# High Performance Pipelined Design for FFT Processor based on FPGA

## A.A. Raut<sup>1</sup>, S. M. Kate<sup>2</sup>

<sup>1</sup>Sinhgad Institute of Technology, Lonavala, Pune University, India

<sup>2</sup>Sinhgad Institute of Technology, Lonavala, Pune University, India

Abstract: It is important to develop a high-performance FFT processor to meet the requirements of real time and low cost in many different systems. So a radix-2 pipelined FFT processor based on Field Programmable Gate Array (FPGA) for Wireless Local Area Networks (WLAN) is proposed. Unlike being stored in the traditional ROM, the twiddle factors in our pipelined FFT processor can be accessed directly. This paper concentrates on the development of the Fast Fourier Transform (FFT), based on Decimation-In-Time (DIT) domain, Radix-2 algorithm, this paper uses VHDL as a design entity, and their Synthesis by Xilinx Synthesis Tool on SPARTAN kit has been done. The input of Fast Fourier transform has been given by a PS2 KEYBOARD using a test bench and output has been displayed using the waveforms on the Xilinx Design Suite 12.1.The synthesis results show that the computation for calculating the 32-point Fast Fourier transform is efficient in terms of speed. The implementation was made on a Field Programmable Gate Array (FPGA) because it can achieve higher computing speed than digital signal processors, and also can achieve cost effectively ASIC-like performance with lower development time, and risks. This results show that the processor achieves higher throughput and lower area and latency.

Keywords: FFT; radix-2; FPGA; Butterfly; VHDL.

#### 1. Introduction

This paper proposes the design of 32-points FFT processing block. The work of the project is focused on the design and implementation of FFT [13] for a FPGA kit. This design computes 32-points FFT and all the numbers follow fixed point format of the type Q8.23, signed type input format is used. The direct mathematical derivation method is used for this design.

In this project the coding is done in VHDL [3][8] & the FPGA synthesis and logic simulation is done using Xilinx ISE Design Suite 12.1. The Discrete Fourier Transform (DFT) [5][7] plays an Important role in the analyses, design and implementation of the discrete-time signal-processing algorithms and systems it is used to convert the samples in time domain to frequency domain. The Fast Fourier Transform (FFT) is simply a fast (computationally efficient) way to calculate the Discrete Fourier Transform (DFT). The wide usage of DFT's in Digital Signal Processing applications is the motivation to Implement FFT's. Almost every branch of engineering and science uses Fourier methods. The words "frequency," "period," "phase," and "spectrum" are important parts of an engineer's vocabulary. The Discrete Fourier transform is used to produce frequency analysis of discrete non periodic signals. The FFT is another method of achieving the same result, but with less overhead involved in the calculations [1].



Figure 1: Radix-2 Decimation in Time Domain FFT Algorithm for Length of 32 Signals

Transforms basically convert a function from one domain to another with no loss of information. Fourier Transform converts a function from the time (or spatial) domain to the frequency domain. The mathematical formula used for the Fourier transform is as follows

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

DFT is identical to samples of the Fourier transform at equally spaced frequencies. Consequently, computation of the N-point DFT corresponds to the computation of N samples of the Fourier transform at N equally spaced

Volume 2 Issue 5, May 2013 www.ijsr.net frequencies  $\neg k = 2^{\lfloor} k/N$ . Considering input x[n] to be complex, N complex multiplications and (N-1) complex additions are required to compute each value of the DFT, if computed directly from the formula given as

$$\mathcal{X}(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1$$
$$W_{kn} = e^{-j2\pi/N}$$

To compute all N values therefore requires a total N2 complex multiplications and N (N-1) complex additions. multiplication Each complex requires four real multiplications and two real additions and each complex addition requires two real additions. Therefore a total of 4N2 real multiplications and N (4N-2) real additions are required. Besides these multiplications and additions there should be provision for storing N complex input sequences and also to store N output values. Contrary to this by using Decimation in Time [11] FFT radix-2 algorithm the number of complex multiplications and additions will be reduced to (N/2) log2N and Nlog2N to compute the DFT of a given complex x[n]. Hence in this project the Decimation in Time FFT radix-2 algorithm is implemented to compute the DFT of a sequence.

