Design and Synthesis of Radix-4 FFT Processor using Vedic Multiplier

1.PANTHAGANI PRAVEEN KUMAR, Pkumar806@gmail.com,RISE GANDHI GROUP OF INSTITUTIONS VALLURU, ONGOLE.

2. S.CHANDRA SEKHAR, HOD OF ECE &ASSOCIATE PROFESSOR ,RISE GANDHI GROUP OF INSTITUTIONS VALLURU, ONGOLE.

Abstract - The FFT processor is a critical block in all multi-carrier systems used primarily in the mobile environment. The portability requirement of these systems is mainly responsible for the need of low power FFT architectures. In this study, an efficient addressing scheme for radix-4 64 point FFT processor is presented. It avoids the modulo-r addition in the address generation; hence, the critical path is significantly shorter than the conventional radix-4 pipelined FFT processor by modifying its operation sequence. The complex multiplier is one of the most power consuming blocks in the FFT processor. A significant property of the proposed method is that the critical path of the address generator is independent from the FFT transform length N, making it extremely efficient for large FFT transforms. The results confirm the speed and area advantages for large FFTs. Although only radix-4 FFT address generation is presented in the paper, it can be used for higher radix FFT.

Index Terms - Pipelined FFT, Switching activity. Coefficient ordering

I.INTRODUCTION

Fast Fourier transform (FFT) is one of the key components for various signal processing and communications applications such as software defined radio and OFDM. A typical FFT processor is composed of butterfly calculation units, an address generator and memory units. This study is primarily concerned with improving the performance of the address generation unit of the FFT processor by eliminating the complex critical path components.

Pease observed that the two data addresses of every butterfly differ in their parity. Parity check can be realized by modulo-r addition in hardware. Based on Pease's observation, Cohen proposed simplified control logic for radix-2 FFT address generation. Johnson proposed a similar way to realize radix-r FFT addressing. In

This method, the address generator is composed of several counters, barrel shifters, multiplexers and adder units. Other FFT processors have been designed to realize high-radix FFT. A common drawback of all these methods is the need for successive addition operations to realize the address generation. The number of addition operations depends on the length of the FFT, so the address generation speed is slower as the FFT transform length increases. Several methods have been proposed to avoid the addition for radix-2 FFT but these methods cannot be used for higher radix FFT. This study presents a new architecture to realize the address generation for radix-4 FFT. The new address generator is composed of counters, barrel shifters, multiplexers and registers, but no addition operation is required. The critical path of the address generator is shorter, and furthermore, the critical path of this address generator is independent of the FFT length making it extremely efficient for long length FFT.

II. PARTIAL-COLUMN RADIX-2 AND RADIX-2/4 FFTS

Parallel FFT processors can be divided into the following principal classes' fully parallel, pipelined, column, and partial-column. In general, the pipelined FFTs have been popular due to their principal simplicity. The advantage in using partial-column organization is that partial column processing is saleable where as in pipeline and column FFTs the number of butterfly units is dependent on the FFT size. The partial-column organization can also be combined to pipeline class, which results in parallel pipelines as proposed,. The high-level organization of the FFT processor is not the only characteristic, which defines the key properties of the implementation. Most of the power in FFT processor is consumed by the butterfly operations. Therefore, the power optimizations should be performed for butterfly units. Butterfly units can be realized with several different principles'. Butterfly units based on bit-parallel multipliers, CORDIC, and DA have been reported. The organization of the

proposed energy-efficient partial-column FFT processor is described in the following sections.

2.1. Organization

In general, in the partial-column processing all operands to the butterfly computations are transferred simultaneously from the memory implying need for high memory bandwidth. In our approach, the butterfly units are pipelined in a sense that a single operand (2*K*-bit word if real and imaginary parts take *K* bits each) is transferred to the butterfly unit at a time, thus each butterfly unit has a dedicated bus to and from memory. Such an arrangement increases the computation time but this can be compensated by increasing the number of butterfly units. Our approach is to minimize the RAM storage, thus the computation is performed in-place, i.e., results of butterfly units are stored into the same memory locations they were obtained. Therefore, *N* complex-valued memory

Locations are needed for an *N*-point FFT. The organization requires that there are 2*M*-ports in the RAM memory.

