

A Project Report on

**DESIGN & IMPLEMENTATION OF IEEE 754 STANDARD
SINGLE PRECISION FLOATING POINT UNIT**

Submitted in partial fulfilment of the requirements of the award of

Degree

BACHELOR OF TECHNOLOGY (ELECTRONICS ENGINEERING)

Submitted By

Akash Arun Ambekar (PRN : 1819000423)

Mandar Vaijinath Kapare (PRN : 1617020009)

Megha Mahesh Shanbhag (PRN : 1819000472)

Ruturaj Ramakant Patil (PRN : 1819000459)

Under the guidance of

Dr. Sameer S. Nagtilak

Assistant Professor, Dept. of Electronics & Telecommunication Engineering



**DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
KOLHAPUR INSTITUTE OF TECHNOLOGY'S COLLEGE OF ENGINEERING
(AUTONOMOUS)**

2021-2022

A Project Report on

**DESIGN & IMPLEMENTATION OF IEEE 754 STANDARD
SINGLE PRECISION FLOATING POINT UNIT**

Submitted in partial fulfilment of the requirements of the award of

Degree

BACHELOR OF TECHNOLOGY (ELECTRONICS ENGINEERING)

Submitted By

Akash Arun Ambekar (PRN : 1819000423)

Mandar Vaijinath Kapare (PRN : 1617020009)

Megha Mahesh Shanbhag (PRN : 1819000472)

Ruturaj Ramakant Patil (PRN : 1819000459)

Under the guidance of

Dr. Sameer S. Nagtilak

Assistant Professor, Dept. of Electronics & Telecommunication Engineering



**DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING
KOLHAPUR INSTITUTE OF TECHNOLOGY'S COLLEGE OF ENGINEERING
(AUTONOMOUS)**

2021-2022



**KOLHAPUR INSTITUTE OF TECHNOLOGY's COLLEGE OF ENGINEERING
(AUTONOMOUS)**

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that, the project report entitled '**Design & Implementation of IEEE 754 Standard Single Precision Floating Point Unit**' submitted by:

Akash Ambekar, Mandar Kapare, Megha Shanbhag & Ruturaj Patil

for partial fulfillment for the award of the degree **BACHELOR OF TECHNOLOGY (ELECTRONICS ENGINEERING)** is a bonafide record of project work carried out by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

Dr. S.S. Nagtilak

Project Mentor
Dept. of EnTC Engg.
KIT's CoEK, Kolhapur

Dr. A.L. Renke

Project Coordinator
Dept. of EnTC Engg.
KIT's CoEK, Kolhapur

Dr. N.B. Sambre

Head
Dept. of EnTC Engg.
KIT's CoEK, Kolhapur

External Examiner

SEAL

Dr. M.M. Mujumdar

Name:

Director, KIT's CoEK, Kolhapur

DECLARATION

I declare that this report represents our ideas in our own words and where other's ideas or words included; I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Akash Arun Ambekar

Team Leader

ACKNOWLEDGEMENTS

I express my sincere and heartfelt gratitude towards my guide Prof. Mr. Sameer S. Nagtilak, Assistant Professor, Electronics Engineering Department, for his supervision, sympathy, and inspiration and above all help in all regards during the entire duration of my project without which, completion of the project was not possible at all. His guidance has been crucial for giving me a deep insight into the project. Also, A special gratitude we give to our project coordinator Prof. Mr. Amar L. Renke, whose contribution in stimulating suggestions and encouragement, helped us to co-ordinate our project as well as Prof. Mrs. Komal Jadhav, for solving each & every query regarding the VLSI domain. I would also like to thank all the professors of the department of Electronics Engineering, KIT's College of Engineering, Kolhapur for their constant motivation and guidance.

I am really thankful to all of my LinkedIn connections from various companies like Intel, Xilinx, ARM, Synopsys Inc., for suggesting such a wonderful project idea in RTL design & for being there for constant mentoring.

My sincere thanks to all my friends from 'VSD Tapeout Program' slack channel who has provided me with kind words, a welcome ear, new ideas, constructive criticism and their valuable time, I am truly indebted.

Akash Arun Ambekar

Team Leader

ABSTRACT

Whenever we talk about the mega projects submitted by students of Department of Electronics as well as EnTC Engineering from KIT's CoEK, we will find 99% of projects based IoT, Cloud Computing, Arduino etc..

The most underrated domain within KIT's CoEK for mega projects which has the infinite research & innovative possibilities is VLSI. Very Large Scale Integration (VLSI) is the technology of combining billions of transistors on a single small chip. So, considering the opportunity, we have done our mega project in VLSI domain.

The IEEE 754 is the efficient technical standard established in 1985 for diverse floating point implementation. 'A Floating Point Unit' (FPU) which is also known as math coprocessor, is specifically designed for performing floating point arithmetic operations like Addition, Subtraction, Multiplication and Division. The aim of the project was not only RTL design & implementation of the FPU using Verilog HDL, but also optimization of design.

Major three domains that any chip designer tries to optimize are: Time, Area & Power. Working on this project, our main focus was on minimization of timing constraints like maximum propagation delay, setup & hold time etc. Comparing with the time constraints from the international journals & papers that we have referred like IEEE, IRJET, IJSER etc, our design gives 75% reduction in maximum propagation delay, setup & hold time which leads to design optimization.

Along with time, we have also done the analysis of design with respect to area & power. Moreover, we have implemented the physical design steps like synthesis, floorplanning, placement, routing, STA etc. on this design with the help of Artix-7 XC7A100T FPGA which completes our project in wholesome manner

List of Contents

Topics	Page No.
List of Figures.....	v
List of Tables.....	vi
Abbreviations.....	vii

Chapter 1 Introduction.....01

1.1 Floating Point Unit.....	02
1.2 IEEE 754 Single Precision Standard.....	03
1.3 Literature Review.....	06
1.4 FPGA Design Flow.....	09

Chapter 2 Proposed Design Overview.....13

2.1 Design Goals & Objectives.....	14
2.2 Design Blocks.....	15
2.3 Proposed Methodology & Algorithm.....	17

Chapter 3 Design Simulation & Result Reports.....22

3.1 Code & Design Simulation.....	23
3.2 RTL Schematics.....	26
3.3 Synthesis Reports & Netlist.....	30
3.4 Placement & Routing Reports.....	33
3.5 Static Timing Analysis (STA).....	38
3.6 Power Reports.....	47
3.7 Design Summary.....	50

Chapter 4 Inferences & Conclusion.....51

4.1 Inferences.....	52
4.2 Conclusion.....	53
4.3 Bibliography.....	54

List of Figures

Figure 1	IEEE 754 Single Precision Standard.....	03
Figure 2	FPGA Design Flow	09
Figure 3	Block Diagram	15
Figure 4	Design Approach	17
Figure 5	Setup Timing Check.....	19
Figure 6	Hold Timing Check.....	20
Figure 7	Minimization in Setup & Hold Time	20
Figure 8	Design Properties	23
Figure 9	Simulation of Normal Case	24
Figure 10	Simulation of Infinity	24
Figure 11	Simulation of NaN	25
Figure 12	Simulation of 0/0 Case	25
Figure 13	RTL Schematics of adder module.....	26
Figure 14	RTL Schematics of multiplier module.....	27
Figure 15	RTL Schematics of divider module	28
Figure 16	RTL Schematics of top module	29
Figure 17	Netlist	32
Figure 18	Atrix7 XC7A100T FGPA Standard Cell.....	35
Figure 19	Routed Design.....	36
Figure 20	Clock Routing	37
Figure 21	DRC Check.....	37
Figure 22	STA of Routed Design	46
Figure 23	Power Report	47
Figure 24	Design Summary.....	50

List of Tables

Table 1	I/O Pin Description.....	15
Table 2	Opcode.....	16
Table 3	Status Register.....	16
Table 4	Module Name & Functionality	16

Abbreviations

ASIC	Application Specific integrated circuit
BJT	Bipolar Junction Transistor
CMOS	Complementary Metal Oxide Semiconductor
DUT	Design under test
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IC	Integrated Circuit
IEEE	Institute of Electrical & Electronics Engineers
NCD	Native Circuit Description
NGC	Native Generic Circuit
NGD	Native Generic Database
SDC	Synopsys Design Constraints
SoC	System On Chip
STA	Static Timing Analysis
UCF	User Constraint File
VHDL	Very High Speed Integrated Circuit HDL

CHAPTER 1

INTRODUCTION

1.1 FLOATING POINT UNIT

1.2 IEEE 754 SINGLE PRECISION STANDARD

1.3 LITERATURE REVIEW

1.4 FPGA DESIGN FLOW

1.1 FLOATING POINT UNIT

Floating-point unit (FPU) colloquially is a math coprocessor which is designed specially to carry out operations on floating point numbers [1]. Typically FPUs can handle operations like addition, subtraction, multiplication and division. FPUs can also perform various transcendental functions such as exponential or trigonometric calculations, though these are done with software library routines in most modern processors.

When a CPU executes a program that is calling for a floating-point (FP) operation, there are three ways by which it can carry out the operation. Firstly, it may call a floating-point unit emulator, which is a floating-point library, using a series of simple fixed-point arithmetic operations which can run on the integer ALU. These emulators can save the added hardware cost of a FPU but are significantly slow. Secondly, it may use an add-on FPUs that are entirely separate from the CPU, and are typically sold as an optional add-ons which are purchased only when they are needed to speed up math-intensive operations. Else it may use integrated FPU present in the system [2].

The FPU designed by us is a single precision IEEE754 compliant integrated unit. It can handle not only basic floating point operations like addition, subtraction, multiplication and division but can also handle operations like shifting, square root determination and other transcendental functions like sine, cosine and tangential function.

1.2 IEEE 754 SINGLE PRECISION STANDARD

IEEE754 standard is a technical standard established by IEEE and the most widely used standard for floating-point computation, followed by many hardware (CPU and FPU) and software implementations. Single-precision floating-point format is a computer number format that occupies 32 bits in a computer memory and represents a wide dynamic range of values by using a floating point. In IEEE 754-2008, the 32-bit with base 2 format is officially referred to as single precision or binary32. It was called single in IEEE 754-1985. The IEEE 754 standard specifies a single precision number as having sign bit which is of 1 bit length, an exponent of width 8 bits and a significant precision of 24 bits out of which 23 bits are explicitly stored and 1 bit is implicit 1.

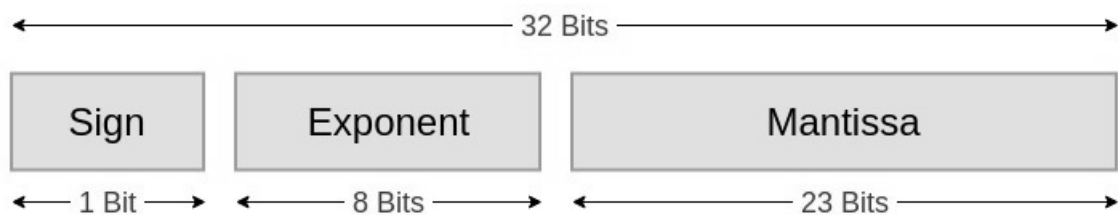


Fig. 1 : IEEE 754 Single Precision Standard

Sign bit determines the sign of the number where 0 denotes a positive number and 1 denotes a negative number. It is the sign of the mantissa as well. Exponent is an 8 bit signed integer from -128 to 127 (2's Complement) or can be an 8 bit unsigned integer from 0 to 255 which is the accepted biased form in IEEE 754 single precision definition. In this case an exponent with value 127 represents actual zero. The true mantissa includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros. Thus only 23 fraction bits of the mantissa appear in the memory format but the total precision is 24 bits.

IEEE754 also defines certain formats which are a set of representation of numerical values and symbols. It may also include how the sets are encoded.

The standard defines:

- **Arithmetic formats** which are sets of binary and decimal floating-point numbers, which consists of finite numbers including subnormal number and signed zero, a special value called "Not a Number" (NaN) and infinity.
- **Interchange formats** which are bit strings (encodings) that are used to exchange a floating-point data in a compact and efficient form.
- **Rounding rules** which are the properties that should be satisfied while doing arithmetic operations and conversions of any numbers on arithmetic formats.
- **Exception handling** which indicates any exceptional conditions (like division by zero, underflow, overflow, etc.) occurred during the operations.

The standard defines the following five rounding rules:

- Round to the nearest even which rounds to the nearest value with an even (zero) least significant bit.
- Round to the nearest odd which rounds to the nearest value above (for positive numbers) or below (for negative numbers)
- Round towards positive infinity which is a rounding directly towards a positive infinity and it is also called rounding up or ceiling.
- Round towards negative infinity which is rounding directly towards a negative infinity and it is also called rounding down or floor or truncation.

The standard also defines five exceptions, and all of them return a default value. They all have a corresponding status flag which are raised when any exception occurs, except in certain cases of underflow. The five possible exceptions are:

- **Invalid** operation are like square root of a negative number, returning of NaN by default, etc., output of which does not exist.
- **Division by zero** is an operation on a finite operand which gives an exact infinite result for e.g., $1/0$ or $\log(0)$ that returns positive or negative infinity by default.
- **Overflow** occurs when an operation results a very large number that can't be represented correctly i.e. which returns $\pm\text{infinity}$ by default (for round-to-nearestmode).
- **Underflow** occurs when an operation results very small i.e. outside the normal range and inexact (denormalised value) by default.
- **Inexact** occurs when any operation returns correctly rounded result by default.

1.3 LITERATURE REVIEW

Open Floating Point Unit, the open source project done by Rudolf Usselmann. His FPU described a single precision floating point unit which could perform add, subtract, multiply, divide, and conversion between FP number and integer. It consists of two pre-normalization units that can adjust the mantissa as well as the exponents of the given numbers. One unit is for addition and subtraction operation and the other one is for multiplication and division operations. It also has different units for different operations that perform an actual addition subtraction, multiplication and division. It also has a shared post normalization unit that normalizes the fraction part. The final result after post-normalization is directed to a valid result which is in accordance to single precision FP format. The main drawback of this model was that most of the codes were written in MATLAB and due to this it is non-synthesizable.