#### 2. Radix-2 DIT FFT Algorithm

The radix-2 algorithms are the simplest FFT algorithms. The decimation-in-time (DIT) radix-2 FFT recursively partitions a DFT into two half-length DFTs [13] of the even-indexed and odd-indexed time samples. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix-2 decimation-in-time and decimation-in-frequency fast Fourier transforms (FFTs) are the simplest FFT algorithms. Like all FFTs, they gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs.

The radix-2 decimation-in-time algorithm rearranges the discrete Fourier transform (DFT) equation into two parts: a sum over the even-numbered discrete-time indices  $n=[0,2,4,\ldots,N^{\perp}\ 2]$  and a sum over the odd-numbered indices  $n=[1,3,5,\ldots,N^{\perp}\ 1]$ .

$$\begin{split} X(k) &= \sum_{s=0}^{N-1} x(n) e^{-\left(i\frac{2s-k}{N}\right)} \\ &= \sum_{s=0}^{\frac{N}{2}-1} x(2n) e^{-\left(i\frac{2s-(k)}{N}\right)} + \sum_{s=0}^{\frac{N}{2}-1} x(2n+1) e^{-\left(i\frac{2s(2n+1)k}{N}\right)} \\ &= \sum_{s=0}^{\frac{N}{2}-1} x(2n) e^{-\left(i\frac{2s-k}{N}\right)} + e^{-\left(i\frac{2s}{N}\right)} \sum_{s=0}^{\frac{N}{2}-1} x(2n+1) e^{-\left(i\frac{2s-k}{N}\right)} \\ &= DFT_{\frac{N}{2}} \left[ \left[ x(0), x(2), \dots, x(N-2) \right] \right] + W_N^k DFT_{\frac{N}{2}} \left[ \left[ x(1), x(3), \dots, x(N-1) \right] \right] \end{split}$$

This is called decimation in time because the time samples are rearranged in alternating groups and a radix-2[5] algorithm because there are two groups. A basic butterfly [13] operation is shown in Figure 2, which requires only N2 twiddle-factor multiplies per stage.



Figure 2: Basic butterfly computation in the decimation-intime FFT algorithm.

The same radix-2 decimation in time can be applied recursively to the two length N2 DFTs to save computation. When successively applied until the shorter and shorter DFTs reach length-2, the result is the radix-2 DIT FFT algorithm.



### 3. FPGA

The introduction of field programmable gate arrays (FPGAs), has made it feasible to provide hardware for application specific computation design. The changes in designs in FPGA's can be accomplished within a few hours, and thus result in significant savings in cost and design cycle. FPGAs offer speed comparable to dedicated and fixed hardware systems for parallel algorithm. the 32-point FFT proposed in this paper is been simulated and synthesized using the Xilinx design suite 12.1 with the device family as Spartan 3 (low power). The summary of the device description of the vertex FPGA used is explained in the table below

Table 1. Summary of FPGA Features

Device Family	Spartan 3
Device	XC3S200
Package	vq100
Speed Grade	-5

This is called decimation in time because the time samples are rearranged in alternating groups and a radix-2 the features of the Spartan 3 FPGA [2] used in this proposed work with Xilinx Design Suite 12.1 are listed in the table below.

