IMPLEMENTATION OF CONSTANT MULTIPLIER FFT

D.R. SANDEEP, Department of ECE, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India. Sandeepdr123@gmail.com
KUNCHE RAMA LAKSHMI, Department of ECE, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India. lakshmikunche6301@gmail.com
MAAKINEEDI SAI LAKSHMANA SURYA TEJ, Department of ECE, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India. Suryatejmsl@gmail.com
CH.V.P. PRIYA, Department of ECE, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India. Priyachodisetty123@gmail.com
SUNNAM PRAVEEN BABU, Department of ECE, Sri Vasavi Engineering College, Tadepalligudem, Andhra Pradesh, India. Sunnampraveen123@gmail.com

Abstract-The Constant Multiplier (CM) FFT hardware architecture presented in this work is a new Fast Fourier Transform (FFT) hardware design. CM-FFT uses a constant multiplier to calculate the rotation, unlike previous designs that required rotating the rotor between different angles. In this work, we examine 4-parallel and 8-parallel radix--2 CM FFTs. This performs the entire FFT calculation using a constant multiplier. Then radices 2k are provided, with radix24 and radix25 being the best options around. The proposed method achieves the highest clock frequency with 4-parallel FFT and at the same time, 1024-point-based experimental evidence shows that the number of random access blocks (BRAM) and digital signal processing (DSP) are reduced compared to previous methods. Reduce the amount -2 ^ 5 CM FFT a reported architecture on field-programmable gate arrays (FPGAs).

1.INTRODUCTION

One of the most important techniques in the field of digital signal processing is DFT. Due to the computational complexity of the Fast Fourier Transform (FFT), several FFT algorithms have been developed over the years. Modern digital communication systems such as digital video broadcasting (DVB) and orthogonal frequency division multiplexing (OFDM) systems rely heavily on FFTs. Algorithms such as radix-4,[1], split-radix[2], radix-22[3] are built using the underlying radix-2 FFT technique. The Radix-2 Multipath Delay Exchanger (R2MDC) is one of the oldest ways to implement the Radix-2 FFT in a pipeline [4]. The common use of memory buffers in R2MDC creates a less memory radix-2 single-pass delay feedback (R2SDF)[5] architecture.

Additionally, most of these hardware architectures are underutilized and require sophisticated hardware. High-throughput, low-power designs are important in the era of high-speed digital communications because they can meet speed and performance demands with minimal hardware overhead. In this work, we introduce a new method for designing architectures using FFT flow graphs. Some known FFT designs are derived using convolution transforms [6] and register minimization techniques [7][8].

2. FOLDING TRANSFORMATION

Several butterflies in the same column can be mapped to single butterfly unit using the folding transformation For an architecture with FFT size N, a convolution factor of N/2 results in a 2-parallel architecture, and a convolution factor of N/4 results in a design where four samples are processed in one clock cycle. A family of FFT designs is generated by various folding sets [9].

3. Design Methodologies for FFT Architectures

This section provides a general presentation of convolution transforms and register minimization approaches used in various well-known FFT structures. [9] An 8-point radix-2 DIF FFT is used as a diagram to illustrate the procedure. It can be used like any other radical. A flowchart for a radix-2 8-point DIF FFT is shown in Figure 1 [9]. This section provides a comprehensive presentation of convolution transforms and register reduction approaches used to construct many well-known FFT structures. [9] An 8-point radix-2 DIF FFT is used as a diagram to illustrate the procedure. It can be used like any other radical. A flowchart for a radix-2 8-point DIF FFT is shown in Figure 1 [9].

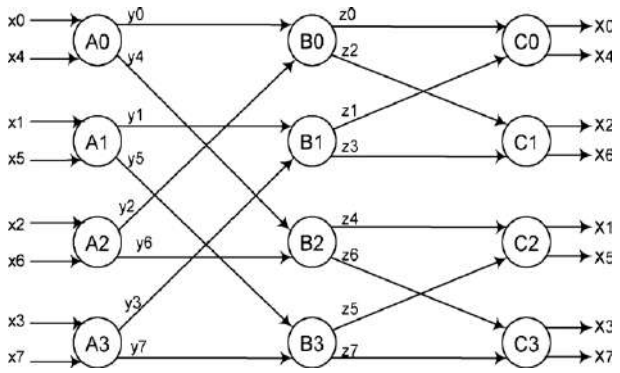


Fig 1:DataFlow graph (DFG) using 8-point radix-2 DIF FFT

As seen in fig. 1, this technique can be expressed as a data flow graph (DFG). The DFG's nodes stand in for tasks or calculations. All of the nodes here depict the radix-2 FFT [11] algorithm's butterfly calculations. A folding set, which is an organized set of procedures carried out by an identical functional unit, is necessary to change the DFG. The folding factor is the number K of entries in each folding set, some of which might be null operations.

