IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 3, Ver. I (May-Jun. 2014), PP 88-94
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design of Efficient Pipelined Radix-2^2Single Path Delay Feedback FFT

Nisha Laguri[#1], Deepshikha Bharti[*2], K. Anusudha[#3]
[#1] *M.Tech Student, Electronics, Department of Electronics Engineering, Pondicherry University*
[*2] *M.Tech Student, Electronics, Department of Electronics Engineering, Pondicherry University*
[#3]*Assistant Professor, Department of Electronics Engineering, Pondicherry UniversityPuducherry, India*

Abstract**:** *Digital Signal Processing (DSP) has become a very important and dynamic research area. Now-a-day's many integrated circuits are dedicated to DSP functions. Fourier Transform is widely used in industrial applications as well as in scientific research. The performance in terms of throughput of the processor is limited by the multiplication. Therefore the multiplier is optimized to make the input to output delay as short as possible. A new FFT architecture based on distributed arithmetic has been developed and is compared to a previous multiplier based FFT. An efficient implementation of a pipelined Radix-2^2 Single Path Delay Feedback FFT is proposed in this paper. In this, Conventional R-2^2 SDF FFT architecture, complex multiplier and multiplier-less architecture based on distributed arithmetic and parallel prefix adder technique are explained. The advantages, performance and timing of the communication modules after implementation of the proposed technique are then discussed at the end. The comparison clearly indicates that the R-2^2 SDF FFT with the proposed architecture is efficient than the other implementations of its kind.*

Keywords: *FFT, Twiddle Factor, Butterfly Unit, Complex Multiplier*

I. Introduction

Pipeline FFT processor is a specified class of processors for DFT computation utilizing fast algorithms. It is characterized with real-time, non-stop processing as the data sequence passes through the processor. It is an AT^2 non-optimal approach with $AT^2=O(N^3)$, since the area lower bound is O(N). The class of pipeline FFT processors has probably the smallest "constant factor" among the approaches that meet the time requirement, due to its least number O(log N), of Arithmetic Elements (AE). The difference comes from the fact that an AE, especially, the multiplier, takes the much larger area than a register in digital VLSI implementation.

For hardware implementation, various FFT processors have been proposed. These implementations can be mainly classified into memory-based and pipeline architecture based. Memory-based architecture is widely adapted to design FFT processor, also known as single processing element (PE) approach. This design style is usually composed of a main PE and several memory units, thus the hardware cost and power consumption both is lower than the other architecture style.

However, this kind of architecture style has long latency, low throughput and can not be parallelized. On the other hand, the pipeline architecture style can get rid of the disadvantages of the foregoing style, at the cost of an acceptable hardware overhead. Generally, the pipeline FFT processors have two popular design types. One uses single-path delay feedback (SDF) pipeline architecture and the other uses multiple-path delay commutator (MDC) pipeline architecture. Such implementations are advantageous to low-power design, especially for the applications in portable DSP devices. Based on these reasons, the SDF pipeline FFT is adopted in this. The proposed architecture includes a distributed arithmetic based complex multiplier instead of using multiplier to store twiddle factors.

In this paper, a more detailed and completed description of the entire work is provided. The rest of this paper is organized as follows. First, a brief review of the Radix-2^2 Fast Fourier transform is described in Section II. Section III presents the Radix-2^2 FFT architecture. Section IV discusses the existing architecture. The performance evaluation of distributed arithmetic based complex multiplier is then discussed in Section V. Section VI presents proposed architecture. Results are compared with other architectures in Section VII and lastly, section VIII provides the conclusion.

II. Radix-2^2 Decimation In Frequency FFT Algorithm

The Discrete Fourier Transforms (DFT) X_k of an N-point discrete-time signal x_N is defined by:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, 0 \le k < \mathrm{N} \qquad \ldots\ldots\ldots\ldots\ (1)$$

Where W_N denotes the primitive N^{th} root of unity, with its exponent modulo N, x(n) is the input sequence and X (k) is the DFT. Applying a 3-dimensional linear index map.

$$n=\left\langle \frac{N}{2}n_1+\frac{N}{4}n_2+n_3 \right\rangle_N$$
$$k=\left\langle k_1+2k_2+4k_3 \right\rangle_N \qquad \text{.............} (2)$$