## International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064

Features	Spartan 3
System gates	200K
Equivalent logic cells	4320
CLB array	480
Distributed RAM bits	30K
Blocked RAM Bits	216K
Dedicated Multipliers	12
DCMs	4
Maximum User I/O	173
Maximum Differential I/O Pairs	76

### 4. Software Simulation and Results

The proposed FFT block of signal length 32 is been simulated and synthesized using the Xilinx Design Suite 12.1. The RTL block thus obtained for the decimation in time domain radix -2 Fast Fourier transform algorithm is shown

fft32	e(140.K91 <u>.00</u>		_	<u>x(1</u> 0).454.00
P 9	10103-001005-		-	5000.001105
105-02-02 MSP-020-0	n(47).1(24 <u>(0)</u>			300070-1024-40
02101000210100	NOTED ADDITION			SCHOLOM 10
4054310	NC110-2024-00			30000-0024-00
<0.431-0 y(0.431-0	01100.001100		_	witten A194-00
121210	BC193-20941005		-	20200-201105
1000000	N(240) 1(24,0)	_	_	30220 1011 10
(3)(3)(3) 	NOP 15 408 1105			-30213-4041-00
120-031.0	1.(21) =(24) (2)			2021) 2(21:0)
(3)-(31-0) y(3)-(31-0)	00000 A091.80		-	3(222) AP4 30
ve.3250	s.(22) 2(34).05.		_	20220-0241-05
00.030.0	#12/30 X/34 100			
10000	10210-103-103			90840-1014-00
78-GLD	#GH0.+GH <u>1.02</u>		_	20210-021-02
d3+d31_0 y(\$+d31.0	002300-009-1200-		_	242203.4194.003
V\$1310 V\$1310	1020 A01 00		_	2020 101 10
0101-010 dt d	#12503 ACT 1.02			
URINE URINE	$n(27) \pm 0.4(0)$		_	30273-1074-03
301-6,41-60	#(277).+(3-1 <u>,05</u>		_	3(27) +(21:0)
(2) (31:0)	6CE05 ACS 1105		_	\$1200 A[54:00
VB131D YB131D	*C200.203.1.00		_	-3030 A04 R
05-010 (020-05)	+c2943.20941205			\$1200 Activities
vielana vielana	n(240) + (24) + (2)		-	20240-1024-00
0.010	s(30).+(31.0)	_	_	-y(20) +(21.0)
	NO10 203 (198)			30213-024-00
Y00000		in.		
10.131(0				
110-104120 V010-104120				
70-624.20				
12.015 y02.010				
12x3450				
120-324-32				
18-01.0				
140.431-00 y(140.431-00				
140.431:0 y(540.431.0				
18-1310 <u>y(19-131-0</u>				

Figure 4: RTL view of a 32-point FFT

The RTL view of the butterfly structure obtained after the simulation of the 32-point FFT block, Decimation in time domain[11] is shown next and also the internal architecture of the butterfly block is shown.



Figure 5: RTL View of a Butterfly Component Used In 32-Point FFT



Figure 6: Internal Architecture of The Butterfly Component

The next are shown the simulation results of the 32-point FFT block. The s is the given simulation result is the applied input, its 32 complex numbers, y is the output in binary format while y1 is same as y but its in "real" format, so we can easily see the outputs in waveform.

Name	Value	alaria.	999,996 ps	999,997 ps	999,998 ps	999,999 ps
a (0:31)	1(1111111	1011111100		0000,0000010	100110011001100	1000004111111
₩ <u>4</u> 0:31]	[(11111101	((())))))))))))))))))))))))))))))))))))	100110010001000	0011,1111110001	100110011001100	0000/111100-
N (8-23)	000000000				huunn	
¥[0:31]	1(-5,60000	[(-5.60000);-	1 3000001, (-10, 3365	01,41.397280),(46.1	39745, 1, 67764), (-	212752,3.2256
📲 n(63:0]	1111111111	11111	1111111111011005	001100110013001		0000000

