1. PROPOSED KMD GATES AND ITS PERFORMANCE CHARACTERISTICS

**1.1 The Proposed KMD Gates**

Here, we propose 4 reversible gates, namely KMD gate 1, KMD Gate 2, KMD Gate 3 and KMD Gate 4 as shown in Fig. 1 (a-d). These gates satisfy the fundamental requirements (reversibility and universality) of a reversible gate [1]. In addition, they are fault-tolerant in nature, i.e. EXOR function of the inputs and the outputs are equal (parity preservation).

|  |  |
| --- | --- |
|  |  |
| **(a)** | **(b)** |
|  |  |
| **(c)** | **(d)** |
| **Fig.1: Block Diagram (a) KMD Gate 1 (b) KMD Gate 2 (c) KMD Gate 3 and (d) KMD Gate 4** | |

For universality, a reversible gate must be able to produce NOT, AND & OR functions of 2-input format or it must be able to generate NOT, NAND / NOR functions of 2-input format [1]. The universality property of KMD Gates is represented in Table 1. From the above table, it is evident that all KMD Gates satisfy the universality property as stated in [1].

**Table 1: Universality Property of KMD Gates**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Reversible Gate** | **Constant Input** | **Logic Function** | **Expression** |
|  | KMD Gate 1 | A=1; C=0 / 1 | Q=NOT (B) | Q = R’ |
| C=1 | Q = NAND (A,B) | Q = A’ + AB’ |
| C=0 | R = OR (A, B) | Q = A + A’B |
|  | KMD Gate 2 | B=C=0 | R = NOT (A) | R = A’ |
| C=0 | Q = NOR (A,B) | Q = A’B’ = (A+B)’ |
| C=1 | R = AND (A,B) | R = AB |
|  | KMD Gate 3 | B=D=0; C=1 | S = NOT (A) | S = A’ |
| B=1 | R = OR (A, C) | R = A + A’C |
| C=0 | R = AND (A, B) | R = AB |
|  | KMD Gate 4 | B=C=1;D=0 | Q = NOT(A) | Q = A’ |
| B=1 | Q = NAND (A,C) | Q = A’ + AC’ |
| B=1; D=0  C=0 | T = OR (A,C)  Q = OR(A,C) | T = AC’ + C  Q = AC’ + C |

In addition, they are also having Parity Preserving capability; i.e. EXOR function of the inputs and outputs are equal. The Parity Preserving capability of these gates is shown in Table 2. The Performance evaluations of KMD Gates for the 13 Standard Functions realization is shown in Table 3.

**Table 2: Parity Preservation of KMD Gates**

|  |  |  |
| --- | --- | --- |
| **Parity Preservation of KMD Gate1** |  | **Parity Preservation of KMD Gate3** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Inputs** | | | **Outputs** | | | **Parity Preservation** | | **A** | **B** | **C** | **P** | **Q** | **R** | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 0 | 0 | 1 | |  | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Inputs** | | | | **Outputs** | | | | **Parity Preservation** | | **A** | **B** | **C** | **D** | **P** | **Q** | **R** | **S** | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| **Parity Preservation of KMD Gate2** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Inputs** | | | **Outputs** | | | **Parity Preservation** | | **A** | **B** | **C** | **P** | **Q** | **R** | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

**Table 3: Performance evaluation of KMD gates**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S. No** | **Standard Functions** | **Toffoli** | | **Peres** | | **TR** | | **Fredkin** | | **RM** | | **RUG** | | **KMD** | |
| **G** | **QC** | **G** | **QC** | **G** | **QC** | **G** | **QC** | **G** | **QC** | **G** | **QC** | **G** | **QC** |
| 1 | F=ABC | 2 | 10 | 2 | 8 | 2 | 12 | 4 | 20 | 2 | 70 | 2 | 62 | **2** | **18** |
| 2 | F=AB | 1 | 5 | 1 | 4 | 1 | 6 | 2 | 10 | 1 | 35 | 1 | 31 | **1** | **9** |
| 3 | F=ABC+AB’C’ | 3 | 15 | 3 | 12 | 2 | 12 | 3 | 15 | 2 | 70 | 2 | 62 | **4** | **36** |
| 4 | F=ABC+A’B’C’ | 10 | 50 | 10 | 40 | 9 | 54 | 4 | 20 | 3 | 105 | 3 | 93 | **3** | **27** |
| 5 | F=AB+BC | 5 | 25 | 4 | 16 | 4 | 24 | 2 | 10 | 2 | 70 | 2 | 62 | **2** | **18** |
| 6 | F=AB+A’B’C | 8 | 40 | 6 | 24 | 7 | 42 | 5 | 25 | 2 | 70 | 3 | 93 | **3** | **27** |
| 7 | F=ABC+A’BC’+AB’C’ | 8 | 40 | 7 | 28 | 8 | 48 | 6 | 30 | 3 | 105 | 3 | 93 | **5** | **45** |
| 8 | F=A | 1 | 5 | 1 | 4 | 1 | 6 | 1 | 5 | 1 | 35 | 1 | 31 | **1** | **9** |
| 9 | F=AB+BC+AC | 9 | 45 | 9 | 36 | 6 | 36 | 5 | 25 | 5 | 175 | 1 | 31 | **1** | **24** |
| 10 | F=AB+B’C | 6 | 30 | 6 | 24 | 5 | 30 | 1 | 5 | 1 | 35 | 1 | 31 | **1** | **9** |
| 11 | F=AB+BC+A’B’C’ | 4 | 20 | 4 | 16 | 3 | 18 | 6 | 30 | 2 | 70 | 2 | 62 | **2** | **18** |
| 12 | F=AB+A’B’ | 2 | 10 | 2 | 8 | 2 | 12 | 2 | 10 | 2 | 70 | 1 | 31 | **1** | **9** |
| 13 | F=ABC+A’B’C+AB’C’+A’BC’ | 2 | 10 | 2 | 8 | 2 | 12 | 3 | 15 | 2 | 70 | 2 | 62 | **1** | **24** |
|  | **AVERAGE** | **4.69** | **23.4** | **4.38** | **17.5** | **4.23** | **24** | **3.15** | **16.9** | **2.15** | **75.3** | **1.84** | **57.2** | **1.76** | **18.23** |

