

Creating a Logic Divider Based on BCD and Utilizing the Vedic Direct Flag Method

Perumal B¹, Balamanikandan A^{2*}, Arunraja A³, Venkatachalam K⁴, Shaik Rahamtula⁵, and Dhanalakshmi M⁶

¹Electrical and Electronics Engineering, Adhiyamaan college of engineering, Hosur, India; balanperumal.b@gmail.com

²Electronics and Communication Engineering, Mohan Babu University (Erstwhile SreeVidyanikethan Engineering College), Tirupati, India; balamaniee83@gmail.com

³Electronics and Communication Engineering, Christ University, Bangaluru, India; arunraja21@gmail.com

⁴Electronics and Communication Engineering, Audisankara college of engineering and technology, Gudur, India; venkatmek12@gmail.com

⁵Electronics and Communication Engineering, Ramireddy Subbarami Reddy (RSR) Engineering College, Kavali, India; dr.rahamath1986@gmail.com

⁶Electrical and Electronics Engineering, Selvam College of Technology, Namakkal, India; dhanaeee81@gmail.com

*Correspondence: Balamanikandan A; balamaniee83@gmail.com

ABSTRACT- Reversible logic has potential for a variety of applications demanding low energy usage since it prevents information loss and energy waste. The purpose of this work is to design a new Vedic divider circuit with reversible gates. Efficiency in quantum and ASIC parameters is demonstrated by the Reversible Direct Flag Vedic Division Method (RDFVDM), which has been devised. Block-level reversible gates are used in the RDFVDM to provide benefits including lower quantum costs and less trash outputs. The performance of Cadence EDA Tool is validated by simulation trials. Based on a comparative examination utilizing current methodologies, RDFVDM performs better than comparable designs. Interestingly, it improves energy usage by 26%. Moreover, RDFVDM performs exceptionally well in terms of quantum cost while employing the RSA cryptographic technique, efficiently managing 1276,293 constant inputs and 311 garbage outputs.

Keywords: RSA Cryptographic, Reversible Gate technique, Direct Flag Vedic Division Process.

ARTICLE INFORMATION

Author(s): Perumal B, Balamanikandan A, Arunraja A, Venkatachalam K, Shaik Rahamtula, Dhanalakshmi M;

Received: 01/05/2024; **Accepted:** 26/07/2024; **Published:** 10/08/2024;

e-ISSN: 2347-470X;

Paper Id: IJEER 0105-06;

Citation: 10.37391/ijeer.120321

Webpage-link:

<https://ijeer.forexjournal.co.in/archive/volume-12/ijeer-120321.html>



Publisher's Note: FOREX Publication stays neutral with regard to Jurisdictional claims in Published maps and institutional affiliations.

1. INTRODUCTION

A divider plays a crucial role in various hardware applications, including digital signal processing and cryptography. This study explores the implementation of a reversible array divider circuit using a non-restoring division algorithm with k-CNOT gates. The proposed approach aims to minimize garbage outputs and gate count. Additionally, in the context of RSA cryptography, double-precision floating-point Vedic dividers are employed [1]. These dividers combine subtraction and floating-point multiplication using the Goldschmidt algorithm. Notably, time delays are minimized by utilizing the Parvartya Sutra and the Nikhilam Sutra [2][3]. Additionally, a different study focuses on an RGB to HMD converter circuit that makes use of registers, multiplexers, comparator functions, addition, subtraction, and multiplication reversible modules. In

applications involving low-power video processing, this circuit is essential. Lastly, taking exact rounding, division from single-sided approximations, and floating-point data into account, a scalable reversible binary division circuit is built [4]. Reversible blocks have been used in research recently to create a big divider circuit that includes registers and multiplexers. To cut down on iterations, a different design has been developed that centre on reversible greatest common divisor (GCD) calculation [5]. The binary GCD algorithm is updated in this design. Furthermore, a 32-bit divider is constructed by employing the antiquated Parvartya Sutra technique [6][7]. The Parvartya Sutra offers an all-encompassing universal division formula that is efficient in terms of both power usage and latency when dividing big numbers. Divider circuits that are now in use are usually classified as either restoring or non-restoring. Vedic dividers on the other hand make use of the Nikhilam and Parvartya Sutras [8]. These dividers play a crucial role in arithmetic units, involving complex computations that can impact overall processing speed [9]. In our work, we explore a novel Vedic sutra design using reversible arithmetic circuit blocks, optimizing both structural and quantum parameters. Furthermore, we delve into RSA cryptography, a well-known public key cryptography technique used for data protection. The RSA algorithm is used in the Vedic divider implementation, referred to as "Dhvajanka" (at the highest point of the flag) [10]. Finally, a reversible shift register is used in a fault-tolerant reversible divider (FTRD) that has a parallel adder. Similarly, in another scenario, RSA cryptography uses the Vedic divider,

which uses double-precision floating-point division [11]-[13]. Designing VLSI systems often involves a delicate balancing act. While speed, gate count and low power usage are all important, achieving all three simultaneously can be challenging. Often, area and speed present conflicting constraints.

This revised version emphasizes the originality of our work by highlighting the:

- *Novel implementation of DFVDM:* This has not been documented in existing literature.
- *Utilization of reversible gates:* This suggests a focus on potentially low-power and fault-tolerant design.

