International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 11 78 - 83

Simulation of Parallel Pipeline Radix 2^2 Architecture

Ankita. S. Dubey
Student MTech Electronics Engineering(Communication)
Vidharbh Institute of Technology
Nagpur,india.

Prof. Nilesh P. Bodne
Assistant Professor
Vidharbh Institute of Technology
Nagpur,india.

Abstract- In popular orthogonal frequency division multiplexing (OFDM) communication system processing is one of the key procedures Fast Fourier transform (FFT) and inversely for that Fast Fourier Transform (IFFT) is one of them. In this VLSI implementation Structured pipeline architectures, low power consumption, high speed and reduced chip area are the important concerns. In this paper, presentation of the worthy implementation of FFT/IFFT processor for OFDM applications is described. We obtain the single-path delay feedback architecture, to get a ROM of smaller size and this proposed architecture applies a reconfigurable complex multiplier. To minimize the error of truncation we apply a fixed width modified booth multiplier. As a result, the proposed radix-2k feed forward architectures even offer an attractive solution for current applications, and also open up a new research line on feed forward structures.

*****

INTRODUCTION

In the literature there are numerous FFT/IFFT designs. Highly among them are devoted to efficient realization of core FFT architecture and butterfly design. The needed twiddle factors in FFT are mostly assumed stored in memories in higher advanced and retrieved for butterfly multiplication whenever needed. This basically ends up with a very large lookup table in comparison with the core FFT processing elements and main data memory, basically for large FFT lengths as 8192. Thus, a capable TF genera-tor with lesser area and high speed performance is in dispensable, assuming mainly for portable and more data rate design. In earlier times, TF generation techniques were not taken into consideration because of the fact that OFDM systems were not as pervasive as they are now. It was mainly applied to off-line, non-real-time applications. But, there are many popular generation techniques for trigonometric functions that can be applied to TF generation. For the designs of direct digital frequency synthesizer (DDFS) these computing techniques are mainly used.

Fast Fourier Transform (FFT):

In view of the importance of the DFT in various digital signal processing applications, such as linear filtering, correlation analysis, and spectrum analysis, its efficient computation is a topic that has received considerable attention by many mathematicians, engineers, and applied scientists. From this point, we change the notation that $X(k)$, instead of $y(k)$ in previous sections, represents the Fourier coefficients of $x(n)$. Basically, the computational problem for the DFT is to compute the sequence $\{X(k)\}$ of N complex-valued numbers given another sequence of data $\{x(n)\}$ of length N, according to the formula

$$X(k)=\sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \quad 0 \le K \le N-1 \text{----------1}$$

$$W_N = e^{-j2\pi/N} \text{-------2}$$

In general, the data sequence $x(n)$ is also assumed to be complex valued. Similarly, The IDFT becomes,

$$X(n)= 1/N \sum_{n=0}^{N-1} x(k)\, W_N^{-nK}, \quad 0 \le K \le N-1 \text{------3}$$

Since DFT and IDFT involve basically the same type of computations, our discussion of efficient computational algorithms for the DFT applies as well to the efficient computation of the IDFT. We observe that for each value of k, direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and N-1 complex additions ($4N$-2 real additions). Consequently, to compute all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions..

Radix-2 FFT Algorithms:

Assuming the computation of the $N = 2^v$ point DFT by the approach of divide-and conquer. We divide the N-point data sequence into two $N/2$-point data sequences $f_1(n)$ and $f_2(n)$, relating to the even-numbered and odd-numbered samples of $x(n)$, in sequence and that is,

$$f_1(n) = x(2n) \text{--------4}$$

$$f_2(n) = x(2n+1), \quad n = 0,1,\text{-----},(N/2)-1\text{---------5}$$

Thus f1(n) and f2(n) are obtained by decimating x(n) by a factor of

$$X(k)= \sum_{n=0}^{N-1} x(n)\, W_N^{nK}, \quad K = 0,1,---N-1 \text{ ------- } 6$$

$$=\sum_{n=even}^{N-1} x(n)\, W_N^{nK} + \sum_{n=odd}^{N-1} x(n)\, W_N^{nK} \text{-------7}$$

$$=\sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m)\, W_N^{mK} + \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m+1)\, W_N^{K(2m+1)} \text{----8}$$

