# Design and Analysis of Synthesizable RTL Verilog For Discrete Fourier Transformation Using FFT

**Apoorva K. S[1], Nataraj Urs H. D[2]**

[1]SchoolofECE, REVAUNIVERSITY, Bangalore
[2] School of ECE, REVA UNIVERSITY, Bangalore
[1]r19mve02@ece.reva.edu.in, [2]natarajurs.hd@reva.edu.in

**Article History:** Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

**ABSTRACT**: The use of FFT is very efficient in the field of digital signal processing and communication. It is one of the finest operations in the area of digital signal processing. Verilog implementation of floating point FFT with reduced generation logic is the proposed architecture, where the two inputs and outputs of any butterfly can be exchanged all data and addresses in FFT dispensation can be reordered. The Discrete Fourier Transform (DFT) can be implemented very fast using Fast Fourier Transform (FFT) It as the most important numerical algorithm of our lifetime. The Decimation- In-Time radix-2 FFT using butterfly structure has been designed with optimized area and power. The butterfly operation is faster. The outputs of the shorter transforms are reused to compute many outputs thus, the all-computational cost becomes less. The 8-bit input FFT is synthesized using Verilog. The simulation result and the implementation details such as design summary, RTL schematic and others can be noticed. This paper reports the area, delay and power analysis of 8-point FFT by developing a Verilog model in cadence tool kit. The implementation of fast algorithm for the DFT for evaluating their performance. The performance of this algorithm by implementing them on the cadence virtuoso digital design by developing our own FFT processor architecture.

**Keywords:** Hybrid Full Adder, fault tolerance, Self -Repairing Full Adder, multiplier.

## 1. Introduction

The fast Fourier transform (FFT) is a highly efficient method for computing the discrete Fourier transform (DFT) coefficients of a finite sequence of complex data. Fast Fourier Transform (FFT) is one of the most efficient to implement DFT. Reduced computational complexity and low latency are two driving factors for implementing DFT using FFT. Other FFT architectures that have been developed are based on radix – 2, radix – 4, hybrid, split radix.

Cooley and Tucky algorithm developed a very efficient algorithm to implement the discrete Fourier transform of a signal. This algorithm is called the Fast Fourier Transform (FFT). This FFT algorithm is very efficient in terms of computations of DFT. By using these algorithm, number of arithmetic operations involved in the computation of DFT is greatly reduced. According to the definition of DFT,

In addition to its wide use, properties of the FFT make it particularly attractive for hardware acceleration. Divide-and-conquer algorithms describe how large FFTs can be calculated using compositions of many smaller FFTs. These algorithms introduce the requirement of the composing FFT size factors the large FFT size. Although, applications tend to pick FFT sizes with this constraint in mind and thus restrict the FFT to numbers composed of small roots such as 2, 3, and 5. For analysis, smaller FFTs with undesirable factorizations can generally be approximated by using a larger FFT composed of a more desirable factorization. Thus, a general-purpose processor with an attached fixed-size FFT accelerator can still make good use of the accelerator, even if the initially desired application FFT is of different size than the accelerator.

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

k = 0,1,2,3.....N-1

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

n = 0,1,2,3.....N-1

## 2. Literature review

Akarshika Singhal, Anjana Goen and Tanu Trushna Mohapatrara proposed a method of Implementing of different schemes of FFT algorithms and applications received much attention in literature [1]. They developed a

model and hardware description of different schemes of FFT approaches including Cooley Tukey, Good-Thomas, Radix-2 and Rader methods by Verilog HDL and realization of them on Xilinx FPGA chip. Then the performance of different algorithms is compared for critical path time and chip area utilization. The accuracy in obtained results has been increased with the efficient coding in Verilog. The accuracy results depend upon the equations obtained from the butterfly diagram and then on the correct drawing of scheduling diagrams based on these equations.

