FPGA Implementation of 64/128/256 Point Radix-8 Pipelined FFT/IFFT Core

M.S.Naidu[1], Dr.B.Sridhar[2] and V.Nanchariah[3]

[1]M.Tech student, Dept of ECE,LIET,Vizianagaram
[2] Professor,Dept of ECE,LIET,Vizianagaram
[3] Associate Professor, Dept of ECE,LIET,Vizianagaram
[1] naidu2010in@gmail.com, [2]srib105@gmail.com, [3] nanch84@gmail.com

Abstract

Orthogonal frequency-division multiplexing (OFDM) is one of the techniques for parallel transmission, which has received great attention in high-speed data communication systems. It has been selected for several communication standards such as Ultra-wideband, Long Term Evolution and Digital Video Broadcasting – Terrestrial. The most important operations in the OFDM are the FFT operations. Discrete Fourier Transform (DFT) is a fundamental digital signal processing algorithm used in many applications, including frequency analysis and frequency domain processing. DFT is the decomposition of a sampled signal in terms of sinusoidal (complex exponential)components. The symmetry and periodicity properties of the DFT are exploited to significantly lower its computational requirements. The resulting algorithms are known as Fast Fourier Transforms (FFTs). A 64-point DFT computes a sequence x(n) of 64 complex valued numbers given another sequence of data X(k) of length 64 .The basis of the FFT is that a DFT can be divided into smaller DFTs. In the processor. USFFT64 a radix-8 FFT algorithm is used. It divides DFT into two smaller DFTs of the length 8. In this paper, we proposed radix-8 highly pipelined FFT architecture for 64-point, 128-point and 256-point. The proposed FFT is implemented in Xilinx ISE 14.7 using Verilog coding on FPGA device XC4SX25-12. The simulation results are achieved in terms of device utilization and RTL schematic, Power Analysis for 64-point, 128-point and 256-point FFTs.

Keywords: VLSI signal Processing, FFT algrotihm, FGGA architecture, OFDM, power analysis

1. Introduction

Congratulations! Your paper has been accepted for journal publication. Please follow the steps outlined below when submitting your final draft to the IJAMTES Press. These guidelines include complete descriptions of the fonts, spacing, and related information for producing your proceedings manuscripts. Please follow them and if you have any questions, direct them to the production editor in charge of your journal at the IJAMTES. Fourier transform is a mathematical Tool which is analyze the circuit and frequency synthesis for various systems in electronics and communications system such as filter design in signal processing image processing, stochastic signal model for various type physical singles measurements[1]. Similarly other areas for real world applications those are in biomedical engineering, seismic transducers in Earth Sciences, antennas in electromagnetic, and microphones in communication engineering, etc. Observing the signals in time domain is very difficult later Baron Jean Baptiste Fourier, more than a century ago, showed that any waveform that exists in the real world can be represented (i.e., generated) by adding up sine waves. In order to desire a high speed data transfer system which would satisfy the several standards is received great attention selected for several communication standards such as Ultra-wideband, Long Term Evolution and Digital Video Broadcasting, a Orthogonal frequency-division multiplexing (OFDM) is used[2,3]. FFT module is a part of the OFDM which compute the Discrete fourier transform. The FFT is implement on dsp system by radix-n, where n is the smallest computation value of sequence. Simple form is Radix-2 based FFT processors, however now days a most complex value radix-4 algorithm, but requires a 4-point butterfly unit with higher complexity, radix-22, radix-23, radix- 24 and radix-2k FFT algorithms based FFT processors are being developed. FFT architectures are chosen which offers a high throughput rate with low hardware complexity and low power consumptions. The proposed radix-8 highly pipelined FFT architecture for 64-point, 128-point and 256-point. The proposed FFT is implemented in Xilinx ISE 14.7 using Verilog coding on FPGA device XC4SX25-12. The results are obtain in terms of device utilization and RTL schematic, Power Analysis for 64-point, 128-point and 256-point FFTs[4].