And Common factor algorithm (CFA) to derive a set of 4 DFTs of length N/4 as,

$$X(k_1+2k_2+4k_3)=\sum_{n_3=0}^{\frac{N}{4}-1}\left[H(k_{1,}k_2,n_3)W_N^{n_3(k_1+2k_2)}\right]W_{\frac{N}{4}}^{n_3k_3}$$
$$\text{..........} (3)$$

where n_1,n_2,n_3 are the index terms of input sample 'n' and k_1, k_1, $k_{1,}k_{2,}k_3$ are the index terms of the output sample 'k' and where $H(k_{1,}k_{2,}k_3)$ is expressed in equation (4).

$$H(k_1,k_2,n_3)=\left[x(n_3)+(-1)^{k_1}x\left(n_3+\frac{N}{2}\right)\right]+$$
$$(-j)^{(k_1+2k_2)}\left[x\left(n_3+\frac{N}{4}\right)+(-1)^{k_1}x\left(n_3+\frac{3N}{4}\right)\right]$$
$$\text{..........} (4)$$

.

Above equation (4) represents the first two stages of butterflies with only trivial multiplications in SFG, as BFI and BFII. Full multipliers are required after the two butterflies in order to compute the product of the decomposed twiddle factor $W_N^{n_3(k_1+2k_2)}$ in equation (3).

Applying this CFA procedure recursively to the remaining DFTs of length N/4 in equation (3), the complete radix-2^2Decimation-in-frequency (DIF FFT) algorithm is obtained. The corresponding FFT signal flow graph for N=16 is shown in Figure1, where small diamonds represent trivial multiplication by-$W_N^{N/4}=-j$, it involves only real-imaginary swapping and sign inversion.

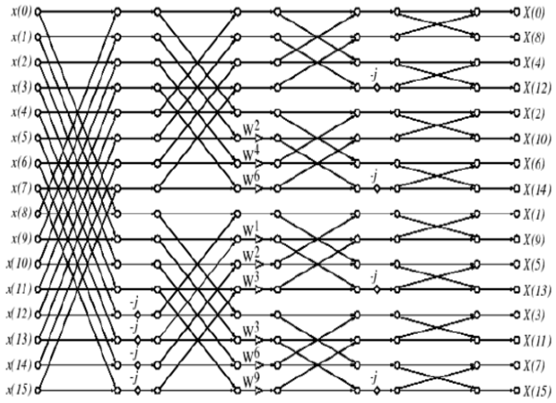

Figure 1: Radix-2^2 DIF FFT flow graph for N=16

III. Radix- 2^2 FFT Architecture

Mapping radix-2^2 DIF FFT algorithm derived to the radix-2 SDF architecture, a new architecture of R-2^2 SDF approach is obtained. Figure 1 outlines an implementation of the Radix-2^2 SDF signal flow graph for N=16, note the similarity of the data-path to R2SDF and the reduced number of multipliers. The implementation uses two kinds of butterflies; one identical to that in the R2SDF, the other contains also the logic to implement the trivial twiddle factor multiplication.

Due to the spatial regularity of Radix-2^2 algorithm, the processor's synchronization control is very simple. A (log2N)-bit binary counter serves two purposes: synchronization controller and address counter for twiddle factor reading in each stage. With the help of butterfly structures shown in Figure 3, the scheduled operation of the R-2^2 SDF processor in Figure 2 is as follows.

On first N/2 cycles, the 2-to-1 multiplexers in the first butterfly module switch to position "0", and the butterfly is idle. The input data from left is directed to the shift registers until they are filled. On next N/2 cycles, the multiplexers switch to position "1", the butterfly computes a 2-point DFT with incoming data and the data stored in the shift registers.

$$Z1(n) = x(n) + x\left(n + \frac{N}{2}\right)$$

$$Z1\left(n + \frac{N}{2}\right) = x(n) - x\left(n + \frac{N}{2}\right), 0 \le n < \frac{N}{2} \qquad \text{..... (6)}$$

The butterfly outputs Z1(n) and Z1(n + N/2) are computed according to the equation. Z1(n) is sent to apply the twiddle factors, and Z1(n + N/2) is sent back to the shift registers to be "multiplied" in still next N/2 cycles when the first half of the next frame of time sequence is loaded in. The operation of the second and third butterfly is similar to that of the first one, except the "distance" of butterfly input sequence are just N/4 and the trivial twiddle factor multiplication has been implemented by real-imaginary swapping with a commutator and controlled add/subtract operations, which needs two bit control signal from the synchronizing counter. Data then goes through a full complex multiplier. Further processing repeats this pattern with the distance of the input data decreases by half at each consecutive butterfly stages. After N-1 clock cycles, the result of complete DFT transform streams out to the right, in bi-transversed order. The next frame of transform can be computed without pausing due to the pipelined processing of each stage.