Anup Tiwari, Samir Kumar PandeY [3] proposed a method of Implementation of Fast Fourier Transform Using Verilog HDL As the Fast Fourier Transform (FFT) is simply a professional technique to compute the Discrete Fourier Transform (DFT). Whilst memory based FFT processors need less hardware resource but require operating at higher clock frequency and with to meet the throughput. To achieve FFT calculation with a many point maximum number of samples the MACs requirement could not be matched by efficient hardware's like DSP. In this way, a fine solution is to use dedicated equipment processor to perform efficient FFT working out at high sample rate, while the DSP could perform the less concentrated parts of the processing. Verilog implementation of FFT with reduced generation logic is the proposed architecture, where the two inputs and outputs of any butterfly can be exchanged hence all addresses and data in FFT dispensation can be reordered. In this, A Verilog implementation of floating point FFT has been generated with reduced addressing logic using single precision floating point number IEEE 754 standard and improved the throughput of the system with respect to the speed in terms of high clock frequency. The proposed FFT algorithm is synthesized using vertex 6 as a target device. Synthesis is performed with Xilinx version 13. The synthesis results for a 16-point FFT with 64- bit complex number inputs show a maximum clock frequency of 463.6MHz compared to existing method.

## 3. Methodology

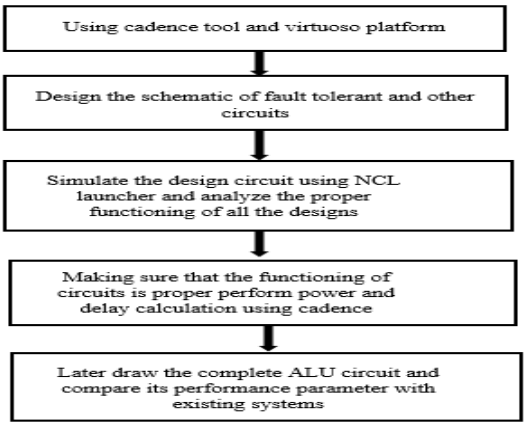



Fig 1: flowchart for the proposed concept

## 4. Proposed methodology

Functional Block Diagram of the Radix-2 8-point FFT the proposed system, which has a functional diagram as shown in figure 2, has divided into three stages- Input Stage, output Stage, and compute Stage. In Input Stage, eight samples are read form the Analog to Digital Converter (ADC) and are stored in 8*64-bit Input Buffer, which takes eight clock cycles. Compute stage performs the computation of FFT out of eight input samples and generates eight frequency samples in eight clock cycles. Compute stage has three blocks- x-Buffer to hold eight input samples from the input buffer, in FFT Core that computes the FFT, and temporary buffer that holds the intermediate results. Finally, Output Stage presents the output in the compute stage and the results are finally sent to the output port from the temp-buffer in output stage. each of the stage requires one clock-cycle to operate, it is straightforward that the core takes eight clock cycles to compute FFT and present the output frequency samples in the output ports.

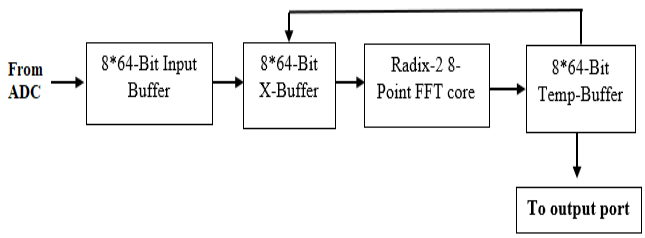



Fig 2: Proposed architecture of the radix-2 8-point FFT

### 5.1. Implementation method of Butterfly Network

The computation of the FFT is done by implementing the method of Butterfly signal in an efficient way. Whereas the direct implementation of the butterfly arrangement requires twelve subtracters, adders and multipliers, the FFT core uses four adders, subtracters and multipliers in order to conserve resources without sacrificing the performance of the signal. Had all twelve adders, multipliers, and subtracters been used then the output frequency samples would have been computed in one clock cycle since the entire design will be a single combinational circuit, the output is generated as soon as new input is available i.e., in one clock cycle.

