A Modified RADIX-4 Architecture to Improve The Speed Of Digital Signal Processor Using VLSI Implementation

[1]S. Gokul, [2]V. Kishorekumar, [3]M. Bennish
[1]Student, [2]Student, [3]Student
[1]Department of ECE,
[1]Panimalar engineering college, Chennai, India

Abstract: *This* paper presents a modified Radix - 4 architecture to compute Discrete Fourier Transform (DFT) using FFT architecture to reduce the computation time and power thereby increase the speed of the Digital Signal processor. The proposed design reduces the number of multipliers and adders that are used to compute large sequence of real time data signals with less power consumption. This is achieved by avoiding repeated multiplication of twiddle factor (real and complex values) which makes the delay in the computation of FFT. The twiddle factor value is already calculated with the number of input bits to be used and the value is stored in the register. The value is called whenever the FFT requires twiddle factor for computation. The appropriate twiddle factor is selected using 3 x 8 decoder. The modified Radix-4 FFT architecture is designed, implemented and simulated using the software Xilinx - Spartan 3E version 8.2.

***IndexTerms* - FFT, SDC, DFT, DIT, DIF.**

I. Introduction

FFT mostly included important part in gadgets and communication field. The different kinds of FFT radix calculation have broke down and is to be adjusted in future. As of it's necessities, it possesses substantial territory and has high power utilization if executed in equipment. Effective calculations been created to enhance it's design. For investigation, control utilization, memory prerequisite, equipment and all through of every calculation must be seen.
FFT(Fast Fourier Transform) is a usually utilized method for the calculation of Discrete Fourier Transform(DFT).DFT calculations are required in the fields like filtering, spectral analysis and so forth, to compute the recurrence range. However, coordinate calculation of Discrete Fourier Transform(DFT) requires on the request of N^2 tasks where N is the change measure. FFT is used in digital video broadcasting and OFDM systems[1]. The FFT algorithm, first explained by Cooley and Tukey, opened a new area in digital signal processing by reducing the order of complexity of DFT from N^2 to N$\log_2 N$[1]. There are number of various Fast Fourier Transform calculations that empowered the estimations considerably quicker than DFT. FFT calculations utilized for count of Discrete Fourier Transform of an information vector. FFT is utilized to accelerate the DFT, it diminishes the calculation time required to compute a Discrete Fourier Transform and enhances the execution by factor 100 or in addition coordinate evaluation of DFT.
In Cooley-Tukey radix 2 algorithm, the N point DFT is subdivided into two(N/2) point DFTs and then (N/2) point DFT is recursively divided into smaller DFTs until a two point DFT[1], whose butterfly is just an addition and subtraction of input complex numbers[1]. It is the best reasonable calculation for number N, which is an energy of 2. Higher radix calculations, for example, radix 4, radix 8, etc., can be utilized to decrease the complex multiplication however the butterfly structure winds up noticeably complex with numerous info complex adders.
As of late, there has been some examination in the outline of multipath pipelined FFT design that gives a high throughput. Numerous FFT processor structures are acquainted all together with use the OFDM design, for example, single path delay commutator (SDC), multipath delay commutator(MDC), single path delay feedback(SDF), multipath delay feedback(MDF) among the different FFT models ,the MDF models is oftentimes utilized as an answer for give a throughput rate of more than 1GS/s. With a specific end goal to diminish the zone and power utilization, a few FFT calculations and dynamic scaling plans have been proposed.

II. Existing Algorithms

A. Radix 2 FFT Algorithm

Fast Fourier Transform is an efficient method for computing Discrete Fourier Transform(DFT)[1]. The DFT of a signal time domain signal x(n) is given in equation

$$X(k)=\sum_{n=0}^{N-1} x(n)W_N^{Kn}, k=0,1,2,\ldots N-1$$

The 4-point transform can be diminished to two 2-point transforms: one for even elements, one for odd elements the odd one will be multiplied by W_{4k}[1].Diagrammatically ,this can be spoken as two levels of butterflies.
The Fig.1 number demonstrates that 4 point radix 2 FFT butterfly chart to process the DFT focuses. The radix 2 calculations are the least complex FFT calculations. The decimation in time (DIT) radix 2 FFT recursively segments a DFT into two half-length DFTs of the even ordered and odd ordered [1]. The yields of these shorter FFTs are reused to process numerous yields therefore extraordinarily reducing the aggregate computational cost. The radix 2 annihilation in-time and destruction in-recurrence Fast Fourier Transforms(FFTs) are the least complex FFT calculations. Like all FFTs they pick up their speed by reusing the consequences of smaller, intermediate computations to compute different DFT recurrence yields.

