ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, *January 2015*

Synthesis and analysis of Fourier and format number transform operator on FPGA

Dr.Godbole B. B[1], Miss.PanditMadhuri D[2]

Associate Professor of Department of Electronic KBPCEP., SataraShivaji University Kolhapur, Maharashtra,India[1]

Student of Department of Electronic KBPCEP., SataraShivaji University Kolhapur, Maharashtra, India [2]

Abstract: Reconfiguration is an essential part of Software Radio (SR) technology. In this SR context, the Fast Fourier Transform (FFT) operator is defined as a common operator for many classical telecommunications operations. The systems are designed with a new architecture for this operator that makes it a device intended to perform two different transforms. The first one is the Fast Fourier Transform (FFT) used for the operations in OFDM based communication. The second one is the Fermat Number Transform (FNT) used for the finite operations in the Galois Field (GF). This operator can be reconfigured to switch from an operator dedicated to compute the FFT to the FNT in the Galois Field.

Keywords: Software Rodio, FPGA, Reconfigurable operator, FFT,FNT.

I INTRODUCTION

Over the past years, a proliferation of communication standards has substantially increased the complexity of radio designs, the communication standards are implemented separately using dedicated instantiations which are difficult to upgrade for the support of new features. So the concept software radio (SWR)[1][2] introduced by J. Mitola.Software Radio basically refers to an ensemble of techniques which permits the reconfiguration of a communication system without the need to change any hardware system element. This reconfiguration implies the optimization of the hardware-software resources in the terminal architecture design[3]. So this optimization is helpful in a new area of research& DEVELOPMENT.In this optics, it present a new architecture for the FFT whose Butterfly is reconfigurable so as to perform two kinds of transforms over two different fields. The first one is the FFT carry out some function of OFDM operation. The second one in the GF where the FFT will be reconfiguredas an FNT.

II FFT AND FNT

The Fast Fourier Transform (FFT) is a major block in DSP and digital Communication system. The FFT is a faster version of the Discrete Fourier Transform (DFT) and efficiently calculates Discrete Fourier Transform. The DFT is one of the specific forms of Fourier analysis. FFT is also used in various application such as image processing, optical communication, radar, audio and video broadcasting. Thus, due to the huge application of FFT, there is need of efficient FFT which should occupies less area and high speed. FFT is an extremely robust algorithm that lends itself well to machine computation and is efficiently applied in wireless communication. The invention of FFT is attributed to Cooley and Tukey in 1965. FFT compute the DFT with greatly reduced number of operations. FFT and its inverse play a significant role in many DSP applications. FFT algorithm started a new era in DSP by reducing the order of complexity of DFT from N^2 to $N\times\log_2 N$ reduces the number of required complex multiplication compared to the normal DFT.

The Number Theoretic Transform (NTT) has been introduced as a generalization of the Discrete Fourier Transform (DFT)[4][5] over residue class rings of integers. Interesting applications of the NTT lies in fast coding, decoding, long integer multiplication, cryptography, digital filtering, image processing and deconvolution[6]. In this ring, number theoretic transforms similar to the DFT, the cyclic convolution can be performed very efficiently and without any round-off errors. In modulo arithmetic, all basic operations such as addition, subtraction, and multiplication are carried out modulo an integer M.

The NTT can be represented as the following equations:

$$X(k) = \sum_{n=0}^{N-1} x(n)\alpha^{nk} \ mod\ M \quad , \qquad 0 \le k \le N-1$$

Where α is primitive element of field. A primitive element of the field $GF(q)$ is an element such that every field element except zero can be expressed as a power of α.When we replace the above M of Number Theoretic transform by Fermat Number F_t, where $F_t = 2^{2^t} + 1$are primes and only when t=0~4 this $2^{2^t} + 1$is a positive integer, the transform is called Fermatnumber transform.

For transform length equal to $F_t$where $F_t = 2^{2^t} + 1$is the Fermat number, the NTTis called the Fermat Number Transform (FNT) whichpresents some advantages. It is quite obvious, that FNTis suitable for VLSI implementations. The structure ofthe FNT is identical to that of the DFT for power oftwo lengths. Then the same algorithms can be used for the classical radix-2 FFT and the radix-2 FNT. Theonly one difference is the substitution of the complexmultiplication in the Fourier transform by a modulo $F_t$real multiplication in the case of the FNT. The following gives the definitions of FFT and FNT.

