Energy Efficient Floating Point Fft/Ifft Processor For Mimo-Ofdm Applications

C. Padma[1], Dr. P. Jagadamba[2], Dr. P. Ramana Reddy[3]

[1]Research Scholar, Department of ECE, JNTUA, Ananthapuramu, Andhra Pradesh, India
[2]Assistant Professor (Sr), Department of ECE, SKIT, Srikalahasti, Andhra Pradesh, India
[3]Professor, Department of ECE, JNTUA, Ananthapuramu, Andhra Pradesh, India
padmasekhar85@gmail.com[1]

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: There are several methods to accomplish Fast Fourier Transform and Inverse Fast Fourier Transform processor for multiple inputs multiple output-orthogonal frequency division multiplexing applications. It requires high performance and low power implementation methodologies for reducing the hardware complexity and cost. In conventional fixed point arithmetic calculation is complex to utilize because the dynamic range of computations must be limited in order to overcome overflow and under flow problems. This paper presents floating point arithmetic optimization technique to implement radix-2 butterfly structures for the reduction of complex multipliers presented. Implementations of 32 bit floating point multiplier and floating point adder are presented by using single precision and compare the synthesis results with conventional system. In order to reduce the error we used floating point arithmetic for the butterfly structure. Energy efficient multiplier based on modified booth algorithm is used in radix-2 butterflies. By adopting this architecture the FFT/IFFT implementation using Xilinx FPGA Vertex-7 will improve the 25% logic utilization and the reduction in space utilization. Using arithmetic reduction the power delay product for radix 2 butterfly is reduced by 2.5% compared to normal implementation.

Keywords: Decimation In Frequency **(DIF),** Multiple Inputs Multiple Output-Orthogonal Frequency Division Multiplexing **(MIMO-OFDM),** Fast Fourier Transform **(FFT)**, Complex Multipliers

1. Introduction

Orthogonal frequency division multiplexing is a one of the leading modulation technique in wire line and wireless communication system. OFDM has integrated to numerous communication standards like WLAN, WPAN, Optical OFDM and UWB requires a high performance and low power FFT/IFFT to meet the demand of higher data rates.

Fast Fourier Transform is an algorithm that computes discrete Fourier transform of a sequence or its Inverse transform. Each frequency component represents its contributing amount to the original signal and the phase offset of the sinusoid available in the complex value of that frequency. Many researchers proposed optimized algorithms and multiple architectures to implement efficient FFT/IFFT computational units, that can be classified into sequential (memory-based) and pipelined. The major pipelined FFT architectures are Single-path delay feedback (SDF) and Multipath delay commutator (MDC), offers high-performance with increase in hardware resources.

Complex multipliers and adders are required to design butterfly structures in Decimation in frequency (DIF) as well as Decimation in Time (DIT). Implementation of complex multipliers is more critical than complex adders. Using optimization of complex multipliers will enrich the performance of FFT/IFFT computation in real time applications.

The different algorithms for FFT are Hexagonal [1], Cooley-Tukey [2], Prime-factor [3] and Bruun [4]. To reduce the amount of operations within the Fourier Transform calculation at each stage Cooley-Tukey is that the popular FFT algorithm, which recursively breaks a discrete time series into smaller. More processing paths are often implemented in parallel at the value of additional hardware and power to get higher throughput, In [5], for MIMO OFDM applications a new frequency scaling and dynamic voltage FFT processor presented to reduce hardware cost and power consumption. In [6] and [7] super-pipeline FFT cores were proposed to reduce power consumption.

Currently hardware design engineer is trying to implement efficient architectures to compute FFT algorithm in order to satisfy the real time application and high speed requirements. To meet out this requirement pipelined hardware architectures [8]- [12] are used widely to cater for high performance and with small latency for real time applications as well as low power and reasonably low silicon area. [13] A 1-dimensional (1D) FFT

architecture with 16-bit 64-point sequential algorithm implemented to achieve an area efficient, high-speed processor suitable for WLAN.

