Design Of A Compact Reversible Carry Look-Ahead Adder Using Dynamic Programming

1. M.UREKHA ,2. MOHAMMAD ZUBAIR,Assistant Professor
1,2. Department of Electronics and Communication Engineering,Dr.K.V.Subba Reddy College of Engineering For Women
1.ureka.472@gmail.com,2. mdzubair.1345@gmail.com

***Abstract:** This paper presents a new method for designing a reversible carry look-ahead adder (RCLA) based on dynamic programming. In this method, we propose a faster technique for generating carry output, which also outperforms the existing ones in terms of number of operations. In addition, we design a compact reversible carry look-ahead circuit based on the proposed technique. In order to optimize our design, we propose a first ever known Reversible Partial Adder (RPA) circuit with the optimum numbers of the quantum cost and garbage outputs which concurrently produce carry propagation signal, carry generation signal and summation of the inputs. Using RPA as a unit element of RCLA construction, we optimize the designs of RCLA and show that the proposed design is better than the existing ones in terms of the number of gates, quantum cost, garbage outputs and delay with the help of Microwind DSCH 3.5, e.g., the proposed 128-bit adder improves 77.55% on number of gates, 10% on garbage outputs, 2.16% on delay and 77.61% on quantum cost over the existing best one.*

Keywords- Quantum Cost; Garbage Output; Logic Design; Reversible

I.INTRODUCTION

A minimal heat generation of kTln(2) joules of energy per computing cycle requires for logical computing devices were demonstrated by Landauer [1]. This resultant dissipated heat also causes noise in the remaining circuitry. The number of lost bits is directly connected to the dissipated energy. Resultantly, a new pattern with a logical structure consisting of the same number of inputs and outputs along with one-to-one mapping between the input and output states came in computer design. Any device designed to these constraints is known as a reversible logic device which reduces the energy dissipation[1]. Dynamic programming[2] is an optimization methodology that changes a complex problem into a sequence of simpler problems. The multi-stage nature of the optimization procedure is the crucial characteristic of dynamic programming.

Reversible logic has received great attention in the recent years due to its ability to reduce the power dissipation which is the main requirement in low power digital design. It has wide applications in advanced computing, low power CMOS design, Optical information processing, DNA computing, bio information, quantum computation and nanotechnology. Conventional digital circuits dissipate a significant amount of energy because bits of information are erased during the logic operations. Thus, if logic gates are designed such that the information bits are not destroyed, the power consumption can be reduced dramatically. The information bits are not lost in case of a reversible computation. This has led to the development of reversible gates. ALU is a fundamental building block of a central processing unit (CPU) in any computing system; reversible arithmetic unit has a high power optimization on the offer. By using suitable control logic to one of the input variables of parallel adder, various arithmetic operations can be realized. In this project, ALU based on a Reversible low power control unit for arithmetic & logic operations is proposed. In our design, the full Adders are realized using synthesizable, low quantum cost, low garbage output DPeres gates. This project presents a novel design of Arithmetic & Logical Unit using Reversible control unit. These Reversible ALU has been modeled and verified using Verilog and Altera simulator. Comparative results are presented in terms of number of gates, number of garbage outputs, number of constant inputs and Quantum cost.

II. SYSTEM ARCHITECTURE

2.1 Method for carry generation:

A problem f into a set of other problems can be decomposed using dynamic programming where the answer for f can be found in terms of a simple operation from the answers of sub-problems. The dynamic algorithm is completely specified by a set of operations for a set of sub-functions of the computation. In our proposed method, we divide the final carry output into a set of sub carry outputs and solve them individually using dynamic programming algorithm. The fastest adder is the carry look-ahead adder (CLA) and they achieve the speed through parallel carry computations. In a pair of binary sequences, a bit-pair is added and the CLA logic determines whether that bit-pair will generate a carry or propagate a carry. This allows the circuit to "pre-process" the two numbers being added to determine the carry ahead of time. Therefore, there is no delays like the ripple carry effect, when the actual addition is performed.