But $W_N^2 = W_{N/2}$. With this substitution, the equation can be expressed

$$X(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} f1(m)\, W_{N/2}^{mK} + W_N^K \sum_{m=0}^{\left(\frac{N}{2}\right)-1} f2(m)\, W_{N/2}^{Km} \text{ --9}$$

$$= f_1(k) + W_N^{Km} F2(k), \quad k = 0,1,\text{-----},N-1 \text{ -----10}$$

Here $F_1(k)$ and $F_2(k)$ are the $N/2$ - point DFTs of the sequences $f_1(m)$ and $f_2(m)$, respectively . Since $F_1(k)$ and $F_2(k)$ are not constant, with period $N/2$, we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. even, the factor $W_N^{k+N/2} = -W_N^k$. thus the equation can be expressed as,

$$X(k) = F_1(k) + W_{\frac{N}{2}}^{mK} F2(k), k=0,1,\text{-----},(N/2)-1 \text{ ---.11}$$

$$X(k+(N/2)) = F_1(k) - W_N^K F2(k), \quad k = 0,1,\text{-----},(N/2)-1 \text{ --12}$$

We can see that the direct computation of $F_1(k)$ need $(N/2)^2$ complex multiplications. The same is needed by the computation of $F_2(k)$. Even further, there are $N/2$ even more complex multiplications needed to calculate $W_N^k F_2(k)$. Thus the computation of $X(k)$ requires $2(N/2)^2 + N/2 = N^2/2 + N/2$ complex multiplications. With this first step there is a reduction of the number of multiplications from N^2 to $N^2/2 + N/2$, which is about a factor of 2 for N large.

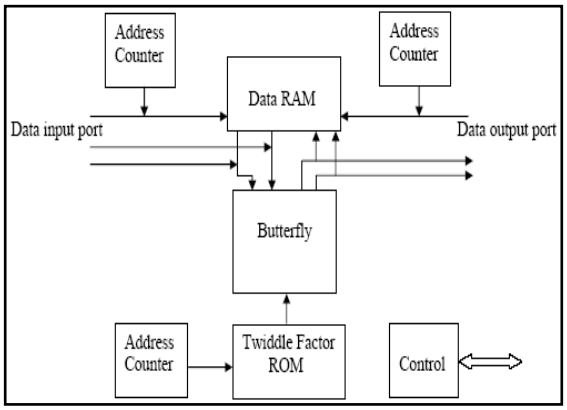


fig a.Butterfly parallel pipelined architecture

By computing $N/4$-point DFTs, we would obtain the $N/2$-point DFTs $F_1(k)$ and $F_2(k)$ from the relations.

$$F_1(k) = F_1\{f_1(k)\} + W_{\frac{N}{2}}^K F\{f1(2n+1)\}, \quad k = 0,1,\text{-----},(N/2)-1; \quad n = 0,1,\text{-----},(N/4)-1 \text{ --------13}$$

$$F_1(k+(N/4)) = F_1\{f_1(2n)\} - W_{\frac{N}{2}}^K F\{f1(2n+1)\}, \quad n = 0,1,\text{-----},(N/4)-1; \quad n = 0,1,\text{-----}, (N/4)-1\text{------- } 14$$

$$F_2(k) = F\{f_2(2n)\} - W_{\frac{N}{2}}^K F\{f2(2n+1)\}, \quad k= 0,1,\text{-----},(N/4)-1; \quad n = 0,1,\text{-----}, (N/4)-1\text{--------- } 15$$

$$F_1(k+(N/4)) = F\{f_2(2n)\} - W_{\frac{N}{2}}^K F\{f2(2n+1)\}, \quad k = 0,1,\text{-----},(N/4)-1; \quad n = 0,1,\text{-----}, (N/4)-1\text{------- } 1.16$$

F(*) Represents Fourier Transform

The decimation of the data sequence can be repeated again and again until the resulting sequences are reduced to one-point sequences. For $N = 2^v$, this decimation can be performed $v = \log_2 N$ times.

Thus the total number of complex multiplications is reduced to $(N/2)\log_2 N$. The number of complex additions is $N\log_2 N$. For illustrative purposes, depicts the computation of $N = 8$ point DFT. We observe that the computation is performed in tree stages, beginning with the computations of four two-point DFTs, then two four-point DFTs, and finally, one eight-point DFT. The combination for the smaller DFTs to form the larger DFT is illustrated for $N = 8$.