Fourier transform theory over complex field as well as finitefield is given as, In the complex field (**C**), the

Discrete Fourier Transform of f_n= (f_0, f_1, ...,f_{N-1}), a vector of real or complex numbers, is a vector F= (F_0, F_1, ..., F_{N-1}), given by[8][10],

$$F_k = \sum_{n=0}^{N-1} f_n W_N^{kn} \qquad k = 0 \ldots\ldots, N-1$$

Where,W_N= exp(−2jπ/N)and j = −1 .W_N^{kn} is referred as the twiddle factor. The Fourier kernel exp(−2jπ/N) is an N^{th} root of unity in the field **C**. In the finite field GF(q), an element α of order N is an N^{th} root of unity. Drawing on the analogy between exp(-2jπ/N) and α, Fourier transform over finite field can be defined as follows let f=(f_0, f_1, ..., f_{N-1}) be a vector over $GF(q)$, and let α be an element of $GF(q)$ of order N. The Fouriertransform of vector f is the vector F=(F_0, F_1, ..., F_{N-1}) whose components are given by[3][8],

$$F_j = \sum_{n=0}^{N-1} f_i \alpha^{ij} \qquad j = 0, \ldots .\mathrm{N}-1.$$

Vector f is related to its spectrum F by,

$$f_j = \frac{1}{N} \sum_{n=0}^{N-1} F_j \alpha^{ij} \qquad i = 0, \ldots .\mathrm{N}-1.$$

The application of the Discrete Fourier Transform in the complex field occurs throughout the subject of signal processing. The same transform technique can play an important role in the study and processing of GF(q) valued signals, q a prime number. It describes in details the practical realization of the FFT [7] defined in complex field and which can be reconfigured to become the FNT operator with arithmetic carried out modulo Fermat numbers. This reconfiguration consists in reconfiguring each Butterfly of the FFT structure. In the next section it will present the Butterfly itself as a function which is constituted by several reconfigurable arithmetic operators.

III RECONFIGURABLE BUTTERFLY

In the SWR concept, new area of research called "Parametrization" has been defined [8][9]. This technique consists to identify common resources, i.e CommonOperator (CO) or Common Function (CF) between all the standards involved in the reconfiguration and in the standards themselves. Then, the trick is to exploit the same resources to execute two or more applications.[8]. In this context, the main goal of this work is to exploit the resources already present in the FFT structure to get the FNT one. With this purpose, the arithmetic operators i.e multiplier, adder and subtracter realizing operations over C should be redefined to realize a modulo(F_t) operations[9]. Then the reconfiguration of the Butterfly, consists to reconfigure the aforementioned arithmeticelements. Then, one need to define a modulo(F_t) multiplier, adder and subtracter.

A. Modular Multiplication in GF (F_t)

The modulo $2^n + 1$ multiplication is widely used in the computation of convolutions and in Residue Number Systems (RNS) arithmetic.Several architectures of modulo $(2^n + 1)$ multiplier based on Ma's algorithm [10].Indeed, there are two categories of algorithms for the modulo $(2^n + 1)$multiplication. The first one consists to perform the multiplication and after the correction [11]. The second one consists in the reduction of partial product. This modular multiplier can implemented inSparten3,Virtex-II.

In Figure 1, the white elements indicate the elements not used in the case of operation over GF. As it is noticed in this figure, there is a reconfiguration of the connections inter-operators. The dotted lines connections represent the additional connections in this operating mode over GF. A basic modulo $(2^n + 1)$ multiplication algorithm consists in computing p=xy, and dividing this product by $2^n + 1$:

$$xy \bmod (2^n + 1) = p \bmod (2^n + 1)$$

Here C_L and C_H the lower and higher words respectivelyof the product p asfollows:

$$C_L = \sum_{n=0}^{N-1} p_i 2^i \qquad C_H = \sum_{n=0}^{N-1} p_i 2^i$$

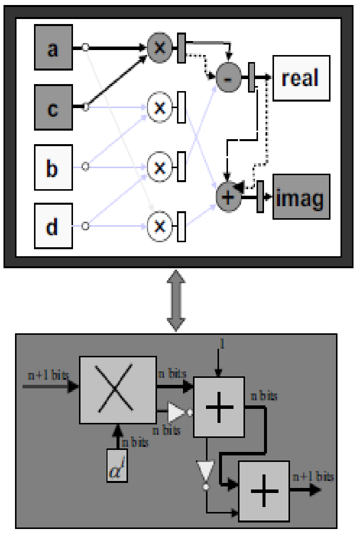


Figure. 1. The modulo (2^n+1) multiplier

The modulo (2^n+1) operator depicted in Figure 2 is carried out by :