The method for carry generation is based on dynamic programming algorithm which is described below. The adder is based on the fact that a carry signal will be generated in two cases:

1. When Ai and Bi both bits are 1; or
2. When Ai or Bi bit is 1 and the third input, carry-in Ci is 1.

Thus, we can write:

Cout = Ci+1=Ai. Bi . Ci' + (Ai + Bi).Ci……..3.1

The above expression can also be represented as:

Ci+1=Pi G*i-1+Gi.G*i-1'……..3.2

Here, Gi = Ai . Bi and Pi = Ai Bi, where 0 <= i < n

Applying this to a 4-bit adder, we have:
C0 = 0
C1 = P0 C0 + G0 C0'=P0.0+G0.1=G0=G*0
C2 = P1G*0+G1 G*0' = G*1'
C3 = P2G*1+G2 G*2
C4 = P3G*2 +G3G*3G*2=G*3
The sum signal can be calculated as follows:
Si = Ai ⊕Bi ⊕= Pi Ci⊕
In Table 3.1, the number of required operations to implement the carry generation of the proposed adder technique for various numbers of bits has shown. From this table, we can say that the total number of operations required for the existing carry generation method is much
less than the previous carry generation methods. Algorithm I describes the computational process for the sum and carry operations of the proposed method, where the carry operation of Algorithm I is performed by Algorithm II
Table 3.1 Comparison among the CLAA and the Reversible CLAA carry generation technique in terms of number of operations

No. of bits	Existing	Proposed
4	30	24
8	60	48
16	120	96
32	240	192

Algorithm I: Technique for sum and carry generation of an n-bit carry look-ahead adder:
Input: A, B; (Both are n-bit binary numbers)
Output: Sum (n-bit), Carry (1-bit)
1. Begin
2. For i = 0 to (n-1) do
3. Take one partial adder, Fi
4. Set, Fi.a = Ai ; Fi.b = Bi ; Fi.c = 0 ; Fi.d = 0
5. Output Sumi = Fi.r
6. End For
7. For i = 0 to (n-1) do
8. Pi = Ai ⊕i
9. Gi = Ai.Bi
10. End For
11. Set Carry = Generate_Carry(C0, P0, G0, P1, G1, …, Pn-1, Gn-1) [Algorithm II]
12. End

Alogrithm II: Carry generation algorithm for generate_carry( ) procedure
Input: C0, P (P0, P1, …, Pn-1), G (G0, G1, …, Gn-1)
Output: Carry (1-bit)
1. Begin
2. For i =1 to n do
3. Ci = Pi-1. G*i-2 + Gi-1. G*i-2! where, G*i-2 = Pi-2.G*i-3 + Gi-2.G*i-3!
4. End For
5. Set, Carry = Cn
6. End

2.2 Design of a compact reversible carry look-ahead adder:

In this section, i use a new method for designing a reversible n-bit carry look-ahead adder circuit. To implement this method addition algorithm, carry generation circuit is needed. At first, i propose a new reversible partial adder named RPA to produce carry propagation (P) signal, carry generation (G) signal and sum (S) and then, i present a reversible 4-bit carry look-ahead adder circuit. Finally, i show the circuit of a reversible n-bit carry look-ahead adder using the proposed algorithm for addition. Sections A, B, C, D and E present the proposed reversible partial adder, comparison of reversible partial adders, proposed design of a compact reversible carry look-ahead adder, complexity of the proposed reversible n-bit carry look-ahead adder circuit and performance analysis and simulation results of the proposed design, respectively.

2.2.1 Reversible Partial Adder

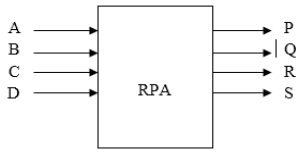


Fig 2.1 Block diagram of Reversible Partial adder, When D=0

In this subsection, a new 4_4 reversible partial adder, namely RPA, is proposed. The input vector, Iv and the output vector, Ov of the proposed circuit are as follows:
Iv={A,B,C,D}; and
Ov = {P=A,Q=A B,R=A B C, S=AB D⊕ ⊕⊕

Fig. 3.1 shows the diagram of the proposed 4_4 RPA. The quantum cost of RPA is five which is shown in Fig. 3.2 The corresponding truth table of the circuit is shown in Table 2.1. It can be verified from the truth table that the input pattern corresponding to a particular output pattern can be uniquely determined. The proposed RPA is designed in such a way that it can be efficiently used as a reversible partial adder by setting the fourth input bit as a constant zero 0. Algorithm III describes the design of RPA as a reversible partial adder.

