Design of Efficient Fast Fourier Transform

Shymna Nizar N. S
PG student, VLSI & Embedded Systems, ECE Department
TKM Institute of Technology
Karuvelil P.O, Kollam, Kerala-691505, India

Abhila R Krishna
Assistant professor, ECE Department
TKM Institute of Technology
Karuvelil P.O, Kollam, Kerala-691505, India

***Abstract*-Fast Fourier Transforms have become an integral part of any digital communication system and a wide variety of approaches have been tried in order to optimize the algorithm for a variety of parameters, primarily being memory and speed. Major problem in FFT calculation is the increased number of complex multiplication units. Folding transformations are used to design FFT architectures with reduced number of functional units. In the folding transformation, many butterflies in the same column can be mapped to one butterfly unit. A highly efficient pipelined folded FFT architecture for 8 point R2 FFT is presented here. When compared with the normal R2 FFT architecture, the pipelined architecture shows efficiency both in speed and area consumption. The FFT block is designed to be capable of computing 8 point FFT and employs R2 (Radix2) architecture which is simple, elegant and best suited for communication applications. VHDL coding is simulated and synthesized in Xilinx ISE Design Suite 12.1.**

***Keywords*—FFT, Pipelining, folding transformation, life time chart, register allocation, FPGA**.

I. INTRODUCTION

The discrete Fourier transform (DFT) is an important signal processing block in various applications, such as communication systems, speech, signal and image processing. The efficient implementation of DFT is fundamental in many cost and hardware constraint applications. One such method is co referred as fast Fourier transform (FFT) algorithm. FFT is an important digital signal processing (DSP) technique to analyse the phase and frequency components of a time-domain signal. An N point FFT can compute the DFT in only ($N\log_2 N$) operations. The difference in speed can be substantial, especially for long data sets where N may be in the thousands or millions. In practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to $N=\log_2 N$.

This huge improvement made the calculation of the DFT practical. FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

II. LITERATURE SURVEY

The Discrete Fourier Transform (DFT) is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. Computing the DFT of N points in the naive way, using the definition, takes N^2 arithmetical operations. The DFT is the most important discrete transform, used to perform Fourier analysis in many practical applications The DFT of of input sequence x(n), according to the DFT formula is:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\Pi nk/N} \qquad (1)$$

$$X(k) = \sum_{n=0}^{N-1} x(n) \left(W_N^{kn}\right) \qquad (2)$$

where $\left(W_N^{kn}\right)$ is the twiddle factor

A. Applications of DFT

The discrete Fourier transform finds limitless applications in many areas of signal processing. In the era of fast computing it has become increasingly important to enhance the existing FFT algorithms to meet the ever increasing applications in the field of digital signal processing. DFTs have also been extensively used in multi-carrier transmission systems like orthogonal frequency domain multiplexing (OFDM) as FFT processors. In multi-carrier modulation, such as OFDM and discrete multi tone (DMT), data symbols are transmitted in parallel on multiple sub-carriers. Multi-carrier modulation based transceivers involve real-time DFT computations [9]. FFT based channel estimation method is derived from the maximum likelihood criterion, which is originally proposed for OFDM systems with pilot preambles. In order to save bandwidth and improve system performance, decision-feedback (DF) data symbols are usually exploited to track channel variations in subsequent OFDM data symbols, and this method is called DF DFT-based channel estimation.

DFT is extensively used in sonar and radar systems. These systems use millions of multiplications per second. Computerized tomography is widely used to synthetically form images of internal organs of the human body where in massive amounts of signal processing is required. Remote sensing is another field employing huge amount of processing. Satellite photographs can be processed digitally to merge several images or enhance features or combine information received on different wavelengths.

B. Fast Fourier Transform(Fft)

The idea behind the FFT is the divide and conquer approach, to break up the original N point sample into two (N=2) sequences. This is because a series of smaller problems is easier to solve than one large one. The DFT requires (N - 1) *2 complex multiplications and N *(N - 1) complex additions as opposed to the FFT's approach of breaking it down into a series of 2 point samples .Decimation is the process of breaking down something into its constituent parts. Decimation in time involves breaking down a signal in the time domain into smaller signals, each of which is easier to handle. If the input (time domain) signal, of N points, is x(n) then the frequency response X(k) can be calculated by using the DFT . Since the recombination algebra of FFT takes N complex multiplications and there are $\log_2$ (N) stages, the approximate number of complex multiplications is N $\log_2$ (N).This means that this decimation approach has reduced the number of complex multiplications from N squared (N^2) to $N\log_2(N)$. At high values of N (i.e., large signals) this is a massive saving. The recombination uses flow graph notation and is called a butterfly network .