$xy \bmod (2^n+1) = (c_L + \overline{cH} + 2) \bmod 2^n$ if $(c_L + \overline{cH} + 1 < 2^n$

$=(c_L + \overline{cH} + 1) \bmod 2^n$ otherwise

B. Modular Addition in GF(F_t)

To perform an addition that returns directly the desired result, and adder shown in Figure 2. We define s^1, s^2 the sums at the first and second adders respectively with the (n+2)-bit integer

$s^1 = [s^1_{n+1} s^1_n \ldots s_0]$

$= x+y$

Themodulo (2^n+1) addition can be expressed as:

$x + y \bmod (2^n + 1)$

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, *January 2015*

$$=\begin{cases}(x+y)2^n \text{if } 0 \le x+y < 2^n \\ (x+y)mod2^n + 2^n - 1 \quad if\ 2^n < x+y \le 2^{n+1} \\ 2^n \quad if\ (x=2^n\ and\ y=0) \\ or(x=0, y=2^n)\end{cases}$$

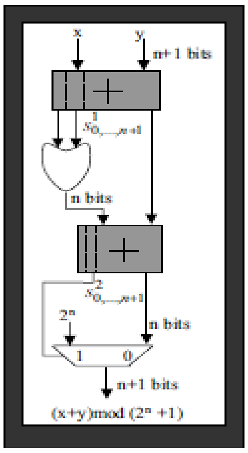


Figure. 2. The proposed mod (2^n+ 1) Adder

Now, check the correctness of equation using figure 3
First of all, let us consider x and y two elements ofGF(F_t),0≤x , y≤ 2^n ,Then
0≤ x+y ≤ 2^{n+1}
It distinguish the three following cases toestablish the correctness of our algorithm

1. For$x$+y = 2^{n+1}*(i.e x=y= 2^n)*

Here s^1=$2^{n+1}$$(i.e s^1_{n+1} = 1, s^1_i = 0$ for i = 0, … n)
Consequently s^2 = 0+ 2^n – 1, $s^2_n = 0,$
And algorithms return 2^n -1.

2. For$x$+y = 2^n (i.e=*0 and y =2^n or x= 2^n and y=0)*
we have:

$$s^1_n = 1, s^1_{n+1} = 0\ and$$
$$s^2 = 2^n + 2^n - 1 = 2^{n+1} - 1,$$

In this case $s^2_n = 1$ and the multiplexer selects 2^n as result.
This is only case where$s^2_n = 1$.

3. Finally , for 0≤ $x + y < 2^n$, we have:
$s^1_{n+1} = s^1_n = s^2_n = 0,$
And (x+y)mod $2^{n+1} = x + y$

As known, the arithmetic subtracter is usually based on the arithmetic adder structure. For the modulo $(2^n + 1)$ subtracter, we propose an operator shown in Figure 3. The subtraction modulo can be $(2^n + 1)$ expressed as follows:

$$(x-y)mod\ (2^n+1) = \begin{cases}2^n\ \ if\ (x = 2^n\ and\ y = 0 \\ (x+\overline{y}+1+s_n)mod\ 2^n\ otherwise\end{cases}$$

The three following cases toestablish the correctness of our algorithm:

1. If x≥ y ⇒ x + $\overline{y} + 1 \ge 2^{n+1}$,
$s_{n+1} = 1, s_n = 0,$
And algorithm returns $x + \overline{y} + 1$.

2. If x≤ y ⇒ x + $\overline{y} + 1 < 2^{n+1}$
$s_{n+1} = 0, s_n = 1,$
and algorithm returns $x + \overline{y} + 1 + 1$.

3. If (x =$2^n$$and\ y = 0)$ ⇒ $s_{n+1} = s_n = 1$
And the algorithm returns 2^n.

Once the different elements of the Butterfly are defined, one can implement them to obtain the reconfigurable Butterfly. Figure 5 depicts the resulting hardware operator. The switch from an operating mode to another requires a change of the Fourier kernel and the reconfiguration of connection inter-operators. Assuming that the Butterfly is configured to operate over C and one wants to perform a calculation over GF(F_t). Todo this, the Butterfly should download the primitive elementα^i, activate the different logic gate (AND,OR and the multiplexers) and reconfigure the connection inter-operators as shown in Figure 4.