\*G – Number of gates; \*QC – Quantum Cost

1. DESIGN OF FAULT-TOLERANT REVERSIBLE FLOATING POINT DIVISION

**2.1 The Proposed Division Methodology**

A proposed fault tolerant floating point division unit consists of the following elements: multiplexer, Parallel In Parallel Out (PIPO) left shift register, adder/subtractor unit, and rounding and normalization registers. All these functional units are designed as fault-tolerant.

For n bit D is a dividend (2n bits to store remainder and quotient after division), V is divisor, and Q is quotient register,

**Algorithm:**

Inputs: D (Dividend); V (Divisor) and Sel=0.

Outputs: R (Remainder) and Q (Quotient)

**Steps:**

1. Initial: Clk=High; SP=0; Sel=0; Count=0; D=0; V=0 (Registers are Initialized)
2. If (Clk)

If (SP= =0)

n-bit Inputs are parallel loaded in operand registers(D & V).

1. Else if (SP= =1 & Hold = = 0& Count <n)

The operands are forwarded to n-bit parallel adder (as per non-restoring algorithm 2’s complement addition).

The output of the previous step is loaded into the D register and serial left shift one position in PIPO shift register.

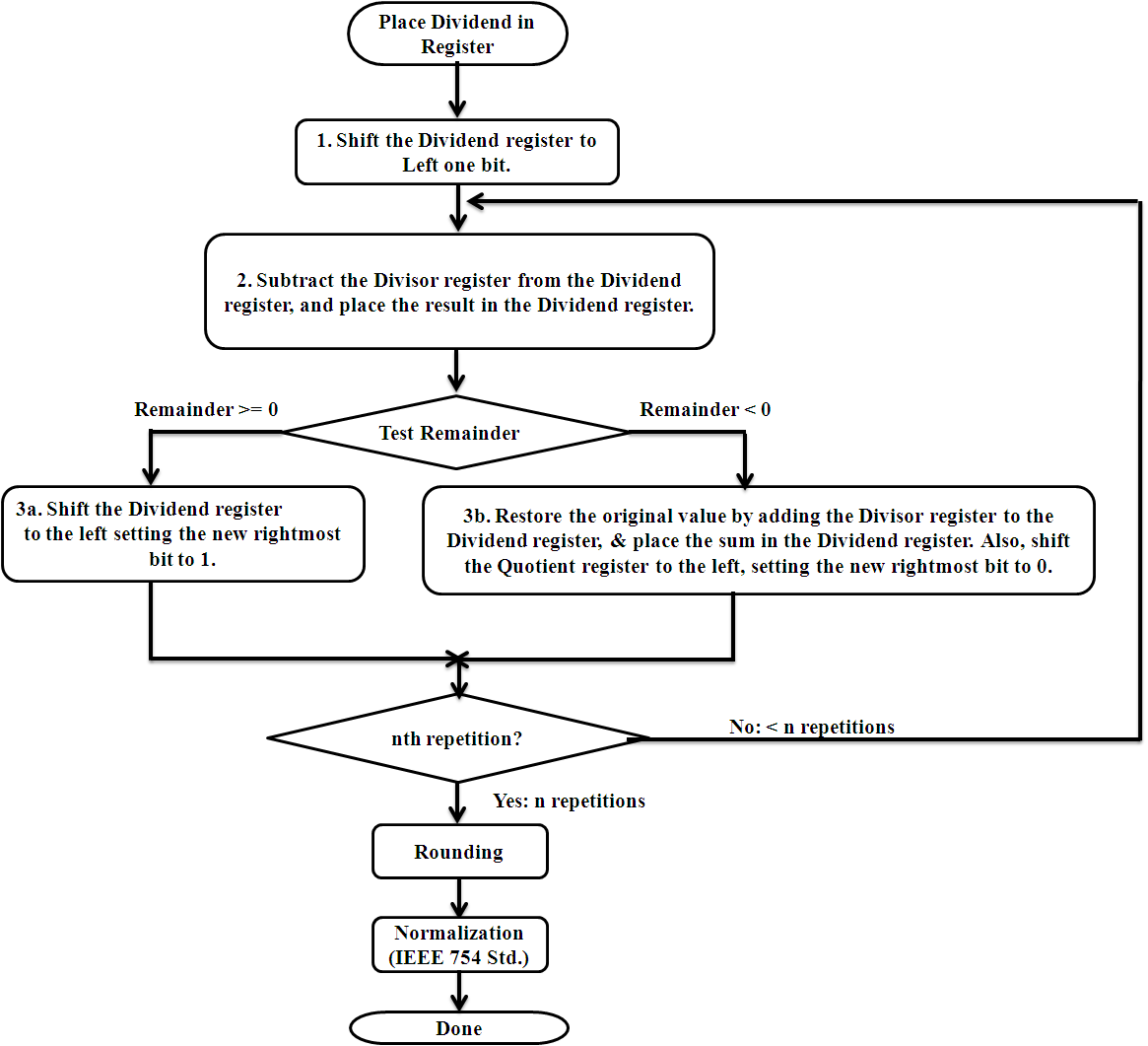
1. If the partial result is positive (MSB=0) set Q0=1; otherwise set Q0=0;
2. Count = Count + 1;
3. If (Count>=n)