2. FFT ALGORITHMS

The radix-n (FFTs) are the simplest FFT algorithms used to compute DFT of various types of signals with low cost of computation. Simplest is radix – 2. It Computation made up of radix – 2 butterflies. In Radix-2 two types of methods are generally implemented. There are Decimation in time(DIT),Decimation in frequency(DIF).[5]

The radix-2 FFT algorithms are used for data vectors of lengths N = 2K. They proceed by dividing the DFT into two DFTs of length N=2 each, and iterating. Radix-2 small DFT is called basic butterfly in the diagram, which have two inputs, output is sum of the two inputs and subtraction of two inputs.

DECIMATION ON FREQUENCY (DIF)

The radix-2 algorithms are the simplest FFT algorithms. The decimation-in-frequency (DIF) radix-2 FFT shown in fig:1 that partitions the DFT computation into even-indexed and odd-indexed outputs, which can each be computed by shorter-length DFTs of different

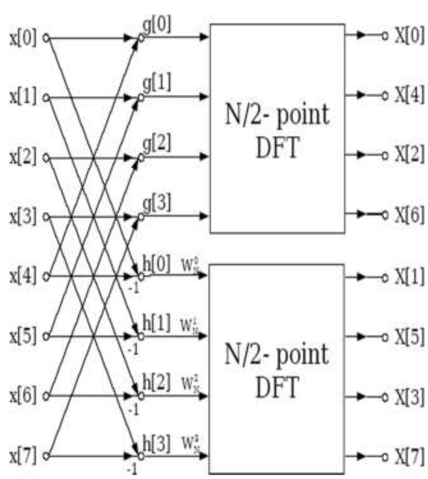

Fig:1:DIF FFT decimation in stage 1

Combinations of input samples. Recursive application of this decomposition to the shorter-length DFTs results in the full radix-2 decimation-in-frequency FFT.

In general Complex multiplication of DFT is: N^2 Complex multiplication of FFT is (N/2)log2(N) .

The basis of the FFT is that a DFT can be divided into smaller DFTs. In the processor USFFT64 a radix-8 FFT algorithm is used. It divides DFT into two smaller DFTs of

the
length 8, as it is shown in the formula:

$$X(k) = X(8r+s) = \sum_{m=0}^{7} W_8^{mr} W_{64}^{ms} \sum_{l=0}^{7} x(8l+m) W_8^{sl}, r = 0 \text{ to } 7, s = 0 \text{ to } 7, \quad (1)$$

Which shows that 64-point DFT is divided into two smaller 8-point DFTs? The input complex data x(n) are represented by the 2-dimensional array of data x(8l+m). The columns of this array are computed by 8-point DFTs. The results of them are multiplied by the twiddle factors W_{64}^{ms}. And the resulting array of data X(8r+s) is derived by 8-point DFTs of rows of the intermediate result array. The 8-point DFT, named as the base FFT operation, is implemented by the Wino grad FFT algorithm, which provides the minimum additions and multiplications (only 2 complex multiplications to the factor W_8^1). As a result, the radix-8 FFT algorithm needs only 64complex multiplications to the twiddle factors $W_{64}^{ms}$and 32 multiplications to the twiddle factor W_8^1except of 4096 complex multiplications in the origin DFT. Note that the well-known radix-2 64-point FFT algorithm needs 192 complex multiplications[6].

3. Radix- 8 development architecture

Each base FFT operation is computed by the procedure unit, referred to as FFT8, that is that the information path for FFT calculations.FFT8 calculates the 8-point DFT in the high pipelined mode.
Therefore, in every clock cycle one imaginary number is scan from the computer file buffer RAM and also the complicated result's written within the output buffer RAM. The 8-point DFT algorithm is divided into several stages which are implemented in the stages of the FFT8 pipeline. This supports the increasing the clock frequency up to 250 rate and better. The latent delay of the FFT8 unit from input the primary information to output the primary result's up to fourteen clock cycles[7].

High precision computations
In the core the inner information bit dimension is higher to four digits than the input file bit dimension.
The main error source is the result truncation after multiplication to the factorsW64 ms as a result of the foremost of base FFT operation calculations are additions, they are calculated without errors. The FFT results have the information bit dimension that is higher in 3digits than the input file bit dimension, which provides the high data range of results when the input data is the sinusoidal signal. The maximum result error is a smaller amount than the one least vital little bit of the input file .Besides, the normalizing shifters are attached to the outputs of FFT8 pipelines, which provide the proper bandwidth of the resulting data.The overflow detector outputs give the chance to input the right shift left bit range for these shifters[8].
Low hardware volume
The USFFT64 processor has the minimum multiplier factor variety that is capable four DSP units..
This truth makes this core enticing to implement in ASIC. When configuring in Xilinx FPGA, these multipliers are enforced in four DSP48 units severally. The user will choose the computer file, output knowledge, and constant widths which give application dynamic vary wants. This can minimize each logic hardware and memory volume.