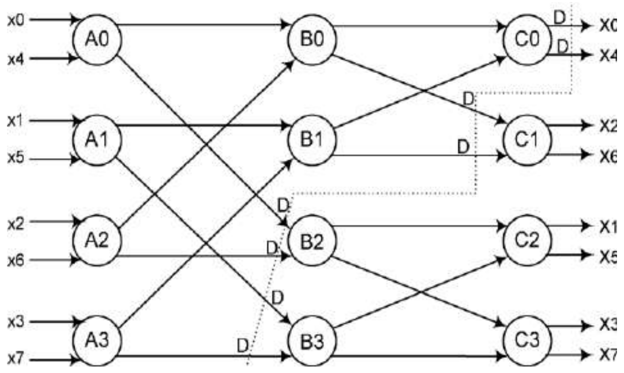


Figure 2 shows the pipelined DFG of the 8-point DIF-FFT before convolution. As an example, consider the convolution set A =,,,, A0, A1, A2, A3 with K=8. Convolution set A with convolution order 4 contains operation A0. Functional units are idle during null operations while performing operations A0, A1, A2, and A3 at appropriate times. The 8-point FFT architecture is derived using an orderly convolution technique. Consider an edge (e) with w(e) delay connecting nodes U and V.

The convolution equation (1) for edge e is

DF (U - V)=K w(e)-PU + v - u (1)

Assume that the number of pipeline stages of the hardware module processing node U is PU. Convolution equations with negative delays (non-pipelined) and non-negative delays are obtained using convolution sets (pipelined or retiming). Consider folding the DFG using the fold set in Figure 2.

A = {f, f, f, f, A0, A1, A2, A3}
B = {B2, B3, f, f, f, f, B0, B1}
C = {C1, C2, C3, f, f, f, f, C0}.

Suppose the butterfly operation has no pipeline stages (PA=0, PB=0, PC=0).
Retiming and/or pipelining [10] can be used to show that the convolution set is infeasible or satisfy DFU-V) 0.You may see negative delays on some edges. there is an equation

DF (A0 - B0) = 2   DF (B0 - C0) = 1

DF (A0 - B2) = -4 DF (B0 - C1) = -6
DF (A1 - B1) = 2 DF (B1 - C0) = 0
DF (A1 - B1) = -4 DF (B1 - C1) = -7
DF (A2 - B0) = 0 DF (B2 - C2) = 1
DF (A2 - B2) = - 6 DF (B2 - C3) = 2

DF (A3 - B1) = 0 DF (B3 - C2) = 0
DF (A3 - B3) = - 6 DF (B3 - C3) = 1…. (2)

It is shown that the DFG can be pipelined to ensure that the folded hardware delay numbers are non-negative. Here is his DFG delay for the folded pipeline:

DF (A0 - B0) = 2 DF (B0 - C0) = 1
DF (A0 - B2) = 4 DF (B0 - C1) = 2
DF (A1 - B1) = 2 DF (B1 - C0) = 0
DF(A1- B1) = 4 DF(B1- C1) = 1
DF(A2- B0) = 0 DF(B2- C2) = 1
DF(A2- B2) = 2 DF(B2- C3) = 2
DF(A3- B1) = 0 DF(B3- C2) = 0
DF(A3- B3) = 2 DF(B3- C3) = 1..(3)

4. DESIGN TECHNIQUES Of FFT

A block diagram of the folding structure used in the FFT method is depicted in Figure 3

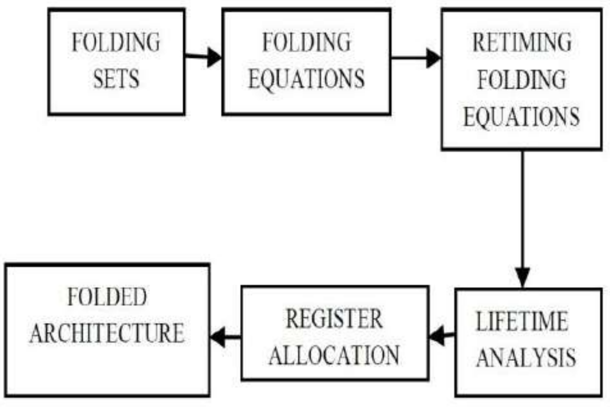


Fig. 3: Diagram of FFT design methods

Lifetime analysis, which evaluates the time between a data's production (Tinput) and eventual consumption, is an approach for reducing registers (Toutput).

