JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A HIGH PERFORMANCE RTL IMPLEMENTATION OF QUANTUM LOGIC

K.Yasmin, Y. Ravi Kiran Varma, S.Vijaya Kumar, M. Saravanan
Electronics and communication Engineering,
Sreenivasa Institute of Technology and Management Studies (Autonomous), Chittoor, Andhra Pradesh, India-517127

Abstract : Applications like processors and others depend on mathematical calculations like square root. After the basic mathematical operations of addition, subtraction, multiplication, and division, square-root is the most useful and important in scientific calculations. The square root circuit is also used to boost the hardware's low power gain. Due to low power, there are additional benefits as well, including quick speed and small area. Area, latency, and power are often kept in balance. Low power consumption, circuit minimization, and speed enhancement are the newest technological trends. In this study, we use a modified XOR gate to modify the Samiur Rahman gate for the difference equation. A 4:1 multiplexer was used in the design of the improved XOR gate. We use Xilinx Vivado tool for synthesis and simulation respectively. On performing the proposed work on the tool, we found the performance parameter of the circuit is increased significantly.

***IndexTerms* - Square root, reversible logic, multiplexer, low power gain, XOR gate, Saimur Rahman Gate, Xilinx Vivado.**

I. INTRODUCTION

Square root is considered to be one of the essential mathematical operations. They are involved in a wide range of applications namely data processing, computer graphics, DSP (digital signal processing) a global positioning system (GPS), and many mathematical calculations. For portable and wireless devices, Low power is the basic necessity. Square rooters in use for many years for various purposes. We use reversible logic in our design to attain low power. The number of input ports and output port is not uniform in Irreversible logics, and therefore it resulted in the loss of information bits. A single bit present in the information dissipates kTln2 Joules of energy which was proposed by Launders Theorem.[1]
To avoid the loss of information bits and energy loss, reversible logic is used.[2]
The loss of heat can be avoided by keeping the number of input and output ports present in the reversible logic in a uniform manner. Power consumption will be reduced in square rooter. A reversible nonlinear feedback shift register will be developed by a low power Authors in.[3]
The calculation of square root can be performed either by restoring or non-restoring using the digit-by-digit technique. The methods proposed by Newton Raphson and Goldschmidt [4] requires more hardware resources and has many steps involved than the digits-by-digits method.
More hardware resources are necessary for restoring the approach. Hardware will be consumed less in the non-restoring process,[5] when compared to restoring practice. So thereby, square root will select non-restoring process.
Many hardware architectures which use irreversible logic has been proposed for digit-by-digit technique. Array-based arithmetic[6] computation can also be able to reduce power.
In calculating square root in,[7] the design of hardware realization of non-restoring algorithm will be done by reversible logic.
The power obtained was high. Computation of Square root requires many reversible logic circuits. In restoring method, the circuit adds the divisor back and put "0" as the circuit's next quotient digit. Whereas in the non-restoring process, there is no mechanism like that. So, the negative remainder and the number "01" are added and radically right things by a supplementary addition later.
A significant part of the existing algorithms, such as BDD based realization, Positive Polarity Reed Mullerare, is restricted to small and medium process though optimized in gate counts and information bits. In contrast, some realization methods succeed in addressing large functions but expensive in terms of additional gate lines, gate counts and quantum price, mainly from the specifications of irreversible logic.
This paper proposes a structured approach for performing the computation circuit. The generation of the reversible embedding, which is an array arrangement of primary blocks, and can apprehend square root networks of any length. To be precise, the standard non-restoring array arrangement of square-root circuit, which operates 2's Complement subtraction regulated by the result of digit-by-digit square root.
The proposed design has a reversible controlled subtractor multiplexer (RCSM) block, which produces 2's Complement calculation, and applied in a modular technique to perform the operation of square root. In Newton-Raphson's method or Heron's method, the solution of square root of a given number is calculated using the formula: $x(k+1) = (1/2)(xk+(a/xk))$ (1) where "a" is a number, whose square root value will be calculated and xk is iterative.

II. EXISTING METHOD

Integer Square Root Using Square and Compare Method
The integer square root is implemented using subtractor and multiplier method. In this each digit in a binary number represents a power of two. By successively rotating through each bit or power of two and testing the result against the desired value, i.e. squaring the guess and checking if it is greater than the original argument, the approximate root gets closer and closer to the actual value. In conjunction with this, the value is achieved quickly. The flow chart for square and compare method is shown in figure 1. For 8-bit integer input, only four program loops required to complete test and to obtain the output. The table 1 gives the steps for an 8-bit integer square root ISSN: 2319-5967 ISO 9001:2008 Certified International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 4, Issue 1, January 2015 106 input A = 1100 0100 (in decimal 196) using square and compare method. The architecture of 8-bit integer square root is shown figure2, where HS and FS are the half subtractor and full subtractor. For 16-bit and higer bits the integer square root needs a multiplier also.
Integer Square Root using Successive Subtraction of Odd Integers
In this method subtraction operation is performed on input, such that successive odd integers are subtracted from the input number until to get the square root.
Integer Square Root Using Modified Non-Restoring Method
In digit-by-digit calculation method, each digit of the square root is found in a sequence where it only one digit of the square root is generated at each iteration. In general, this method can be divided in two classes, i.e. restoring and non-restoring digit-by-digit algorithm
8-Bit Integer Square Root using Modified Non-Restoring Method
Outputs Integer and Decimal Part The block diagram of 8-bit integer square root using modified non-restoring method with integer and decimal part outputs
Disadvantages:
1. Delay due to reduction stage.
2. Lesser efficiency