Table 1.Bit Reversal Property

In-order index	In-order index in binary	Bit-reversed binary	Bit-reversed index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

The input-output relation is obtained by using the Bit reversal property. The process of decimating the signal in the time domain has caused the input samples to be re-ordered. For an 8 point signal the original order of the samples is 0, 1, 2, 3, 4, 5, 6, 7 .But after decimation the bit reversed order as per the table 2.1 is 0, 4, 2, 6, 1, 5, 3, 7 .

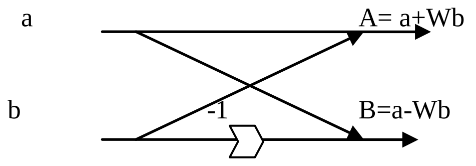


Figure 1. Basic butterfly computation in the DIT FFT algorithm

The FFT butterfly is a graphical method of showing multiplications and additions involving the samples. Standard graph flow notation is used where each circle with entering arrows is an addition of the two values at the end of the arrows multiplied by a constant. The constant is a number which appears beside the arrow, if there is no value then the constant is taken as one. Figure 1 shows the butterfly diagram of DIT FFT. If the inputs are a and b ,then the output after twiddle factor multiplication will be a+Wb and a-Wb respectively. For DIF FFT the result will be a+b and (a-b)W. The FFT has a fairly easy algorithm to implement, and it is shown step by step in the list below.

1. Pad input sequence, of N samples; with zeros until the number of samples is the nearest power of two. E.g. 500 samples are padded to 512.

2. Bit reverse the input sequence. E.g. 3 = 011 goes to 110 = 6.

3. Compute (N=2) two sample DFT's from the shuffled inputs.

4. Compute (N=4) four sample DFT's from the two sample DFT's.

5. Compute (N=2) eight sample DFT's from the four sample DFT's.

6. Until the all the samples combine into one N-sample DFT.

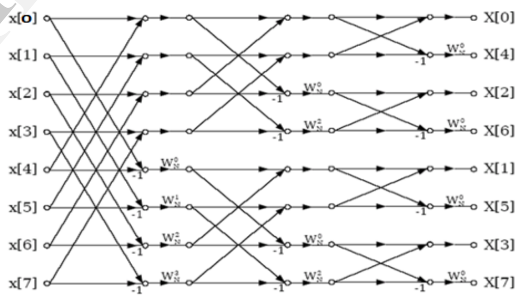


Figure 2. Basic butterfly computation in 8 point R2 DIF FFT algorithm.

C. Computational Complexities

Discrete Fourier Transform is complicated to work out as it involves many additions and multiplications involving complex numbers. Even a simple eight sample signal would require 49 complex multiplications and 56 complex additions to work out the DFT. At this level it is still manageable; however a realistic signal could have 1024 samples which requires over thousands of complex multiplications and additions. The number of calculations required soon mounts up to unmanageable proportions.

The paper is organized as follows. The different existing techniques for DFT calculations and the problems related to these techniques are explained in section II. section III explains about FFT, it’s butterfly method of calculation and the proposed method of FFT calculation by using folding and pipelining concept are also in it .The simulation results are discussed in detail in Chapter IV.

Conclusions and on going works are discussed in the last section.

III. PROPOSED METHOD

From the hardware perspective, Field Programmable Gate Array (FPGA) devices are increasingly being used for hardware implementations in communications applications. FPGAs at advanced technology nodes can achieve high performance, while having more flexibility, faster design time, and lower cost. As such, FPGAs are becoming more attractive for FFT processing applications and are the target platform of this discussion. The architecture applies well to real time data streaming scenarios and the architecture is discussed with respect to Decimation in Frequency (DIF) type of decomposition. The architecture design for pipeline FFT Processor's had been the subject of intensive research as early as 1970's when real time processing was demanded in such applications such as radar signal processing, well before the VLSI technology had advanced to the level of system integration. Several architectures have been proposed over the last 2 decades since then, along with the increasing interest and the leap forward of the technology.

Radix-2 multi-path delay commutator is one of the most classical approaches for pipelined implementation of radix-2 FFT. Efficient usage of the storage buffer in R2 multi-path delay commutator leads to the Radix-2 Single-path delay feedback architecture with reduced memory. The prior FFT architectures were derived in an adhoc way, and their derivations were not explained in a systematic way. Here a pipelined folded DIF FFT architecture is derived by using folding transformation, register minimization and linear life time chart.