Canonical signed digit is a special technique to encode a value in a signed digit representation which itself is a unique representation and allows one number to be represented in many ways. Canonical signed digit technique of multiplication increases the speed of multiplication [14]. Further synthesis shows that due to reduced partial products, canonical signed digit multiplier has reduced dynamic power consumption and reduced area. In addition to fulfilling these three major aspects of floating point multiplier, CSD technique is proven to be useful in implementing multiplier with reduced complexity, because the cost of multiplication may be a direct function of the amount of non-zero bits within the multiplier [15]. The learning of the digit-slicing method has been explained in [16] for the digital filters. Digit-slicing FFT hardware design and implementation is discussed in [17]. In order to accomplish multiplier less FFT architecture Distributed arithmetic based method proposed [18]. "For Orthogonal Frequency Division Multiple – Access (OFDMA) [19] A mixed pipelined/cached 128 to 1024 point FFT was designed using power-aware twiddle factor multiplication". [20] A radix - $2^4/2^2/2^3$ based MDF FFT architecture is proposed for IEEE 802.16e applications in order to minimize the complexity in twiddle factor complex multiplication. For Wireless Personal Area Network (WPAN) applications [21] proposed memory based architecture in a 2.4 GS/s 8 data path-pipelined FFT processor provides low throughput for lesser radix. In [22, 23] by using Ripple Carry Adder (RCA) single precision floating point is discussed and it is distinguished with different floating-point multiplier. The design is developed based on the TSMC 180nm technology and modelled in verilog HDL. In [24] FFT implemented using modified booth multiplier and CLA is discussed and the author suggested that it requires less power and delay.

The paper is arranged as follows. Section II gives introduction to FFT algorithms. Different types of implementation are discussed in Section III. In section IV Modified implementation FFT using floating point multiplier-based radix-2 DIF butterfly is explained. Results and Discussions of the proposed designs are given in the Section V and finally Conclusion and Future Scope are discussed in Section VI.

2. II Fast Fourier Transform - Algorithms

In the radix-*r* algorithm radix-*r* butterflies are the basic building blocks to perform the basic computations. How the SFG structure is derived (for the radix-2 case) is shown in figure 1, only the proof of the first stage will be shown, the other proofs are analogous. The butterfly obtained is a radix-2 DIF (decimation-in-frequency)

$$x_1(k_0, n_{\propto-2}, \dots, n_0) = \sum_{n_{\propto-i}=0}^{1} x(n_{\propto-1}, n_{\propto-2} \dots n_0).W_N^{k_0.\sum_{\beta=0}^{\alpha-1} 2^\beta n_\beta} \quad (1)$$

$$= \sum_{n_{\propto-i}=0}^{1} x(n_{\propto-1}, n_{\propto-2} \dots n_0).W_N^{k_0.2^{\propto-1} n_{\propto-1}}.W_N^{k_0.\sum_{\beta=0}^{\alpha-2} 2^\beta n_\beta} \quad (2)$$

Cooley-Tukey algorithm is almost used to calculate the DFT and FFT. The amount of operations is diminish from $O(N^2)$ for the DFT to $O(N \log_2 N)$ for the FFT for this algorithm. The flow graph for sixteen-point "radix-2 FFT" constructed on Cooley-Tukey algorithm, deteriorate using decimation in frequency (DIF) is shown in below Figure 1. The FFT design consists of $n = \log_2 N$ stages.