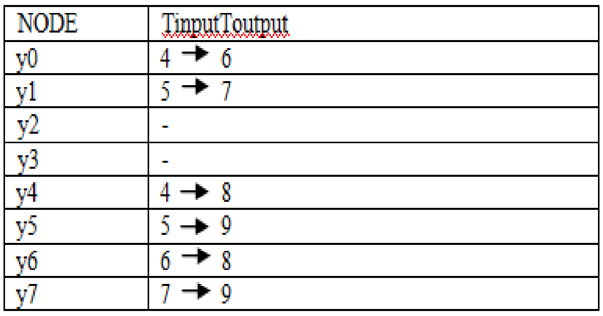
T input = u plus PU (4)
T output = u+PU+maxv DF (U,V)
T output = u+PU+maxv DF (U,V) (5) where PU is the number of pipelining stages in the functional unit that performs u and u is the folding order of U. The folded architecture may be implemented using the 24 registers (3).

4.REGISTER TECHNIQUES FOR MINIMIZATION

The number of alive variables at each time unit is determined in lifetime analysis, and the maximum number of alive variables at every time unit is.

specified. This is the absolute minimum number of registers need to run the DSP programme.

The folding architecture is designed with as few registers as possible using lifetime analysis techniques. Let variables y0, y1, ..., y7 correspond to the results at nodes A0, A1, A2, and A3 of the current 8-point FFT configuration, respectively. These edges are synthesized in a folded architecture using 16 registers. The maximum number of variables active in any unit of time is the minimum number of registers required to implement this DSP program.



NODE	TinputToutput
y0	$4 \rightarrow 6$
y1	$5 \rightarrow 7$
y2	-
y3	-
y4	$4 \rightarrow 8$
y5	$5 \rightarrow 9$
y6	$6 \rightarrow 8$
y7	$7 \rightarrow 9$

Fig.4 Table of linear lifetimes

The figures 4 and 5 depict the linear lifespan table and lifespan chart for these factors. As can be observed from the lifespan chart[4], the folded architecture only needs 4 registers as oppose to 16 registers in a simple implementation. Allocating registers in a forward-backward manner follows next.

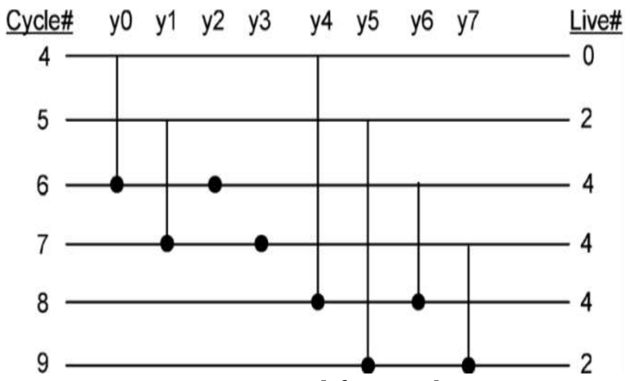
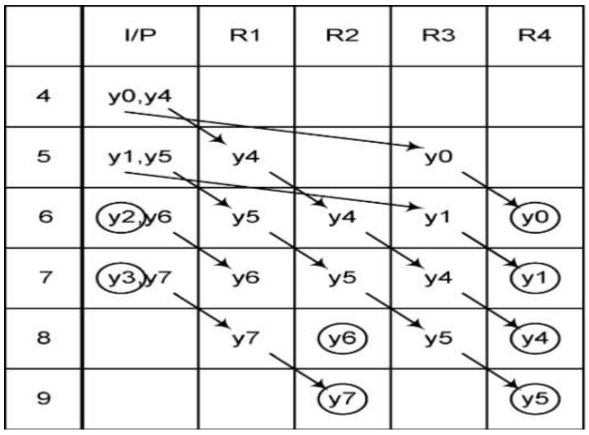


Fig. 5: Linear lifespan chart

An allocation table can be used to conduct register allocation. The variables are allocated to registers in the allocation table according to the allocation methodology. Figure 6 illustrates the allocation table that employs the forward-backward strategy to distribute the data for the 3 x 3 matrix transposer. [6][8].