2. DFVDM IMPLEMENTATION

The Direct Flag Method, also known as the Flag Method, is a division technique in Vedic mathematics that provides shortcuts for dividing any type of numbers. We will break down the steps involved in this method: Begin with the given dividend and divisor. Split the divisor into two parts—the first part (usually a single digit) becomes the new divisor, and the second part (called the ‘flag’) is retained [14]. For the initial division, divide the first digit of the dividend by the new divisor. Record the quotient (Q) and remainder. Next, multiply the new quotient by the flag and subtract this product from the original dividend. Divide the result of the subtraction by the new divisor to get the next quotient and remainder. Repeat this process until you encounter a negative remainder. When the remainder becomes negative, adjust the quotient and remainder [15]. Decrease the quotient by 1 and increase the remainder by the first multiplier of the new divisor (i.e., the product of the new quotient and the flag). Continue the process until no negative remainders occur. Finally, concatenate the quotients obtained at each stage to form the final quotient. The last remainder is the actual remainder. Let us illustrate this with an example,

Example: Dividend = 1732 (D3=1, D2=7, D1=3 and D0=2)
Divisor = 23 (Split into 2 and 3)
Divide 1 by 2: Q2 = 0, R = 1
Multiply Q2 (0) by the flag (3): $0 \times 3 = 0$
Subtract 0 from 2: $2 - 0 = 2$
Divide 2 by 1: Q1 = 2, R = 0
Multiply Q1 (2) by the flag (3): $2 \times 3 = 6$
Subtract 6 from 3: $3 - 6 = -3$
Adjust quotient: Q1 = 1, R = 3
Multiply Q1 (1) by the flag (3): $1 \times 3 = 3$
Subtract 3 from 4: $4 - 3 = 1$
Divide 1 by 1: Q0 = 1, R = 0
Multiply Q0 (1) by the flag (3): $1 \times 3 = 3$
Remainder: 2 (positive)
Result: Quotient = 102, Remainder = 2

In figure 1. The Direct Flag Method in Vedic mathematics tackles division with a shortcut. You split the divisor in two - one part becomes your new divisor, the other a guide. Repeatedly, you try dividing by the new divisor, adjust based

on a flag-related multiplication, and subtract (16). After these steps, the remaining number is your remainder, and the temporary quotients you got give you the final answer. This method makes division faster, especially for mental math.

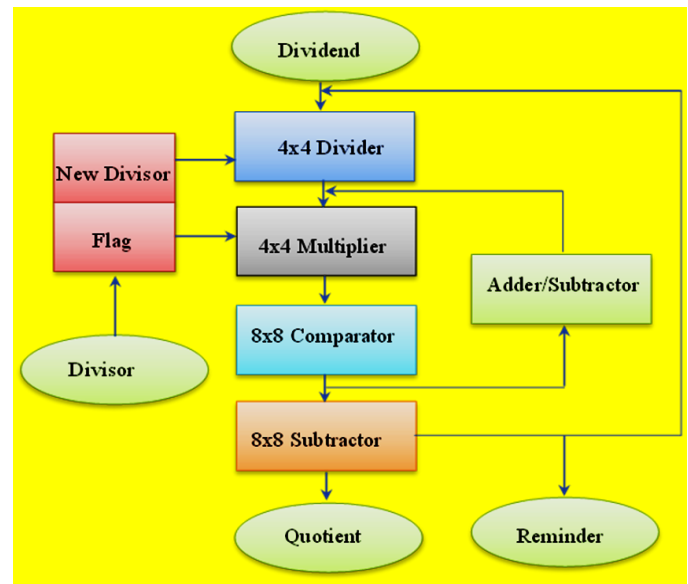


Figure. 1 Block Diagram of DFVDM

3. BLOCK LEVEL IMPLEMENTATION OF RDFVDM

RDFVDM leverages existing building blocks proposed in various literature sources. These blocks serve as the foundation for implementing the division method. They could include arithmetic circuits, quantum gates, or other computational elements. When evaluating the performance of different blocks, several parameters are considered. Area (referring to the physical space occupied by the circuit or component), Power (representing energy consumption during computation), and Delay (measuring the time taken for a computation to complete) [17]. Additionally, Quantum Parameters (or Quantum Cost) quantify the resources (such as qubits and gates) required for quantum computation. The Circuit Implementation (CI) describes how well the circuit can be implemented using available quantum gates, while the Gate Overhead (GO) indicates the additional gates needed due to error correction and fault tolerance.

3.1 Subtractor/ Adder 8x8

An 8x8 reversible subtractor/adder was created by H. G. Rangaraju et al. [13] by combining complete adder/subtractor blocks composed of FG (Fredkin Gate) and PG (Peres Gate). The following are some of their approach's main points: Implemented using three reversible gates, a PG gate (QC = 4) and two FG gates (QC = 1), the reversible half adder/subtractor yields a garbage output of 2. They also built an invertible full subtractor/adder by connecting 2 FG and 2 PG gates, respectively. With the help of these building components, an eight-bit parallel binary adder/subtractor was successfully built. In comparison to Lamjed Touil's method, the control input

performs addition and subtraction operations differently, improving overall performance. However, Amjed Touil et al. [4] employed a RAS gate with five inputs and five outputs. The RAS Gate is a full 1-bit subtractor/adder composed of 2 ASIG (Ancilla-Swap-Increment-Decrement) blocks. The Quantum Cost (QC) for this gate is 18, and the Gate Overhead (GO) is 4.

While their approach is interesting, H. G. Rangaraju's design outperformed it [18]. The Reversible 8-Bit Adder/Subtractor consists of one reversible half adder/subtractor and seven reversible full adder/subtractor units. Inputs include A and B (both 8-bit) along with a control signal. The outputs are S_D (8-bit output) and C_D (Carry/Difference output). The circuit has a total quantum cost of 76. This research contributes to the advancement of reversible logic and its applications *figure 2* illustrates the structure of the reversible 8-bit adder/subtractor.