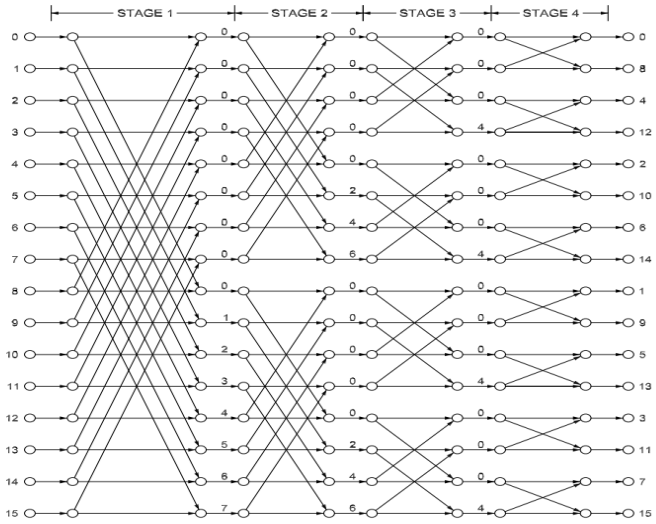


Figure 1 16-point radix-2 DIF FFT Flow graph

In general the complex multiplier can be comprehended by one adder, one subtractor and four real multipliers as shown in equation 3. In VLSI implementation this complex multiplier structure covered large chip area.

$(a_r+ja_i)(b_r+jb_i)= (a_rb_r-a_ib_i) +j\ (a_ib_r+a_rb_i)$ (3)

The basic radix-2 butterfly diagram for DIF is shown below figure 2. It can be shown that A and B designate the complex input to the present stage or complex output from previous stage, where C and D designate the complex output of this stage or complex input to the subsequent stage. The W_N is represented as a twiddle factor.

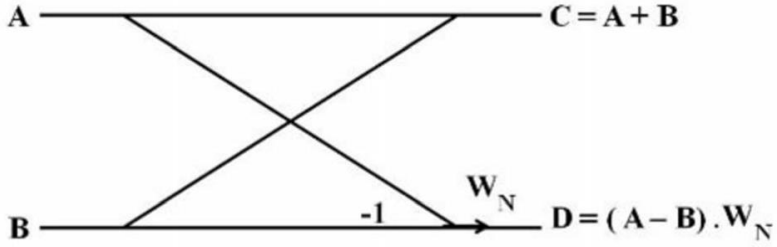


Figure 2 Radix-2 DIF butterfly

This proposed complex multiplier are often realized by five real adder/ subtractor and only three real multipliers shown on equation (4); this may save an enormous area in hardware implementation as shown in Figure 3.

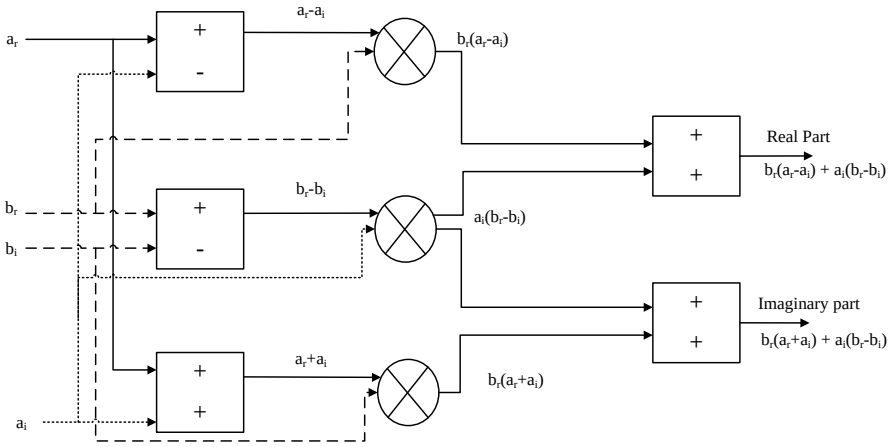


Figure 3 Hardware architecture for complex multiplier

$(a_r+ja_i)(b_r+jb_i)=\{(b_ra_r-b_ra_i)+(a_i\ b_r-\ a_i\ b_i)\}+j\{(b_i\ a_r+\ b_i\ a_i)+(a_i\ b_r-\ a_i\ b_i)\}$ (4)

