. Zimmerman, M. F. Garyantes, and M. J. Grimm, "A 640 MHz 32 megachannel real-time polyphase-FFT spectrum analyzer," in *Conf. Rec. 25th Asilomar Conf. Signals, Systems, Computers*, 1991, vol. 1, pp. 106–110.

[4] V. D. Lecce and D. E. Sciascio, "A VLSI implementation of a novel bit-serial butterfly processor for FFT," in *Proc. Advanced Computer Technology, Reliable Systems Applications Proc. 5th Eur. Comput. Conf. Adv. Comput. Technol. Reliable. Syst. Appl. Comp. Euro'91*, 1991, pp. 875–879.

[5] T. Chen and L. Zho, "An expandable column FFT architecture using circuit switching networks," *J. VLSI Signal Process.*, vol. 6, pp. 243–257, Dec. 1993.

[6] V. Szwarc, L. Desormeaux, W. Wong, S. P. C. Yeung, H. C. Chan, and A. T. Kwasnievski, "A chipset for pipeline and parallel pipeline FFT architectures," *J. VLSI Signal Process.*, vol. 6, pp. 253–265, Dec. 1994.

[7] R. Storn, "Radix-2 FFT-pipeline architecture with reduced noise to signal ratio," *IEE Proc.—Vision, Image, Signal Process.*, vol. 141, pp. 81–88, Apr. 1994.

[8] J. Melander, T. Widhe, P. Sanbarg, K. Palmkvist, M. Vesterbacka, and L. Wanhammar, "Implementation of a bit-serial FFT processor with a

hierarchical control structure," in *Proce. EECCTD'95 Eur. Conf. Circuit Theory Design*, Sept. 1995, pp. 423–426.
[9] B. J. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, 2nd ed. New York: Macmillan, 1992.
[10] W. B. Jervis and E. C. Ifeachor, *Digital Signal Processing: A Practical Approach.* Reading, MA: Addison-Wesley, 1993.

Ediz Çetin was born in Lefkosa, Cyprus, in 1976. He received the B.Eng. (Hons.) degree in control and computer engineering in 1996 from the University of Westminster, London, U.K., where he is now pursuing the Ph.D. degree.

His research interests include rapid prototyping of digital signal processors, VHDL, digital signal processing, silicon systems design, and global positioning systems. He has worked on design and silicon implementation of a $\Sigma\Delta$ CODEC for GSM applications.

Richard C. S. Morling (M'79) was born in Rochford, U.K. He received the B.Sc. (honors) degree in physics from the Polytechnic of Central London, London, U.K., in 1971, and the Ph.D. degree in information engineering from the City University, London, in 1989.

He worked in the field of television for Decca Records Ltd., and, during a period at the Imperial College of Science and Technology, London, was engaged in the design of electro-myographic equipment for the Migraine Trust. He joined the staff of the University of Westminster (formerly The Polytechnic of Central London) in 1971. He is currently Director of the Division of Electronic Systems. His research activities have included packet-switching local-area networks, discrete-time signal processing structures, aircraft braking systems, design methodologies for integrated circuit design, sigma–delta conversion techniques and teaching methods in electronic engineering. He is currently working on the theory and practical realization of high-fidelity sigma–delta A/D and D/A converters.

İzzet Kale (M'88), for a photograph and biography, see this issue, p. 44.