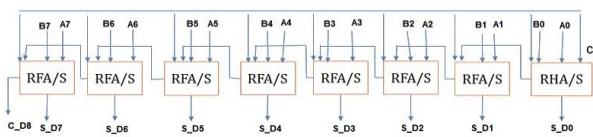


Figure 2. Reversible Adder 8x8 /Subtractor

3.2 8x8 Comparator

Let us say the inputs are A and B. $A=B$ equals 1 when A and B are equal. When A exceeds B, output $A>B$ equals 1. $A<B$ equals 1 when B exceeds A. *Table 3.1* displays the 1-bit comparator's truth table. The 1-bit comparator's Boolean expression is provided by

$$(A > B) = AB' \quad (1)$$

$$(A < B) = A'B \quad (2)$$

$$(A = B) = (A \oplus B) \quad (3)$$

Table 1. Truth Table for 1-bit Comparators

INPUT		OUTPUT		
A	B	A>B	A<B	A=B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

The most important bit is used for comparison by the MSB Comparator. *Figure 3* depicts the MSB comparator's construction. One MIG gate and two F2G gates are needed. A and B are the inputs. The outputs are P, Q, and R. Upon receiving an input of B, the first F2G gate produces an inverted output of B'. The MIG gate produces the outputs $P(A>B)$ and $R(A=B)$ from two inputs, A and B'. The MIG gate's final output is routed to another F2G gate to generate $Q(A<B)$. P, Q, and R are the inputs of the reversible 1-bit comparator. F2G gate has two quantum costs, compared to seven for MIG gate. F2G gate is another name for the double Feynman gate. The overall quantum cost of the MSB comparator is 11. In this case, four constant inputs with the value 0 are employed. There are five outputs of waste. G represents the trash outputs [19].

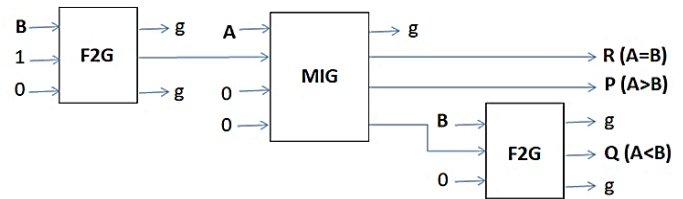


Figure 3. Reversible MSB Comparator

The NFT Block is used to compare the following important bits. In *figure 4*, an NFT block is shown. A and B are the inputs of the NFT block. The NFT Block consists of one F2G gate and one NFT gate. Whereas the NFT block produces output AB. The quantum cost of the NFT gate is 5. The quantum cost of NFT Block is 7. This block has two constant inputs that are both set to 0. The letter A stands for one of three waste outputs.

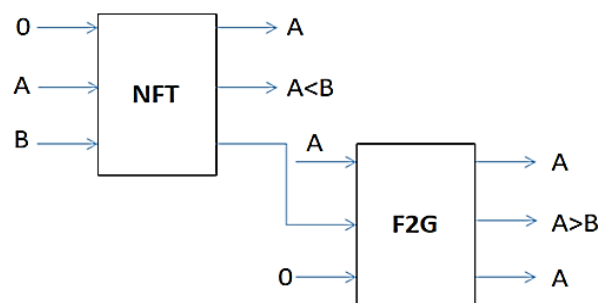


Figure 4. Reversible NFT Block

Figure 2 displays the block diagram for a 1-bit comparator. One TVG gate, two MIG gates, and one NFT block are needed for a 1-bit comparator. A and B are the inputs. The outputs of the NFT blocks are connected to MIG gates. P, Q, and R are the outputs of the block before it that are inserted into the MIG gates. The MSB comparator provides P to the first MIG gate, which generates $A>B$, and Q to the second MIG gate, which generates A P, Q, and R are the outputs, and A and B are the 8-bit inputs. The MSB comparator receives the two most important input bits, A7 and B7, while the 1-bit comparator receives the remaining inputs. Every comparator displays the matching input's comparison result. The outcome of the 8-bit inputs is provided by the last comparator. The connections of an 8-bit comparator are shown in *figure 5*. The overall quantum cost of the MSB comparator is 11. The quantum cost of the NFT gate is 5. The quantum cost of NFT Block is 7. The quantum cost of the 8x8 reversible comparator is 179. The total consists of 47 garbage outputs and 33 constant inputs. The reversible 8-bit comparator consists of one MSB comparator block and seven 1-bit comparator blocks. The 8-bit inputs are A and B, and the outputs are P, Q, and R. The 1-bit comparator receives the remaining inputs, whereas the MSB comparator receives the two most crucial input bits, A7 and B7. Each comparator shows the comparison result for the matched input. The result of the eight-bit inputs is given by the last comparator. *Figure 4* depicts the connections of an 8-bit comparator. The MSB comparator has a total quantum cost of 11. The NFT gate has a quantum cost of 5. NFT Block has a quantum cost of 7. The 8x8 reversible comparator has a quantum cost of 179. 33 constant inputs and 47 trash outputs make up the total.