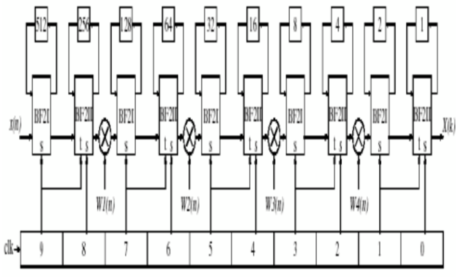

Figure 2: Radix-2^2 SDF pipeline FFT architecture

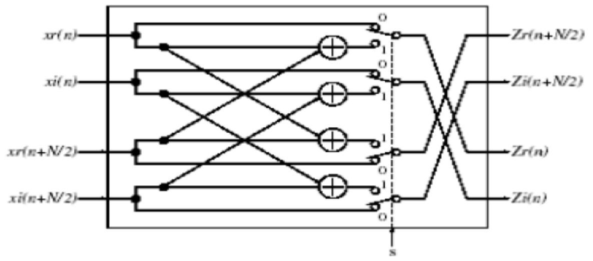

Figure 3: Butterfly Structure-I

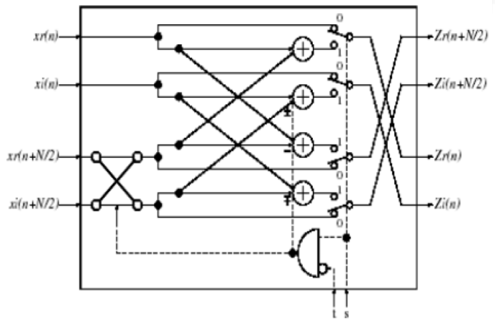

Figure 4: Butterfly Structure-II

IV. Existing Architecture

Traditional hardware implementation of FFT/IFFT processors usually employs a ROM to look up the wanted twiddle factors, and the word length of complex multipliers to perform FFT computation. However, this introduces more hardware cost, thus a bit-parallel complex multiplication scheme is used to improve the foregoing issue.

Since the twiddle factors have a symmetric property, the complex multiplications which are used in the FFT computation can be one of the following three operation types-

$$W_N^K.(a+jb) = W_N^{k-\left(\frac{N}{4}\right)}(b-ja), N/4 < k < N/2 \quad \text{...... (7)}$$

$$W_N^k.(a+jb) = -W_N^{k-(\frac{N}{2})}(b-ja), N/2 < k < 3N/4 \text{..... (8)}$$

$$W_N^k.(a+jb) = -W_N^{k-\left(\frac{N}{2}\right)}(b-ja), 3N/4 < k < N \text{..... (9)}$$

Given the above three equations, any twiddle factor can be obtained by combination of these twiddle-factor primary elements. In other words arbitrary twiddle factor used in FFT can utilize these operation types to derive the wanted value, thus can significantly reduce the size of ROM used to store the twiddle factors.

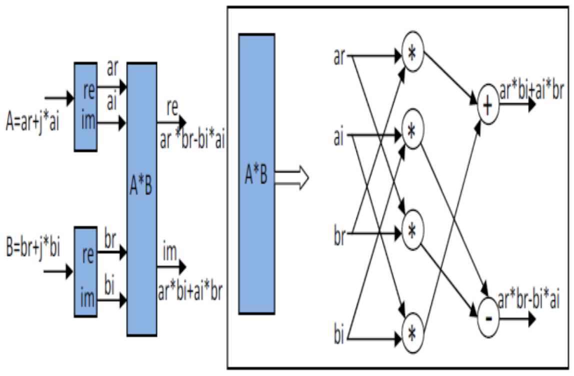

Figure 5: Conventional Complex Multiplier

Booth multiplication algorithm has been used for complex multipliers. Thus the overall latency of the real-time implementation varies as the processing word length changes.

