Design And Development of Efficient Reversible Floating Point Arithmetic unit

Jenil Jain
Electronics Engineering Department,
G. H. Raisoni College Of Engineering, Nagpur
jeniljain1008@gmail.com

Rahul Agrawal
Electronics Engineering Department,
G.H.Raisoni College Of Engineering,Nagpur
rahul.agrawal@raisoni.net

***Abstract*- For calculation or representation of very large or small numbers, large range is essential. These values can be represented using the IEEE-754 standard based floating point arithmetic representation. The paper presents efficient approach towards designing of high speed floating point unit using reversible logic. Programmable reversible logic design is trending as a prospective logic design style for implementation in recent nanotechnology and quantum computing with low impact on circuit heat generation. There are various reversible implementations of logical and arithmetic units have been proposed in the existing research, but very few reversible floating-point designs has been designed. Floating-point operations are used very frequently in nearly all computing disciplines. The analysis of proposed reversible circuit can be done in terms of quantum cost, garbage outputs, constant inputs, power consumption, speed and area.**

***Keyword*- Reversible logic, Garbage output, Quantum cost, Floating Point, Arithmetic Unit etc**

I. INTRODUCTION

An arithmetic circuit which performs digital arithmetic operations has many applications in digital coprocessors, application specific circuits, etc. Because of the advancements in the VLSI technology, many complex algorithms that appeared impractical to put into practice have become easily realizable today with desired performance parameters so that new designs can be incorporated. Modern computers use conventions for representing non integer numbers, the most widespread of which is the IEEE 754 Standard for Floating-Point calculations. This standard defines binary representation for floating-point numbers of varying precision, giving specific examples of the binary32 (or single precision) format, binary64 (or double precision) format, and it defines operations on floating-point numbers. The essential components in IEEE 754 standard floating point numbers are the sign, the exponent, and the mantissa.

Table I-Bit Range For Floating-Point Values

	Sign	**Exponent**	**Fraction**	**Bias**
Single precision	1(31)	8(30-23)	23(22-00)	127
Double precision	1(63)	11(62-52)	52(51-00)	1023

Table II-Floating Point Number Representation

32 Bit		
Sign	**Exponent**	**Mantissa**
1 bit	8 bit	23 bit

There are four types of exceptions that arise during floating point operations. Whenever the result cannot be shown as a definite number in the precision format of the destination the Overflow exception is occurred. The Underflow exception take place when an intermediate result is very small to be calculated correctly. When zero divides a finite nonzero number, the Division by zero exception arises. The Invalid operation exception is raised if the given inputs are not appropriate for the operation to be performed. Reversible logic is a promising computing design paradigm which presents a method for constructing computers that produce no heat dissipation. Reversible computing emerged as a result of the application of quantum mechanics principles towards the development of a universal computing machine. The basic principle of reversible computing is that a bijective device with an identical number of input and output lines will produce a computing environment where the electrodynamics of the system allow for prediction of all future states based on known past states, and the system reaches every possible state, resulting in no heat dissipation. Reversible computing differs from conventional computing in that it performs the computation in a logically reversible way: The output of a (fully) reversible circuit always uniquely identifies the input. Circuits can take advantage of this logical reversibility to reduce power by reusing the information instead of discarding it: Landauer showed that any time a bit of information is discarded, it equates to some quantum of energy lost as heat [1].Moreover in 1973, Bennett has shown that this energy loss can be reduced or even removed if the circuits are designed using reversible gates [2].

The remaining paper is structured as follows. Section II consist of important details of reversible logic design, with coverage of some reversible logic primitive gates. Section III provides details of floating-point addition algorithm and architecture. Section IV outlines approaches towards multiplier and divider design. Section v presents our final measurements, a brief analysis of the architecture, and direction for future work. Section VI concludes the paper with our list of references.