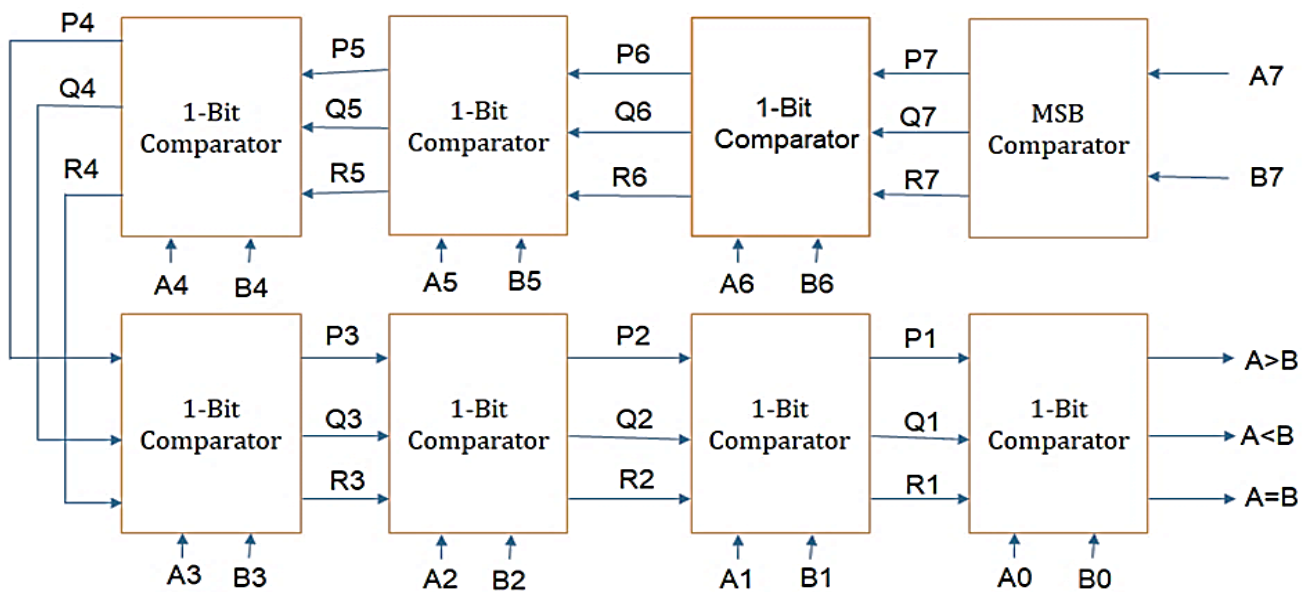


Figure 5. Reversible 8-bit Comparator 3.3.4*4 Multiplier

The 4-bit RCA consists of one Peres gate and three HNG gate. A and B are 3-bit inputs. S is 4-bit output and C0 is the Carry output. The quantum cost of HNG gate is 7. The quantum cost of 4-bit RCA is 22. It has 4 constant inputs which are set as 0. PG gate has a single garbage output and HNG gate has two garbage outputs. Hence, the total garbage output of Reversible 4bit RCA is 7. The construction of the circuit is shown in figure 6 and 7.

