Split-Radix FFT Processors Using Radix-4 Butterfly Units

Sonali D. Patil [1], Manish Sharma [2]
[1, 2]Electronics and Telecommunication
[1, 2]D.Y. Patil College of Engineering, Akurdi, Pune, India

Abstract- *For the implementation of low-power FFT processor, Split-radix fast Fourier transform (SRFFT) is widely used. SRFFT uses the lowest number of arithmetic operations amongst all the FFT algorithms. Split-radix FFT has the same signal flow graph that of conventional radix-2 and radix-4 FFT. Therefore, the address generation method could also be applied to the SRFFT. In this, a shared-memory low-power SRFFT processor architecture is presented. Here, it is shown that the SRFFT is computed using radix-4 butterfly unit. Dynamic power is saved at the cost of using more hardware resources. In this paper, we have increased the architecture size, from radix-2 to radix-4 and 2048 point complex valued transform, and shown the performance of area, power and delay.*

Keywords- Butterfly unit, low power, radix-4, split-radix fast Fourier transform (SRFFT), twiddle factors

I. INTRODUCTION

In digital signal processing (DSP) area, fast fourier transform (FFT) algorithm plays very essential and fundamental role. Since the invention of FFT, various types of FFT have been developed, such as radix-2, radix-4 and radix-8 FFT. In the year 1984, Duhamel and Hollmann [1] invented a new variant of FFT algorithm called as split-radix FFT (SRFFT). SRFFT requires least number of multiplications and additions amongst all the known FFT algorithm. Arithmetic operations significantly contribute to overall system power consumption, SRFFT is a good candidate for the implementation of a low-power FFT processor.

FFT processors are of two types: pipelined processors and shared-memory [2] and [3] processors. Pipelines architectures produce high throughputs, but needs more hardware resources. Opposite to this, shared-memory architecture requires very less hardware resources, but, produces slow throughputs. FFT data in radix-2 shared-memory architecture are arranged into two memory banks. On every clock cycle, two FFT data are given by memory banks and one butterfly unit is used to process the data. At the next clock cycle, the results are written back to the memory banks and the old data is replaced. This process is limited to the shared-memory only.

An efficient address generation scheme for FFT data as well as twiddle factors is required in shared-memory architecture. Split-radix FFT involves an L-shaped butterfly data-path. In this, we show that the SRFFT can be computed by using a modified radix-4 butterfly structure.

The remaining paper is organized as follows. Section II provides the background of complex multiplications between the radix-2 FFT and the SRFFT. Section III discusses the architecture of the proposed system. Section IV provides the implementation and comparison results and Section V concludes this paper.

II. BACKGROUND

A. Conventional Radix-2 FFT processor

The N-point discrete Fourier transform is defined by

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

Where k = 0,1,.....N-1 and $W_N^{nk} = e^{-2\pi nk/N}$. If we separate the even and odd terms, radix-2 is obtained as

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) + x(n + N/2)] W_{N/2}^{nk} \quad (2)$$

$$X(2k+1) = \sum_{n=0}^{\frac{N}{2}-1} [x(n) - x(n + N/2)] W_N^n W_{N/2}^{nk} \quad (3)$$

The basic idea behind SRFFT is the radix-2 index map to the even-index terms and a radix-4 map to the odd-index terms. For the even index terms it can be obtained as (2). For the odd one's, it can be obtained as

$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} [x(n) - x\left(n + \frac{N}{2}\right) - j(x\left(n + \frac{N}{4}\right) - x(n + 3N/4))] W_N^n W_{N/4}^{nk} \quad (4)$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} [x(n) - x\left(n + \frac{N}{2}\right) + j(x\left(n + \frac{N}{4}\right) - x(n + 3N/4))] W_N^n W_{N/4}^{nk} \quad (5)$$

Where k = 0,1,…….N/4. The above formulas produce L-shaped split-radix butterfly structure. If we have $N = 2^S$ point FFT, both SRFFT and radix-2 requires S passes to finish the computation, as seen in Figs. 1 and 2. The total number of L butterflies for SRFFT N_{SR} is given by [2]

$$N_{SR} = [(3S-2)2^{S-1} + (-1)^S]/9 \qquad (6)$$

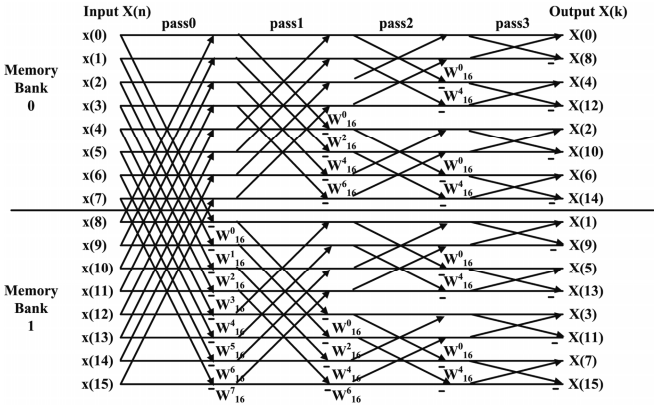

Fig. 1. Signal flow graph for radix-2 FFT.

