J Inf Process Syst, Vol.15, No.3, pp.670~681, June 2019
https://doi.org/10.3745/JIPS.03.0119

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Implementation of Low Complexity FFT, ADC and DAC Blocks of an OFDM Transmitter Receiver Using Verilog

Alok Joshi*, Dewansh Aditya Gupta**, and Pravriti Jaipuriyar**

Abstract

Orthogonal frequency division multiplexing (OFDM) is a system which is used to encode data using multiple carriers instead of the traditional single carrier system. This method improves the spectral efficiency (optimum use of bandwidth). It also lessens the effect of fading and intersymbol interference (ISI). In 1995, digital audio broadcast (DAB) adopted OFDM as the first standard using OFDM. Later in 1997, it was adopted for digital video broadcast (DVB). Currently, it has been adopted for WiMAX and LTE standards. In this project, a Verilog design is employed to implement an OFDM transmitter (DAC block) and receiver (FFT and ADC block). Generally, OFDM uses FFT and IFFT for modulation and demodulation. In this paper, 16-point FFT decimation-in-frequency (DIF) with the radix-2 algorithm and direct summation method have been analyzed. ADC and DAC in OFDM are used for conversion of the signal from analog to digital or vice-versa has also been analyzed. All the designs are simulated using Verilog on ModelSim simulator. The result generated from the FFT block after Verilog simulation has also been verified with MATLAB.

Keywords

ADC, DAC, DFT, FFT, OFDM, Verilog

1. Introduction

1.1 Orthogonal Frequency Division Multiplexing

Orthogonal frequency division multiplexing (OFDM) is a highly-preferred system for the high data rate applications due to its improved immunity against channel impairments like intersymbol interference (ISI) and fading. OFDM is used as a downlink technique in 4th generation mobile system and many other broadband applications like HDSL, DVB, DAV, etc. With the increasing demand for high data rate, the concept of multiple-inputs multiple-outputs (MIMO) has also evolved. Many techniques have been proposed about this concept like vector permutation techniques and linear pre-coding techniques [1]. One of the important advantages of an OFDM transceiver system is its robustness against frequency selective fading. Rahaman et al. [2] wrote a paper which talked about using multiple antenna schemes over a fading channel. It also talked about OFDM being used in a relay and transmitting links to overcome/reduce these fading effects.

Manuscript received June 17, 2016; first revision November 29, 2016; accepted December 14, 2016.

Corresponding Author: Alok Joshi (20.alok@gmail.com)
* Dept. of Electronics and Communication Engineering, Jaypee Institute of Information Technology, Noida, India (20.alok@gmail.com)
** Software Engineer, Cadence Design System, India ({guptadewansh1, pravritijaipuriyar}@gmail.com)

1.2 Fast Fourier Transform

The fast Fourier transform (FFT) and inverse FFT (IFFT) blocks are used for modulation and demodulation in an OFDM system. In this paper, we have implemented the FFT block using Verilog, outputs of which have been compared to the same blocks designed using MATLAB to verify the designs and check the efficiency of the used algorithms.

Discrete Fourier transform (DFT) is a difficult process with a time complexity of an order of $O(n^2)$. To implement DFT on hardware is itself a very complex task. FFT has a large range of applications in the telecom domain. Thus, engineers have put in great effort in this field to achieve a real-time hardware implementation of FFT. Different algorithms have been developed in recent years to achieve an efficient implementation of FFT which include split radix algorithm, twisted radix algorithm, parallel FFT, mixed radix algorithm, and vector radix algorithm.

Cooley and Tukey proposed an FFT algorithm which is used to divide the complex DFT into the smaller DFTs with multiplication by twiddle factors. Dividing the DFT in smaller DFTs reduce the time complexity thus increasing efficiency. The general equation for the DFT is given by:

$$Z(k) = \sum_{m=0}^{M-1} z(m) e^{-j(\frac{2\pi}{N})mk} \tag{1}$$

The FFT problem gets decimated by two units in time in radix-2 implementation. Much has been proposed to implement FFT/IFFT efficiently. The main aim was to reduce time, calculations and increase accuracy. A paper published in 2015 [3] a pure design in VHDL was implemented, integrated with IP blocks using XILINX ISE 14.2. The modulator was implemented using 16-QAM in which different bit combinations were converted into corresponding constellation values. In demodulator, care was taken for the values like 2.999 to take it as 3 instead of 2. Fatima [4] designed FFT for 32-point using radix-2 using a smaller number of slices and computations.

In 2010, Merlyn [5] implemented a prototype FFT/IFFT processor which has a variable length to cover different specifications of OFDM applications. In this, they proposed a programmable MDC (mixed-radix FFT processor). An extension was made in the binary number while implementing through butterfly structure to avoid overflow danger.

In 2014, a paper published on design of a high-speed OFDM transmitter and receiver [6] analyzed 8-point IFFT/FFT with the radix-2 algorithm. These FFT and IFFT are pass modules, i.e., from Path0 to Path7. Complexity increases as the complex number come to these paths.

Mehta [7] had implemented 512-point IFFT and 512-point FFT blocks using radix-2. In this proposed work, the number of multipliers is reduced using an adder circuit and inter-carrier interface (ICI) self-cancellation techniques which can improve the performance parameters.

A paper issued in 2013, Palekar and Ingole [8] proposed to use efficient multiplication technique to reduce the partial product in FFT/IFFT which used to happen in conventional multiplication technique. A 32-point FFT and IFFT blocks for OFDM by using VEDIC multiplication was coded in VHDL and was synthesized using XILINX ISE software. The speed performance of this design easily satisfies most application requirements. In 2015, Sharma and Singh [9] carry out the same work efficiently using better resource utilization parameter.

To study the effect of various design parameters on the system performance OFDM system using MATLAB program was simulated [10]. It was observed that the optimum length of FFT/IFFT was 1024 points with the fixed signal-to-noise ratio (SNR) and the best SNR (fixed) was 60 dB and after this value, there was no effect on varying the SNR value.

A significant method to improve radix-2 and radix-4 algorithm was proposed in 2004. It was indexing the output samples again once the output comes from traditional algorithms which save effort while evaluating twiddle factor or accessing lookup table [11]. Later Rao et al. [12] used the same strategy efficiently while calculating 32-point DIT FFT. Mahajan and Chitode [13] implemented novel 16-point DIT FFT using radix-2 based on 8-point FFT. Ramachandran and Vanmathi [14] implemented the design of 32-bit radix-2 DIT in FPGA Spartan 3E using fewer computations as the output of shorter FFTs were used while computing the final output.