2.2.2. Comparison of Reversible Partial Adder with Others

In this subsection, we analyze the performance of the proposed circuit as a reversible partial adder and compare it with the existing circuits. Table III and Fig. 4 show the comparative result analysis among the proposed reversible partial adder and the existing partial adders. From this table, we can see that our design is much better than existing designs, especially in terms of quantum cost. As partial adder is basic units of a reversible carry look ahead adder circuit, this improvement has significant impact on this circuit.

Table 2.3 Comparison of partial adders obtained by different methods

Methods	Quantum cost	Garbage outputs
RPA	5	1
CLAA	9	3

2.2.3. Implementation of a Reversible Carry Look-Ahead Adder:

In the previous subsections, we propose one new reversible circuit to design the reversible carry look-ahead adder. In the following Subsections i and ii, we describe the design of a 4-bit and an n-bit reversible carry look-ahead adder, respectively.

i) Design of a 4-bit Reversible Carry Look-Ahead Adder Circuit

A 4-bit reversible carry look-ahead adder is constructed using proposed RPA circuits, Fredkin gates and CNOT gates.

ii) Design of an n-Bit Reversible Carry Look-Ahead Adder Circuit

The construction procedure of an n-bit reversible carry look-ahead adder circuit is as follows: Firstly, I generate n numbers of carry propagation (P) and carry generation (G) signals using n numbers of reversible partial adder circuits. Secondly, we produce carry-out signals using a Fredkin gate, where the inputs are the carry propagation (P) and carry generation (G) signals. Algorithm IV describes the whole process of addition. In Fig. 6, we show the design of an n-bit reversible carry look-ahead adder circuit.

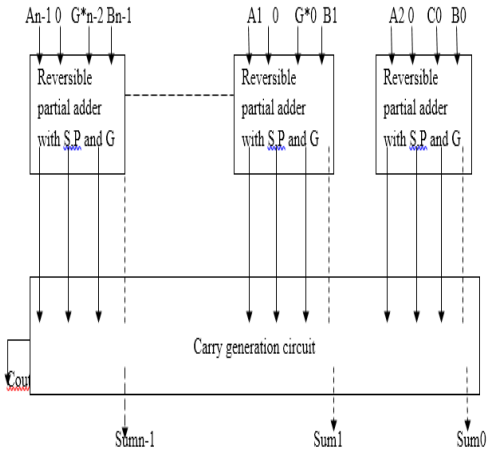


Fig 2.4 Block diagram of n-bit reversible carry look-ahead adder circuit

Algorithm IV: Algorithm for construction of an n-bit reversible carry look-ahead adder circuit:

Input: (x0, x1, x2, … , xn-1) , (y0 , y1, y2, … , yn-1)
Output: (s0, s1, s2, … ,sn-1) , (c1, c2, … ,cn-1)
1. Begin
2. For i:=0 to (n-1) do
3. Apply CNOT gate to make a copy of cj
4. Apply RPA circuit where
5. Input:= {xj, yi, cj, 0} and Output:= {xj, pi, sj, gi}
6. Apply Fredkin Gate where
7. Input:= {cj, pi, gj} and Output:= {cj, ci+1, G}
8. End Loop
9. End

III. SUB MODULES OF THE PROJECT

3.1 BASIC DEFINITIONS PERTAINING TO REVERSIBLE LOGIC

3.1.1 Reversible function:

The multiple output Boolean function F(x1; x2; :::; xn) of n Boolean variables is called reversible if:

a. The number of outputs is equal to the number of inputs;

b. Any output pattern has a unique pre-image. 418 In other words, reversible functions are those that perform permutations of the set of input vectors.

