Grenze International Journal of Engineering and Technology, January Issue

Development of High-Precision Floating Point Comparator based on IEEE 754 Standard

Abhay Chopde[1], Amit Samarth[2], Sanchit Sawarkar[3] and Riya Shah[4]
[1-4]Department of Electronics and Telecommunication, Vishwakarma Institute of Technology, Pune, India
Email: abhay.chopde@vit.edu, amit.samarth21@vit.edu, sanchit.sawarkar21@vit.edu, riya.shah21@vit.edu

***Abstract*—IEEE standards give us a clear representation of technical terms involved in the electronics field. Comparisons of the floating-point operators have a significant industrial application diversified from scientific calculations to Control systems of sensors for higher accuracy of data. IEEE 754 develops a standard guideline for floating-point numbers defining a conventional format of encodings of numbers, in accordance to achieve consistent behavioral performance. The paper discusses a comparator based on the implementation of various aspects specified in IEEE 754 standard guidelines which helps identify the comparator to present a precisive Performa over others. The proposed comparator is designed and programmed into Vivado 2023.1 and Verilog respectively. Having a power consumption of 3.859 W it is capable of representing all the specificities of the IEEE 754 standards mentioned. Fewer conditions predefined in standard is also treated as individual condition for it to be a perfect comparator. The output wire more accurately derivers not only the nature but the magnitude in a broader manner, which then can be utilized for attaining perfection over other systems. The power dissipation of the system is utilized in a perfect manner which allows the system to work efficiently. A total of 4.089 W of power is dissipated into the circuit and a total of 22 percent of power is utilized by the comparator to perform its tasks followed by dividing it into small factors depending on outputs and utilization.**

***Index Terms*— Comparator, IEEE 754, Exponent, Mantissa.**

I. INTRODUCTION

Requirements for precisive data transfers and calculations, DSP helps the individual achieve perfection over various electronic simplifications. Amongst these comparators signifies its importance in various applications with diversified outcomes, of which FPCs have influenced scientific and engineering calculations for precise data analysis. This includes the comparison of FPN which is the basis of virtual solutions for mathematics operations on real numbers in computer systems. The operation of comparison consists of operands by combining them with the infix operators. The outputs of the comparison have arithmetic significance and hence an implementation of comparing designs acts as an important division of independent structures.

The characteristics of flexible representations of real floating numbers are derived from the radix, which classifies the numbers in two forms. The storing of precise data is accomplished through these values. The divisions are as follows: -

1. For the fixed radix. It's called decimal numbers. These numbers use the base-10 representation formats.
2. And for floating radix its called floating-point numbers. These values use the base-2 representation formats.

Grenze ID: 01.GIJET.11.1.39_1

The free flow of radix is the reason for its representing name. This name is derived from its characteristics of free-flowing points in between the numbers. The size of these numbers varies according to the number of bits for its representation and so the total possibilities of its different forms.
The standard specified for floating point numbers is IEEE 754 and it specifies various aspects of floating representations which includes the single and double precision, followed by special values for treating the extreme conditions and overflow and underflow for precision of comparison. It classifies the floating numbers in terms like denormalized number, special number rounding modes for numbers, and finally NaN.
Floating point numbers have division for the classification of number formats which states the basis based on binary bits. These binary bits consist of a sum of sign bit, exponent bit, and significand bits. This classification consists of half precision which utilizes a 16-bit binary representation followed by single precision and then the proposed methodology which uses a 64-bit binary representation which is renamed as double precision. Double precision contains 53 significand bits which is often referred to as mantissa and this value stores the real or actual numerical value of the number for representation. It also includes the sign bit which justifies the positive or negative nature of the number, followed by 11 exponent bits for visualization of scale of real numbers. Together these signify the magnitude of the floating-point number.
The Paper reviews the proposed project in the following sections starting with briefing the basics of the project it discusses about the double precision representation. Normalizations required for it, biasing techniques and finally the derived formulas for general representations. Section 2 discusses about the procedure followed to attain the desired results and form a complete comparator. This also explains the reason for more accurate and precisive data outcome as it discusses about how the numbers are broadly classified into various aspects of representation.