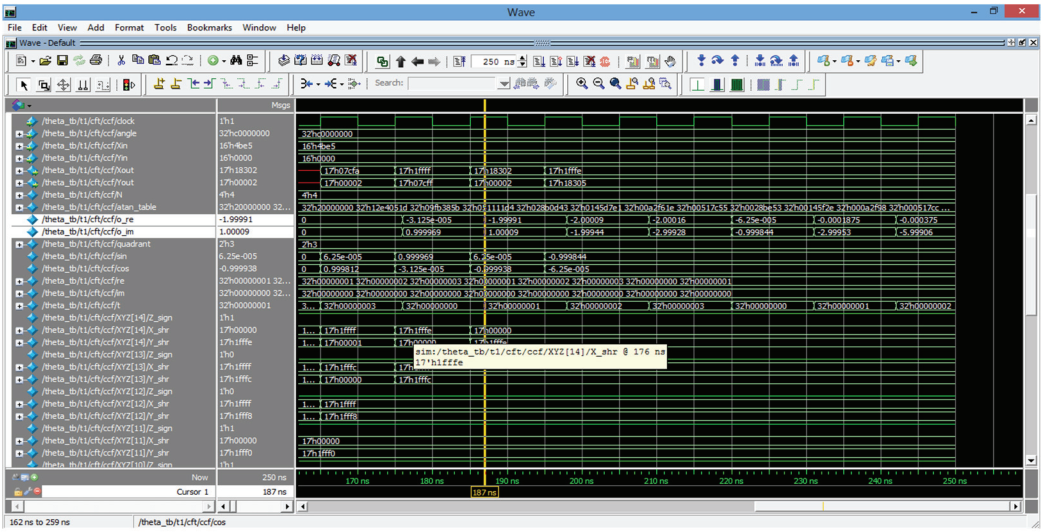
Direct computation requires almost 4N multiplications and 4N additions for every value of *k*. it is computed by implementing:

$$Z(k) = \sum_{m=0}^{M-1} z(m) W_M^{km} \qquad k = 0,1,2..M - 1 \tag{2}$$

where

$$W_M^m = e^{-j(\frac{2\pi}{M})m} \tag{3}$$

The same formula has been implemented which shows that this algorithm is slow and requires many computations. Its time complexity is $O(n^2)$. Results are shown for 4-point FFT in Fig. 1. In Fig. 1, ‘o_re’ and ‘o_im’ are the real and imaginary outputs of the FFT function, respectively.



```
/theta_tb/t1/cft/ccf/clock                 1'h1
/theta_tb/t1/cft/ccf/angle                 32'hc0000000
/theta_tb/t1/cft/ccf/Xin                   16'h4be5
/theta_tb/t1/cft/ccf/Yin                   16'h0000
/theta_tb/t1/cft/ccf/Xout                  17'h18302
/theta_tb/t1/cft/ccf/Yout                  17'h00002
/theta_tb/t1/cft/ccf/N                     4'h4
/theta_tb/t1/cft/ccf/atan_table            32'h20000000 32...
/theta_tb/t1/cft/ccf/o_re                  -1.99991
/theta_tb/t1/cft/ccf/o_im                  1.00009
/theta_tb/t1/cft/ccf/quadrant              2'h3
/theta_tb/t1/cft/ccf/sin                   6.25e-005
/theta_tb/t1/cft/ccf/cos                   -0.999938
/theta_tb/t1/cft/ccf/re                    32'h00000001 32...
/theta_tb/t1/cft/ccf/im                    32'h00000000 32...
/theta_tb/t1/cft/ccf/t                     32'h00000001
/theta_tb/t1/cft/ccf/XYZ[14]/Z_sign        1'h1
/theta_tb/t1/cft/ccf/XYZ[14]/X_shr         17'h00000
/theta_tb/t1/cft/ccf/XYZ[14]/Y_shr         17'h1fffe
/theta_tb/t1/cft/ccf/XYZ[13]/Z_sign        1'h0
/theta_tb/t1/cft/ccf/XYZ[13]/X_shr         17'h1ffff
/theta_tb/t1/cft/ccf/XYZ[13]/Y_shr         17'h1fffc
/theta_tb/t1/cft/ccf/XYZ[12]/Z_sign        1'h0
/theta_tb/t1/cft/ccf/XYZ[12]/X_shr         17'h1ffff
/theta_tb/t1/cft/ccf/XYZ[12]/Y_shr         17'h1fff8
/theta_tb/t1/cft/ccf/XYZ[11]/Z_sign        1'h1
/theta_tb/t1/cft/ccf/XYZ[11]/X_shr         17'h00000
/theta_tb/t1/cft/ccf/XYZ[11]/Y_shr         17'h1fff0
/theta_tb/t1/cft/ccf/XYZ[10]/Z_sign        1'h1
Now                                        250 ns
Cursor 1                                   187 ns
162 ns to 259 ns    /theta_tb/t1/cft/ccf/cos

sim:/theta_tb/t1/cft/ccf/XYZ[14]/X_shr @ 176 ns
17'h1fffe
```

Fig. 1. Simulation result of 4-point FFT on ModelSim (Wave Window).

Table 1. Comparison of calculations between direct summation method and radix-2 method for 16-point FFT

Calculations needed	Radix-2	Direct summation
Number of additions	32	256
Number of subtractions	32	-
Number of Multiplications	32	256
Total number of calculations	96	512

It is later implemented using radix-2 algorithms which have better efficiency and time complexity. The comparison has been shown in Table 1.

1.3 ADC/DAC

ADC and DAC is an essential block in OFDM used in receiver and transmitter respectively. DAC is used in OFDM as after the application of IFFT (can apply only on digital data) in transmitter block the signal is transmitted. And the transmission medium is air thus it is required to convert the digital data obtained from the previous block to analog signal. Similarly, the ADC block is used after the FFT block in the receiver.

Many designs were modeled and simulated like Generic ADC and DAC, flash ADC, SAR (successive approximation register ADC), pipelined ADC, etc. Suarez [15] in "Behavioral Modeling of Data Converters using Verilog" had stated that these ADCs are modeled using transistor level modeling which can be useful up to 16-bits but for higher bits the model becomes a lot complex making the design impractical. In generic ADC, there is a mismatch between ADC units which are later removed using dynamic element match (DEM).

Flash ADC is designed using high-speed comparator in a cascade fashion. It can be used up to 8-bit resolution because for higher resolution the design becomes complex and consumes more power, but these are ideal for devices requiring large bandwidth.

An ADC (N bits) requires N comparison periods as it starts the next conversion only after the previous one is complete. Pipelined ADC is most popular ADC architecture with the resolution of fewer bits at faster sample rates and of more bits at the lower rates. Ashraf [16] in 2016 designed an 8-bit SAR ADC with an input voltage of 1.2-V. The SAR control logic was designed asynchronously using Verilog coding. The design was implemented in Cadence Virtuoso using 1.8-nm technology. Designing of low power comparator was achieved which results in overall power consumption of 0.75 mW.

A 10-bit DAC with (6+4) bit segmentation was designed in 2006 by Murali Shanmugasundaram and Shanthi Pavan [17]. The simulations run on full transistor level schematic as well as on the macro design over a range of frequencies. This design uses current controlled current sources which is time-varying in nature and thus reduces the simulation time and increases simulation speed. It was implemented in a 0.35um CMOS process. In 2015, Santhanalakshmi and Yashoda [18] also designed a low power SAR ADC using Verilog-A and proved that bypass window technique is more efficient with a lesser number of transitions, i.e., 3 over 4.

Menzter and Wey [19] developed a mixed signal model of a 10-bit pipelined ADC using Verilog. The overall aim of this project was to identify design flaws based on input/output tests. Implementation of FIR filter was done using onboard ADC-DAC and FPGA.

To optimize design and computations it was also modeled using MATLAB and Simulink [20]. Although transistor level modeling is the most appropriate approach for mixed-signal circuits, but alternate modeling is required to avoid the complexity and long computation time.

The ADC results in a ratiometric value. Analog to digital conversions totally depend on the system voltage. Here the ADC assuming maximum voltage to be 10 V is converted to 1023 and any value less than 10 V will be a ratiometric value correspondingly after quantization.

2. Proposed Work

2.1 Fast Fourier Transform

Fig. 2 illustrates the butterfly structures of radix-2 for 16-point. The basic butterfly structure is implemented in the design of 16-point FFT.