Block Diagram

The basic block diagram of the USFFT64 core [9]with two data buffers is shown in the fig..

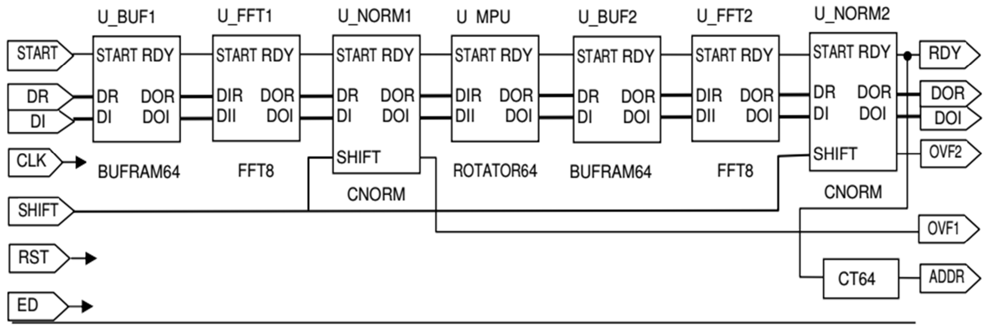

Fig 2:Block diagram of 64-point Radix-8 FFT implementation

Sorting

Sorting is ordering a list of objects. We can distinguish two types of sorting. If the number of objects is small enough to fits into the main memory, sorting is called internal sorting. If the number of objects is so large that some of them reside on external storage during the sort, it is called external sorting.

FIFO (First In First Out)

A FIFO is a special type of buffer. The name initial out FIFO inventory accounting stands for initial in initial out and implies that the info written into the buffer initial comes out of it first.

There are different kinds of buffers just like the {lifo|last in initial out|LIFO|inventory accounting} (last in first out), often called a stack memory, and the shared memory. The choice of buffer design depends on the appliance to be solved .FIFOs can be implemented with software or hardware.

The choice between package and a hardware answer depends on the appliance and also the options desired.

When requirements change, a software FIFO easily can be adapted to them by modifying its program, while a hardware FIFO ,the advantage of the hardware FIFOs shows in their speed. Every memory within which the info word that's written in initial conjointly comes out initial once the memory is scan may be a first-in first-out memory.

Exclusive Read/Write FIFOs

In exclusive read/write FIFOs, the writing of information isn't freelance of browse clock.

There are temporal arrangement relationships between the write clock and therefore the browse clock.

For instance, overlapping of the read and the write clocks could be prohibited. To permit use of such FIFOs between 2 systems that employ asynchronously to 1 another, an external circuit is required for synchronization. But this synchronization circuit usually considerably reduces the datarate.

Concurrent Read/Write FIFOs

In synchronic read/write FIFOs, there is no dependence between the writing and reading of data. Simultaneous writing and reading are potential in overlapping fashion or in turn. This means that 2 systems with completely different frequencies will be connected to the FIFO. The designer needn't worry regarding synchronizing the 2 systems as a result of this can be taken care of within the FIFO[10]. Concurrent read/write FIFOs, depending on the control signals for writing and reading, fall into two groups:

•SynchronousFIFOs

•AsynchronousFIFOs

Synchronous FIFOs

Synchronous FIFOs are management lead supported strategies of control evidenced in processor systems.