The high Performance IEEE754 FPU was designed at Gaisler Research for the improvement of FP operations of a LEON based systems. It supports both single precision and double precision operands. It implements all floating point arithmetic operations defined by the IEEE754 standard in hardware. All operations are dealt with the exception of denormalized numbers which are flushed to zero and supports all rounding modes. This advanced design combines low latency and high throughput. The most common operations such as addition, subtraction and multiplication are fully pipelined which has throughput of one CC and a latency of three CC. More complex divide and square root operation takes between 1 to 24 CC to complete and execute in parallel with other FP operations. It can also perform operations like converse and compliment. It supports all SPARC V8 FP instructions. The main drawback of this model is that it is very expensive and complex to implement practically.

Kodali, R.K., Gundabathula, S.K. and Boppana, L investigated the floating point arithmetic, specifically multiplication, is a widely used computational operation in

many scientific and signal processing applications. In general, the IEEE-754 single-precision multiplier requires a 23×23 mantissa multiplication and the double-precision multiplier requires a large 52×52 mantissa multiplier to obtain the final result. This computation exists as a limit on both area and performance bounds of this operation. A lot of multiplication algorithms have been developed during the past decades. In this research, the two of the popular algorithms, namely, Booth and Karatsuba (Normal and Recursive) multipliers have been implemented, and a performance comparison is also made. The algorithms have been implemented on an uniform reconfigurable FPGA platform providing a comparison of FPGA resources utilized and execution speeds. The recursive Karatsuba is the best performing algorithm among the algorithms.

Hang Zhang, Wei Zhang, Lach, J. proposed a low-power accuracy-configurable floating point multiplier based on Mitchell's Algorithm. Post-layout SPICE simulations in a 45nm process show same-delay power reductions up to 26X for single precision and 49X for double precision compared to their IEEE-754 counterparts. Functional simulations on six CPU and GPU benchmarks show significantly better power reduction vs. quality degradation trade-offs than existing bit truncation schemes.

Ramesh, A.P., Tilak, A.V.N. and Prasad, A.M. researched on the Floating Point multiplication is widely used in large set of scientific and signal processing computation. Multiplication is one of the common arithmetic operations in these computations. A high speed floating point double precision multiplier is implemented on a Virtex-6 FPGA. In addition, the proposed design is compliant with IEEE-754 format and handles over flow, under flow, rounding and various exception conditions. The design achieved the operating frequency of 414.714 MHz with an area of 648 slices.

Sheikh, B.R., Manohar and R. presented the details of our energy-efficient asynchronous floating-point multiplier (FPM). Authors discussed design trade-offs of various multiplier implementations. A higher radix array multiplier design with

operand-dependent carry-propagation adder and low handshake overhead pipeline design is presented, which yields significant energy savings while preserving the average throughput.

Su Bo, Wang Zhiying and Huang Libo expanded the large computing platforms and the increasing of on chip transistors; power consumption becomes a significant problem. Many methods are proposed in different design levels to solve the problem. In this paper, researchers introduce asynchronous technique to an IEEE-754 double precision floating-point multiplier aiming to reduce its power consumption. The control path of the asynchronous multiplier employs redundant four-phase latch controllers and asymmetric delay elements. Experimental results show the power consumption of the asynchronous multiplier is at least 16% lower than its synchronous equivalent when running the PARSEC benchmarks.

Lots of works have been proposed for FPU models, and all suggests some improvements in certain levels at various stages. An island-style with embedded FPU is proposed by Beauchamp M. J. Beauchamp, S. Hauck, and K. S. Hemmert, and it gives a delay improvement than standard implementation for floating point applications. A coarse grained FPUs were suggested by Ho et al., and they presents a Hybrid architecture, where word blocks were used for computing simple operations. This also suggests improvement in delay as whole complex operations were moved to the coarse grained to reduce routing delays.

In all these works done suggested the delay improvements and area reductions. In our design, we suggest an efficient model for performing the floating point operations by reducing the operation complexities, time constraints, area etc.

1.4 FPGA DESIGN FLOW

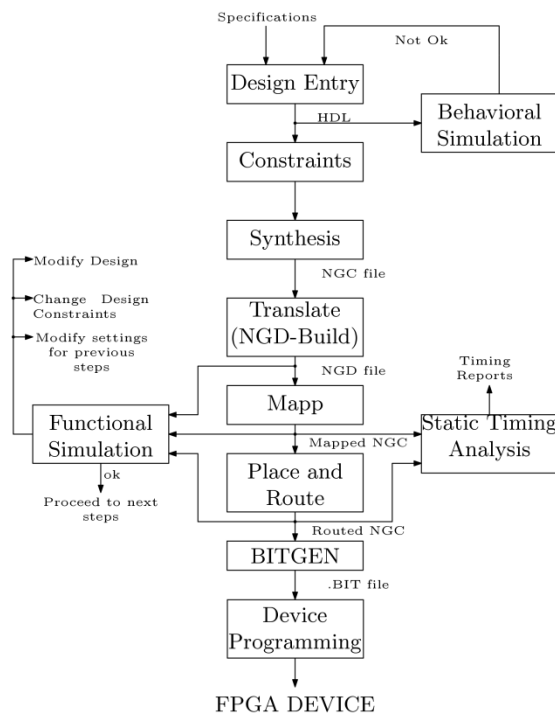


Fig. 2: FPGA Design Flow

Design Entry

There are different techniques for design entry.

- Schematic based,
- Hardware Description Language (Verilog, SystemVerilog, VHDL)
- Combination of both.

Selection of a method depends on the design and the designer. If the designer wants to deal more with hardware, then Schematic entry is a better choice. When the design is complex or the designer wants to realize the design in an algorithmic way then HDL is the better choice. Language based entry is faster but lag in performance and density. HDLs represent a level of abstraction that can isolate the designers from the details of the hardware implementation. Schematic based entry gives designers much more visibility into the hardware. Design entry using the structural style is combination of both schematic and HDL. We preferred design

entry by Verilog HDL in structural style to understand hardware details without much complexity.

Synthesis

Synthesis is the process which translates HDL code into a device netlist format i.e. a complete circuit with logical elements (gates, flip flops, etc.) for the design. If the design contains more than one sub designs, then the synthesis process generates netlist for each design element. Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist is saved to an *NGC (Native Generic Circuit)* file. The synthesis step gives an estimate of the hardware utilization. More actual resource utilization can be found after MAP process.

Implementation

This process consists of a sequence of three steps:

1. **Translate process** – Translate process combines all the input netlists and constraints to a logic design file. This information is saved as a *NGD (Native Generic Database)* file. This can be done using NGD Build program. Here, defining constraints is nothing but, assigning the ports in the design to the physical elements (ex. pins, switches, buttons etc) of the targeted device and specifying time requirements of the design. This information is stored in a file named *UCF (User Constraints File)*. Tools used to create or modify the UCF are PACE, Constraint Editor etc.
2. **Map process** – MAP process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (IOB)) and generates an *NCD (Native Circuit Description)* file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose. A PCF

(Physical Constraints File) is also generated containing all the constraint related information.

3. **Place and Route** – PAR program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Ex. if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may affect some other constraint. So a tradeoff between all the constraints is taken account by the place and route process. The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consists of the routing information.

The hardware utilization summary after synthesis and MAP process may not match. The MAP process runs several optimization algorithms which remove or trims irrelevant, duplicate and unused logic elements. The maximum frequency achieved after synthesis process and after MAP process can be matched for smaller designs. But for complex designs generally the frequency achieved after MAP process is lower than the frequency achieved after synthesis process. The maximum frequency depends on the total delay time which can be expressed as:

$$\text{Total Delay} = \text{Data Path Delay} + \text{Clock Path Skew} + \text{Clock Uncertainty} + \text{Routing Path Delay}$$

Device Programming

Now the design must be loaded on the FPGA. But the design must be converted to a format so that the FPGA can accept it. BITGEN program deals with the conversion. The routed NCD file is then given to the BITGEN program to generate a *bit stream* (*.BIT file*) which can be used to configure the target FPGA device. This can be done using a cable. Selection of cable depends on the design.

Design Verification

Verification can be done at different stages of the process steps.

- **Behavioural Simulation (RTL Simulation):** This is the first of all simulation steps; those are encountered throughout the hierarchy of the

design flow. This simulation is performed before synthesis process to verify RTL (behavioural) code and to confirm that the design is functioning as intended. Behavioural simulation can be performed on either VHDL or Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is a very fast simulation and so allows the designer to change the HDL code if the required functionality is not met within a short time period. Since the design is not yet synthesized to gate level, timing and resource usage properties are still unknown.

- **Functional simulation (Post Translate Simulation):** Functional simulation gives information about the logic operation of the circuit. Designer can verify the functionality of the design using this process after the Translate process. If the functionality is not as expected, then the designer has to make changes in the code and again follow the design flow steps.
- **Functional simulation (Post PAR Simulation):** Some time functional simulation works but design doesn't produce the expected result when targeted to a FPGA device. This is due to the fact of rigorous optimization in the PAR process and failing to meet the constraints. Post PAR simulation can be run to check the functionality after PAR. It is expected that if the post PAR simulation is ok then the design can be successfully implemented.
- **Static Timing Analysis:** This can be done after MAP or PAR processes. Post MAP timing report lists signal path delays of the design derived from the design logic. Post Place and Route timing report incorporates timing delay information to provide a comprehensive timing summary of the design.

CHAPTER 2

PROPOSED DESIGN OVERVIEW

2.1 DESIGN GOALS & OBJECTIVES

2.2 DESIGN BLOCKS

2.3 PROPOSED METHODOLOGY & ALGORITHM

2.1 DESIGN GOALS & OBJECTIVES

Problem Statement:

Design an optimized IEEE 754 standard single precision floating point unit with FPGA design flow which leads to minimization of time constraints of the design like minimum clock period (delay), setup time and hold time as well as minimization in resource utilization.

Objectives:

- To design a clock synchronous sequential circuit
- To minimum clock period (delay), setup time and hold time
- To increase the reusability of the code
- To handle exceptions like NaN, Infinity and Zero
- To minimize resource utilization
- Implementation of FPGA design flow for the entire floating point unit

2.2 DESIGN BLOCKS

Our Floating Point Unit is a single precision IEEE754 compliant integrated unit. It incorporates various basic operations like addition, subtraction, multiplication and division. The basic block of our design is shown as:

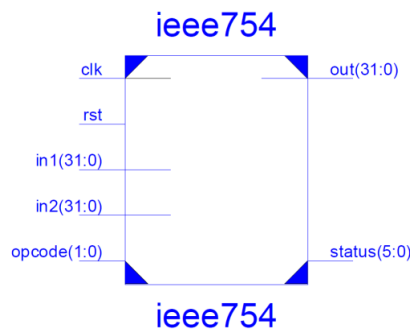


Fig. 3: Block diagram

I/O Pin Description:

Signal Name	Width	Type	Description
clk	1 bit	Input	System Clock
rst	1 bit	Input	Reset values for initializing
in1	32 bit	Input	Operand 1
in2	32 bit	Input	Operand 2
out	32 bit	Output	Result of the selected operation in IEEE format
status	6 bit	Output	Status of operation

Table 1: I/O Pin Description

The block is in synchronization with clock input. The two inputs are denoted by in1 & in2 which are in 32-bit IEEE 754 standard format. The operation to be performed is specified by opcode input as:

opcode[1:0]	Operation
00	Addition
01	Subtraction
10	Multiplication
11	Division

Table 2 : Opcode

The output is also available in 32-bit IEEE 754 standard format. Also, the output status register is used to indicate the valid operation as well as the exceptions like NaN, Infinity, Zero etc. indicated as:

5	4	3	2	1	0
NaN	Infinity	Zero	Subnormal	Normal	Valid

Table 3 : Status Register

The floating point unit is fully IEEE 754 compliant. The design implemented here incorporates the following modules. Both the module name and its functionality have been specified here in sequence of the manner they appear in the code:

Module Name	Functionality
ieee754.v	Main top module
adder.v	Performs addition
adder_normaliser.v	Normalizes the result of add/sub operation to its IEEE754 form
multiplier.v	Performs multiplication
mul_normaliser.v	Normalizes the result of multiplication to its IEEE754 form
divider.v	Performs division
reciprocal.v	Finds reciprocal of a number

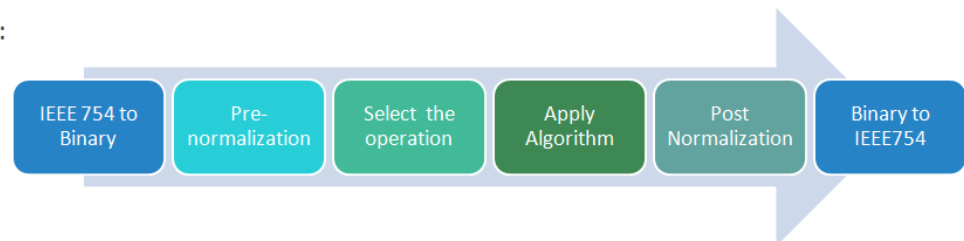
Table 4 : Module Name & Functionality

2.3 PROPOSED METHODOLOGY & ALGORITHM

DESIGN APPROACH:

We have gone through various international journals, magazines and papers like IEEE, IJRET, IJSEER etc. In majority of papers, we found two approaches to design a FPU.

- Approach 1 :



- Approach 2 :

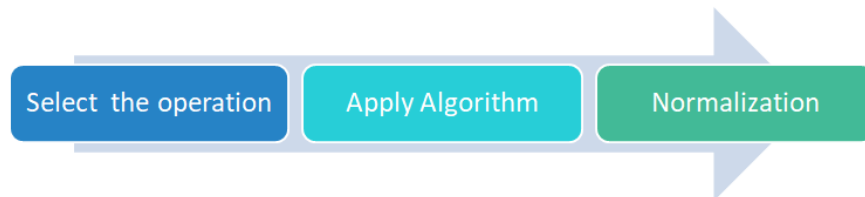


Fig. 4: Design Approach

Talking about 1st approach of design, for performing various arithmetic operations, they have used high performance algorithms. For example, block carry look ahead adder approach for addition & subtraction, Karatsuba algorithm for multiplication, Non-restoring division algorithm for division. But, the biggest disadvantage of following this approach is, here we first convert input IEEE754 numbers to normal binary floating point numbers, apply the suitable algorithm for arithmetic operations, calculate the result and again convert it into IEEE754 format. Following this approach, the IEEE754 standard format become meaningless. Secondly, all of these highly efficient algorithms are based on integer based calculations, there is no facility given for radix point. Hence the primary task is to decide the location of radix point i.e. how many bits we are reserving after radix point for representation of floating point number. Ultimately the system is not going to know the radix point location; it's just for the sake of our understanding.