A. Folding transformation

The folding transformation is used to systematically determine the control circuits in DSP architectures where multiple algorithm operations are time multiplexed to a single functional unit there by reducing the number of functional unit in implementation resulting in an IC with low silicon area[8].

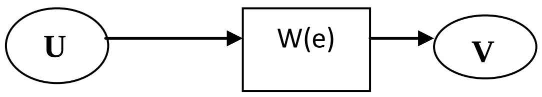


Figure 3. Data flow graph

Folding transformation can be explained with an example .Consider two edges U and V with delay w(e) in between them . Data Flow Graph (DFG) of it is shown in the figure 3.

B. Folding order (u,v)

It is the time partition to which a particular node is scheduled to execute the hardware function.

C. Folding factor (N)

It is the number of operations folded to a single functional unit. If hardware Hu is pipelined by Pu stages ,then the result of lth iteration of node U is available at the time unit Nl+ U + Pu . Since the edge U $\rightarrow$ V has w(e) delay's, the result of l' th iteration of node U is used by the (l+w(e))' th iteration of node v, which is executed at

$$N(l + w(e)) + v \qquad (3)$$

There for result must be stored for Df (U$\rightarrow$V)

$$= [N(l+w(e))+v]-[Nl+Pu+u] \qquad (4)$$

$$= Nw(e)-Pu+v-u \qquad (5)$$

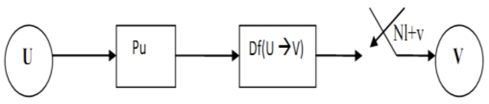


Figure 4. Folded DFG

So the the new delay Df can be calculated by using the equation 3.5.Folded data flow graph of figure 3 is shown in figure 4 with the new delay Df.

D. Folding set

It is an ordered set of operations executed by the same functional unit. Each folding set contains N entries .Some of which may be null operations . For a folding set to be realizable Df(U$\rightarrow$V) >= 0 must hold for all of the edges in the Data Flow Graph. The DFG can be pipelined to ensure that the folded hardware has non-negative number of delays.

E. Feed-forward architecture

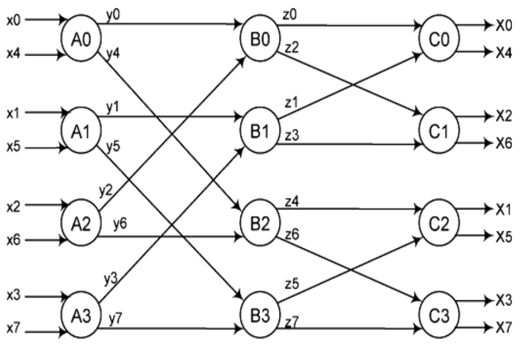


Figure 5. DFG of 8 point DIF FFT

For obtaining a feed forward architecture of FFT, a folding set has to be derived from the DFG of FFT which is shown in figure 5. Here FFT is divided in to 3 sections of 4 operations each. Where nodes A0,A1,A2,A3 performs the functions of section 1. Similarly nodes B0,B1,B2,B3 and nodes C0,C1,C2,C3 performs the functions of sections II and II respectively. Each node performs a butterfly operation.

The folding sets considered are

A = {Ǿ, Ǿ, Ǿ, Ǿ, A0,A1,A2,A3}
B = {B2,B3,Ǿ, Ǿ, Ǿ, Ǿ,B0,B1}
C = {C1,C2,C3,Ǿ, Ǿ, Ǿ, Ǿ,C0 }

Assume that the butterfly operations do not have any pipeline stages, i.e Pa = Pb=Pc=0. The folded architecture can be derived by writing the folding equation Df(U→V) = Nw(e)-Pu+v-u for all the edges in the DFG. The folded delays for the DFG are given by

DF(A0→B0) = 2	DF(B0→C0) = 1
DF(A0→B2) = - 4	DF(B0→C1) = -6
DF(A1→B1) = 2	DF(B1→C1) = 0
DF(A1→B2) = -4	DF(B1→C1) = -7
DF(A2→B0) = 0	DF(B2→C2) = 1
DF(A2→B2) = -6	DF(B2→C3) = 2
DF(A3→B1) = 0	DF(B3→C2) = 0
DF(A3→B3) = -6	DF(B3→C3) = 1

For example,
Consider,
DF(A0→B0) = 2

It means that there is an edge Df (U→V) from the butterfly node A0 to the node B0 in the folded DFG that is having two delays. For the system to be realizable, Df (U→V) >=0 must hold for all the edges in the DFG.