1) Use lifespan analysis to determine the minimal number of registers.
2) Enter each variable in the time step that corresponds to the start of its lifespan.
3) Once a variable is dead or reaches the final register, it is allotted in a forward fashion.
4) The allocation of the present iteration continues in later iterations because this allocation is periodic.
5) The remaining life time is computed for variables that approach the last register but are not yet dead, and all these variables are assigned to a register in a reverse manner according to first come, first served.
6) Continue performing steps 4 and 5 till the allocation is finished.



	I/P	R1	R2	R3	R4
4	y0,y4				
5	y1,y5	y4		y0	
6	y2,y6	y5	y4	y1	y0
7	y3,y7	y6	y5	y4	y1
8		y7	y6	y5	y4
9			y7		y5

Fig. 6 Table of register allocation

The final structure in Fig. 7 may be synthesized from the allocations table in Fig. 6 and also the folding equations, and it can be obtained by simultaneously reducing the registers across all variables. Just 50% of the hardware is utilized in the resulting architecture. With this technology, pipelined parallel FFT designs are described.

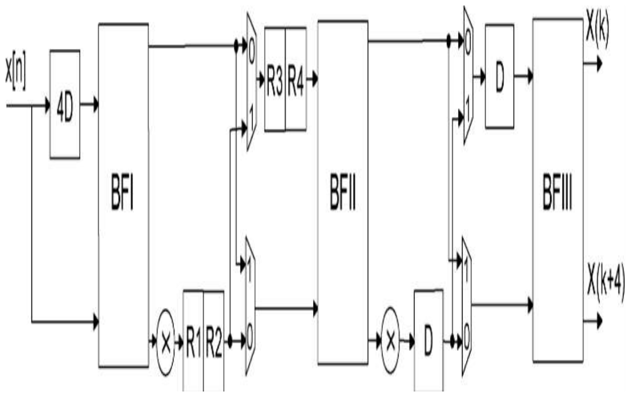
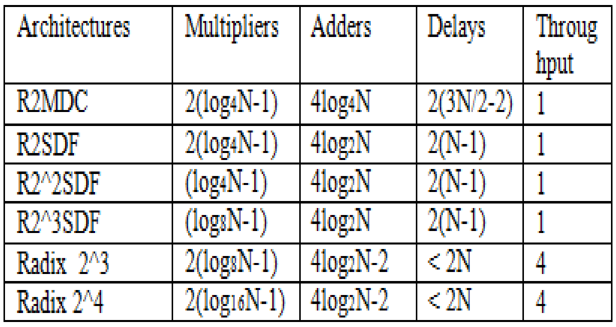


Figure 7. Folded Architecture

The required number of multipliers, adders, and delay components determines the hardware complexity of the design. Throughput indicates performance. The radix-24 architecture requires fewer multipliers than previous systems. The proposed FFT design reduces hardware complexity.

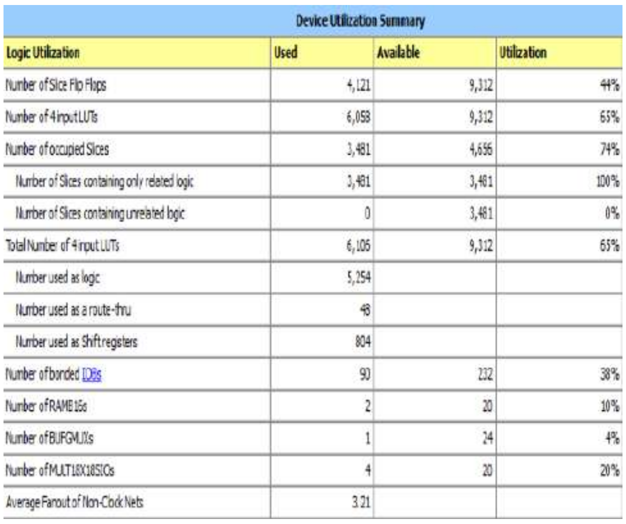


Architectures	Multipliers	Adders	Delays	Throughput
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	$2(3N/2-2)$	1
R2SDF	$2(\log_4 N-1)$	$4\log_2 N$	$2(N-1)$	1
R2^2SDF	$(\log_4 N-1)$	$4\log_2 N$	$2(N-1)$	1
R2^3SDF	$(\log_8 N-1)$	$4\log_2 N$	$2(N-1)$	1
Radix 2^3	$2(\log_8 N-1)$	$4\log_2 N-2$	$< 2N$	4
Radix 2^4	$2(\log_{16} N-1)$	$4\log_2 N-2$	$< 2N$	4