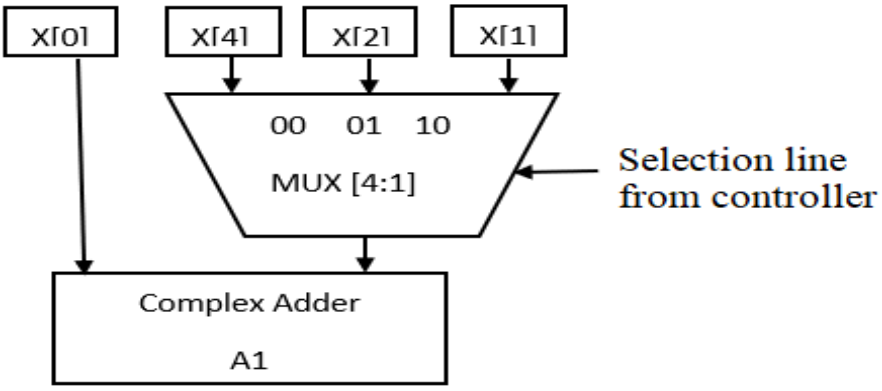



Fig 3: Illustration of an adder reused in various sub-stages of Compute Stage using 4- to-1 multiplexer

The input stage takes eight clock cycles independent of the architecture, since it will always take eight clock cycles to fetch eight input samples from ADC. Thus, in order to complete the computation of FFT before the next set of input samples are available from ADC, Compute Stage has at most seven clock cycles to compute the FFT and Output Stage has one clock cycle to host output samples in the output ports without introducing extra cycle consumption in the overall instruction cycle of the core. For this, the Compute Stage has been mathematically divided into three sub-stages each sub-stage requiring four adders, four subtracters and four multipliers as shown in figure. Here, for each of the three sub-stages of Compute Stage i.e., Compute Stage I, Compute Stage II, and Compute Stage III; the four adders, four subtracters, and four multipliers are reused by using the computational architecture as shown in fig 3 with the help of 4-to-1 multiplexers whose one input line is not used.

### 5.2. Verilog Design of the FFT Core

The Verilog module named fft8point whose block diagram is as shown in figure 4. The eight input samples are taken into the core using the input ports Px0 to Px7 each of which is 14-bit wide.

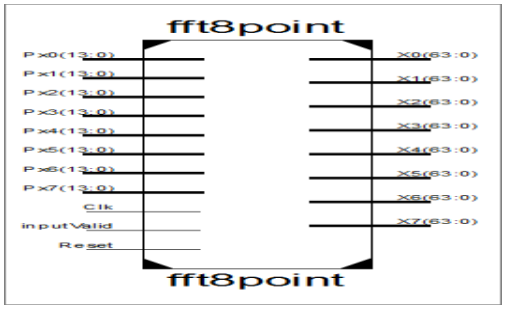



fig 4: Verilog module of the FFT core

The core then evaluates the FFT of the eight-time domain samples thus generating eight frequency samples. These frequency samples are presented in the eight output ports X0 to X7 each of which is 64-bit wide where the upper 32-bits is the real part and the lower 32-bits is the imaginary part of the output frequency samples.

## 5. Fft processor implementation

The algorithm used in the Verilog implementation was FFT radix-2 with decimation in time. The decimation in time means that before computing the transform the samples must be stored following the bit reversed order and read sequentially, otherwise the samples are read in the bit reversed order, this is defined in the DFT decomposition that will result in the FFT algorithm. Radix-2 means that the DFT will be decomposed in two parallel equations to make-up the FFT algorithm, using this decomposition the whole basic mathematical operations that will result in the transform are done using two samples at a time.

