Design and Optimization of 2048-Point FFTs in Altera FPGAs

[1]Kalyani B. Bhopi, [2]Dr. M. S. Panse

[1]Student of M.Tech. Electronics and Telecommunication, [2]Professor,
[1,2]Department of Electrical Engineering,
[1,2]Veermata Jijabai Technological Institute, Mumbai, India.

Abstract : Many areas of research, especially astrophysics, require high-speed logic circuits to perform real time signal processing. A common algorithm used in signal processing for such purpose is the Fast Fourier Transform (FFT). It is one of the vital operations used in the field of signal and image processing. Some of the important applications of FFT include spectrum analysis, image filtering, data compression, linear filtering of signals, etc. In this work we have focused on efficient implementation of FFT IP core available in Quartus Prime software for Intel FPGAs, in terms of reduced processing time or resource usage. It is more efficient than direct use of FFT IP core but with additional logic elements. The approach shows optimization of 2048-point FFT IP core implemented for Cyclone V FPGA and various supporting modules developed with the help of Verilog language.

Index Terms **– Fast Fourier Transform (FFT), Altera Cyclone V 5CGTFD9E5F35C7N FPGA, Verilog, Quartus Prime.**

I. INTRODUCTION

To understand the various facts about the universe, electromagnetic radiations coming towards the earth from different objects need to be examined. To analyze the frequency content of the radiations received from these objects we usually employ a tool known as Fourier Transform. The Fourier Transform is a mathematical tool that maps a signal from a time domain to an equivalent representation in terms of sinusoidal waveforms of varying frequencies and amplitudes. We can plot the amplitude versus frequency for each result obtained from the transform to obtain a visual representation of frequency content of the incoming signal [1] [5].

The Fourier Transform however is a time consuming computation. It is of interest to process the data in real time, rather than wait significant amount of time for processing old data. By using an algorithm known as the Fast Fourier Transform it is possible to implement a real time signal processing system [3]. The fast Fourier transform (FFT) algorithms are widely used in many fields, whether for spectral analysis to detect a signal, or for computation purposes (since for example the convolution of two signals can be computed efficiently using FFTs, and the convolution is the operation performed by finite impulse response filters), or for compression purposes or signal enhancements [6]. Now a days, many applications require the use of FPGAs as they are massively parallel structures containing a uniform array of configurable logic blocks (CLBs), memory, DSP slices and some other elements in contrast to traditional DSPs. They can be programmed using high-level hardware description languages like VHDL, Verilog and System Verilog, or at a block diagram level using System Generator. Moreover, many dedicated functions and IP cores are available for direct implementation in a highly optimized form within the FPGA. Therefore, computing FFTs in FPGA is the challenging task in many research areas. Like many other FPGA companies, Intel provides Intellectual Property (IP) core which computes FFTs of various transform length that is usually a power of two. These FFT IP cores are already designed to be highly optimized in their performance. However, it can be shown that more optimization of these FFT IP cores are possible using different algorithms.

In this work we focus on design and optimization of FFT IP core available in Quartus Prime software for Cyclone V FPGA devices, in terms of reduced processing time with moderate increase in resource usage. It is more efficient than direct use of readily available FFT IP core but with additional logic elements such as complex multipliers, adders and exponential generator like dual port memory or numerically controlled oscillator. The approach shows optimization of 2048-point FFT IP core implemented for Cyclone V FPGA with streaming I/O data flow, 16-bit input, output and twiddle factor widths and various supporting modules developed with the help of Verilog language.

II. FFT IP CORE

The discrete Fourier transform (DFT) of a sequence x_n of N points is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{j2\pi kn}{N}}, \quad (1)$$

with k=0, 1, … , N-1.

The implementation of an FFT algorithm on an FPGA using hardware description languages is not an easy task. Fortunately, FPGA companies usually provide an FFT IP core. The FFT IP core is a high performance, highly parameterizable Fast Fourier transform (FFT) processor. The FFT IP core implements a complex FFT or inverse FFT (IFFT) for high-performance applications [9].

The Altera FFT is highly configurable, for example we can select:

- The transform length of FFT.
- The input/output (I/O) data flow.
- The number of bits to quantify the input data, the output data, and the twiddle factors.

- The order of the input and of the output (natural order or bit-reversed order).
- Some options for the FFT engine.
- Some options for the implementation of the complex multipliers.

Regarding the I/O data flow, four are available: variable streaming, streaming, buffered burst, burst. For the variable streaming and the streaming I/O data flows, the input and output data flows can be continuous, without any break between consecutive transforms. The corresponding timing diagram of an N-point FFT (shown in Fig. 1) is given in Fig. 2. Between the last input sample and the first output sample of the FFT, there is a latency, denoted L_N, which depends on the transform length. Therefore, in this case, the P th FFT result is fully available after $N + L_N + PN = (P+1)N + L_N$ clock cycles.