Every digital processor system works synchronal with a system- wide clock signal. This system temporal arrangement continues to run notwithstanding and actions are being dead. Enable signals, also often called chip-select signals, start the synchronous execution of write and read operations in the various devices, such as memories and ports.The block diagram shows all the signal lines of a synchronous FIFO. It needs a free-running clock from the writing and another from the reading system. Writing is controlled by the

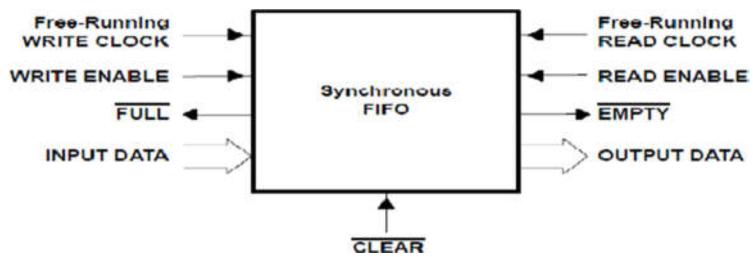

Fig3 . Synchronous FIFO

WRITE modify input synchronous with WRITE CLOCK. The FULL standing line will be synchronal entirely with WRITE CLOCK by the free-running clock.

In an identical manner, data words are read out by a low level on the READ ENABLE input synchronous with READ CLOCK.

Here, too, the free-running clock permits one hundred pc synchronization of the EMPTY signal with browse CLOCK.

Asynchronous FIFOs:

The control signals of an asynchronous FIFO correspond most closely to human intuition and were, in the past, the only kind of FIFO driving. The block diagram in Figure 3.8 shows the control lines of an asynchronous FIFO, and illustrates the typical timing on these lines in a read and writes operation.

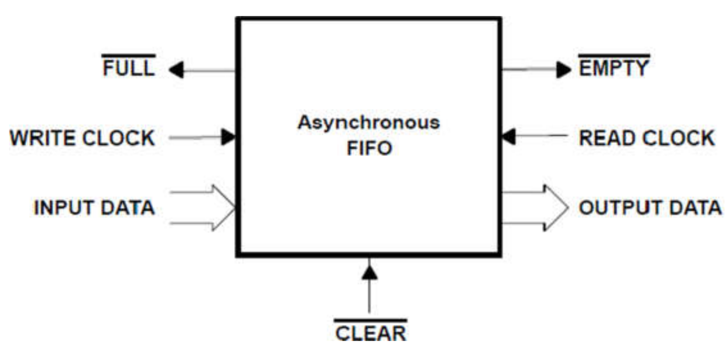

Fig 4: Asynchronous FIFO

FIFO Push/Pop Operations

The Queue block stores a sequence of input samples **during a first** in, **first** out (FIFO) register.

The register **capability is about** by the Register size parameter, and inputs **are often** scalars, vectors, or matrices.

The block pushes the input at the In port on to the end of the queue when a trigger event is received at the Push port. When a trigger event is received at the Pop port, the block pops the first element off the queue and holds the Output port at that value. The first input to be pushed onto the queue is always the first to be popped off. A trigger event at the **optional** RST port empties the queue contents.

When select Clear output port on reset, then a trigger event at the RST port empties the queue and sets the value at the Out port to zero. This setting also applies when a disabled subsystem containing the Queue block is re enabled; the Out-port value is only reset to zero in this case when select Clear output port 2048 as an example and explain the input scheduling as follows. Initially the 12 memory banks are logically grouped into four sets and , Each set is in charge of one input stream. From the first to the 3N/4th cycle, the memory banks keep the first to$3N/4^{th}$ samples of each input stream. For the case of N=2048, the memory banks and **d3">**store the samples 1th–512th, 513th–1024th, 1025th–1536th} of the first, the second, the third, and the fourth input streams, respectively.

From the $(3N/4+1)^{th}$ to the Nth cycle the radix-4 butterfly processes the read- out data from the memory set and then this memory set are updated with the incoming samples from stream B,C, and D. That is, together with the previously stored first to 3N/4th samples, now the radix-4 butterfly can process the samples of stream A, because the $(3N/4 + 1)^{th}$ to the N^{th} samples square measure prepared at this moment, also, since only one butterfly is used at each stage, the $(3N/4 + 1)^{th}$ to the Nth samples for input streams B, C,and D are stored in the vacated memories a1, a2, and a3, respectively.

Continuing with **the instance** of N = 2048, at the end of the 2048th clock cycle, the radix-4 butterfly has computed the 2048 samples of stream A, and the memory set is updated with the 1537th to the 2048th samples of stream B,C, and D respectively.