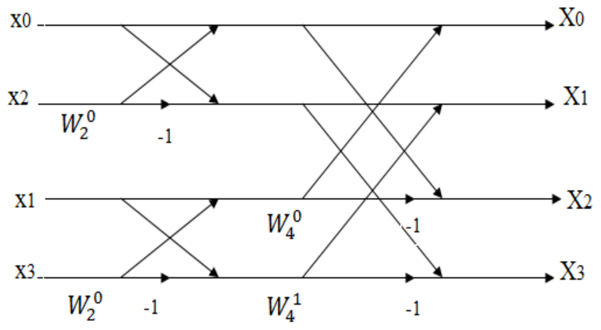


Fig.1 Existing RADIX 2 Flow Diagram

B. Radix 4 FFT Algorithms

Radix-4 is an another FFT calculation which was studied to enhance the speed of working by diminishing the calculations, this can be acquired by change the base to 4. For a same number if base builds the power/record will diminishes. For radix 4 the quantity of stages are decreased to half since N=43(N=4M) i.e just 3 phases .Radix 4 is having 4 information sources and 4 yields and it follows set up calculation. The radix 4 DIT-FFT recursively partitions a DFT into four quarter - length DFTs of gatherings of each fourth time sample[2]. The output of these shorter FFTs require just 75% the same number of complex augmentations as the radix 2 FFTs [1] .The radix - 4 decimation - in-time and decimation in-recurrence Fast Fourier Transform (FFT) pick up their speed by reusing the result of smaller, intermediate of the road calculations to register different DFT recurrence yields.

In radix 4 FFT algorithms N point input signal are decompose into following equation. The equations are derived in following manner

X(4K) = $\sum_{n=0}^{\frac{N}{4}-1}[x(n) + x\left(n+\frac{N}{4}\right) + x(n+N/2) + x(n+\frac{3N}{4})]W_{N/4}^{Kn}$ --------------------------------(1)

X(4K+1) = $\sum_{n=0}^{\frac{N}{4}-1}[x(n) - jx\left(n+\frac{N}{4}\right) - x(n+N/2) + jx(n+\frac{3N}{4})]W_{N/4}^{Kn}W_N^{n}$--------------------------------(2)

X(4K+2) = $\sum_{n=0}^{\frac{N}{4}-1}[x(n) - x\left(n+\frac{N}{4}\right) + x(n+N/2) - x(n+\frac{3N}{4})]W_{N/4}^{Kn}W_N^{2n}$--------------------------------(3)

X(4K+3) = $\sum_{n=0}^{\frac{N}{4}-1}[x(n) + jx\left(n+\frac{N}{4}\right) - x(n+N/2) - j\,x(n+\frac{3N}{4})]W_{N/4}^{kn}W_N^{3n}$--------------------------------(4)