II. Previous Work

The upgrade from a fixed radix to a floating radix started in the late 1940s. In the early computing days fixed point arithmetic (FPA) was taken into consideration for the data treatment. Initially, these arithmetic duties were effectively entertained by fixed point number operations, but the increase in data led to the involvement of floating operations and then it overtook those computing operations. With the introduction of Floating-point comparators, the upgrade of comparators in various fields was considered and hence different innovative comparators were introduced. The design represented IEEE 754 standardised implementation making it compatible with handling special cases like infinite numbers exceeding the real number range.
Using Verilog HDL and Xilinx for synthesis, Rony Antony P et al. (2016) & [1] created a double precision floating point comparator. The design represented IEEE 754 standardised implementation making it compatible with handling special cases like infinite numbers exceeding the real number range. The comparison module contained features like a decision handling module followed by the treatment of exception handling.
In 2006 Anuja J. Thakkar et. al & [2] introduced a pipelined design of double precision divider and square root unit. This implementation overpassed all the architectural balances and provided a all-round feature which would support any FPGA. As the IEEE 754 standards were renounced then so they had a better guidelines structure for implementation.
The introduction to the comparator is not new. As in 2005 James E. Stine et. al & [3] developed a comparator contain floating point implementations. Using IEEE 754 rules, hybrid two's complement and floating-point comparator is created in AMI C5N CMOS technology. Comparing 32-bit and 64-bit data with the 2's complement was compatible with the comparator.
Laixiong Wang et. al & [7] In their paper discusses about the algorithm implementation and comparison of floating-point multiplier. This research gives a brief about how variations can be implemented using floating points. When a floating point multiplier is not needed, the suggested method offers a better trade-off between hardware complexity and latency than alternatives that are currently available.
A comparator u sing single precision floating point comparator was proposed by A. T. Samuel, J. Senthilkumar et al. & [8]. Verilog is used to simulate this design utilizing the CADENCE ENCOUNTER tool and TSMC 180nm technology. This brings to mind the IEEE 754 standards guidelines. For improved performance, the NOR-NAND gate is replaced with an OR gate in the performance.
Prabhjot et.al & [16] looks at different floating point arithmetic units implemented using the IEEE-754 standard Arithmetic circuits are an important class of circuit in the digital system. Since the invention of the FPGA, increasing their size and performance has allowed designers to use FPGA for more complex designs. A floating-point representation based on the IEEE754 standard can be used to represent very big and very small numbers, as

they are difficult to express with a fixed number point unit. The unit of measurement used for many mathematical operations, including addition and subtraction, is the floating-point unit.

Low-cost FPGA floating point arithmetic circuits for all common math operations, such as addition and subtraction, multiplication, division, and square root, are presented by Paschalakis et al. & [17]. These circuits can be very helpful in FPGA implementations of complicated systems that need an arithmetic unit that is specifically designed for the device but also benefit from the parallelism and reprogramability of the FPGA device.

For effective floating-point comparison, a double-precision floating-point comparator design is suggested by P. Rony Antony et al. & [18]. Utilizing the full potential of the parallel prefix tree design is this comparator. It starts by comparing the most important bit and only goes on to the least important bit if the comparisons show that the two bits are equivalent. IEEE 754 is the foundation upon which floating-point numbers are represented.

Govindu et.al & [19] examines FPGA usage in high-performance and scientific computing, focusing on floating-point multiplier and adder/subtractor units. Through deep pipelining, they achieve over 240MHz (single precision) and 200MHz (double precision) throughput rates. For matrix multiplication, an FPGA reaches 15GFLOPS (single precision) and 8GFLOPS (double precision). FPGAs can improve GFLOPS/W by up to 6x compared to general-purpose processors, impacting energy-efficient architecture design.

A unique comparator design, as presented by Stine et al. & [20], combines IEEE 754-compliant floating-point operations with two's complement. In contrast to earlier designs, this comparator preserves efficiency in terms of size and speed by integrating both operand types into a single unit. A special magnitude comparator with logarithmic delay is included in the design, along with extra circuitry to handle floating-point and two's complement operands with ease. In compliance with IEEE 754, the comparator fully allows comparisons between 32- and 64-bit floating points.

A low-power, high-speed dynamic comparator for digital circuits and analog-to-digital converters is introduced by Yao Wan et al. & [21]. It lowers delay and power consumption by substituting a more effective gate-biased latching stage for the traditional cross-coupled inverter layout. According to experimental data, this novel comparator may achieve amazing sampling speeds of up to 2 GS/s while operating at 1.2V and consuming only 110-fJ each comparison. The breakthrough is in improving the initial transconductance, which enables quicker comparisons and lower energy usage. An effective multiple-precision floating-point fused multiply-add (FMA) unit is shown in this paper, with support for half, single, double, and quadruple precision operations, among other precision levels.

A system that can perform one quadruple-precision operation, two double-precision operations in parallel, four single-precision operations in parallel, or eight half-precision operations every clock cycle was proposed by Hao Zhang et al. & [22]. Its versatility is further enhanced by the ability to perform dot-product operations and mixed-precision FMA with little additional space. Compared to existing designs, this FMA unit offers a significantly expanded range of supported operations, including half-precision FMA, with just a modest 10.6 percent increase in area footprint, enhancing its appeal for various computational tasks.

Ata Khorami et.al & [23] introduces a low-power comparator design featuring pMOS transistors in both the preamplifier and latch stages, controlled by a specialized local clock generator. The unique delay in latch activation ensures optimal preamplification gain and reduced power consumption. Small cross-coupled transistors enhance preamplifier gain and reduce latch input common mode, improving pMOS transistor activation speed. The comparator also offers a wide input voltage range at high frequencies (500 MHz).