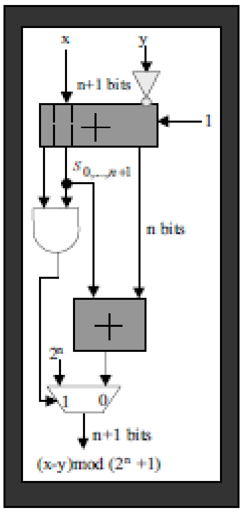


Figure . 3. The proposed mod $(2^n + 1)$ Subtracter

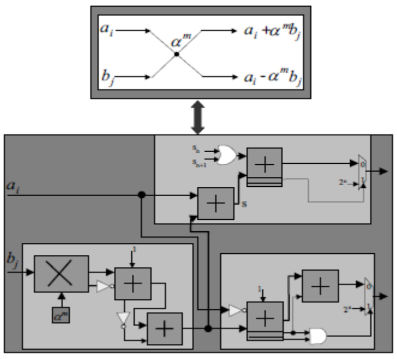


Figure. 4. The architecture of the Butterfly over GF(F_t)

IV. RECONFIGURABLE FAST FOURIER TRANSFORM

Butterfly is the important part of FFT. So reconfiguration is achieved at butterfly.by consideringTwiddle Factor(w^r_N

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, *January 2015*

)or α^i.Reconfigurable FFT flow Graph is shown in Figure 5.

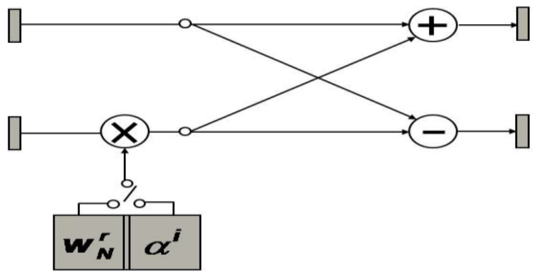


Figure.5 ReconfigurableButterfly

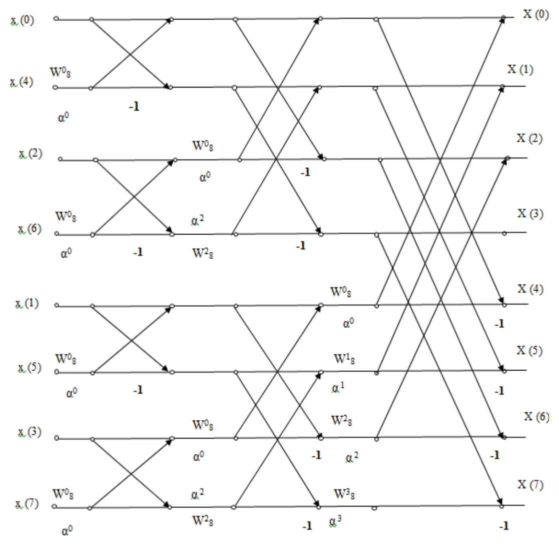


Figure 6 Reconfigurable FFT flow Graph

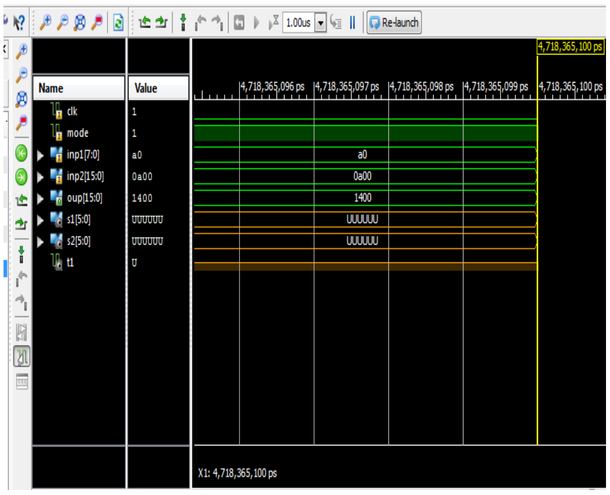
V PROPOSED RECONFIGURABLE FFT WITH RESULT

For 8 point reconfigurable operation of Fourier and Fermat number Transform,control signal selects the operating mode. Control signal value(1 or 0) indicate that that the Fourier Transform and Fermat Transform are performed respectively. The table below shows the synthesis report of proposed work with the logic resource utilization.