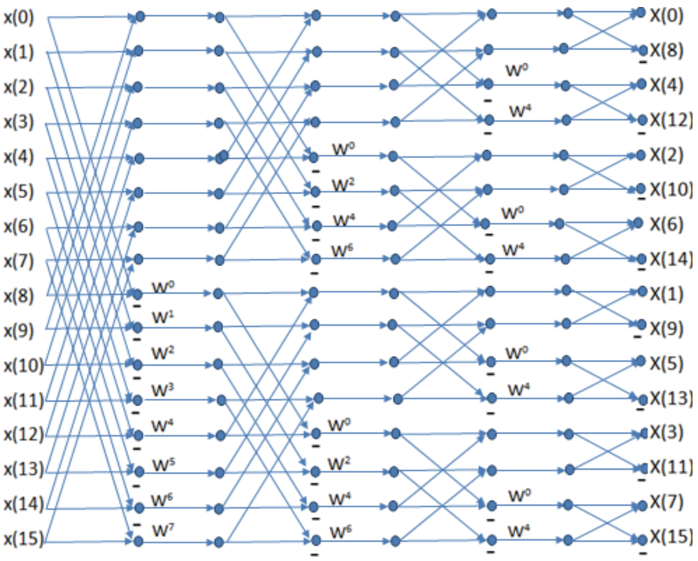


Fig. 2. 16-point FFT using radix-2.

The radix-2 is computed by dividing the Eq. (2) into odd and even terms as mentioned,

$$Z(k) = \sum_{m=even} z(m) W_M^{km} + \sum_{m=odd} z(m) W_M^{km} \tag{4}$$

$$= \sum_{m=0}^{M/2-1} z(2m) e^{-\frac{2\pi}{M}(2m)k} + \sum_{m=0}^{M/2-1} z(2m+1) e^{-\frac{2\pi}{M}(2m+1)k} \quad ; \quad (m = 0,1,2,..,\frac{M}{2}-1) \tag{5}$$

$$\text{Let} \quad \begin{matrix} f_1(m) = z(2m) \\ f_2(m) = z(2m+1) \end{matrix} \qquad (m = 0,1,2,..,\frac{M}{2}-1) \tag{6}$$

But $W_M^2 = W_{M/2}$. Using this substitution, the above equation becomes:

$$Z(k) = \sum_{m=0}^{M/2-1} f_1(m) W_{M/2}^{km} + W_M^k \sum_{m=0}^{M/2-1} f_2(m) W_{M/2}^{km} \tag{7}$$

$$Z(k) = F_1(k) + W_M^k F_2(k) \qquad k = 0,1,2..M-1 \tag{8}$$

Where

$$F_1(k) = DFT\{f_1(m)\} \tag{9}$$

and

$$F_1(k) = DFT\{f_1(m)\} \tag{10}$$

$$F_1(k+M/2)=F_1(k) \tag{11}$$

and

$$F_2(k+M/2)=F_2(k) \tag{12}$$

It implies $F_1(k)$ and $F_2(k)$ are periodic, with M/2 period.

$$W_M^{k+M/2}=-W_M^k\ . \tag{13}$$

Hence,

$$Z(k)=F_1(k)+W_M^kF_2(k) \qquad k=0,1,2..M-1 \tag{14}$$

$$Z(k+\frac{M}{2})=F_1(k)-W_M^kF_2(k) \qquad k=0,1,2..M-1 \tag{15}$$

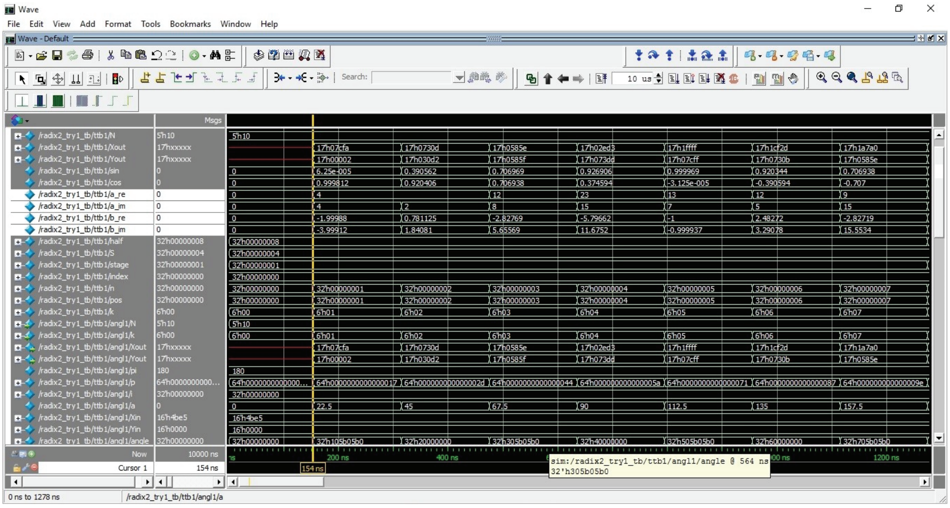
By using Eqs. (14) and (15) we compute radix-2 in our program.

Since the input values are complex therefore two different arrays one for real and the other for the imaginary part have been used.

The twiddle factors which are used in FFT/IFFT blocks are generated using the CORDIC algorithm. Each time while performing the calculation using butterfly structure the angle which is needed is passed to the CORDIC algorithm which gives the sine and cosine of the angle except that in the last stage. In the last stage, the twiddle factor is 1 always, therefore the direct value is saved for cosine and sine, i.e., sine of the angle is 0 and cosine of the angle is 1.

Saving the value 1 in a variable makes it easy to compute the values in the last stage which reduces time by 4 times while using the CORDIC algorithm in the fourth (last) stage of 16-point FFT. Thus, in the last stage, both the complexity as well as time taken is reduced.

The code for radix-2 written in Verilog is universal, i.e., on varying the number of data and data values the design calculates FFT itself, so this can be used for calculating any point-FFT. The output has been shown in Figs. 3–7. The final output comes at 4015 ns. But extra delays were added just for the sake of proper visibility in Fig. 7.