# of Multipliers	Area [gates]		Power [mW]		Min. Critical Path [ns]
	50 MHz	100 MHz	50 MHz	100 MHz	
4, Eqn. (6)	6996	9483	2,96	4,22	8
3, Eqn. (7)	5616	8149	2,94	4,34	9
2, Eqn. (8)	3915	5181	1,44	2,13	8

Table 1. Characteristics of 16-bit complex multipliers on an 0.11μm ASIC technology.

When *M* butterfly units with throughput of one is used. This can be arranged with multi-port memories, but more area-efficient approach is to use interleaved memories with a conflict-free access scheme. This arrangement requires that for an *N*-point FFT, there are 2*M* single-port memories with *N*/2*M* words and the memories are interconnected with a permutation network

Butterfly operation and these are discussed in the following sections.

III. PROPOSED METHODRADIX 4

The *N*-point discrete Fourier transform is definedby

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

The *N*-point FFT can be decomposed to repeated micro operations called *butterfly* operations. When the size of the butterfly is *r*, the FFT operation is called a radix-*r* FFT. For FFT hardware realization, if only one butterfly structure is implemented in the chip, this butterfly unit will execute all the calculations recursively. If parallel and pipeline processing techniques are used, an *N* point radix-*r* FFT can be executed by $\frac{N}{r}\log_r N$ clock cycles. This indicates that a radix-4 FFT can be four times faster than a radix-2 FFT. Fig.1 shows the signal flow graph of 64-point radix-4 FFT, and Fig. 2 shows the general structure of the radix-4 butterfly. For hardware realization of FFT, multi-bank memory and "in place" addressing strategy are often used to speed-up the memory access time and minimize the hardware consumption. For radix-*r* FFT, *r* banks of memory are needed to store data, and each memory bank could be two-port memory. With "in-place" strategy, the *r* outputs of the butterfly can be written back to the same memory locations of the *r* inputs, and replace the old data. In this case, to realize parallel and pipelined FFT processing, an efficient addressing scheme is needed to avoid the data conflict. A popular addressing scheme for radix-*r* (*r*>2) was presented by Johnson, however due to the modulo-*r* addition, this method is slow and the speed depends on the length of FFT.

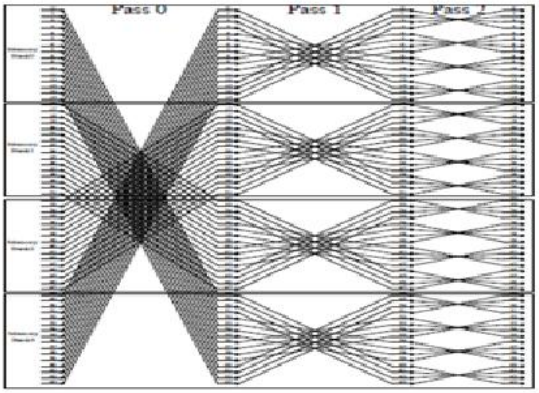


Fig 2: General Structure of Radix 4 butterfly unit

IV ALGORITHM

The N-point DFT of a finite duration sequence x (n) is defined by (2) as follows.

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

Where $W_N=e^{-j(2\pi/N)}$

Let N be a composite number of v integers so that N = r3r2...r, and define

$$N_t=N/r_1r_2\ldots r_t \qquad 1\le t\le v-1$$

Where t is the stage number of the decomposed DFT and rt its radix. The pipelined FFT processor is obtained by decomposing an N-point DFT into v stages. The final stage is defined in as follows.

$$X(r_1r_2\ldots r_{v-1}m_v+\ldots+m)=\sum_{q_{v-1}=0}^{r_v-1}x_{v-1}(q_{v-1},m_{v-1})W_{r_v}^{q_{v-1}m_v} \qquad (3)$$

Whereas intermediate stages (t) are given by the following recursive equation

$$x_t(q_t,m_t)=W_{N_{t-1}}^{q_tm_t}\sum_{p=0}^{r_t-1}x_{t-1}(N_tp+q_t,m_{t-1})W_{r_t}^{pm_t} \qquad (4)$$

Where $2 \le t \le v\text{-}1$, $0 \le m_i \le r_i\text{-}1$, $0 \le q_i \le N_i\text{-}1$ and $2 \le i \le v$

For rl = 4, the flow graph of a 16-point FFT based on the above formulation is shown in Fig. 2. The corresponding equations are as follows.

$$X(4m_2 + m_1) = \sum_{q_1=0}^{3} x_1(q_1, m_1) W_4^{q_1 m_2} \qquad (5)$$

$$x_1(q_1, m_1) = W_{16}^{q_1 m_1} \sum_{p=0}^{3} x_1(4p + q_1) W_4^{pm_1} \qquad (6)$$

In Fig. 3, each open circle represents the summation while the dots define the stage boundaries. The number inside the open circle is the value of ml(for stage 1) or m2 (for stage 2). The number outside the open circle is the FFT coefficient applied.