TABLEI. SUMMARY OF SYNTHESIS REPORT

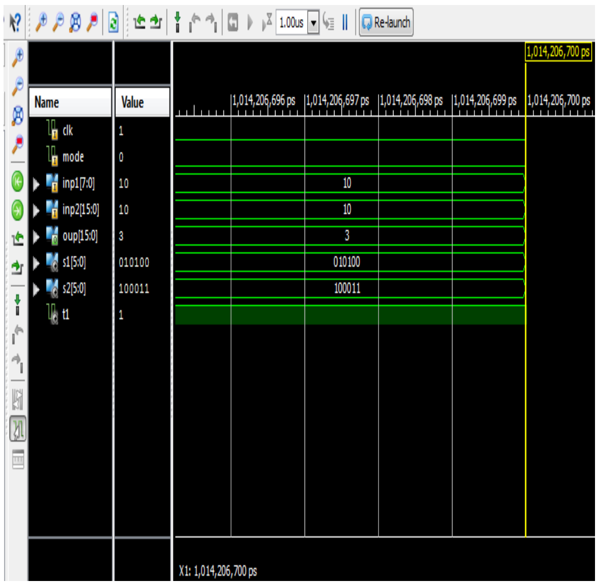
Device Utilization Summary(Estimated Value)			
Logic Utilization	Used	Available	Utilization
Number of Slice	1509	3584	42%
Number of Slice Flip flop	1494	7168	20%
Number of 4 input LUTs	2727	7168	28%
No of Bonded IOBs	258	141	182%
Number of Mul 18*18s	16	16	100%
No of GCLKS	1	8	12%

The following are simulationresults of proposed work for 8 point FFT using Sparten3.



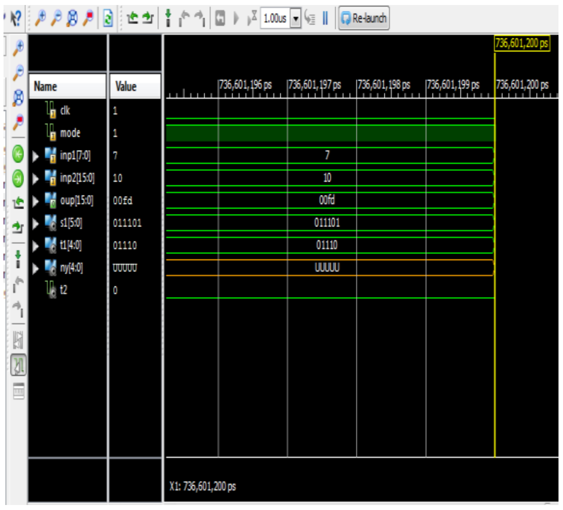
```
Name        Value
clk         1
mode        1
inp1[7:0]   a0
inp2[15:0]  0a00
oup[15:0]   1400
s1[5:0]     UUUUUU
s2[5:0]     UUUUUU
t1          U
X1: 4,718,365,100 ps
```

Figure.7.Simulation result of reconfigurable adder FFT mode



```
Name        Value
clk         1
mode        0
inp1[7:0]   10
inp2[15:0]  10
oup[15:0]   3
s1[5:0]     010100
s2[5:0]     100011
t1          1
X1: 1,014,206,700 ps
```

Figure.8.Simulation result of reconfigurable adder FNT mode

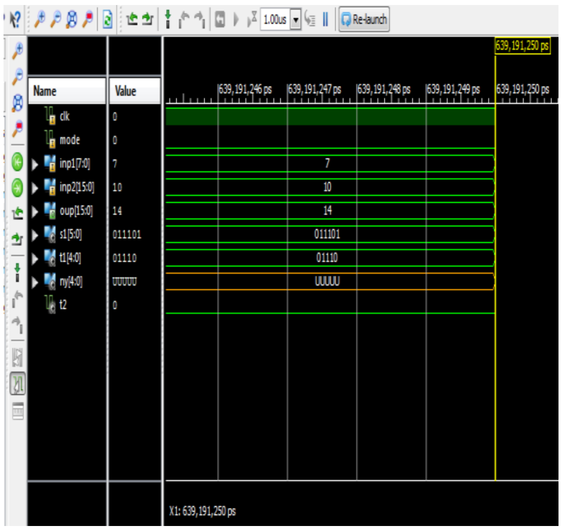


```
Name        Value
clk         1
mode        1
inp1[7:0]   7
inp2[15:0]  10
oup[15:0]   00fd
s1[5:0]     011101
t1[4:0]     01110
ny[4:0]     UUUUU
t2          0
X1: 736,601,200 ps
```

