DESIGN & SIMULATION PARALLEL PIPELINED RADIX -2^2 FFT ARCHITECTURE FOR REAL VALUED SIGNALS

Madhavi S.Kapale[#1],
[1]Student Mtech Electronics Engineering (Communication)
Vidarbha Institute of Technology,Nagpur,India
[1]madhvikapale@gmail.com,

Prof.Nilesh P. Bodne[#2]
[2] Assistant Professor
Vidarbha Institute of Technology,Nagpur , India
[2]nileshbodne@gmail.com

ABSTRACT*- The efficient implementation of FFT/IFFT processor for OFDM applications is presented. Indeed, the proposed radix-2^k feed-forward architectures require fewer hardware resources than parallel feedback ones, also called multi-path delay feedback (MDF), when several samples in parallel must be processed. Multiplier make a parallel connection will give output in one clock cycle ,independent of the number cases form bit multiplier it requires n clock cycle which makes it slow so, for 16 clock cycle while in our case output will come in one clock cycle, when several samples in parallel must be processed. Multiplier make a parallel connection will give output in one clock cycle ,independent of the number cases form bit multiplier it requires n clock cycle which makes it slow so, for 16 clock cycle while in our case output will come in one clock cycle.*

Keywords: Parallel, Radix-2^2^,FFT, IFFT, OFDM, Modified FFT/IFFT, OFDM, Pipeline, spectral efficiency.

I. Introduction

1.1 Fast Fourier Transform (FFT)

In this section we present several methods for computing the DFT efficiently. 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

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

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

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

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

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 (4*N* real multiplications) and *N*-1 complex additions (4*N*-2 real additions). Consequently, to compute all *N* values of the DFT requires N^2 complex multiplications and N^2-*N* complex additions. Direct computation of the DFT is basically inefficient primarily because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular, these two properties are :

$$\text{Symmetry property}: W_N^{k+N/2} = -W_N^k$$

$$\text{Periodicit y property}: W_N^{k+N} = W_N^k$$

The computationally efficient algorithms described in this section, known collectively as fast Fourier transform (FFT) algorithms, exploit these two basic properties of the phase factor.

1.2 Radix-2 FFT Algorithms

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

$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1), \qquad n = 0,1,\ldots,N/2 - 1$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating $x(n)$ by a factor of 2, and hence the resulting FFT algorithm is called a *decimation-in-time algorithm.*

Now the N-point DFT can be expressed in terms of the DFT's of the decimated sequences as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots,N-1$$
$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$
$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$

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

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km}$$
$$= F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,N-1$$

where $F_1(k)$ 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 periodic, with period $N/2$, we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. In addition, the factor $W_N^{k+N/2} = -W_N^k$. Hence the equation may be expressed as

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2} - 1$$
$$X(k+\frac{N}{2}) = F_1(k) - W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2} - 1$$

We observe that the direct computation of $F_1(k)$ requires $(N/2)^2$ complex multiplications. The same applies to the computation of $F_2(k)$. Furthermore, there are $N/2$ additional complex multiplications required to compute $W_N^k F_2(k)$. Hence the computation of $X(k)$ requires $2(N/2)^2 + N/2 = N^2/2 + N/2$ complex multiplications. This first step results in 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.

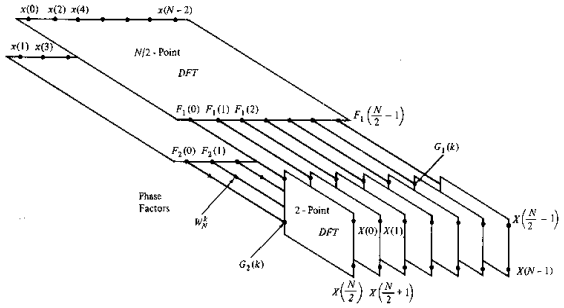


Fig.1.1 First step in the decimation-in-time algorithm.

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

$$\mathrm{F}\{f_1(2n+1)\}, \qquad k = 0,1,\ldots,\frac{N}{4} - 1; \qquad n = 0,1,\ldots,\frac{N}{4} - 1$$
$$\mathrm{F}\{f_1(2n+1)\}, \qquad k = 0,1,\ldots,\frac{N}{4} - 1; \qquad n = 0,1,\ldots,\frac{N}{4} - 1$$
$$\mathrm{F}\{f_2(2n+1)\}, \qquad k = 0,1,\ldots,\frac{N}{4} - 1; \qquad n = 0,1,\ldots,\frac{N}{4} - 1$$
$$\mathrm{F}\{f_2(2n+1)\}, \qquad k = 0,1,\ldots,\frac{N}{4} - 1; \qquad n = 0,1,\ldots,\frac{N}{4} - 1$$