3. Different types of implementation

The vital metrics to compute the performance of the FFT structure incorporate the arithmetic complexity, recursive structure of the FFT algorithm and therefore the overhead for memory access. Most of the radix-2 circuit's needs complex-multipliers and a ROM to store twiddle factor. The FFT circuit comprise of many subtractors, adders and multipliers for complex numbers, that's why a suitable representation should be select reasonably for, obtain good accuracy. All FFT architectures utilize"fixed-point arithmetic"; until latterly that "floating-point arithmetic" based FFTs is expand. As a portion of the implemented sketch, IEEE 754 single precision floating point characterization design is worn to constitute data. Floating-Point representations produce a good dynamic range and also ease the utilization of FFT processors as co-processors in association with "general purpose processors" (GPP). Although there has been substantial examination on the hardware implementation of the FFT algorithms, there are few inherent disadvantages of conventional works. They're create and optimized as FPGA, ASIC and DSP are fixed type platform, fixed type FFT algorithms and glued design parameters like input and output data, word length (L) and transform size (N).

There are different ways to implement the FFT (serial and parallel) algorithm. Previously designer used fixed point implementation and compared different architecture, different algorithms and word lengths. We will describe in this paper the implementation of FFT using single precision floating point.

4. Implementation of fft using floating point

The criteria that need to be considered when choosing between alternate FFT algorithms developed within the hardware are execution speed, system cost, and hardware design effort, programming effort, flexibility and precision. However, for concurrent signal processing system the primary criteria is latency. There are different FFT algorithms which have been implemented to diminish the number of computations. FFT could be implemented on hardware with efficient algorithm and the complexity of N- input FFT unit is divided into two sub (N/2) units continuing this decomposition until we get 2 inputs FFT is called butterfly unit. The proposed architecture which is shown in figure 2 is employed to implement the "two point FFT butterfly unit".

Table 1 Computational complexity for the direct DFT computation and Cooley-Tukey FFT algorithm

FFT Points (N)	Conventional DFT		Proposed FFT	
	No. of Complex Multipliers (N^2)	No. of Complex adders N(N-1)	No. of Complex Multipliers $(N/2)\log_2^N$	No. of Complex Adders($N\log_2^N$)
8	64	56	12	24
16	256	240	32	64
32	1024	992	80	160
64	4096	4032	192	384
128	16384	16256	448	896
256	65536	65280	1024	2048
512	262144	261632	2304	4608
1024	1048576	1047552	5120	10240

The proposed architecture in table 1 shows the number of floating point multiplication and floating point addition required to implement various input data.

Table 2 Comparison of Multipliers and Adders in both normal and proposed FFT

Size (N)	No. of stages ($\log_2^N$)	No. of BUs/stage (N/2)	Total No. of BUs	No. of complex multipliers		No. of complex adders	
				Normal FFT	Proposed FFT	Normal FFT	Proposed FFT
8	3	4	12	48	36	72	108
16	4	8	32	128	96	192	288
32	5	16	80	320	240	480	720
64	6	32	192	768	576	1152	1728
128	7	64	448	1792	1344	2688	4032
256	8	128	1024	4096	3072	6144	9216
512	9	256	2304	9216	6912	13824	20736
1024	10	512	5120	20480	15360	30720	46080

5. Result and discussion

The floating point adder having single precision (32 bit) is coded using HDL language. The simulation result of adder and synthesis report is shown in figure 3 and 5. The modified booth algorithm for the mantissa term and the exponent term is normal adder is used to implement 32 bit floating point multiplication. The result of fully hardware unit is done and the simulation result is shown in figure 4. The synthesis report for the floating point multiplicationis shown in figure 6.