4. Implementation FFT on FPGA board

4.1 Simulation tools

behavioral Simulation Using ISE Simulator

Since we need to test set of top modules in your task, we can perform behavioral recreation on the configuration utilizing the ISE Test system. ISE has full reconciliation with the ISE Test system. ISE empowers ISE Test system to make the work registry, aggregate the source records, stack the configuration, and perform reproduction focused around recreation properties[9].

To choose ISE Test system as project test system:

- Right click the tab of source in device(xc3s700A-4fg484).
- After selecting properties, ISE simulator needs to be selected in the field.

Locating the Simulation Processes:

To run the simulation process, enable ISE simulator design simulation processes. For locating process of ISE simulator:

- Select Behavioral Simulation in the Sources tab for field.
- Create test bench file.

The tracing simulation processes are available

(i) **Syntax Checking:** This procedure weighs for linguistic structure flaws in the test.

(ii) Behavioral Model: simulation of design process starts at this stage.

(iii) **Self-checking HDL test bench**: Enables to obtain a self-checking HDL test bench similar to a test bench waveform (TBW) folder and include the test bench to the project. We can also utilize this process to update a previous self-checking test. The test bench obtained by this method contains output data and self-checking data that can be used to evaluate the data from afterwards simulation runs.

Specifying Simulation Properties

For performing a behavioral recreation on the stopwatch plan after you set a few procedure properties for reproduction. ISE permits you to set a few ISE Test system properties notwithstanding the recreation net rundown properties. To see the behavioral re-enactment properties, and to adjust the properties for this exercise. In the Sources tab, select the test bench file.

- For expanding ISE simulator press + for hierarchy
- Press Right-button for Simulate Behavioral Model process.
- After Properties being selected, set display level to advanced.
- This global setting enables us to see all available properties.
- Click Apply and click OK

5. Results

The proposed radix-8 FFT/IFFT is implemented for 64/128/256 point on XC4VSX25-12 FPGA Device in Xilinx ISE 14.7.

The device utilization report for 64-point FFT is shown in Fig.4 and RTL schematic as shown in table:1

Device Utilization Summary [-]				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2,025	20,480	9%	
Number of 4 input LUTs	3,095	20,480	15%	
Number of occupied Slices	2,128	10,240	20%	
Number of Slices containing only related logic	2,128	2,128	100%	
Number of Slices containing unrelated logic	0	2,128	0%	
Total Number of 4 input LUTs	3,118	20,480	15%	
Number used as logic	3,027			
Number used as a route-thru	23			
Number used as Shift registers	68			
Number of bonded IOBs	87	320	27%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number of FIFO16/RAMB16s	4	128	3%	
Number used as RAMB16s	4			
Number of DSP48s	4	128	3%	
Average Fanout of Non-Clock Nets	2.91			

RTL schematic of 64-point is shown in Fig.4

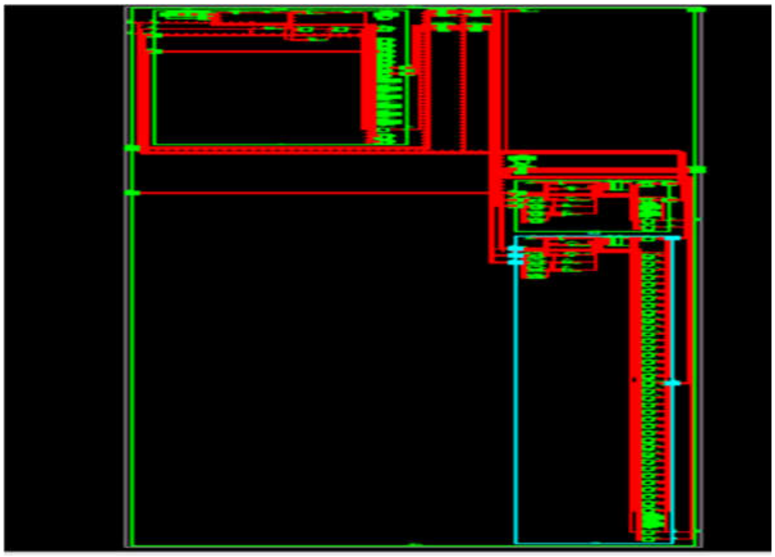

Fig.5 RTL schematic

Power Analysis report of 64-point FFT is shown in Table:2.