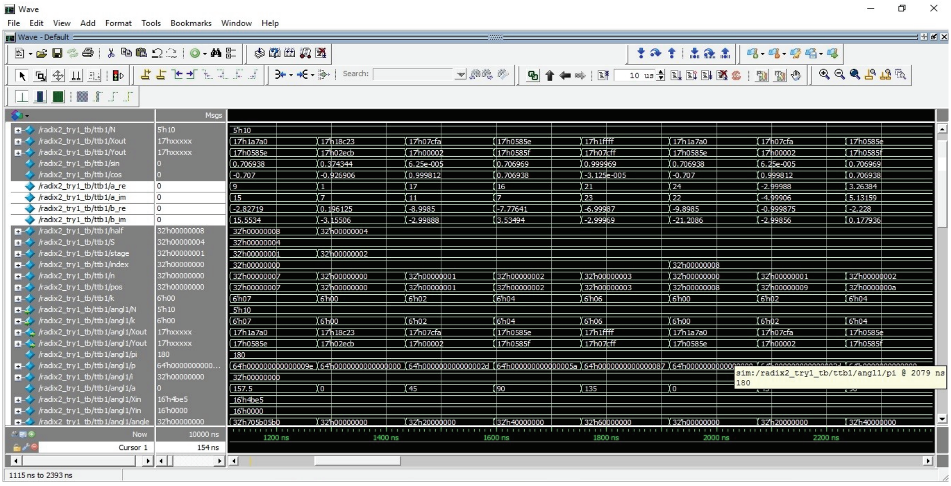
```
/radix2_try1_tb/ttb1/N             5'h10             5'h10
/radix2_try1_tb/ttb1/Xout          17'hxxxxx                17'h07cfa   17'h0730d   17'h0585e   17'h02ed3   17'h1ffff   17'h1cf2d   17'h1a7a0
/radix2_try1_tb/ttb1/Yout          17'hxxxxx                17'h00002   17'h030d2   17'h0585f   17'h073dd   17'h07cff   17'h0730b   17'h0585e
/radix2_try1_tb/ttb1/sin           0                 0      6.25e-005   0.390562    0.706969    0.926906    0.999969    0.920344    0.706938
/radix2_try1_tb/ttb1/cos           0                 0      0.999812    0.920406    0.706938    0.374594    -3.125e-005 -0.390594   -0.707
/radix2_try1_tb/ttb1/a_re          0                 0      4                       12          23          13          12          9
/radix2_try1_tb/ttb1/a_im          0                 0      4           2           8           15          7           5           15
/radix2_try1_tb/ttb1/b_re          0                 0      -1.99988    0.781125    -2.82769    -5.79662    -1          2.48272     -2.82719
/radix2_try1_tb/ttb1/b_im          0                 0      -3.99912    1.84081     5.65569     11.6752     -0.999937   3.29078     15.5534
/radix2_try1_tb/ttb1/half          32'h00000008      32'h00000008
/radix2_try1_tb/ttb1/S             32'h00000004      32'h00000004
/radix2_try1_tb/ttb1/stage         32'h00000001      32'h00000001
/radix2_try1_tb/ttb1/index         32'h00000000      32'h00000000
/radix2_try1_tb/ttb1/n             32'h00000000      32'h00000000 32'h00000001 32'h00000002 32'h00000003 32'h00000004 32'h00000005 32'h00000006 32'h00000007
/radix2_try1_tb/ttb1/pos           32'h00000000      32'h00000000 32'h00000001 32'h00000002 32'h00000003 32'h00000004 32'h00000005 32'h00000006 32'h00000007
/radix2_try1_tb/ttb1/k             6'h00             6'h00  6'h01       6'h02       6'h03       6'h04       6'h05       6'h06       6'h07
/radix2_try1_tb/ttb1/angl1/N       5'h10             5'h10
/radix2_try1_tb/ttb1/angl1/k       6'h00             6'h00  6'h01       6'h02       6'h03       6'h04       6'h05       6'h06       6'h07
/radix2_try1_tb/ttb1/angl1/Xout    17'hxxxxx                17'h07cfa   17'h0730d   17'h0585e   17'h02ed3   17'h1ffff   17'h1cf2d   17'h1a7a0
/radix2_try1_tb/ttb1/angl1/Yout    17'hxxxxx                17'h00002   17'h030d2   17'h0585f   17'h073dd   17'h07cff   17'h0730b   17'h0585e
/radix2_try1_tb/ttb1/angl1/pi      180               180
/radix2_try1_tb/ttb1/angl1/p       64'h0000000000... 64'h0000000000000... 64'h0000000000000017 64'h000000000000002d 64'h0000000000000044 64'h000000000000005a 64'h0000000000000071 64'h0000000000000087 64'h000000000000009e
/radix2_try1_tb/ttb1/angl1/i       32'h00000000      32'h00000000
/radix2_try1_tb/ttb1/angl1/a       0                 0      22.5        45          67.5        90          112.5       135         157.5
/radix2_try1_tb/ttb1/angl1/Xin     16'h4be5          16'h4be5
/radix2_try1_tb/ttb1/angl1/Yin     16'h0000          16'h0000
/radix2_try1_tb/ttb1/angl1/angle   32'h00000000      32'h00000000 32'h105b05b0 32'h20000000 32'h305b05b0 32'h40000000 32'h505b05b0 32'h60000000 32'h705b05b0

Now        10000 ns
Cursor 1   154 ns          154 ns

sim:/radix2_try1_tb/ttb1/angl1/angle @ 564 ns
32'h305b05b0

0 ns to 1278 ns     /radix2_try1_tb/ttb1/angl1/a
```

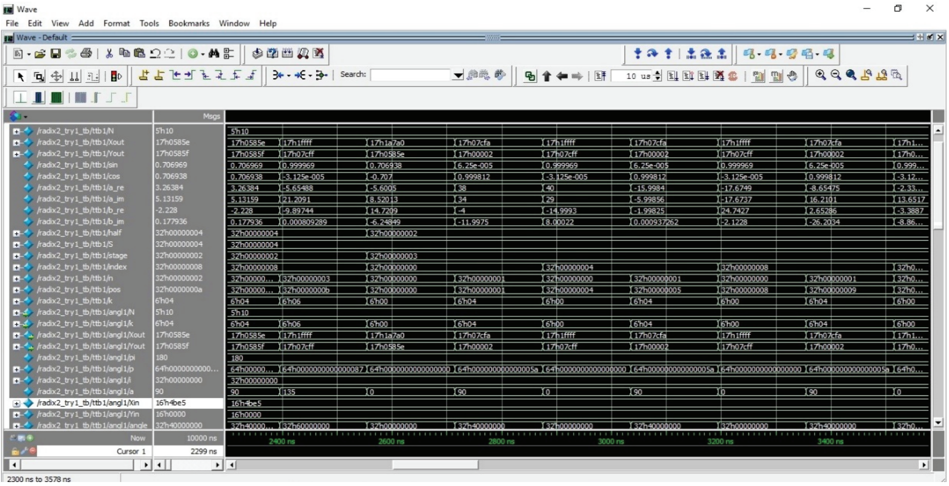
Fig. 3. Simulation result of 16-point FFT on ModelSim (Wave Window) [0 ns–1200 ns].


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
10 us
1200 ns
1400 ns
1600 ns
1800 ns
2000 ns
2200 ns
sim:/radix2_try1_tb/ttb1/angl1/pi @ 2079 ns
180

	Msgs
/radix2_try1_tb/ttb1/N	5'h10
/radix2_try1_tb/ttb1/Xout	17'hxxxxx
/radix2_try1_tb/ttb1/Yout	17'hxxxxx
/radix2_try1_tb/ttb1/sin	0
/radix2_try1_tb/ttb1/cos	0
/radix2_try1_tb/ttb1/a_re	0
/radix2_try1_tb/ttb1/a_im	0
/radix2_try1_tb/ttb1/b_re	0
/radix2_try1_tb/ttb1/b_im	0
/radix2_try1_tb/ttb1/half	32'h00000008
/radix2_try1_tb/ttb1/S	32'h00000004
/radix2_try1_tb/ttb1/stage	32'h00000001
/radix2_try1_tb/ttb1/index	32'h00000000
/radix2_try1_tb/ttb1/n	32'h00000000
/radix2_try1_tb/ttb1/pos	32'h00000000
/radix2_try1_tb/ttb1/k	6'h00
/radix2_try1_tb/ttb1/angl1/N	5'h10
/radix2_try1_tb/ttb1/angl1/k	6'h00
/radix2_try1_tb/ttb1/angl1/Xout	17'hxxxxx
/radix2_try1_tb/ttb1/angl1/Yout	17'hxxxxx
/radix2_try1_tb/ttb1/angl1/pi	180
/radix2_try1_tb/ttb1/angl1/p	64'h0000000000...
/radix2_try1_tb/ttb1/angl1/i	32'h00000000
/radix2_try1_tb/ttb1/angl1/a	0
/radix2_try1_tb/ttb1/angl1/Xin	16'h4be5
/radix2_try1_tb/ttb1/angl1/Yin	16'h0000
/radix2_try1_tb/ttb1/angl1/angle	32'h00000000
Now	10000 ns
Cursor 1	154 ns