Due to the large number of possibilities for the FFT implementation, for the evaluation of the resources in the following sections, we will consider the streaming I/O data flow, a data and twiddle precision of 16 bits, complex multipliers implemented in DSP blocks using four real multipliers, and no logic function implemented in memory.

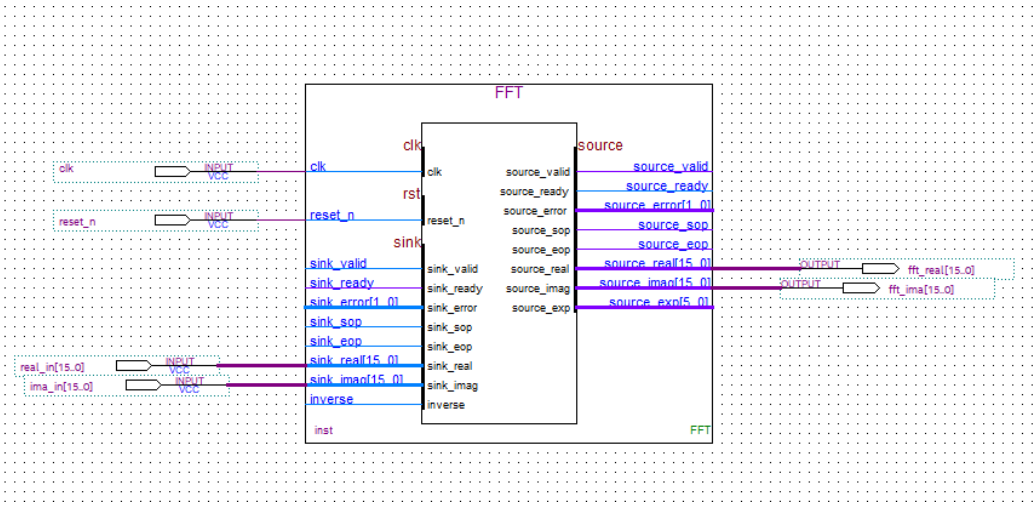


Fig. 1: Schematic/block diagram of FFT IP module in Quartus Prime

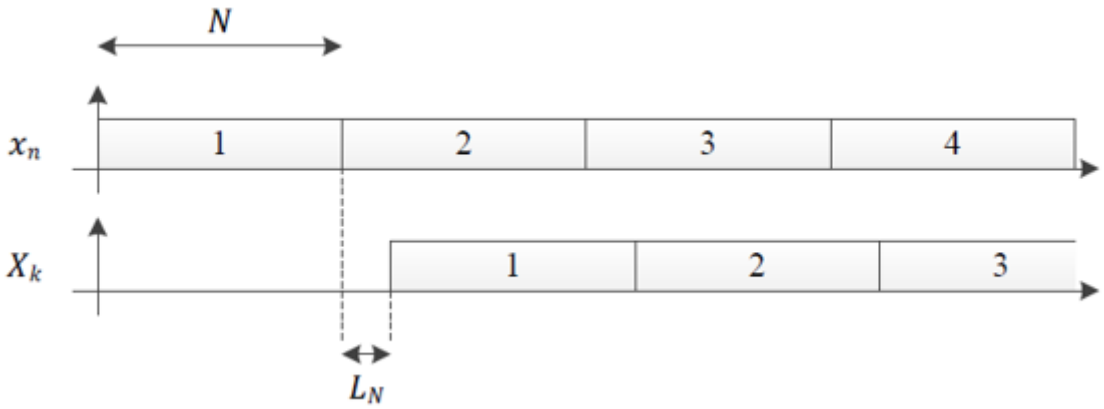


Fig. 2: Timing diagram of the direct implementation of FFT IP core with streaming I/O data flow

Under these conditions, the resources according to the FFT length are given in Table 1. It can be observed that doubling the FFT length doubles the processing time, increases slightly the logic (except for some cases), roughly doubles the memory (except for lengths lower than 512 points), and the number of DSP blocks stays the same. Seeing this, and knowing that an FFT can be computed as two FFTs of length halved (or using time multiplexing technique, only one FFT of length halved can be used), one can imagine that these characteristics can be exploited to improve the efficiency of the implementation, which is shown in the next section.

Table 1. Resource estimated with FFT IP core, considering the streaming I/O data flow and 16 bits for the data and twiddle precision

FFT length	Inverse of the throughput (cycle)	Logic usage (LE)	Memory usage (bit)	Multiplier usage (DSP blocks)
64	64	3578	10186	6
128	128	3366	19976	6

1024	1024	4128	78298	6
2048	2048	6544	311807	12
8192	8192	6628	1245747	12
32768	32768	6712	4981351	12
65536	65536	7029	9962087	12

III. RADIX-2 FFT ALGORITHM

The radix-2 FFT algorithm consists in separating the input or the output in even and odd samples, and repeats the process until a certain limit.