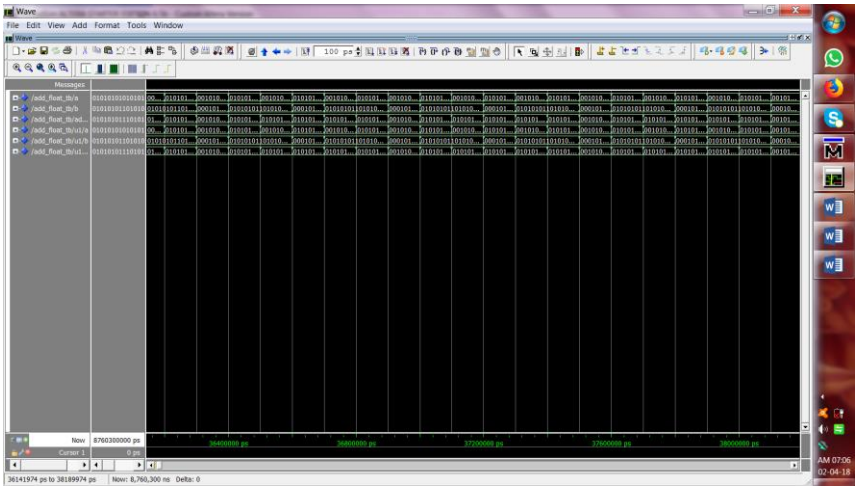


Figure 3 Simulation result of floating point adder

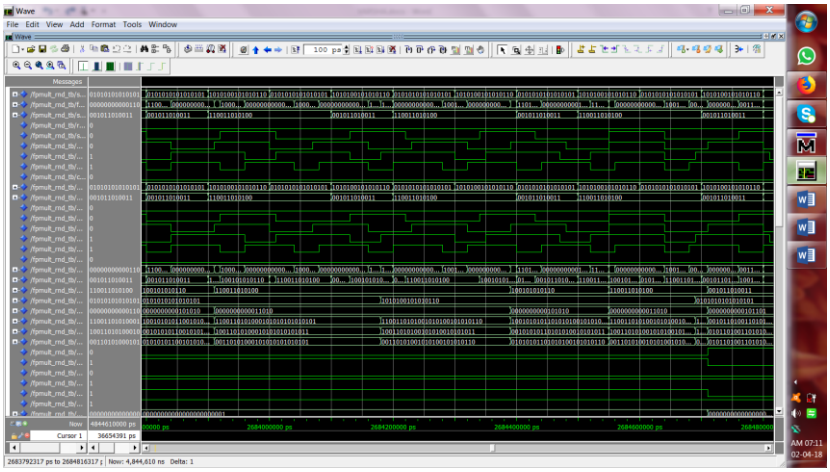


Figure 4 floating point multiplier simulation results

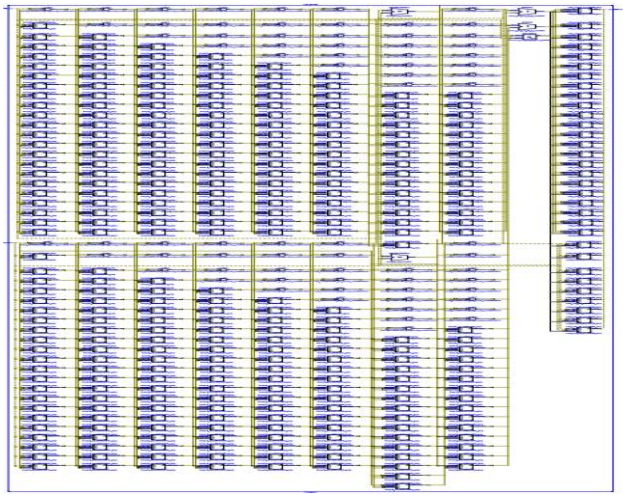


Figure 5 Floating point adder synthesis report

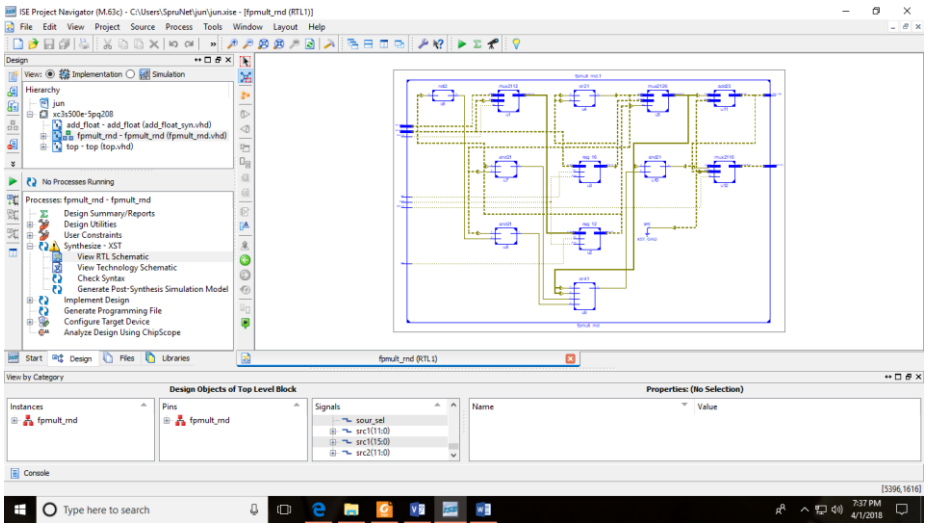


Figure 6 Synthesis report for proposed floating point multiplier

The following table 3 gives the utilization summary for the floating point adder. The device used is 3s500epq208-5

Description	No. of device used	Total obtainable devices	Utilization (%)
No. of Slices	1205	4656	25%
No. of 4 input LUTs	2143	9312	23%
Number of bonded IOBs	96	158	60%
Path delay :	63.247 ns		

The following table 4 compare the normal and proposed floating point multiplier implementation

Description	No of device used		Total number available	% utilization	
	Normal	Proposed		Normal	Proposed
Number of Slices	2280	1850	4656	49 %	40%