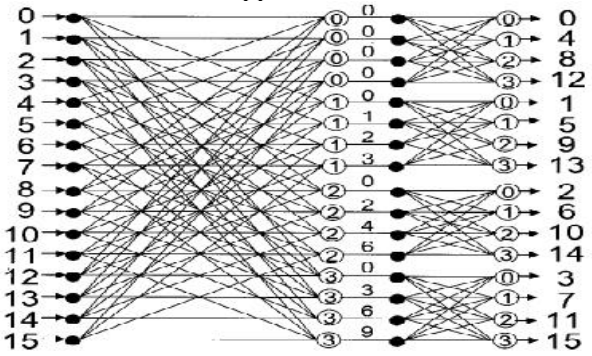
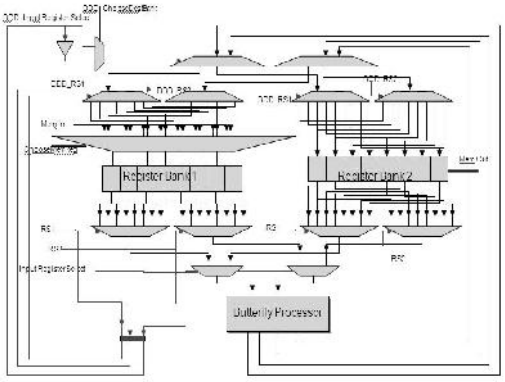


Fig.3.Signal flow graph of a radix-4 16 point FFT

V ARCHITCTURAL DESIGN

The FFT processor will calculate a 64-point FFT on incoming data. For the radix 4 algorithm, the first 3 stages of the flow graph involve choosing every 8th term to yield 8 octets, and for the last 3 stages, every successive octet makes up the input the FFT processor. The Figure 2 is the flow graph for the decimation in frequency algorithm

The FFT data-path is as shown below, from the point data enters the processor module from memory, to the point where it is written back to memory. Red lines represent the control signals and their delayed versions. A 'D' is prefixed to represent the delayed versions of the original signal, each D signifies a clock delay. The pipeline is 4 stages long, and completes 3 stages of the FFT calculation before writing the data the data back to memory. Another point to note is that data is always written out to the memory from Register Bank 2, and it is always read to Register bank 1. The two register banks allow 2 octets to be in the pipeline at any given time. In place computations make it a simplified design. The outputs of the FFT computation are in bit-reversed order, and need to be shuffled back into normal order. The results of butterfly computations are scaled down by a factor of 2 to avoid arithmetic overflow. The 64 point FFT takes a total of 196 cycles. Clocking the processor at 40 MHz will result a latency of about 2 microseconds.



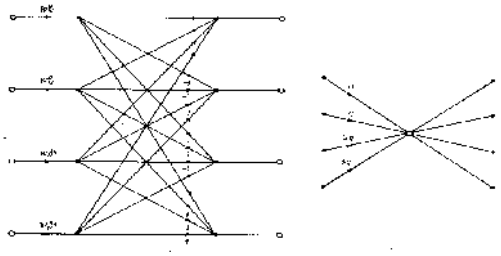
The FFT processor has a modular design and comprises of 3 modules.

1. Butterfly Processor
2. Address Generation Unit (AGU)
3. Micro-Coded State Machine(MCSM)

1. Butterfly Processor:

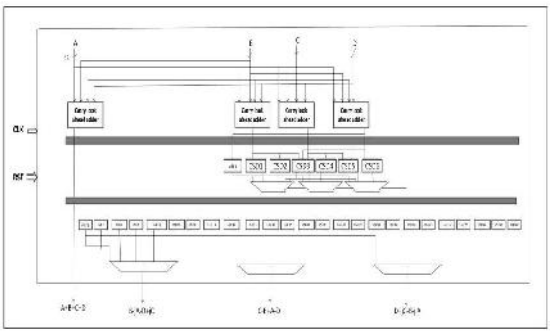
The Butterfly processor's task is to carry out the complex butterfly computation.

$$X(k) = \sum_{n=0}^{3} x(n) \cdot W^{nk} (k = 0,1,2,3)$$



Where $W_N^r = \exp(-2\pi j\, r/N)$

To avoid using a complete digital multiplier to carry out multiplication with the twiddle factors, we used CSD (canonical signed digit multiplication, using shifts and adds). The results desired multiplication is controlled by 2 stages of multiplexing feeding into the CSD's. The butterfly processor has two pipelined stages. The data-path of the butterfly processor is shown below.