978-1-4799-1797-6/15 $31.00 © 2015 IEEE
DOI 10.1109/CSNT.2015.215

II. REVERSIBLE LOGIC PRIMITIVE

Many traditional logic gates such as the AND, OR, NAND, NOR, and XOR gates are fundamentally irreversible. That is to say that the output combination of any of these gates does not expose the input combination that caused the output. Thus we have a need for primitive reversible logic gates.

A.Reversible Gate –

Reversible gates are the circuits having one to-one relationship between vectors of input and output. Therefore from output vector state we can reconstruct the vector of input states.

B. Quantum Cost

Every quantum circuit is built using 1X1 and 2X2 quantum primitives and its cost is calculated as a total sum of 2X2 gates used since 1X1 gate has no cost, i.e., zero. Basically the quantum primitives are matrix operation which is applied on qubits state. All the gates of the form 2X2 has equal quantum cost and the cost is unity, i.e., one [3]. Since every reversible gate consists of 1X1 or 2X2 quantum gate, the quantum cost of a reversible design calculates the total number of 2X2 gates used. The quantum costs of Feynman gate [4], Peres gate [5] and DPG gate (as full adder) [6] are one, four and six respectively.

C. Garbage Output

Unwanted or unused output of a reversible gate (or circuit) is known as *garbage output*, i.e., the output(s) which is(are) needed only to maintain the reversibility is(are) known as garbage output(s).

D. Delay

The maximum number of gates in a route from any input signal line to any output line is known as delay of a circuit. At the beginning, each gate performs the design computation in one unit time. Secondly, all inputs to the circuit are known before the computation begins.

E.Popular Reversible Gate –

1) Feynman Gate:

The input and output vectors of 2x2 Feynman Gate (FG) [4] are $(Ip)v$ and $(Op)v$ respectively and can be defined as follows:

$(Ip)v$ ={a} and

(Op)v={a xor b}

2)Peres Gate:

The input and output vectors of 3*x*3 Peres Gate (PG) [5] are $(Ip)v$ and O_v respectively and can be defined as follows:

$(Ip)v$ = {a,b,c}and

$(Op)_v$ = {a,a xor b,ab xor c}

The 3 bit Peres gate has Quantum cost of 4.

3)Double Peres Gate-

Input vector, $(Ip)v$ and output vector,$(Op)v$ of 4x4 Double Peres Gate (DPG) [6] are defined as follows:

$(Ip)v$ = {a,b,c,d} and

(Op)v = {a,a xor b,a xor b xor d,(a xor b)d xor ab xor c}

4) Fredkin Gate-

Input vector, $(Ip)v$ and output vector,$(Op)v$ of 3x3 Fredkin Gate(FR) are defined as follows:

$(Ip)v$ ={a,b,c} and

(Op)v={a,a'b xor ab,a'c xor ab}

5)HNG Gate-

Input vector,$(Ip)v$ and output vector, $(Op)v$ of 4x4 HNG are defined as follows:

(Ip)v={a,b,c,d} and

(Op)v={a,b,a xor b xor c,(a xor b)c xor ab xor d}

III.FLOATING POINT ADDITION

Given two floating-point numbers to be added, the IEEE754 Standard for Floating-Point Arithmetic details how their sum can be found [7].First, if the exponents are not equal, the smaller is incremented until it aligns with the larger. To align the floating-point number with the smaller exponent without altering its value, its respective trailing significand must be shifted one place to the right for every time the exponent is incremented. Once the exponents are equal, the significands can be summed. The sum is then normalized and rounded. Fig. 1 illustrates the block-level schematic of the architecture our proposed reversible floating-point adder design uses.