Figure 7: Simulation results of the 32-point FFT

# International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064

larae	Value	1999,996 ps	999,997 26	999,998 pi	999,999 fs
▶ ₩ [1]	(11))))011	1111110111100100000000	001100110,000000	0110110011001100	010001010
▶ 121	(00000000010	(200000001)20000000000	000000000,111111	100000000000000000000000000000000000000	0.000000
• • • • • • • • • • • • • • • • • • •	(1111111010	111111 0 111000000000000000000000000000	0000000,11111	001000100010001	001100110
▶ ¶∰ [4]	(0000001001)		000000000,111111	011000000000000000	000000
[5] 🕨 🛊	(1111111100)	113131303100110011001100	100110011.000000	0000110011001100	110011010
Þ 🖬 [6]	(00000033003)	100000000000000000000000000000000000000		011000000000000	
► 101	(11)1111010	1111110 1100 1001 1001 100	11001106.0,111111	113001300130011	0011003183
▶ ₩[10]	(00000033000)	000000000000000000000000000000000000000			011100061
▶ ₩[9]	(0000003300)	100000000000000000000000000000000000000			aaaaa 👘
▶ 🖬 [10]	(00000003300)				0000000
▶ M(11)	(00000000000)	100000000000000000000000000000000000000			(000000)
Þ 📲 [12]	(000000333000	100000000000000000000000000000000000000			000000003
▶ ₩[14]	(00000033000	120000000000000000000000000000000000000			
▶ 🖬 [14]	(00000000000)	100000000000000000000000000000000000000			
[15]	(0000003300)				00000000
▶ 📲 [16]	(00000033000		00000000,00000		00000000
▶ ₩[[J]	(000000000000				000000
<ul> <li>M [11]</li> </ul>	(000000000000	(20000000000000000000000000000000000000			(2000000)
▶ ₩ [15]	(00000000000	100000000000000000000000000000000000000			annon an
[20]	(00000000000)				(0000000)
M [21]	(00000000000)				0000000
122]	(00000033000)	100000000000000000000000000000000000000			000000000
Þ 📲 [23]	(00000033000)	100000000000000000000000000000000000000			accoccore;
[24]	(000000000000	100000000000000000000000000000000000000	,000,000,0000		accornore)
▶ ₩[2]	(00000033000	hann ann an			

lame	Value	load one be load day, be load day the load day be
[3]	(\$0000000000	+2000000000000000000000000000000000000
▶ 1 [26]	(0000000000)	100000000000000000000000000000000000000
[27]	(\$2000000038)	120000000000000000000000000000000000000
▶ 11 [28]	(0000000000)	
(29)	(0000000000	+201004000000000000000000000000000000000
▶ ♥ [30]	(0000000000)	100000000000000000000000000000000000000
Þ 📢 [31]	(0000000000	(2000) 000000000000000000000000000000000
[11:0]Y 🔡 1	[(111111010	(3111310100 [100110001005]0011,11311100010[100510011001000]010[;[1111101.]]
▶ 📲 [0]	(1111110100.	4111310100110011001100410031,13311100010110013003009100130105
▶ 🐏 [1]	(1111101010	{111130001001010001110130110000130,1151131000010010010010101011051105
i 🖌 🖬 [2]	(1111101111)	1111130 1110000100100100100000000000000
🕨 📲 [3]	(1111110011)	(111111001100010001101010001100,000000011001100110011100010001001
▶ 🌿 [4]	(1111101110	411130113011310000100011000130,0000000000
🕨 📢 (S)	(1111101111)	41111100111001010100100000000000000000
[6] 🐏 🕷	(00000000000	(0000000000000000000000000000000000000
▶ № [7]	(0000320100	400001010010111011110001000110001001010110110001100111010
• • • [3]	(0000010100	(00000101001005100510019001501000,111111001005100510051000100115)
(1) M	(00000000000	+poonexpectation:accuationaction(); [111110 [1111110:01100100:000];
▶ 📲 [10]	(1111110001)	(11111)(00010300000111)(000001011,11111)(011010000000000
▶ 🐏 (t)	(11111101010	411111010101110110101010100000000000000
▶ № [12]	(111111110)	(111111) 13000010051110(11011001,000000)0001301001100000(s0163101)
» 📢 (13)	(1111111100)	4111111 1001 1000 1001 110 000101 1011,00000000
► M [14]	(1111111010)	(11111)(000111010010000001100,000000100110010011(01110001)
(15)	(0000001000	40000000 0000 10000 111 100 1011 1011,000000 100000 1110000 110110000
▶ 駴 [16]	(00000310110	(0000000) 001000100010001001100,0000000000