Juzhe Li et.al & [24] presents a high-speed, high-precision time-domain comparator integrated circuit. The design combines a voltage-controlled oscillator (VCO) circuit and a time amplifier to accurately convert analog input voltage into a precise time-domain phase difference. By incorporating VCO feedback and a phase detector, this comparator rapidly compares input signals. Implemented the SMIC 0.18 μm CMOS process technology and simulate results to demonstrate performance: it operates on a 1 V power supply, providing 0.1 μV accuracy, a maximum response speed of 181 MHz, and consumes only 86 μW of power. This advancement offers a promising solution for applications requiring high-speed and high-precision comparisons.

Mayur Agrawal et.al & [10] introduces a novel adaptive algorithm with low computational complexity, employing minimum and maximum variance criteria and normalized cross-correlation to enhance image quality across different imaging depths. It also presents an efficient hardware architecture based on IEEE single precision arithmetic. Evaluation using the Field II toolbox and medical ultrasound data indicates that this approach offers superior resolution and contrast compared to traditional cross-correlation dual apodization techniques.

III. METHODOLOGY

The proposed comparator is comprised of attaining more accuracy as the results derived can be directly utilized for reducing more interactions of hardware's for attain desired outputs. All the standards are justified in accordance with the predefined representations by the IEEE these representations consist of deriving the binary display. The comparator contains some in general modules for classification.

A. IEEE 754 representations

IEEE derived a standardised method for representation of real numbers in terms of binary bits for data processing virtually in computers. The basis of this representation consists of significand bits also called as mantissa, exponent bits and a sign bit to represent the nature in positive or negative numbers. These values on summation gives us the total count of the binary bits required to form a virtual representation. Each set of binary values is uniquely proposed with the names, they are as follows:

TABLE I. REPRESENTING THE COMMON NAMES FOR DIFFERENT BITS

Binary bits	**Names**	**Significand bit (Mantissa)**	**Exponent bit**	**Exponent bias**
Binary 16	Half Precision	11	5	15
Binary 32	Single Precision	24	8	127
Binary 64	Double Precision	53	11	1023
Binary 128	Quadruple Precision	113	15	16383
Binary 256	Octuple Precision	237	19	

The organization of the display of how the computing system considers both a single and a double precision while using data is a significant derivation. The representation consists of combination of significand bit exponent bit and a sign bit. The exponent bias also plays an important role as it allows you to treat the exponent as an unsigned integer for better comparisons.

Single Precision Floating Number Representation:

Single precision is a common way of representing the real number in a 32-bit binary format for computer memory to understand. The representation is presented by IEEE 754 standard. The composition required to form the single precision consists of the following bits:

1. Sign bit (1 bit): This bit classifies the real value into positive or negative domains
2. Exponent bits (8 bits): This represents the power to which the 2 is raised which also signifies the order of magnitude of number.
3. Significand bits: the mantissa represents the fractional part of the number.

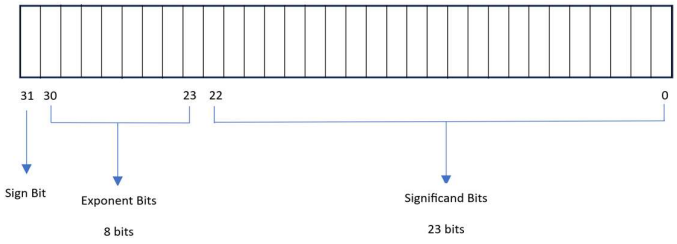


Fig 2. Representation of 32 bits utilization Single precision representation

The representation of double precision floating point holds the similar arrangements where the sign bit hold 1 bit for deciding the nature of the number, 11 bits exponents for the magnitude of the number and 52-bit mantissa for fractional portion of number.

The IEEE 754 resides some rules according to which the numbers are classified into special values or normalized values, this also helps you limit your algorithms to attain higher accuracy. As mentioned, a number consists of exponents(E), mantissa(M) and a Sign bit(S). The following conditions are used for the derivation of the number into floating number keeping IEEE 754 in accordance.