Suppose after conversion from IEEE754 format to binary format which is the first step of these design, we reserve 2 bits after point for representation of floating point number, then we can represent 4 floating point numbers which are .00, .01, .10, .11 in binary i.e. .00, .25, .50, .75 in decimal. The next task will be padding the zeros in order to bring the equal radix point of both operands. Next step, if there is the addition or subtraction operation, then the radix point of result won't change. If there is the multiplication operation, then the radix point of result jumps twice the number of bits reserved for floating point towards left (E.g. $1.25 \times 1.25 = 1.5625$) ; if we keep the radix point fixed the 1.56 will be calculated and substantially accuracy and precision gets compromised. For division, this task becomes most difficult.

Summing up all, following this design approach are high speed algorithms, various extra design modules are added in the design which are : two converter blocks i.e. IEEE754 to binary & binary to IEEE754, radix point location deciding module, zeros padding module, radix point shifting module etc. which increases the complexity as well as design becomes bulky, clock constraints gets affected. The biggest flaw in the design that while conversion from binary to IEEE754 or vice versa, maintaining preciseness of result will be most crucial task and somehow there will be error in the result.

PROPOSED ALGORITHM:

The algorithm that we have proposed for the implementation of FPU is shown below:

Addition	$X + Y = (X_M 2^{X_E - Y_E} + Y_M) \times 2^{Y_E}$
Subtraction	$X - Y = (X_M 2^{X_E - Y_E} - Y_M) \times 2^{Y_E}$
Multiplication	$X \times Y = (X_M \times Y_M) \times 2^{X_E + Y_E}$
Division	$X/Y = (X_M / Y_M) \times 2^{X_E - Y_E}$

MINIMIZATION IN MAX. PROPAGATION DELAY:

Among all time constraints, we are focusing on time constraints minimization like clock period, maximum clock delay, critical path delay, setup time and hold etc. For achieving this, we are going with clock synchronous sequential circuit design approach which will reduce the clock delay. E.g. suppose we have a flip-flop with operational delay of 10ns, then if we design a 4-bit ripple counter without any clock, then minimum delay of the counter will be 40ns (10×4 Flip-Flops) i.e. we can operate the design at maximum frequency of 25MHz. But, if we design a clock synchronous 4-bit counter, then delay of the counter will be 10ns only i.e. we can operate the design at maximum frequency of 100MHz. From this example we can infer that, we can reduce clock delay by designing a system as sequential circuit.

MINIMIZATION IN SETUP & HOLD TIME:

A **setup timing check** verifies the timing relationship between the clock and the data pin of a flip-flop so that the setup requirement is met. In other words, the setup check ensures that the data is available at the input of the flip-flop before it is clocked in the flip-flop. The data should be stable for a certain amount of time, namely the setup time of the flip-flop, before the active edge of the clock arrives at the flip-flop. This requirement ensures that the data is captured reliably into the flip-flop. Following diagram shows the setup requirement of a typical flip-flop. A setup check verifies the setup requirement of the flip-flop.

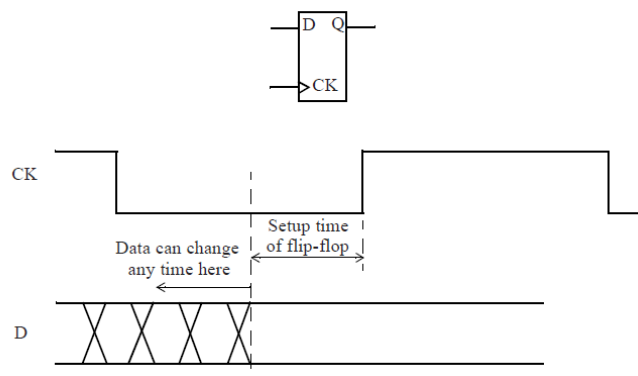


Fig. 5: Setup timing check

A **hold timing check** ensures that a flip-flop output value that is changing does not pass through to a capture flip-flop and overwrite its output before the flip-flop has had a chance to capture its original value. This check is based on the hold requirement of a flip-flop. The hold specification of a flip-flop requires that the data being latched should be held stable for a specified amount of time after the active edge of the clock. Following diagram shows the hold requirement of a typical flip-flop.

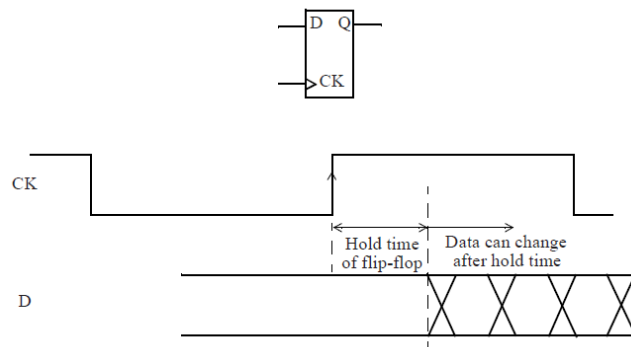


Fig. 6: Hold timing check

Now, consider the typical circuit diagram to calculate setup & hold time:

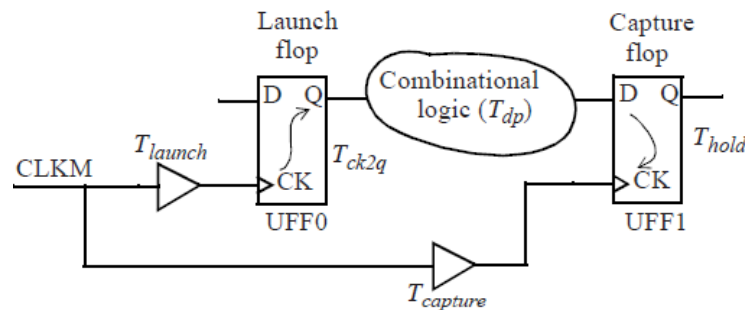


Fig. 7: Minimization of setup & hold time

The setup & hold time can be mathematically expressed as:

$$T_{setup} > T_{launch} + T_{ck2q} + T_{dp} - T_{capture} - T_{cycle}$$

$$T_{hold} < T_{launch} + T_{ck2q} + T_{dp} - T_{capture}$$

where T_{launch} is the delay of the clock tree of the launch flip-flop $UFF0$, T_{dp} is the delay of the combinational logic data path and T_{cycle} is the clock period. $T_{capture}$ is the delay of the clock tree for the capture flip-flop $UFF1$.

Now, considering the equations, in order to minimize setup & hold time, the clock to q delay as well as delay of launch & capture flop is not possible at our end. Hence, the only way to minimize setup & hold time is minimizing combinational delay. In verilog code when we write `always @(posedge clk)`, the entire circuit behaves as a single fully clock synchronous finite state machine (FSM). Hence, as the entire circuit becomes sequential, all combination path gets deleted due to which the combinational path delay will no more in existence, which ultimately leads to minimization in setup & hold time.

MINIMIZED RESOURCE UTILIZATION:

We believe that component instantiation is the best method to reduce resource utilization. To achieve this, we have defined addition & multiplication as our two major blocks. For subtraction, we use addition only by inverting sign bit of second input. ($A - B = A + (-B)$). Also, the division operation is achieved by method ($A/B = A * (\text{Reciprocal of } B)$) where reciprocal algorithm is also created on basis of multiplication & addition.

CHAPTER 3

DESIGN SIMULATION & RESULT REPORTS

3.1 CODE & DESIGN SIMULATION

3.2 RTL SCHEMATICS

3.3 SYNTHESIS REPORT & NETLIST

3.4 PLACEMENT & ROUTING REPORTS

3.5 STATIC TIMING ANALYSIS (STA)

3.6 POWER REPORT

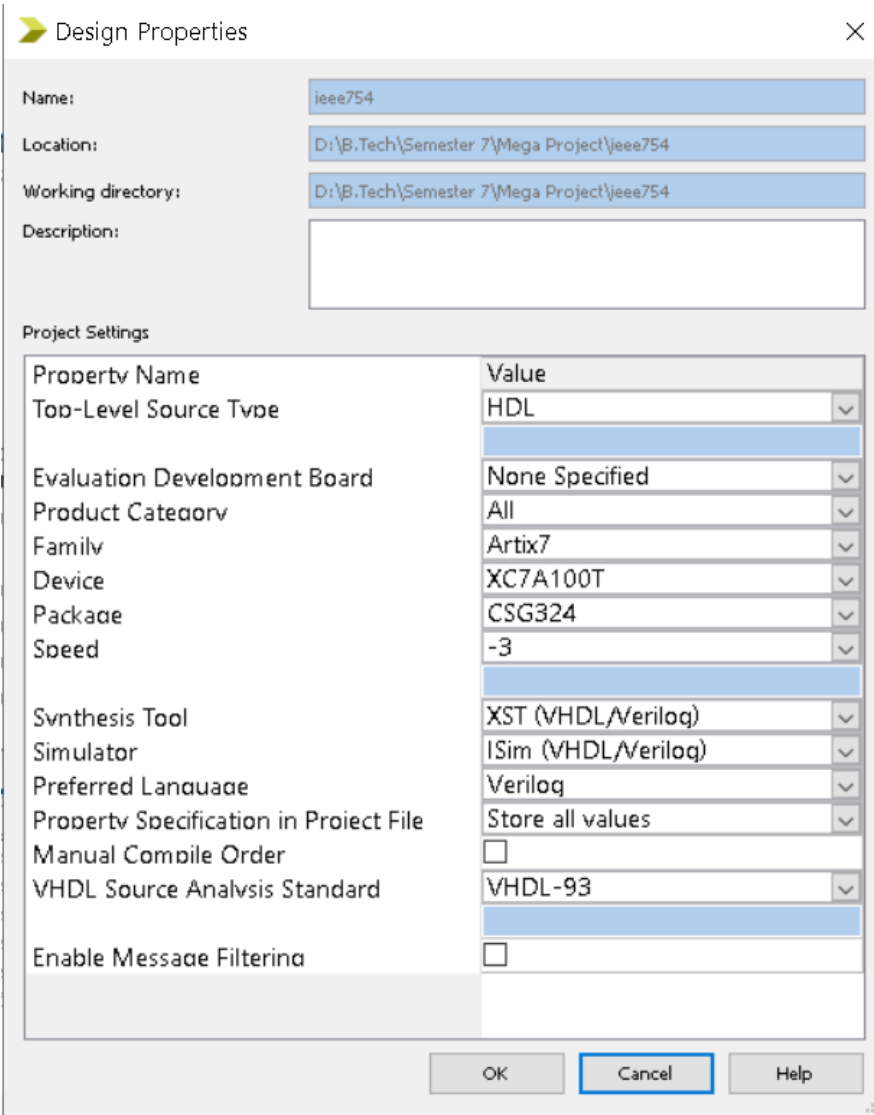
3.7 DESIGN SUMMARY

3.1 CODE & DESIGN SIMULATION

The Google Drive link to access the code & testbench (.v) files as well as all other files after synthesis, floorplanning, placement, routing, STA etc. is given below:

https://drive.google.com/drive/folders/15qzyPon22_JcLYHlrvxYt6PQ2fbEhYTC?usp=sharing

DESIGN PROPERTIES:



The Design Properties dialog box is shown with the following settings:

Property Name	Value
Name	ieee754
Location	D:\B.Tech\Semester 7\Mega Project\ieee754
Working directory	D:\B.Tech\Semester 7\Mega Project\ieee754
Description	
Project Settings	
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Artix7
Device	XC7A100T
Package	CSG324
Speed	-3
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	<input type="checkbox"/>
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	<input type="checkbox"/>

Buttons: OK, Cancel, Help

Fig. 8: Design Properties

SIMULATION:

// Brackets () indicates IEEE 754 standard format representation of decimal number in hex format

1) Normal Cases :

Inputs: 5096.75 (0x459f4600) & 1111.25 (0x448ae800)

- Outputs:
- 1) Addition = 6208 (0x45c20000)
 - 2) Subtraction = 3985.5 (0x45791800)
 - 3) Multiplication = 5663763.4375 (0x4aacd827)
 - 4) Division = 4.586501687289089 (0x425ef0f3)

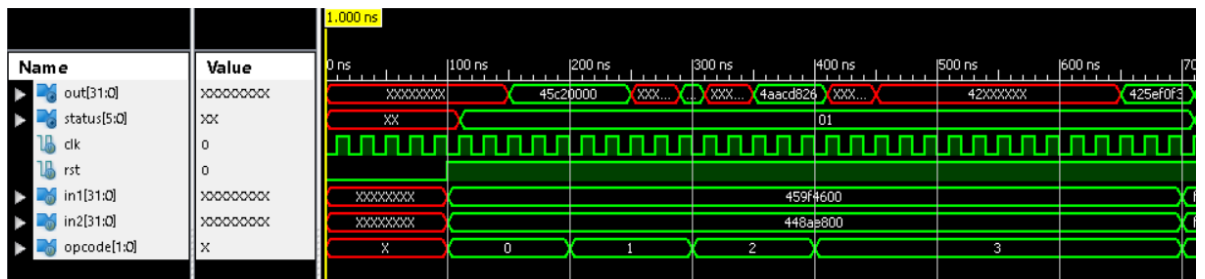


Fig. 9: Simulation of Normal Case

2) Infinity :

Inputs: Infinity (0xffffffff) & Infinity (0xffffffff)

- Outputs:
- 1) Addition = Infinity (0xffffffff)
 - 2) Subtraction = Infinity (0xffffffff)
 - 3) Multiplication = Infinity (0xffffffff)
 - 4) Division = Garbage Values (Indeterminate Form)

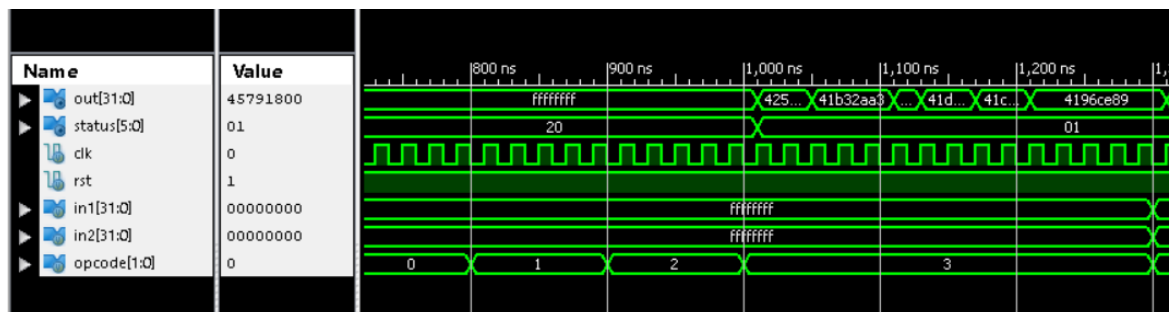


Fig. 10: Simulation of Infinity

3) Not a Number (NaN) :

When either one or both input are NaN, output will be also NaN and it will show any garbage mantissa.