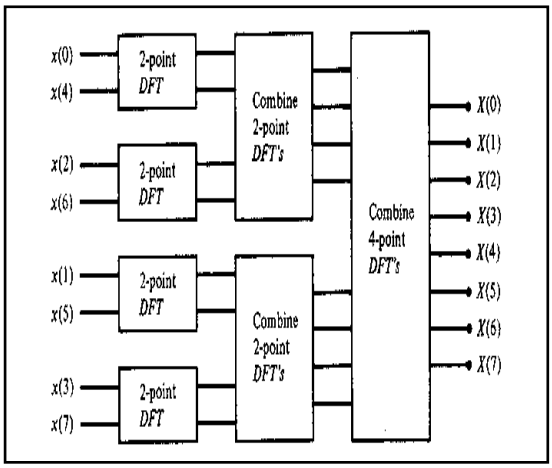


Fig b.Three stages in the computation of an $N = 8$-point DFT.

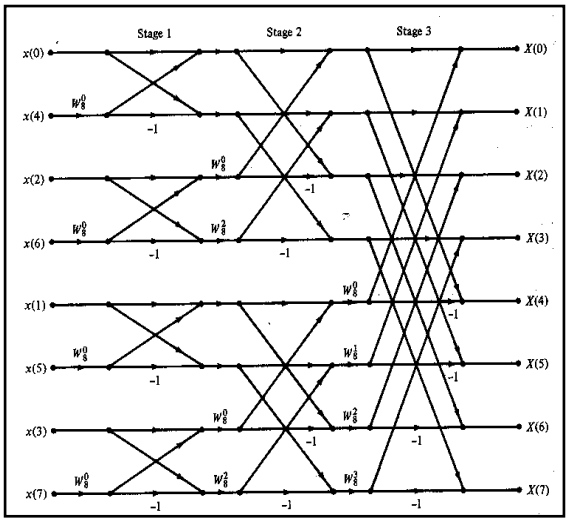


fig c. Eight-point decimation-in-time FFT Algorithm.

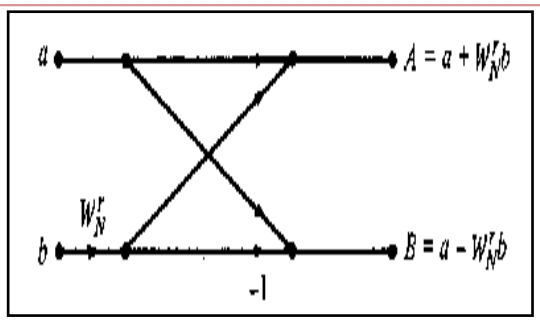


Fig d.Basic butterfly computation in the decimation-in-time FFT Algorithm.

An important observation is concerned with the order of the input data sequence after it is decimated (v-1) times. For example, if we consider the case where N = 8, we know that the first decimation yields the sequence x(0), x(2), x(4), x(6), x(1), x(3), x(5), x(7), and the second decimation results in the sequence x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7). This shuffling of the input data sequence has a well-defined order as can be ascertained from observing, which illustrates the decimation of the eight-point sequence.

PROPOSED METHODOLOGY

Pipelined FFT Hardware Architectures:

In the design of pipelined FFT hardware architectures the appearance of radix-2^2 was a milestone. Further, radix-2^2 was expanded to radix-2*k*. But, radix-2*k* was proposed only for single-path delay feedback (SDF) architectures, and it was not for feed forward ones, and also known as multi-path delay commutator (MDC). With the help of reference of few papers they used radix-2*k* feed forward (MDC) FFT architectures.

Here the use of low power techniques is employed so that power consumption is done and is reconfigurable complex multiplier. With the use of radix-4 algorithm single-path delay feedback pipelined architecture increase the computational speed, further decrease the chip area by three different processing elements (PE's) and these were proposed in this radix-4 64-point FFT/IFFT processor.