III. PROPOSED METHOD

Digit by digit procedure is used for obtaining a particular binary number's derived square root. Restoring algorithm and non-restoring algorithm are classified from digit G-by-digit procedure. The system's total power is high using the restoring algorithm because it involves a more significant number of hardware resources. When compared to restoring algorithm, the non-restoring algorithm involves a smaller number of hardware resources. The total number of bits N is divided into a group of two digits in the nonrestoring algorithm. Therefore, the length of the Quotient is N/2.
Steps involving in the non-restoring algorithm are as follows:
• Step 1: Separation of the total bits of N into two parts.
• Step 2: From the leftmost group of significant bits we have to subtract '1'. If the subtraction will be positive, then Quotient will be 1, then if the difference will be negative, then the Quotient is 0.
• Step 3: From the next group of two digits, subtract after appending the previous quotient along with "01".
• Step 4: Until we reach till the last of groups of two digits we have to proceed to step -2.
It can be a whole number or radical number in case of radicand. Let us assume that it will be a decimal number, then it will be denoted like 0010.001…And if the radicand is going to be a whole number, then it will be denoted as 1101. In this, 0010 will be linked to 6 and 0011 will be pointed to 0.4 in terms of binary sign. On the basis of requirement the number of bits which are obtained after the decimal point will be prolonged to N that is the total number of bits. Lets consider a binary square rooter (N7 N6 N5 N4. N3 N2 N1 N0). In Quotient for square rooter it will be going to be (U3 U2. U1 U0).
The user can decide the number of bits previously and succeeding the decimal point. For example, the square root of 1101 and 0010.0011 is given in Fig.1. From Fig. 1, the square root of 13 and 2.2 are found. Value for the square root obtained form 13 will be 3.6. From the quotient 11.10(U3 U2. U1 U0),11(U3U2) corresponds to 3 and 10 (U1U0) to 0.6. The representation of 0.6 in the binary system is 1001.Value obtained for the square root of 2.2(0010.0011) will be 1.4. And the value of the Quotient 01.01, 01(U3U2) will be represented to 1 and 01 (U1U0) to 0.4. The representation of 0.4 in the binary system is 0110.
As mentioned, application and user requirement will be deciding count of bits previously and after the decimal point. If there will be a large number of requirement in significant number of bits in the Quotient, then the user shall increases bit size after the decimal point to fulfil the requirement.

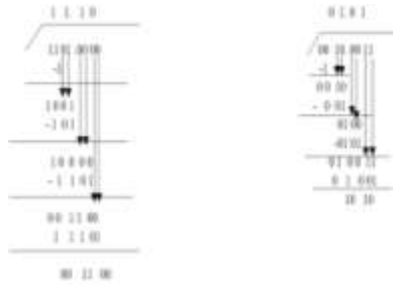
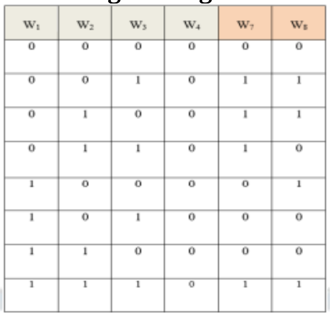


Fig : Non-Restoring Algorithm example

A reversible full subtractor is required for computation process which will be done by non-restoring algorithm. The gates like Feynman Gate and Saimur Rahman gate will be designing the square rooter. a full subtractor will be the serviceability of the Saimur Rahman gate.
Saimur Rahman Gate
A Three-bit (3-bit) subtractor is called a Full subtractor. It has 3 inputs, W1, W2, W3 and another input, W4=0 and two outputs difference as W8 and borrows as W7. The mathematical expressions for difference and borrow are
• Difference (W8) =W1-W2-W3

• Borrow W7=1 if W1< (W2-W3) else W7=0
The logical expressions for difference and borrow are
• Difference (W8) = W1 ⊕ W2 ⊕ W3 ⊕ W4
• Borrow W7=W1’W2⊕W1’W3⊕W2W3
where ⊕ is Logical XOR operation and W1 ‘represents compliment operation of W1.
Only in the case of W4=0, a full subtractor will be used which comes from the SRG gate, and also in case of W4 is not equal to 0, in that case, the gate will not be able to work as full subtractor. Garbage output will be represented by W5 and W6. Table.1 will be representing the truth table of SR gate. The radicands (N7 N6 N5 N4 . N3 N2 N1 N0). with a total length of 8 binary bits. the number of binary bits for the Quotient is half of 8 in the essence of 8/2= 4(U3 U2 . U1 U0) 8-bit square rooter. FG and SR gates will be implemented with the 8-bit square rooter.