From the delay calculation some edges are obtained with negative delays which are not realizable. DFG has to be pipelined for avoiding these negative delays so that the system can be realized.

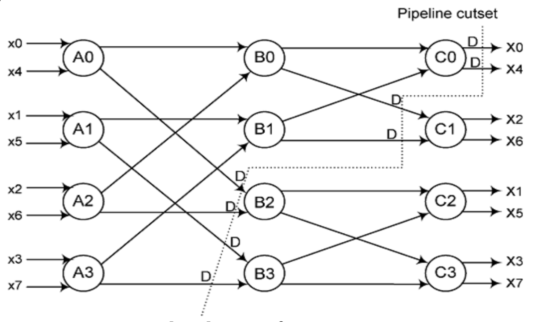


Figure 6. Pipelined DFG of 8 point FFT

Figure 6. shows the pipelined DFG of 8 point DIF FFT. Dotted lines shows the cutset which is used to pipeline the DFG so as to ensure that folded hardware has non-negative delays.

The folded delays for the pipelined DFG are given by

DF(A0→B0) = 2	DF(B0→C0) = 1
DF(A0→B2) = 4	DF(B0→C1) = 2
DF(A1→B1) = 2	DF(B1→C1) = 1
DF(A1→B2) = 4	DF(B1→C1) = 1
DF(A2→B0) = 0	DF(B2→C2) = 1
DF(A2→B2) = 2	DF(B2→C3) = 2
DF(A3→B1) = 0	DF(B3→C2) = 0
DF(A3→B3) = 2	DF(B3→C3) = 1

F. Life time analysis

From delay calculations of pipelined DFG, it can be observed that ,24 registers are required to implement the folded architecture. Lifetime analysis technique is used to design the folded architecture with minimum possible registers[5]. A data sample (also called a variable) is live from the time it is produced through the time it is consumed. After the variable is consumed, it is dead. A variable occupies one register during each time unit ,in which it is live. In lifetime analysis, the number of live variables at each time is computed, and the maximum number of live variables is determined . This is the minimum number of registers required to implement the DSP program.

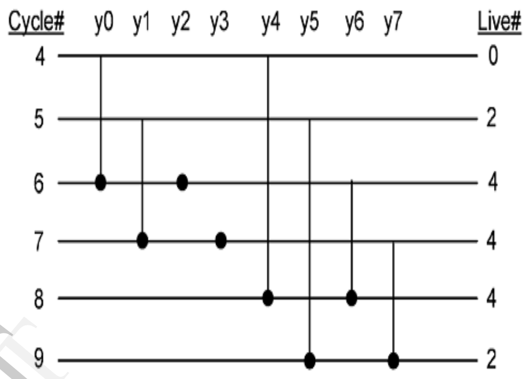


Figure 7. Linear life time chart

A linear life time chart graphically represent the lifetime of each variable in a linear fashion. For example, in the current 8-point FFT design, consider the variables y0,y1,….y7 i.e.,the outputs at the nodes A0,A1,A2,A3 respectively. It takes 16 registers to synthesize these edges in the folded architecture.

The linear lifetime chart for the variables y0,….y7 is shown in Figure 7. The horizontal lines represent clock cycles and the vertical lines represent the life time of the variables. Here the variable y0 is produced at the clock cycle 4 and is dead at clock cycle 6 .By the convention that the variable is not live during the clock cycle it is produced but is live during the clock cycle it is consumed, y0 is live during clock cycles 5 and 6. Similarly for each variables we can determine the lifetime.

From the lifetime chart, it can be seen that the folded architecture requires 4 registers as opposed to 16 registers in a straight forward implementation since the maximum number of live variables is equal to the minimum number of registers. The next step is to perform forward-backward register allocation.

G. Register minimization technique

Once the minimum number of registers required to implement the DSP program has been determined, the data need to be allocated to these registers. Forward backward register allocation is an allocation scheme that can be used to allocate data to the minimum number of registers[7].

Register allocation can be performed using an allocation table. The allocation scheme dictates how the variables are assigned to registers in the allocation table. The steps used to perform forward –backward register allocation are as follows.

Step 1: Determine the minimum number of registers using lifetime analysis.