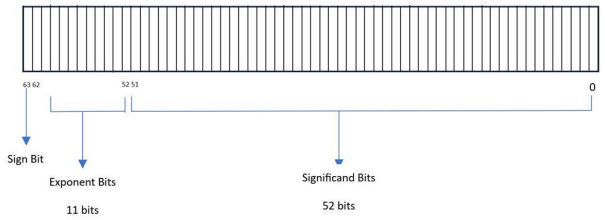


Fig 3. Representation of 64 bits utilization Double precision representation

TABLE II. CONDITIONS FOR FORMING A DOUBLE PRECISION, ALSO SHOWING THE CONDITIONS FOR SPECIAL VALUES

S (1 bit)	E (11 bits)	M (52 bits)	Representation
0 or 1	E=0	M=0	0
0 or 1	E=2047	M=0	Positive or negative infinities
0 or 1	1<=E<=2046	M not 0	Implicit Normalized form
0 or 1	E=0	M not 0	Fractional Form
0 or 1	E=2047	M not 0	NaN

The table 2. is a representation of how the bits are used for the formation of virtual values. NaN or not a number is represented by setting exponent bits (all 11) to 1 with an exception for more special NaN values, the mantissa bits contain any non-zero combinations. The positive infinity and negative infinity have just sign bits in different, rather both share the same condition for its visualization. The mantissa bits are set to 0 and the exponent bits to 1. In case of subnormal numbers, they are represented by setting all exponent bits to 0, indicating that the number is subnormal number. the special condition is that there is always a hidden bit in the mantissa which is always zero. Zero is also considered to be a special value and it is represented by setting mantissa sign bit and exponent bit to 0. These are considered to be the condition or parameters used for calculation of floating-point comparators.

Special Values in floating point numbers

- Positive or Negative Infinity: - these values are used to treat the overflow values or numbers, giving another advantage of using floating numbers.
- NaN (not a number): - NaN consists of all those values that can't be counted down in number formats or represented in the standardized ways.
- Denormalized Numbers: -These numbers represent the forms where all the exponents are zeroes. They are used for underflow numbers, numbers with less precision.

B. Converting Floating-point Numbers from Real Numbers

A number can has to converted into double precision floating point number from a real value to attain higher precision as compared to the others. As the computing operations supports the floating-point number as compared to fixed point numbers it gets necessary to convert a real value to floating value. The conversion can be broadly classified into 3 steps which are as follows: -

- Convert Decimal number to Binary values
- The conversion of binary representation to normalized form
- Representation in double precision floating number.

The fig 5. shows how the 64-bit binary representation are formed in the software's. It contains three major notations sign bit(S), 11-bit excess-1023 exponent(E') which ranges from [52,62] and 52-bit mantissa fraction(M) ranging from [0,52].

Conversion of Decimal value to binary is necessary as it contains the required information for arithmetic operations in a simpler and easier way of representation which allows sorting of a vast amount of data into floating-point. If the real number consists of a decimal value the conversion is treated separately for the values on the right as well as on left of radix and then it is replaced with their subsequent values with same position taken into considerations. This is the first step of conversion.

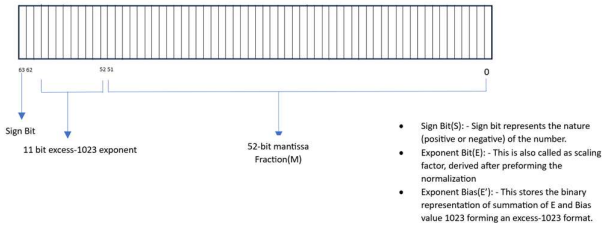


Fig 5. This represents the way the real numbers are represented in floating numbers after conversion

The second step provides a better representation by introducing normalization form. The need of normalization can be satisfied with the explanation that the free flow of radix can form many numbers with different representations storing same values which can be projected correctly but sometimes the system requirement can be insufficient for them to perform operations so a particular and universal representation has to be derived and this representation decides the placement of radix. There are two subdivisions on the basis of which the normalization can be performed: -

- Explicit Normalization: - In these types of normalization, the radix point is shifted towards the most significant left-hand side of the '1' in the binary sequence derived.
- Implicit Normalization: - In this type of normalization, the radix point is made to move to the RHS of the most significant '1' in the expression.

Amongst both the Implicit Normalization is prioritized over the other as it provides you with extra precision.
Step 3 involves the proper representation in floating format. The representation requires a Sign bit S which is decided prior according to the given number as it provides you with the details of the nature of the number. The exponent details, which are derived by performing Implicit normalization over the binary representation. The value of E depends upon the exponent or power of the base considered that is 2. This value is then added to bias value of 1023 which gives a exponent bias value but with the base $_{(10)}$. This is in the form of decimal value so it is converted into base (2) floating.

Step 1: - A Decimal Number is converted into a Binary representation

Decimal Number ⟶ Binary Representation

Step 2: - Conversion into Normalized form

Binary number: - 1xxxxx.xx-------

Performing Implicit normalization: -

$M \times R^E$ - 1.xxxxxx------- X 2^E ..... (E=Exponent value)

Step 3: - Double precision representation

S=0 or 1 (Positive and negative)

E=value derived from normalization (Power with base 2)

E'=E+1023 (This will give base $_{(10)}$ value so convert it to base $_{(2)}$ value)

S	Exponent Bias	Mantissa

Fig 6. Explaining the conversion of real values to binary bits values

C. Floating point comparator with double precision

The double precision floating point comparator compares two 64-bit real numbers which are represented in floating-point numbers and classifies them into special values categories, to be precise it helps you classify into NaN format, denormalized number format, other overflow and underflow format of the numbers with a comparison of equal to or greater than or less than values predictions. This comparator not only helps you classify the dominant magnitudes but helps you differentiate it in
different number formats which then can be used for more accuracy and reduce a few or more system requirements for arithmetic operations.