If the result is negative; restore D and do rounding and normalization.

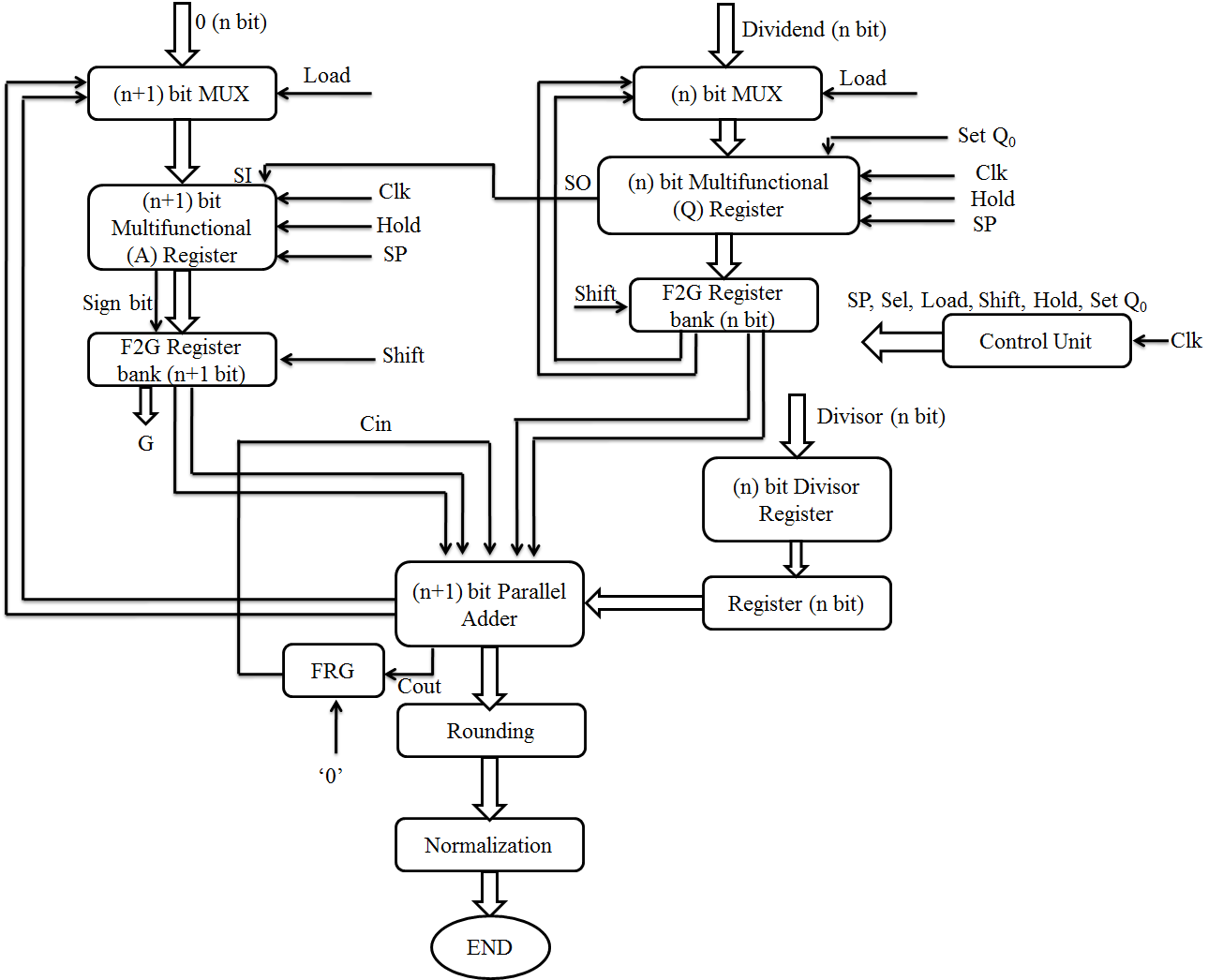
1. Dividend register MSB = Remainder; LSB = Quotient.
2. End.

**2.2 Working Principle:**

The complete data path of fault-tolerant floating point division is shown in Fig.3, which is derived from the algorithmic structure of Fig.2. The significant units are multiplexers, registers (F2G), parallel adder, rounding and normalization units. The necessary control signals are Clk, load, sel, SP, set q0, shift and hold. These signals are released from the control unit at the appropriate time based on the Clk timing.