1115 ns to 2393 ns

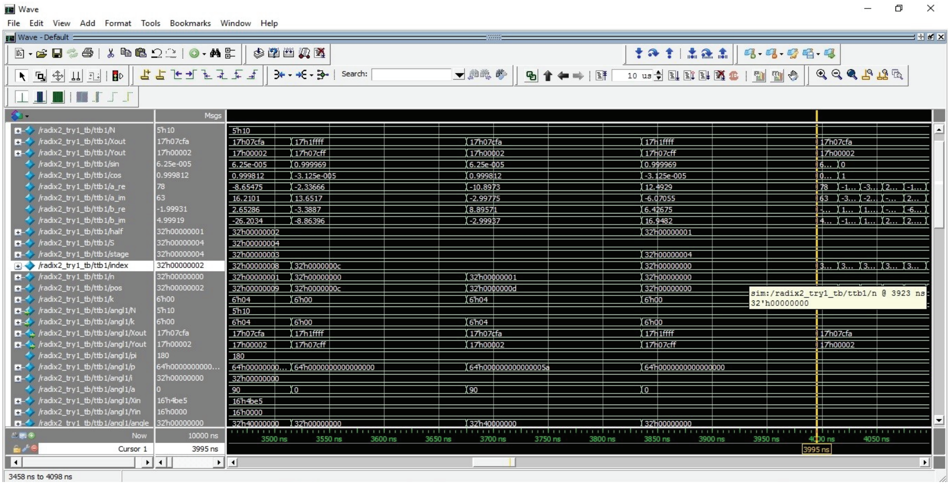
Fig. 4. Simulation result of 16-point FFT on ModelSim (Wave Window) [1200 ns–2300 ns].



	Msgs
/radix2_try1_tb/ttb1/N	5'h10
/radix2_try1_tb/ttb1/Xout	17'h0585e
/radix2_try1_tb/ttb1/Yout	17'h0585f
/radix2_try1_tb/ttb1/sin	0.706969
/radix2_try1_tb/ttb1/cos	0.706938
/radix2_try1_tb/ttb1/a_re	3.26384
/radix2_try1_tb/ttb1/a_im	5.13159
/radix2_try1_tb/ttb1/b_re	-2.228
/radix2_try1_tb/ttb1/b_im	0.177936
/radix2_try1_tb/ttb1/half	32'h00000004
/radix2_try1_tb/ttb1/S	32'h00000004
/radix2_try1_tb/ttb1/stage	32'h00000002
/radix2_try1_tb/ttb1/index	32'h00000008
/radix2_try1_tb/ttb1/n	32'h00000002
/radix2_try1_tb/ttb1/pos	32'h0000000a
/radix2_try1_tb/ttb1/k	6'h04
/radix2_try1_tb/ttb1/angl1/N	5'h10
/radix2_try1_tb/ttb1/angl1/k	6'h04
/radix2_try1_tb/ttb1/angl1/Xout	17'h0585e
/radix2_try1_tb/ttb1/angl1/Yout	17'h0585f
/radix2_try1_tb/ttb1/angl1/pi	180
/radix2_try1_tb/ttb1/angl1/p	64'h0000000000...
/radix2_try1_tb/ttb1/angl1/i	32'h00000000
/radix2_try1_tb/ttb1/angl1/a	90
/radix2_try1_tb/ttb1/angl1/Xin	16'h4be5
/radix2_try1_tb/ttb1/angl1/Yin	16'h0000
/radix2_try1_tb/ttb1/angl1/angle	32'h40000000
Now	10000 ns
Cursor 1	2299 ns

2300 ns to 3578 ns

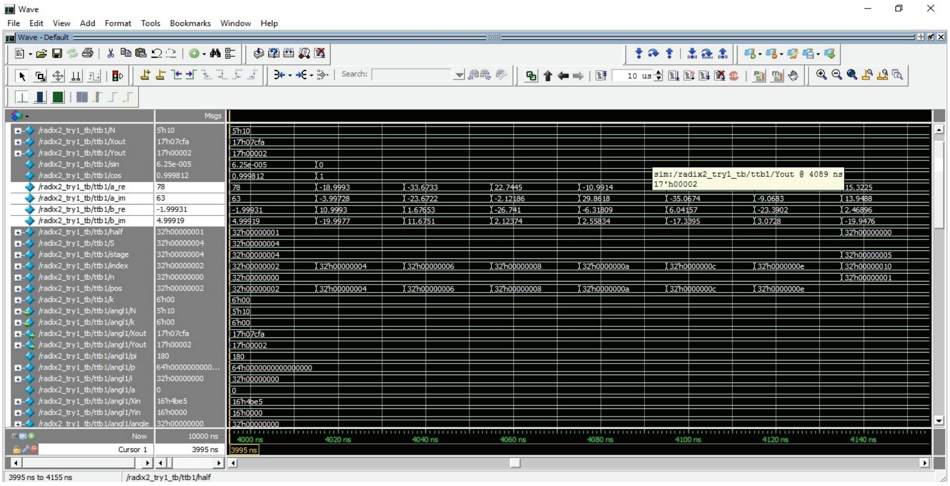
Fig. 5. Simulation result of 16-point FFT on ModelSim (Wave Window) [2300 ns–3500 ns].



	Msgs
/radix2_try1_tb/ttb1/N	5'h10
/radix2_try1_tb/ttb1/Xout	17'h07cfa
/radix2_try1_tb/ttb1/Yout	17'h00002
/radix2_try1_tb/ttb1/sin	6.25e-005
/radix2_try1_tb/ttb1/cos	0.999812
/radix2_try1_tb/ttb1/a_re	78
/radix2_try1_tb/ttb1/a_im	63
/radix2_try1_tb/ttb1/b_re	-1.99931
/radix2_try1_tb/ttb1/b_im	4.99919
/radix2_try1_tb/ttb1/half	32'h00000001
/radix2_try1_tb/ttb1/S	32'h00000004
/radix2_try1_tb/ttb1/stage	32'h00000004
/radix2_try1_tb/ttb1/index	32'h00000002
/radix2_try1_tb/ttb1/n	32'h00000000
/radix2_try1_tb/ttb1/pos	32'h00000002
/radix2_try1_tb/ttb1/k	6'h00
/radix2_try1_tb/ttb1/angl1/N	5'h10
/radix2_try1_tb/ttb1/angl1/k	6'h00
/radix2_try1_tb/ttb1/angl1/Xout	17'h07cfa
/radix2_try1_tb/ttb1/angl1/Yout	17'h00002
/radix2_try1_tb/ttb1/angl1/pi	180
/radix2_try1_tb/ttb1/angl1/p	64'h0000000000...
/radix2_try1_tb/ttb1/angl1/i	32'h00000000
/radix2_try1_tb/ttb1/angl1/a	0
/radix2_try1_tb/ttb1/angl1/Xin	16'h4be5
/radix2_try1_tb/ttb1/angl1/Yin	16'h0000
/radix2_try1_tb/ttb1/angl1/angle	32'h00000000
Now	10000 ns
Cursor 1	3995 ns

3458 ns to 4098 ns

Fig. 6. Simulation result of 16-point FFT on ModelSim (Wave Window) [3500 ns–4000 ns].