Number of 4 input LUTs	4570	3700	9312	49 %	40%
Number of bonded IOBs	96	50	158	60 %	31%
Path delay : Normal : 49.484ns Proposed: 38.275ns					

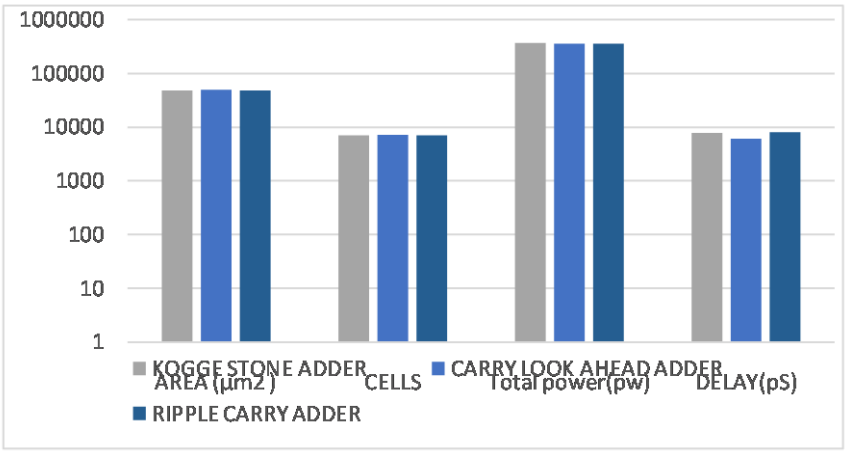
From the above table it is found that the delay is reduced from 49.48ns to 38.27ns. The performance improvement by 1.3 times with respect to normal floating point multiplier using kogge stone adder and modified booth multiplier for the mantissa multiplication.

The design is synthesized using Cadence back end tool. The comparison Table 5 shows the synthesized results for Kogge stone adder (KSA), ripple carry adder (RCA) and carry look ahead adder (CLA). From the table it can be found carry look ahead adder is better in terms of power and delay.

Table 5 Different implementation of floating point adders

Parameters	Kogge Stone Adder (KSA)	Carry Look Ahead Adder (CLA)	Ripple Carry Adder (RCA)
Area (μm^2)	48812	49290	48804
Cells	7121	7306	7031