Where k=0,1,……N/4-1 and used the property W_N^{4kn}=$W_{N/4}^{kn}$,X(4k),X(4k+1),X(4k+2) and X(4k+3) are N/4-point DFTs.Each N/4 yield is an aggregate of four information tests all duplicated by - 1,j,or 1,- j. Twiddle factors are increased by above whole .Every N/4 sample DFTs is separated into N/16 sample DFTs .Every N/16 DFT is circulated encourage in four N/64 point et cetera.
These are the statements of radix 4 FFT calculations .The radix 4 butterfly contains 3 complex augmentations and 12 complex increments N/4 butterfly includes in each stage and number of stage is $\log_4 N$ for N point grouping .Therefore ,the quantity of complex duplications is 3N/4$\log_4 N$and number of complex increases is12N/$\log_4 N$.In correlation of radix 2 FFT number of complex duplications are lessened by 25% however number of complex augmentations are expanded by 50%.
Essentially, this Fast Fourier Transform calculation utilize separate and overcome approach technique to partition the calculation recursively and remove however many regular twiddle factors as possible. The quantity of required genuine increases and augmentations is normally used to analyse the effectiveness of various FFT calculations regarding multiplicative correlation, split radix FFT is computationally better to the various calculations since it has most paltry duplications. In the end, this calculation has a disadvantage in view of unpredictable structure that leads this calculation not appropriate for execution of computerized signal processors. Auxiliary normality is likewise critical in usage of FFT calculations on committed chips, for example, ASIC(Application Specific Integrated Chip).Hence radix 2 and radix 4 FFT calculations are best as far as speed and precision.

III. Proposed Radix-4 Algorithm

The radix 4 16-point FFT composed utilizing Verilog code and reproduced in Verilog keeping in mind the end goal to check it's usefulness. The outline is combined utilizing 0.18µm innovation gave by Artisan library. Timing imperative is set with working recurrence 50MHz.
FFT architecture is divided into three main process blocks. The block diagram of process block is shown in the below Fig.2 [3].

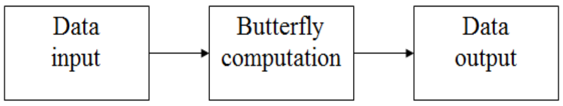


Fig.2 RADIX-4 Block diagram

This block comprise of information input, butterfly calculation and information yield. The information is perused in each rising edge of check and put away in the memory enlist. Butterfly calculation piece register the put away information before going to yield process. The information is kept in the enrol before it is perused out.

3.1Butterfly Architecture

The Fig.3 shows the Radix-4 butterfly architecture. The most vital component in FFT processor is a butterfly structure [4]. It takes two marked settled point information from memory register and figures the FFT calculation. The yield comes about are composed back in same memory area as the previous input stored. This strategy is brought in arrangement memory storage whereby it can diminish the equipment use. The butterfly design is appeared in the below figure. The adder sums the contribution before being increased by the twiddle factor [5]. The multiplier frames the incomplete result of the complex multiplication and deliver 2 times greater than the input bit. Shift register would move the bits to maintain a strategic distance from the overflow issue [6]. Output of this butterfly would be kept in the register for the subsequent range [4].

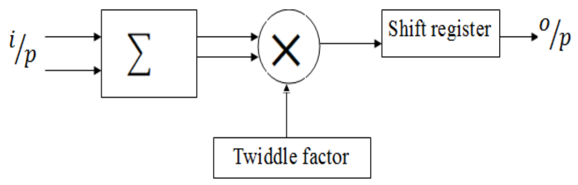


Fig.3 Radix-4 Butterfly Architecture

3.2 Controller

The proposed Radix-4 controller circuit is given in Fig.4. The FFT processor event is controlled by the control circuit depending upon the feedback it gets from the surrounding unit [5].

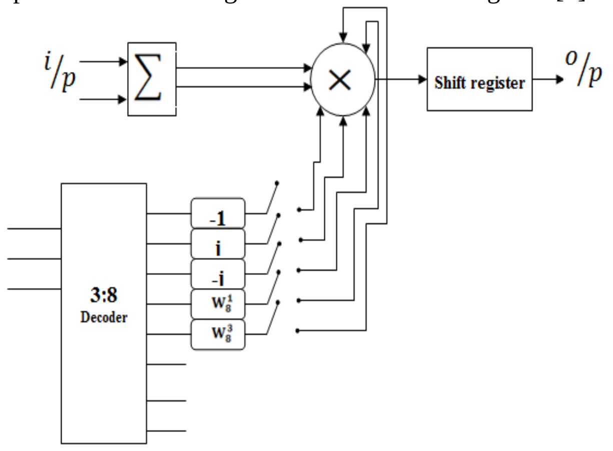


Fig.4 Proposed RADIX-4 Controller

Moore machine approach is adapted whereby the output signal dependent to the value of the next state. This outline capacities as a synchronous design which controlled by "CLK" flag. The information flag "RST" is utilized to reset the FFT processor including the input buffer which holds information for next stage. Input "EN" signal is utilized to control the state change of the processor. Flag "SYNC_OUT" would be empower when the yield flag is created.

