A Low Power Binary Square Rooter using Reversible Logic and Modified Non-Restoring Algorithm

Bharath AR[1], Manikandan M[1], Karthick J[1] and Ms. Sindhu S[2]

[1]Students, B.E. Electronics and Communication Engineering
[2]Assistant Professor, Department of Electronics and Communication Engineering
Meenakshi Sundararajan Engineering College, Kodambakkam, Chennai, Tamil Nadu, India.

---***---

Abstract- *Square root computation is an important mathematical operation which has wide range of applications. The design of square rooter needs to achieve low power, low area and high speed. Often there can be a trade-off among the three metrics. As the current technology aims for low power, designs require major architectural modification. This paper presents a low power binary square rooter using reversible logic and modified non-restoring algorithm. It uses reversible logic to achieve low power. The binary square rooter is designed and implemented using RCSM (Reversible Controlled Subtract Multiplexer). For further development such as number of quantum cost, garbage outputs and the constant inputs , binary square rooter is implemented using SRG (Samiur Rahman Gate).Binary square rooter using non-restoring algorithm is designed using both SRG and conventional approach. Simulations are carried out using ModelSim software and the power is obtained using X-Power Analyser. The power obtained for SRG and conventional technique are compared.*

Key words: **Square rooter, low power, reversible logic, conventional logic, modified non-restoring algorithm.**

1. INTRODUCTION

Square root is a vital mathematical operation which has a lot of applications. Square rooters are used in computer graphics, global positioning system (GPS), digital signal processing (DSP) computations, mathematical calculations and data processing. In the present world of developments in various fields, the main novelty in the field of VLSI is Low Power. In order to attain low power Reversible logic is used in this design. In Irreversible logics, the amount of output port and input ports are not uniform and therefore the will be a loss of information bits. According to Launders kTln2 Joules of energy is dissipated for a single bit information. In order to avoid the energy and information loss, reversible computation has to be reversible. In reversible logic, the amount of output ports and input ports are uniform and so the loss of heat can be prevented. It reduces the consumption of more power. Authors in developed a low power reversible nonlinear feedback shift register. Square root methods such as Newton Raphson method, Goldschmidt methods are complicated as it involves more steps and hardware resources when compared to the digit-by-digit method. The square root calculation using digit-by-digit method can be performed either by restoring method or non-restoring method. In restoring method more hardware resources are required whereas in non-restoring method it consumes less hardware compared to restoring method. Hence non-restoring algorithm is preferred. Many hardware architectures for digit-by-digit square root calculation using irreversible logic has been proposed. There are many reversible logic circuits that are used for the computation. Power reduction can also be done using array based arithmetic computation. Reversible logic is used to design hardware realization of non-restoring algorithm for computation of square root. Power obtained was high. Reversible circuits emerge as a plausible alternative to classical designs due to their information loss-less computation performed with considerably less energy. Their close relation to quantum circuits elevated the ongoing research interests. Further, they find applications in vast areas of computing such as low power designs, adiabatic circuits, cryptography, digital signal processing and optical computing. A reversible function realizes a unique one-to-one mapping of inputs to outputs, while providing no data loss during the computation. So far, the synthesis of reversible circuits from irreversible or reversible specification has been done intensively and finding an optimized reversible realization is still a challenging issue. Moreover, a great part of the existing algorithms are restricted to small functions but optimized in garbage bits and gate counts, while some approaches successfully address large functions though quite costly in terms of gate count, additional lines and quantum cost, especially from irreversible specification which include Positive Polarity Reed Muller, BDD based, ESOP based realization. In addition, a great effort has been made towards implementing the basic reversible arithmetic units such as adders, subtractors and multipliers by finding a direct translation from classical to reversible forms In scientific computations next to basic mathematical operations of addition, subtraction, multiplication and division, square-root is most useful and vital. For example, numerical analysis, complex number computations, statistical analysis, computer graphics and signal processing are among the fields where square root is of relevance. Although, the realization of a square root in classical circuits is well established, the way of implementing this operation in emerging technologies is not yet sound. We find only one example of reversible 8-bit square root circuit as benchmark result. However, this is a discrete example of square root operation, not a regular structure or generalized method of building reversible square root circuit. In this paper, we propose a structured methodology of implementing this arithmetic circuit. Based on its classical realization, we generate the reversible embedding which is an array structure of basic blocks, and can realize square root circuits of any size. In particular, we follow the classical non-restoring array structure of square-root circuit, which performs 2's Complement addition/subtraction controlled by a digit-by-digit square root result. Here, we design a reversible controlled adder/subtractor (RCAS) block, which performs 2's Complement computation, and can be used in a modular way to execute a square root operation.