Doing so once for the input, we have

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi k(2n)}{N}} + \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi k(2n+1)}{N}} = \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} + e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}, \quad (2)$$

still with k=0, 1, … , N-1. The two sums does not correspond to DFTs because the range for k and n is different. The first half of X_k is obtained from above for k=0, 1, … , N/2-1, and is

$$X_k = \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} + e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}, \quad (3)$$

with k=0, 1, … , N/2-1 (in fact the equation is the same, only the range of k has changed). The two sums correspond to the DFT of the sequences x_{2n} and x_{2n+1} respectively. The second half of X_k is obtained from equation for k=N/2, N/2+1 , … , N-1, and can be expressed as

$$X_{k+N/2} = \sum_{n=0}^{N/2-1} x_{2n} e^{-\frac{j2\pi kn}{N/2}} - e^{-\frac{j2\pi k}{N}} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-\frac{j2\pi kn}{N/2}}, \quad (4)$$

with k=0, 1, … , N/2-1. The two sums are the same as for the previous equation.

1. Radix-2 FFT implementation in Quartus Prime

An FFT of N-points can be computed using two FFTs of N/2 points as explained and shown in Fig. 3.

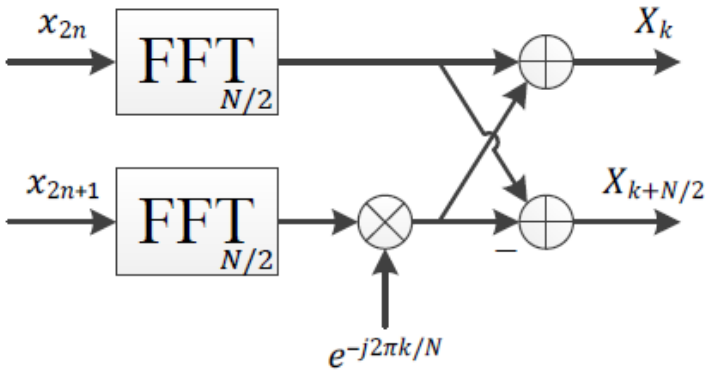


Fig. 3: N-Point FFT computation using two N/2-point FFTs

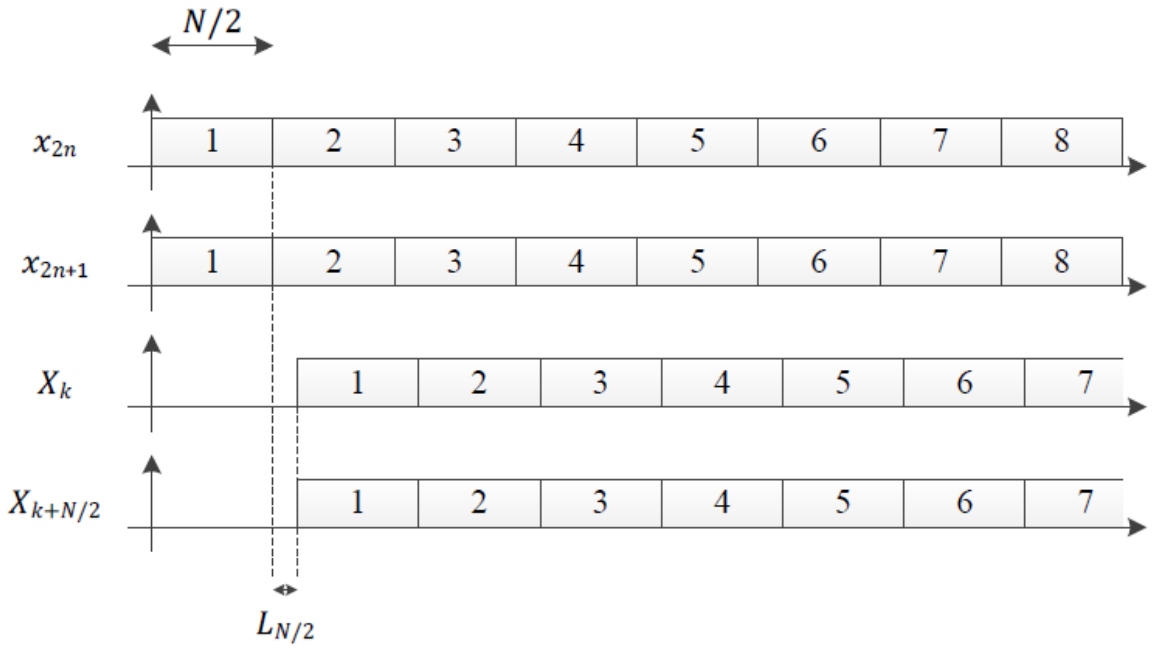
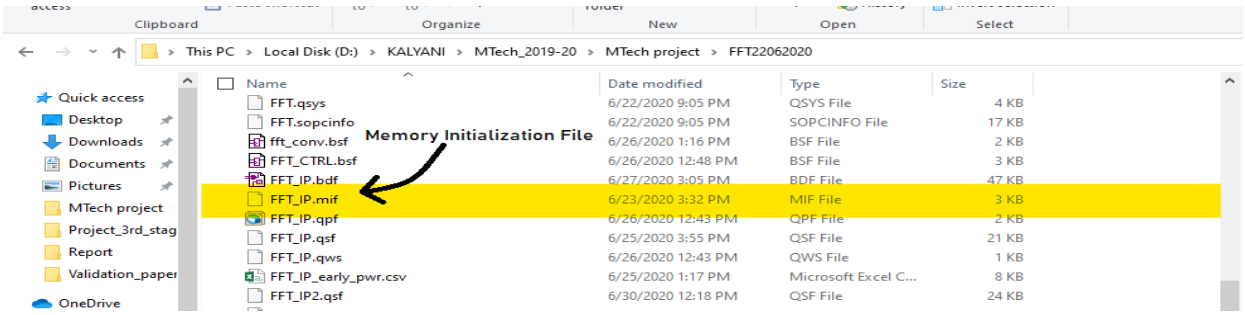