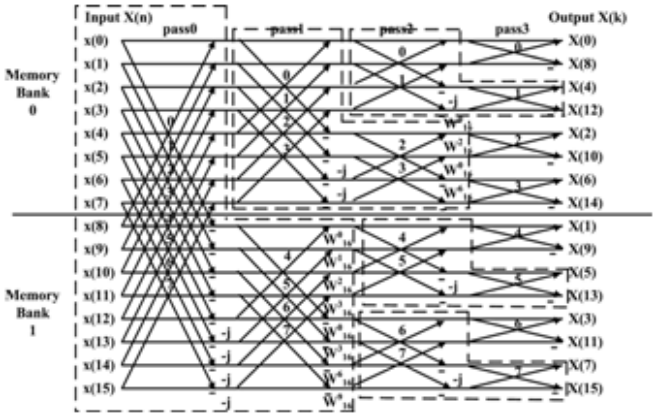

Fig. 2. Signal flow graph for SRFFT.

Each L butterfly contains two nontrivial complex multiplications, and therefore, the total number of nontrivial complex multiplications MSR in SRFFT is

$$MSR = [(3S - 2)2S-1 + (-1)S]2/9. \qquad (7)$$

In the (S − 1)th pass, the number of SR butterfly NS−1 is

$$NS-1 = [2 + (-1/2)S-2]N/12 . \qquad (8)$$

However, in the (S 1)th pass, each L butterfly does not contain any nontrivial twiddle factors and hence, the total number of nontrivial multiplications M'_{SR} in SRFFT is

$$M'_{SR} = M_{SR} - 2N_{S-1} \qquad (9)$$

For the conventional radix-2 FFT, the total number of complex multiplications M_{R2} is

$$MR2 = 2S-1(S - 1). \qquad (10)$$

B. Shared-Memory Architecture

Fig. 3 shows the architecture of shared-memory processor. The RAM and ROM banks, respectively stores the FFT data and twiddle factors. It is seen that the flow graph of split- radix algorithm is the same as radix-2 FFT therefore, the conventional radix-2 FFT data address generation methods could also be applied to SRFFT.

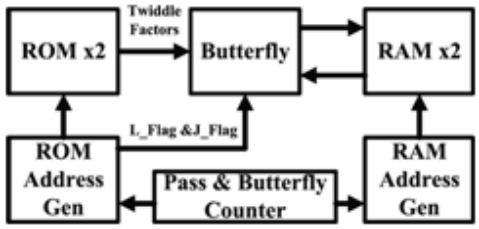

Fig. 3. Shared-memory architecture

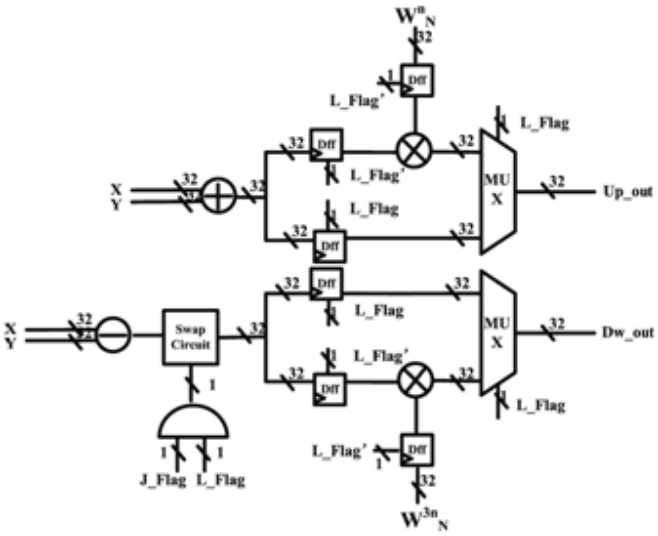

Fig. 4. Modified butterfly structure.

C. Modified Radix-4 butterfly Unit

We have proposed a modified butterfly unit which is shown in Fig. 4. The SRFFT has multiplications of both the upper and lower legs. For the unnecessary switching activity, we have put the clock gating registers in the multiplier path and a few registers are placed at the address port of memory banks to synchronize the whole design. The idea to use this architecture is to know about which butterflies need no multiplications, trivial multiplications those are swapped one's, and non-trivial multiplications.