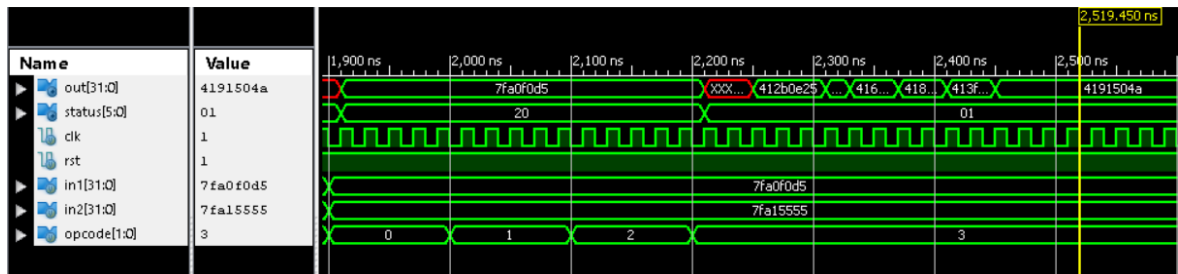


Fig. 11: Simulation of Normal Case

4) 0/0 Case :

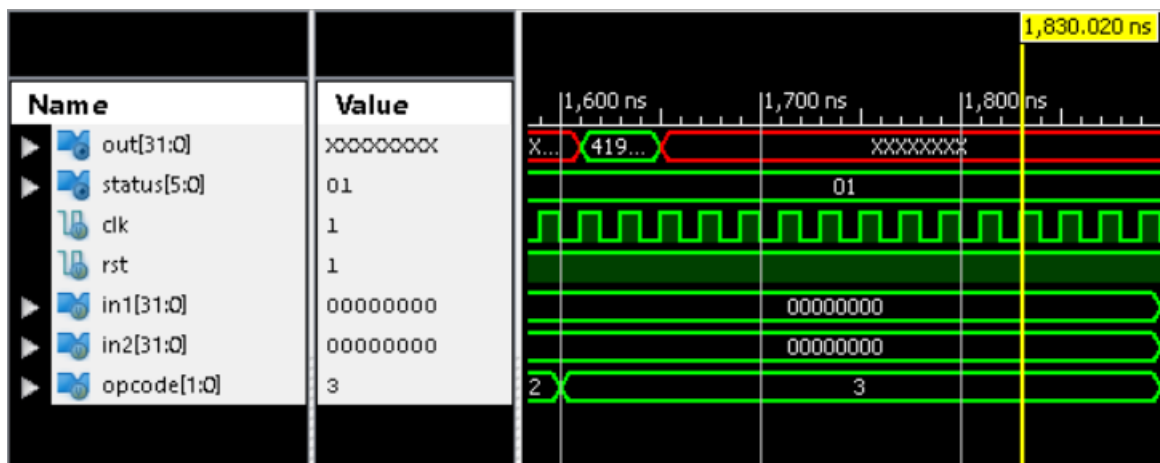


Fig. 12: Simulation of 0/0 Case

3.2 RTL SCHEMATICS

Addition/Subtraction Module:

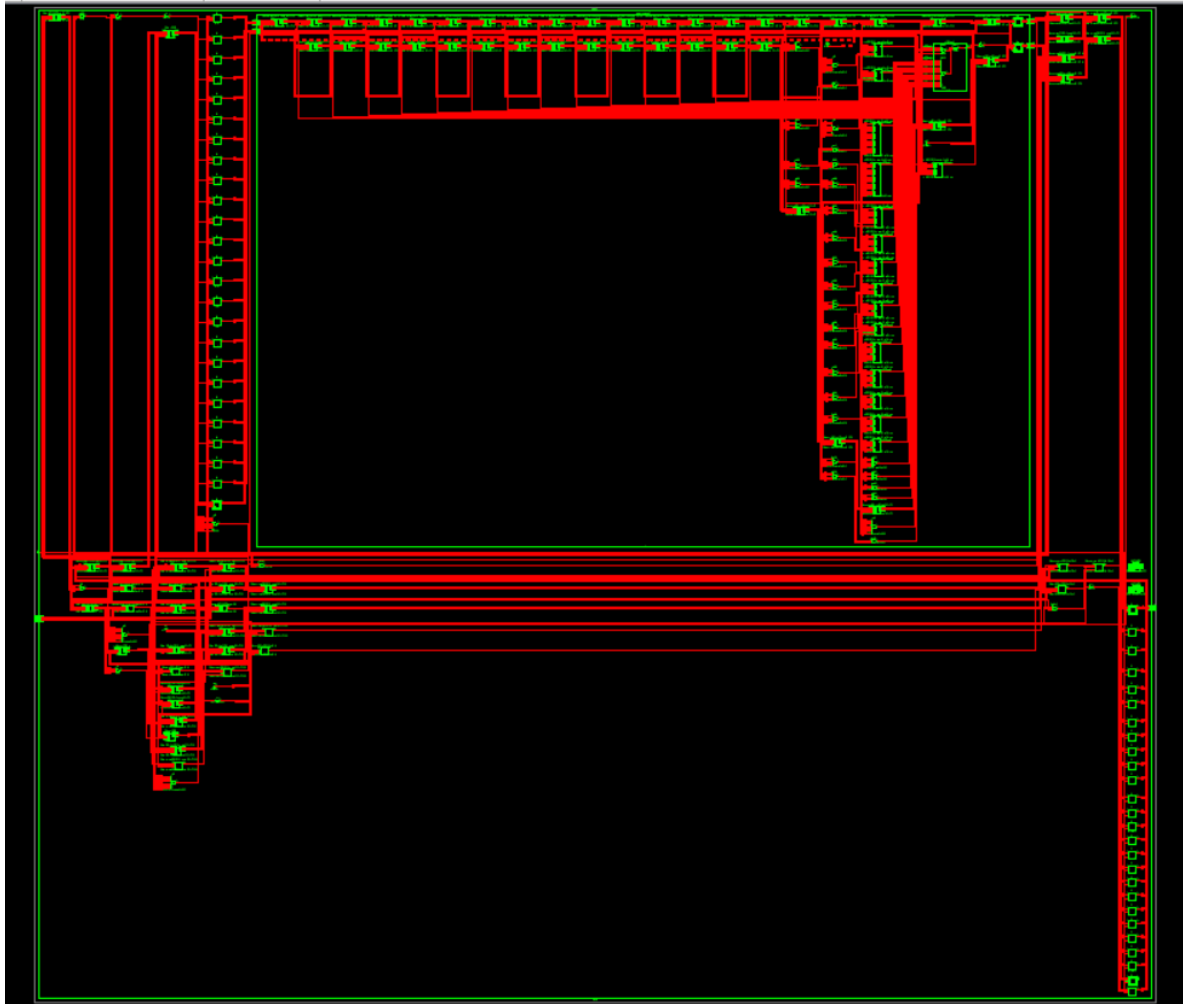


Fig. 13: RTL Schematics of adder module

Multiplication Module:

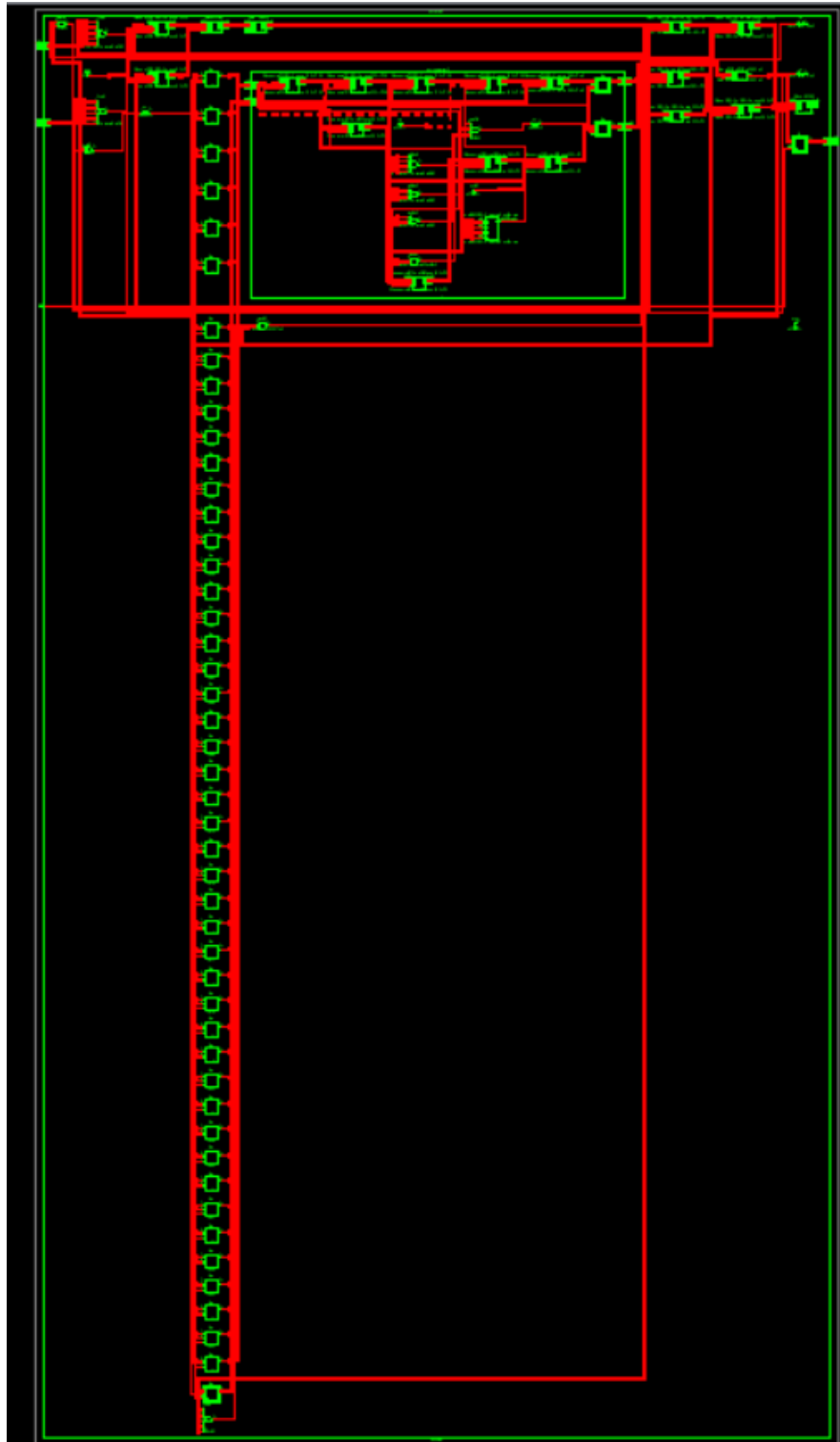


Fig. 14: RTL Schematics of multiplier module

Division Module:

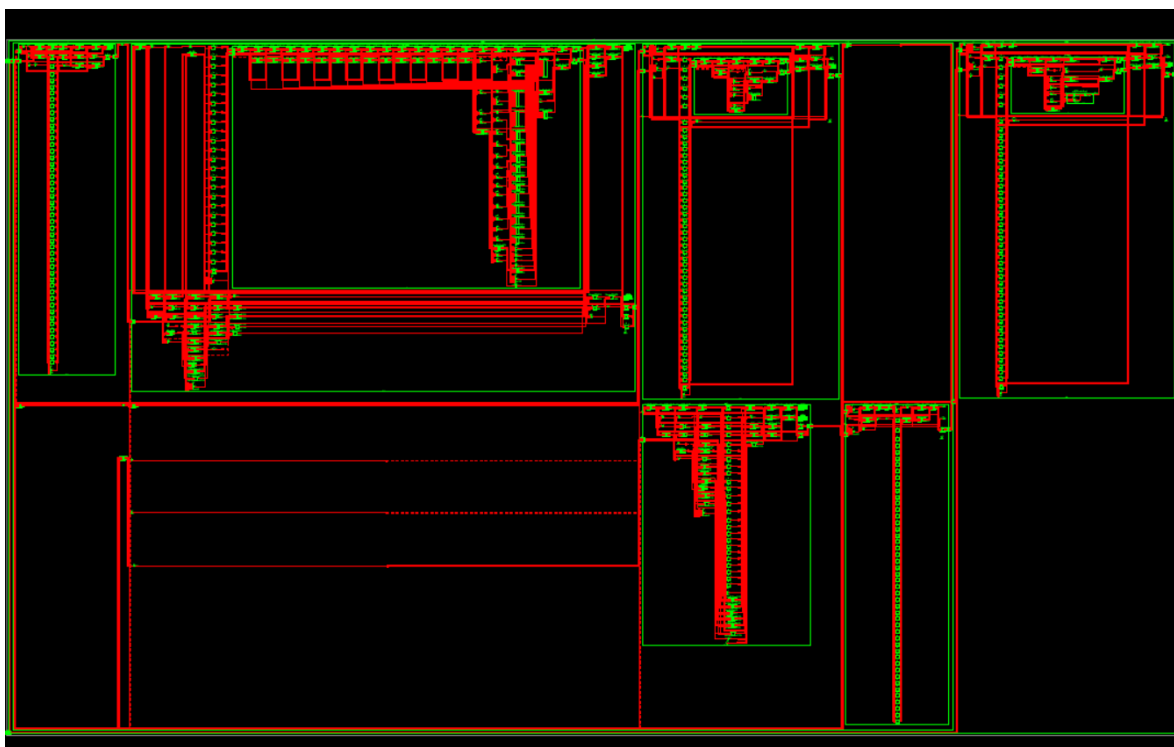


Fig. 15: RTL Schematics of divisor module

Entire IEEE 754 Module: (*Rotated View*)