V. Distributed Arithmetic Method for Complex Multiplication

In this section, complex multiplication operation using DA is explained. DA can be used to implement multiplication operation if either the multiplicand or the multiplier value is fixed. It stores the possible combinations of fixed operand in ROM and suitable combination is added and shifted with respect to bits of other operand. The method for DA base complex multiplication can be summarized as:

$$Z_R + jZ_I = (B_R + jB_I)*(T_R + jT_I) \quad \text{............ (10)}$$

Where $Z_R = B_R T_R - B_I T_I$

$$Z_I = B_R T_I + B_I T_R$$

It shows that 4 real multiplications and 2 real additions are required to compute $Z_R$and Z_I. But these equations can be considered as one ‘multiply and accumulate operation

$$y = \sum_{k=1}^{k} c_k x_k \quad \text{............ (11)}$$

Let, C_k are fixed coefficients and x_k are the input words. If x_k is M-bit fractional number in 2's complement form then it can be expressed in following form-

$$x_k = -b_{k0} + \sum_{m=1}^{M-1} b_{km} 2^{-m} \quad \ldots\ldots\ldots\ldots\ldots (12)$$

VI. Proposed Architecture

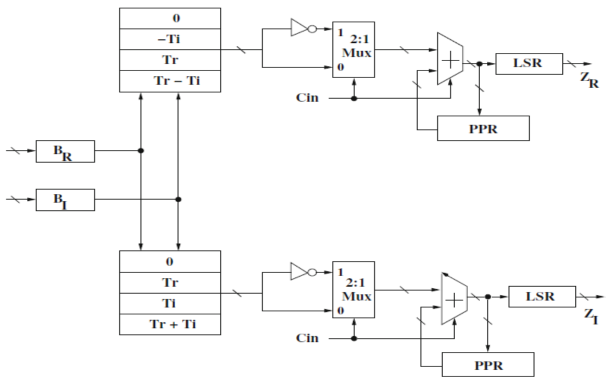

Figure 6: DA based complex multiplier

In proposed architecture a pipelined Radix-2^2SDF FFT unit is designed without using multipliers. All the complex multiplications required for this type of FFT are implemented using Distributed Arithmetic (DA) technique.

The detailed architecture for complex multiplier is shown in above Figure 6. The real and imaginary parts of incoming words B_R and B_I are stored in two 8 bits wide parallel in serial out register. Shifting is carried out starting from LSB to MSB.

Each output bit of these two registers is used as address lines of the ROMs. The ROM stores pre-calculated outcomes for both $Z_R$and Z_I. The size of each ROM is 4×8. One of the input to the 2:1 MUX is directly fed from the output of ROM and the other input to MUX is inverted. Input and output bit width for MUX is also 8 bits. The select line of MUX is 'cin' signal and it remains as '0' till the MSB arrives at output. If select line 'cin' of Mux is 1, it selects inverted output from ROM and it is added to the value stored in the partial product register (PPR). The PPR is 8 bit wide 'parallel in parallel out' register which also performs 1-bit right shift operation. Finally the output is taken from the left shift register.

VII. Parallel Prefix Adder

The Brent-Kung adder is a parallel prefix adder. Parallel prefix adders are special class of adders that are based on the use of generate and propagate signals. Simpler Brent-Kung adders was been proposed to solve the disadvantages of Kogge-Stone adders. The cost and wiring complexity is greatly reduced. It considered as one of the better tree adders for minimizing wiring tracks, fan out and gate count and used as a basis for many other networks. The block diagram of 8-bit Brent-Kung adder is shown in Fig.7.

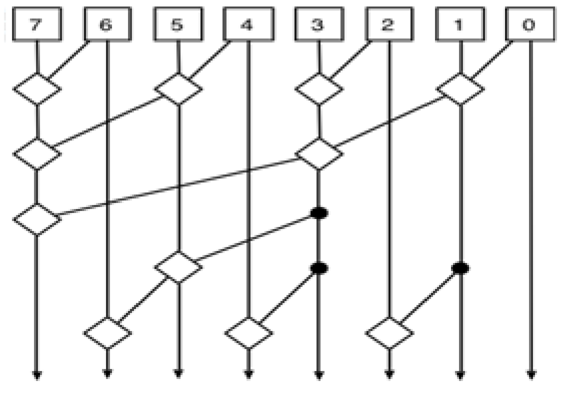

Figure 7: 8-bit Brent-Kung adder

VIII. Result And Discussion

The design of Single Path Delay Feedback FFT was modeled in VERILOG by making use of a module based structure approach. Both the SDF FFT were designed and synthesized by using Xilinx-13.2 version on Spartan-3 device. All the operation can be simulated at one time; the behavioural simulation is done by executing the test bench file.