Figure.9 .Simulation result of reconfigurable Subtractur FFT mode

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

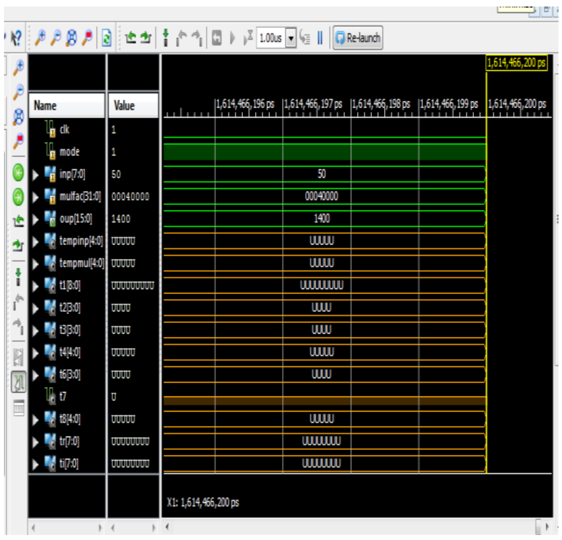
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 1, *January 2015*



Name	Value
clk	0
mode	0
inp1[7:0]	7
inp2[15:0]	10
oup[15:0]	14
s1[5:0]	011101
t1[4:0]	01110
ny[4:0]	UUUUU
t2	0

X1: 639,191,250 ps

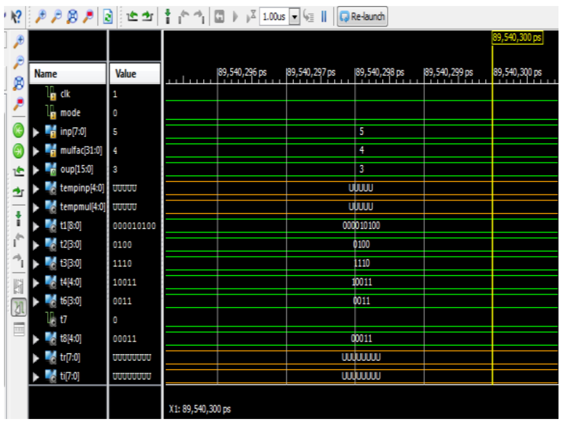
Figure.10Simulation result of reconfigurable Subtractor FNT mode



Name	Value
clk	1
mode	1
inp[7:0]	50
mulfac[31:0]	00040000
oup[15:0]	1400
tempinp[4:0]	UUUUU
tempmul[4:0]	UUUUU
t1[8:0]	UUUUUUUUU
t2[3:0]	UUUU
t3[3:0]	UUUU
t4[4:0]	UUUUU
t6[3:0]	UUUU
t7	U
t8[4:0]	UUUUU
tr[7:0]	UUUUUUUU
ti[7:0]	UUUUUUUU

X1: 1,614,466,200 ps

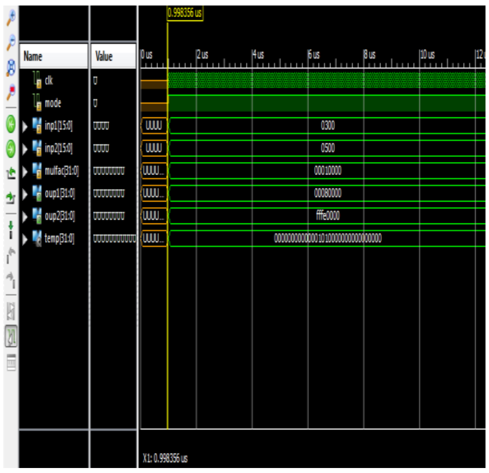
Figure.11.Simulation result of reconfigurable Multiplier FFT mode



Name	Value
clk	1
mode	0
inp[7:0]	5
mulfac[31:0]	4
oup[15:0]	3
tempinp[4:0]	UUUUU
tempmul[4:0]	UUUUU
t1[8:0]	000010100
t2[3:0]	0100
t3[3:0]	1110
t4[4:0]	10011
t6[3:0]	0011
t7	0
t8[4:0]	00011
tr[7:0]	UUUUUUUU
ti[7:0]	UUUUUUUU