The flow starts with the inputs stored in the I/O Port Bus which holds the properties of the floating-point numbers that is the mantissa, exponent, and the sign bit forming a 64-bit I/O Port Bus. Fig shows the pictorial representation of the bus. This bus is then connected to RTL Operators designed for performing other RTL operations such as comparing if the number is a Special value, denormalized value, equal, less, greater to each other, NaN value, denormalized value, overflow, and finally underflow value.

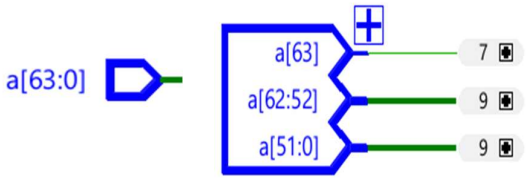
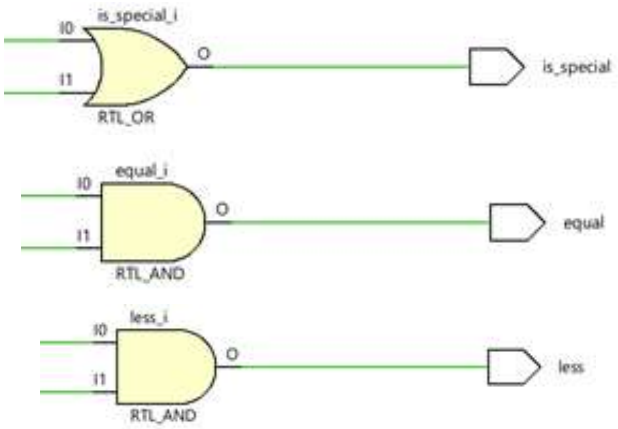


Fig 7. Shows the properties of the I/O Port Bus

Each comparison requires some limiting parameters for the calculations which then according to Fig is compared and decided whether the number falls in either of the criteria.

The special values consist of calculations of whether the number falls in the Infinite criteria or not. The values of A(1st floating point number) and b(2nd floating point number) are then compared to the predefined parameters of the positive or negative infinities which are the limiting values of the 64-bits.

The positive infinity is parameterized with the value of 64'h7FF0000000000000, negative infinity stores the value of 64'hFFF0000000000000, followed by positive and negative forms of NaN storing values 64'h7FF8000000000000 and 64'hFFF8000000000000. The equal operator consists of comparisons of every bit of the mantissa the exponent and sign bits as the numbers may be equal but the sign values may differ, and it uses AND RTL operators for the comparisons as every value must be equal, if not then they are not equal numbers. The greater than and less than use the same formulation that compares every bit of mantissa exponent and sign bits. The operators use OR RTL operators with the combination of EQ (equal operators) for triggering if the number satisfies the given conditions but the main point to be noted is that the exponent bits must be equal to the other as it stores the power of the base and if it differs then the number will face issue comparing while comparing each other. Is NaN value or the Not a Number comparison consists of a comparison of exponent bits with the 11'b11111111111 or 2047 representation in decimal notations with one more condition which must hold that the mantissa portion is not equal to zero according to the [table 2]. Denormalized values consist of two parameters one is comparing exponent with the value 11'b00000000000 and mantissa must not be equal to 0 value, it can store any value as it stores all significand values. Implementation of overflow values makes it quite complex for calculations as it stores many combinations for a number to be declared overflow or underflow. The exponent of A must be equal to 11'b11111111110 followed by the combination that the exponent of B must not attain a value of 11'b11111111111. If A is taken into consideration, then the conditions follow the rules that the mantissa of A must be greater than the mantissa of B and vice versa similarly if underflow B is taken into consideration, then all the Above conditions are to be satisfied but with an exponent of B and mantissa of B respectively. This comparator stores the parameters according to the IEEE. The testbench for this holds the 4 conditions if A is greater smaller equal to B or the 4th condition that the values store NaN values.