The Proposed RADIX-4 Equations as follows

X(K)=$\sum_{n=0}^{N-1} x(n)W_N^{Kn}$ --------------------------------(5)

Let us take N=8 point

$=\sum_{n=0}^{\frac{N}{4}-1} x(n)W_N^{Kn}+\sum_{n=N/4}^{\frac{N}{2}-1} x(n)W_N^{Kn}+\sum_{n=N/2}^{\frac{3N}{4}-1} x(n)W_N^{Kn}+\sum_{n=3N/4}^{N-1} x(n)W_N^{Kn}$--------------------------------(6)

The value of the twiddle factor for the algorithm is found

1.$W_N^{KN/4}=e^{-j(2\pi/N)x(\frac{KN}{4})}=(-j)^K$

2.$W_N^{KN/2}=e^{-j(2\pi/N)x(\frac{KN}{2})}=(-1)^K$

3.$W_N^{3KN/4}=e^{-j(2\pi/N)x(\frac{3KN}{4})}=(j)^K$

The 'K' sequence is divided into X(4K), X(4K+1), X(4K+2), X(4K+3) where k=0,1

Case 1: let K=4K , Therefore

X(4K) = $\sum_{n=0}^{\frac{N}{4}-1}[x(n)+x\left(n+\frac{N}{4}\right)(-j)^{4K}+x(n+N/2)(-1)^{4K}+x(n+\frac{3N}{4})(j)^{4K}]W_N^{4Kn}$ ---------------(7)

Substitute n=0,1 and N=8 in equation (7), then

X(4K)={[x(0)+x(2)+x(4)+x(6)]W_8^0}+{[x(1)+x(3)+x(5)+ x(7)]W_8^{4K}} --------------------------------(8)

Then substitute k=0,1 in equation 5 therefore we get two values of X(0) and X(1).

X(0)={x(0)+x(2)+x(4)+x(6)}+{x(1)+x(3)+x(5)+x(7)}W_8^0 --------------------------------(9)

X(1)={x(0)+x(2)+x(4)+x(6)}+{x(1)+x(3)+x(5)+x(7)}W_8^4 -----------------(10)

Now substitute the twiddle factor value in equation 9 and 10.

$W_8^4=e^{-j(\frac{2\pi}{8})x(4)}$=-1

Then we get

X(0)={x(0)+x(2)+x(4)+x(6)}+{x(1)+x(3)+x(5)+x(7)} ---(11)

X(4)={x(0)+x(2)+x(4)+x(6)}+(1){x(1)+x(3)+x(5)+x(7)} ---(12)

Case 2: let K=4K+1

X(4K+1)= $\sum_{n=0}^{\frac{N}{4}-1}[x(n)+x\left(n+\frac{N}{4}\right)(-j)^{4K+1}+x(n+N/2)(-1)^{4K+1}+x(n+\frac{3N}{4})(j)^{4K+1}]W_N^{(4K+1)n}$---------(13)

Substitute n=0,1 and N=8 in equation 13

Then

X(4K+1)={[x(0)+$(-j)^{4K}(-j)^1$x(2)+$(-1)^{4K}(-1)^1$x(4)+$(j)^{4K}(j)^1$x(6)]W_8^0}+{[x(1)+$(-j)^{4K}(-j)^1$x(3)+$(-1)^{4K}(-1)^1$x(5)+$(j)^{4K}(j)^1$x(7)] W_8^{4K+1}}

X(4K+1)={[x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)]W_8^0}
+{[x(1)+$(-j)^1$x(3)+$(-1)^1$x(5)+$(j)^1$x(7)]W_8^{4K+1}}
-----------------(14)

Then substitute k=0,1 in equation11 therefore we get two values X(1) and X(5)

X(1)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}+{x(1)+$(-j)^1$
x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}W_8^1 -----------------(15)

X(5)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}+{x(1)+$(-j)^1$
x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}$W_8^4 W_8^1$-----------------(15)

Now substitute the twiddle factor value in equation 14and 15.