Fig: SRG gate



W_1	W_2	W_3	W_4	W_7	W_8
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Table 1: Samiur Rahman Gate (SRG) Truth Table

In this proposed method we have modified the Samiur Rahman gate for Difference equation (w7 equation) using modified XOR gate. The modified XOR gtae is designed by using 4:1 multiplixer.

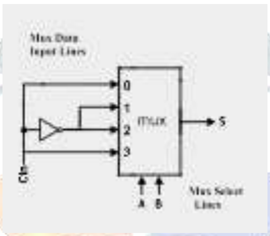


Fig: Modified XOR Gate

Reversible Multiplexer

A multiplexer allows many incoming signals to interact with one device, circuit or resource. Multiplexers also used for the implementation of multiple variable’s Boolean functions. The realization of a Multiplexer using reversible logic gates is called a Reversible multiplexer. The proposed design has RT reversible logic gate multiplexer. If the remainder will be taken as positive, and then the Quotient, i.e. U=1 and the difference will be carried over for the following according to non-restoring computation. But in case if the remainder is taken as negative, then the Quotient will be (U=0), and the previous inputs will then be carried for the following procedure, By using RT reversible gate, a control unit is designed to interchange between the inputs(W1) and the difference(W1). The input(W1), the difference (W8) and the Quotient will give as input to the gates. For the process of switching among the inputs and the difference we use reversible multiplex. Based on the Quotient we can be able to see the configuration which is shown in Fig. 3. The difference and the input will be auto switched. Non-restoring algorithm will be proposed by the design with the help of these reversible gates. The basic logic gates OR, XOR, NOT, AND being applied to the traditional approach. We use irreversible gates for the administration of the conventional design. We can finally obtain the power outcomes of conventional and reversible designs.

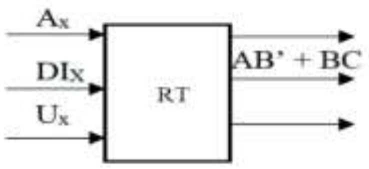
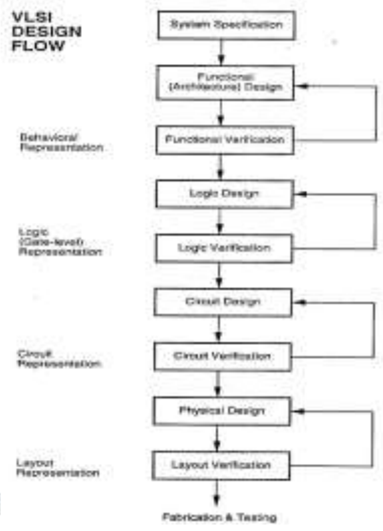


Fig: RT Multiplexer

Figure 4 depicts that the square root of a binary number can be observed with the help of the Modified Non-restoring algorithm by using the digit-by-digit approach. By using the modified non-restoring algorithm, the provided binary digit is primarily separated into collections of two numbers. Soon “1” is deducted from the left-most significant digits (N7 N6), being the initial step. Next, if the deducted number is zero or greater than zero, the quotient(U3) is regarded as “1”, and the RT reversible multiplexer transfers the variation value received of the gate essentially “d” bits (D1D2…) to the next step. Where (D1D2…) are difference bits If the subtracted value is less than zero, i.e. negative, the quotient(U3) is regarded as “0”. The reversible multiplexer transfers the former input data bit to the following step; along with the value received of the previous level, the next couple of bits are considered simultaneously with that value. Those values were saved as “A” bits (A1A2A3…). The quotient from the preceding step and “01” is added and stocked as B bits (B1B2…), and B bits are deducted from A bits, and this process is iterated till all the digits are finished. The Square root of the provided binary number is the terminal quotient, which is the input provided by the user to the circuit. The additional outputs are garbage outputs(G1G2…). Therefore, using reversible gates such as RT reversible multiplexer, Samiur Rahman Gate (SRG) works as a multiplexer and full subtractor, respectively, by following the Non-restoring algorithm, the square root of a binary number is obtained.

IV. VLSI DESIGN FLOW

The VLSI design cycle begins with a formal specification for a VLSI chip, then proceeds through a number of phases to generate a packaged chip.



System Specification:
In every design process, the first step is to define the system s specifications. A system specification is a high-level description of a system. Throughout this process, performance, utility, and physical criteria such as the chip s size must all be take care of. The systems specification is a trade-off between requirements, technological capabilities, and cost efficiency.