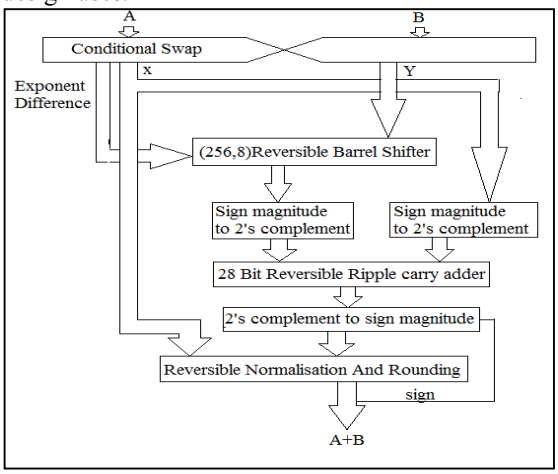

Fig-1 Proposed design of reversible addition architecture

A)Reversible Conditional Swapping

The first step in our reversible floating-pointer adder architecture is to swap the floating-point operands conditionally and reversibly. The rest of the architecture operates assuming that X is the floating-point number with the greater exponent, and Y is the floating-point number that possibly needs to be aligned with X. The exponents of the two floating-point numbers both are unpacked and expanded to nine bits. In order to find their difference the

exponent that is the minuend is complemented using two's complement, and with it the difference is calculated. The nine HNG gates implement the reversible subtracter circuit [8]. The sign bit of the difference of the exponents acts as the condition (control) by which the two entire floating-point numbers are swapped: If expA<expB ,then the floating-point numbers will swap positions, otherwise expA≥expB and the floating-point numbers are passed through to the next stage without swapping.

B)Reversible Barrel Shifter-

In our proposed system we require (256,8) reversible barrel shifter. But due to complexity & space problem we shown here design for (4,2) reversible barrel shifter. The circuit works as follows: Each stage of Fredkin gate shifts the input according to the control value of ***sk.*** Suppose, to design a (4, 2) shifter which takes *i0, i1, i2, i3* as data inputs and ***s0s1*** = 11 as select input. So the input will be shifted 2^0+ 2^1 = 3 times to the left. Thus the sequence of the shift/rotate operation will be *i1i2i3 i0* for the first stage and then *i3i0i1i2* for the next. On the other hand, for the select input ***s0s1*** = 00, the input sequence will remain same for both stages of multiplexing. Thus, each Fredkin gate chooses between two input lines it receives and performs the appropriate operation according to the select input of that particular stage. Hence, for the first stage (Stage 0) of above (4:2) shifter, the first Fredkin gate will either select input *i0* or *i1*, the second one will do either *i*1 or *i*2 and so on. All other stages perform the selection task in the same way.

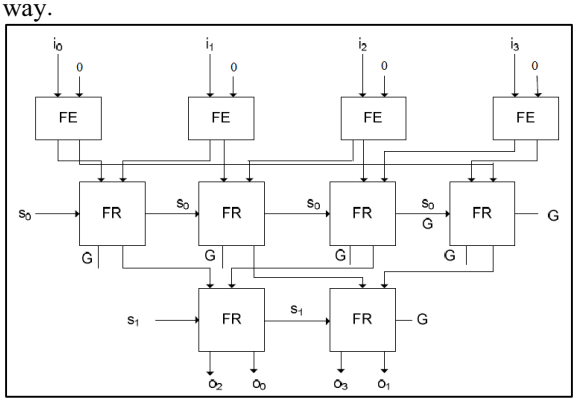

Fig2-Proposed Design of reversible barrel shifter[9]

C)Conversion unit –

we have designed a reversible conversion unit that serves simultaneously as a sign magnitude-to-two's complement unit and a two's complement-to-sign magnitude unit. Of note is that our unit performs a reversible mapping, f, with the following fixed point:

$$f(10\ldots0)=10\ldots0$$

This implies, for an n bit signed integer:

$$f([-2n-1]2's\ complement)=[-0]sign-magnitude$$
$$f([-0]sign-magnitude)=[-2n-1]2's\ complement$$

Our design requires two of the 28 bit reversible converters (one for each trailing significand), and a single 29 bit converter for the output of the 28 bit full adder. Before the ensuing reversible normalization step, this 29 bit sign extended sum is converted back to sign magnitude with a single instance of a 29 bit reversible conversion unit.