represents Fourier tr ansform

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 $N = 8$

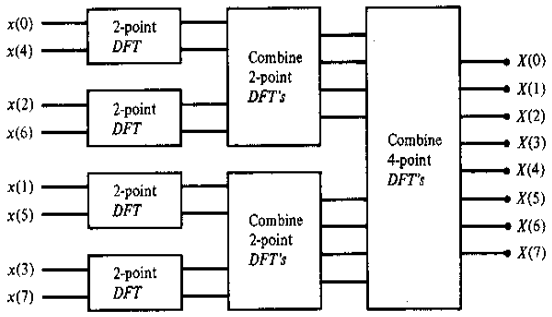


Fig 1.2 Three stages in the computation of an $N = 8$-point DFT.

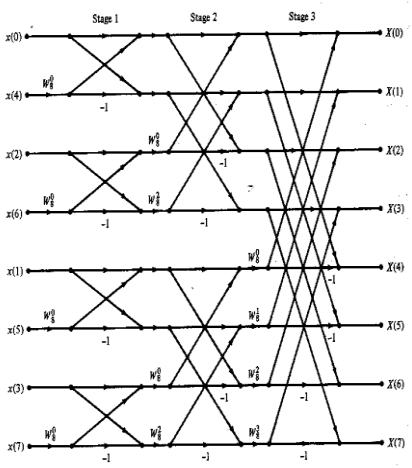


Fig 1.3 Eight-point decimation-in-time FFT algorithm.

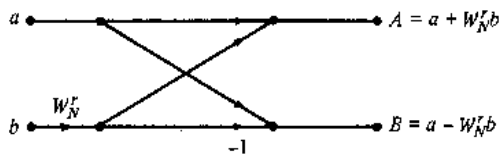


Fig. 1.4 Basic butterfly computation in the decimation-in-time FFT algorithm.

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.

1.3 Radix-4 FFT Algorithm

When the number of data points N in the DFT is a power of 4 (i.e., $N = 4^v$), we can, of course, always use a radix-2 algorithm for the computation. However, for this case, it is more efficient computationally to employ a radix-r FFT algorithm. Let us begin by describing a radix-4 decimation-in-time FFT algorithm briefly. We split or decimate the N-point input sequence into four subsequences, $x(4n)$, $x(4n+1)$, $x(4n+2)$, $x(4n+3)$, $n = 0, 1, \ldots, N/4-1$.

$$X(p,q) = \sum_{l=0}^{3} \left[W_N^{lq} F(l,q) \right] W_4^{lp}$$

$$F(l,q) = \sum_{m=0}^{(N/4)-1} x(l,m) W_{N/4}^{mq}$$

$$p = 0,1,2,3; \quad l = 0,1,2,3; \quad q = 0,1,2,\ldots,\frac{N}{4}-1$$

and

$$x(l,m) = x(4m+1)$$

$$X(p,q) = X(\frac{N}{4}p + q)$$

Thus the four $N/4$-point DFTs $F(l, q)$obtained from the above equation are combined to yield the N-point DFT. The expression for combining the $N/4$-point DFTs defines a radix-4 decimation-in-time butterfly, which can be expressed in matrix form as

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^q F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix}$$

The radix-4 butterfly is depicted in and in a more compact form . Note that each butterfly involves three complex multiplications, since $W_N^0 = 1$, and 12 complex additions.

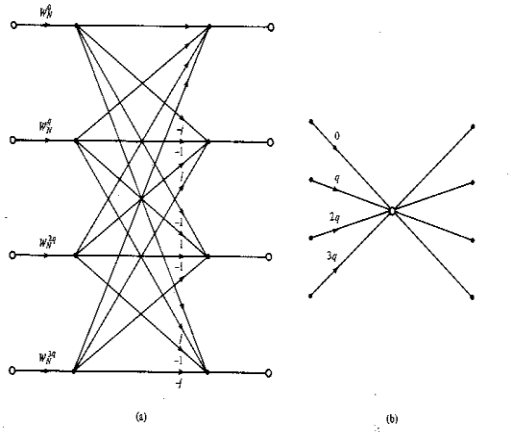


Fig. 1.5 Basic butterfly computation in a radix-4 FFT algorithm.

II.Radix-2 Decimation In Frequency FFT Review Stages

In this section we present several methods for computing the DFT efficiently. 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

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

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

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

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

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.Direct computation of the DFT is basically inefficient primarily because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular, these two properties are :

$$\text{Symmetry property}: W_N^{k+N/2} = -W_N^k$$

$$\text{Periodicit y property}: W_N^{k+N} = W_N^k$$

The computationally efficient algorithms described in this section, known collectively as fast Fourier transform (FFT) algorithms, exploit these two basic properties of the phase factor.