SHOC

11 10000

	<ul> <li>M [17]</li> </ul>	10310011303	400000 (000010001611001110000000,111110001010101
	Þ 📲 [18]	(1111)111110	+1111101101101011000001100000,11100100100
	[19]	(1111100110)	(111110)1010010001001111001000,111100000000
	▶ 📲 [20]	11111101001	{  1111dp.00300001111011 [01111033,000000010010011111300000100111}}
	▶ ₩ [21]	(1111111001	{
	[22]	(1111111110	{  11111
	▶ 📲 [23]	(1111110101	(1) 11 1 (0) 10000101 100001 000 1011000, 11111 11 (0010100101
	▶ 📲 [24]	(1111101111)	(111110;1110110011001100;1000,0000000000
	(۵) 🚰 🔺	(1111110110	{  111   \$1 1000   001   01   \$20   \$20   101   \$1,00000   \$000   1000   111   \$101   \$100
	þ 📲 [26]	(11111111100	{111111111000101113010000111000101,00000000
	▶ ₩ [27]	(1111111000	{111111p0005011113000101311030000,0000000001113013000001013000001}
	Þ 📲 [28]	(1111110101.	{111113[00111001300001[001000131,0000000[111111100113003[10000311]}
	[29]	(11111111110	(11111) p110(111000) 1010(00110000, 000000) 101100000(11100)
	[30]	[0386001383	40000000001138100001101391000011,0000000010010001011010001100110
	[11]	10310003101	4000000011051000001100511110150051,11111901001001111105000001510310540}
Þ	📲 (8-21)	umuumu	
۷	¥ pt[0:31]	1(-5.600000	}+5.600000,+7.3000000;;+10.336501,+1.3972805;;+6.209745;3.4774641;;+6.232752;3.2256
	▶ ₩111	1-5.600000,	(5.60000,-7.30000)
	▶ 🖬 D1	(-10.00650)	(+10.336501,+1.397200)
	▶ ₩ [2]	1-8.209745,	(-0.209745,3.4774 <del>84</del> )
	▶ 📲 [J]	1+6.232752,	(6.232752,3.22564)
	Þ 🙀 [4]	(-8.618090,	(-0.515090,-4.119230)-
	▶ 📲 [5]	1-0.417169,	( 6.417599, t0.5990(b0)
	Þ 📲 [1]	10.154291,1	40.154281,15.3722(6)
	In 10 10 10 10 10 10 10 10 10 10 10 10 10	110.366737.3	410.366737.9.444295)

Name	Value	1	1999,997 ps	999,996 ps	1999, 999 ps
P 1/2	120.055107.		10.000000000000000000000000000000000000	MY.	
is) and the	(20.300000,		1013000072300	un La	
N 191	(9.283700,-4		40.202040(-0.0213	61)	
[11]	(-7.194626,		1/7.191626, (2.304	681	
[11]	(-5.268839,		÷5.266839,4.4454	(c)	
[12]	(-0.949410,-		+0.949410,4.4121	×1	
Þ 📲 [13]	(-1.615407,:		(+1.605407,2.7047	36}	
Þ 📲 [14]	(-2.539759,-		42,539759,6,2992	纲	
[15]	(4.132621,1		(4.132521,10.0590	80)	
Þ 📲 [16]	(13.600000,		(13 60000,3 7000	<b>0</b> 0}	
Þ 🖬 [17]	(12.696217,		12 636217,410-338	635)	
[11]	(-0.790255,		0.790255,-17.148	63)	
Þ 🖬 [15]	(-12.668705		-12.668706,-9.421	907}	
[21] 🖌 📢	(-11,484910		(411.484900,2.2800	90)-	
[21]	(-9.096283,-		÷3.096283,4.5376	65)	
[22]	(-0.558480,		\$-0.568480,-1.1438	00	
[23]	(-5.456899,		(-5.456899,-3.3576	54)	
[24]	(-8.300000,		(-0.30000),1.2000	061	
[25]	(-4.926616,-		44,926616,4,7576	761	
► 🖬 [26]	(-1.815374;:		÷1.815374,2.7754	환	
[27]	(-3.628500,4		1-3.428500,0.8985	41)	
24]	(-5.050590,		15.050590, 3.9878	70)	
► M [25]	1-0.527941.4		10.527941.6.7573	971	
► M [31]	(4.953959.2		(4.953959,2.2723)	(5)	
[21] 🎽 🖌	(2.756338,+		<i>{2.756338,-5.6889</i>	2)	