Fig. 4: Timing diagram of the radix-2 implementation of FFT IP core with streaming I/O data flow

In Fig. 4, $L_{N/2}$ denoted the latency of the FFT when the transform length is N/2. In this case, the P th FFT result is fully available after $N/2 + L_{N/2} + PN/2 = (P+1)N/2 + L_{N/2}$ clock cycles. Therefore, compared to the direct implementation of an N-point FFT, the processing time is approximately halved (see Fig. 2). Note however that this requires having access to even and odd samples of the input sequence simultaneously.

The complex exponential in Fig. 3 can be generated either using a numerically controlled oscillator (NCO), such as the NCO IP core provided by Altera, or using a memory. We have used a memory in this system. Only 512 samples of 16 bits are stored, since we need to generate half the period of a cosine and a sine, which can be obtained from only a quarter of the period of a sine wave by inverting the value and reading the memory in both directions. Thus, using a dual port memory would require only one M10K.



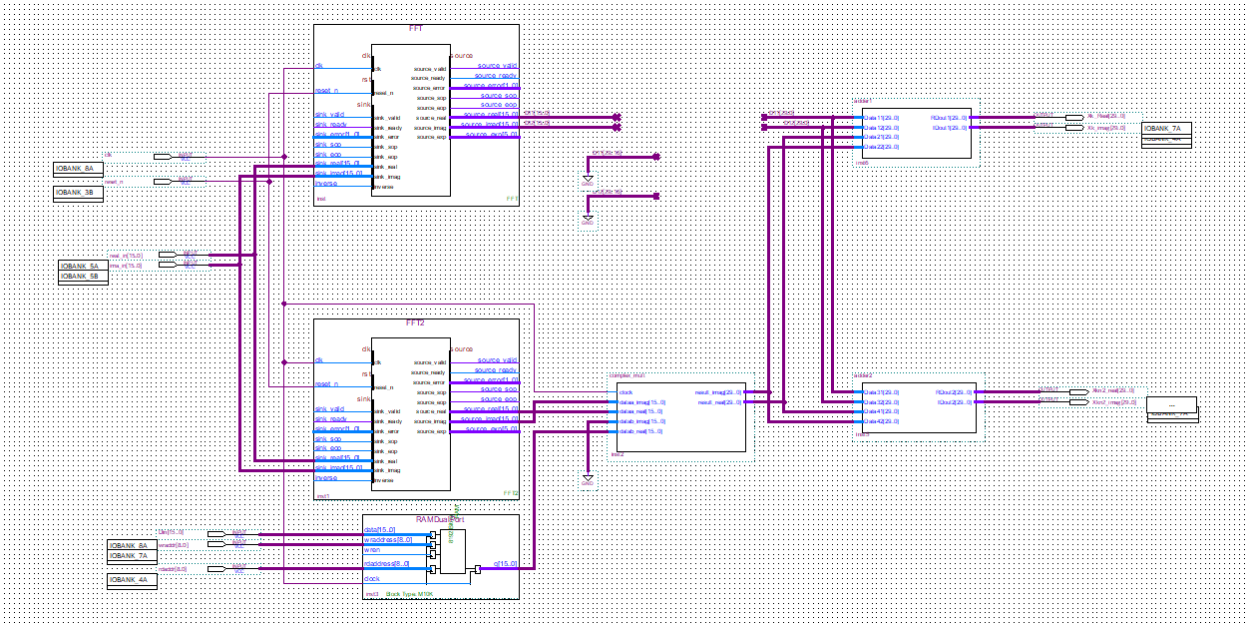


Fig. 5: Schematic/block diagram of two N/2 FFT IP modules in Quartus Prime

FFT IP core, included with Quartus Prime 18.1 software, is used in this work. Using this module, the spectrum of the input signal with frequency of 5 KHz is obtained. Avalon-ST buses are used to transfer data to the module and read processed data from it. The control signal generation module "FFT_CTRL" which is developed using VHDL is used to control the FFT module i.e. to provide control signals like SOP, EOP, etc. The modules process 1024 data samples each, the width of the input and output data is 16 bits with Twiddle factor width of 16 bits. The method of receiving reading and writing data to the module is streaming. The FFT - streaming mode of operation is selected due to lesser processing time and maximum throughput with moderate increase in resource usage.

In order to load data into the FFT module, the data source must wait for the FFT ready signal - sink_ready. After that, the source must transmit the SOP signal and data samples (the FFT has two data inputs - for the real and imaginary parts).

- At the end, the source transmits an EOP signal. The received data is stored in the module memory.
- Next, the FFT module starts the calculations. The duration of the calculation can be found when setting the FFT parameters - in the Transform Calculation Cycles column.