Architectural Design
At this step, the system's needed architecture is established. This comprises, among other things, RISC vs. CISC, the number of ALUs and Floating Point Units, pipeline number and structure, and cache size. Architects produce a Micro-Architectural Specification as a consequence of their work.

Behavioral or Functional Design:
This step determines the system's core functional components. The needs for unit connections are also taken into account. Each unit's area, power, and other attributes are calculated.
Modules.
The basic goal is to figure out how each unit behaves in terms of input, output, and time while keeping the underlying structure undetermined.
A timing diagram or other unit relationships are frequently produced as a part of the functional design process.

Logic Design:
This step generates and tests the functional design's control flow, word widths, register allocation, arithmetic operations, and logic operations.
The Register Transfer Level (RTL) description is what this description is known as. A high-level language (HDL) such as VHDL or Verilog is used to create RTL. This description can be used for verification and simulation.

Circuit Design:
The creation of a circuit representation from a logic design is known as circuit design. The Boolean formulas are converted to a circuit representation considering requirements of the original design. Circuit simulation is used to check each component s precision and timing.
The connections between the circuit parts (cells, macros, gates, and transistors) are shown in this diagram. This is referred to as a netlist. At each stage, the logic is examined.

Physical design:
This stage involves converting the circuit representation to a geometric representation. A layout is a geometric depiction of a circuit. Each logic component is converted to a geometric representation which gives the required logic function of the component. To depict relationships between different components, geometric patterns, frequently lines in several layers, are utilized.

Layout verification:
Layout Synthesis methods can generate a layout directly from a netlist, enabling for complete or partial physical design automation. While layout synthesis techniques are rapid, they come at a cost in terms of area and time, which limits their use to specific designs. This has been shown.

Fabrication and Testing:
Silicon crystals are sliced and grown into wafers. The wafer is manufactured and sliced into unique chips at a fabrication plant. After that every chip is packaged and tested to confirm all design specifications and functions works properly.

MODULE:
The most fundamental building unit in Verilog is the module. A single piece or a collection of low-level design components might be used. Components are typically organised into modules to provide common functionality that may be accessed via port interfaces from several portions of the design while masking the internal implementation.

Syntax:
module<module name> (<module_port_list>);
…..

<module internals> //contents of the module
….
Endmodule
Instances
A module serves as a starting point for object creation. When we call a module, Verilog instantiate a new object from the template, complete with its own name, variables, arguments, and I/O interfaces. Instances are the term used to describe these situations.
Ports:
Ports enable a module to communicate with its surroundings.
All modules in a hierarchy have ports, with the exception of the top-level ones.
You can associate ports by name or by order.
Input, output, and inout ports can all be declared. The following is the syntax for a port declaration:
Input [msb_val:lsb_val] input variables;
output[msb_val:lsb_val] output variables;
inout[msb_val:lsb_val] inout variables;
Identifiers
Identifiers are user-defined terms that are used to name variables, functions, modules, and instances. Letters, numerals, and the underscore character can all be used as identifiers.
An identifier s initial character must not be a number. The length of an identifier is completely up to you.
All characters in identifiers are meaningful, and they are case-sensitive.
An identifier can be defined as an escaped identifier if it comprises special characters, starts with digits, or is the same name as a keyword. A backslash character() precedes an escaped identifier, which is followed by a series of characters and white space.
Keywords:
To interpret an input file, Verilog employs keywords.
Unless you use an escaped identifier, you cannot use these terms as user variable names.
Reserved identifiers are used to specify language constructs and are known as keywords.
Always, case, assign, begin, case, end, and end case, among others, are some of the keywords.
Data Types:
There are two primary data types in the VERILOG Language:
Nets - structural relationships between components are represented by nets.
Registers - these are the variables that are utilized to store data.
A data type is assigned to each signal.
Explicitly declared in VERILOG code with a declaration.
Implicit declarations are always of the wire type and are only one bit wide. Used to connect structural building pieces in the code but not explicitly specified.
Register Data Types
Registers preserve the value assigned to them until another assignment statement changes it.
Registers represent data-storage constructs.
The term memory refers to register arrays.
In procedural blocks, variables are data types from the register.
A register data type is required when a signal is provided a value within a procedural block.
The keywords initial and always begin procedural blocks.
Register, integer, time, and real are the data types used in register.