3.1.2 Reversible logic gates:

Reversible Gates are circuits in which number of outputs is equal to the number of inputs and there is a one to one correspondence between the vector of inputs and outputs[8- 10]. It not only helps us to determine the outputs from the inputs but also helps us to uniquely recover the inputs from the outputs.

3.1.3 Ancilla inputs/ constant inputs :

This refers to the number of inputs that are to be maintain constant at either 0 or 1 in order to synthesize the given logical function.

3.1.4 Garbage outputs:

Additional inputs or outputs can be added so as to make the number of inputs and outputs equal whenever necessary. This also refers to the number of outputs which are not used in the synthesis of a given function. In certain cases these become mandatory to achieve reversibility. Garbage is the number of outputs added to make an n-input k-output function ((n; k) function) reversible.

We use the words ―constant inputs‖ to denote the present value inputs that were added to an (n; k) function to make it reversible. The following simple formula shows the relation between the number of garbage outputs and constant inputs .

Input + constant input = output + garbage.

3.1.5 Quantum cost:

Quantum cost refers to the cost of the circuit in terms of the cost of a primitive gate. It is calculated knowing the number of primitive reversible logic gates (1*1 or 2*2) required to realize the circuit. The quantum cost of a circuit is the minimum number of 2*2 unitary gates to represent the circuit keeping the output unchanged. The quantum cost of a 1*1 gate is 0 and that of any 2*2 gate is the same, which is 1.

3.1.6 Flexibility:

Flexibility refers to the universality of a reversible logic gate in realizing more functions.

3.1.7 Gate Level:

This refers to the number of levels in the circuit which are required to realize the given logic functions.

3.1.8 Hardware Complexity:

This refers to the total number of logic operation in a circuit. Means the total number of AND, OR and EXOR operation

VI. ANCIENT VEDIC MATHEMATICAL ALGORITHMS

The Vedic mathematics mainly reduces the complex typical calculations in to simpler by applying sutras as stated above. These Vedic mathematic techniques are very efficient and take very less hardware to implement. These sutras are mainly used for multiplication of two decimal numbers and we extend these sutras for binary multiplications. Some of the techniques are discussed below.

A. Urdhva -Tiryagbhyam Sutra (Vertically and Crosswise):

Booth multipliers are generally used for multiplication purposes. Booth Encoder, Wallace Tree, Binary Adders and Partial Product Generator are the main components used for Booth multiplier architecture. Booth multiplier is mainly used for 2 applications are to increase the speed by reduction of the partial products and also by the way that the partial products to be added. In this section we propose a Vedic multiplication technique called "Urdhva-Tiryakbhyam – Vertically and crosswise."

B. Example for Decimal Multiplication Using Vedic Mathematics:

To illustrate this technique, let us consider two decimal numbers 252 and 846 and the multiplication of two decimal numbers 252×846 is explained by using the line diagram shown in below figure1. First multiply the both numbers present on the two sides of the line and then first digit is stored as the first digit of the result and remaining digit is stored as pre carry for the next coming step and the process goes on and when there is more than one line then calculate the product of end digits of first line and add the result to the product obtained from the other line and finally store it as a result and carry. The obtained carry can be used a carry for the further steps and finally we will get the required result which is the final product of two decimal numbers 252x846. Take the initial carry value as the zero. For clear understanding purpose we explained the complete algorithm in the below line diagram such that each bit represents a circle and number of bits equal to the number of circles present.