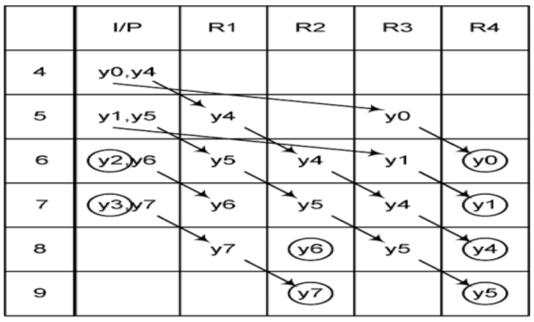
Step 2: Input each variable at the time step corresponding to the beginning of its lifetime. If multiple variables are inputted in a given cycle , these are allocated to multiple registers such that the variable with the longest lifetime is allocated to the initial registers and the other variables are allocated to consecutive registers in decreasing order of lifetime.

Step 3: Each variable is allocated in a forward manner until it is dead or it reaches the last register .If the register is not available, then the variable is allocated to the first available forward register.
Step 4: Since the allocation is periodic, the allocation of the current iteration also repeats itself in subsequent iterations.

Step 5: For variables that reach the last register and are not yet dead, the remaining life period is calculated ,and these variables are allocated to a register in a backward manner on a first come first served based. If multiple registers are available for backward allocation, first the register that was already used for backward allocation from the last register to this register has to be used.

In the case where more than one register qualifies for backward allocation, choose the register with the minimum number of forward registers among all candidate registers that have a sufficient number of forward registers to complete the allocation of the variable. After a variable has been allocated backward, allocate it forward until it is dead or it again reaches the last register.

Step 6: Repeat steps 4 and 5 as required until the allocation is complete.



	I/P	R1	R2	R3	R4
4	y0,y4				
5	y1,y5	y4		y0	
6	y2,y6	y5	y4	y1	y0
7	y3,y7	y6	y5	y4	y1
8		y7	y6	y5	y4
9			y7		y5

Figure 8. Register allocation table

For example, let's take y0,y4 which are inputted at the 4 th clock cycle. From the life time chart y4 has the largest clock cycle than y0,so by the steps of register minimisation technique y4 is inputted to R1 and y0 in to R4 so that y0 is outputted at the 6th clock cycle .

H. Pipelined folded 8 point DIF FFT architecture

Similarly lifetime analysis and register allocation techniques can be applied for the variables x0,x1,….x7 and z0,z1…z7 , inputs to the DFG and the outputs from nodes B0,B1,B2,B3 and C0,C1,C2,C3respectively.

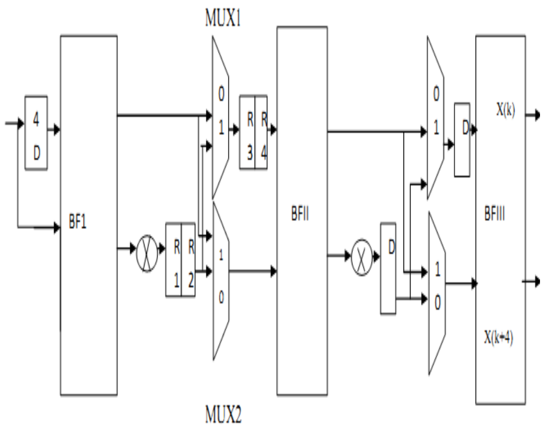


Figure 9. Pipelined Folded 8 point DIF FFT architecture

From the allocation table in figure 8. and the folding equations, the final architecture in figure 9. can be made. Where BF1,BFII and BFIII represents the butterfly units .D represents the delay units for the proper synchronization of the inputs. The first input of BF1 will be x(0) and x(4) .Similarly, first output will be x(0) and x(4).

IV. SIMULATION RESULTS

The design entry is modelled using VHDL in Xilinx ISE Design Suite 12.1 and the simulation of the design is performed using Isim from Xilinx ISE to validate the functionality of the design. Figure 4.1 shows the magnitude and phase response of the FFT results by using the MATLAB FFT function .MATLAB results are used for checking the FFT outputs

A. MATLAB result

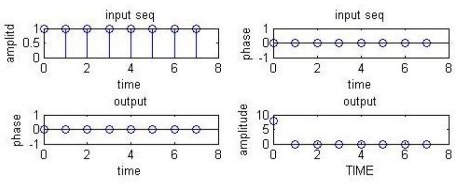


Figure 10. Matlab FFT results

B. XILINX simulation result of 8 point FFT

Here the real inputs are represented by x(0),x(1)……x(7) and it's corresponding FFT outputs are shown by y(0),y(1)……..y(7).The outputs are obtained at 1,150,000ps.