D)28-Bit Ripple Carry Full Adder-

For this implementation, we will be using the Peres gate as it is the gate with the lower quantum cost. The Peres gate implementation of Full Adder with its corresponding quantum cost can be seen below:

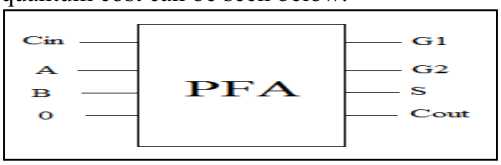

Fig3-Ripple carry full adder[10]

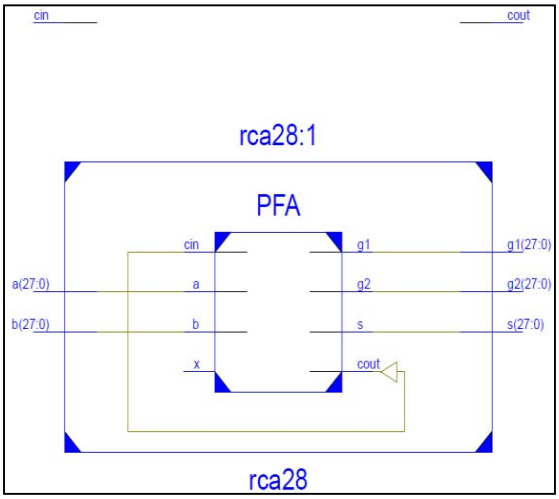

Fig-4 Schematic view of PFA

E. Reversible Normalization

After the sum of the trailing significands has been converted back into sign-magnitude representation, the sign bit is connected directly to the final stage as the sign of the floating-point sum, and the magnitude may need to be normalized. This normalization step may involve either left shifting or right shifting. If a right shift is required, only a right shift of a single position will be required. Otherwise a left shift of possibly several placed may be required. A synchronous floating-point adder architecture might accomplish this behavior using synchronous leading-one detectors and synchronous shift registers, but in keeping with our asynchronous reversible design methodology, we design a completely asynchronous reversible normalization unit.

F. Reversible Rounding

Our reversible rounding unit performs the reversible round toward zero rounding algorithm specified in the IEEE754

standard. This unit is a pseudo unit, in that it consists of no extra hardware: It operates simply by transforming some of the input bits into garbage outputs by ignoring them altogether.

IV.EFFICIENT APPROACHES TOWARDS EFFICIENT FLOATING POINT MULTIPLICATION UNIT AND DIVISION UNIT –

A)Reversible Multiplication Unit-

For to design single precision floating-point multiplier, there is requirement of efficient 24x24 bit integer multiplier. Operand decomposition approach is efficient for the propose reversible design of 32 bit floating point multiplier. To design the reversible 24x24 (AxB) bit multiplier, the values are divided into three subgroups of 8 bits each. Thus, the 24x24 bit reversible multiplication can be performed through nine reversible 8x8 bit Wallace tree multipliers, of which outputs are then summed.There are three conceptual stages in wallance tree multiplication:partial product compression using 4:2 compressors, Partial product generation, full &half adders and then the final addition stage to generate the product. In this work there is requirement of optimization at each of these three stages.