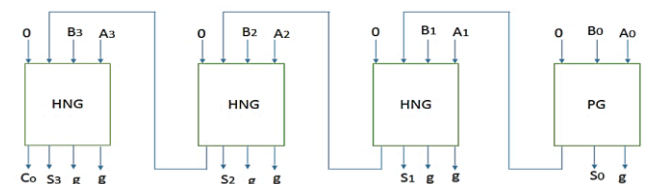


Figure 6. Reversible 4-bit RCA

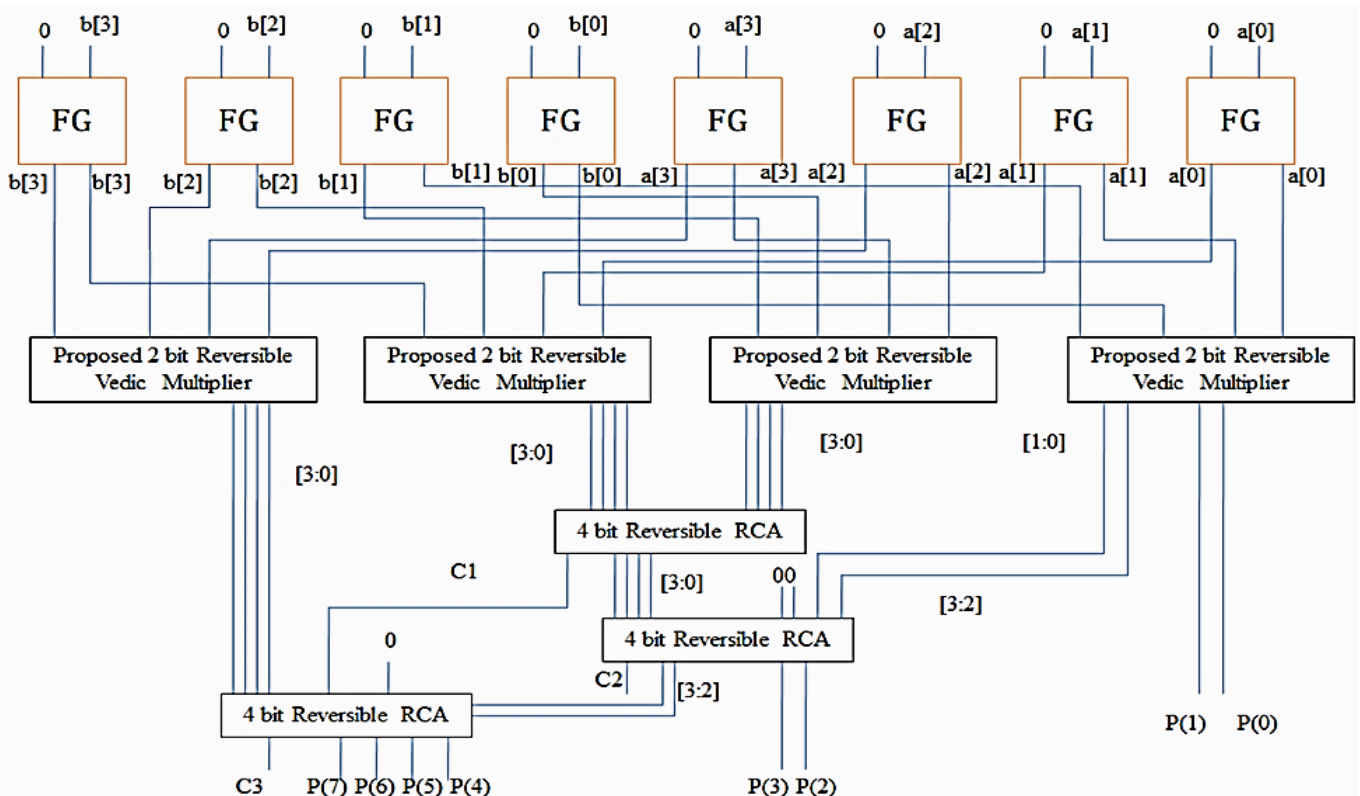


Figure 7. Reversible 4x4 Vedic Multiplier

3.3 8/4 Divider

Division algorithms play a crucial role in computing quotients and remainders when dividing two numbers. Among the various approaches, two common types stand out: Slow Division (Restoring Division) and Fast Division (Non-Restoring Division). The former restores the remainder at each step, resulting in additional complexity due to extra operations. In contrast, non-restoring division simplifies the process by avoiding the restoration of the remainder during intermediate steps. Non-restoring division becomes particularly valuable when efficiency is a priority. Here's how non-restoring division works: 1. *Inputs*: Dividend (D) with n bits. - Divisor (d) with m bits. 2. *Initialization*: Accumulator (A) is set to 0. - Quotient (Q) is initialized to D. 3. *Repeat the following steps until the count reaches zero*: Shift A and Q left by one bit. - If the most significant bit (MSB) of A is 1, perform an addition ($A = A - d$). - If the MSB of A is 0, perform a subtraction ($A = A + d$). - Update Q0 based on the MSB of A ($Q_0 = 0$ if $MSB(A) = 1$, else $Q_0 = 1$). - Decrement the count. 4. *At the end*: - If $A < 0$, perform an addition ($A = A + d$). - If $A > 0$, the process stops. 5. The quantum cost of a 5-bit reversible adder or subtractor is 46. However, the total quantum cost of the circuit in DFVDM (Direct Flag Vedic Division Method) is 305, with a constant input of 40. 6. Reversible Adder/Subtractor: - This component is critical for non-restoring division. - It operates on two n-bit inputs (A and B) and produces two n-bit outputs (Sum and Borrow). - The quantum cost reflects the resources needed for reversible computation. 7. *Remember*: While non-restoring division is efficient, it requires careful handling of intermediate results.

INPUTS

Dividend = 1732 Divisor = 23

OPERATION

New divisor= 2 Flag = 3

D3=1, D2=7, D1=3 and D0=2

Step1:

Division: $D_3/\text{New Divisor} \Rightarrow 1/2 \Rightarrow Q_2=0 \text{ \& } R=1$

Multiplication: $Q_2 \times \text{Flag} \Rightarrow 0 \times 3 = 0$

Checking: $RD_2 > 0 \Rightarrow 17 > 0$; Condition satisfied

Subtraction: $RD_2 - 0 = 17 - 0 \Rightarrow 17$

Step 2:

Division: (17) divided by (2)

Quotient ((Q₁)): (8)

Remainder ((R)): (1)

Multiplication: (Q₁) multiplied by the divisor ((3)): (8 \times 3 = 24)

Checking: Is the result of the multiplication greater than (24)? ($RD_1 > 24$)

Since (13) is not greater than (24), the condition is not satisfied.

Adjusting: Decrease (Q₁) by (1) and add the new divisor to the remainder:

New (Q₁): (7)

New (R): (3)

Multiplication: (Q₁) multiplied by the divisor ((3)): (7 \times 3 = 21)

Checking: Is the result of the multiplication greater than (21)? ($RD_1 > 21$)

Yes, (33) is greater than (21), so the condition is verified.

Final Subtraction is ($RD_1 - 24 = 33 - 24 = 9$)

Step 3:

Division: (12) divided by (2)

Quotient ((Q₀)): (6)

Remainder ((R)): (0)

Multiplication: (Q₀) multiplied by the divisor ((3)): (6 \times 3 = 18)

Checking: Is the result of the multiplication greater than (18)? ($RD_0 > 18$)

Since (2) is not greater than (18), the condition is not satisfied.

Adjusting: Decrease (Q₀) by (1) and add the new divisor to the remainder:

New (Q₀): (5)

New (R): (2)

Multiplication: (Q₀) multiplied by the divisor ((3)): (5 \times 3 = 15)

Checking: Is the result of the multiplication greater than (15)? ($RD_0 > 15$)

Yes, (22) is greater than (15), so the condition is satisfied.

Final Subtraction is ($RD_0 - 15 = 22 - 15 = 7$)

So final Output:

Quotient (Q): (086).

Remainder: (7).

The flow of the RSA algorithm will be,

- Select primary number p&q public key= {e,n}
- $N=p \times q$
- $\Phi(n)=(p-1) \times (q-1)$ private key= {d}
- Choose e , $1 < e < \Phi(n)$ & $\text{Gcd}[\Phi(n), e] = 1$
- $d=e-1 \bmod \Phi(n)$, Figure 8 additionally included clarified information of the RSA algorithm flow.

4. RDFVDM IN RSA ALGORITHM.

RSA Key Generation involves two keys: a public key and a private key. The keys are produced as follows: Select two big prime numbers: Decide on two different prime numbers, p and q. Using the formula ($n = p \cdot q$), get the modulus (n). Both the public and private keys use this value. Calculate the Totient Function of Euler: The count of positive integers less than (n) that are coprime—that is, have no common factors—with (n) is

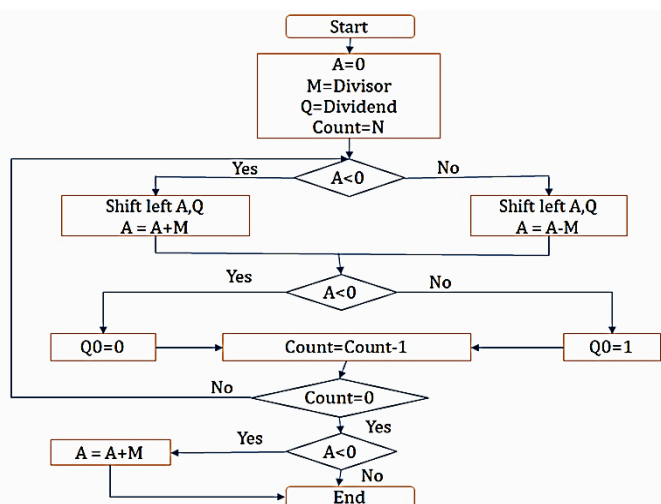


Figure 8. Flow Chart of RSA algorithm

provided by Euler's totient function, represented by the symbol $(\phi(n))$. $(\phi(n) = (p-1) \cdot (q-1))$ for RSA. Select the Exponent of Public Key ((e)): Choose an integer (e) such that $(\gcd(e, \phi(n)) = 1$ and $(1 < e < \phi(n))$. (e), the public key exponent, is made available. Determine the Exponent of Private Key ((d)): Determine the value of (d) so that $(d \equiv e^{-1} \pmod{\phi(n)})$.