E. Address Generation of Twiddle Factors

Fig. 2 shows the flow graph for 16-point SRFFT. There are two types of twiddle factor j and W_n. Operations involving j are called trivial multiplications and those involving W_n are said to nontrivial. Each area surrounded by dashed line represents one L block [8]. And there are total five L blocks for a 16-point SRFFT.

III. PROPOSED SYSTEM

In this paper, we have increased the size of radix-2 to radix-4, and also increased the point 1024 to 2048, with complex valued transform, and show the performance of area, power and delay. The FFT uses Radix-4 burst I/O engine to process butterfly unit [6]. Data is separately loaded and unloaded from calculating the transform.

Data I/O and processing are not simultaneous. The data gets loaded after FFT is started. After the loading of a full frame, the core computes the transform. When the computation has finished, the data can be unloaded, but cannot be loaded or unloaded during the calculation process. The data loading/unloading processes gets overlapped if the data is unloaded in digit reversed order. This architecture uses lowest resource than the Pipelined, Streaming I/O architecture, but a longer transform time. Data and phase factors can be stored in RAM.

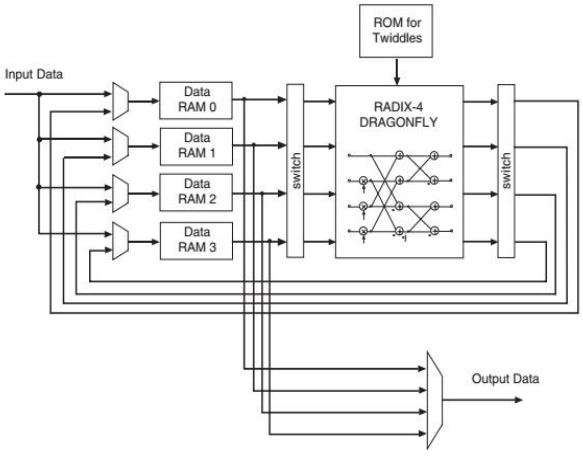

Fig. 5 Proposed radix-4 FFT system

A. Block Diagram of Proposed System

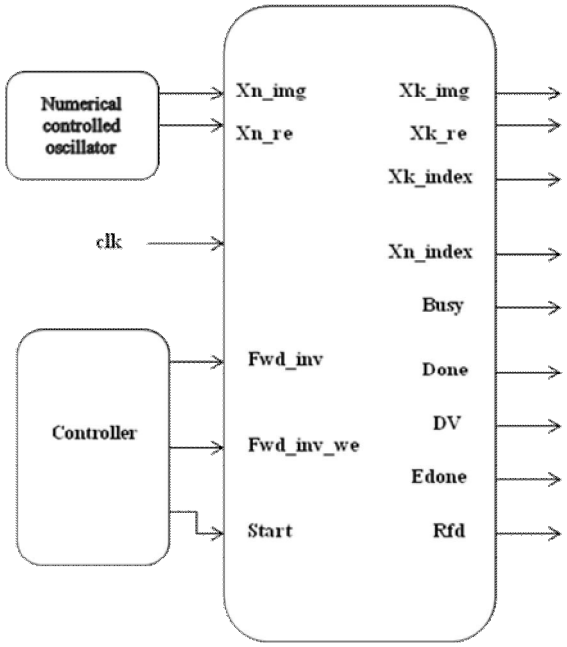

Fig 6. Block Diagram of Proposed System

In this system the input is given from the NCO to FFT Block of X_n. NCO generates the sine signal.

C. Numerical Control Oscillator

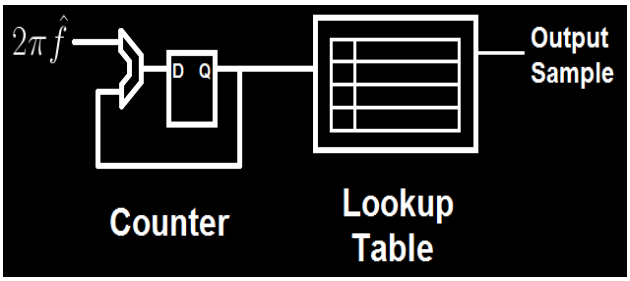

Fig. 7 Basic Numerical Control Oscillator

A basic NCO consists of a lookup table containing waveform data (usually a sinusoid) for exactly one period and a counter for indexing into the table. The rate of change of the counter determines the frequency of the output wave, in normalized units, because the output wave still exists in the discrete time domain. The counter is referred to as a 'phase accumulator,' or simply an accumulator, because it stores the current value of the sine's phase. In this, one of the simplest explanations of how an NCO operates is that it tracks the argument to sin(2πfn) in a counter and uses a look up table to calculate the value of sin(2πfn).