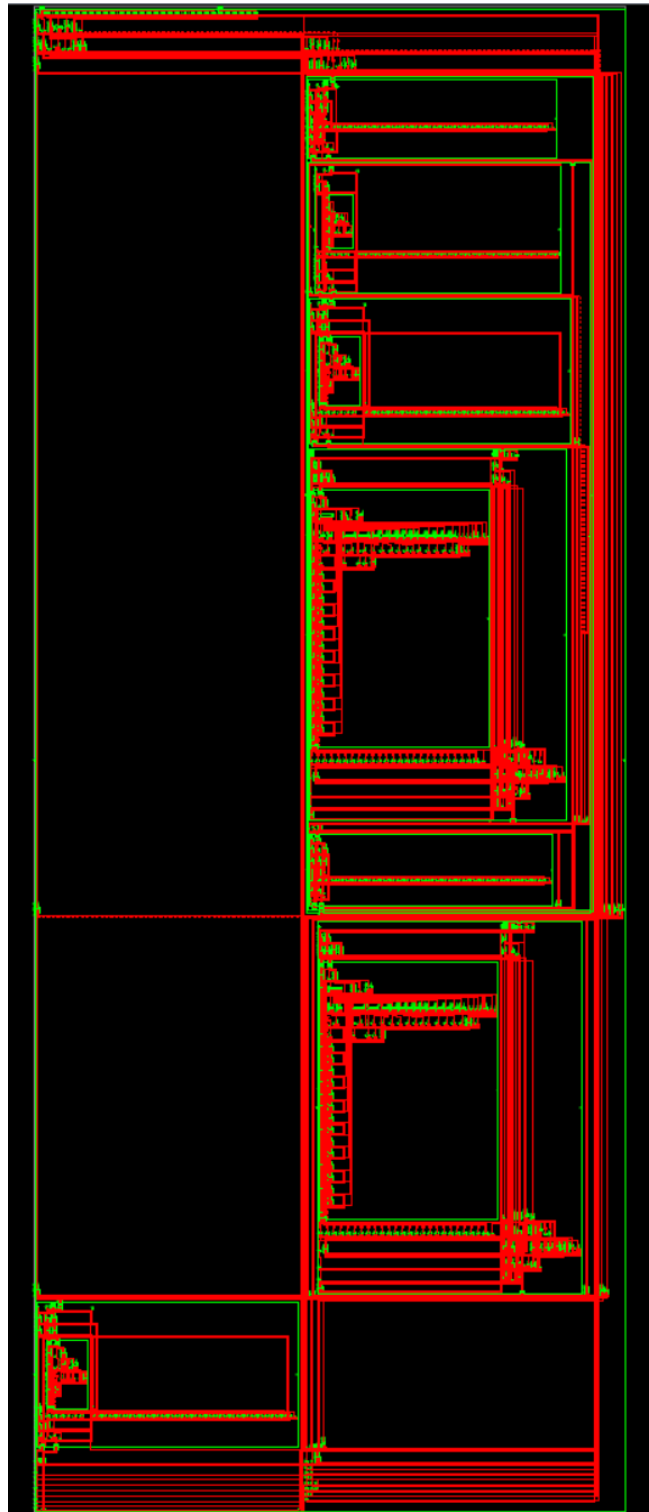


Fig. 16: RTL Schematics of top module

3.3 SYNTHESIS REPORT & NETLIST

```
=====
                        HDL Synthesis Report
=====
Macro Statistics
# Multipliers                : 7
  24x24-bit multiplier       : 7
# Adders/Subtractors        : 53
  24-bit adder               : 4
  25-bit addsub              : 4
  8-bit adder                : 11
  8-bit subtractor           : 26
  9-bit adder                : 7
  9-bit subtractor           : 1
# Registers                  : 87
  1-bit register             : 12
  23-bit register            : 7
  25-bit register            : 13
  32-bit register            : 6
  48-bit register            : 14
  6-bit register             : 1
  8-bit register             : 34
# Comparators                : 20
  1-bit comparator equal     : 4
  24-bit comparator greater  : 4
  8-bit comparator equal     : 4
  8-bit comparator greater   : 8
# Multiplexers               : 390
  1-bit 2-to-1 multiplexer   : 37
  24-bit 2-to-1 multiplexer  : 8
  25-bit 2-to-1 multiplexer  : 132
  32-bit 2-to-1 multiplexer  : 1
  48-bit 2-to-1 multiplexer  : 42
  6-bit 2-to-1 multiplexer   : 2
  8-bit 2-to-1 multiplexer   : 168
# Logic shifters              : 8
  24-bit shifter logical right : 8
# Xors                        : 8
  1-bit xor2                 : 8

=====
*                Design Summary                *
=====

Top Level Output File Name      : ieee754.ngc

Primitive and Black Box Usage:
-----
# BELS                          : 4184
```


#	GND	:	1
#	INV	:	27
#	LUT1	:	46
#	LUT2	:	196
#	LUT3	:	504
#	LUT4	:	225
#	LUT5	:	757
#	LUT6	:	1884
#	MUXCY	:	258
#	MUXF7	:	19
#	VCC	:	1
#	XORCY	:	266
#	FlipFlops/Latches	:	1752
#	FD	:	589
#	FDE	:	1130
#	FDRE	:	23
#	FDSE	:	10
#	Clock Buffers	:	1
#	BUFGP	:	1
#	IO Buffers	:	105
#	IBUF	:	67
#	OBUF	:	38
#	DSPs	:	14
#	DSP48E1	:	14

Device utilization summary:

Selected Device: 7a100tcsg324-3

Slice Logic Utilization:

Number of Slice Registers:	1568 out of 126800	1%
Number of Slice LUTs:	3639 out of 63400	5%
Number used as Logic:	3639 out of 63400	5%

Slice Logic Distribution:

Number of LUT Flip Flop pairs used:	4041	
Number with an unused Flip Flop:	2473 out of 4041	61%
Number with an unused LUT:	402 out of 4041	9%
Number of fully used LUT-FF pairs:	1166 out of 4041	28%
Number of unique control sets:	33	

IO Utilization:

Number of IOs:	106	
Number of bonded IOBs:	106 out of 210	50%
IOB Flip Flops/Latches:	184	

Specific Feature Utilization:

Number of BUFG/BUFGCTRL/BUFHCEs:	1 out of 128	0%
Number of DSP48E1s:	14 out of 240	5%

NETLIST:

```

File Edit Format View Help
// ~~~~~
// Copyright (c) 1995-2012 Xilinx, Inc. All rights reserved.
// ~~~~~
//
//      / \
//     /   \
//    /     \
//   /       \
//  /         \
// /           \
// \           /
//  \         /
//   \       /
//    \     /
//     \   /
//      \ /
//
Vendor: Xilinx
Version: P.58f
Application: netgen
Filename: ieee754_synthesis.v
Timestamp: Fri Apr 22 23:01:20 2022
//
//
// Command : -intstyle ise -insert_glbl true -w -dir netgen/synthesis -ofmt verilog -sim ieee754.ngc ieee754_synthesis.v
// Device : xc7a100t-3-csg324
// Input file : ieee754.ngc
// Output file : D:\B.Tech\Semester 7\Mega Project\ieee754\netgen\synthesis\ieee754_synthesis.v
// # of Modules : 1
// Design Name : ieee754
// Xilinx : C:\Xilinx\14.5\ISE_05\ISE\
//
// Purpose:
// This verilog netlist is a verification model and uses simulation
// primitives which may not represent the true implementation of the
// device, however the netlist is functionally correct and should not
// be modified. This file cannot be synthesized and should only be used
// with supported simulation tools.
//
// Reference:
// Command Line Tools User Guide, Chapter 23 and Synthesis and Simulation Design Guide, Chapter 6
//
// ~~~~~
timescale 1 ns/1 ps

module ieee754 (
    clk, rst, in1, in2, opcode, out, status
);
    input clk;
    input rst;
    input [31:0] in1;
    input [31:0] in2;

    // ~~~~~
    .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lutdi9_532 )
    };
    MUXCY \A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cyc8> (
        .CI(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cy [7]),
        .DI(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lutdi8_535 ),
        .S(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut [8]),
        .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cy [8])
    );
    LUT4 #(
        .INIT ( 16'h9009 ))
    \A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut8> (
        .I0(addr_in1_in[16]),
        .I1(addr_in2_in[16]),
        .I2(addr_in1_in[17]),
        .I3(addr_in2_in[17]),
        .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut [8])
    );
    LUT4 #(
        .INIT ( 16'h08AE ))
    \A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lutdi8 (
        .I0(addr_in2_in[17]),
        .I1(addr_in2_in[16]),
        .I2(addr_in1_in[16]),
        .I3(addr_in1_in[17]),
        .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lutdi8_535 )
    );
    MUXCY \A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cyc7> (
        .CI(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cy [6]),
        .DI(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lutdi7_538 ),
        .S(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut [7]),
        .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_cy [7])
    );
    LUT4 #(
        .INIT ( 16'h9009 ))
    \A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut7> (
        .I0(addr_in1_in[14]),
        .I1(addr_in2_in[14]),
        .I2(addr_in1_in[15]),
        .I3(addr_in2_in[15]),
        .0{(\A1/Mcompar_GND_2_o_GND_2_o_lessThan_11_o_lut [7])
    );
    LUT4 #(

```

Fig. 17 : Netlist

(As netlist is of 10000+ lines, we are attaching screenshot of it, the actual netlist is available in the Google drive link mentioned earlier)

3.4 PLACEMENT & ROUTING REPORTS

Specific Feature Utilization:

Number of RAMB36E1/FIFO36E1s:	0 out of	135	0%
Number of RAMB18E1/FIFO18E1s:	0 out of	270	0%
Number of BUFG/BUFGCTRLs:	1 out of	32	3%
Number used as BUFGs:	1		
Number used as BUFGCTRLs:	0		
Number of IDELAYE2/IDELAYE2_FINEDELAYS:	0 out of	300	0%
Number of ILOGICE2/ILOGICE3/ISERDESE2s:	64 out of	300	21%
Number used as ILOGICE2s:	64		
Number used as ILOGICE3s:	0		
Number used as ISERDESE2s:	0		
Number of ODELAYE2/ODELAYE2_FINEDELAYS:	0		
Number of OLOGICE2/OLOGICE3/OSERDESE2s:	34 out of	300	11%
Number used as OLOGICE2s:	34		
Number used as OLOGICE3s:	0		
Number used as OSERDESE2s:	0		
Number of PHASER_IN/PHASER_IN_PHYs:	0 out of	24	0%
Number of PHASER_OUT/PHASER_OUT_PHYs:	0 out of	24	0%
Number of BSCANS:	0 out of	4	0%
Number of BUFHCEs:	0 out of	96	0%
Number of BUFRs:	0 out of	24	0%
Number of CAPTUREs:	0 out of	1	0%
Number of DNA_PORTS:	0 out of	1	0%
Number of DSP48E1s:	14 out of	240	5%
Number of EFUSE_USRs:	0 out of	1	0%
Number of FRAME_ECCs:	0 out of	1	0%
Number of IBUFDS_GTE2s:	0 out of	4	0%
Number of ICAPs:	0 out of	2	0%
Number of IDELAYCTRLs:	0 out of	6	0%
Number of IN_FIFOs:	0 out of	24	0%
Number of MMCME2_ADVs:	0 out of	6	0%
Number of OUT_FIFOs:	0 out of	24	0%
Number of PCIE_2_1s:	0 out of	1	0%
Number of PHASER_REFs:	0 out of	6	0%
Number of PHY_CONTROLS:	0 out of	6	0%
Number of PLLE2_ADVs:	0 out of	6	0%
Number of STARTUPs:	0 out of	1	0%
Number of XADCs:	0 out of	1	0%

Overall effort level (-ol): High

Router effort level (-rl): High

Starting initial Timing Analysis.

REAL time: 22 secs

Finished initial Timing Analysis.

REAL time: 22 secs

Starting Router

Phase 1 : 21677 unrouted; REAL time: 25 secs
 Phase 2 : 18607 unrouted; REAL time: 26 secs
 Phase 3 : 7859 unrouted; REAL time: 34 secs
 Phase 4 : 7866 unrouted; (Par is working to improve performance) REAL time: 41 secs

Updating file: ieee754.ncd with current fully routed design.

Phase 5 : 0 unrouted; (Par is working to improve performance) REAL time: 50 secs
 Phase 6 : 0 unrouted; (Par is working to improve performance) REAL time: 50 secs
 Phase 7 : 0 unrouted; (Par is working to improve performance) REAL time: 50 secs
 Phase 8 : 0 unrouted; (Par is working to improve performance) REAL time: 50 secs
 Phase 9 : 0 unrouted; (Par is working to improve performance) REAL time: 53 secs
 Total REAL time to Router completion: 53 secs
 Total CPU time to Router completion: 53 secs

Partition Implementation Status

 No Partitions were found in this design.

Generating "PAR" statistics.

Asterisk (*) preceding a constraint indicates it was not met.
 This may be due to a setup or hold violation.

Constraint	Check	Worst Case	Best Case	Timing	Timing	
		Slack	Achievable	Errors	Score	
Autotimespec constraint for clock net clk	SETUP		N/A	9.341ns	N/A	0
_BUFGP	HOLD	0.026ns		0	0	

 All constraints were met.

Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: 56 secs
 Total CPU time to PAR completion: 56 secs

Peak Memory Usage: 4926 MB

Placer: Placement generated during map.
 Routing: Completed - No errors found.