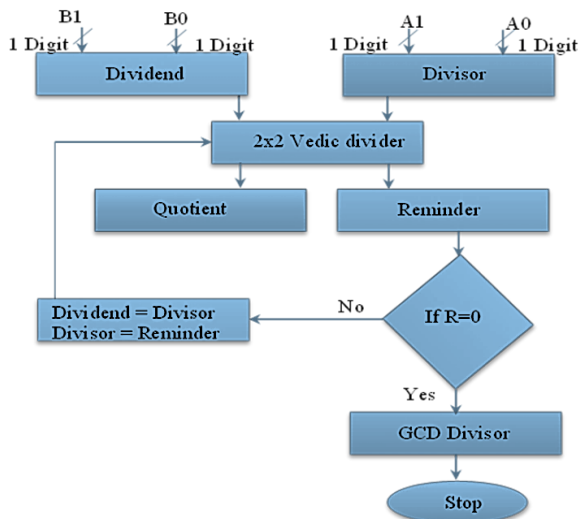


Figure 9. Block Diagram of GCD

The private key's exponent (d) is kept secret. Public Key: The public key is the pair ((n, e)). The public exponent is (e), while the modulus is (n). Private Key: The private key consists of the pair ((n, d)). The private exponent (d) and the modulus (n) are the same. GCD, or greatest common divisor, their greatest common divisor, or GCD figure 9, is the largest positive integer that divides both numbers. It is necessary to ensure the public key exponent (e) and the coprimality of the totient function $(\phi(n))$. Keep in mind that for secure RSA key generation, comprehension of these procedures is necessary.

An 8x8 multiplier, subtractor 8x8, comparator 8x8, divider 8x8, and GCD 8x8 are needed for the RSA algorithm. The RSA

algorithm has a quantum cost of \$1276. There are 293 consistent inputs, and 311 garbage outputs are produced.

4.1 An illustration of the RSA Cryptographic Algorithm's key generation concept

- Let $p=5$ and $q=7$
- $n = p \times q = 5 \times 7 = 35$
- $\phi(n) = (p-1) \times (q-1) = 4 \times 6 = 24$
- Let $e = 23$
- Here $1 < 23 < 24$ & $\text{GCD}(23, 24) = 1$
- $d = 23^{-1} \pmod{24} = 1$
- Secret key is $\{23, 35\}$
- Public key = $\{1\}$
- Let $p=11$ & $q=19$
- $n = p \times q = 11 \times 19 = 209$
- $\phi(n) = (p-1) \times (q-1) = 10 \times 18 = 180$
- Let $e=166$, Here $1 < 166 < 180$ and $\text{GCD}(166, 180) = 2$
- The value of e needs to be adjusted
- because the GCD of e and $\phi(n)$ is not equal to 1.
- Suppose $e=167$. In this case, $1 < 167 < 180$ and $\text{GCD}(167, 180) = 1$.
- $d = 167^{-1} \pmod{180}$ equals 13.
- Secret key is $\{167, 209\}$.
- $\{13\}$ is the public key.

5. THE RESULTS AND DISCUSSION

5.1 The Results from the Simulation

Figure 10 Greatest Common Divisor (GCD) Calculation Inputs: Number 1 (aa, ab): 00100100 Number 2 (ND, FL): 00010010 Output: GCD (g): 00010010, Figure.11.RSA Algorithm Output Inputs: p (prime number): 0101q (prime number): 0111e (public exponent): 00100011 Output: Public key (n,e): (00100011, 00110101) Private key (d):01.