X1: 89,540,300 ps

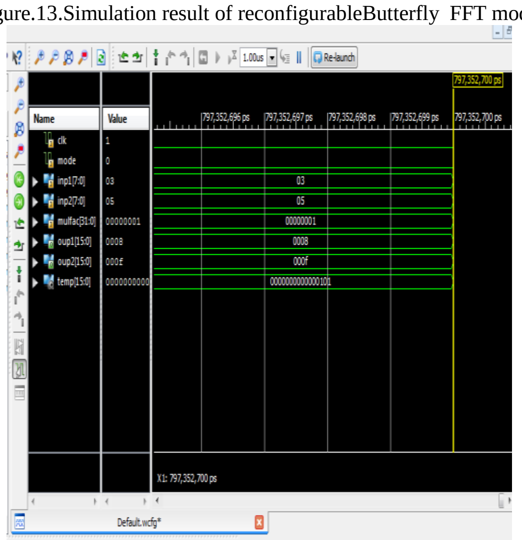
Figure.12.Simulation result of reconfigurable Multiplier FNT mode



Name	Value
clk	0
mode	0
inp1[15:0]	UUUU
inp2[15:0]	UUUU
mulfac[31:0]	UUUUUUUU
oup1[31:0]	UUUUUUUU
oup2[31:0]	UUUUUUUU
temp[31:0]	UUUUUUUUUUU

X1: 0.998356 us

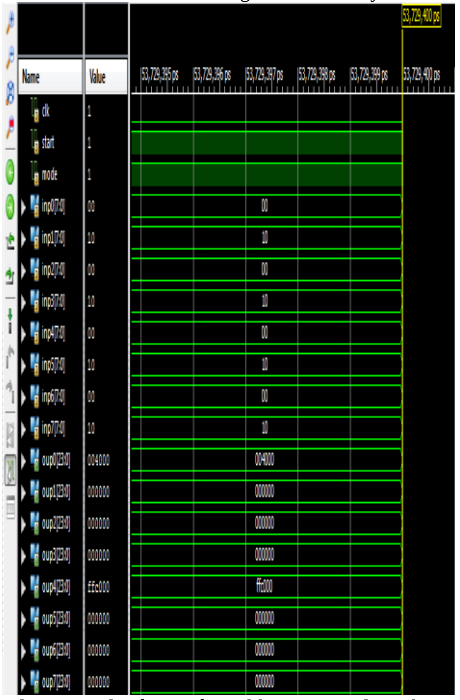
Figure.13.Simulation result of reconfigurableButterfly FFT mode



Name	Value
clk	1
mode	0
inp1[7:0]	03
inp2[7:0]	05
mulfac[31:0]	00000001
oup1[15:0]	0008
oup2[15:0]	000f
temp[15:0]	0000000000

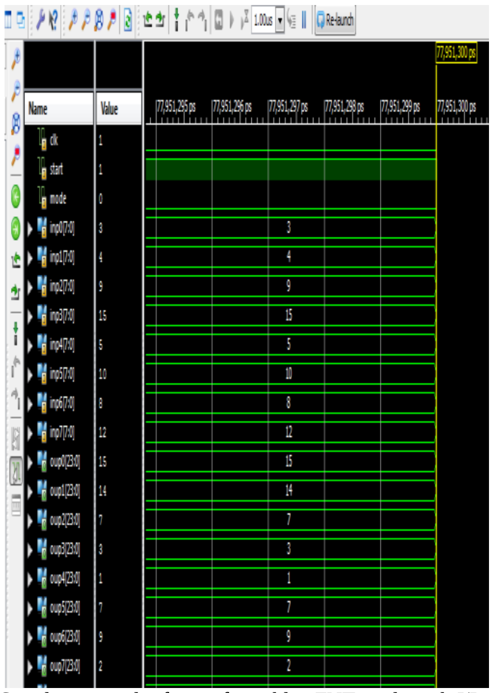
X1: 797,352,700 ps

Figure.14.Simulation result of reconfigurableButterfly FNT mode



Name	Value
clk	1
start	1
mode	1
inp0[7:0]	00
inp1[7:0]	10
inp2[7:0]	00
inp3[7:0]	10
inp4[7:0]	00
inp5[7:0]	10
inp6[7:0]	00
inp7[7:0]	10
oup0[23:0]	004000
oup1[23:0]	000000
oup2[23:0]	000000
oup3[23:0]	000000
oup4[23:0]	ffc000
oup5[23:0]	000000
oup6[23:0]	000000
oup7[23:0]	000000

Figure.15.Simulation result of reconfigurable FFT mode with I/P=[0 1 0 1 0 1 0 1]