Figure 8. Comparison of N point FFT pipelined FFT architectures

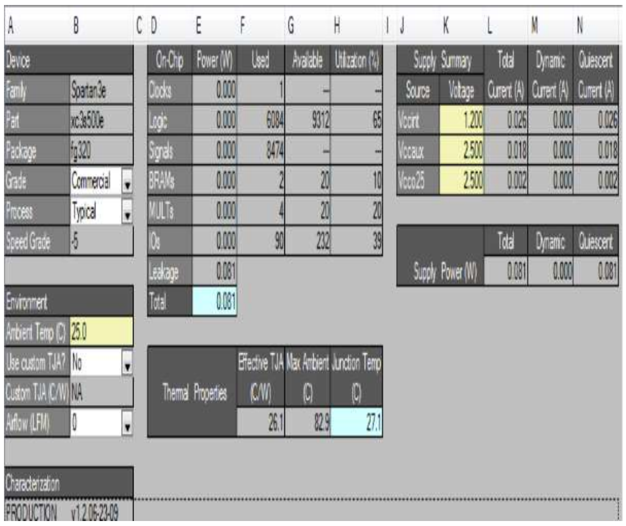
6.RESULT

6.1 Overview of Device Usage:



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	4,121	9,312	44%
Number of 4 input LUTs	6,058	9,312	65%
Number of occupied Slices	3,481	4,656	74%
Number of Slices containing only related logic	3,481	3,481	100%
Number of Slices containing unrelated logic	0	3,481	0%
Total Number of 4 input LUTs	6,106	9,312	65%
Number used as logic	5,254		
Number used as a route-thru	48		
Number used as Shift registers	804		
Number of bonded IOBs	90	232	38%
Number of RAMB16s	2	20	10%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	4	20	20%
Average Fanout of Non-Clock Nets	3.21		

In this case, a SPARTAN 3 device and XILINX 13.2 freeware were used to create and simulate foldable FFT and traditional FFT designs



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Grade	Commercial
Process	Typical
Speed Grade	-5

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	--	--
Logic	0.000	6084	9312	65
Signals	0.000	8474	--	--
BRAMs	0.000	2	20	10
MULTs	0.000	4	20	20
IOs	0.000	90	232	39
Leakage	0.081			
Total	0.081			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	82.9	27.1

Characterization	
PRODUCTION	v1.2,06-23-09

.

6.2. Time Overview:

Time restrictions inform the implementation tools of all design requirements. This indicates that the relevant restriction covers every route. This gives concepts that describe how to discover and restrict the most frequent timing pathways.

Minimum duration:10.666ns

(maximum frequency: 93.758MHz)

Minimum arrival time: 5.224ns

Maximum required end time after clock: 4.063ns

Maximum combined path delay: path not found

6.3.Report on Place & Route:

It describes where and how to place design in order to complete it on schedule.

Number of External IOBs 90

Number of IOBs with External Input 40

Number of IBUFs from External Input 40

Number of IOBs for External Output 50

Number of IOBs for External Output 50

6.4. Report on power generation:

The tool X Power Analyzer (XPA) is used to estimate post-implementation power. This tool gives you access to the implemented design database, which is the most precise since it enables you to pinpoint the precise logic and routing resources employed.

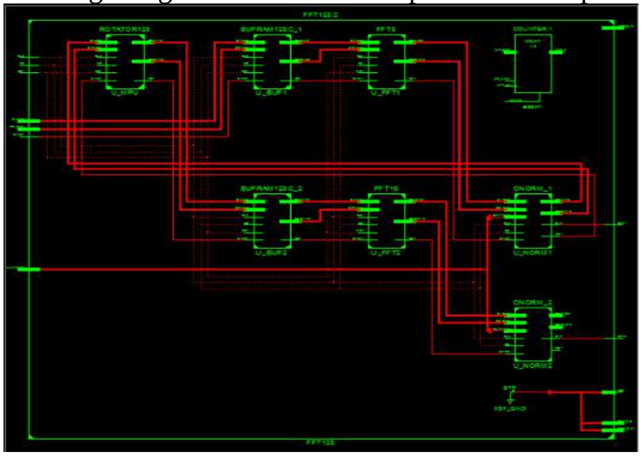
The several viewpoints, such as resource kinds, design hierarchy, clock domains, overall performance analysis, and more, let you browse your design.