2. SAIMUR RAHMAN GATE

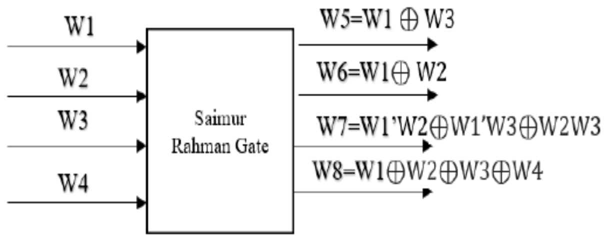


Fig -1: Saimur Rahman Gate

The SRG gate acts as a full subtractor only if W4=0, if W4 is ≠ 0 then the gate won't act as a full subtractor. W5 and W6 acts as a garbage output. The truth table of the SR gate is given in Table-1.

Table -1: Truth Table for Saimur Raman Gate

W1	W2	W3	W4	W7	W8
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

Here the output W7 represents the borrow (BO) and W8 represents the difference (DI). Let the radicand be N7N6N5N4.N3N2N1N0 with a total length of 8 bits. Since it is an 8-bit square rooter the no of bits for the quotient is 8/2= 4(U3U2.U1U0). The implementation of 8-bit square rooter using SR and FG gates is shown in fig 1.

3. REVERSIBLE MULTIPLEXER:

According to the non-restoring computation if the remainder is positive then quotient (U=1) and the difference should be carried over for the next process. If the remainder is negative then quotient is (U=0) and the previous inputs should be carried over for the next process. So a control unit is designed using RT reversible gate to switch between the difference (W8) and the inputs (W1).A reversible multiplex shown in fig 2 is used for all the gates to switch between the difference and the inputs.

The input (W1), the difference (W8) and the quotient is given as input to the gates. The configuration is shown So depending upon the quotient (U) the input and the difference will get switched automatically.

Thus with these reversible gates the design is proposed for the non-restoring algorithm. The conventional design is carried upon using the irreversible gates. The basic xor, and, or, not gates are used for conventional approach. The power results of reversible and conventional design is obtained

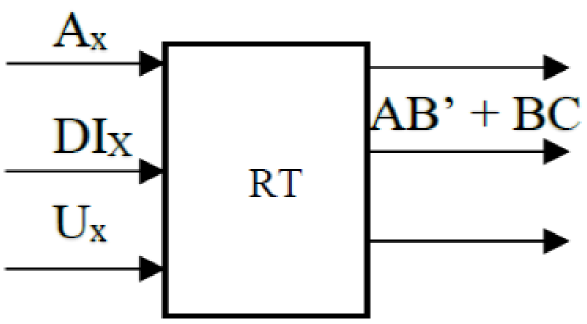


Fig-2: Reversible mux

4. BLOCK DIAGRAM

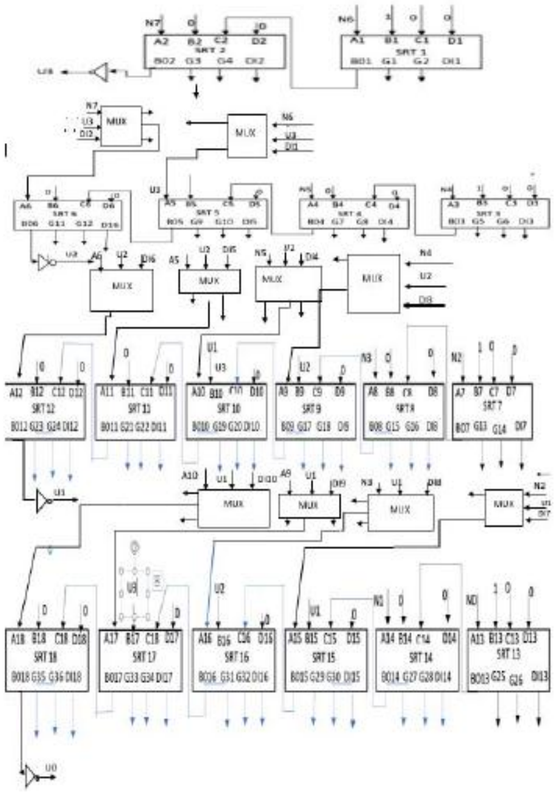


Fig-3: Design of an 8 Bit Binary square rooter

4.1 Working

From figure 3 it can be seen that the square root of a binary number is found by digit by digit process and with the help of Modified Non-restoring algorithm. Since modified non restoring algorithm is used the given binary number is first divided into groups of two digits. Now 1 is subtracted from the most significant group of digits (N7 N6) as the initial step. Then if the subtracted value is positive the quotient is taken as 1 (U3) and the reversible multiplexer passes the difference value which is obtained from the gate as d bits (D1D2…) to the next step.