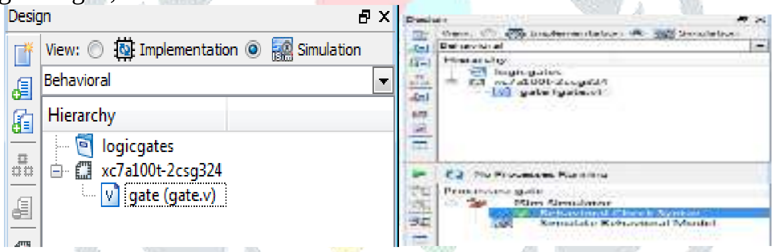
V. XILINX TOOL:

The Fpga Techniques application software is used to develop circuitry employing Xilinx Fpga Array (FPGA) or Cpld Device. (a) model intake, (b) system synthesis and realization, (c) functionality checking and output generation for checking the outcomes, and (d) In the suggested framework for implementing automation, evaluation phase and testing the circuits are the procedures to implement. To build a design models which use the aforementioned computer-aided modeling software, users can use a schematic entryway application, a verilog hardware description (HDL) - Verilog or VHDL, or a mixture of the two.
Header: Name of module which can be anything as per user and inputs and outputs variables for that particular design.
Declarations: the variables of inputs and outputs and any inside connections in the module in terms of reg or wire.
Formulas, finite state, and logically evaluated operators are some of the logic formulations.
Final end of the module
Each of your concepts in this lab must be in the Verilog input file.
Programmable Logic Device: FPGA
The Basys2 board, having device parameters such as a family of Xilinx Spartan3E with device part XC3S250E FPGA having a package of CP132, is considered in this lab. The Spartan line of FPGAs includes this FPGA component. These devices are available in huge variations. We ll use among them Spartan 3E with 250T of CP132 devices, having 132pins in that package.
Starting a project
Double click on Xilinx Navigator Icon on desktop. It will open shown as below figure. On the device type starting with Xc, click on that and create sources to write code which are called as modules of user defined names.
Project opening
File->New Project.
A pop will raise fill them as follows
Name of project is as per our wish means we can give anything with a combination of letters numbers and underscore.
Storing the project in a particular location: This is the directory in which you wish to save the new project in one of your drives. They are stored in c drive in the above window, which is incorrect; software and code should not be placed in the same area.
By selecting NEXT, the following window should appear:

Select as per the requirement and availability of hardware with us.
Device type and Family of board: The FPGA/CPLD used belongs to this family. The Spartan3E FPGAs will be used in this experiment.
Device: The device s unique identifier. You can use the code XC3S250E for this lab
Package type: according to the availability and mentioned package need to be selected. In this lab, the Spartan FPGA is packaged in a CP132 package.
Speed Grade: -4 is the speed grade.
XST [VHDL/Verilog] Synthesis Tool
Simulator: The software that is used to test and replicate the design s functionality. Modelsim- or ISE can be invoked for simulation. After that click next and finish it off.
Related documentaries or files will be automatically saved in a subdirectory which id followed by the project name.
To open existing files go to select file and click on open project and click ok to select the project that need to be opened as per our wish. Which is the window as follows:
A new source pop-up
Select NEW SOURCE from the previous window.
As seen in Figure 4, a window appears.
Select verilog related and name it and finish it by giving next or finish at the end. A default module with name as given with basic syntax will be generated in a v file with given module name if any existing files are getting added make sure to check box of copying them into this project such that it won t get reloaded and change the original content if any modifications are done in the present usage location..
Define Verilog Source window (snapshot from Xilinx ISE software)
Provide the variable names of both input and output and provide width f they have otherwise leave it unchecked box.
Then, above the window, click Next> to bring up a window with all of the new source data. If you need to make any changes, simply click Back to return to the previous screen and make the necessary adjustments. After that it need to followed some nexts with finish to produce default module with basic syntax.
Writing the code in generated V file
The source code would now be visible as per Figure 8. We can even use the text file pane as a text file editor to perform any necessary modifications to the source code generated. All i/o ports are obtained. to save it directly ctrl+s or go to File and there Save it.

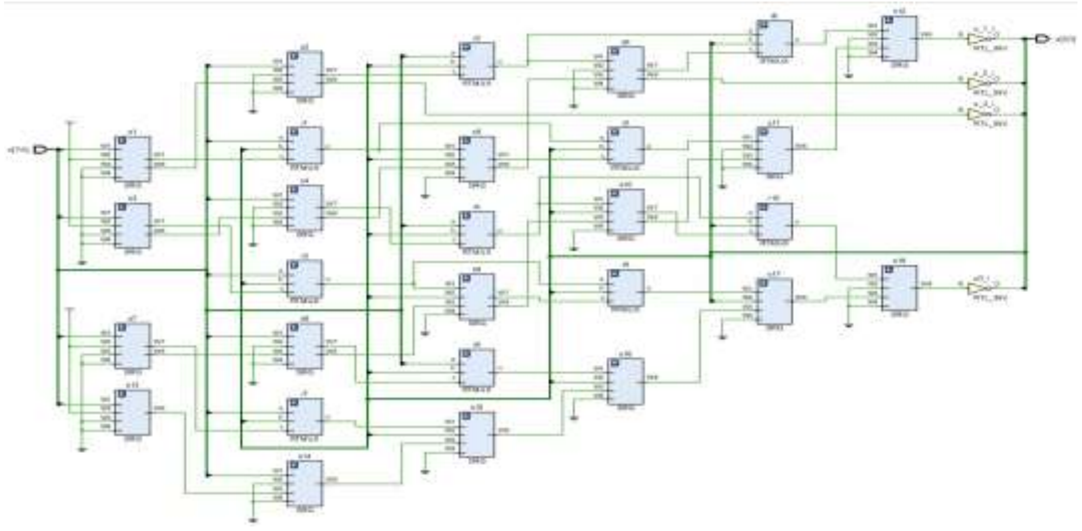
VI. XILINX SIMULATION PROCEDURE:

After the synthesis is completed, we will run simulations to ensure that the implemented design is functional.
As illustrated in the following dialogue, click Simulation and select the module that needs to run.

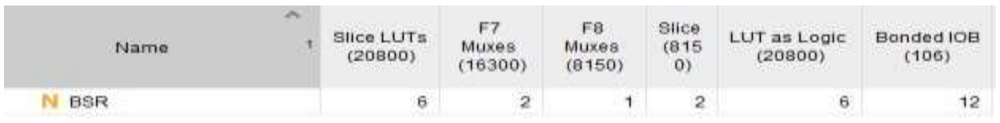


To check syntax errors click two times on behav. Check syntax in simulation under Isim section after that if it shows green tick it indicates no errors in syntax then we can click two times simulation and waveform window can be obtained where inputs are forced and outputs are verified.
After clicking on simulate behavioral model, the following window will appear
If the input values are constant, the simulation widow will appear, and if it is a clock, the simulation widow will appear. Mention the simulation period, run for a set amount of time, and the results will appear in the window below. Compare the output to the specified parameters.

VII. RESULTS

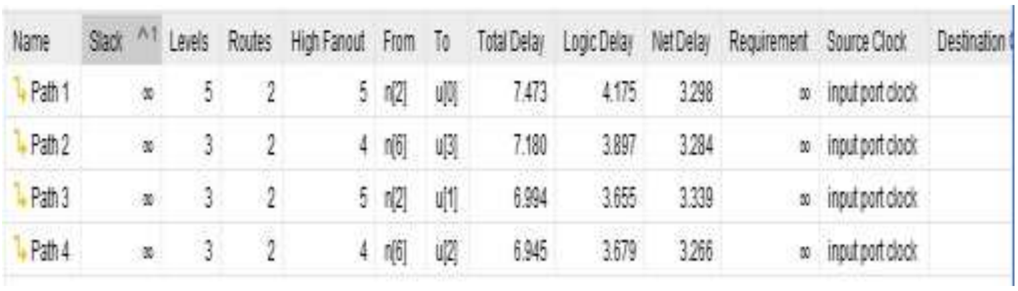


Area



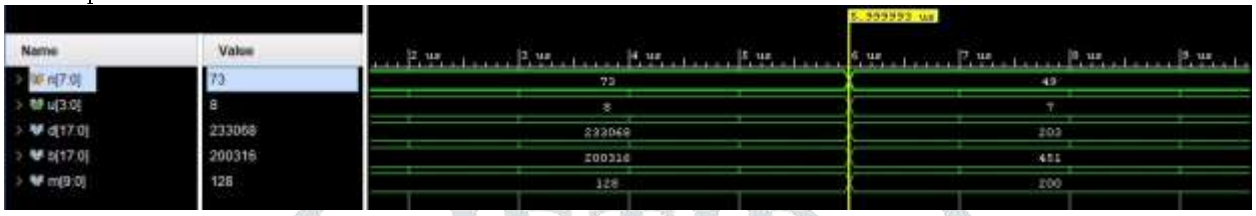
Name	Slice LUTs (20800)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)
N BSR	6	2	1	2	6	12

Delay



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination
Path 1	∞	5	2	5	n[2]	u[0]	7.473	4.175	3.298	∞	input port clock	
Path 2	∞	3	2	4	n[6]	u[3]	7.180	3.897	3.284	∞	input port clock	
Path 3	∞	3	2	5	n[2]	u[1]	6.994	3.655	3.339	∞	input port clock	
Path 4	∞	3	2	4	n[6]	u[2]	6.945	3.679	3.266	∞	input port clock	

Simulated Output



Name	Value
n[7:0]	73
u[3:0]	8
d[17:0]	233068
b[17:0]	200316
m[9:0]	128

VIII. CONCLUSION

An SRG reversible logic-based binary square root implementation has been devised, featuring a modified XOR gate and a foundation on a 4:1 Multiplexer. This novel model exhibits several advantages in contrast to the existing one. It boasts a reduced gate count and enhanced power efficiency in the context of a binary square root implementation reliant on a modified XOR gate. The suggested model offers remarkable flexibility in its resolutions.