Number of error messages: 0
 Number of warning messages: 0
 Number of info messages: 2

Writing design to file ieee754.ncd

PAR done!

ROUTED DESIGN:

Artix7 XC7A100T FPGA Standard Cell:

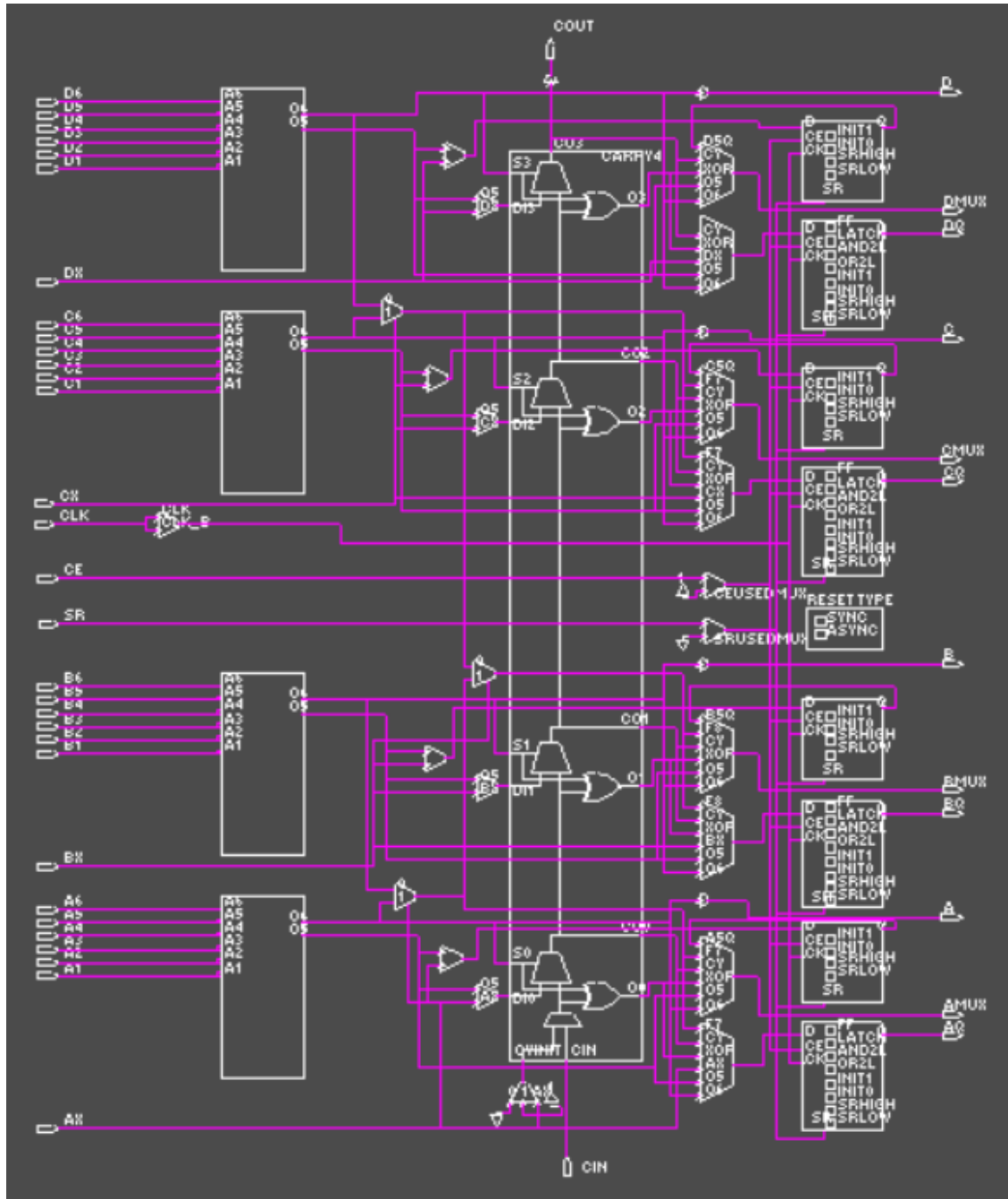
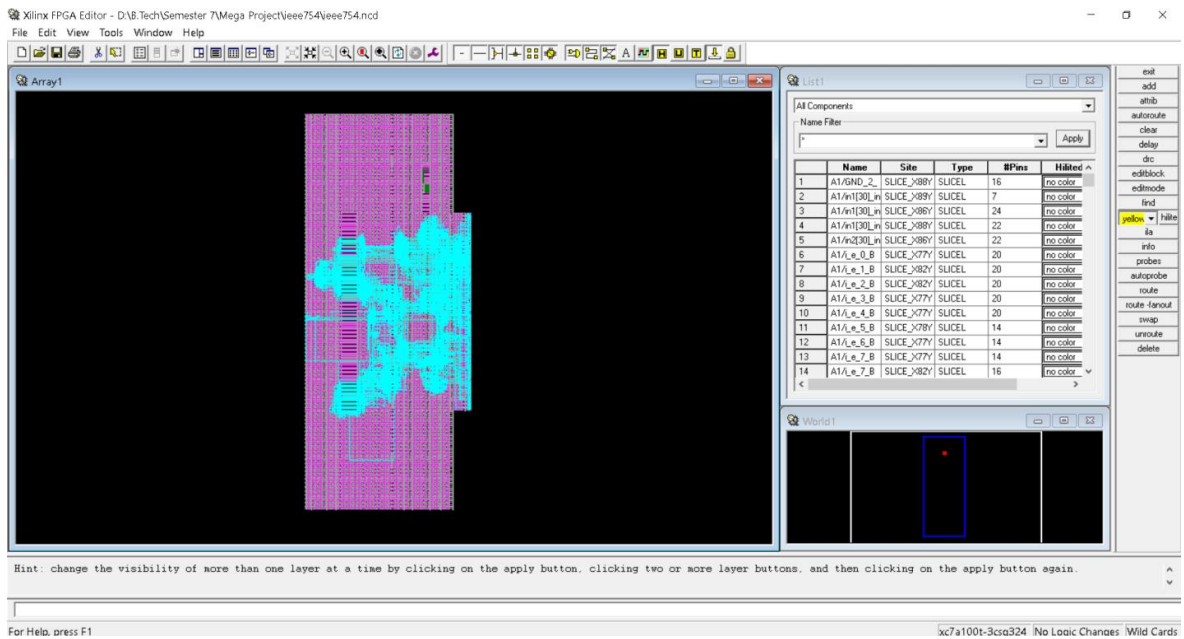


Fig. 18 : Artix7 XC7A100T FPGA Standard Cell

Routed Design:



Close View :

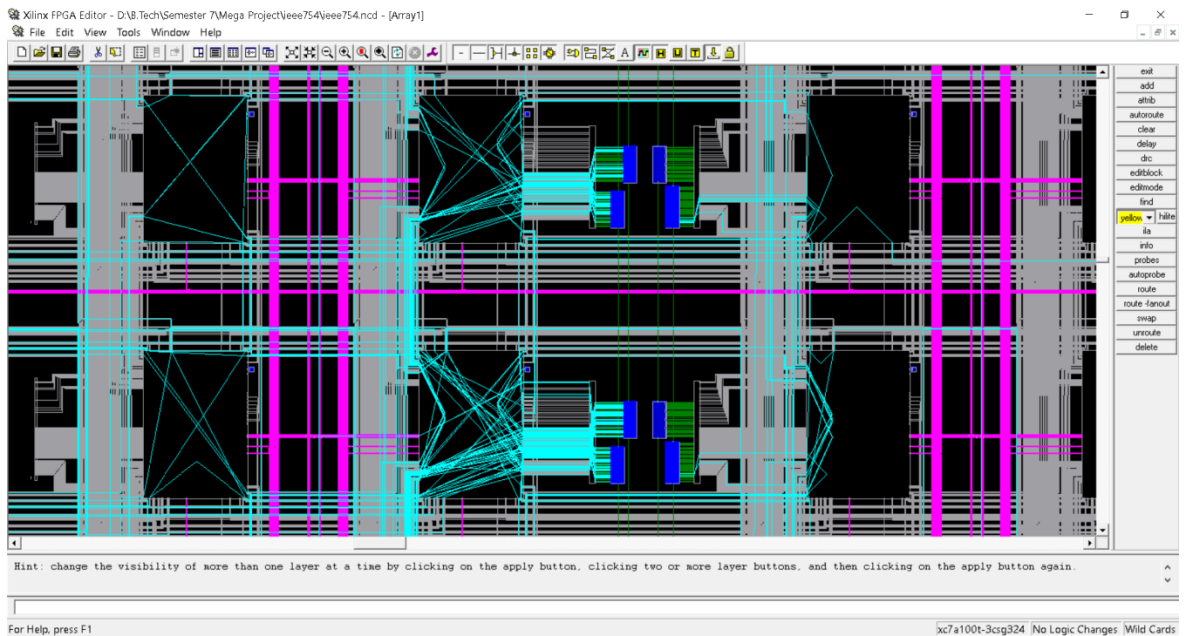


Fig. 19 : Routed Design

Clock Routing:

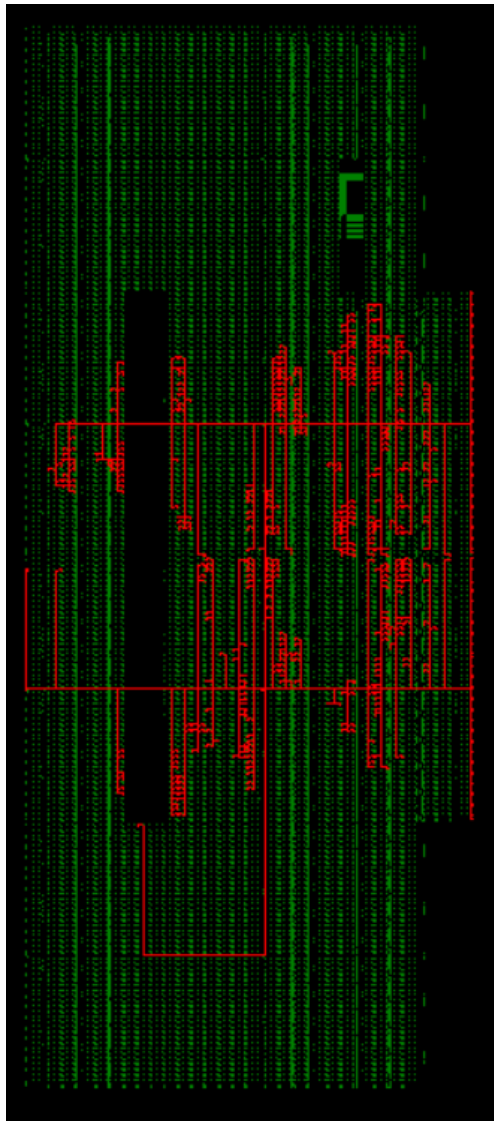


Fig. 20 : Clock Routing

DRC Check:

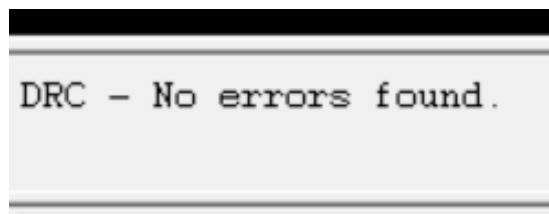


Fig. 21 : DRC Check

3.5 STATIC TIMING ANALYSIS (STA)

Post Routing STA Report:

=====

Timing Report

=====

Clock Information:

-----+-----+-----+			
Clock Signal		Clock buffer(FF name)	Load
-----+-----+-----+			
clk		BUFGP	1760
-----+-----+-----+			

Asynchronous Control Signals Information:

No asynchronous control signals found in this design

Timing Summary:

Speed Grade: -3

Minimum period: 7.243ns (Maximum Frequency: 138.068MHz)

Minimum input arrival time before clock: 2.646ns

Maximum output required time after clock: 1.033ns

Maximum combinational path delay: No path found

Timing Details:

All values displayed in nanoseconds (ns)

=====

Timing constraint: Default period analysis for Clock 'clk'

Clock period: 7.243ns (frequency: 138.068MHz)

Total number of paths / destination ports: 13172422 / 2657


```

-----

Delay:          7.243ns (Levels of Logic = 6)

Source:         D1/ recip/S1_N1/out_exp_7 (FF)

Destination:    D1/mult/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT (DSP)

Source Clock:   clk rising

Destination Clock: clk rising

Data Path: D1/ recip/S1_N1/out_exp_7 to D1/mult/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT

          Gate   Net

Cell:in->out   fanout Delay Delay Logical Name (Net Name)
-----

FD:C->Q          5 0.361 0.530 D1/ recip/S1_N1/out_exp_7 (D1/ recip/S1_N1/out_exp_7)

LUT3:I0->O       1 0.097 0.295 D1/ recip/S2_DN1/GND_5_o_INV_74_o_SW0 (N180)

LUT6:I5->O       2 0.097 0.284 D1/ recip/S2_DN1/GND_5_o_INV_74_o
(D1/ recip/S2_DN1/GND_5_o_INV_74_o)

DSP48E1:A23->PCOUT47 1 2.970 0.000
D1/ recip/S2_N2/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT
(D1/ recip/S2_N2/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT_PCOU
T_to_Mmult_GND_5_o_GND_5_o_MuLt_9_OUT1_PCIN_47)

DSP48E1:PCIN47->P29 2 1.107 0.299
D1/ recip/S2_N2/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT1
(D1/ recip/S2_N2/GND_5_o_GND_5_o_MuLt_9_OUT<46>)

LUT6:I5->O      51 0.097 0.405 D1/ recip/S2_N2/GND_5_o_GND_5_o_AND_582_o9
(D1/ recip/S2_N2/GND_5_o_GND_5_o_AND_582_o)

LUT5:I4->O       1 0.097 0.279 D1/ recip/S2_N2/Mmux_n0054171
(D1/ recip/S2_N2/n0054<39>)

DSP48E1:B16      0.324      D1/mult/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT
-----

Total          7.243ns (5.150ns logic, 2.093ns route)

              (71.1% logic, 28.9% route)

=====

Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'

Total number of paths / destination ports: 3415 / 473