**Fig.2: Flow Diagram of Floating Point Division**



**Fig.3: Fault-Tolerant Floating Point Division Unit**

**2.3 Key Elements of Reversible Fault-Tolerant Division Unit**

The major functional units of reversible fault-tolerant division unit are multiplexers, operands registers, adder/subtractor, PIPO register, rounding and normalization registers. These functional units are being constructed using fault-tolerant gates; thereby the circuit becomes a fault-tolerant one. The output behavior of the multifunctional register is tabulated in Table 4 which is derived from equation (1). and Performance improvement of D-Latch using proposed gates are shown in Table 5.

**Qi+ = Hold’SP’Ii + Hold’SP.Qi-1 + Hold.Qi ……. (1)**

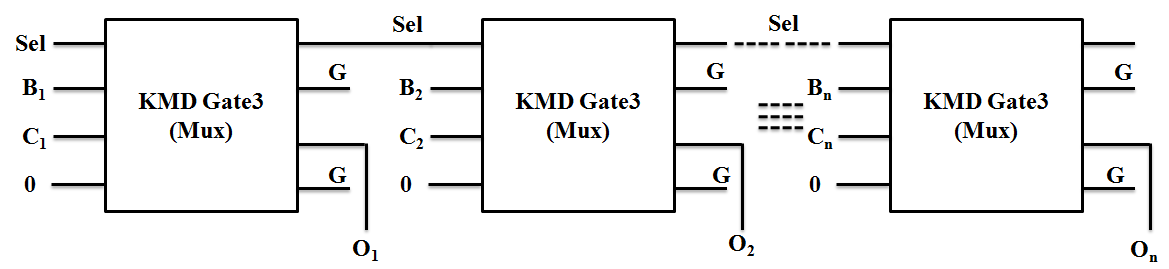
**Table 4: Truth Table for Multifunctional Register**

|  |  |  |
| --- | --- | --- |
| **Hold** | **SP** | **Qi+ (Next Output)** |
| 0 | 0 | Ii (Input Loaded to Register in Parallel) |
| 0 | 1 | Qi-1 (Left Shift & LSB receiving input from Serial Input) |
| 1 | X | Qi (Maintaining Previous Value) |

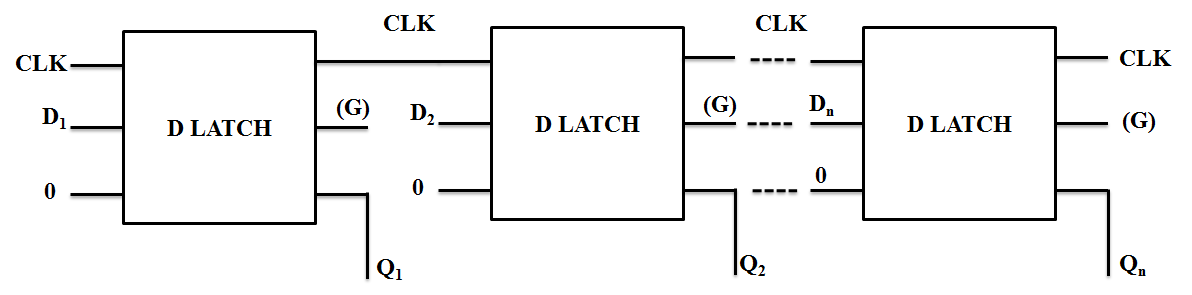
**Table 5: Performance analysis of D-latch**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Parameters** | **[2]** | **[3]** | **[4]** | **Proposed design** | **Improvement % w.r.t** | | |
| **[2]** | **[3]** | **[4]** |
| **Quantum cost** | 14 | 10 | 47 | 9 | 35% | 10% | 80.8% |
| **Garbage outputs** | 4 | 2 | 6 | 1 | 75% | 50% | 83.3% |
| **Gate Count** | 3 | 3 | 7 | 1 | 66.6% | 66.6% | 85.7% |

The hardware architecture of the Multiplexer, D-Latch based PIPO register, multi-functional register and reversible adder are shown in Fig.4, Fig.5, and Fig.6.

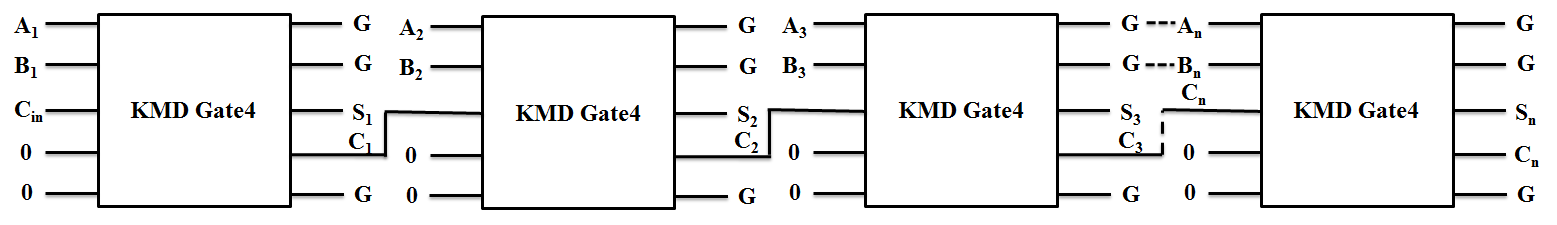


**Fig.4: 2-input n-bit Multiplexer**

****

**(a)**

|  |  |
| --- | --- |
|  |  |
| **(b)** | **(c)** |
| **Fig.5: (a) n-bit PIPO Register (b) Construction of Multifunctional Register and (c) Symbol of Multifunctional Register** | |