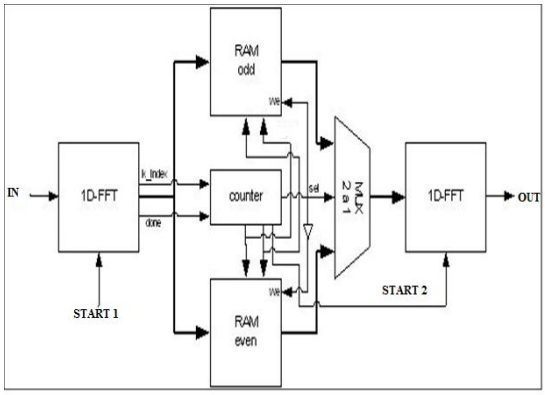


Fig.e Block diagram of FFT processor

Our proposed architecture uses a low complexity reconfigurable complex multiplier instead of ROM tables to generate twiddle factors and fixed width modified booth multiplier to reduce the truncation error. Figure Shows the Block diagram of FFT processor. We implement the processor in SDF architecture with radix-4 algorithm.

There are various algorithms to implement FFT, such as radix-2, radix-4 and split-radix with arbitrary sizes [1]. Radix-2 algorithm is the simplest one, but its calculation of addition and multiplication is more than radix-4's. Though being more efficient than radix-2, radix-4 only can process 4n-point FFT. The radix-4 FFT equation essentially combines two stages of a radix-2 FFT into one, so that half as many stages are required.

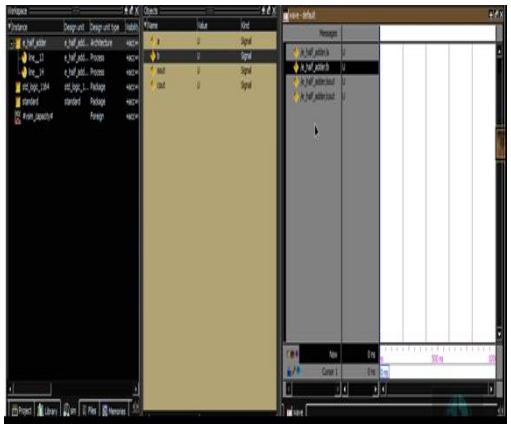
SIMULATION RESULT

Stage 1

1.Half Adder:

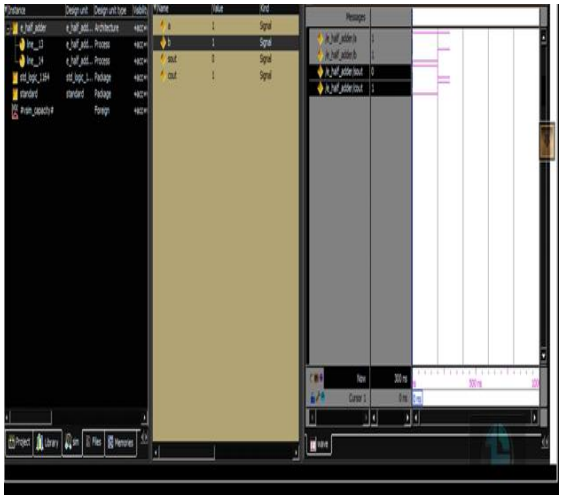
To combine two binary digits and produce a carry half adder is designed. Figure 5.1 depict two ways of constructing a half adder. To generate the carry an AND gate is added in parallel to the quarter adder. Output of the quarter adder is represented by the CARRY and the output of AND gate. As we come to know the output of the quarter adder is HIGH when either input, but not both, is HIGH. It is seen in the situation when only when both inputs are HIGH that the AND gate is activated and a carry is produced.

Input**:**



A library is one in which all new all designs are compiled. For starting a new simulation in MODELSIM is done by creating a working library called "work". The compiler uses 'work' as the default destination for compiled design units. Once the working library is created, our own design units can be compiled into it. Across all supported platforms The MODELSIM library format is compatible. Without having design recompiled we can simulate our design on any platform. Let us assume that the design loads successfully, zero is the time where stimulation is set up, and to begin simulation run command has been entered.

Output:



The proposed design has been simulated using MODELSIM , the output obtained after simulating is as shown in simulation is divided in to three parts first is simulation result of half adder ,in half adder there are two input a and b ,put a=1 and b=0, we get the result sout=1 and cout=0.If we put a=0 and b=0 ,we get the result sout=0 and cout=0.