III. Radix-2 FFT Algorithms

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

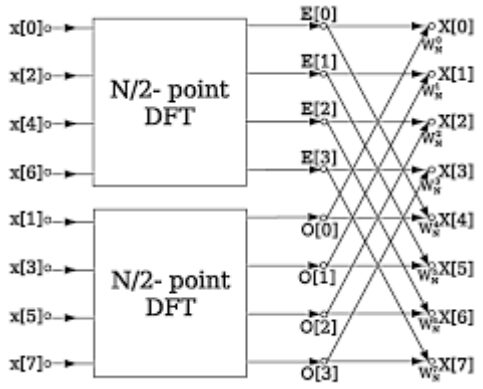
$$f_1(n) = x(2n)$$

$$f_2(n) = x(2n+1), \qquad n = 0,1,\ldots,\frac{N}{2}-1$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating $x(n)$ by a factor of 2, and hence the resulting FFT algorithm is called a *decimation-in-time algorithm.*

Now the N-point DFT can be expressed in terms of the DFT's of the decimated sequences as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots, N-1$$
$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$
$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$



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

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km}$$
$$= F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots, N-1$$

where $F_1(k)$ 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 periodic, with period $N/2$, we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. In addition, the factor $W_N^{k+N/2} = -W_N^k$. Hence the equation may be expressed as

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2}-1$$
$$X(k+\frac{N}{2}) = F_1(k) - W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2}-1$$

We observe that the direct computation of $F_1(k)$ requires $(N/2)^2$ complex multiplications. The same applies to the computation of $F_2(k)$. Furthermore, there are $N/2$ additional complex multiplications required to compute$W_N^k F_2(k)$. Hence the computation of $X(k)$ requires $2(N/2)^2 + N/2 = N^2/2 + N/2$ complex multiplications. This first step results in 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

2. Proposed Methodology:-

2.1 Pipelined FFT Hardware Architectures

The appearance of radix-2^2 was a milestone in the design of pipelined FFT hardware architectures. Later, radix-2^2 was extended to radix-2k . However, radix-2k was only proposed for single-path delay feedback (SDF) architectures, but not for feed forward ones, also called multi-path delay commutator (MDC). By referring some papers they used radix-2k feed forward (MDC) FFT architectures.

Here we use the low power techniques are employed for power consumption using reconfigurable complex multiplier. Using radix-4 algorithm with single-path delay feedback pipelined architecture increase the computational speed, further reduce the chip area by three different processing elements (PE's) were proposed in this radix-4 64-point FFT/IFFT processor.

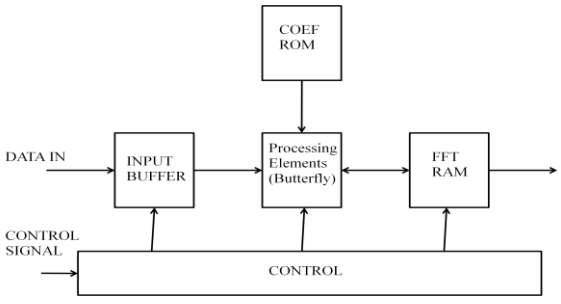


Fig.4.1 Block diagram of FFT processor

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. Fig. 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.

2.2 Low Power Consumption For The FFT

We propose efficient radix-2 pipeline architecture with low power consumption for the FFT/IFFT processor. Our proposed architecture includes a reconfigurable complex constant multiplier and bit-parallel complex multipliers instead of using ROM's to store twiddle factors, which is suited for the power-of-2 radix style of FFT/IFFT processors.

The complex multipliers used in the processor are realized with shift-and-add operations. So the processor uses only a two-input digital multiplier and does not need any ROM for internal storage of coefficients.