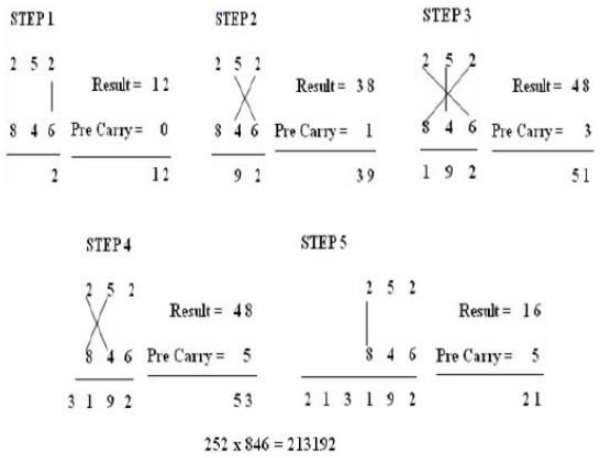


Figure 1. Multiplication of two decimal numbers

4.1 Modified Vedic Multiplier Architecture

The architectures for 2×2, 4×4, 8×8, 16×16. . .N×N bit modules are discussed in this section. In this section, the technique used is 'Urdhva-Tiryakbhyam' (Vertically and Crosswise) sutra which is a simple technique for multiplication with lesser number of steps and also in very less computational time. The main advantage of this Vedic multiplier is that we can calculate the partial products and summation to be done concurrently. Hence we are using this Vedic multiplier in almost all the ALU's.

A. 2×2 Vedic Multiplier Block

To explain this method let us consider 2 numbers with 2 bits each and the numbers are A and B where A=a0a1 and B=b0b1 as shown in the below line diagram. First the least significant bit (LSB) bit of final product (vertical) is obtained by taking the product of two least significant bit (LSB) bits of A and B is a0b0. Second step is to take the products in a crosswise manner such as the least significant bit (LSB) of the first number A (multiplicand) is multiplied with the next higher bit of the multiplicand B in a crosswise manner. The output generated is 1-Carry bit and 1bit used in the result as shown below. Next step is to take product of 2 most significant bits (MSB) and for the obtained result previously obtained carry should be added. The result obtained is used as the fourth bit of the final result and final carry is the other bit.

$s0=a0b0$ (1)
$c1s1=a1b0+a0b1$ (2)
$c2s2=c1+a1b1$ (3)

The obtained final result is given as c2s2s1s0. A 2×2 Vedic multiplier block is implemented by using two half adders and four two input and gates as shown in below Figure 2.

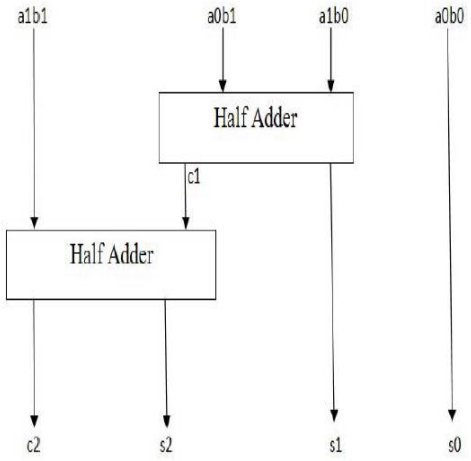


Figure 2. Block Diagram of 2×2 Vedic Multiplier

B. 4x4 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier. For explaining this multiplier let us consider two four bit numbers are A and B such that the individual bits can be represented as the A3A2A1A0 and B3B2B1B0. The procedure for multiplication can be explained in terms of

line diagram shown in below figure. The final output can be obtained as the C6S6S5S4S3S2S1S0. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure 3. After performing all the steps the result (Sn) and Carry(Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on.

$S0 = A0B0$ (4)

$C1S1 = A1B0 + A0B1$ (5)

$C2S2 = C1 + A0B2 + A2B0 + A1B1$ (6)

$C3S3 = C2 + A0B3 + A3B0 + A1B2 + A2B1$ (7)

$C4S4 = C3 + A1B3 + A3B1 + A2B2$ (8)

$C5S5 = C4 + A3B2 + A2B3$ (9)

$C6S6 = C5 + A3B3$ (10)

For clear understanding, observe the block diagrams for 4x4 as shown below figure 3 and within the block diagram 4x4 totally there are four 2x2 Vedic multiplier modules, and three ripple carry adders which are of four bit size are used. The four bit ripple carry adders are used for addition of two four bits and likewise totally four are use at intermediate stages 3 of multiplier. The carry generated from the first ripple carry adder is passed on to the next ripple carry adder and there are two zero inputs for second ripple carry adder. The arrangement of the ripple carry adders are shown in below block diagram which can reduces the computational time such that the delay can be decrease.