**Fig.6: n-bit Fault-Tolerant Reversible Adder**

**2.4 Results and Discussion**

The cost and other parameters calculated for individual units of the n-bit division unit is shown in Table 6. The major components are derived from KMD Gate 3, F2G, and KMD Gate 4. Since KMD Gate 3 is utilized to construct the multiplexer and multifunctional register, the uniformity of the divider is majorly improved.

**Table 6: Performance Measure Calculations of individual modules of the division unit**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Module** | **Number of Bits** | **Gates Used** | **Number of Gates** | **Delay** | **Quantum Cost** | **Garbage Output** | **Constant Input** |
|  | Multiplexer | n | KMD Gate 3 | n | n | 6n | 2n | n |
| n+1 | n+1 | n+1 | 6n + 6 | 2n+2 | n+1 |
|  | Multifunctional Register | n | 3n | 3n | 18n | 5n | 3n |
| n+1 | 3n+1 | 3n+1 | 18n+18 | 5n+5 | 3n+3 |
|  | Divisor Register | n | F2G | n | n | 2n | 2n | n |
|  | Parallel Adder | n+1 | KMD Gate 4 | n+1 | n+1 | 12n+12 | 3n+3 | 2n |
|  | Register | n | F2G | n | n | 3n | - | n |
| n+1 | n+1 | n+1 | 3n+3 | n+1 | n+1 |
|  | Other Gates | 1 | Fredkin | 1 | 1 | 5 | 2 | 1 |
|  | **Total Cost** | | | **12n+4** | **12n+4** | **68n+44** | **20n+13** | **13n+6** |

The estimated performance measurement of n-bit division unit is tabulated in Table 7. It is observed that the dependency factor 'n' – the number of bits is greatly reduced by the proposed method with respect to the existing [5, 6].

**Table 7: Comparison of performance measures for n-bit Division Unit**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameters** | **Existing [25]** | | **Existing [26]** | | **Proposed** |
| **Restoring** | **Non-Restoring** | **Conventional Division Array** | **High Speed Division Array** |
| **No. of Gates** | 18n+23 | 18n+21 | 3(n+2)2+2n/4 | (n+2)(3n+11)/2 | **12n+4** |
| **Delay** | 17n+20 | 17n+18 | 3(n+2)2/2 | - | **12n+3** |
| **Garbage outputs** | 12n+18 | 12n+16 | (n+2)2/2 | (n+2)(3n+22)/4 | **20n+13** |
| **Quantum Cost** | 75n+60 | 75n+53 | 4(n+2)2+n/2 | (n+2)(7n+27)/2 | **68n+44** |
| **Constant Input** | 11n+14 | 11n+12 | (n+1)2+1/2 | (n+2)(3n+14)/4 | **13n+6** |

1. **DESIGN OF INTEGRATED REVERSIBLE FAULT-TOLERANT ARITHMETIC AND LOGIC UNIT**

**3.1 Proposed Reversible Fault-Tolerant ALU Architecture**

The ALU is an inherent building block of a computing device in today’s scenario. The expected requirement of an ALU must perform maximum possible operations with minimum hardware complexity. The proposed ALU structure is shown in Fig 2. The significant signals of this ALU are three inputs (A, B & Cin), Select lines (SA, SB & SOP) and Control Signals (Constant Inputs). The major functional units are Inverter, AND, OR, XOR gates and a 1-bit adder. The multiplexers are acting as a data router to forward the necessary signals from input to output result.

The proposed structure has 3 major modules;

* **Input module**: the 2\*1 multiplexer selects either True of Complement form of the Input and forwards it to the next computing module.
* **Data Processing module**: It does the arithmetic and logical operations on the input data come from the input module. The possible list of arithmetic, logical operations are shown in Table 5 & 6.
* **Output module**: The data received from various functional modules are available in the output module; the select line of the 4\*1 multiplexer decides which output has to be forwarded to the final Result.

The proposed structure is having two advantages over [7, 8],

* The same structure is performing both arithmetic and logical operations.
* The input signals (A, B & Cin) are functioning as constant inputs also, which in turn reduces the number of constant inputs.

The ALU is performing most popular arithmetic as well as logical operations. The possible list of operations and its related control signals are listed in Table 8 & Table 9. The Select lines (SA, SB & SOP) and the input signal (A, B & Cin) are providing various functionalities like Transfer, Addition, Increment, Subtraction and 1’s complement.