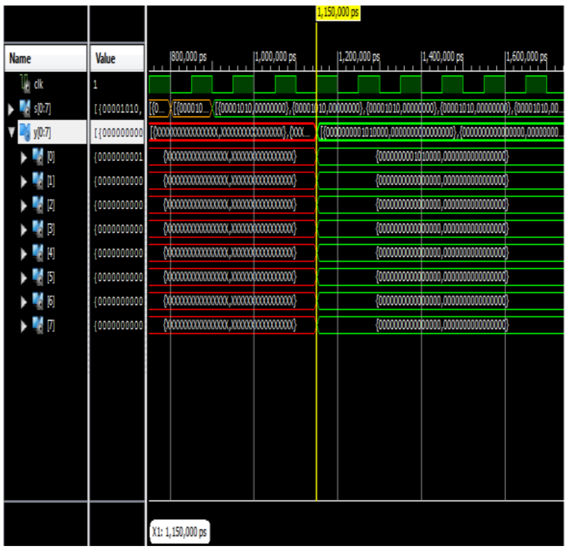


Figure 11: 8 point FFT results

C. XILINX simulation results of pipelined folded R2 8 point DIF FFT

By using the blocked diagram of pipelined folded 8 point R2 DIF FFT, architecture was developed in VHDL .For the correct synchronisation of the inputs delay units are used . Simulation results of 4 delay and 1 delay are shown in figure 12. and 15. As per the architecture, output of the 4 D and the inputs are given to butterfly units one after the other by using mux and registers. Output of BF1 is shown in figure13. The full output of 8 point R2 pipelined folded DIF FFT is shown in figure 15 .

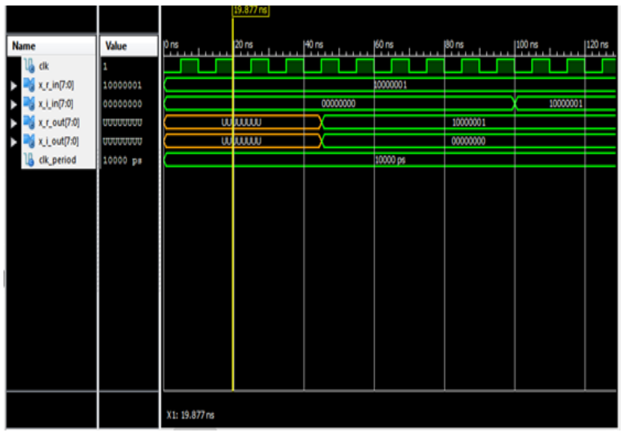


Figure 12.4 Delay unit

Here the real and imaginary part of the input are given separately as an array of 8 input where .x_r and x_i represents the real and imaginary parts of it.

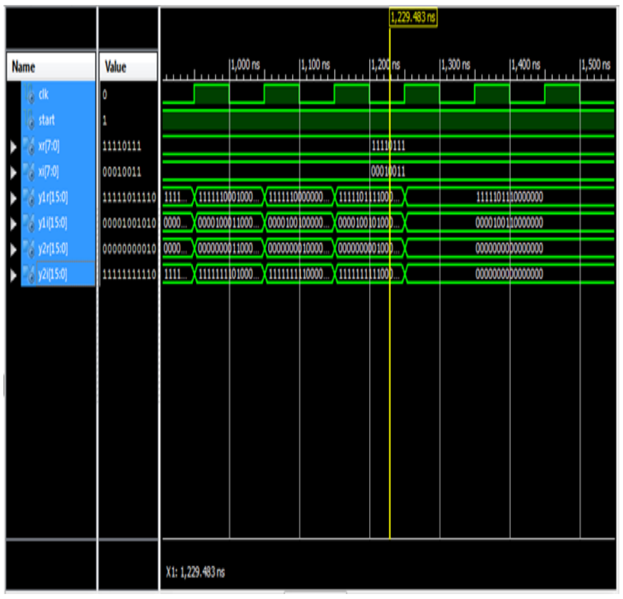


Figure 13. Butterfly unit I

when the input is given to 4D ,the output is produced only after the 4 clock cycle which is done because of the fact that BF unit I needs x(0) and x(4) as the first input for processing. Butterfly unit performs the corresponding butterfly operations along with the twiddle factor multiplication.