- After the calculations are completed, the module begins to wait for the data transfer from the receiver (in our case, it is SOPC), while the module checks the status of the source_ready line.
- After the receiver sets 1 on this line, the FFT module starts transmitting data. It also transmits 2048 pairs of data, and generates SOP, EOP, Ready signals, which are controlled by the receiver.
- When transmitting data to the receiver, first data on the positive region of the spectrum (N / 2 samples) is transmitted, then data on the negative region (the remaining N / 2 samples)
- The data is issued in the format of Block Floating Point.

IV. HARDWARE AND SOFTWARE

1. Intel Cyclone V FPGA

The Intel Cyclone V GT FPGA Development Kit can be used to prototype Cyclone V GT FPGA or Cyclone V GX FPGA applications. It offers a quick and simple way to develop low-cost and low-power FPGA system-level applications and achieve rapid results. The board provides a wide range of peripherals and memory interfaces to facilitate the development of Cyclone V designs.

1.1 Board Components

- One Cyclone V GT FPGA (5CGTFD9E5F35C7N) in a 1152-pin FineLine BGA (FBGA) package
- Clock Circuitry- 50-MHz, 100-MHz, and 125-MHz oscillators
- General user input/output- Push buttons, DIP switches
- User Eight user LEDs, One configuration load LED, One configuration done LED, One error LED

Table 2. Cyclone V GT features

Resource		**5CGTFD9E5F35C7N**
Logic Elements (K)		301
ALMs		113,560
Register		454,240
Memory (Kb)	M10K	12,200
	MLAB	1,717
18-bit×18-bit Multiplier		684
PLLs		8

2. Quartus Prime 18.1

The Quartus Prime design software is a multiplatform design environment that easily adapts to your specific needs in all phases of FPGA and CPLD design. Quartus Prime enables analysis and synthesis of HDL designs, which enables the developer to compile their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. Quartus Prime includes an implementation of VHDL and Verilog for hardware description, visual editing of logic circuits, and vector waveform simulation. Quartus Prime software delivers superior synthesis and placement and routing, resulting in compilation time advantages [9]. Compilation time reduction features include:

- Multiprocessor support
- Rapid Recompile
- Incremental compilation

V. RESULTS AND DISCUSSION

With Reference to Implemented system, following results have been acquired on simulator.