For the synthesis and simulation tasks, the Xilinx Vivado Software will be the designated tool. This transition is anticipated to yield a significant reduction in the total gate count. Furthermore, in comparison to the traditional restoring method, the non-restoring operation will make more efficient use of hardware resources. Consequently, the choice is to employ the non-restoring approach for the square root operation.

The successful development of an energy-efficient reversible binary square rooter has yielded all the requisite parameters. In the realm of reversible computing, this square rooter will be harnessed, particularly within ALUs. When juxtaposed with reversible computation employing a modified XOR gate, this transition results in a reduction in gate count and an enhancement in design performance.

IX. REFERENCES

[1] Dr. S. Selvakanmani, Mr. Rajeev Ratna Vallabhuni, Ms. B. Usha Rani, Ms. A. Praneetha, Dr. Urlam Devee Prasan, Dr. Gali Nageswara Rao, Ms. Keerthika. K, Dr. Tarun Kumar, Dr. R. Senthil Kumaran, Mr. Prabakaran.D, "A Novel Global Secure Management System with Smart Card for IoT and Cloud Computing," The Patent Office Journal No. 06/2021, India. International classification: H04L29/08. Application No. 202141000635 A.

[2] Nalajala Lakshman Pratap, Rajeev Ratna Vallabhuni, K. Ramesh Babu, K. Sravani, Bhagyanagar Krishna Kumar, Angothu Srikanth, Pijush Dutta, Swarajya Lakshmi V Papineni, Nupur Biswas, K.V.S.N.Sai Krishna Mohan, "A Novel Method of Effective Sentiment Analysis System by Improved Relevance Vector Machine," Australian Patent AU 2020104414. 31 Dec. 2020

[3] S.V.S Prasad, Chandra Shaker Pittala, V. Vijay, and Rajeev Ratna Vallabhuni, "Complex Filter Design for Bluetooth Receiver Application," In 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, July 8-10, 2021, pp. 442-446.

[4] Chandra Shaker Pittala, J. Sravana, G. Ajitha, P. Saritha, Mohammad Khadir, V. Vijay, S. China Venkateswarlu, Rajeev Ratna Vallabhuni, "Novel Methodology to Validate DUTs Using Single Access Structure," 5th International Conference on Electronics, Materials Engineering and Nano-Technology (IEMENTech 2021), Kolkata, India, September 24-25, 2021, pp. 1-5.

[5] Chandra Shaker Pittala, M. Lavanya, V. Vijay, Y.V.J.C. Reddy, S. China Venkateswarlu, Rajeev Ratna Vallabhuni, "Energy Efficient Decoder Circuit Using Source Biasing Technique in CNTFET Technology," 2021 Devices for Integrated Circuit (DevIC), Kalyani, India, May 19-20, 2021, pp. 610- 615.

[6] Chandra Shaker Pittala, M. Lavanya, M. Saritha, V. Vijay, S. China Venkateswarlu, Rajeev Ratna Vallabhuni, "Biasing Techniques: Validation of 3 to 8 Decoder Modules Using 18nm FinFET Nodes," 2021 2nd International Conference for Emerging Technology (INCET), Belagavi, India, May 21–23, 2021, pp. 1-4.

[7] P. Ashok Babu, V. Siva Nagaraju, Ramya Mariserla, and Rajeev Ratna Vallabhuni, "Realization of 8 x 4 Barrel shifter with 4-bit binary to Gray converter using FinFET for Low Chandrakumar Rasanjani, et al. : Fundamental Digital Module Realization Using RTL Design for Quantum Mechanics 6 Journal of VLSI circuits and systems, , ISSN 2582-1458 Power Digital Applications," Journal of Physics: Conference Series, vol. 1714, no. 1, p. 012028. IOP Publishing. doi:10.1088/1742-6596/1714/1/012028

[8] Vallabhuni Vijay, C. V. Sai Kumar Reddy, Chandra Shaker Pittala, Rajeev Ratna Vallabhuni, M. Saritha, M. Lavanya, S. China Venkateswarlu and M. Sreevani, "ECG Performance Validation Using Operational Transconductance Amplifier with Bias Current," International Journal of System Assurance Engineering and Management, vol. 12, iss. 6, 2021, pp. 1173-1179.

[9] Vallabhuni, Rajeev Ratna, M. Saritha, Sruthi Chikkapally, Vallabhuni Vijay, Chandra Shaker Pittala, and Sadulla Shaik, "Universal Shift Register Designed at Low Supply Voltages in 15 nm CNTFET Using Multiplexer," In International Conference on Emerging Applications of Information Technology, pp. 597-605. Springer, Singapore, 2021.

[10] B. M. S. Rani, Vallabhuni Rajeev Ratna, V. Prasanna Srinivasan, S. Thenmalar, and R. Kanimozhi, "Disease prediction based retinal segmentation using bi-directional ConvLSTMU-Net," Journal of Ambient Intelligence and Humanized Computing, 2021, pp. 1-10. https://doi. org/10.1007/s12652-021-03017-y

[11] Rajeev Ratna Vallabhuni, A. Karthik, CH. V. Sai Kumar, B. Varun, P. Veerendra, and Srisailam Nayak, "Comparative Analysis of 8-Bit Manchester Carry Chain Adder Using FinFET at 18nm Technology," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020, pp. 1579-1583, doi: 10.1109/ ICISS49785.2020.9316061.