**Table 8: Reversible Arithmetic operations**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SA** | **SB** | **SOP** | | **RESULT** | **OPERATION** | **CONSTANT INPUTS** | **INPUTS** |
| 0 | 0 | 0 | 0 | A | TRANSFER A | B=1 | A=A |
| 0 | 0 | 1 | 0 | A+B | ADD | Cin=0 | A=A; B=B |
| 0 | 0 | 1 | 0 | A+B+1 | ADDITION WITH CARRY | Cin=1 | A=A; B=B |
| 0 | 0 | 1 | 1 | A+1 | INCREMENT A | B=1 | A=A |
| 0 | 1 | 1 | 0 | A+B’ | SUB | - | A=A; B=B |
| 0 | 1 | 1 | 1 | A+B | 1’S COMPLEMENT | - | A=A; B=B |
| 1 | 0 | 0 | 0 | B | TRANSFER B | A=0 | B=B |
| 1 | 0 | 1 | 0 | A⊕B⊕C | SUM | Cin=Cin | A=A; B=B |
| 1 | 0 | 1 | 0 | (A.B)+(A+B).C | CARRY | - | A=A; B=B; Cin=Cin |
| 1 | 0 | 1 | 1 | A+B | 1’S COMPLEMENT | - | A=A; B=B |
| 1 | 1 | 1 | 1 | B+1 | INCREMENT B | A=0 | B=B |

**Table 9: Reversible Logical operations**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SA** | **SB** | **SOP** | | **RESULT** | **OPERATION** | **CONSTANT INPUTS** | **INPUTS** |
| 0 | 0 | 0 | 0 | AB | AND | - | A=A; B=B |
| 0 | 0 | 0 | 1 | A | B | OR | - | A=A; B=B |
| 0 | 0 | 1 | 0 | 0 | Constant | A=0; B=0; | - |
| 0 | 0 | 1 | 1 | 1 | Constant | A=1; B=0; | - |
| 0 | 1 | 0 | 1 | A | COPY | B=1 | A=A |
| 0 | 1 | 1 | 1 | A⊕B | EXOR | - | A=A; B=B |
| 1 | 0 | 0 | 1 | B | COPY | A=1 | B=B |
| 1 | 1 | 0 | 0 | (A.B)’ | NAND | - | A=A; B=B |

Using the proposed reversible ALU architecture, 18 distinct operations (10 Arithmetic & 8 Logical) could be done; whereas in [7, 8], only 16 distinct operations are possible as shown in Table 10.

**Table 10: Number of Operations in Proposed Reversible ALU**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE OF OPERATION** | **RM [7]** | **RUG[8]** | **PROPOSED ARCHITECTURE** |
| Arithmetic Operation | 7 | 7 | **10** |
| Logical Operation | 9 | 9 | **8** |
| **Total** | 16 | 16 | **18** |

**3.2 Results and Discussion**

From the Table 11 and Table 12, the quantum cost of the proposed ALU system is tremendously reduced up to 99; which is approximately 69% better than the existing methodology. Also, the number of gates needed to design the ALU is also less compared to [7, 8], which is 21% of improvement. More functionality could be obtained from the KMD gates by changing the constant inputs; which increases the garbage output of the system; that is the reason behind more number of garbage outputs in the proposed method than the available methods.

**Table 11. Performance measure comparison of proposed and existing ALU**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ARCHITECTURE** | **ARITHMETIC** | | **LOGICAL** | | **EXISTING ALU ARCHITECTURE** | | **PROPOSED ALU ARCHITECTURE** | |
| **METHOD** | **RM [7]** | **RUG [8]** | **RM [7]** | **RUG [8]** | **RM [7]** | **RUG [8]** | **APPROACH 1** | **APPROACH 2** |
| **QUANTUM COST** | 140 | 67 | 140 | 94 | 320 | 197 | 118 | 99 |
| **CONSTANT INPUTS** | 2 | 0 | 2 | 1 | 10 | 6 | 6 | 7 |
| **GARBAGE OUTPUTS** | 6 | 3 | 7 | 6 | 15 | 11 | 21 | 22 |
| **NO.OF. GATES**  **USED** | 4 | 3 | 4 | 4 | 14 | 14 | 11 | 11 |
| **LOGICAL CALCULATION** | 2α+8β+3γ | 3α+β | α+9β+4γ | α+6β+6γ | 8α+20β+9γ | 9α+12β+11γ | 12α+14β+6γ | 12α+14β+6γ |

**Table 12. Percentage of improvement in proposed ALU.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Improvement w.r.t. RM [7]** | | **Improvement w.r.t. RUG [8]** | |
| **APPROACH 1** | **APPROACH 2** | **APPROACH 1** | **APPROACH 2** |
| **QUANTUM COST** | **63%** | **69%** | **40%** | **49%** |
| **CONSTANT INPUT** | **40%** | **30%** | 0% | - |
| **NO.OF.GATES USED** | **21%** | **21%** | **21%** | **21%** |

1. **DESIGN OF FAULT-TOLERANT REVERSIBLE VEDIC MULTIPLIER IN QUANTUM CELLULAR AUTOMATA**

**4.1 Procedure for Vedic Multiplication:**

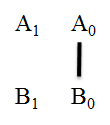
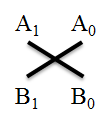
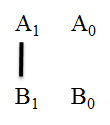
1. The Least significant bit of the two numbers is multiplied to form the LSB of the result (Vertical).
2. The next higher bits are cross multiplied with each other and the results are added together to produce next digit of the multiplication.
3. The carry part of the step 2 is added to the next level sum output to form the next higher bit.
4. All the bits are processed in the same way as in step 2 and step 3 until the last digit of the operand is reached.
5. Record the final product of the Vedic multiplication.