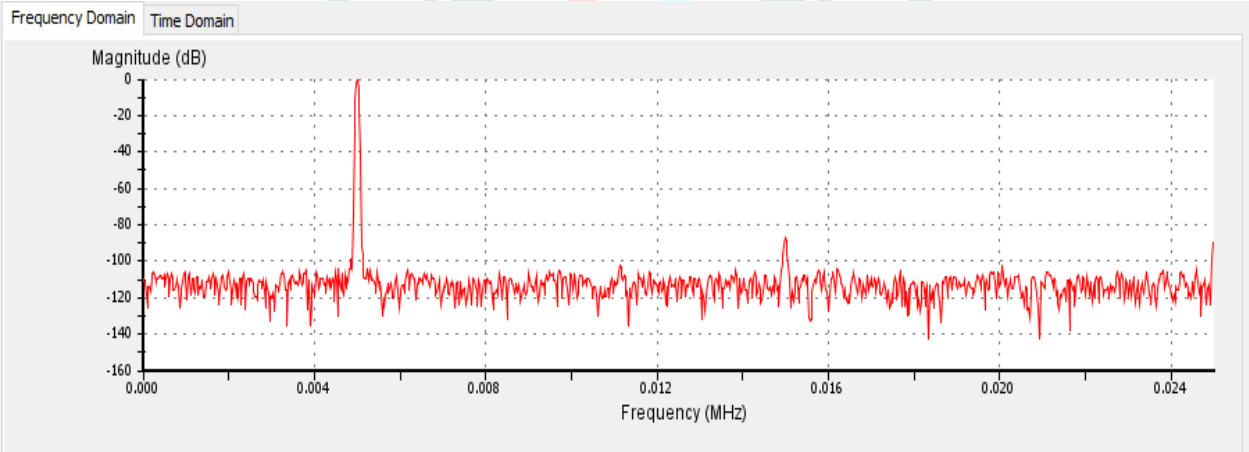


Fig. 6: Spectrum of NCO generated signal (Frequency = 5 KHz), Theoretical

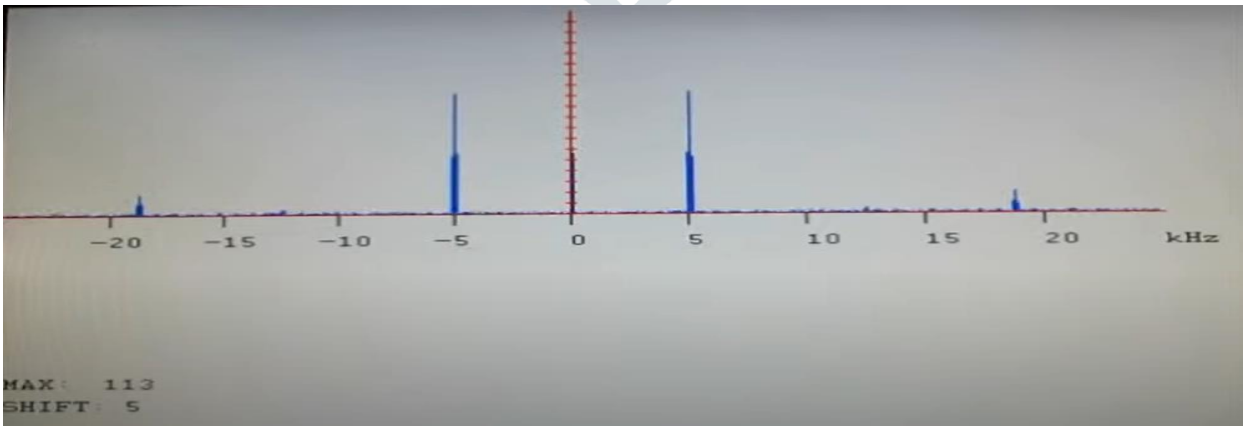


Fig. 7: Frequency spectrum (both sided) of NCO generated signal, Practical

Following figures show resource usage of optimized FFT module in terms of logic elements (LEs), number of I/O pins, memory and DSP blocks, comparison between two revisions of the same module based on fitting approach viz. balanced and power-aggressive respectively.