The FFT processor treats the data as fixed point. The number of bits reserved to the integer and decimal parts are input parameters of the FFT processor. To use this data representation the FFT processor uses the fixed point VHDL package which includes facilities for working with fixed point, such mathematical operations and fixed-point conversion routines. The number of points to the FFT can be either parametrized before compilation, but it must be a power of two less than 512. This limitation exists because it is necessary to store the complex exponentials values in the butterfly code. Currently are stored the complex exponentials values for 512 point and these numbers can be used to lower transforms. To describe the algorithm in VHDL has been adopted the internal organization of the FFT processor components shown in Fig. 5. The following subsections provides a brief description of the input-journals and internal blocks of Fig.

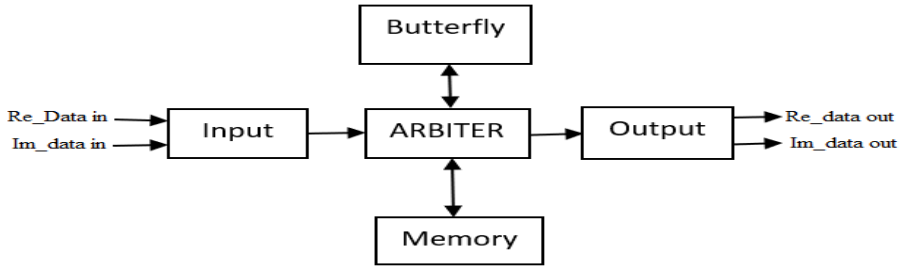



Fig 5: The internal organization of the FFT processor.

**6.1. Software Implementation**

The Cadence tool kit consist of several programs for different applications such as schematic drawing, layout, verification, and simulation. The technology used to build the circuit is gpdk 45 nano meter technology. These applications can be used on various computer platforms. This makes it possible to switch between different applications without having to convert the data base. The Cadence Virtuoso System Design Platform is a holistic, system-based solution that provides the functionality to drive simulation and LVS-clean layout of ICs and packages from a single schematic. There are two key flows: implementation and analysis. The implementation flow is used to create an IC package schematic in Virtuoso Schematic Editor and then transfer the schematic data to Cadence Sip Layout to layout the physical design.Virtuoso Specter simulator provides fast, accurate SPICE-level simulation for tough analog and mixed-signal circuits. It is tightly integrated with the Virtuoso custom design platform and provides detailed transistor-level analysis in multiple domains. Superior architecture allows for low memory consumption and high-capacity analysis.

Cadence for Digital IC Designthree types:

- Front-End
- Back-End
- Digital Design Flow

Front-End

Compile and Simulate: Use of NC-Verilog and Division to analyze, compile and simulate an example up-down counter

Synthesis: Convert the Verilog code into gate-level netlist using Cadence's Encounter™ RTL Compiler

Power estimation: TCF file generation and early power estimation of the design using SimVision and RTL Compiler.

Back-End

Physical Implementation: Floor planning to place and route of a test circuit using Cadence's Encounter Digital Implementation.

Digital Design Flow

Methodology for successful front-end design to back-end implementation of the chip at System on Chip (SoC) level. This involves using different tools from Synopsys and Cadence.

## 6. Verilog implementation of 8-point fft

The basic signal flow diagram for 8-point DIT based FFT is shown below. In this case, input data samples are out of order but the output samples are in order. The different twiddle factors are also mentioned in that diagram.