Total Power (pw)	372164.619	365688.433	359375.906
Delay (ps)	7923	6149	8158



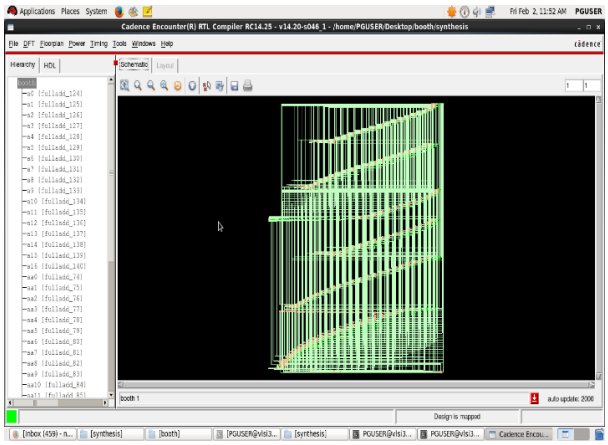


Figure 7 RTL view of Booth Recoded Multiplier using Cadence tool

Table 6 Comparison of different multipliers

Different Implementation	Delay (ns)	Area (μm^2)	P.D (mW)	PDP (pJ)
BFPM using RCA(180nm) [22]	8.0	71946	20.305	162.44
BFPM using RCA(180nm) [23]	9.2	65109	15.024	138.22
Modified Booth Mult and CLA [24]	8.481	-	119	1009.23
Proposed BFPM using KSA (90nm)	20.659	59995	5.183	107.07

The proposed multiplier using KSA and the floating point adder is used for the comparison of radix 2 butterfly architecture using 4 multiplications and using modified architecture using 3 multiplications

Table 7 Comparison of radix 2 butterfly

Different	Floating point adder	Floating point	No. of	No. of multiplier	Delay	Power	PDP

Imple-mentation			multiplication		adders required	required	(ns)	(mw)	(pj)
	Delay (ns)	Power (mw)	Delay (ns)	Power (mw)					
Existing with 4 mul	6.149	0.365	20.66	5.183	6	4	32.96	22.92	755.4
Using 3 mul					9	3	39.11	18.83	736.4

Table 8 Various point FFT Comparison

Different implementation	Total number of butterflies	Power delay product using 4 multiplier (nj)	Power delay product using 3 multiplier (nj)
8	12	9.0648	8.837
16	32	24.1728	23.565
32	80	60.432	58.912
64	192	145.036	141.389
128	448	338.419	329.907
256	1024	773.529	754.074
512	2304	1740.441	1696.666
1024	5120	3867.648	3770.368

Since radix 2 butterfly implementation using 3 multiplications require more delay due to one additional adder circuit before multiplication. The power required to implement this is less compared with 4 multiplication unit. The radix 2 butterfly power delay product of the proposed reduced by 2.5 % compared withthe conventional implementation.

6. Conclusion and future work

In this paper implementations of 32 bit floating point multiplier and floating point adder are presented. The proposed architecture for butterfly-2 structure is implementations with floating point arithmetic. The design is to optimize the complexity of implementing radix 2 butterfly structures. Here the number of multiplication is reduced and the power consumed by this arithmetic optimization. By adopting this architecture the FFT/IFFT implementation will improve the performance and the reduction in space utilization for realising the proposed structure. Here the energy utilized for the proposed work is about 2.5% reduction compared to normal implementation of radix 2 butterfly architecture. Since the work presents is single precision FP implementation double precision FP implementation will be the future work. The proposed design can also be expand to implement the design in the state of the technology to reduce delay, area and power.

References

1. M. Heideman, D. Johnson, and C. Burrus, “Gauss and the history of the fast Fourier transform,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 1, no. 4, pp. 14–21, 1984.
2. L. X. Jiang, C. Y. Liu, and P. Zhang, “A novel overall in-place in order prime factor FFT algorithm,” in *Proc. 5th Int. Congr. Image Signal Process. (CISP)*, Chongqing, China, 2012, pp. 1500–1503.
3. S. Mittal, Z. A. Khan, and M. B. Srinivas, “Area efficient high speed architecture of Bruun’s FFT for software defined radio,” in *Proc. IEEE Global Telecommun. Conf.*, Nov. 2007, pp. 3118–3122.
4. J. B. Birdsong and N. I. Rummelt, “The hexagonal fast fourier transform,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 1809–1812.
5. Y. W. Lin, Y. Chen, C. Y. Lee and Y. C. Tsao, “A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems”, *IEEE Journal Solid-State Circuits*, no.5, volume. 43, pp. 1260–1273, May 2008.
6. A. Chandrakasanand A. Wang, “A 180-mV subthreshold FFT processor using a minimum energy design methodology”,*IEEE Journal. Solid-State Circuits*,no.1, volume. 40, pp. 310–319, Jan. 2005.