W_8^4=$e^{-j\left(\frac{2\pi}{8}\right)x(4)}$=-1

Then we get

X(1)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}+{x(1)+
$(-j)^1$x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}W_8^1 -----------------(16)

X(5)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}-
{x(1)+$(-j)^1$x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}W_8^1 ---------(17)

Case 3: let K=4K+2.

X(4K+2)= $\sum_{n=0}^{\frac{N}{4}-1}[x(n)+x\left(n+\frac{N}{4}\right)(-j)^{4K+2}+x(n+\ N/2)(-1)^{4K+2}+\ x(n+\frac{3N}{4})(j)^{4K+2}]W_N^{(4K+2)n}$-------(18)

Substitute n=0,1 and N=8 in equation18

X(4K+2)={[x(0)+$(-j)^{4K}(-j)^2$x(2)+$(-1)^{4K}(-1)^2$x(4)+$(j)^{4K}(j)^2$x(6)]W_8^0}+{[x(1)+$(-j)^{4K}(-j)^2$x(3)+$(-1)^{4K}(-1)^2$x(5)+$(j)^{4K}(j)^2$x(7)] W_8^{4K+2}}

X(4K+2)={[x(0)+$(-j)^2$x(2)+$(-1)^2$x(4)+$(j)^2$x(6)]W_8^0}+{[x(1)+$(-j)^2$x(3)+$(-1)^2$x(5)+$(j)^2$x(7)]W_8^{4K+2}} --(19)

Then substitute k=0,1 in equation 18then we get two values of X(2) and X(6)

X(2)={x(0)+x(2)+x(4)-x(6)}+{x(1)+x(3)+x(5)-x(7)}W_8^2
---------(20)

X(6)={x(0)+x(2)+x(4)-x(6)}+{x(1)+x(3)+x(5)
-x(7)}$W_8^4 W_8^2$ -------(21)

Now substitute the twiddle factor values in equation 20 and 21

W_8^4=$e^{-j\left(\frac{2\pi}{8}\right)x(4)}$=-1 , W_8^2=$e^{-j\left(\frac{2\pi}{8}\right)x(2)}$=1

Therefore

X(2)={x(0)+x(2)+x(4)-x(6)}+{x(1)+x(3)+x(5)-x(7)}
---------------------------(22)

X(6)={x(0)+x(2)+x(4)-x(6)}+(-1){x(1)+x(3)+x(5)-x(7)}
------------------------(23)

The sequences are:

X(0)={x(0)+x(2)+x(4)+x(6)}+{x(1)+x(3)+x(5)+x(7)}
X(4)={x(0)+x(2)+x(4)+x(6)}+(1){x(1)+x(3)+x(5)+x(7)}
X(1)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}+ {x(1)+$(-j)^1$x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}W_8^1
X(5)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(j)^1$x(6)}- {x(1)+$(-j)^1$x(3)+$(-1)^1$x(5)+$(j)^1$x(7)}W_8^1
X(2)={x(0)+x(2)+x(4)-x(6)}+{x(1)+x(3)+x(5)-x(7)}
X(6)={x(0)+x(2)+x(4)-x(6)}+(-1){x(1)+x(3)+x(5)-x(7)}
X(3)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(-j)^1$x(6)}+{x(1)$(-j)^1$x(3)+$(-1)^1$x(5)+$(-j)^1$x(7)}W_8^3
X(7)={x(0)+$(-j)^1$x(2)+$(-1)^1$x(4)+$(-j)^1$x(6)}- {x(1)+$(-j)^1$x(3)+$(-1)^1$x(5)+$(-j)^1$x(7)}W_8^3