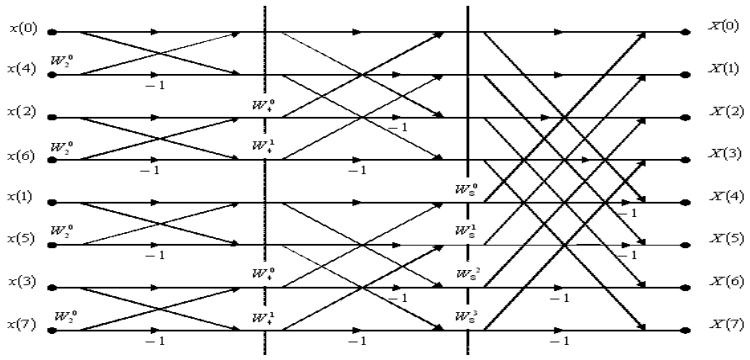


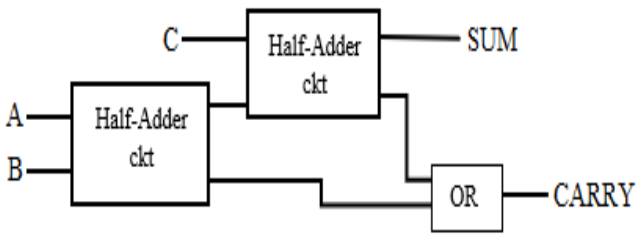

Fig 6: Signal Flow Diagram for 8-Point DIT Based FFT

The FFT processor to be designed should have a smaller number of input and outputs. The processor will have a clk pin, a reset pin and start pin as inputs. A pulse at the start pin will start the transform operation. The processor has two output vectors, one for real and one for imaginary part. The FFT processor is shown in Fig 6.The main aim of this paper is to implement a full adder utilizing a butterfly design. The architecture of the full adder is built using two half adders. In the context of fast Fourier transform algorithms, a butterfly is a part of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT. The name "butterfly" comes from the shape of the data-flow diagram in the radix-2 case and there by obtaining the simulation result using the cadence tool. The following steps are to be considered to build a full adder using butterfly design.

### 7.1 Implementation of A Half Adder

A half adder is a type of adder, an electronic circuit that performs the addition of numbers. The half adder is able to add two single binary digits and provide the output plus a carry value. It has two inputs, called A and B, and two outputs S (sum) and C (carry).2 Half Adders and a OR gate is required to implement a Full Adder. With this logic circuit, two bits can be added together, taking a carry from the next lower order of magnitude, and sending a carry to the next higher order of magnitude. The adder works by combining the operations of basic logic gates, with the simplest form using only a XOR and an AND gate. This can also be converted into a circuit that only has AND, OR and NOT gates.

Fig 7: Half Adder Block Diagram



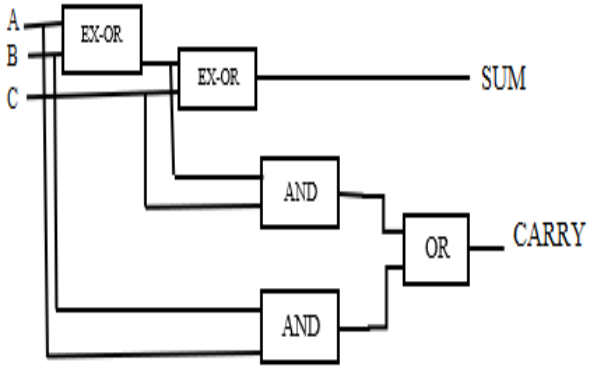

A half adder is used for adding together the two least significant digits in a binary, The four possible combinations of two binary digits A and B are shown in Figure7. The sum of the two digits is given for each of these combinations, and it will be noticed for the case A = 1 and = 1 that the sum is (10)2 where the 1 generated is the carry to the next stage of the addition. In the sum shown in Figure 12.1(a), a carry is generated in the least significant column and is then added in at the second stage where a further carry is generated. The carry has rippled through two stages of the addition.

### 7.2. Implementation of Full Adder

The full adder adds to input bits A and B plus a carry input bit and produces the sum and carry output bits as output. In classical control electronics the full adder has therefore three inputs and four outputs. There are various methods for implementing Full Adder which includes using only gate like AND, OR and NOT, moreover, full adder can also be implemented by using butterfly unit of multiplexer, adder and subtractor further using two Half adders. This is the basic model of the adder design to design different topologies. In the case of the radix-2 Cooley–Tukey algorithm, the butterfly is simply a DFT of size-2 that takes two inputs (P, Q) and gives two outputs (P^°,Q^° ).