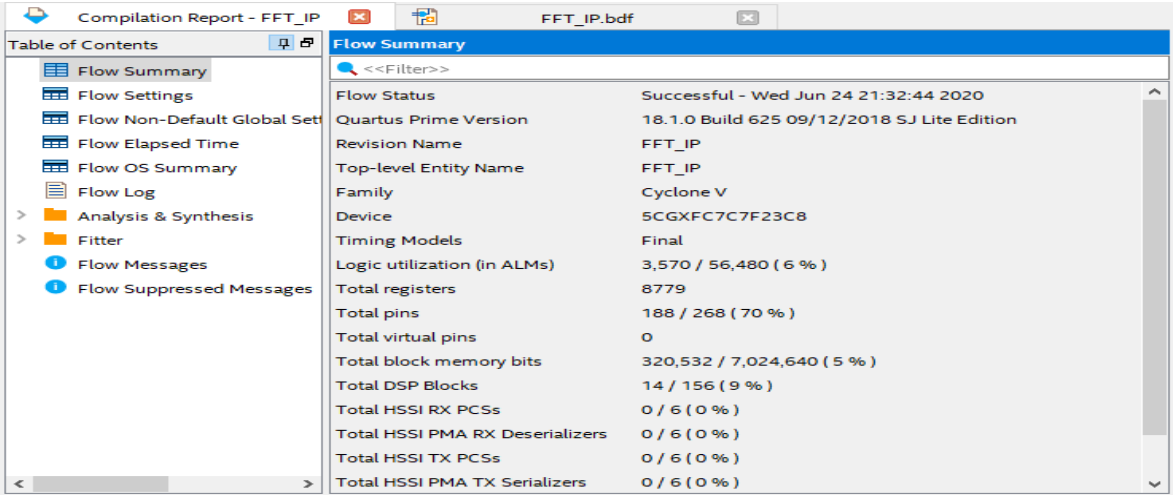


Fig. 8: Compilation report showing resource usage of optimized FFT module system

Fitter	FFT_IP	FFT_IP2
Fitter Status	Successful - Thu Jun 25 16:02:05 2020	Successful - Thu Jun 25 15:38:44 2020
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	FFT_IP	FFT_IP2
Top-level Entity Name	FFT_IP	FFT_IP
Family	Cyclone V	Cyclone V
Device	5CGXFC7C7F23C8	5CGXFC7C7F23C8
Timing Models	Final	Final
Logic utilization (in ALMs)	3,570 / 56,480 (6 %)	3,733 / 56,480 (7 %)
Total registers	8779	9074
Total pins	188 / 268 (70 %)	188 / 268 (70 %)
Total virtual pins	0	0
Total block memory bits	320,532 / 7,024,640 (5 %)	320,384 / 7,024,640 (5 %)
Total RAM Blocks	49 / 686 (7 %)	47 / 686 (7 %)
Total DSP Blocks	14 / 156 (9 %)	14 / 156 (9 %)
Total HSSI RX PCSs	0 / 6 (0 %)	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)	0 / 4 (0 %)

Fig. 9: Comparison of two revisions of FFT module which are optimized in terms of balanced and power-aggressive mode during fitting

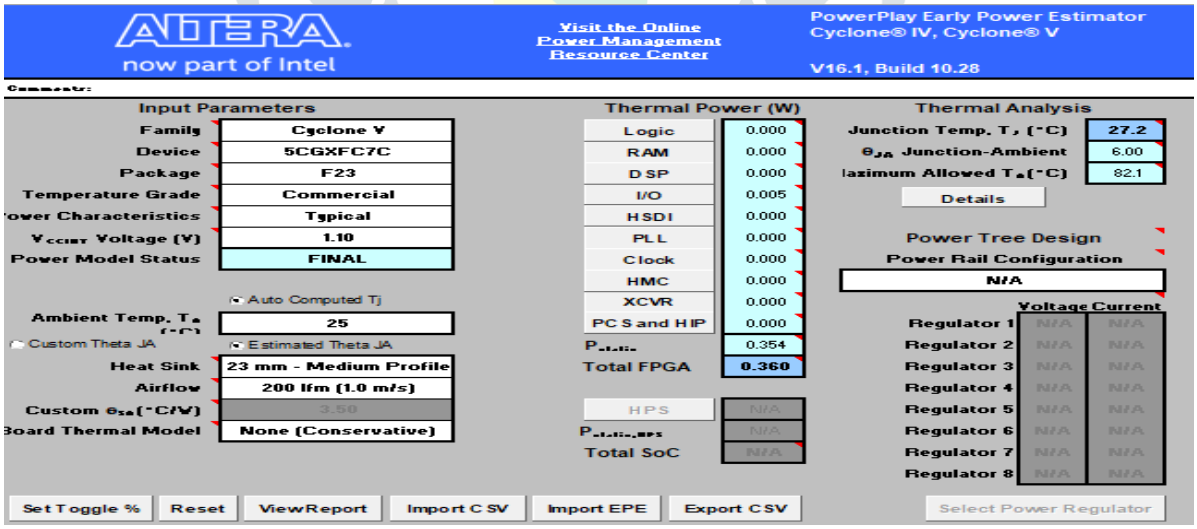


Fig. 10: Early power estimation of FFT module using Early Power Estimator (EPE.xls) tool

We have estimated the power consumption of Cyclone V FPGA using Early Power Estimator tool which shows that static power of the device is 0.354W and I/O consumes 0.005W making total power consumption of 0.360W. This is the estimation of power before fitting task. Once the fitting is done, accurate results of power consumption by the device can be obtained using Powerplay Power Analyzer tool as shown in Fig. 10.