I. Proposed RADIX-4 Architecture of FFT

The Fig.5 shows the proposed modified Radix-4 architecture of FFT. The X(0) term obtained by the addition of the terms A and B ,where A and B are represented as adder elements. The adder element A is represented by the sum of the data arrays x(0), x(2),x(4),x(6) and the adder element B is represented by the sum of the data arrays x(1), x(3),x(5),x(7).The X(4) term obtained by the subtraction of the terms A and B, where A and B are represented as adder elements. The adder element A is represented by the sum of the data arrays x(0), x(2),x(4),x(6) and the adder element B is represented by the sum of the data arrays x(1), x(3),x(5),x(7).The ben X(1) term is obtained by the addition of C and the term D multiplied by the twiddle factor W_n^1,where C and D are adder elements. ”C” is obtained by the summing of x(0),x(2) multiplied by “-j”, x(4) multiplied by “-1” and x(6) multiplied by “j” ,whereas the term “D” is obtained by the addition of x(1),x(3)multiplied by “-j”, x(5) multiplied by “-1”,x(7) multiplied by “j”. The X(5) term is obtained by the subtraction of C and the term D multiplied by the twiddle factor W_n^1,where C and D are adder elements.”C” is obtained by the addition of x(0),x(2) multiplied by “-j”,x(4) multiplied by “-1” and x(6) multiplied by “j” ,whereas the term “D” is obtained by the addition of x(1),x(3)multiplied by “-j”,