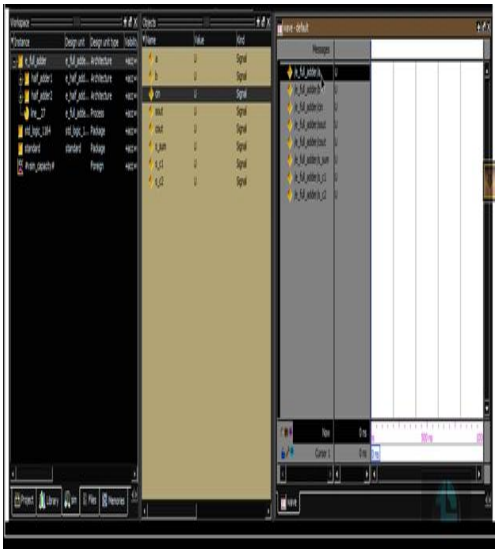
Stage 2

2.Full Adder:

To obtain the correct sum of two binary digits the full adder becomes compulsory when a carry input need to be added. From previous circuits a half adder has no input for carrying.

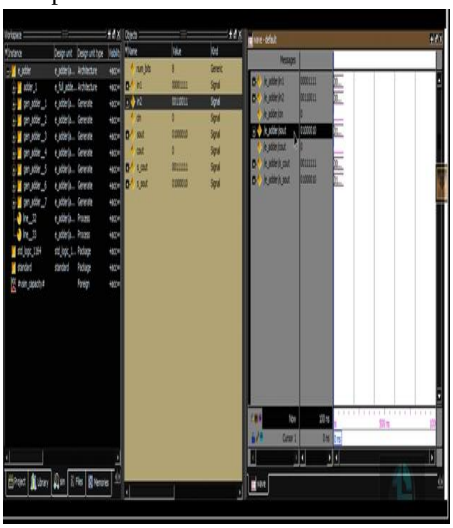
To use two half adders and an OR gate to get one full adder is one of the methods which is described in figure 3. The inputs A and B are applied to gates 1 and 2. This creates one half adder. As a proven input to the second this half adder and the carry-from a previous circuit tends to be one. To produce the carry-out for the circuit the carry from each half adder is applied to gate 5.

1.Input**:**



The library name used by the compiler as the default destination for compiled design units. After creating the working library, compile your design units into it. The MODELSIM library format is compatible across all supported platforms. The simulation is divided in to three parts second part is simulation result of full adder ,in full adder there are input a , b and c , put a=1, b=1,and c=1 can simulate your design on platform without having to recompile design. Assuming the design loads successfully, the simulation time is set to zero, and you enter a run command to begin simulation.

2. Output:

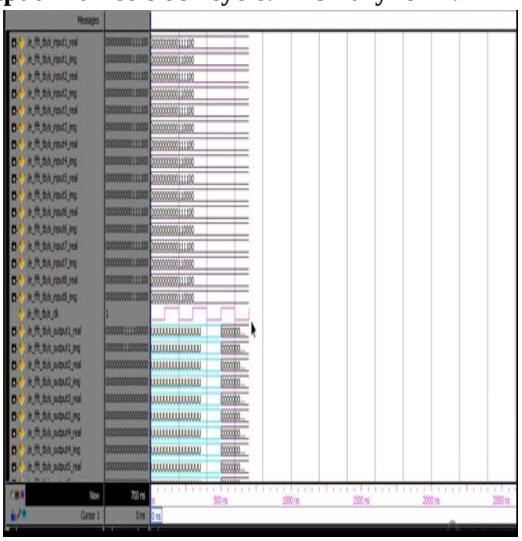


With the help of MODELSIM this design has been processed, in fig it is shown how the output is obtained after simulating. The simulation is separated in to three parts second part is simulation result of full adder, in full adder there are input a , b and c , put a=1, b=1,and c=1 we get the result sout=1 and cout=1.If we put a=0 ,b=1 and c=1 ,we get the result sout=0 and cout=1.

Stage 3

3.Full FFT:

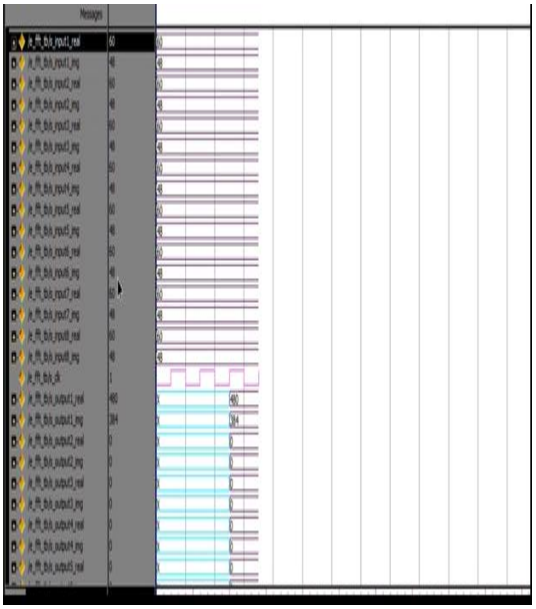
1. Output in three clock cycle: In binary form:



Here whole result of the full FFT converted in to the binary

form for futher calculations.