D. Radix-4 Butterfly Calculation

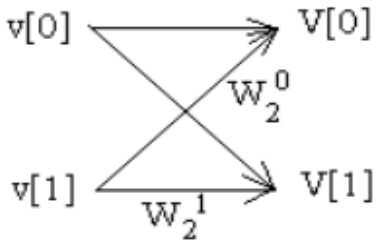

Fig. 8. Butterfly Calculation

To save computations the radix-4 decimation-in-frequency FFT arranges every fourth output sample into shorter-length DFTs. The radix-4 FFTs require only 75% as many complex multiplies as the radix-2 FFTs. The DFT sums over all groups of the every fourth discrete-time-index n=[0,4,8,…,N−4],n=[1,5,9,…,N−3],n=[2,6,10,…,N−2].

Mathematical manipulation shows that the length-N DFT can be computed as the sum of the outputs of four length-N4 DFTs, of the even-indexed and odd-indexed discrete-time samples, respectively, The figure shows that 16-point FFT needs two operations and a full sequence operation in all, while 256- points FFT needs four operations and a full sequence operation in all.

E. Optimization of Complex Multiple

From Fig.2 it is observed that complex multiplier plays an important role in butterfly calculation unit. Traditional complex multiplier has been obtained as follows.

$$(a+jb)(c+ jd)=ac-bd+j(ad+bc) \quad (11)$$

Where a, b, c and d are four independent real numbers, four multipliers and two adders are used. After doing some optimization deformation, (11) can be moved to (12) as follows,

$$(a+jb)(c+jd) = ac-bd + j(ad+bc) = ac+bc - (bc+bd) + j(bc+ac-ac+ad) = c(a+b) - b(c+d) + j[(a+b)c-a(c-d)] \quad (12)$$

IV. IMPLEMENTATION AND RESULTS

The proposed design is compared with the previous work. The FFT's are synthesized in Xilinx 14.7 targeting for Spartan-6 XC6SLX4 device. The simulation is done using Modelsim software. Power is measured by Xilinx Xpower analyzer. For a 2048 the proposed algorithm could achieve lower power than existing system. The reduction in dynamic power consumption is due to the fact that the ROM banks and multipliers are enabled only when necessary. The SRFFT has the irregular signal flow graph. A software solution is given in [9] for the indexing problem. The proposed architecture provides more efficiency, less delay and burst mode data transfer.

SRFFT	FF	LUT	BRAM	POWER(mw)
Radi-2 1024 point	166	468	3	20.01
Radix-4 2048 point	3091	2322	3	17

V.CONCLUSION

In this topic, a shared-memory-based SRFFT processor is presented. The proposed methodology reduces the dynamic power consumption at the cost of more hardware resources. SRFFT has the least number of multiplications compared to other types. Compared to radix-2, radix-4 is more suitable to use.

REFERENCES

[1] P. Duhamel and H. Hollmann, "'Split radix' FFT algorithm," Electron. Lett., vol. 20, no. 1, pp. 14–16, Jan. 1984.

[2] M. A. Richards, "On hardware implementation of the split-radix FFT," IEEE Trans. Acoust., Speech Signal Process., vol. 36, no. 10, pp. 1575–1581, Oct. 1988

[3] J. Chen, J. Hu, S. Lee, and G. E. Sobelman, "Hardware efficient mixed radix-25/16/9 FFT for LTE systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 2, pp. 221–229, Feb. 2015.

[4] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 39, no. 5, pp. 312–316, May 1992.

[5] D. Cohen, "Simplified control of FFT hardware," IEEE Trans. Acoust., Speech, Signal Process., vol. 24, no. 6, pp. 577–579, Dec. 1976.

[6] N. Amarnath Reddy, D. Srinivasa Rao and J. Venkata Suman "Design and Simulation of FFT Processor Using Radix-4 Algorithm Using FPGA,"International Journal of Advanced Science and Technology, vol. 61, pp.53-62, 2013.

[7] Z. Qian, N. Nasiri, O. Segal, and M. Margala, "FPGA implementation of low-power split-radix FFT processors," in Proc. 24th Int. Conf. Field Program. Logic Appl., Munich, Germany, Sep. 2014, pp. 1–2.

[8] A. N. Skodras and A. G. Constantinides, "Efficient computation of the split-radix FFT," IEE Proc. F-Radar Signal Process., vol. 139, no. 1, pp. 56–60, Feb. 1992.
[9] H. V. Sorensen, M. T. Heideman, and C. S. Burrus, "On computing the split-radix FFT," IEEE Trans. Acoust., Speech Signal Process., vol. 34, no. 1, pp. 152–156, Feb. 1986.
[10] J. Kwong and M. Goel, "A high performance split-radix FFT with constant geometry architecture," in Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), Dresden, Germany, Mar. 2012, pp. 1537–1542.
[11] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," IEEE Trans. Signal Process., vol. 51, no. 3, pp. 864–874, Mar. 2003.