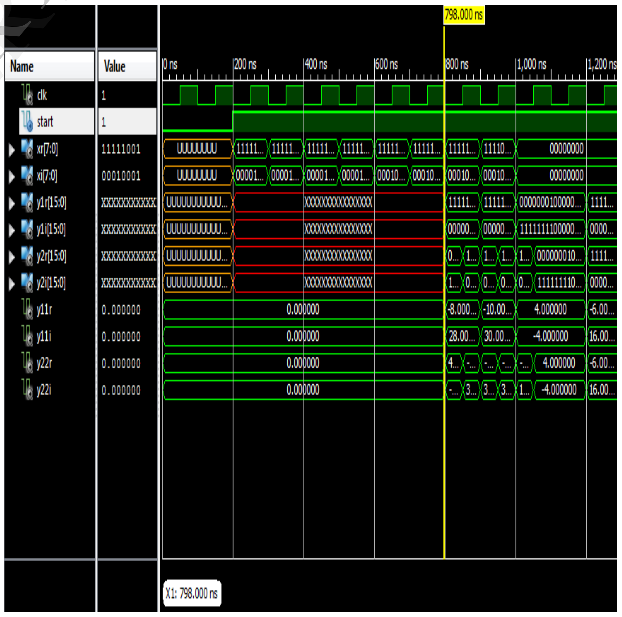


Figure 14. Output of 6 th clock cycle from mux 2 and R4

The allocation table of figure 8 shows that the first output of the R4 is available only after the 6 th clock cycle .Simulation result of figure 14. shows this ,where output from mux 2 and R4 which is the input of BFII is obtained only after the 6 th clock cycle.

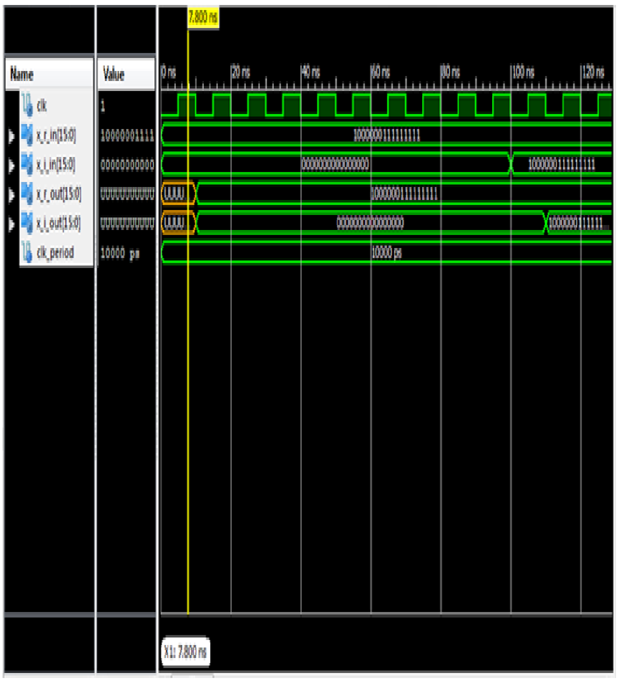


Figure 15. 1Delay unit

Similar to 4 Delay generation 1 delay unit as represented in figure 15. is also used for proper synchronisation of inputs to BFIII.

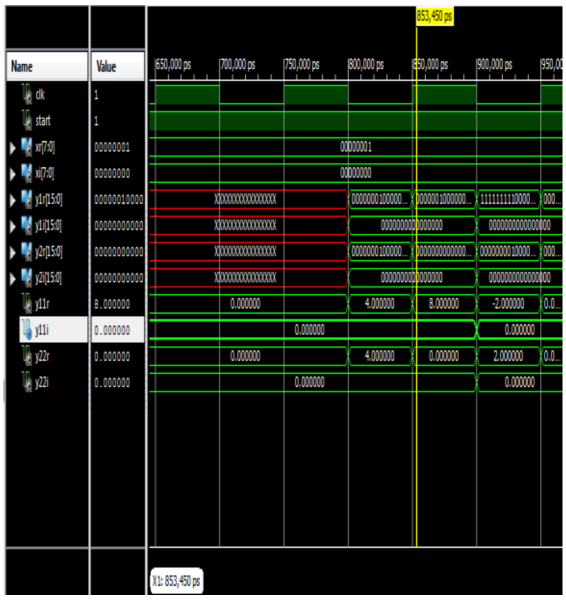
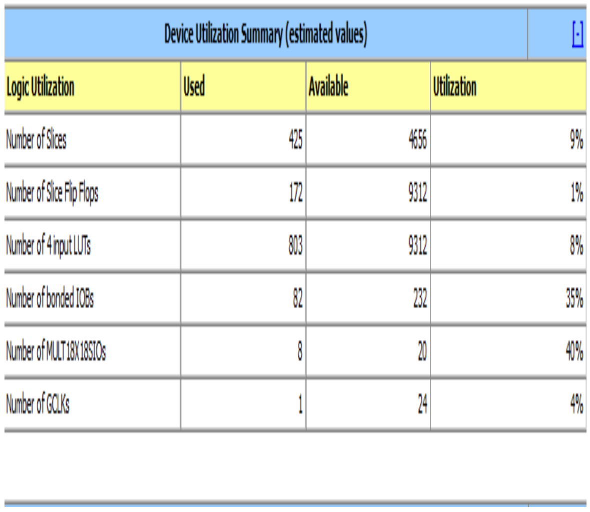