2. Output in three clock cycle: In decimal form:

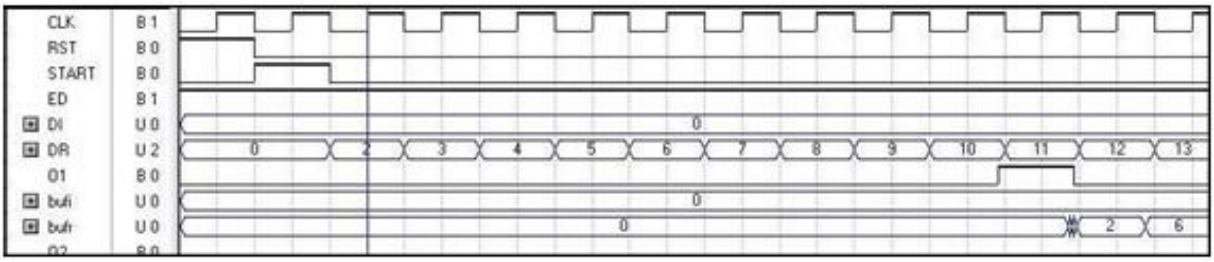


Parallel multiplier will give output in one clock cycle independent of the number of bits at the input in normal cases for n bit multiplier it requires n clock cycle which makes it slows so, for 16 bit clock cycle while in or case output will come in three clock cycle, here we reduce forty eight clockcycle in to only three clock cycle hence by using parallel pipelling technique we reduce the delay 93.75%.

CONCLUSION

For the FFT processor with low power consumption and efficient radix-2^2 parallel pipeline architecture has been proposed here. In this proposed article by us there is inclusion of architecture a reconfigurable complex constant multiplier and bit-parallel complex multipliers either using ROM's to store twiddle factors, which is suitable for the power-of-2^2 radix style having FFT processors.

The reason behind employing Low-power techniques is to minimize the power consumption in various FFT architectures. In the first technique there is use of a parallel-pipelined architecture at a lesser frequency to go till the assigned throughput. The complex multiplier is replaced with a minimum number of adders and shifters by using both two's complement in the second technique. In one clock cycle Parallel multiplier will provide the output which is not depended on the number of bits at the input in normal cases. For n bit multiplier it need n clock cycle that makes it slows so, for 16 bit clock cycle while in or case output will come in one clock cycle.



A memory based recursive FFT design has been proposed that has lesser gate counts, lower power consumption and more speed. The proposed architecture has three main advantages (1) fewer butterfly iteration to reduce power consumption, (2) pipeline of radix-2*2butterfly to speed up clock frequency, (3) even distribution of memory access to make utilization efficiency, in this paper, a number of high-performance FFT cores depended on combinations of hybrid low-power techniques were allowed. These low-power techniques are parallel-pipelined architectures, the multiplier less architecture, and the low-power butterfly architecture. Study had been carried out on the impact of parameterization on power/area. Which has main base as the combination of the proposed low-power techniques, up to 78% power saving is obtained.

REFERENCES

[1] Mario Galvez, J Grajal, M A. Sanchez and Oscar Gustafsson. *"**Pipelined Radix-2(k)** Feed Forward FFT Architectures"*2013, IEEE Transaction on Very Large Scale Integration (VLSI) Systems

[2] Manohar Ayinala, Keshab K. Parhi ***"Parallel-Pipelined Radix-2^2 FFT Architecture for Real Valued Signals"***2013 Department of Electrical and Computer Engineering ,University of Minnesota, Mimmeapolis, Mn, Usa.

[3] Preeti. G. Biradar, Uma reddy.N.V "***Implementation of Area Efficient OFDM Transceiver on FPGA"*** International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-3, July 2013.