7. D. Jeon, M. Seok, C. Chakrabarti, D. Blaauw, and D. Sylvester, "A super-pipelined energy efficient subthreshold 240 MS/s FFT core in 65 nm CMOS," *IEEE J. Solid-State Circuits,* vol. 47, no. 1, pp. 23–34, Jan. 2012.
8. L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An efficient locally pipelined FFT processor," *IEEE Trans. Circuits Syst. II, Exp. Briefs,* vol. 53, no. 7, pp. 585–589, Jul. 2006.
9. M. Garrido, R. Andersson, F. Qureshi, and O. Gustafsson, "Multiplierless unity-gain SDF FFTs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,* vol. 24, no. 9, pp. 3003–3007, Sep. 2016.
10. S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.,* May 1998, pp. 131–134.
11. M. Sänchez, M. Garrido, M. López, and J. Grajal, "Implementing FFT based digital
12. channelized receivers on FPGA platforms," *IEEE Trans. Aerosp. Electron. Syst.,* vol. 44, no. 4, pp. 1567–1585, Oct. 2008.
13. A. Cortés, I. Vélez, and J. F. Sevillano, "Radix r*k*FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.,* vol. 57, no. 7, pp. 2824– 2839, Jul. 2009.
14. Raja J, Mangaiyarkarasi P, Moorthi K (2015) Area efficient lowpower high performance cached FFT processor for MIMOOFDM application. Int J ApplEng Res 10:11853–11868
15. Vishwanath B.R.L &Theerthesha.T.S, "Multiplier using canonical signed digit code", International journal for research in applied science & engineering technology (IJRASET), (2015)
16. Rajdeepkaur and Tarandipsingh,"Design of 32-point mixed radix FFT processor using
17. CSD multiplier", fourth international conference on parallel distributed grid computing (2016)
18. Z. A. M. Sharrif, "Digit slicing architecture for real time digital filters." vol. Ph.D UK: Loughborough University, 1980.
19. Yazan Samir and T. Rozita, "The Effect Of The Digit Slicing Architecture On The FFT Butterfly," in 10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010) Kuala Lumpur, Malaysia, 2010, pp. 802-205.
20. Laguri N, Anusudha K (2014) VLSI implementation of efficient split radix FFT based on distributed arithmetic. In: IEEE Int. conference on Green Computing Communication and Electrical Engineering (ICGCCEE), pp 1–5
21. C. Chen, C. Hung and Y. Huang, An energy-efficient partial FFT processor for the OFDMA communication system in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 2, pp. 136–140, 2010
22. Patil MS, Chhatbar TD, Darji AD (2010) "An area efficient and low power implementation of 2048 point FFT/IFFT processor for mobile WiMAX". In: 2010 International Conference on Signal Processing and Communications (SPCOM), Bangalore, pp 1–4
23. S. Tang, J. Tsai and T. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 6, pp. 451– 455, 2010
24. Bhavesh Sharma, Ruchika Mishra et,al, (2015). Comparison Of Single Precision Floating Point Multiplier Using Different Multiplication Algorithm. International Journal of Electrical, Electronics and Data Communication 2320-2084. 3. 106-109.
25. Sharma, B., &Bakshi, A. "Design And Implementation Of An Efficient Single Precision Floating Multiplier Using Vedic Multiplication", International Journal of Scientific and Engineering Research, Vol.6, Issue 4, April-2015.
26. Senoj Joseph, I. Shyam, K. SalaiMathiazhagan, R. Vishnu, "FFT Implementation using Modified Booth Multiplier and CLA", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-3, February, 2020.