Else if the subtracted value is negative the quotient is taken as 0 (U3) and the reversible multiplexer passes the previous input value to the next step. And along with this value obtained from the previous step the next two bits is taken along with that value. These values are stored in A bits (A1A2A3A4…).

The previous step quotient is taken and along with it 01is appended and stored in B bits (B1B2B3…) and it is subtracted from A and the above process is repeated until all the digits are completed. The final quotient is the square root of the given binary number which is the given input to the code by the user. The other outputs like (G1G2G3…) are garbage outputs.

Thus the square root of a binary number is found by using reversible gates like Samiur Rahman gate to act as full subtractor and reversible multiplexer to act as multiplexer and by following the Non-restoring algorithm.

4.2 Methodology

To find the square root of a binary number is performed by digit by digit procedure. The algorithm is further classified into non-restoring and restoring algorithms. The restoring algorithm involves more number of hardware resources which in turn increases the total power of the system. Hence the non-restoring algorithm is used for the computation. The non-restoring algorithm simply uses subtraction and appending 01. In the non-restoring algorithm, the total number of bits N is divided into groups of two digits. In general, the length of the quotient is N/2. The algorithm is as follows:

Step1: Split the number of bits N into groups of two digits.

Step2: Subtract 1 from the left most significant group of digits. Quotient is 1 if the difference is positive and quotient is 0 of the difference is negative

Step3: Bring the next group of two digits. Append 01 and the previous quotient and then subtract.

Step4: Proceed to step 2 until the end of groups of two digits.

The radicand may be a whole number (13) or may be a decimal number (2.2). If the radicand is a whole number (13) it can be represented as 1101 and if it is a decimal number 2.2 it can be represented as 0010.0011

The number of bits after the decimal can be extended to N number of bits depending upon the application and accuracy that is required. For example, consider a binary square rooter (N7N6N5N4.N3N2N1N0). The quotient for the square rooter is U3U2.U1U0.The count of bits before and after the decimal point can be decided upon the user. For example the square root of 13(1101) and 2.2(0010.0011) is given in figure 4.

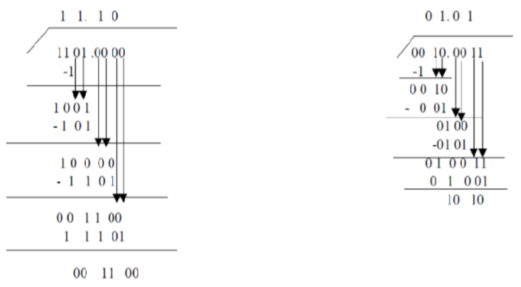
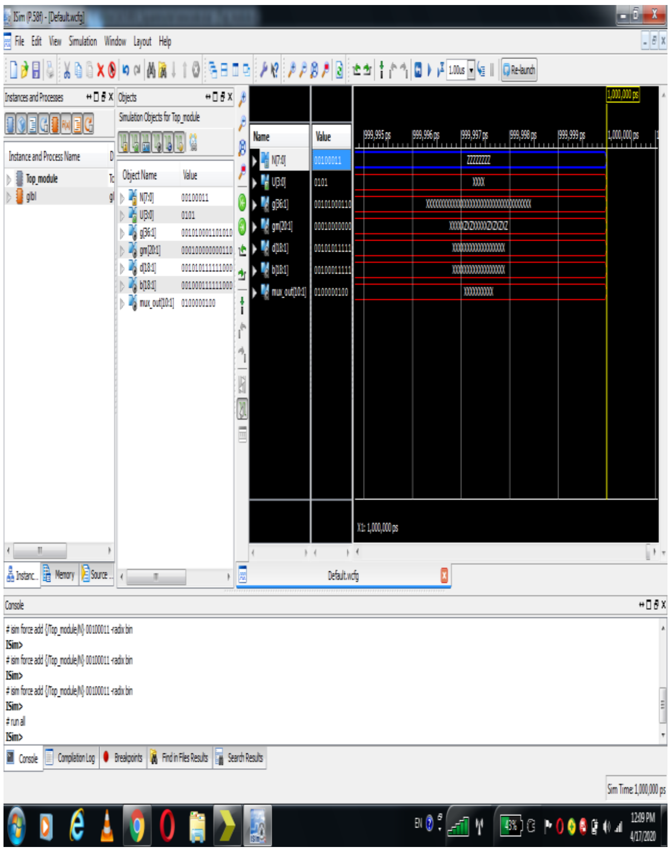


Fig-4: Non-Restoring Algorithm

From Fig.4, the square root of 13 and 2.2 are found. The square root of 13 is approximately equal to 3.6.From the quotient 11.10(U3U2.U1U0), 11(U3U2) corresponds to 3 and 10 (U1U0) corresponds to .6. The binary representation of .6 is 1001..The square root of 2.2(0010.0011) is approximately equal to 1.4..From the quotient 01.01, 01(U3U2) corresponds to 1 and 01 (U1U0) corresponds to .4.The binary representation of .4 is 0110.