Table I. Comparison between conventional and proposed complex multiplier

Parameters	Existing Complex Multiplier	Proposed Complex Multiplier
No. of slices	100	25
Delay (ns)	20.071	14.945

Table I. shows the comparison between existing and proposed complex multiplier. There is a huge difference between existing and proposed complex multiplier because in existing we have used multiplier blocks but in case of proposed, it is without the use of multiplier, it is basically made up of registers. And register takes less area when compared with the multiplier. Reduction in delay, because of the use of parallel prefix adder.

Table II. Comparison between existing and proposed butterfly structure

Type of model	Parameters	Butterfly-I	Butterfly-II
Existing Model	No. of slices	232	421
	Delay (ns)	18.004	25.034
Proposed Model	No. of slices	215	420
	Delay (ns)	14.495	17.945

Table III. Comparison between existing and proposed R-2^2 SDF FFT

Type of model	No. of slices	Delay (ns)	No. of inputs
Existing	2584	28.813	16X16
Proposed	2317	17.945	16X16

From above table it can be seen that, proposed FFT is better in terms of area and delay when compared with existing one.

IX. Conclusion

A multiplier less pipelined Radix-2^2 Single Path Delay Feedback FFT has been described in this paper. Mainly area and delay are considered for performance evaluation needed by the SDF FFT. Distributed Arithmetic based complex multiplier has been used as a proposed model.

Considering the symmetric property of twiddle factors in FFT, we have designed a distributed arithmetic based complex multiplier such that the size of twiddle factor is significantly shrunk. A new approach is proposed in this paper to reduce the area and delay of SDF FFT. The replacement of DA based complex multiplier with parallel prefix adder in place of conventional complex multiplier offers great advantage. This result shows that our design owns less area as well as less delay as compared to the existing one. Of course, our proposed architecture can also be adapted to high-point FFT applications, with lower size of twiddle-factor ROM's.

References

[1]. Xue LIU, Feng YU, Ze-ke WANG, "A pipelined architecture for normal I/O order FFT", Journal of Computers & Electronics, Springer online publication, Vol.12, No.1, pp. 76-82, May2011

[2]. K.Harikrishna, T.Rama Rao and Vladimir A. Labay, "A Radix-2^2 Pipeline FFT for Ultra Wide Band Technology", International Conference on computer & Network Technology (ICCNT), conference proceedings published by World Scientific Press, Singapore. Chennai, India, July 24-28, 2009

[3]. Sunil P. Joshi, Roy Paily, "Distributed Arithmetic Based Split- Radix FFT", Journal of Signal Processing System, Springer, Online Publication, 31 May 2013

[4]. Omri and Bouallegue, "New Transmission Scheme for MIMO-OFDM System", International Journal of Next-Generation

Networks, Vol.3, No.1,pp. 11-19, March 2011

[5]. Chang, Y.N., 2008, "An efficient VLSI architecture for normal I/O order pipeline FFT design" IEEE Trans. Circ. Syst. II: Exp. Briefs, volume 55, No. 12, pp. 1234-1238.

[6]. Omri and Bouallegue, "New Transmission Scheme for MIMO-OFDM System", International Journal of Next-Generation Networks, Vol.3, No.1,pp. 11-19, March 2011

[7]. Cheng, C., Parhi, K.K., 2007 "High-throughput VLSI architecture for FFT computation", IEEE Trans. Circ. Syst. II: Exp. Briefs, volume 54, No.10, pp. 863-867. [doi:10.1109/TCSII.2007. 901635]

[8]. Mian Sijjad Minallah, Gulistan Raja, "Real Time FFT Processor Implementation", IEEE—ICET 2006International Conference on Emerging Technologies.Peshawar, Pakistan 13-14, Pages: 192-195, November 2006

[9]. Nisha Laguri, K. Anusudha, " Design of Delay Efficient Distributed Arithmetic Based Split Radix FFT, International Journal of Engineering Trends and Technology, Volume 5, Number 7, November 2013