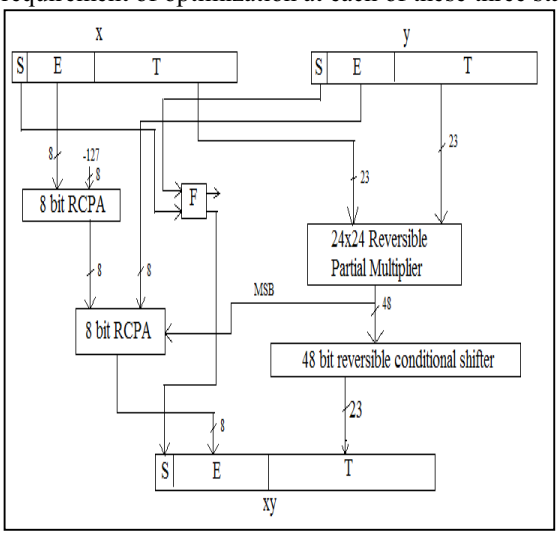

Fig-5 Algorithm for floating point multiplication

B)Reversible Division Unit(Conventional) -

Let,

Dividend, $A = A0{:}A1A2{:::}An$

Divisor, $D = D0{:}D1D2{:::}Dn$

Remainder, $R = R0{:}R1R2{:::}Rn$

Quotient, $Q = Q0{:}Q1Q2{:::}Qn$

The operands are assumed to be positive, normalized fracsigned fractions.

So, $A0 = D0 = 0$ and $A1 = D1 = 1$. The quotient is positive and the partial remainder R is a signed fraction, and $R0$ is the sign bit with R being represented in 2's complement form.

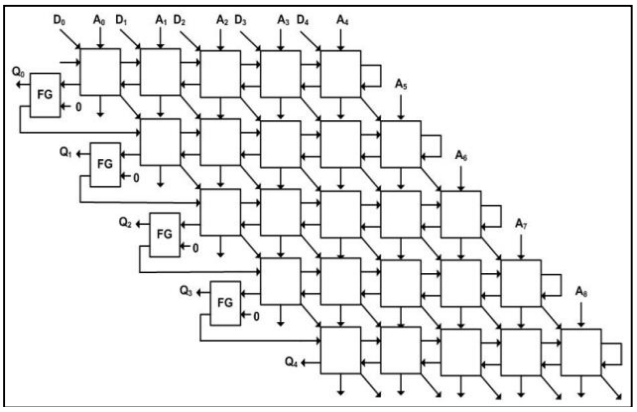

Fig-6 Conventional Division Array :8-bit Dividend and 4-bit Divisor[11]

C) Proposed reversible divider using high speed division array-

The reversible divider using high speed division array is designed by some major modifications of the design previous section. The carry ripple time, which is proportional to n, has been omitted in this design. The partial remainder R is not developed in each row of the array, but is represented by two binary vectors S and C which, if added, would produce the correct partial remainder at that row level. A single carry-look ahead circuit is used to determine from the S and C vectors what the carry sign would be, facilitating the determination of the sign of the resulting partial remainder, the quotient bit for that row level, and the control (add or subtract divisor) to the next row. Subtraction of the divisor is implemented using 2's complement addition as in the conventional array. Three types of cell, namely, A cell, S cell and CLA cell have been used in the high speed array design. The Ajs (dividend) are input from the top, and the complemented Djs (divisor) are input through the diagonal lines. Addition or subtraction is to be performed in the A cells.

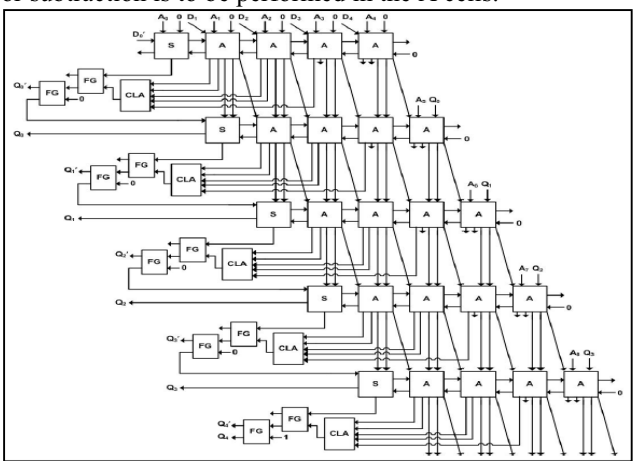

Fig-7 Reversible High Speed Division Array[11]

V. Results And Statistics

We functionally verified each unit in addition algorithm. For multiplication & division unit we provided proposed algorithm. Xilinx 13.2 is used for simulation purpose. Also the analysis is done in terms of quantum cost, garbage outputs, constant inputs, speed and delay parameter. For floating point calculations IEEE 754 standard is used.