Also, XPA allows you to adjust your environment settings and design activities so you can evaluate ways to reduce your design's supply and thermal power consumption.

This report shows less power required and also provides performance information.

6.5. Folded Transformation schematic:

This image of the folded transformation displays the 256-point FFT's folding architecture. Reduced multipliers and adders in folding designs result in reduced power consumption.



These designs have been designed to work best when the input values for RFFT are actual. Two alternate scheduling strategies have been suggested, one of which has less sophisticated control logic and the other of which has less delay components.

Since two input samples can be processed simultaneously, the operating frequency can be reduced by two, reducing power consumption by 50%. They are excellent candidates for use in portable applications. The actual FFT structure is underutilized. Future work will include developing FFT structures for real-valued signals that take full advantage of this technique.

7.CONCLUSION:

In this work, we propose a new approach **to** FFT implementation **using a** minimized hardware architecture called **pipeline** architecture and **a** design technique called **convolutional transform**

technique. **For an** FFT size **of N, N/2 convolution factors lead to a 2-parallel architecture,** and in another design, **N/4 convolution factors lead to** a 4-parallel **architecture design, where 4** samples are **processed.** in the same clock cycle. **A set of different convolutions** lead to a family of FFT **architectures,** design techniques **reduce** buffer **registers,** and parallel execution in **a** single clock cycle reduces power consumption and increases **speed. This** is Because several butterflies within a same columns could be mapped to a single butterfly unit, this is the case

In the proposed method, hardware utilization is only 50% in the derived architecture the pipelined parallel FFT architectures are presented by using this methodology, which in turn reduces the power consumption up to 50%.

REFERENCES:

[1] Glittas, AX, Sellathurai, M and Lakshminarayanan, G 2016, “A Normal I/O Order Radix-2 FFT Architecture to Process Twin Data Streams for MIMO”, IEEE Transactions on Very Large Scale Integration(VLSI) Systems, vol. 24, no. 6, pp. 2402-2406, doi:10. 1109/TVLSI. 2015. 2504391

[2] Ansuman Diptisankar Das, Abhishek Mankar, N. Prasad, K. K. Mahapatra, Avas Kanta Swain, "Efficient VLSI Architectures of Split Radix FFT using New Distributed Arithmetic", International Journal of Soft Computing and Engineerimg (IJSCE) ISSN: 2231-2307, vol. 3, Issue-1, march 2013, **doi:** 10.1109/ICGCCEE.2014.6922237

[3] S. He and M. Torkelson, “A new approach to pipeline FFT processor”, in Proc. 10th Int. Parallel Process. Symp. , 1996, pp. 766-770.

[4] H. Liu and H. Lee, “A high performance four-parallel 128/64- point radix-24 FFT/IFFT processor for MIMO-OFDM systems,” in Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS), Nov. 2008, pp. 834–837

[5] Y. N. Chang, “An efficient VLSI architecture for normal I/O order pipeline FFT design”, IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 12, pp. 1234-1238, Dec. 2008, doi:10. 1109/TCSII. 2013. 2268411

[6] A. M. Despain, “Fourier transform computers using CORDIC iterations”, IEEE Trans. Comput. , vol. C-3, pp. 993-1001, doi:10. 1109/T-C. 1974. 223800

[7] L. Li and A. M. Wyrwicz, “Parallel 2D FFT implementation on FPGA suitable for real-time MR image processing,” Rev. Sci. Instrum., vol. 89, no. 9, pp. 1–9, Sep. 2018.

[8] H. L. Groginsky and G. A. Works, “A Pipeline Fast Fourier Transform”, IEEE Trans. Comput. , vol. C-19, no. 11, pp. 1015-1019, doi:10. 1109/T-C. 1970. 222826

[9] 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.

[10] L. Liu, J. Ren, X. Wang , and F. Ye, " Design of low-power, 1GS/s throughput FFT processor for MIMO-OFDM UWB communication system," in Proc.IEEE Int, Symp, Circuits Syst., May 2007, pp.2594-2597.

[11] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, “A 2.4-GS/s FFT processor for OFDM-based WPAN applications,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 6, pp. 451–455, Jun. 2010.