Device	
Family	Virtex4
Part	xc4vsx25
Package	ff668
Temp Grade	Commercial
Process	Typical
Speed Grade	-12

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250

Characterization	
PRODUCTION	v1.0,02-02-08

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.048	1	---	---
Logic	0.000	3105	20480	15
Signals	0.000	4774	---	---
BRAMs	0.000	4	128	3
DSPs	0.000	4	128	3
DCMs	0.000	0	4	0
IOs	0.000	87	320	27
Leakage	0.332			
Total	0.380			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	8.7	81.7	53.3

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.185	0.040	0.145
Vccaux	2.500	0.062	0.000	0.062
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.380	0.048	0.332

The device utilization report for 128-point FFT is shown in Table:3.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	26	20,480	1%	
Number of 4 input LUTs	11	20,480	1%	
Number of occupied Slices	20	10,240	1%	
Number of Slices containing only related logic	20	20	100%	
Number of Slices containing unrelated logic	0	20	0%	
Total Number of 4 input LUTs	26	20,480	1%	
Number used as logic	11			
Number used as a route-thru	15			
Number of bonded IOBs	69	320	21%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number of FIFO16/RAMB16s	1	128	1%	
Number used as RAMB16s	1			
Average Fanout of Non-Clock Nets	1.75			

RTL schematic of 128-point is shown in Fig.6

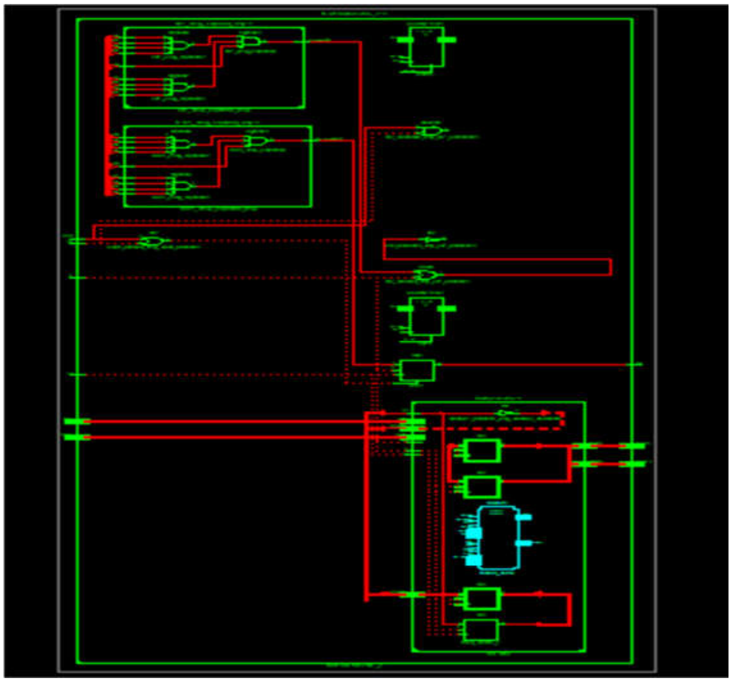

Fig:6 RTL schematic 128

Power Analysis report of 128-point FFT is shown in Table:4.

A	B
Device	
Family	Virtex4
Part	xc4vsx25
Package	ff668
Temp Grade	Commercial
Process	Typical
Speed Grade	-12
Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Characterization	
PRODUCTION	v1.0,02-02-08

C	D	E	F	G	H
	On-Chip	Power (W)	Used	Available	Utilization (%)
	Clocks	0.008	1	---	---
	Logic	0.000	26	20480	0
	Signals	0.000	115	---	---
	BRAMs	0.000	1	128	1
	DCMs	0.000	0	4	0
	IOs	0.000	69	320	22
	Leakage	0.331			
	Total	0.338			
	Thermal Properties		Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
			8.7	82.1	52.9

I	J	K	L	M	N
	Supply	Summary	Total	Dynamic	Quiescent
	Source	Voltage	Current (A)	Current (A)	Current (A)
	Vccint	1.200	0.150	0.006	0.144
	Vccaux	2.500	0.062	0.000	0.062
	Vcco25	2.500	0.001	0.000	0.001
			Total	Dynamic	Quiescent
	Supply	Power (W)	0.338	0.008	0.331