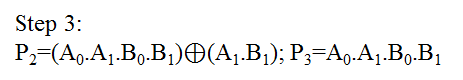
***2×2 Vedic Multiplier***

A two bit multiplication of two numbers A×B could be carried out in the following manner. The logical expression of the final product is,

P0=A0.B0; P1=(A1.B0) ⊕(A0.B1); P2=(A0.A1.B0.B1)⊕(A1.B1); P3=A0.A1.B0.B1

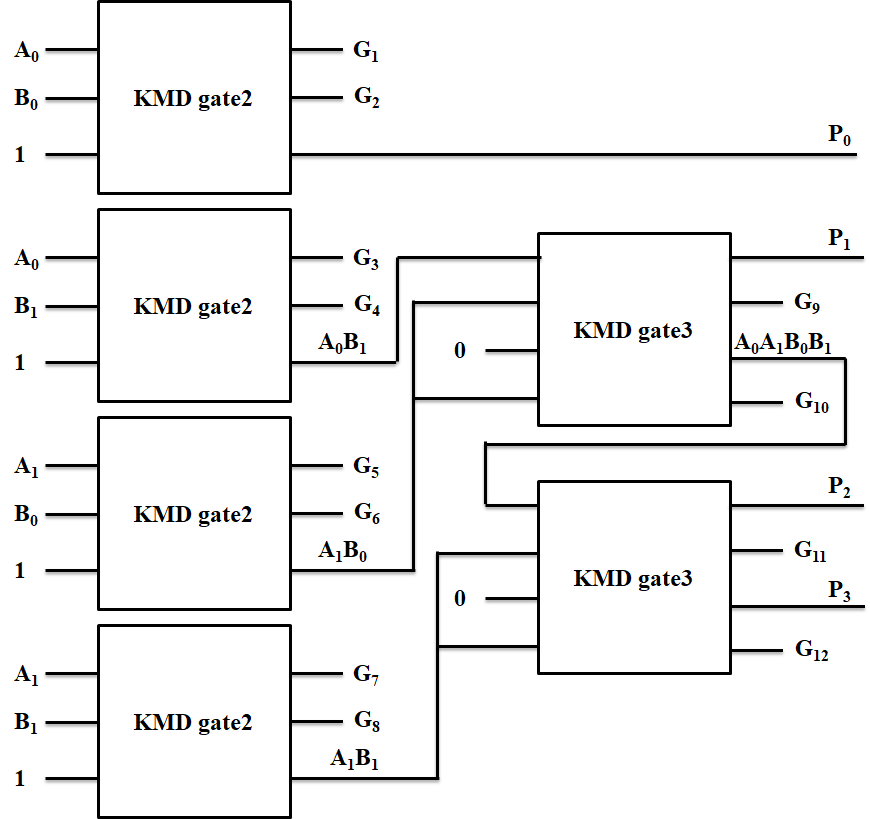
It requires 4 AND operations and 2 ADD operations. Graphically, it is represented as in Fig.7,

**Fig.7. Graphical representation of 2×2 Vedic multiplication steps**

The corresponding hardware architecture of the 2×2 Vedic multiplier using KMD gates is depicted in Fig.8.

****

**Fig.8. The Reversible Logic structure of 2×2 Vedic multiplier using KMD gates**

**4.2 Results and Discussion**

The 2×2 Vedic multiplier has AND gate and half adders as described in earlier section. The total cost for construction in approach1 is 110. Similarly, the higher order multipliers are constructed from the lower order multipliers. The cost calculation of individual modules of the multiplier and the total cost are listed in Table 13 and Table 14.

**Table 13: Performance of Fault-tolerant Reversible Vedic Multiplier (Approach 1)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Functional Module** | **Quantum Cost** | **Garbage Output** | **Constant Input** | **Number of Gates (Primitive)** | **Number of Gates** | **Total Cost (QC + GO + CI + NoG)** | **Logical Calculations** |
| **2-bit Fault-tolerant Reversible Vedic Multiplier** | | | | | | | |
| **AND gate** | 10 | 2 | 1 | 6 | 1 | 19 | 2α+4β+3γ |
| **Half Adder** | 9 | 2 | 1 | 5 | 1 | 17 | 5α+2β+1γ |
| **Full Adder** | 22 | 3 | 2 | 8 | 1 | 35 | 6α+5β+2γ |
| **2-bit VM** | 58 | 12 | 6 | 34 | 6 | 110 | 18α+20β+14γ |
| **4-bit Fault-tolerant Reversible Vedic Multiplier** | | | | | | | |
| **4-bit Adder** | 88 | 12 | 8 | 32 | 4 | 140 | 24α+20β+8γ |
| **4-bit VM** | 496 | 84 | 48 | 232 | 36 | 860 | 144α+140β+80γ |