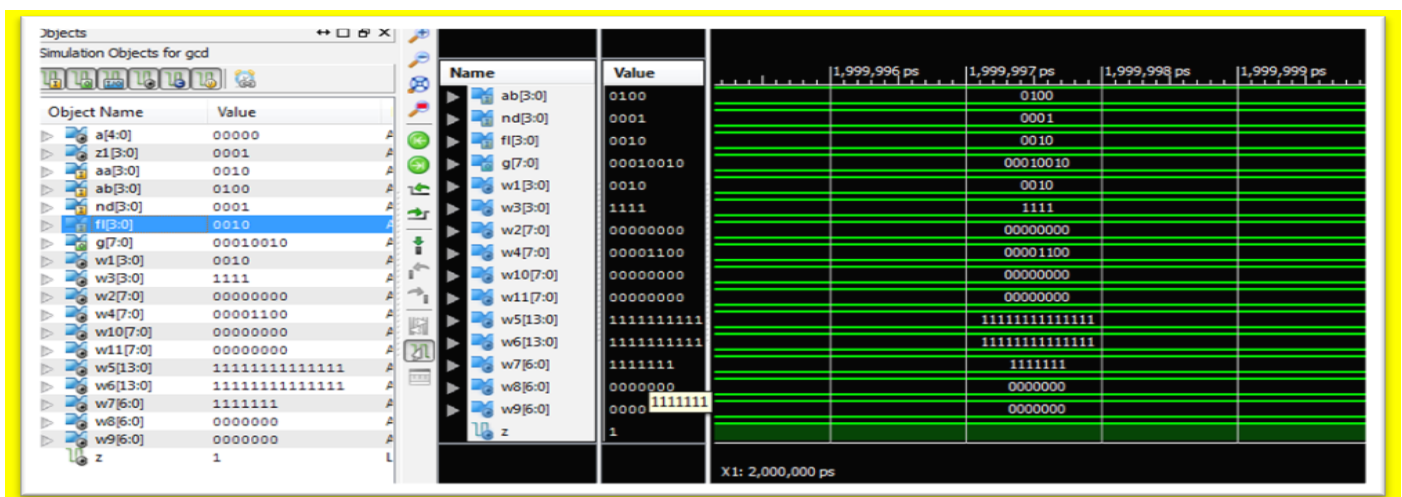


Figure 10. Simulation results for 2-digit GCD

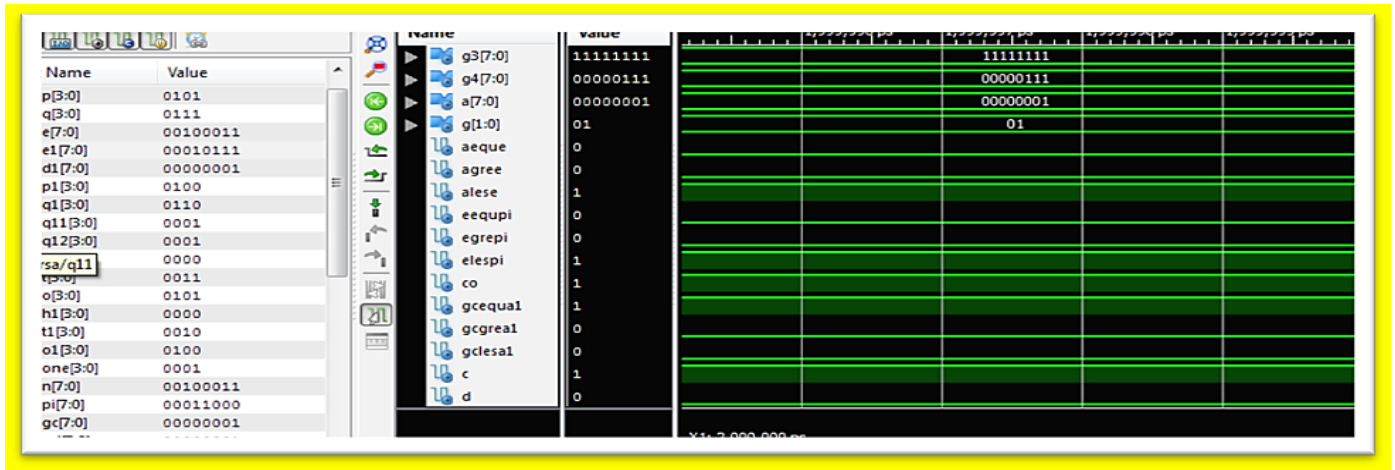


Figure 11. Simulation results for RSA algorithm

5.2 ASIC Parameters

Surface Area, Leakage Power in nW, Dynamic Power, Total Power in nW (sum of leakage and dynamic power), and Delay in ps are all displayed in Table 1 of the 2-Digit Vedic Divider. Comparing Other Methods Similar to [6,], [1], [3], [5], and [12] are area and delay. Notable improvements in power consumption were reduced by around 16% (due to reversible gates). 2-Digit Greatest Common Divisor (GCD), The overall

power is 235274.608 nW, dynamic power is 217471.916 nW, leakage power is 17802.692 nW, and area is 2854. (sum of leakage and dynamic power). Delay: 11948 ps Comparison: Power consumption was reduced by around 19% compared to [6]. Area reduction of about 3%. It is fascinating to see how different architectural choices impact power efficiency and area utilization.

Table 1. Shows the 2-digit Vedic divider's area power and delay results

Parameter	Proposed method	(6)	(1)	(3)	(5)	(12)
Area (no of Gate's)	2775	2833	2891	2928	3165	3122
Power (nW)	227577.350	272589.67	303922.02	311127.21	321313.12	342492.62
Delay (ps)	11199	11388	11661	11707	12088	12655

Table 2. Results of Area, Power & Delay for a 2-Digit GCD

Parameter	Proposed method	(6)	(1)	(3)	(5)	(12)
Area (no of Gate's)	2855	2944	3089	3125	4242	4624
Power (nW)	235275.618	291343.87	313299.3	323433.3	333198.4	399845.3
Delay (ps)	11948	11988	12892	12994	15196	16679

Table 3 displays the ASIC parameters of the suggested and current RSA algorithms. The suggested RSA algorithm's area is 6266. The suggested RSA algorithm has a power of 3159842.00

nW. Table 2 shows that compared to [6], there is an approximate 17% decrease in power usage and a 9% reduction in area.

Table 3: RSA algorithm results for Area Power and Delay

Parameter	Proposed method	(6)	(9)	(5)	(12)
Area (no of Gate's)	6266	6890	11313	9915	11915
Power (nW)	31,59,742.03	38,26,667.5	76,26,000	54,55,646.09	74,55,646.09
Delay (ps)	6957	7100	9974	13218.3	15988.08

5.3 Quantum Parameters

The quantum cost in the suggested method is 432. superior to current techniques in comparison to 5: A 40% gain Improvement of 5.09% as compared to 6 In contrast to 12 and 52.77% growth There are 104 constant inputs and 87 garbage outputs. Comparing the results with the current approaches (constant inputs and garbage output), 5, 6: Garbage Output is 113 while Instant Inputs are 97. There are 103 constant inputs

and 89 garbage outputs. Fixed Inputs: 103 Trash Production: 114 GCD with two digits: 514 is the quantum cost. RSA has 125 constant inputs and 131 garbage outputs. The algorithm Cost in Quantum: 1276 293 constant inputs; 311 constant outputs for trash Additional information about the quantum parameters of the RSA and GCD algorithms may be found in *figures 8 and 9*.