	Msgs	
/radix2_try1_tb/ttb1/N	5'h10	5'h10
/radix2_try1_tb/ttb1/Xout	17'h07cfa	17'h07cfa
/radix2_try1_tb/ttb1/Yout	17'h00002	17'h00002
/radix2_try1_tb/ttb1/sin	6.25e-005	6.25e-005 0
/radix2_try1_tb/ttb1/cos	0.999812	0.999812 1
/radix2_try1_tb/ttb1/a_re	78	78 -18.9993 -33.6733 22.7445 -10.9914 15.3225
/radix2_try1_tb/ttb1/a_im	63	63 -3.99728 -23.6722 -2.12186 29.8618 -35.0674 -9.0683 13.9488
/radix2_try1_tb/ttb1/b_re	-1.99931	-1.99931 10.9993 1.67653 -26.741 -6.31809 6.04157 -23.3902 2.46896
/radix2_try1_tb/ttb1/b_im	4.99919	4.99919 -19.9977 11.6751 2.12374 2.55834 -17.3395 3.0728 -19.9476
/radix2_try1_tb/ttb1/half	32'h00000001	32'h00000001 32'h00000000
/radix2_try1_tb/ttb1/S	32'h00000004	32'h00000004
/radix2_try1_tb/ttb1/stage	32'h00000004	32'h00000004 32'h00000005
/radix2_try1_tb/ttb1/index	32'h00000002	32'h00000002 32'h00000004 32'h00000006 32'h00000008 32'h0000000a 32'h0000000c 32'h0000000e 32'h00000010
/radix2_try1_tb/ttb1/n	32'h00000000	32'h00000000 32'h00000001
/radix2_try1_tb/ttb1/pos	32'h00000002	32'h00000002 32'h00000004 32'h00000006 32'h00000008 32'h0000000a 32'h0000000c 32'h0000000e
/radix2_try1_tb/ttb1/k	6'h00	6'h00
/radix2_try1_tb/ttb1/angl1/N	5'h10	5'h10
/radix2_try1_tb/ttb1/angl1/k	6'h00	6'h00
/radix2_try1_tb/ttb1/angl1/Xout	17'h07cfa	17'h07cfa
/radix2_try1_tb/ttb1/angl1/Yout	17'h00002	17'h00002
/radix2_try1_tb/ttb1/angl1/pi	180	180
/radix2_try1_tb/ttb1/angl1/p	64'h0000000000...	64'h0000000000000000
/radix2_try1_tb/ttb1/angl1/i	32'h00000000	32'h00000000
/radix2_try1_tb/ttb1/angl1/a	0	0
/radix2_try1_tb/ttb1/angl1/Xin	16'h4be5	16'h4be5
/radix2_try1_tb/ttb1/angl1/Yin	16'h0000	16'h0000
/radix2_try1_tb/ttb1/angl1/angle	32'h00000000	32'h00000000
Now	10000 ns	
Cursor 1	3995 ns	3995 ns

```
sim:/radix2_try1_tb/ttb1/Yout @ 4089 ns
17'h00002
```

3995 ns to 4155 ns	/radix2_try1_tb/ttb1/half

Fig. 7. Simulation result of 16-point FFT on ModelSim (Wave Window - last stage) [4000 ns–4150 ns].

Figs. 3–7 show the FFT simulation of 16-point input using radix-2 algorithm. The parameter a_re, a_im, b_re, and b_im are the outputs of the radix-2 butterfly structure as shown in Fig. 8.

The output result obtained from this proposed work is compared with MATLAB simulations in Fig. 9 using the same algorithm. The MATLAB simulation is used for verification of results (in Fig. 9, 'x' is the input variable and 'y' is the output variable).

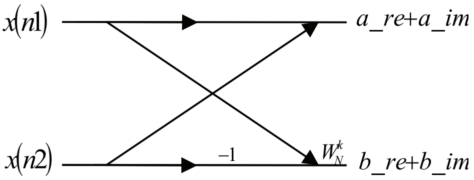
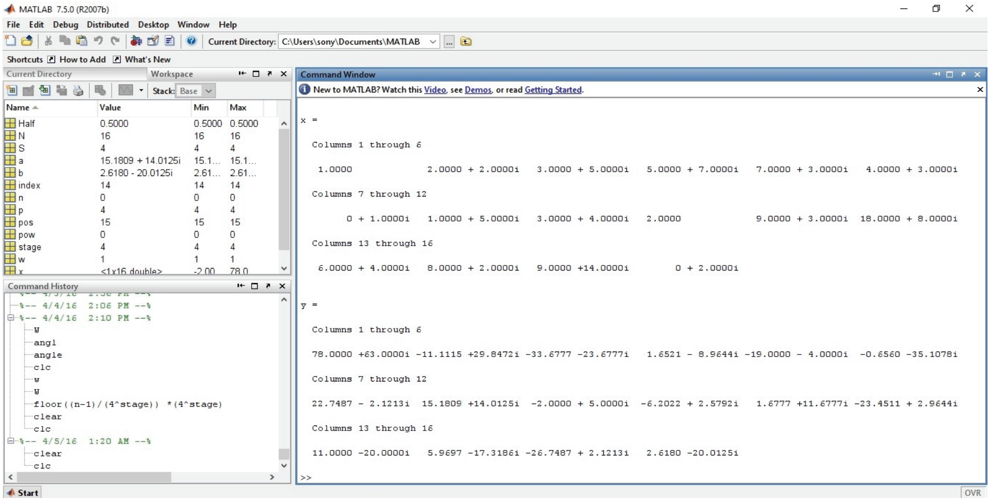


Fig. 8. Butterfly structure.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Users\sony\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command History
Command Window
Start
OVR

Name	Value	Min	Max
Half	0.5000	0.5000	0.5000
N	16	16	16
S	4	4	4
a	15.1809 + 14.0125i	15.1...	15.1...
b	2.6180 - 20.0125i	2.61...	2.61...
index	14	14	14
n	0	0	0
p	4	4	4
pos	15	15	15
pow	0	0	0
stage	4	4	4
w	1	1	1
x	<1x16 double>	-2.00	78.0

```
%-- 4/4/16  2:06 PM --%
%-- 4/4/16  2:10 PM --%
  W
  angl
  angle
  clc
  w
  W
  floor((n-1)/(4^stage)) *(4^stage)
  clear
  clc
%-- 4/5/16  1:20 AM --%
  clear
  clc
```

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
x =

  Columns 1 through 6

   1.0000             2.0000 + 2.0000i   3.0000 + 5.0000i   5.0000 + 7.0000i   7.0000 + 3.0000i   4.0000 + 3.0000i

  Columns 7 through 12

        0 + 1.0000i   1.0000 + 5.0000i   3.0000 + 4.0000i   2.0000             9.0000 + 3.0000i  18.0000 + 8.0000i

  Columns 13 through 16

   6.0000 + 4.0000i   8.0000 + 2.0000i   9.0000 +14.0000i        0 + 2.0000i


y =

  Columns 1 through 6

  78.0000 +63.0000i -11.1115 +29.8472i -33.6777 -23.6777i   1.6521 - 8.9644i -19.0000 - 4.0000i  -0.6560 -35.1078i

  Columns 7 through 12

  22.7487 - 2.1213i  15.1809 +14.0125i  -2.0000 + 5.0000i  -6.2022 + 2.5792i   1.6777 +11.6777i -23.4511 + 2.9644i