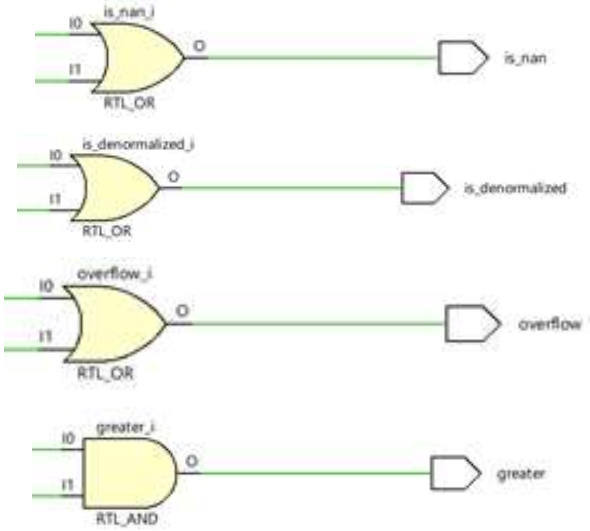


Fig 8. Schematic output of the circuit

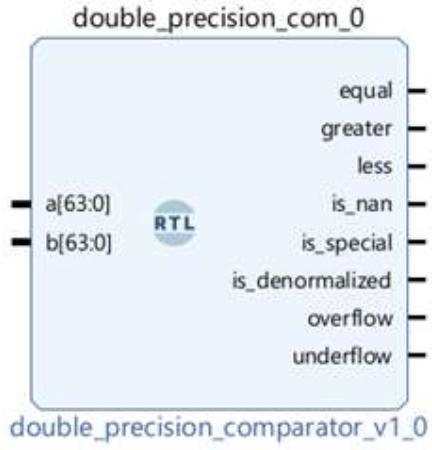


Fig 10. Showing the circuit in black box

Fig 8. can attain a clearer picture by understanding that according to fig 9 the connection in the circuit is a form of black box, where the inputs are form of 63-bits floating values and then these values through RTL analysis gives the 8 desired output and these outputs are used to classify the inputs.

IV. Results and Discussion

Designed in Vivado 2023.1, the double precision floating point comparator is implemented in Verilog. The figure below displays the outcomes of the simulation.

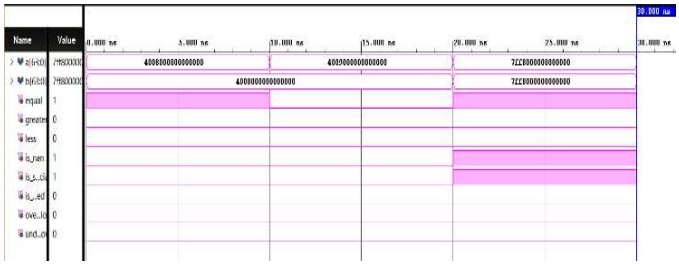


Fig 9. Result showing the comparison of testbenches

The figure fig 9. shows Two initialized 64-bit numbers which are equal in values. When converted into decimal these values depict a non-real value that is infinity. So the wires to get triggered must be of a special Value and a NaN. Now coming to the magnitude of the values, as our system is capable of attain more precise calculative analysis it gives the logic 1 for the equal wire as well as A stores 7ff8000000000000 and B also stores the same value. This post implementation simulation requires a proper testbench simulation to achieve desired results. This comparison is executed within 30,000 ps.

The simulation result can be used to derive the final conclusion that the system is capable of not only performing a single task of comparison but also classifies it into a division that allows it to more precisely provide the type of value inputs the user is working with. This reduces the addition of moreover other integrations for attaining that precision and the compatibility can be a major part of concern.

The figure above is a schematic of the implemented design of a network of wires the circuit must require to attain the same comparator. This circuit contains a total of 636 Cells 136 I/O Ports and 766 Nets.

Circuits can be identified as good or bad in the terms of power consumption. The total power required for this circuit is 3.589 W. This power can be considered to be a good analysis as a normal comparator of floating points requires a power of ~3 W, but our circuit is capable of providing a more detailed view of different forms of numbers and so the requirement of power is obvious.

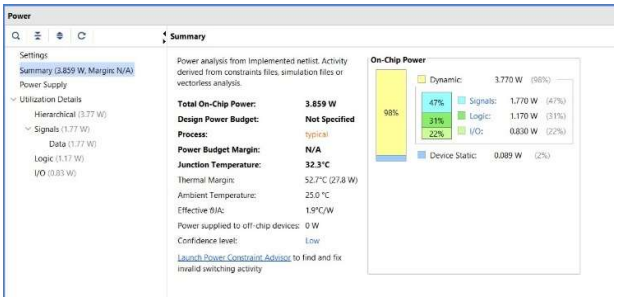


Fig 11. Shows the power consumption report with a brief analysis of other features

The above analysis shows the total on-chip power to be 3.859 W. where the design power budget is not defined for the system. The junction temperature rises to 32.3°C. the "junction temperature" refers to the estimated temperature of the FPGA's semiconductor junctions or cores. Accurate estimation of junction temperature is important for understanding how the FPGA operates under different thermal conditions, which can affect its performance and reliability. The thermal margin is set to 52.7°C or 27.8 W. thermal margins in power analysis refer to the safety margins calculated to ensure that an FPGA device remains within its specified operating temperature range during operation. When an FPGA operates, it generates heat, and if the temperature exceeds the recommended range, it can affect the device's performance, reliability, and longevity. Therefore, thermal analysis is essential to avoid overheating. Ambient temperature refers to the account for temperature variations when estimating power consumption of an FPGA design. The ambient temperature can have an impact on the power dissipation of the FPGA, and it's important to include it in power analysis for accurate results. The proposed comparator holds an ambient temperature of 25.0°C. The effective "θJA" is measured to be 1.9°C/W. "θJA" refers to the thermal resistance from the FPGA junction (the internal silicon die) to the ambient temperature. This thermal resistance is crucial in estimating the power dissipation and temperature rise of the FPGA.