2.Address Generation Unit (AGU): The address generation unit controls the address bus going to memory. The FFT processor reads and writes from and to the 8 dual port memory banks concurrently (each address is 3 bits). The address mapping scheme ensures that no memory location is read from and written to at the same time. There are 8 read address buses, and 8 write address buses. The computations are in place, which simplifies the address generation unit.

Bank 0	Bank 1	Bank 2	Bank 3	Bank 4	Bank 5	Bank 6	Bank 7
0	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
22	23	16	17	18	19	20	21
29	30	31	24	25	26	27	28
36	37	38	39	32	33	34	35
43	44	45	46	47	40	41	42
50	51	52	53	54	55	48	49
57	58	59	60	61	62	63	56

3.Micro-coded State Machine: The Micro-coded state machine stores and generates all the control signals for the FFT processor's operation at every, its progression is controlled by the clock. A reset signal en_fft resets the state machine counter and signals the beginning of a new 64-point FFT calculation. Upon completion the FFT processor asserts a done_fft signal to communicate the completion of the 64-point FFT. The number of states is 196

To perform an Inverse Fast Fourier Transform, all we need to do is swap the real and imaginary parts

VI. CONCLUSION

This paper presented a new, very high speed FFT architecture based on the Radix-43 algorithm. A fully pipelined, systolic processing core of a 64-point FFT has been implemented in both FPGA and standard cell technologies and validated in the former case. The results demonstrate the very high operating frequencies and the low latencies of both the FPGA and VLSI implementations. The proposed FFT architecture demonstrates a significant latency reduction compared to existing solutions and at the same time, has reduced data memory required and improved multiplier utilization while occupying a smaller silicon area occupation consuming less
power compared to similar solutions. The modular design of the Radix-4^3 allows them to be easily incorporated into larger systems for computing large scale FFTs while a fully registered, systolic architecture assures maximum operating frequency. Future research by our group will focus on the implementation of a reconfigurable FFT architecture, capable of performing the FFT transform of 64, 4K, 256K or 16M complex points.

REFERENCES

[1] S. Mittal, Z.A. Khan, and M.B. Srinivas, "Area efficient high speed architecture of Bruun's FFT for software defined radio", *IEEE Global Telecommunications Conference GLOBECOM '07,* pages 3118 -3122, November 2007.

[2] L. Xiaojin, L. Zongsheng Lai, and C. Jianmin Cui, "A low power and small area FFT processor for OFDM demodulator", *IEEE Transactions on Consumer Electronics,* 53(2):274-277, May 2007.

[3] M. C. Pease, "Organization of large scale Fourier processors", *J. Assoc. Comput. Mach.,* 16:474–482, July 1969.

[4] D. Cohen, "Simplified control of FFT hardware", *IEEE Trans. Acoust., Speech, Signal Processing*, 24:577-579, December 1976.

[5] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline fft processor," in *Proc. IEEE Custom Integrated Circuits Conf.*, Santa Clara, CA, May 11-14 1998, vol. 2, pp. 131–134.

[6] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline fft processors for vlsi implementations," *IEEE Trans. Comput.*, vol. 33, no. 5, pp. 414– 426, May 1984.

[7] M. Hasan and T. Arslan, "Implementation of lowpower fft processor cores using a novel order-base processing scheme," in *Proc. IEEE Circuits Devices Syst.*, June 2003, vol. 150, pp. 149–154.

[8] M. Wosnitza, M. Cavadini, M. Thaler, and G. Tr¨oster, "A high precision 1024-point fft processor for 2d convolution," in *Dig. Tech Papers IEEE Solid-State Circuits Conf.*, San Francisco, CA, Feb. 5-7 1998, pp. 118–119.

[9] A. M. Despain, "Fourier transform computers using cordic iterations," *IEEE Trans. Comput.*, vol. 23, no. 10, pp. 993–1001, Oct 1974.

[10] A. Berkeman, V. Owall, and M. Torkelson, "A low logic depth complex multiplier using distributed arithmetic," *IEEE Solid-State Circuits*, vol. 35, no. 4, pp. 656–659, Apr. 2000.