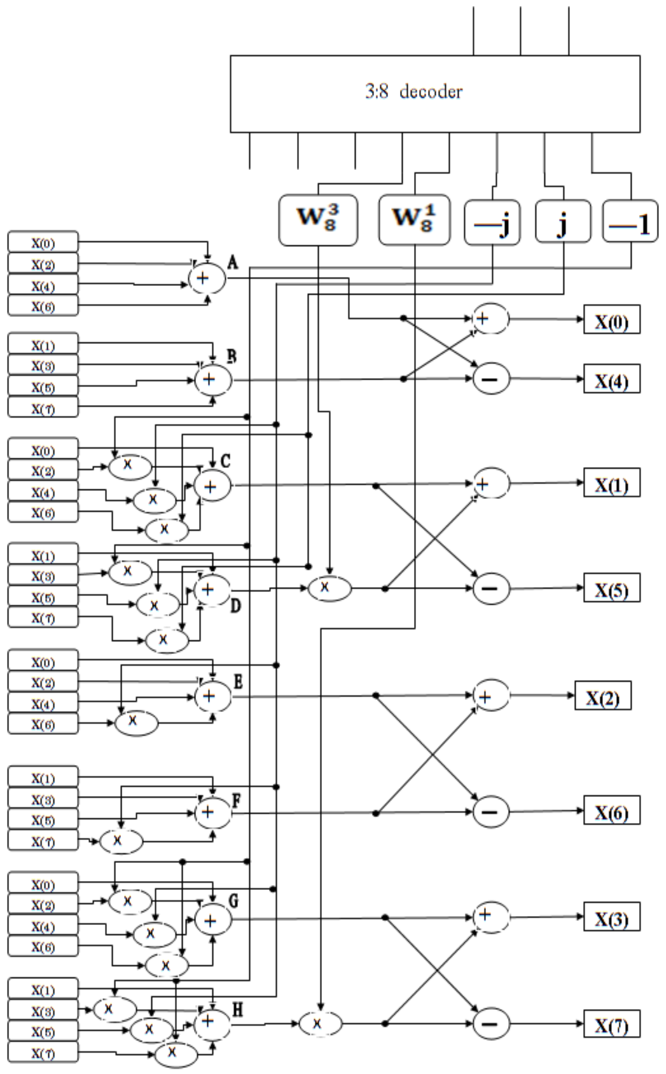


Fig.5 Modified Radix-4 Architecture of FFT

x(5) multiplied by "-1",x(7) multiplied by "j". The term X(2)is obtained by the addition of adder elements E and F, where E is obtained by the addition of the data arrays x(0),x(2),x(4) and x(6) multiplied by "-1" and the element F is obtained by the addition of x(1),x(3),x(5) and x(7) multiplied by"-1".The term X(6) is obtained by the subtraction of adder elements E and F, where E is obtained by the addition of the data arrays x(0),x(2),x(4) and x(6) multiplied by "-1" and the element F is obtained by the addition of x(1),x(3),x(5) and x(7) multiplied by"-1".The term X(3) is obtained by the addition of adder element G and the adder element H multiplied by twiddle factor W_n^3.The adder element G is obtained by the addition of x(0),x(2) multiplied by "-j",x(4) multiplied by "-1",x(6) multiplied by "-j".On the otherhand,the adder element H is obtained by the addition of x(1),x(3) multiplied by "-j", x(5) multiplied by"-1",x(7) multiplied by "-j". The term X(7) is obtained by the subtraction of adder element G and the adder element H multiplied by twiddle factor W_n^3.The adder element G is obtained by the addition of x(0),x(2) multiplied by "-j", x(4) multiplied by "-1",x(6) multiplied by "-j". On the other hand, the adder element H is obtained by the addition of x(1), x(3) multiplied by "-j", x(5) multiplied by"-1",x(7) multiplied by "-j".

3.1Proposed RADIX-4 Output

The Fig.6 shows the RTL view of Modified Radix-4 FFT.

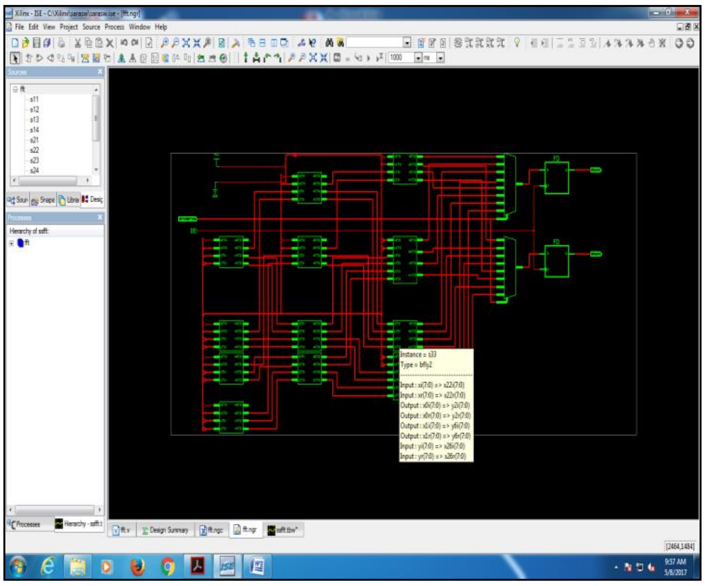

Instance = s33
Type = bfly2
Input : xi(7:0) => x22i(7:0)
Input : xr(7:0) => x22r(7:0)
Output : x0i(7:0) => y2i(7:0)
Output : x0r(7:0) => y2r(7:0)
Output : x1i(7:0) => y6i(7:0)
Output : x1r(7:0) => y6r(7:0)
Input : yi(7:0) => x26i(7:0)
Input : yr(7:0) => x26r(7:0)