REFERENCES

[1] Kaur, Prabhjot, Rajiv Ranjan, Raminder Preet Pal Singh, and Onkar Singh. "Double Precision Floating Point Arithmetic Unit Implementation-A Review." International Journal of Engineering Research & Technology (IJERT) 4, no. 7 (2015): 992-994.

[2] Paschalakis, Stavros, and Peter Lee. "Double precision floating-point arithmetic on FPGAs." In Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT)(IEEE Cat. No. 03EX798), pp. 352-358. IEEE, 2003.

[3] Even, Guy, and P-M. Seidel. "A comparison of three rounding algorithms for IEEE floating-point multiplication." IEEE Transactions on Computers 49, no. 7 (2000): 638-650.

[4] Govindu, Gokul, Ling Zhuo, Seonil Choi, and Viktor Prasanna. "Analysis of high-performance floating-point arithmetic on FPGAs." In 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., p. 149. IEEE, 2004.

Power Supply

Supply Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)	Budget (A)	Margin (A)
Vccint	1.000	3.481	3.452	0.029	Unspecified	NA
Vccaux	1.800	0.038	0.026	0.012	Unspecified	NA
Vcco33	3.300	0.000	0.000	0.000	Unspecified	NA
Vcco25	2.500	0.000	0.000	0.000	Unspecified	NA
Vcco18	1.800	0.152	0.151	0.001	Unspecified	NA
Vcco15	1.500	0.000	0.000	0.000	Unspecified	NA
Vcco135	1.350	0.000	0.000	0.000	Unspecified	NA
Vcco12	1.200	0.000	0.000	0.000	Unspecified	NA
Vccaux_io	1.800	0.000	0.000	0.000	Unspecified	NA
Vccbram	1.000	0.000	0.000	0.000	Unspecified	NA
MGTAVcc	1.000	0.000	0.000	0.000	Unspecified	NA
MGTAVtt	1.200	0.000	0.000	0.000	Unspecified	NA
MGTVccaux	1.800	0.000	0.000	0.000	Unspecified	NA
Vccadc	1.800	0.020	0.000	0.020	Unspecified	NA

Fig 12. Explaining the power supply chains of the system

I/O

Utilization	Name	I/O Type
0.83 W (22% of total)	double_precision_comparator	
0.256 W (7% of total)	a	HR
0.256 W (7% of total)	b	HR
0.154 W (4% of total)	greater	HR
0.154 W (4% of total)	less	HR
0.003 W (<1% of total)	is_denormalized	HR
0.003 W (<1% of total)	is_nan	HR
0.002 W (<1% of total)	overflow	HR
0.002 W (<1% of total)	underflow	HR
<0.001 W (<1% of total)	is_special	HR
<0.001 W (<1% of total)	equal	HR

Fig 13. Representing the I/O utilization of the power in the circuit

[5] Stine, James E., and Michael J. Schulte. "A combined two's complement and floating-point comparator." In 2005 IEEE International Symposium on Circuits and Systems, pp. 89-92. IEEE, 2005.

[6] Wang, Yao, Mengmeng Yao, Benqing Guo, Zhaolei Wu, Wenbing Fan, and Juin Jei Liou. "A low-power high-speed dynamic comparator with a transconductance-enhanced latching stage." *IEEE access* 7 (2019): 93396-93403.

[7] Zhang, Hao, Dongdong Chen, and Seok-Bum Ko. "Efficient multiple-precision floating-point fused multiply-add with mixed-precision support." *IEEE Transactions on Computers* 68, no. 7 (2019): 1035-1048.

[8] Khorami, Ata, and Mohammad Sharifkhani. "A low-power high-speed comparator for precise applications." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, no. 10 (2018): 2038-2049.

[9] Li, Juzhe, Xu Liu, Jiahui Liu, Di Zhao, Wenrui Yan, and Chengju Bi. "Design of a high-precision time-domain comparator." In *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pp. 239-243. IEEE, 2019.

[10] Agarwal, Mayur, Arijit De, and Swapna Banerjee. "An IEEE single precision floating point Arithmetic-based apodization architecture for portable ultrasound imaging system." *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, no. 6 (2019): 2275-2287.