**Table 14: Performance of Fault-tolerant Reversible Vedic Multiplier (Approach 2)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Functional Module** | **Quantum Cost** | **Garbage Output** | **Constant Input** | **Number of Gates (Primitive)** | **Number of Gates** | **Total Cost (QC + GO + CI + NoG)** | **Logical Calculation** |
| **2-bit Fault-tolerant Reversible Vedic Multiplier** | | | | | | | |
| **AND Gate (Fredkin)** | 3 | 2 | 1 | 1 | 1 | 6 | 2α+4β+1γ |
| **Half Adder** | 9 | 2 | 1 | 5 | 1 | 12 | 5α+2β+1γ |
| **Full Adder** | 22 | 3 | 2 | 8 | 1 | 27 | 6α+5β+2γ |
| **2-bit VM** | 30 | 12 | 6 | 14 | 6 | 48 | 18α+20β+8γ |
| **4-bit Fault-tolerant Reversible Vedic Multiplier** | | | | | | | |
| **4-bit Adder** | 88 | 12 | 8 | 32 | 4 | 108 | 24α+20β+8γ |
| **4-bit VM** | 384 | 84 | 48 | 152 | 36 | 516 | 144α+140β+48γ |

The designed Fault-tolerant reversible Vedic multiplier cost performance is compared with the existing conventional and Reversible Vedic multipliers and they are shown in Table 15. It can be seen that performance of the proposed method shows an improvement compared to the existing multipliers in terms of quantum cost, garbage output, constant input, and logical calculations.

**Table 15: Performance comparison of Existing multipliers with Proposed Multiplier**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance Measures** | **Conventional Multipliers** | | **Vedic Multiplier** | | | **Proposed Vedic Multiplier** | |
| **[**9**]** | **[**10**]** | **[**11**]** | **[**12**]** | [13] | **Approach1** | **Approach2** |
| Quantum Cost | 177 | 137 | 164 | 162 | 128 | 110 | 48 |
| Garbage Output | 49 | 28 | 43 | 62 | 40 | 12 | 12 |
| Constant Input | 49 | 28 | 33 | 29 | 31 | 6 | 6 |
| Number of gates | 28 | 28 | 33 | 37 | 31 | 14 | 14 |
| Logical Calculations | 128α+78β+28γ | 71α+36β | - | - | - | 18α+20β+14γ | 18α+20β+8γ |
| Area (µm2) | - | - | - | - | - | 3.7 | 3.7 |

**REFERENCES:**

1. Kalyan S. Perumalla, “Introduction to Reversible Computing”, *CRC Press*, 2014.
2. A.V. Anantha Lakshmi, Gnanou Florence Sudha, “Design of a reversible floating-point square root using modified non-restoring algorithm”, *Microprocessors and Microsystems*Vol. 50, 2017, pp. 39–53.
3. Belayet Ali, Mosharof Hossin, Eneyat Ullah, “Design of Reversible Sequential Circuit Using Reversible Logic Synthesis”, *International Journal of VLSI design & Communication Systems (VLSICS)* Vol.2 (4), December 2011, pp. 37- 45.
4. Hafiz Md. HasanBabu, Md. Solaiman Mia, “Design of a compact reversible fault tolerant division circuit”, *Microelectronics Journal*, Vol. 51, 2016, pp.:15–29.
5. Faraz Dastan, Majid Haghparast, “A novel nanometric fault tolerant reversible divider”, *International Journal of the Physical Sciences* Vol. 6(24), 16 October 2011, pp. 5671-5681.
6. L.Jamal, H.M.H.Babu, “Efficient approaches to design a reversible floating point divider”*, Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*, Vol. 13, 2013, pp.3004–3007.
7. BibhashSen, Manojit Dutta, Mrinal Goswami, Biplab K.Sikdar, “Modular Design of stable reversible ALU by QCA multiplexer with increase in programmability”, Microelectronics Journal, Vol. 45, 2014, pp.1522–1532.
8. Trailokya Nath Sasamala, Ashutosh Kumar Singh b, Anand Mohanaa, “Efficient design of reversible ALU in quantum-dot cellular automata”, Optik Journal, Vol. 127, 2016, pp.6172–6182.
9. Valinataj M. (2017). Novel parity-preserving reversible logic array multipliers. *Journal of Supercomputing*. **73(**11): 4843-4867. DOI: https://doi.org/10.1007/s11227-017-2057-z
10. Haghparast M., Mohammadi M., Navi K. & Eshghi M. (2009). Optimized Reversible Multiplier Circuit. *Journal of Circuits, Systems, and Computers.* **18**(2): 311-323. DOI: https://doi.org/10.1142/s0218126609005083
11. Saligram R. & Rakshith T.R (2013). Optimized Reversible Vedic Multipliers for High Speed Low Power Operations. *IEEE Conference on Information and Communication Technologies (ICT 2013)*, pp.809-814. DOI: https://doi.org/10.1109/cict.2013.6558205
12. Rakshith T.R. & Saligram R. (2013). Design of High Speed Low Power Multiplier using Reversible logic: a Vedic Mathematical Approach. *International Conference on Circuits, Power and Computing Technologies (ICCPCT-2013)*, pp.775-781. DOI: https://doi.org/10.1109/ICCPCT.2013.6528848
13. Gowthami P. & Satyanarayana R.V.S. (2018), Performance Evaluation of Reversible Vedic Multiplier, *ARPN Journal of Engineering and Applied Sciences*, **13**(3): 1002-1008.