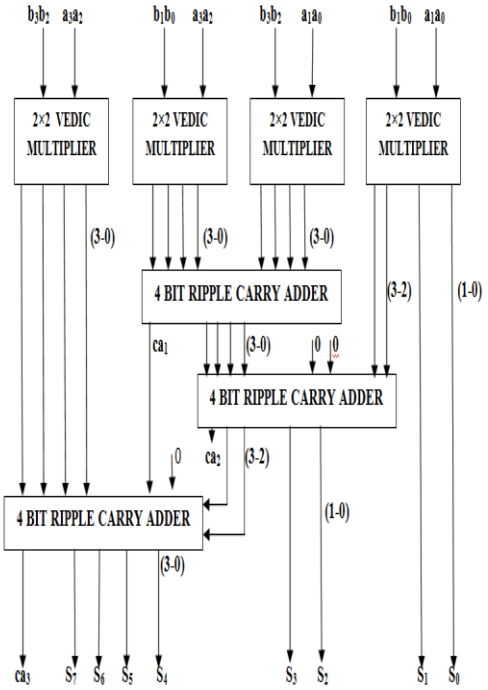


Figure 3. Block Diagram of 4x4 bit Vedic Multiplier

C. 8x8 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier. For explaining this multiplier let us consider two 8 bit numbers are A and B such that the individual bits can be represented as the A7A6A5A4A3A2A1A0 and B7B6B5B4B3B2B1B0. The procedure for multiplication can be explained in terms of line diagram shown in below figure 4. The final output can be obtained as the 16S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure 4. After performing all the steps the result (Sn) and Carry (Cn) is obtained and in the same way at each step the previous stage carry is forwarded to the next stage and the process goes on.

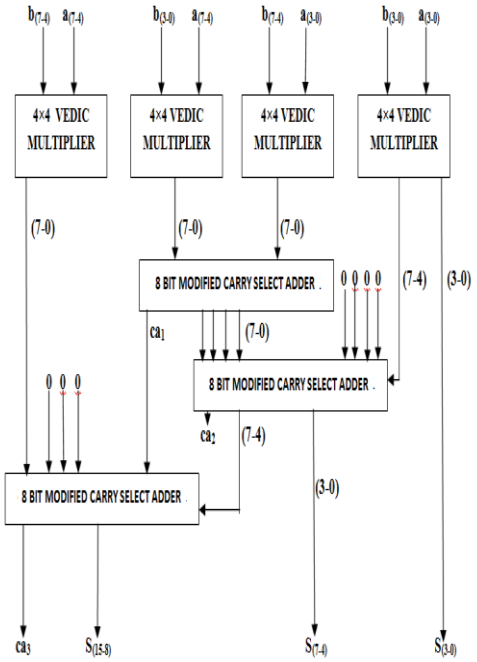


Figure 4. Block Diagram of 8x8 bit Vedic Multiplier

For clear understanding, observe the block diagrams for 8x8 as shown below and within the block diagram 8x8 totally there are four 4x4 Vedic multiplier modules, and three modified carry select adders which are of 8 bit size are used.

D. 16x16 Vedic Multiplier Block

In this section, now we will discuss about 4x4 bit Vedic multiplier[10]. For explaining this multiplier let us consider two 8 bit numbers are A and B such that the individual bits can be represented as the A [15:0] and B [15:0]. The procedure for multiplication can be explained in terms of line diagram shown in below figure 5. The final output can be obtained as the C16S[31:0]. The partial products are calculated in parallel and hence delay obtained is decreased enormously for the increase in the number of bits. The Least Significant Bit (LSB) S0 is obtained easily by multiplying the LSBs of the multiplier and the multiplicand. Here the multiplication is followed according to the steps shown in the line diagram in figure.