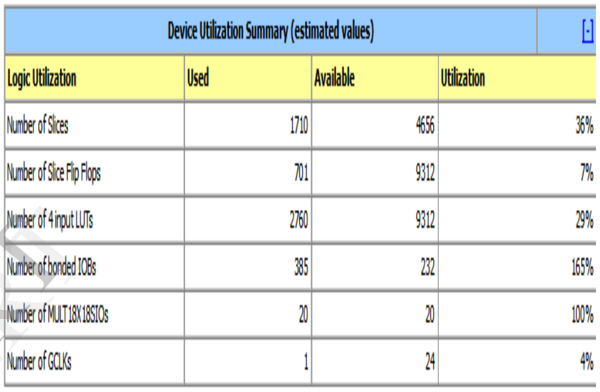
Figure 16. 8 point pipelined folded DIF FFT

Figure 16.8 represents the full FFT output .Here also the real and imaginary part of the input and output are obtained separately.y11_r and y11_i represents the real and imaginary part of the first 4 outputs. Similarly y22_r and y22_i represents the real and imaginary part of the last 4 outputs. Here the outputs starts to come at 850,000 ps.so it is clear that pipelined architecture produces the output with less time than the normal architecture.



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	425	4656	9%
Number of Slice Flip Flops	172	9312	1%
Number of 4 input LUTs	803	9312	8%
Number of bonded IOBs	82	232	35%
Number of MULT18X18SIOs	8	20	40%
Number of GCLKs	1	24	4%

Figure 17. Synthesize report of 8 point pipelined folded DIF FFT



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1710	4656	36%
Number of Slice Flip Flops	701	9312	7%
Number of 4 input LUTs	2760	9312	29%
Number of bonded IOBs	385	232	165%
Number of MULT18X18SIOs	20	20	100%
Number of GCLKs	1	24	4%

Figure 18.Synthesize report of 8 point normal DIF FFT

The synthesize report of two architectures shows that pipelined architectures produces the results very efficiently in less time with less number of slices in FPGA.

V. CONCLUSIONS

Folding transformations are used to design FFT architectures with reduced number of functional units. In the folding transformation, many butterflies in the same column can be mapped to one butterfly unit. The FFT block is designed to be capable of computing 8 point FFT and employs R2 (Radix2) architecture which is simple, elegant and best suited for communication applications. The pipelined folded DIF FFT processor and the normal architecture based FFT processor has been simulated and synthesized successfully in HDL environment. The performance analysis shows that the pipelined architecture produces the results very fastly than normal architecture while consuming less number of slices than normal butterfly architecture of FFT.

REFERENCES

[1] Akash Verma, B.S. Rai "Analysis of synthesis issues about designing DSP devices"*International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering(An ISO 3297: 2007 Certified Organization)* Vol. 2, Issue 8, August 2013

[2] Pramod Kumar Meher, Senior Member, IEEE, and Sang Yoon Park, Member IEEE "CORDIC Designs for Fixed Angle of Rotation" *IEEE Transactions on very large scale integration (vlsi) systems,* vol. 21, no. 2, february 2013

[3] Mario Garrido, J. Grajal, M.A. S ´anchez and Oscar Gustafsson, Pipelined Radix- 2k Feedforward FFT Architectures*" IEEE Transactions on very large scale integration (vlsi) systems, (21), 1, 23-32.june 2013*

[4] Xilinx, Inc., "Virtex-5 FPGA User-Guide,"*UG190*, v4.5, p. 383, Jan. 2009

[5] K.K.Parhi, "Calculation of minimum number of registers in arbitary life time chart,"*IEEE Trans.on Circuits and Systems-II*,vol .41,no.6,pp.434-436 June 1994.

[6] K.K.Parhi,C.-Y.Wang, and A.P.Brown, "Synthesis of control circuits in folded pipelined DSP architecture,"IEEE *Journal of Solid –State Circuits,*vol .27,no.1,pp.29-43,Jan .1992.

[7] K. K. Parhi, "Systematic synthesis of DSP data format converters using lifetime analysis and forward-backward register allocation," IEEE Trans. on Circuits and Systems - II, vol. 39, no. 7, pp. 423-440, July 1992.