The final synthesis report is shown using the table below

Table 3: Summary Oo Spartan 3 Featur	res Used In The 32-
Point Fast Fourier Trans	form

1 Ouric	a mansie	/111	
Device Utilization Summary (estimated values)			
Used	Available	Utilizatio	
1646	1920	85%	
2884	3840	75%	
892	63	1415%	Resource
			overuse
12	12	100%	
	Summa Used 1646 2884 892	Summary (estima           Used         Available           1646         1920           2884         3840           892         63           12         12	Summary (estimated values           Used         AvailableUtilization           1646         1920         85%           2884         3840         75%           892         63         1415%           12         12         100%

 Table4: Shows The Time Delay Of 32-Point Fft(Radix-2

 Decimation In Time Domain)

Min. delay	42.58ns (22.515ns logic, 20.003ns route)	
	(53.0% logic, 47.0% route)	
Total REAL time to Xst completion: 20.00 secs		
Total CPU t	ime to Xst completion: 19.77 secs	
Total memory usage is 219856 kilobytes		

### References

- Sneha N.kherde, Meghana Hasamnis, "Efficient Design and Implementation of FFT", International Journal of Engineering Science and Technology (IJEST), ISSN : 0975- 5462 NCICT Special Issue Feb 2011
- [2] Ahmed Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy, "Efficient FPGA implementation of FFT/IFFT Processor", INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING, Issue 3, Volume
- [3] 3, 2009
- [4] Hardware Description Language. URL: http://en.wikipedia.org/wiki/Hardware\_description\_lang uage
- [5] Very High Speed Integrated Circuit Hardware Description Language.URL: http://electrosofts.com/vhdl/
- [6] Alan V. Oppenheim, Ronald W. Schafer with John R. Buck, Discrete Time Signal Processing, Second Edition

- [7] B. Parhami, Computer Arithmetic, Algorithms and Hardware Designs, 1999
- [8] James W. Cooley and John W. Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series
- [9] Peter J. Ashenden, The Designer's Guide to VHDL, Second Edition.
- [10]N. Weste, M. Bickerstaff, T. Arivoli, P.J. Ryan, J. W. Dalton, D.J. Skellern", and T.M. Percivalt A 50 Mhz 16Point-FFT processor for WLAN applications.
- [11]Saad Bouguezel, M. Omair Ahmad, "IMPROVED RADIX- 4 AND RADIX-8 FFT ALGORITHMS" IEEE. Department of Electrical and Computer Engineering Concordia University 1455 de Maisonneuve Blvd. West Montreal, P.Q., Canada.
- [12]Ali Saidi ," DECIMATION-IN-TIME-FREQUENCY FFT ALGORITHM" Motorola Applied Research, Paging and Wireless Data Group Boynton Beach.
- [13]Rizalafande Che Ismail and Razaidi Hussin "High Performance Complex Number Multiplier Using Booth-Wallace Algorithm" School of Microelectronic Engineering Kolej University Kejuruteraan Utara Malaysia.
- [14]Bergland, G. D. "A Guided Tour of the Fast Fourier Transform." IEEE Spectrum 6, 41-52, July 1969.

## **Author Profile**



Atul Raut received the B.E. degrees in Electronics and Telecommunications Engineering from Sinhgad Institute of Technology in 2009. Currently he is doing M.E. in same institute.



**Prof. Sandeep M. Kate** have completed ME Electronics and working as Assistant Professor in Sinhgad Institute of Technology, Lonavala, Pune University, India. He guided several ME

students for project.