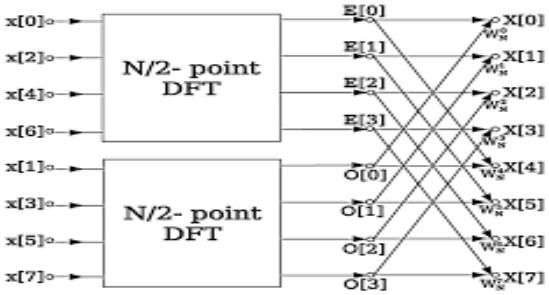


Fig.4.2 N/2 –Point DFT

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

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km}$$
$$= F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,N-1$$

where $F_1(k)$ 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 periodic, with period $N/2$, we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. In addition, the factor $W_N^{k+N/2} = -W_N^k$. Hence the equation may be expressed as

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2}-1$$
$$X(k+\frac{N}{2}) = F_1(k) - W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2}-1$$

2.6 low-power techniques

Low-power techniques are employed to reduce the power consumption in different FFT architectures .The first technique involves the use of a parallel-pipelined architecture at a lower frequency to meet the given throughput .The second technique replaces the complex multiplier with a minimum number of adders and shifters by using both two 's complement and canonical signed-digit (CSD) representations. The third technique proposes a low-power architecture of the commutator (IDR) with a minimal number of memory write operations. These techniques are described in more detail in the following sub-sections.

1. Parallel-Pipelined Architectures

The pipelined FFT is viewed as the leading architecture for real time applications. However, the use of only one processor element (PE) in each stage limits the throughput of pipelined FFTs. Therefore, an increased throughput requires further Based on the preceding equations, the flow graph of parallelization

2. Low-Power Butterfly

In the R4SDC FFT, the butterfly element performs the summations of (2). The conventional butterfly architecture consists of 6 adders/subtractors. In [14], a low-power butterfly architecture was presented. Two 4-input summation blocks were employed to replace six adder/subtractors. However, since the inversions were implemented based on one's complement (and not two's complement), the architecture introduced a small error in the butterfly operations. In this paper, we improve the architecture by eliminating this error. Figure 8 shows the improved low-power butterfly architecture.

2.7 Multiplier less Architecture

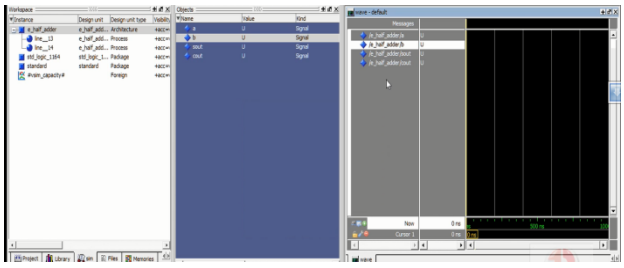
In FFTs, the conventional complex multiplier consists of four real multipliers, one adder and one subtractor. However, since the complex coefficients for all stages can be pre-computed, we can use shift and add operations with common sub expression sharing for those stages that have few

coefficients. In [9], we proposed a multiplier less architecture to substitute the complex multiplier. For example, the number of coefficients used in the second stage of a 64-point FFT or the first stage of a 16-point FFT is 16. These coefficients are shown in Table 1. A close observation of these coefficients reveals that seven of them are (7fff, 0000), and one of them is (0000, 8000).

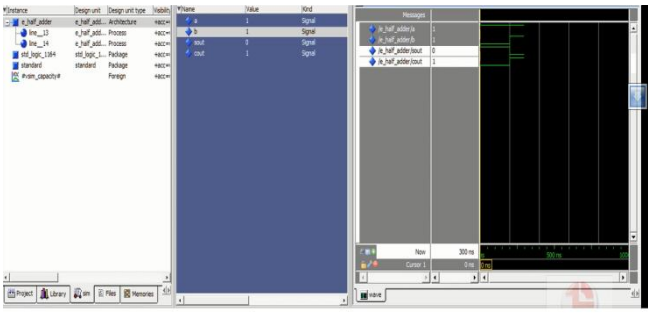
3.Simulation Result

3.1 Study I: Half Adder Simulation

Input**:**

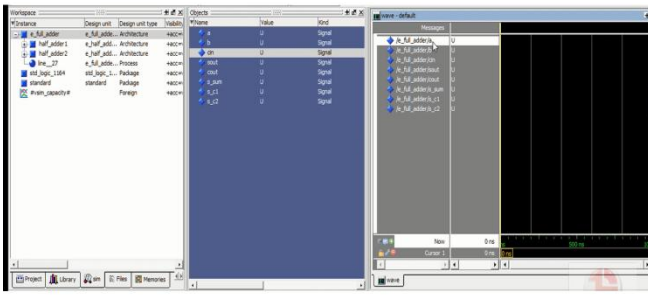


Output:

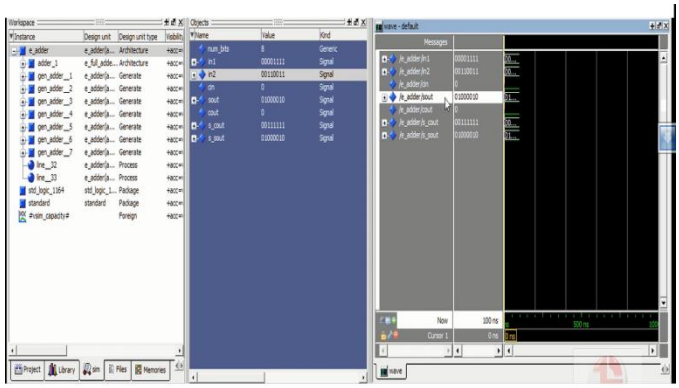


3.2 Study II: Full Adder Simulation

Input**:**



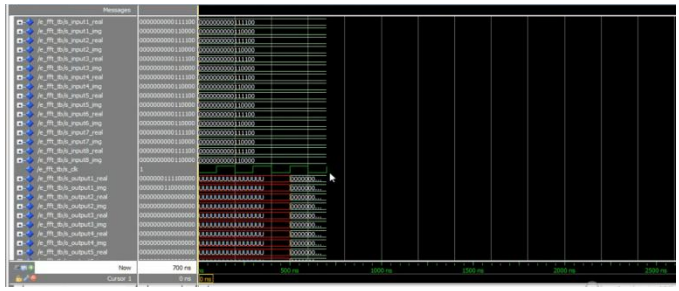
Output:



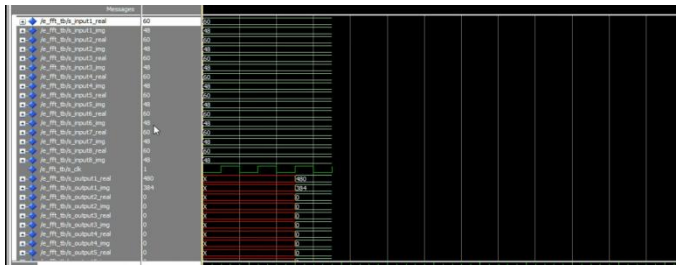
The proposed design has been simulated using Modelsim , the output obtained after simulating is as shown in fig . the simulation is divided in to three parts first is simulation result of half adder ,in half adder there are two input a and b ,put a=1 and b=0, we get the result sout=1 and cout=0.If we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

3.3 Study III: Full fft

Output in three clock cycle: In binary form



Output in three clock cycle: In decimal form



The proposed design has been simulated using Modelsim , the output obtained after simulating is as shown in fig . 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 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.

4. Conclusion

We propose efficient radix-2^2 parallel pipeline architecture with low power consumption for the FFT processor. Our proposed architecture includes a reconfigurable complex constant multiplier and bit-parallel complex multipliers instead of using ROM's to store twiddle factors, which is suited for the power-of-2^2 radix style of FFT processors.Low-power techniques are employed to reduce the power consumption in different FFT architectures .The first technique involves the use of a parallel-pipelined architecture at a lower frequency to meet the given throughput .The second technique replaces the complex multiplier with a minimum number of adders and shifters by using both two 's complement 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 one clock cycle.

REFERENCE

[[1] Mario Galvez, J Grajal, M A. Sanchez and Oscar Gusta fsson."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" Department of Electrical and Computer Engineering ,University of Minnesota, Mimmeapolis, Mn, Usa.
[3] 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.
[4] V. Boriakoff, "FFT computation with systolic arrays, a new architecture," IEEE Trans. Circuits Syst. II, vol. 41, Apr. 1994, pp. 278–284.
[5] 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 [6] 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
[10] Fahad Qureshi and Oscar Gustafson,"Twiddle Factor Memory Switching Activity Analysis of Radix-2^2 and Equivalent FFT Algorithms", IEEE Proceedings, April 2010.
[11] Jianing Su, Zhenghao Lu, "Low cost VLSI design of a flexible FFT processor", IEEE Proceedings, April 2010.
[12] He Jing, Ma Lanjaun, Xu Xinyu, "A Configurable FFT Processor", IEEE proceeding 2010.
[13] M.Merlyn, "FPGA Implementation of FFT Processor with OFDM Transceiver", 2010 IEEE proceeding.
[13] Nuo Li and N.P.van der Meijs, "A Radix based Parallel pipelined FFT processor for MB- OFDM UWB system," IEEE Proceedings, 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] Yuan Chen, Yu-Wei Lin, "A Slock scalin FFT/IFFT processor for WiMAX Applications" IEEE proceedings 2006.
[23] 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.
[24] Jen-Chuan Chi and Sau-Gee Chen" An Efficient Fft Twiddle Factor Generator"National Chiao Tung University Department of Electronics Engineering and Institute of Electronics 1001 Ta Hsueh Rd, Hsinchu, Taiwan, ROC.