[11] Wess, Matthias, PD Sai Manoj, and Axel Jantsch. "Neural network based ECG anomaly detection on FPGA and trade-off analysis." In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-4. IEEE, 2017.

[12] G. Murali Krishna, G. Karthick and N. Umapathi. "Design of Dynamic Comparator for Low-Power and High-Speed Applications." Lecture Notes in Electrical Engineering, vol 698. Springer, Singapore.

[13] Zhuang, Haoyu, Can Tong, Xizhu Peng, and He Tang. "Low-power, low-noise edge-race comparator for SAR ADCs." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, no. 12 (2020): 2699-2707.

[14] Saadat, Hassaan, Haris Javaid, and Sri Parameswaran. "Approximate integer and floating-point dividers with near-zero error bias." In *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1-6. 2019.

[15] Barrois, Benjamin, and Olivier Sentieys. "Customizing fixed-point and floating-point arithmetic—a case study in k-means clustering." In *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 1-6. IEEE, 2017.

[16] Harshita Nair, Keerti Puthran, Shivangi Srivastava andSanchali Kshirsagar. “Design and implementation of floating point processor.” In 2020 International Journal of Advance Research, Ideas and Innovations in Technology [IJARIIT], Vol 6, 2020.
[17] E. Schwartz, Revisions to the IEEE 754 standard for floating-point arithmetic, in Proceedings of the 16th Symposium on Computer Arithmetic, 2003, pp.112–112.
[18] Vijaya Krishna Boppana, N. V., and Saiyu Ren. "A low-power and area-efficient 64-bit digital comparator." Journal of Circuits, Systems and Computers 25, no. 12 (2016): 1650148.
[19] Razavi, Behzad, and Bruce A. Wooley. "Design techniques for highspeed, high-resolution comparators." IEEE journal of solid-state circuits 27, no. 12 (1992): 1916-1926.
[20] Yau, Y. T., K. I. Hwu, and C. W. Wang. "Digital PowerControl with Single-Comparator ADC." In 2018 International Symposium on Computer, Consumer and Control (IS3C), pp. 141-144. IEEE, 2018.
[21] Antony, P.R. and Joseph, A.M., 2016. Design and Implementation of Double Precision Floating Point Comparator. Procedia technology, 25, pp.528-535.
[22] A. J. Thakkar and A. Ejnioui, "Design and implementation of double precision floating point division and square root on FPGAs," 2006 IEEE Aerospace Conference, Big Sky, MT, USA, 2006, pp. 7 pp.-, doi: 10.1109/AERO.2006.1655961.
[23] Jaiswal, M.K., Cheung, R.C., Balakrishnan, M. and Paul, K., 2014. Unified architecture for double/two-parallel single precision floating point adder. IEEE Transactions on Circuits and Systems II: Express Briefs, 61(7), pp.521-525.
[24] Cortés-Antonio, P., Rangel-González, J., Villa-Vargas, L.A., Ramírez-Salinas, M.A., Molina-Lozano, H. and Batyrshin, I., 2014. Design and implementation of differential evolution algorithm on FPGA for double-precision floating-point representation. Acta Polytechnica Hungarica, 11(4), pp.139-153.
[25] Paschalakis, S. and Lee, P., 2003, December. Double precision floating-point arithmetic on FPGAs. In Proceedings. 2003 IEEE International Conference on Field-Programmable Technology (FPT)(IEEE Cat. No. 03EX798) (pp. 352-358). IEEE.
[26] A. T. Samuel and J. Senthilkumar, "A novel floating point comparator using parallel tree structure," 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], Nagercoil, India, 2015, pp. 1-6, doi: 10.1109/ICCPCT.2015.7159395.
[27] Babayan-Mashhadi, Samaneh, and Reza Lotfi. "Analysis and design of a low-voltage low-power double-tail comparator." IEEE transactions on very large scale integration (vlsi) systems 22, no. 2 (2013): 343-352.
[28] Miyahara, Masaya, Yusuke Asada, Daehwa Paik, and Akira Matsuzawa. "A low-noise self-calibrating dynamic comparator for high-speed ADCs." In 2008 IEEE Asian Solid-State Circuits Conference, pp. 269-272. IEEE, 2008.
[29] Chaudhari, Siddharth, and Mahesh Pawar. "Notice of Violation of IEEE Publication Principles: Design of efficient double tail comparator for low power." In 2015 International Conference on Communications and Signal Processing (ICCSP), pp. 1782-1785. IEEE, 2015.
[30] Li, Min Jun, Si Ce Wang, Bing Bing Yao, Jun Li, and Lei Qiu. "A High-gain High-precision Dynamic Comparator with Dynamic Cascading Technique." In 2019 Photonics & Electromagnetics Research Symposium-Fall (PIERS-Fall), pp. 2991-2995. IEEE, 2019.