5. RESULTS

The simulation result of the given binary number is given by using reversible logic with non-restoring algorithm for division process. Hence the reduced power and gate count simulation output for the input binary radicand 0010.0011 is shown in the Fig-5.


ISim (P.58f) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Instances and Processes
Objects
Simulation Objects for Top_module
Default.wcfg
X1: 1,000,000 ps
Sim Time: 1,000,000 ps

Object Name	Value
N[7:0]	00100011
U[3:0]	0101
g[36:1]	001010001101010
gm[20:1]	00010000000110
d[18:1]	00101011111000
b[18:1]	00100011111000
mux_out[10:1]	0100000100

```
# isim force add {/Top_module/N} 00100011 -radix bin
ISim>
# isim force add {/Top_module/N} 00100011 -radix bin
ISim>
# isim force add {/Top_module/N} 00100011 -radix bin
ISim>
# run all
ISim>
```

Fig-5: Simulation Result

6. CONCLUSION

In the present world of developments in various fields, the main novelty in the field of VLSI is Low Power. In order to attain low power Reversible logic is used in this design. In Irreversible logics, the amount of output port and input ports are not uniform and therefore there will be a loss of information bits. Hence reversible logic is used.The square root calculation using digit-by-digit method can be performed either by restoring method or non-restoring method. In restoring method more hardware resources are required whereas in non-restoring method it consumes less hardware compared to restoring method. Hence non-restoring algorithm is preferred. Many hardware architectures for digit-by-digit square root calculation using irreversible logic has been proposed. There are many reversible logic circuits that are used for the computation. Power reduction can also be done using array based arithmetic computation. Reversible logic is used to design hardware realization of non-restoring algorithm for computation of square root. Hence A low power reversible binary square rooter has been designed using reversible logic and non-restoring algorithm. The square rooter can be used in ALUs especially in DSP processors for reversible computing. On comaparing with the reference papers Conventional approach consumes more power than the Reversible computing. Also a decrease in gate count is observed.

REFERENCES

[1] R. Landauer, Irreversibility and heat generation in the computing bprocess, IBM J. Res. Dev. 5 (1967) 183–191.
[2]C.H. Bennett, Logical reversibility of computation, IBM J. Res. Dev. 17 (1973), 525–532. [3] N. Krishna, V. Murugappan, R. Harish, M. Midhun and E. Prabhu,
[3] "Design of a novel reversible nlfsr," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, 2017, pp. 2279-2283.
[4] L. Yamin, C. Wanming, Implementation of single precision floating point square root on FPGAs, in: Proceedings of the IEEE Symposium on FPGA for Custom Computing Machines,1997,pp 226–232.
[5] T. Sutikno, An efficient implementation of the non-restoring square root algorithm in gate level, Int. J. Comput. Theory Eng.3 (1) (Feb. 2011) 1793–8201.
[6] H. Haritha and S. R. Ramesh, "Design of an Enhanced Array Based Approximate Arithmetic Computing Model for Multipliers and Squarers," 2017 14th IEEE India Council International Conference (INDICON), Roorkee, 2017, pp. 1-5.
[7] L. Yamin, C. Wanming, A new non-restoring square root algorithm and its VLSI Implementation ,in :Proceedings of the
[8] L. Yamin, C. Wanming, Parallel-array implementations of a non-restoring square root algorithm, in: Proceedings of the IEEE International Conference on ComputerDesign: VLSI in Computers and Processors, 1997, pp. 690–695.
[9] R. Balakumaran and E. Prabhu, "Design of high speed multiplier using modified booth algorithm with hybrid carry look-ahead adder," 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, 2016, pp. 1-7.
[10] T. Sutikno, An efficient implementation of the non-restoring square root algorithm in gate level, Int. J. Comput. Theory Eng. 3 (1) (Feb. 2011) 1793–8201.
[11] A.V. AnanthaLakshmi , Gnanou Florence Sudha, " Design of a reversible floating-point square root using modified non-restoring algorithm", in Microprocessors and Microsystem, Vol.50, pp.39-53, May 2017.
[12] Md. Samiur Rahman,Sajjad Waheed, Ali Newaz Bahar, "Optimized Design of Full-Subtractor Using New SRG Reversible Logic Gates and VHDL Simulation", in International Conference on Electrical & Electronic Engineering,Rajshahi,2015,pp.69-72
[13] L. Gopal, N Raj, A. A. Gopalai and A. K. Singh, "Design of reversible multiplexer/de-multiplexer," 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014), Batu Ferringhi, 2014, pp. 416-420.