The device utilization report for 256-point FFT is shown in Table:5.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	4,226	20,480	20%	
Number of 4 input LUTs	6,979	20,480	34%	
Number of occupied Slices	4,712	10,240	46%	
Number of Slices containing only related logic	4,712	4,712	100%	
Number of Slices containing unrelated logic	0	4,712	0%	
Total Number of 4 input LUTs	7,057	20,480	34%	
Number used as logic	6,015			
Number used as a route-thru	78			
Number used as Shift registers	964			
Number of bonded IOBs	67	320	20%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number of FIFO16/RAMB16s	3	128	2%	
Number used as RAMB16s	3			
Number of DSP48s	4	128	3%	
Average Fanout of Non-Clock Nets	3.11			

RTL schematic of 256-point is shown in Fig 7.

Fig:7

Power Analysis report of 256-point FFT is shown in table 6.

Device	
Family	Virtex4
Part	xc4vsx25
Package	ff668
Temp Grade	Commercial
Process	Typical
Speed Grade	-12

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250

Characterization	
PRODUCTION	v1.0,02-02-08

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.053	1	---	---
Logic	0.000	7028	20480	34
Signals	0.000	9588	---	---
BRAMs	0.000	3	128	2
DSPs	0.000	4	128	3
DCMs	0.000	0	4	0
IOs	0.000	67	320	21
Leakage	0.332			
Total	0.385			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	8.7	81.7	53.3

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.189	0.044	0.145
Vccaux	2.500	0.062	0.000	0.062
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.385	0.053	0.332

5.Conclusions

In this paper, we proposed highly pipelined architecture for 64/128/256-point Fast Fourier Transform. The proposed architecture is implemented using Verilog coding on Xilinx environment. The proposed radix-8 FFT reduces arithmetic computations required to compute FFT comparing with the radix-2, radix-4 based FFT architectures. The proposed FFT is prototyped on FPGA device XC4SX25-12. The proposed FFT is designed at different clock frequencies with different data widths like 10-bit and 16-bit. The simulation results and power estimation results are shown in results chapter for different 64-point, 128-point and 256-point FFTs.

References

1. S. He, M. Torkelson, Designing pipeline FFT processor for OFDM (de) modulation, in: 1998 URSI International Symposium on Signals, Systems, and Electronics, Pisa, 1998. ISSSE 98, IEEE, 1998, pp.257–262.
2. C.-T.Lin,Y.-C.Yu,L.-D.Van,Alow-power64-pointFFT/IFFTdesignforIEEE 802.11 a WLAN application, in: 2006 IEEE International Symposium on Circuits and Systems, Island of Kos, IEEE, 2006, pp. 4.
3. S.-I. Cho, K.-M. Kang, A low-complexity 128-point mixed-radix FFT processor for MB-OFDM UWB systems, ETRI J. 32 (1) (2010)1–10.
4. S.-Y. Peng, K.-T. Shr, C.-M. Chen, Y.-H. Huang, Energy-efficient128~2048/1536-point FFT processor with resource block mapping for 3GPP-LTE system, in: 2010 International Conference on Green Circuits and Systems (ICGCS), Shanghai, IEEE, 2010, pp. 14–17.
5. R.M. Jiang, An area-efficient FFT architecture for OFDM digital video broad- casting, IEEE Trans. Consum. Electron. 53 (4) (2007) 1322–1326.
6. J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19 (90) (1965)297–301.
7. S. He, M. Torkelson, A new approach to pipeline FFT processor, in: Proceedings of IPPS'96, The 10th International Parallel Processing Symposium, 1996, Honolulu, HI, IEEE, 1996, pp.766–770.
8. Y. Jung, H. Yoon, J. Kim, New efficient FFT algorithm and pipeline implementation results for FDM/DMT applications, IEEE Trans. Consum. Electron. 49 (1) (2003)14–20.
9. O. Jung-Yeol, L. Myoung-Seob, New radix-2 to the 4th power pipeline FFT processor, IEICE Trans. Electron. 88 (8) (2005)1740–1746.
10. A. Cortés, I. Vélez, J.F. Sevillano, Radix FFTs: matricial representation and SDC/ SDF pipeline implementation, IEEE Trans. Signal Process. 57 (7)(2009)2824–2839 June 25.