[12] R. R. Vallabhuni, P. Shruthi, G. Kavya and S. Siri Chandana, "6Transistor SRAM Cell designed using 18nm FinFET Technology," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020, pp. 1584-1589, doi: 10.1109/ICISS49785.2020.9315929.

[13] Rajeev Ratna Vallabhuni, J. Sravana, M. Saikumar, M. Sai Sriharsha, and D. Roja Rani, "An advanced computing architecture for binary to thermometer decoder using 18nm FinFET," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20-22 August, 2020, pp. 510–515.

[14] Rajeev Ratna Vallabhuni, K.C. Koteswaramma, B. Sadgurbabu, and Gowthamireddy A, "Comparative Validation of SRAM Cells Designed using 18nm FinFET for Memory Storing Applications," Proceedings of the 2nd International Conference on IoT, Social, Mobile, Analytics & Cloud in Computational Vision & Bio-Engineering (ISMAC-CVB 2020), 2020, pp. 1-10.

[15] V. Siva Nagaraju, Rapaka Anusha, and Rajeev Ratna Vallabhuni, "A Hybrid PAPR Reduction Technique in OFDM Systems," 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), Bhubaneswar, India, 26-27 Dec. 2020, pp. 364-367.

[16] V. Siva Nagaraju, P. Ashok babu, B. Sadgurbabu, and Rajeev Ratna Vallabhuni, "Design and Implementation of Low power FinFET based Compressor," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 13-14 May 2021, pp. 532-536.

[17] P. Ashok Babu, V. Siva Nagaraju, and Rajeev Ratna Vallabhuni, "Speech Emotion Recognition System With Librosa," 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 18-19 June 2021, pp. 421-424.

[18] P. Ashok Babu, V. Siva Nagaraju, and Rajeev Ratna Vallabhuni, "8-Bit Carry Look Ahead Adder Using MGDI Technique," IoT and Analytics for Sensor Networks, Springer, Singapore, 2022, pp. 243-253.

[19] Rajeev Ratna Vallabhuni, Jujavarapu Sravana, Chandra Shaker Pittala, Mikkili Divya, B.M.S.Rani, and Vallabhuni Vijcaay, “Universal Shift Register Designed at Low Supply Voltages in 20nm FinFET Using Multiplexer,” In Intelligent Sustainable Systems, pp. 203-212. Springer, Singapore, 2022.

[20] V. Vijay, J. Prathiba, S. Niranjan Reddy, V. Raghavendra Rao, “Energy efficient CMOS Full-Adder Designed with TSMC 0.18µm Technology,” International Conference on Technology and Management (ICTM-2011), Hyderabad, India, June 8-10, 2011, pp. 356-361.

[21] Vallabhuni Vijay, Pittala Chandra shekar, Shaik Sadulla, Putta Manoja, Rallabhandy Abhinaya, Merugu rachana, and Nakka nikhil, “Design and performance evaluation of energy efficient 8-bit ALU at ultra low supply voltages using FinFET with 20nm Technology,” VLSI Architecture for Signal, Speech, and Image Processing, edited by Durgesh Nandan, Basant Kumar Mohanty, Sanjeev Kumar, Rajeev Kumar Arya, CRC press, 2021.

[22] P. Chandra Shaker, V. Parameswaran, M. Srikanth, V. Vijay, V. Siva Nagaraju, S.C. Venkateswarlu, Sadulla Shaik, and Vallabhuni Rajeev Ratna, “Realization and Comparative analysis of Thermometer code based 4-Bit Encoder using 18nm FinFET Technology for Analog to Digital Converters,” In: Reddy V.S., Prasad V.K., Wang J., Reddy K.T.V. (eds) Soft Computing and Signal Processing. Advances in Intelligent Systems and Computing, vol 1325. Springer, Singapore. https://doi.org/10.1007/978-981-33-6912-2_50

[23] Rajeev Ratna Vallabhuni, S. Lakshmanachari, G. Avanthi, and Vallabhuni Vijay, “Smart Cart Shopping System with an RFID Interface for Human Assistance,” 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020, pp. 165-169, doi: 10.1109/ICISS49785.2020.9316102.

[24] Saritha, M., Kancharapu Chaitanya, Vallabhuni Vijay, Adam Aishwarya, Hasmitha Yadav, and G. Durga Prasad, “Adaptive And Recursive Vedic Karatsuba Multiplier Using Non Linear Carry Select Adder,” Journal of VLSI circuits and systems, vol. 4, no. 2, 2022, pp. 22-29.