[4] Anbarasan A. Shankar.K ***"Design and implementation of low power FFT/IFFT processor for wireless communication"*** Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012), pp. 152-155, March 2012.

[5] Chu yu Mao-Hsu Yen ***"A Low power 64-point FFT/IFFT Processor for OFDM Applications"*** IEEE

Transactions on Consumer Electronics, Vol. 57, Feb 2011.

[6] N. Kirubanandasarathy, Dr. K.Karthikeyan**," VLSI Design of Mixed Radix FFT Processor for MIMI-OFDM in Wireless Communication"**, 2011 IEEE Proceedings

[7] N.Kirubanandasarathy, Dr.K.Karthikeyan***," VLSI Design of Mixed Radix FFT Processor for MIMI-OFDM in Wireless Communication"***, 2011 IEEE Proceedings.

[8] Fahad Qureshi and Oscar Gustafson**, "Twiddle Factor Memory Switching Activity Analysis of Radix-2^2 and Equivalent FFT Algorithms"**, IEEE Proceedings, April 2010.

[9] Jianing Su, Zhenghao Lu, ***"Low cost VLSI design of a flexible FFT processor",*** IEEE Proceedings, April 2010.

[10] He Jing, Ma Lanjaun, Xu Xinyu, ***"A Configurable FFT Processor",*** IEEE proceeding 2010.

[11] M.Merlyn, ***"FPGA Implementation of FFT Processor with OFDM Transceiver",*** 2010 IEEE proceeding.

[12] Nuo Li and N.P.van der Meijs, ***"A Radix based Parallel pipelined FFT processor for MB- OFDM UWB system,"*** IEEE Proceedings, 2009.

[13] M. Garrido, K. K. Parhi, and J. Grajal, ***"A pipelined FFT architecture for real-valued signals,"*** *IEEE Trans. Circuits Syst. I*, vol. 56, no. 12,pp. 2634–2643, Dec. 2009.

[14] Minhyrok Shin and Hanho Lee, ***" A High-Speed Four Parallel Radix-2^4 FFT/IFFT Processor for UWB Applications,"*** in Proc. IEEE Int. Symp. Circuits and systems, 2008, pp. 960-963.

[15] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph**, "A radix 4 delay commutator for fast Fourier transform processor implementation,"** *IEEEJ. Solid-State Circuits*, vol. 19, no. 5, pp. 702–709, Oct. 1984.

[16] J. A. Johnston**, "Parallel pipeline fast Fourier transformer,"** in *IEE Proc.F Comm. Radar Signal Process.*, vol. 130, no. 6, Oct. 1983, pp. 564–572.

[17] P. A. Milder, F. Franchetti, J. C. Hoe, and M. P¨uschel, *"***Formal data path representation and manipulation for implementing DSP transforms,"** in *Proc. IEEE Design Automation Conf.*, Jul. 2008, pp. 385–390.

[18] Wei Han, Ahmet T. Erdogan, Tughrul Arslan, and Mohd. Hasan ***"High-Performance Low-Power FFT Cores"*** ETRI Journal, Volume 30, Number 3, June 2008

[19] Jen-Chuan Chi and Sau-Gee Chen ***"An Efficient Fft Twiddle Factor Generator"*** 2007 National Chiao Tung University Department of Electronics Engineering and Institute of Electronics 1001 Ta Hsueh Rd, Hsinchu, Taiwan, ROC.

[20] Yuan Chen, Yu-Wei Lin**, "A Slock scalin FFT/IFFT processor for WiMAX Applications"** IEEE proceedings 2006.

[21] J. Choi and V. Boriakoff, *"***A new linear systolic array for FFT computation**,*"* IEEE Trans. Circuits Syst. II, vol. 39, Apr. 1992, pp.236–239.

[22] L.-W Chang and M.-Y. Wu, ***"A new systolic array for discrete Fourier transform,"*** IEEE Trans. Acoust., Speech, Signal Processing, vol.36,Oct. 1988, pp.1165-1167

[23] V. Boriakoff, ***"FFT computation with systolic arrays, a new architecture***,*"* IEEE Trans. Circuits Syst. II, vol. 41, Apr. 1994, pp. 278–284.

[24] B. Gold and T. Bially, ***"Parallelism in fast Fourier transform hardware,"****IEEE Trans. Audio Electroacoust.*, vol. 21, no. 1, pp. 5–16, Feb. 1973.