Fig 8: Full Adder Block Diagram

### 7.3.Implementation of butterfly diagram




To implement the computation of butterfly with C54x instructions we have considered equations. If letting Y(k)=P, X(k)=P^° and X(k+N/2) =Q^° these two equations are,

P^°=P+W_N^k Q

Q^°=P-W_N^k Q

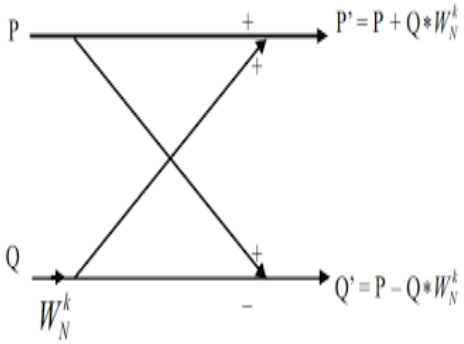



Fig 9: Radix 2 Butterfly Diagram

Since sequences P and Q and twiddle coefficient W are complex, we can divide these butterfly computations into two parts (real and imaginary). In practical computation, data of each part is stored in memory buffer. This is called in-place operation.

## 7. Result and design performance

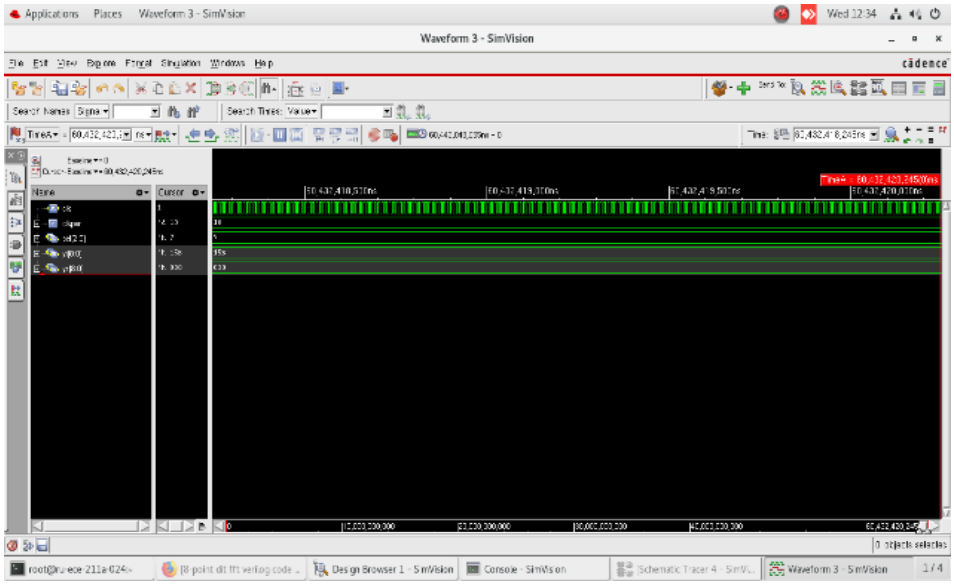



Fig 10: Simulation Result

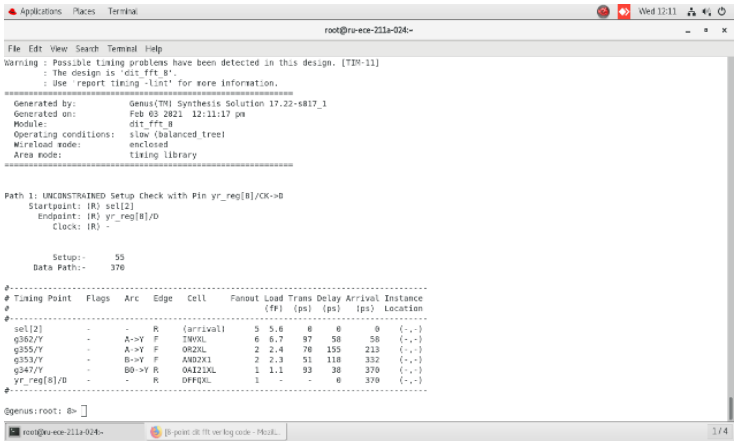


Fig 11: Power analysis

```
@genus:root: 9> report power
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Feb 03 2021  12:11:43 pm
  Module:                 dit_fft_8
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

                  Leakage   Dynamic    Total
 Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------
dit_fft_8    31   623.776 19586.683 20210.459

@genus:root: 10>
```