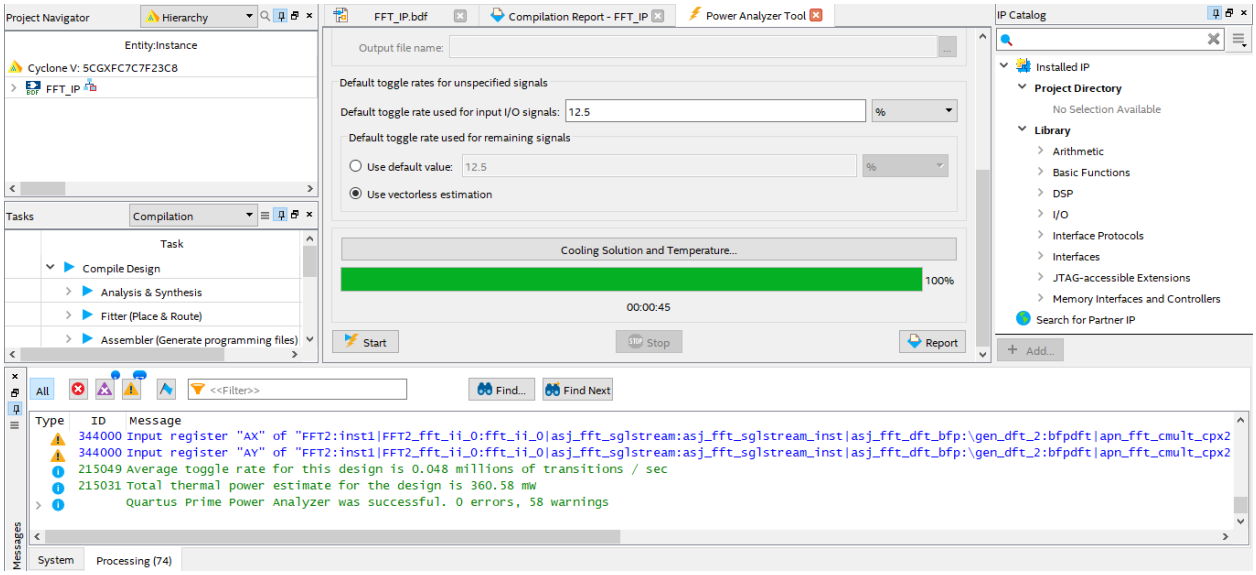


Fig. 11: Power estimation post-fitting of FFT module using Power Analyzer tool

Table 3. Comparison of the resources for Fig. 1 and Fig. 5 using FFT core with N=2048

Implementation	Function	Logic usage (LE)	Memory usage (bits)	Multipliers usage (DSP blocks)
Fig. 1	2 FFTs (1024 points) Exponential generation, 1 Multiplier, 2 Adders	8779	320532	14
Fig. 5	1 FFT (2048 points)	6544	311807	12
Difference between Fig. 1 and Fig. 5		+2235 (25.5%)	+8725 (2.7%)	+2 (14.2%)

It can be seen that the implementation of Fig. 1 requires more resources than the implementation of Fig. 5, especially regarding the logic usage while the memory usage is almost the same. However, as shown previously, the processing time for Fig. 1 is half the one for Fig. 5. Since the resources are increased by a factor less than two, the implementation of Fig. 1 is more efficient than the implementation of Fig. 5.

VI. Conclusion

Fast Fourier Transform (FFT) IP core thus have been successfully configured and tested for the incoming data. Optimization of the system in terms of processing time and power consumption has been achieved successfully. The experimental results show that the system is competent to be used for the real time astronomical applications.

VII. Acknowledgment

I would like to thank my guide Dr. M. S. Panse whose support and cooperation has been an invaluable asset during this entire stage. It would have been impossible to complete this paper without her support, valuable suggestions, criticism, encouragement and guidance.

References

[1] Leshem, JulianChristou, Brian D. Jeffs, Alle-Jan Van Der Veen. 2008. Introduction to the Issue on Signal Processing for Space Research and Astronomy. IEEE Journal Of Selected Topics In Signal Processing. 2(5).

[2] Hans Kanders, Tobias Mellqvist, Mario Garrido, Kent Palmkvist, Oscar Gustafsson. 2019. A 1 Million-Point FFT on a Single FPGA. IEEE Transactions on Circuits and Systems. 66(10).

[3] Trini Sansaloni, Asun Perez-Pascual, Vicente Torres, VicenÇ Almenar, JosÉ F. Toledo, Javier Valls. 2007. FFT Spectrum Analyzer Project for Teaching Digital Signal Processing With FPGA Devices. IEEE Transactions on Education. 50(3).

[4] Peng Wang, John McAllister. 2016. Streaming Elements for FPGA Signal and Image Processing Accelerators. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 24(6).

[5] Czajkowski, Tomasz & Comis, Christopher & Kawokgy, Mohamed & Rogers, Edward. 2004. Fast fourier transform implementation for high speed astrophysics applications on FPGAs.

[6] Jerome Leclère, Cyril Botteron & Pierre-André Farine. 2015. Implementing super-efficient FFTs in Altera FPGAs. EE Times Programmable Logic Designline.

[7] Asmita Haveliya. 2012. Design and Simulation of 32-point FFT using Radix-2 Algorithm for FPGA Implementation. Second International Conference on Advanced Computing and Communication Technologies.

[8] Quartus Prime Standard Edition Handbook Volume 1: Design and Synthesis, Chapter 16

[9] https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_fft.pdf

[10] https://en.wikipedia.org/wiki/Intel_Quartus_Prime