Fig.6 RTL view of Modified Radix-4 FFT

The Fig.7 shows the Output of Modified Radix-4 FFT

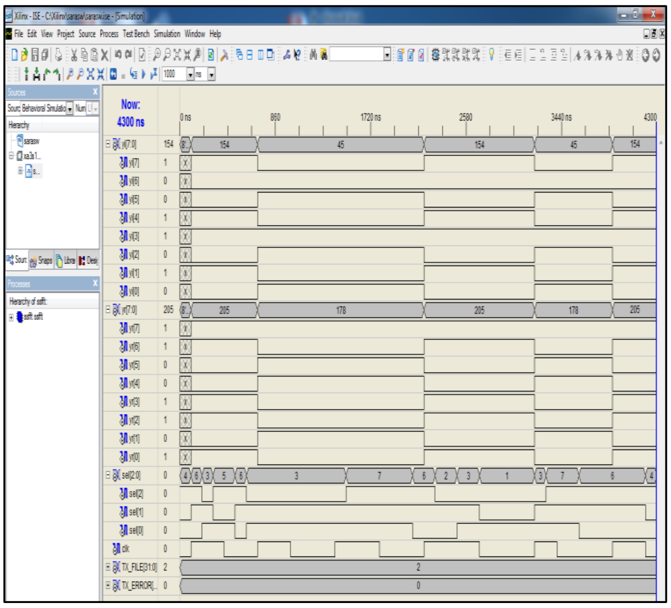
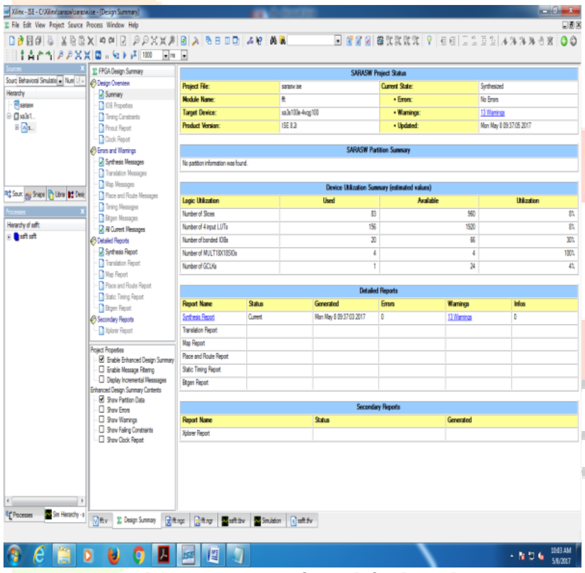


Fig.7 Output of Modified Radix-4 FFT

The Fig.8 shows the Synthesis Report of Modified Radix-4 FFT.



SARASW Project Status

Project File:	sarasw.ise	Current State:	Synthesized
Module Name:	fft	• Errors:	No Errors
Target Device:	xa3s100e-4vqg100	• Warnings:	13 Warnings
Product Version:	ISE 8.2	• Updated:	Mon May 8 09:37:05 2017

SARASW Partition Summary

No partition information was found.

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slices	83	960	8%
Number of 4 input LUTs	156	1920	8%
Number of bonded IOBs	20	66	30%
Number of MULT18X18SIOs	4	4	100%
Number of GCLKs	1	24	4%

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Mon May 8 09:37:03 2017	0	13 Warnings	0
Translation Report					
Map Report					
Place and Route Report					
Static Timing Report					
Bitgen Report					

Secondary Reports

Report Name	Status	Generated
Xplorer Report		

Fig.8 Synthesis Report of Modified Radix-4 FFT

IV. RESULTS AND DISCUSSION

4.1 Results of Descriptive Statics of Study Variables

S.NO	EXISTING ALGORITHM	PROPOSED ALGORITHM
ADDERS	16	12
MULTIPLIERS	20	12

So it is concluded that adders and multipliers is reduced in proposed RADIX-4 algorithm by the use of registers.

REFERENCES

[1] K. Jayaram, C. Arun, "Survey Report for Radix 2, Radix 4, Radix 8 FFT Algorithms",IEEE Vol. 4, Issue 7,July 2015
[2] Priyanka Pateriya, Ankeet Chaora, "Design and Implementation of Real-Time 16-bit Fast Fourier Transform using Numerically Controlled Oscillator and Radix-4 DITFFT Algorithm in VHDL", IRJET Volume: 03 Issue: 05 | May-2016.
[3] Palaniappan,SivaKumarZulkifli, Tun Zainal Azni "Design of 16-point radix-4 Fast Fourier Transform in 0.18[micro]m CMOS technology" ISSN**:** 1546-9239, Date: August, 2007 Source Volume: 4 Source Issue: 8
[4] Anjana R, Krunal Gandhi, Vaishali lad, "Low Power R4SDC Pipelined FFT Processor Architecture" , IOSR-JVSP ,e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197 Volume 1, Issue 6 (Mar. – Apr. 2013), PP 68-75
[5] Sriram kumaran, "project report FFT1" https://www.scribd.com/document/53146191/Project-report-FFT1.
[6] Sumit anand,vipin gupta,"Advanced Shift RegisterDesign using PSDRM Reversible Logic", volume:3, issue:12 https://www.slideshare.net/editorijritcc1/advanced-shift-register-design-using-psdrm-reversible-logic .