Fig 12: Delay analysis

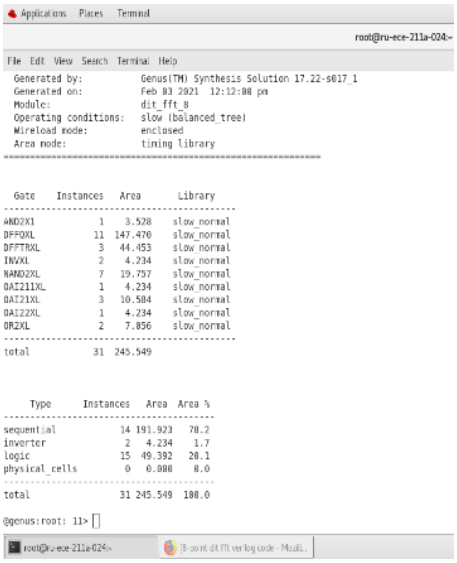



Fig 13: Area analysis

## 8. Conclusion

This paper presents 8-point FFT processor with a new architecture, which indeed has best possible performance with the optimization in resource consumption. The whole design is implemented in Verilog through cadence virtuoso tool and the functional verification is done by using SimVision simulator. Two circuits i.e., half adder and full adder thoroughly analyses the area, power and delay for the results. It was observed that, in cadence, results are based on probabilities for all the possible input combination at one go. Due to this there as an enormous reduction in the power, area and delay.

### References

1. Akarshika Singhal, Anjana Goen and Tanu Trushna Mohapatrara “Design and Implementation of Fast Fourier Transform (FFT) using VHDL Code” Department of Electronics and Communication, Rustamji Institute of Technology (august 2017).
2. S. Correa, Lilian C. Freitas, Aldebaro Klautau and Joao Crisostomo Costa “VHDL Implementation of a Flexible and Synthesizable FFT Processor”, LEA/LASSE, Federal University of Para – UFPA.
3. Anup Tiwari, Samir Kumar PandeY “Implementation of Fast Fourier Transform Using Verilog HDL”.
4. Mateus Beck Fonseca, João B. Martins and Eduardo Costa. “Design of pipelined butterflies from Radix-2 FFT with Decimation in Time algorithm using efficient adder compressor”
5. K. Sobaihi, A. Hammoudeh, D. Scammell: FPGA Implementation of OFDM Transceiver for a 60GHzWireless Mobile Radio System‖ International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp.185-189, 2010.
6. STOCHASTIC MODEL VIEW & FEEDBACK OF QUEUE NETWORK, Dr. Naveen Kumar, Mamta, International Journal Of Advance Research In Science And Engineering http://www.ijarse.com IJARSE, Volume No. 10, Issue No. 03, March 2021 ISSN-2319-8354(E).
7. Peter A. Milder, Franz Franchetti, James C. Hoe, and Markus Puschel, “Discrete Fourier Transform Compiler: From Mathematical Representation to Efficient Hardware” Electrical and Computer Engineering Department Carnegie Mellon University.
8. Remya Ramachandran et al. / International Journal of Computer Science Engineering “SIMULATION OF RADIX-2 FAST FOURIER TRANSFORM USING XILINX “.
9. Kala S, Mathew J, Jose BR, Nalesh S (2018) Radix-43 based two dimensional fft architecture with efficient data reordering scheme. IET Compute Digital Tech 13(2):78–86.