```

Offset: 2.646ns (Levels of Logic = 4)

Source: in2<25> (PAD)

Destination: M1/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT (DSP)

Destination Clock: clk rising

Data Path: in2<25> to M1/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT

	Gate	Net	
Cell:in->out	fanout	Delay	Delay Logical Name (Net Name)

IBUF:I->O	5	0.001	0.530 in2_25_IBUF (in2_25_IBUF)
LUT3:I0->O	1	0.097	0.295 GND_1_o_GND_1_o_OR_279_o<30>_SW0 (N4)
LUT6:I5->O	41	0.097	0.791 GND_1_o_GND_1_o_OR_279_o<30> (GND_1_o_GND_1_o_OR_279_o)
LUT5:I0->O	48	0.097	0.389 _n0459_inv1 (_n0459_inv)
DSP48E1:CEB2		0.349	M1/Mmult_GND_5_o_GND_5_o_MuLt_9_OUT

Total		2.646ns (0.641ns logic, 2.005ns route)	
		(24.2% logic, 75.8% route)	

=====

Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'

Total number of paths / destination ports: 34 / 34

Offset: 1.033ns (Levels of Logic = 2)

Source: status_0_1 (FF)

Destination: status<5> (PAD)

Source Clock: clk rising

Data Path: status_0_1 to status<5>

Gate Net

Cell:in->out fanout Delay Delay Logical Name (Net Name)

FDSE:C->Q	1	0.361	0.279	status_0_1 (status_0_1)
INV:I->O	1	0.113	0.279	status<0>_inv1_INV_0 (status_5_OBUF)
OBUF:I->O		0.000		status_5_OBUF (status<5>)

Total 1.033ns (0.474ns logic, 0.559ns route)

 (45.9% logic, 54.1% route)

=====
Cross Clock Domains Report:
=====

Clock to Setup on destination clock clk

-----+-----+-----+-----+-----+

	Src:Rise	Src:Fall	Src:Rise	Src:Fall
Source Clock	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall

-----+-----+-----+-----+-----+

clk	7.243			
-----	-------	--	--	--

-----+-----+-----+-----+-----+

=====
Setup/Hold to clock clk

-----+-----+-----+-----+-----+-----+

	Max Setup to	Process	Max Hold to	Process		Clock	
Source	clk (edge)	Corner	clk (edge)	Corner	Internal Clock(s)	Phase	

-----+-----+-----+-----+-----+-----+

in1<0>	-0.520(R)	SLOW	3.855(R)	SLOW	clk_BUFGP	0.000
in1<1>	-0.442(R)	SLOW	3.862(R)	SLOW	clk_BUFGP	0.000
in1<2>	-0.442(R)	SLOW	3.865(R)	SLOW	clk_BUFGP	0.000
in1<3>	-0.446(R)	SLOW	3.853(R)	SLOW	clk_BUFGP	0.000
in1<4>	-0.446(R)	SLOW	3.915(R)	SLOW	clk_BUFGP	0.000

in1<5>	-0.429(R)	SLOW	3.846(R)	SLOW	clk_BUFGP	0.000
in1<6>	-0.440(R)	SLOW	3.854(R)	SLOW	clk_BUFGP	0.000
in1<7>	-0.504(R)	SLOW	3.851(R)	SLOW	clk_BUFGP	0.000
in1<8>	-0.724(R)	SLOW	3.856(R)	SLOW	clk_BUFGP	0.000
in1<9>	-0.604(R)	SLOW	3.855(R)	SLOW	clk_BUFGP	0.000
in1<10>	-0.493(R)	SLOW	3.875(R)	SLOW	clk_BUFGP	0.000
in1<11>	-0.414(R)	SLOW	3.880(R)	SLOW	clk_BUFGP	0.000
in1<12>	-0.435(R)	SLOW	3.850(R)	SLOW	clk_BUFGP	0.000
in1<13>	-0.442(R)	SLOW	3.852(R)	SLOW	clk_BUFGP	0.000
in1<14>	-0.374(R)	SLOW	3.845(R)	SLOW	clk_BUFGP	0.000
in1<15>	-0.480(R)	SLOW	3.843(R)	SLOW	clk_BUFGP	0.000
in1<16>	-0.581(R)	SLOW	3.848(R)	SLOW	clk_BUFGP	0.000
in1<17>	-0.457(R)	SLOW	3.852(R)	SLOW	clk_BUFGP	0.000
in1<18>	-0.291(R)	SLOW	3.876(R)	SLOW	clk_BUFGP	0.000
in1<19>	-0.551(R)	SLOW	3.863(R)	SLOW	clk_BUFGP	0.000
in1<20>	-0.232(R)	SLOW	3.853(R)	SLOW	clk_BUFGP	0.000
in1<21>	-0.347(R)	SLOW	3.848(R)	SLOW	clk_BUFGP	0.000
in1<22>	-0.253(R)	SLOW	3.849(R)	SLOW	clk_BUFGP	0.000
in1<23>	0.684(R)	SLOW	3.857(R)	SLOW	clk_BUFGP	0.000
in1<24>	0.759(R)	SLOW	3.864(R)	SLOW	clk_BUFGP	0.000
in1<25>	0.889(R)	SLOW	3.857(R)	SLOW	clk_BUFGP	0.000
in1<26>	0.490(R)	SLOW	3.846(R)	SLOW	clk_BUFGP	0.000
in1<27>	0.587(R)	SLOW	3.873(R)	SLOW	clk_BUFGP	0.000
in1<28>	0.640(R)	SLOW	3.871(R)	SLOW	clk_BUFGP	0.000
in1<29>	0.769(R)	SLOW	3.877(R)	SLOW	clk_BUFGP	0.000
in1<30>	0.778(R)	SLOW	3.871(R)	SLOW	clk_BUFGP	0.000
in1<31>	-0.850(R)	SLOW	3.856(R)	SLOW	clk_BUFGP	0.000
in2<0>	-0.300(R)	SLOW	3.864(R)	SLOW	clk_BUFGP	0.000
in2<1>	-0.310(R)	SLOW	3.864(R)	SLOW	clk_BUFGP	0.000

in2<2>	-0.061(R)	SLOW	3.864(R)	SLOW	clk_BUF	GP	0.000
in2<3>	-0.182(R)	SLOW	3.865(R)	SLOW	clk_BUF	GP	0.000
in2<4>	-0.045(R)	SLOW	3.858(R)	SLOW	clk_BUF	GP	0.000
in2<5>	0.478(R)	SLOW	3.832(R)	SLOW	clk_BUF	GP	0.000
in2<6>	0.434(R)	SLOW	3.888(R)	SLOW	clk_BUF	GP	0.000
in2<7>	0.400(R)	SLOW	3.856(R)	SLOW	clk_BUF	GP	0.000
in2<8>	-0.450(R)	SLOW	3.873(R)	SLOW	clk_BUF	GP	0.000
in2<9>	0.107(R)	SLOW	3.877(R)	SLOW	clk_BUF	GP	0.000
in2<10>	0.034(R)	SLOW	3.877(R)	SLOW	clk_BUF	GP	0.000
in2<11>	0.391(R)	SLOW	3.867(R)	SLOW	clk_BUF	GP	0.000
in2<12>	0.076(R)	SLOW	3.838(R)	SLOW	clk_BUF	GP	0.000
in2<13>	-0.027(R)	SLOW	3.869(R)	SLOW	clk_BUF	GP	0.000
in2<14>	0.494(R)	SLOW	3.825(R)	SLOW	clk_BUF	GP	0.000
in2<15>	0.237(R)	SLOW	3.832(R)	SLOW	clk_BUF	GP	0.000
in2<16>	0.133(R)	SLOW	3.829(R)	SLOW	clk_BUF	GP	0.000
in2<17>	0.004(R)	SLOW	3.876(R)	SLOW	clk_BUF	GP	0.000
in2<18>	0.329(R)	SLOW	3.889(R)	SLOW	clk_BUF	GP	0.000
in2<19>	0.227(R)	SLOW	3.877(R)	SLOW	clk_BUF	GP	0.000
in2<20>	0.118(R)	SLOW	3.838(R)	SLOW	clk_BUF	GP	0.000
in2<21>	0.272(R)	SLOW	3.873(R)	SLOW	clk_BUF	GP	0.000
in2<22>	0.114(R)	SLOW	3.868(R)	SLOW	clk_BUF	GP	0.000
in2<23>	1.212(R)	SLOW	3.891(R)	SLOW	clk_BUF	GP	0.000
in2<24>	1.566(R)	SLOW	3.910(R)	SLOW	clk_BUF	GP	0.000
in2<25>	1.393(R)	SLOW	3.884(R)	SLOW	clk_BUF	GP	0.000
in2<26>	1.054(R)	SLOW	3.884(R)	SLOW	clk_BUF	GP	0.000
in2<27>	1.027(R)	SLOW	3.883(R)	SLOW	clk_BUF	GP	0.000
in2<28>	1.234(R)	SLOW	3.878(R)	SLOW	clk_BUF	GP	0.000
in2<29>	1.360(R)	SLOW	3.889(R)	SLOW	clk_BUF	GP	0.000
in2<30>	1.371(R)	SLOW	3.882(R)	SLOW	clk_BUF	GP	0.000

in2<31>		-0.588(R)	SLOW		3.885(R)	SLOW		clk_BUFGP		0.000
opcode<0>		0.169(R)	SLOW		2.893(R)	FAST		clk_BUFGP		0.000
opcode<1>		0.275(R)	SLOW		3.029(R)	FAST		clk_BUFGP		0.000
rst		-0.516(R)	SLOW		4.218(R)	FAST		clk_BUFGP		0.000

-----+-----+-----+-----+-----+-----+-----+

Clock clk to Pad

-----+-----+-----+-----+-----+-----+-----+

Max (slowest) clk	Process	Min (fastest) clk	Process		Clock	
-------------------	---------	-------------------	---------	--	-------	--

Destination	(edge) to PAD	Corner	(edge) to PAD	Corner	Internal Clock(s)	Phase	
-------------	---------------	--------	---------------	--------	-------------------	-------	--

-----+-----+-----+-----+-----+-----+-----+

out<0>		7.088(R)	SLOW		5.093(R)	FAST		clk_BUFGP		0.000
out<1>		7.085(R)	SLOW		5.090(R)	FAST		clk_BUFGP		0.000
out<2>		7.096(R)	SLOW		5.101(R)	FAST		clk_BUFGP		0.000
out<3>		7.070(R)	SLOW		5.075(R)	FAST		clk_BUFGP		0.000
out<4>		7.071(R)	SLOW		5.076(R)	FAST		clk_BUFGP		0.000
out<5>		7.076(R)	SLOW		5.081(R)	FAST		clk_BUFGP		0.000
out<6>		7.071(R)	SLOW		5.076(R)	FAST		clk_BUFGP		0.000
out<7>		7.051(R)	SLOW		5.056(R)	FAST		clk_BUFGP		0.000
out<8>		7.052(R)	SLOW		5.057(R)	FAST		clk_BUFGP		0.000
out<9>		7.048(R)	SLOW		5.053(R)	FAST		clk_BUFGP		0.000
out<10>		7.046(R)	SLOW		5.051(R)	FAST		clk_BUFGP		0.000
out<11>		7.052(R)	SLOW		5.058(R)	FAST		clk_BUFGP		0.000
out<12>		7.049(R)	SLOW		5.055(R)	FAST		clk_BUFGP		0.000
out<13>		7.027(R)	SLOW		5.033(R)	FAST		clk_BUFGP		0.000
out<14>		7.039(R)	SLOW		5.044(R)	FAST		clk_BUFGP		0.000
out<15>		7.050(R)	SLOW		5.055(R)	FAST		clk_BUFGP		0.000
out<16>		7.051(R)	SLOW		5.057(R)	FAST		clk_BUFGP		0.000
out<17>		7.050(R)	SLOW		5.056(R)	FAST		clk_BUFGP		0.000

out<18>		7.046(R)	SLOW	5.051(R)	FAST clk_BUFGP		0.000
out<19>		7.069(R)	SLOW	5.075(R)	FAST clk_BUFGP		0.000
out<20>		7.060(R)	SLOW	5.065(R)	FAST clk_BUFGP		0.000
out<21>		7.084(R)	SLOW	5.090(R)	FAST clk_BUFGP		0.000
out<22>		7.078(R)	SLOW	5.084(R)	FAST clk_BUFGP		0.000
out<23>		7.088(R)	SLOW	5.093(R)	FAST clk_BUFGP		0.000
out<24>		7.087(R)	SLOW	5.092(R)	FAST clk_BUFGP		0.000
out<25>		7.078(R)	SLOW	5.083(R)	FAST clk_BUFGP		0.000
out<26>		7.082(R)	SLOW	5.087(R)	FAST clk_BUFGP		0.000
out<27>		7.078(R)	SLOW	5.083(R)	FAST clk_BUFGP		0.000
out<28>		7.074(R)	SLOW	5.080(R)	FAST clk_BUFGP		0.000
out<29>		7.063(R)	SLOW	5.068(R)	FAST clk_BUFGP		0.000
out<30>		7.071(R)	SLOW	5.077(R)	FAST clk_BUFGP		0.000
out<31>		7.056(R)	SLOW	5.062(R)	FAST clk_BUFGP		0.000
status<0>		7.061(R)	SLOW	5.066(R)	FAST clk_BUFGP		0.000
status<5>		8.083(R)	SLOW	5.384(R)	FAST clk_BUFGP		0.000

-----+-----+-----+-----+-----+-----+-----+

Clock to Setup on destination clock clk

-----+-----+-----+-----+-----+

| Src:Rise| Src:Fall| Src:Rise| Src:Fall|

Source Clock | Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|

-----+-----+-----+-----+-----+

clk | 9.120| | |

-----+-----+-----+-----+-----+

Analysis completed Sun Apr 10 16:55:17 2022

STA in Routed Design:

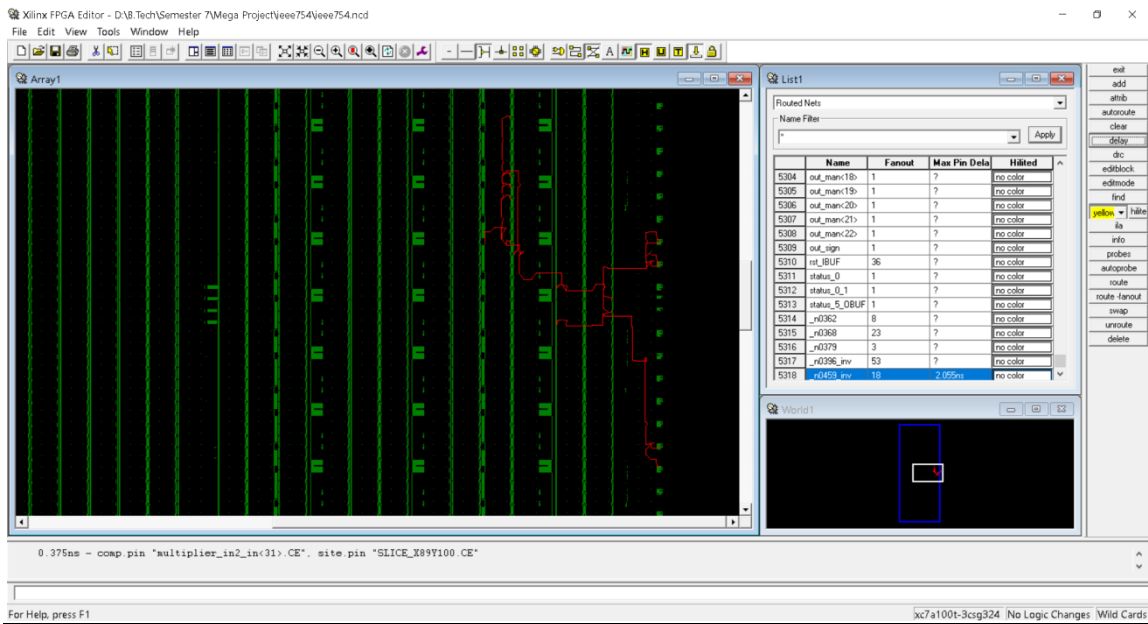


Fig. 22: STA of Routed Design

3.6 POWER REPORT

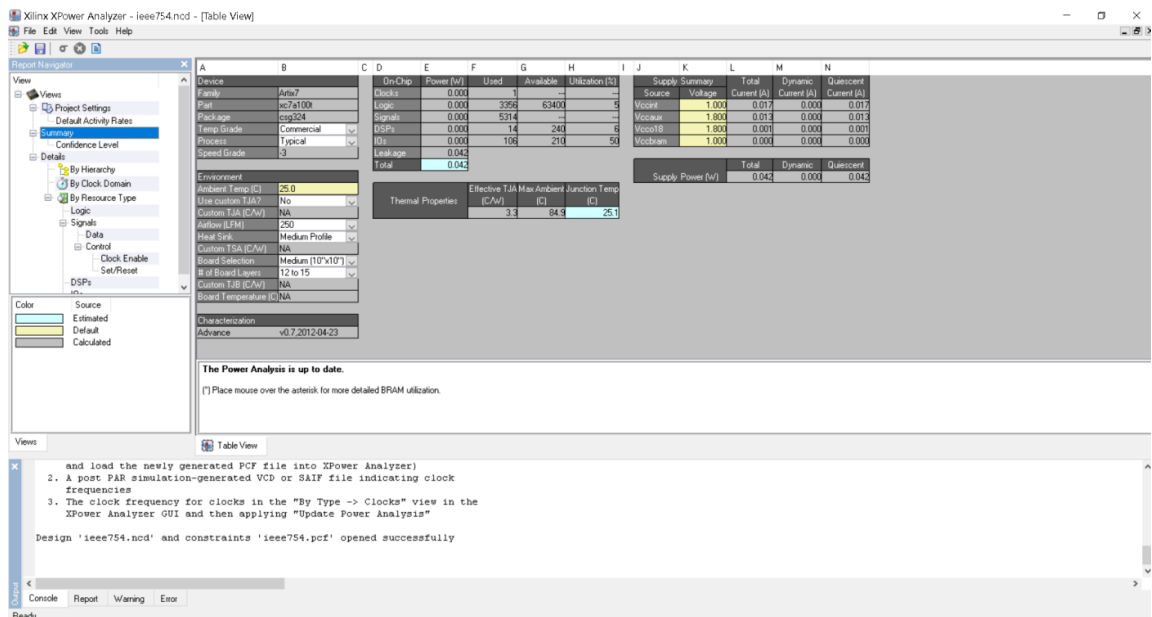


Fig. 23: Power Report

2.1. On-Chip Power Summary

On-Chip Power Summary					
On-Chip	Power (mW)	Used	Available	Utilization (%)	
Clocks	0.00	1	---	---	
Logic	0.00	3048	63400	5	
Signals	0.00	4464	---	---	
IOs	0.00	106	210	50	
DSPs	0.00	14	240	6	
Static Power	42.38				
Total	42.38				

2.2. Thermal Summary

Thermal Summary

Effective TJA (C/W) | 3.3 |

Max Ambient (C) | 84.9 |

Junction Temp (C) | 25.1 |

2.3. Power Supply Summary

Power Supply Summary

Total | Dynamic | Static Power |

Supply Power (mW) | 42.38 | 0.00 | 42.38 |

Power Supply Currents

Supply Source | Supply Voltage | Total Current (mA) | Dynamic Current (mA) | Quiescent Current (mA) |

Vccint | 1.000 | 16.58 | 0.00 | 16.58 |

Vccaux | 1.800 | 13.14 | 0.00 | 13.14 |

Vcco18 | 1.800 | 1.00 | 0.00 | 1.00 |

Vccbram | 1.000 | 0.35 | 0.00 | 0.35 |

2.4. Confidence Level

Confidence Level			

User Input Data		Confidence	
Action		Details	
Design implementation state		High	Design is completely routed
Clock nodes activity		High	User specified more than 95% of clocks
Overall confidence level		Medium	

Analysis completed: Sun Apr 10 16:35:35 2022

3.7 DESIGN SUMMARY

Design Overview

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Static Timing

Errors and Warnings

- Parser Messages
- Synthesis Messages
- Translation Messages
- Map Messages
- Place and Route Messages
- Timing Messages
- Bitgen Messages
- All Implementation Mess...

Detailed Reports

- Synthesis Report
- Translation Report
- Map Report
- Place and Route Report
- Post-PAR Static Timing R...

Design Properties

- ☐ Enable Message Filtering

Optional Design Summary Contents

- ☐ Show Clock Report
- ☐ Show Failing Constraints
- ☐ Show Warnings
- ☐ Show Errors

iee754 Project Status (04/22/2022 - 22:41:18)

Project File:	iee754.xise	Parser Errors:	No Errors
Module Name:	iee754	Implementation State:	Placed and Routed
Target Device:	xc7a100t-3csg324	Errors:	
Product Version:	ISE 14.5	Warnings:	
Design Goal:	Timing Performance	Routing Results:	All Signals Completely Routed
Design Strategy:	Performance with IOB Packing	Timing Constraints:	All Constraints Met
Environment:	System Settings	Final Timing Score:	0 (Timing Report)

Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	1,611	126,800	1%	
Number used as Flip Flops	1,611			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	3,356	63,400	5%	
Number used as logic	3,251	63,400	5%	
Number using O6 output only	2,923			
Number using O5 output only	0			
Number using O5 and O6	328			
Number used as ROM	0			
Number used as Memory	0	19,000	0%	
Number used exclusively as route-thrus	105			
Number with same-slice register load	54			
Number with same-slice carry load	51			
Number with other load	0			
Number of occupied Slices	1,388	15,850	8%	
Number of LUT Flip Flop pairs used	3,655			
Number with an unused Flip Flop	2,143	3,655	58%	
Number with an unused LUT	299	3,655	8%	
Number of fully used LUT-FF pairs	1,213	3,655	33%	
Number of unique control sets	29			

Performance Summary

Final Timing Score:	0 (Setup: 0, Hold: 0)	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sun Apr 24 11:35:36 2022	0	0	0
Translation Report	Current	Sun Apr 24 11:35:55 2022	0	195 Warnings (0 new)	5 Infos (0 new)
Map Report	Current	Sun Apr 24 11:37:31 2022	0	0	3 Infos (0 new)
Place and Route Report	Current	Sun Apr 24 11:38:26 2022	0	0	3 Infos (0 new)
Power Report	Out of Date	Sun Apr 10 16:35:35 2022	0	4 Warnings (4 new)	1 Info (1 new)
Post-PAR Static Timing Report	Current	Sun Apr 24 11:38:53 2022	0	0	4 Infos (0 new)
Bitgen Report	Current	Sun Apr 24 11:41:56 2022	2 Errors (0 new)	0	1 Info (0 new)

Secondary Reports

Report Name	Status	Generated
ISIM Simulator Log	Out of Date	Fri Apr 22 21:59:09 2022
Post-Synthesis Simulation Model Report	Out of Date	Sun Apr 24 11:34:30 2022
Post-Translate Simulation Model Report	Out of Date	Sun Apr 10 16:54:45 2022
Post-Map Static Timing Report	Out of Date	Sun Apr 10 16:55:17 2022
Post-Place and Route Simulation Model Report	Out of Date	Sun Apr 10 16:36:49 2022

Date Generated: 04/24/2022 - 11:41:57

Fig. 17: Design Summary

CHAPTER 4

INFERENCES & CONCLUSION

4.1 INFERENCES

4.2 CONCLUSION

4.3 BIBLOGRAPHY

4.1 INFERENCES

- The simulation of floating point unit gives satisfactory expected outputs.
- The exception cases like infinity, zero, NaN are well handled with proper output.
- The RTL schematics have successfully generated without any errors.
- Design has been successfully synthesized with 5% slice LUT utilization & 28% fully used LUT-FF pairs of overall available in Artix7 XC7A100T FPGA.
- The placement and routing of the entire design have been successfully completed on FPGA without any Design Rule Check (DRC) errors.
- The proper clock tree synthesis & routing has been observed without any error.
- The post-synthesis & post-routing static timing analysis has been analyzed properly for various data paths as well as I/O blocks.
- The expected timing constraints of design have successfully met with minimum clock period of 7.243ns, setup time of 2.646ns & hold time of 1.033ns.
- The power report of design has been generated & analyzed with static power requirement of 42.38 mW.

4.2 CONCLUSION

The design of 'IEEE 754 Standard Single Precision Floating Point Unit' has been successfully implemented in Verilog HDL with the help of Xilinx ISE Design Suite 14.5 EDA Tool. All the 3 domains (*i.e. time, area & power*) for optimization of the design have been analyzed and design has been optimized with respect to timing constraints & resource utilization.

The timing constraints that we aimed to minimize i.e. clock period, setup & hold time have been reduced nearby 75% with respect to the designs we referred from various research papers & journals from IEEE, IJRET and IJSER along with around 20% minimization in resource utilization.

The entire FPGA design flow has been successfully implemented for the design with proper synthesis, placement, routing & STA without failure with generation of netlist.

Concluding, we have successfully optimized the design of IEEE 754 FPU with all possible parameters by the perspective of a Design & Verification Engineer and Physical Design Engineer.

4.3 BIBLOGPRAHY

Books :

1. 'Computer Architecture and Organization' by John P Hayes
2. 'Advanced Computer Architecture' by Kai Hwang
3. 'Computer Architecture & Parallel Processing' by Kai Hwang & Briggs
4. 'Computer Organization' by Hamacher Zaky

IEEE, IJRET, IJSER Journal Papers :

- [1] A. Malik and S. Ko. A study on the floating-point adder in fpgas, 2006.
Available; accessed 14-Jan-2020.
- [2] Nisha Singh and R Dhanabal. Design of single precision floating point arithmetic logic unit. In 2018 4th International Conference on Electrical Energy Systems (ICEES), pages 133–137. IEEE, 2018.
- [3] LoucasLouca, Todd A Cook, and William H Johnson. Implementation of iee single precision floating point addition and multiplication on fpgas., 1996.
- [4] Mohamed Al-Ashrafy,AshrafSalem,Wagdy Anis, An Efficient Implementation of Floating Point Multiplier,2011 IEEE.
- [5] Shamna.K and S.R Ramesh, Design and implementation of an optimised double precision divider on FPGA ,International Journal International Journalof Advanced Science and Technology of Advanced Science and Technology of Advanced Science and Technology Vol. 18 Vol. 18, May, 2010 .
- [6] Peter Malik,High Throughput floating point dividers implemented in FPGA,2015 IEEE.

- [7] Tarun Kumar Rawat Abhay Sharma. Truncated wallace based single precision floating point multiplier, 2018.
- [8] Nihal Nitnaware Adil Shaikh, Akash Ninave and Prof. ShaiwaliBallal. Ieee 754 single precision floating point arithmetic unit using vhdl, 2017. Available; accessed 15- Jan-2020.
- [9] Ayusharma0698. Ieee standard 754 floating point numbers. <https://www.geeksforgeeks.org/ieee-standard754-floating-point-numbers/>, Sep 2019. Online; accessed 09-Oct-2019.
- [10]AnandBurud and Pradip Bhaskar. Design and implementation of fpga based 32-bit floating-point processor for dsp application. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pages 1{5. IEEE, 2018.
- [11]KaranGumber and SharmeleeThangjam. Performance analysis of floating-point adder using VHDL on reconfigurable hardware, 2012. Available; accessed 15- Jan-2020.
- [12]Z. Abid Jasbir N. Patel and Wei Wang. VLSI implementation of a floating-point divider, 2004.
- [13]LokeshKamble, Prachi Palsodkar, and Prasanna Palsodkar. Research trends in development of floatingpoint computer arithmetic. In 2017 International Conference on Communication and Signal Processing (ICCSP), pages 0329{0333. IEEE, 2017.
- [14]MohamedElSaid Khaled A. Shehata, Hassan El-Ghitani. Design of generic floating-point multiplier and adder/subtractor units, 2010.
- [15]S.Koteswara Rao and T. Srinivasa Rao. Design and implementation of 32-bit inexact floating-point arithmetic unit, 2018. Available; accessed 15-Jan-2020.48