A)Reversible addition algorithm-

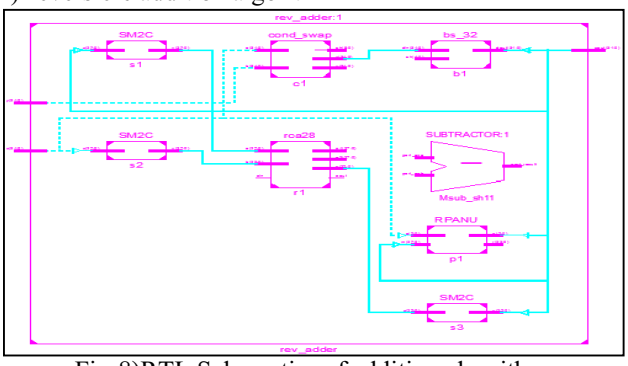

Fig-8)RTL Schematics of addition algorithm

Table III-Cost Measurements For The Reversible modules

Components	Quantum Cost	Constant Inputs	Garbage Outputs
Signed Conversion Unit	140	28	28
28 bit adder	164	52	52

Table IV-Analysis Of Different modules

Designs	Delay	Levels Of Logic
Existing Irreversible Design 28 bit RCA	45.474ns	30
Proposed Reversible Design 28 bit RCA	29.009ns	19
Reversible Barrel shifter design	16.684ns	9
Conditional Swap Unit	46.464ns	33
Signed Conversion Units	9.033ns	3
Reversible Normalization Unit	22.583ns	12
Total addition algorithm	46.544ns	33

VI.CONCLUSION

This paper presents the floating point unit according to IEEE 754 Standard. All modules has been designed in reversible way to reduce power consumption. Analysis of all units has been in terms of quantum cost, garbage outputs, constant inputs, speed and delay parameter. The simulation results of addition unit is provided in this paper. In future the analysis of multiplication unit will be possible.

REFERENCES

[1] R. Landauer, “Irreversibility and heat generation in the computational process,” IBM Journal of Research and Development, 5, pp. 183-191, 1961.

[2] C. H. Bennett, “Logical reversibility of computation,” IBM Journal of Research and Development, pp. 525-532, November 1973.

[3] A. K. Biswas, M. M. Hasan, A. R. Chowdhury, and H. M. H. Babu, “Efficient approaches for designing reversible binary coded decimal adders,” Microelectronics Jounrnal, vol. 39(12), 2008.

[4]R. P. Feynman, “Quantum mechanical computers,” Opt. News, vol. 11(2), pp. 11–20, 1985.

[5] A. Peres, “Reversible logic and quantum computers,” Physical review A, vol. 32, pp. 3266–3276, 1985.

[6] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, “Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis,”IEEE Transactions on, vol. 25, no. 9, pp. 1652 –1663, sept. 2006.

[7]"IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2008, vol., no., pp.1-58, Aug. 29 2008.

[8]M. Haghparast, S. Jassbi, K. Navi, and O. Hashemipour, “Design of a novel reversible multiplier circuit using HNG gate in nanotechnology,” World Applied Sciences Journal, vol. 3, no. 6, pp. 974–978, 2008.

[9] Irina Hashmi, Hafiz Md. Hasan Babu “An Efficient Design of a Reversible Barrel Shifter” 23rd International Conference on VLSI Design,2010

[10] Alejandro Y. Pérez V, “ Reversible Adder Implementation in VHDL” Digital Design Using HDL (ECE 590) Portland State University, Spring 2006

[11] Lafifa Jamal and Hafiz Md. Hasan Babu, “Efficient Approaches to Design a Reversible Floating Point Divider” Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh

[12] David Goldberg, “What Every Computer Scientist Should Know About Floating-Point Arithmetic”, ACM Computing Surveys, Vol 23, No 1, March 1991, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304