  Columns 13 through 16

  11.0000 -20.0000i   5.9697 -17.3186i -26.7487 + 2.1213i   2.6180 -20.0125i

>>
```

Fig. 9. Simulation result of 16-point FFT on MATLAB (x is input and y is output).

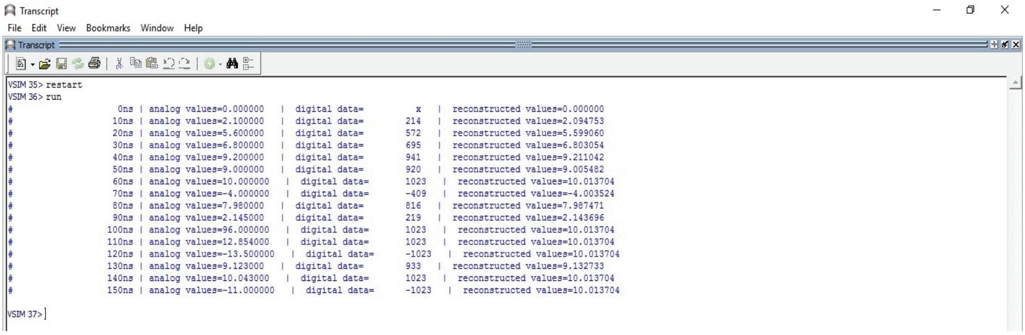
It has been observed that the result of the design simulated using Verilog on ModelSim is almost the same as the results obtained in MATLAB. The comparisons of results of 16-point FFT using Verilog and MATLAB are mentioned in Table 2.

Table 2. Comparison of outputs from Verilog with MATLAB for 16-point FFT

Series no.	Input values	Verilog outputs		MATLAB output	
		Real part	Imaginary part	Real part	Imaginary part
0	1+0j	78	63	78	63
1	2+2j	-10.9914	29.8618	-11.1115	29.8472
2	3+5j	-33.6733	-23.6722	-33.6777	-23.6777
3	5+7j	1.59562	-9.0683	1.6521	-8.9644
4	7+3j	-18.9993	-3.99728	-19	-4
5	4+3j	-0.735838	-35.0674	-0.656	-35.1078
6	0+1j	22.7445	-2.12186	22.7487	-2.1213
7	1+5j	15.3225	13.9488	15.1809	14.0125
8	3+4j	-1.99931	4.99919	-2	5
9	2+0j	-6.31809	2.55834	-6.2022	2.5792
10	9+3j	1.67653	11.6751	1.6777	11.6777
11	18+8j	-23.3902	3.0728	-23.4511	2.9644
12	6+4j	10.9993	-19.9977	11	-20
13	8+2j	6.04157	-17.3395	5.9697	-17.3186
14	9+14j	-26.741	2.12374	-26.7487	2.1213
15	0+2j	2.46896	-19.9476	2.618	-20.0125

2.2 ADC and DAC

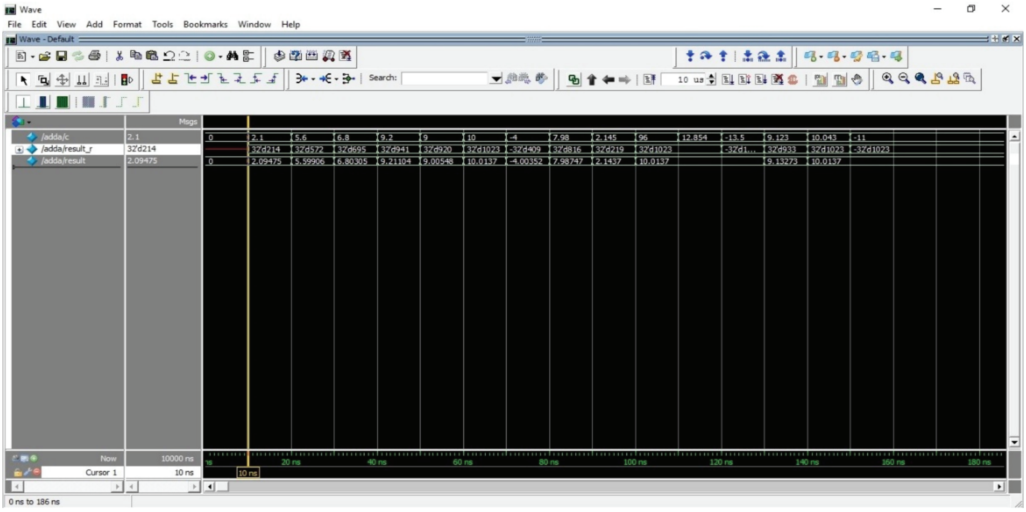
The input signal is analog which takes in values from +10 V to -10 V and converts them to +1023 to -1023 which comes as output from the digital signal. Any value greater than 10 V will be limited to its maximum valu, i.e., +1023. Similarly, any value lower than -10 V will be limited to -1023. The signal gets modulated using a carrier signal and the signal is quantized at few intervals. And the reverse procedure is used for implementing DAC. The simulation result is shown in Figs. 10 and 11. The simulation contains conversion of analog to digital values and reconstruction of digital values back to analog again.



```
VSIM 35> restart
VSIM 36> run
#                  0ns | analog values=0.000000   |  digital data=        x   |  reconstructed values=0.000000
#                 10ns | analog values=2.100000   |  digital data=      214   |  reconstructed values=2.094753
#                 20ns | analog values=5.600000   |  digital data=      572   |  reconstructed values=5.599060
#                 30ns | analog values=6.800000   |  digital data=      695   |  reconstructed values=6.803054
#                 40ns | analog values=9.200000   |  digital data=      941   |  reconstructed values=9.211042
#                 50ns | analog values=9.000000   |  digital data=      920   |  reconstructed values=9.005482
#                 60ns | analog values=10.000000   |  digital data=      1023   |  reconstructed values=10.013704
#                 70ns | analog values=-4.000000   |  digital data=      -409   |  reconstructed values=-4.003524
#                 80ns | analog values=7.980000   |  digital data=      816   |  reconstructed values=7.987471
#                 90ns | analog values=2.145000   |  digital data=      219   |  reconstructed values=2.143696
#                100ns | analog values=96.000000   |  digital data=      1023   |  reconstructed values=10.013704
#                110ns | analog values=12.854000   |  digital data=      1023   |  reconstructed values=10.013704
#                120ns | analog values=-13.500000   |  digital data=      -1023   |  reconstructed values=10.013704
#                130ns | analog values=9.123000   |  digital data=      933   |  reconstructed values=9.132733
#                140ns | analog values=10.043000   |  digital data=      1023   |  reconstructed values=10.013704
#                150ns | analog values=-11.000000   |  digital data=      -1023   |  reconstructed values=10.013704

VSIM 37>]
```

Fig. 10. Simulation result of ADC and DAC on ModelSim (Transcript Window).