Table 4. Comparing proposed and used Vedic Divider methods quantum mechanically

Quantum Parameters	Proposed RDFVDM method	(6)	(5)	(12)
QC	432	454	605	660
CI	104	103	97	102
GO	87	89	113	114

6. CONCLUSION

Reversible Decimal Floating-Point Dividers (RDFVDMs) utilize less power compared to conventional dividers. These innovative dividers blend ancient Vedic arithmetic with decimal logic, resulting in reduced space and effective energy utilization in RDFVDM design. Notably, RDFVDMs surpass previous research in terms of both area and latency, leading to an impressive improvement in energy consumption of approximately 26%. Reversible logic plays a crucial role in cryptography, especially for security applications. Unlike traditional irreversible logic, reversible logic uses less energy, making it advantageous for cryptographic purposes. In the realm of quantum optics, consider the following parameters: The cryptography algorithm of the SA of Quantum Cost is 1276. Recurring Inputs is 293 and Outputs of Garbage is 311. RDFVDM: Quantum Cost: 432.104 Constant Inputs. The output of garbage is 87.

REFERENCES

- [1] S Rakshit, S Mondal, A Chakraborty, A Sarkar, D K Kole Synthesis of Reversible Array Divider Circuit, pp701-707, (2019).
- [2] Zahra Ariafar and Mohammad Mosleh, 'Effective Designs of Reversible Vedic Multiplier', International Journal of Theoretical Physics (2019) 58:2556–2574.
- [3] RamadeviVemula and K Manjunatha Chari, 'A review on various divider circuit designs in VLSI', Conference on Signal Processing and Communication Engineering Systems (SPACES), 2018.
- [4] LamjedTouil and BouraouiOuni, 'Design of hardware RGB to HMD converter based on reversible logic', IET Image Process., 2017, Vol. 11 Iss. 8, pp. 646-655.
- [5] Hafiz Md, HasanBabu, MdSolaiman Mia, Design of a compact reversible fault tolerant division circuit, Microelectronic J (2016), vol. 51, pp. 15-29.
- [6] Ali B and Majid H, 'Optimized reversible divider circuit', Int. J. Innovative Computing and Applications, Vol. 7, No. 1, 2016.
- [7] H V Jayashree, SkandaKotethota and V K Agrawal, 'Reversible circuit design for GCD computation in cryptography algorithms', Int. J. Circ. Theor. Appl. (2016).
- [8] Siba K P and Arati S, 'A Novel Vedic Divider Architecture with Reduced Delay for VLSI Applications', International Journal of Computer Applications, Vol. 120, No.17, pp.0975 – 8887, June 2015.
- [9] Jyotinnayeesubudhi and C Karthick, 'Implementation of Vedic divider on RSA cryptosystem', International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015.
- [10] HuseyinBodur and Resul Kara, 'Secure SMS Encryption Using RSA Encryption Algorithm on Android Message Application', 3RD International Symposium on Innovative Technologies in Engineering and Science, June 2015.
- [11] Jayashree H. V, V K Agarwal, P VenkatasreeCharan, and A M ChiragKariappa, 'Design of Fault Tolerant n Bit Reversible Comparator for optimization of Garbage and Ancilla bits', Proceedings of International Conference on Circuits, Communication, Control and Computing, 21-22 November 2014.
- [12] Faraz D and Majid H. (2011), 'A novel nanometric fault tolerant reversible divider', International Journal of the Physical Sciences Vol. 6, No.24, pp. 5671-5681, 16 October, 2011. doi: 10.5897/IJPS11.981.
- [13] H G Rangaraju, U. Venugopal, K N Muralidhara and K B Raja, 'Low Power Reversible Parallel Binary Adder/Subtractor', International Journal of VLSI Design & Communication Systems, 1.3(2010), pp-23-34.
- [14] A. Balamaniandan and K. Krishnamoorthi, "Low area ASIC implementation of LUT–CLA–QTL architecture for cryptography applications," Wireless Networks, vol. 26, no. 4, pp. 2681–2693, May 2019, doi: 10.1007/s11276-019-02017-3.
- [15] H. Thapliyal and M.B. Srinivas. "Novel Reversible Multiplier Architecture Using Reversible TSG Gate," in Intern. Conf. on Comp. Syst& App, April 2006.
- [16] R. Landauer, Irreversibility and heat generation in the computational process, IBM J. Res. Dev. 5 (1961) 183–191.
- [17] R. Vemula and K. M. Chari, "A review on various divider circuit designs in VLSI," 2018 Conference on Signal Processing and Communication Engineering Systems (SPACES), 2018, pp. 206-209, doi: 10.1109/SPACES.2018.8316347.
- [18] R. Mishra, "An efficient VLSI architecture for a serial divider," 2017 Devices for Integrated Circuit (DevIC), 2017, pp. 482-486, doi: 10.1109/DEVIC.2017.8073996.
- [19] Ramadevi V, Manjunatha K, Design, and implementation of 64-bit divider using 45 nm CMOS technology Int. J. Pure Appl. Math.20181185293301.
- [20] D. Tomic, J. Mikulic, G. Schatzberger, J. Fellner, A. Baric, Programmable low-frequency divider in 180-nm CMOS technology, in Proceedings of the 43rd International Convention on Information. Communication and Electronic Technology (MIPRO), (2020), pp. 89–92.
- [21] Melchert J Behrooz SLi JKim YSAADI-EC: a quality-configurable approximate divider for energy efficiency IEEE Trans. Very Large Scale Integr. Syst.201927112680269210.1109/TVLSI.2019.2926083.

[22] Chen L, Han J, Liu W, Montuschi P, Lombardi F. Design, evaluation, and application of approximate high-radix dividers. *IEEE Trans. Multi Scale Comput. Syst.* 2018;43:29931210.1109/TMSCS.2018.2817608.



© 2024 by the Perumal B, Balamanikandan A, Arunraja A, Venkatachalam K, Shaik Rahamtula, Dhanalakshmi M. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).