Name	Value
clk	1
start	1
mode	0
inp0[7:0]	3
inp1[7:0]	4
inp2[7:0]	9
inp3[7:0]	15
inp4[7:0]	5
inp5[7:0]	10
inp6[7:0]	8
inp7[7:0]	12
oup0[23:0]	15
oup1[23:0]	14
oup2[23:0]	7
oup3[23:0]	3
oup4[23:0]	1
oup5[23:0]	7
oup6[23:0]	9
oup7[23:0]	2

Figure.16.Simulation result of reconfigurable FNT mode with I/P=[3 4 9 15 5 10 8 12]

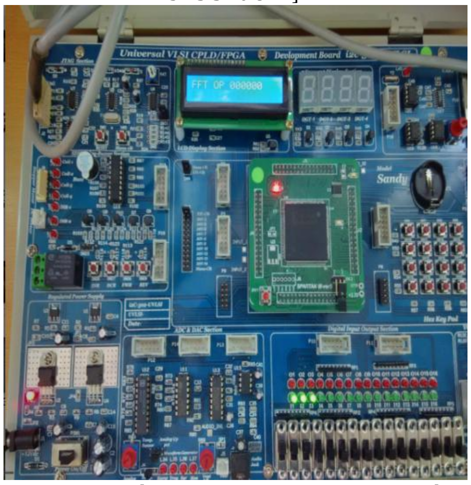


Figure.16 Implementation on Spartan 3 FFT mode

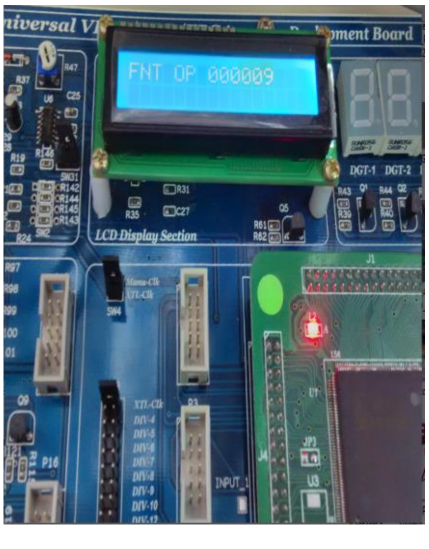


Figure.16 Implementation on Spartan 3 FNT mode

VI CONCLUSION

The re-design of this of basic structure in such way that to operate as FFT as well as FNT. For this work, a new reconfigurable arithmetic operators has been defined to build a reconfigurable Butterfly.Using this reconfigurable butterfly,FFT as common operator isacheived.

REFERENCES

[1] J. Mitola, Software Radio Architecture, Wiley, 2000
[2] J. Mitola, *The software Radio Architecture*, IEEE Communications Magazine, May 95, pp. 26-38.
[3] W. Tuttlebee, Software Defined Radio: Enabling Technologies, Wiley,2002.
[4] M.A. Sonderstrand et al., *Reisdue Number System Arithmetic:Modern Applications in Digital Signal Processing*, New York:IEEE Press, 1986.
[5] M.A. Bayoumi,G.A. Julien, and W.C. Miller, *A Look Up Table VLSI Design Methodology for RNS Structures Used in DSP Applications*, IEEE Trans. Circuits and systems, vol.34, pp. 604-616, June 1987.
[6] Richard E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University press, 2001.
[7] A. Al Ghouwayel, Y. Louët and J. Palicot, *AReconfigrable Architecture for the FFT Operator in a Software Rdio Context*, IEEE ISCAS'2006,Greece, May 21-24, 2006.
[8] W. Tuttlebee, *software defined radio: Enabling technologies*,Wiley, 2003.
[9] Al Ghouwayel, Ali, Yves Louët, and Jacques Palicot. "A reconfigurable butterfly architecture for fourier and fermat transforms." *Proceedings of WSR'06* (2006).
[10] Y. Ma, *A Simplified Architecture for Modulo (2^n + 1) Multiplication*,IEEE Transactions on Computers, 47(3) :333-337, 1998
[11] Jean-Luc Beuchat, *Some Modular Adders and Multipliers for Field Programmable Gate Arrays*, Proceedings of the 17th International Parallel and Distributed Processing Symposium. IEEE Computer Society,2003.
[12] R. Zimmerman, *Efficient VLSI Implementation of Modulo (2^n+1) Addition and Multiplication*, In Proceedings of the 14th IEEE Symposium of Computer Arithmetic, pages 158-167.