```
/adda/c          2.1       0  2.1      5.6      6.8      9.2      9        10        -4        7.98     2.145    96                 12.854  -13.5      9.123    10.043    -11
/adda/result_r   32'd214      32'd214  32'd572  32'd695  32'd941  32'd920  32'd1023  -32'd409  32'd816  32'd219  32'd1023                   -32'd1...  32'd933  32'd1023  -32'd1023
/adda/result     2.09475   0  2.09475  5.59906  6.80305  9.21104  9.00548  10.0137   -4.00352  7.98747  2.1437   10.0137                               9.13273  10.0137

Now       10000 ns   ns  20 ns  40 ns  60 ns  80 ns  100 ns  120 ns  140 ns  160 ns  180 ns
Cursor 1  10 ns      10 ns
```

Fig. 11. Simulation result of ADC and DAC on ModelSim (Wave Window).

In Fig. 10, the transcript shows the input analog signal ('analog value'), its converted digital signal ('digital data') and finally the reconstructed analog signal ('reconstructed value') from digital data. Fig. 11 shows the wave simulation ('c' is the input analog signal, 'result_r' is converted digital signal and 'result' is the reconstructed analog signal.).

3. Conclusion and Future Scope

Nowadays OFDM is an important technology that supports the latest standards of wireless communication. To achieve high data rate, we use OFDM in LTE, WiMAX and WLAN, etc. The main modules of an OFDM system, i.e., the FFT and the IFFT, A/D and D/A have been simulated successfully using radix-2 algorithm and reducing the complex mathematics.

This design works better for large N (in N-point FFT) than radix-M (M>2) because the number of multiplications, addition and subtraction is equal to one in radix-2 while the number of multiplications is more in others thus leading to a more complex design. Also, the last stage of the design is much more time efficient than the existing designs (since the time taken is optimized in the last stage).

For A/D and D/A systems, we implemented the architecture using some common algorithms which include simple mathematics reducing its complexity. The proposed FFT design is both speed and time efficient. It is a method in which the resources have been used in an optimized way.

The high data rate requirement in wireless communication has increased exponentially lately. In this paper, we take a review on the traditional concept in terms of its disadvantages and advantages and have tried to improve them. The future networks will continue to rely on OFDM/OFDMA for satisfying its need.

References

[1] M. Mohaisen, "A review of fixed-complexity vector perturbation for MU-MIMO," *Journal of Information Processing Systems*, vol. 11, no. 3, pp. 354-369, 2015.

[2] A. S. M. M. Rahaman, M. I. Islam, and M. R. Amin, "Selection of the best two-hop AF wireless link under multiple antenna schemes over a fading channel," *Journal of Information Processing Systems*, vol. 11, no. 1, pp. 57-75, 2015.

[3] N. Chide, S. Deshmukh, and P. B. Borole, "Implementation of OFDM system using IFFT and FFT," *International Journal of Engineering Research and Applications (IJERA)*, vol.3, no. 1, pp. 2009-2014, 2013.

[4] A. Fatima, "Design and simulation of 32-point FFT using radix-2 algorithm," *IOSR Journal of Electrical and Electronics Engineering*, vol. 9, no. 1, pp. 42-50, 2014.

[5] M. Merlyn, "FPGA implementation Of FFT processor with OFDM transceiver" in *Proceedings of 2010 International Conference on Signal and Image Processing*, Chennai, India, 2010, pp. 483-489.

[6] F. Ahmed, M. L. Ali, and M. I. H. B. Asad, "Design of a high speed OFDM transmitter and receiver," in *Proceedings of the 8th International Conference on Electrical and Computer Engineering*, Dhaka, Bangladesh, 2014, pp. 425-428.

[7] K. Mehta "High performance and area efficient correlation for IEEE 802.16 OFDM using FPGA," *International Journal of Advanced Engineering and Research Development*, vol. 2, no. 3, pp. 93-97, 2015.

[8] A. D. Palekar and D. P. V. Ingole, "OFDM system using FFT and IFFT," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 12, pp. 675-679, 2013.

[9] H. K. Sharma and A. K. Singh, "OFDM system using FFT and IFFT," *Global Journal of Engineering, Science & Social Science Studies*, vol. 1, no. 2, pp 202-208, 2015.

[10] M. A. Mohamed, A. S. Samarah, and M. F. Allah, "Implementation of the OFDM physical layer using FPGA," *International Journal of Computer Science Issues*, vol. 9, no. 2, pp. 612-618, 2012.

[11] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "Improved radix-4 and radix-8 FFT algorithms," in *Proceedings of 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512)*, Vancouver, Canada, 2004, pp. 561-564.

[12] K. M. Rao, V. R. Tejesvi, and G. A. Rao, "Verilog implementation of 32 point FFT using Radix-2 algorithm on FPGA technology," *IOSR Journal of Electronics and Communication Engineering*, vol. 9, no. 2, pp. 40-43, 2014.

[13] N. V. Mahajan and J. S. Chitode, "Simple computation of DIT FFT," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 5, pp. 353-356, 2014.

[14] R. Ramachandran and K. Vanmathi, "Simulation of radix-2 fast Fourier transform using Xilinx," *International Journal of Computer Science Engineering (IJCSE)*, vol. 3, no. 2, pp. 125-130, 2014.

[15] G. Suarez, "Behavioral Modeling of Data Converters using Verilog-A," Department of Electrical and Computer Engineering, University of Puerto Rico, Mayaguez. Puerto Rico, pp. 17-19, 2014.

[16] A. Ashraf, S. Ashraf, N. Z. Rizvi, and S. A. Dar, "Low power design of asynchronous SAR ADC," in *Proceedings of 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, India, 2016, pp. 4214-4219.

[17] M. Shanmugasundaram and S. Pavan, "Rapid simulation of current steering digital-to-analog converters using Verilog-A," in *Proceedings of IEEE Custom Integrated Circuits Conference 2006*, San Jose, CA, 2006, pp. 201-204.

[18] M. Santhanalakshmi and K. Yasoda, "Verilog-A implementation of energy-efficient SAR ADCs for biomedical application," in *Proceedings of 2015 19th International Symposium on VLSI Design and Test*, Ahmedabad, India, 2015, pp. 1-6.

[19] J. Mentzer and T. Wey, "A Verilog mixed signal model of a 10-bit pipeline analog-to-digital converter," in *Proceedings of 2006 IEEE International Behavioral Modeling and Simulation Workshop*, San Jose, CA, 2006, pp. 115-119.

[20] S. S. Pujari, P. P. Muduli, A. Panda, R. Badhai, S. Nayak, and Y. Sahoo, "Design & implementation of FIR filters using on-board ADC-DAC & FPGA," in *Proceedings of International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai, India, pp. 1-6.

Alok Joshi https://orcid.org/0000-0002-5332-0740

He has received B.E. degree in Electronics and Communication Engineering from G.B. Pant Enggineering College (Hemvati Nandan Bahuguna Garhwal University), Uttarakhand, India in 2001 and M.Tech. in Digital Communication from Uttar Pradesh Technical University, Lucknow, India in 2006. He received Ph.D. degree from Jaypee University of Information Technology in 2014, in the area of wireless communications. He has total teaching experience of 15 years in various technical universities in India. Currently he is working with Jaypee Institute of Information Technology, Noida, India. His research interests are MIMO OFDM systems.

Dewansh Aditya Gupta https://orcid.org/0000-0003-3289-3889

He received his B.Tech. degree in Electronics and Communication Engineering from Jaypee Institute of Information Technology, Noida, India in 2016. He is currently working as a Software Engineer in the Synthesis domain at Cadence Design Systems, Noida. He received his B.Tech. degree in Electronics and Communication Engineering from Jaypee Institute of Information Technology, Noida, India in 2016. His interest areas include frontend optimizations and mapping algorithms. His research interest is mainly in VLSI and EDA domain.

Pravriti Jaipuriyar https://orcid.org/0000-0002-8306-9251

She received her B.Tech. degree in Electronics and Communication Engineering from Jaypee Institute of Information Technology, Noida, India in 2016. She is currently working as a Software Engineer in the Formal and IP verification domain at Cadence Design Systems, Noida. She received her B.Tech. degree in Electronics and Communication Engineering from Jaypee Institute of Information Technology, Noida, India in 2016. Her research interest is mainly in IP development and verification. Her research interest is mainly in VLSI, IP and EDA domain.