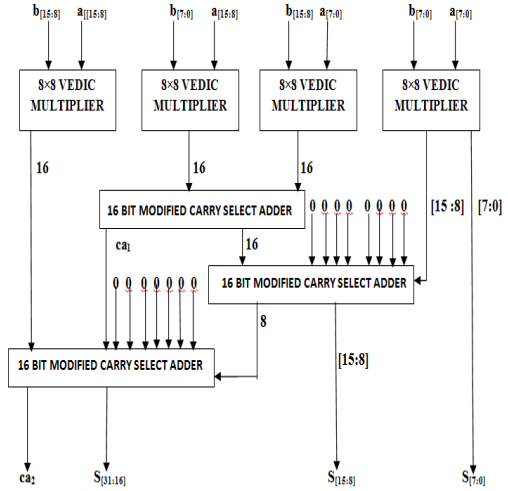


Figure 7. Block Diagram of 16x16 bit Vedic Multiplier

E. 32x32bit vedic multiplier

For any complex number multiplier design, the most important procedure is Multiplication. Here we have designed the 32 bit multiplier using vedic algorithm (Urdhvatiryakbyham sutra). Fig.1 shows the block diagram of 32x32bit vedic multiplier module which is easily designed using four 16x16bit Vedic multipliers modules. Ripple Carry Adders(RCA) are used for the addition of the output of these four 16x16bit multiplier modules.

Suppose the two 32bit numbers are X [31:0] and Y [31:0].

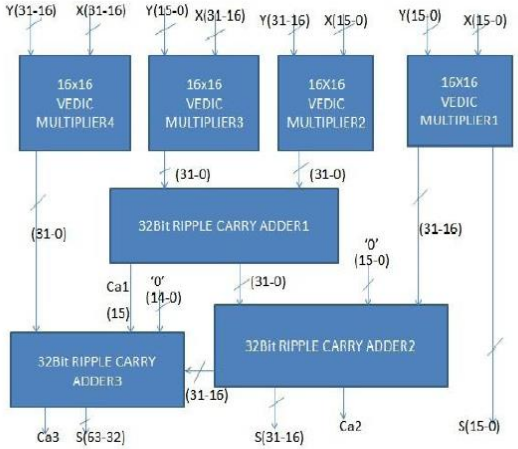


Fig. 8. Block Diagram of 32x32bit vedic multiplier

Each of these numbers are divided into two 16-bit numbers X [31:16]-X[15:0](XH-XL) and Y[31:16]-Y[15:0](YHYL) and given as a input to the 16x16bit vedic multiplier module. The input combinations for 1 to 4 16x16bit multiplier modules are XL-YL, XL-YH, XH-YL and XH-YH respectively. Each multiplier gives the intermediate output of 32-bit. This intermediate output is then added using ripple carry adders (RCA). The output of second and third multiplier module is added using RCA1. The output of RCA1 (32-bit) and the higher order bits of first multiplier is then added using the RCA2 which gives the output S[31:16]. Finally the higher order bits of RCA2, the output of fourth multiplier and the carry from the RCA1 (the fifteenth bit position) are added to get the higher order bits S[63:32] of the final output of 32x32bit multiplier. The LSB bits S[15:0] of final output are obtained directly by taking the lower order bits of the output of first 16x16bit multiplier.

V. RESULT

The corresponding schematics of the adders after synthesis is shown below.

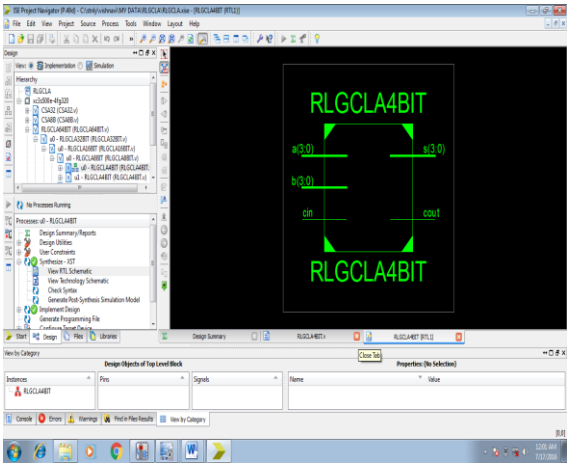


Figure 7.13: RTL schematic of PROPOSED REVERSIBLE CLA4BIT

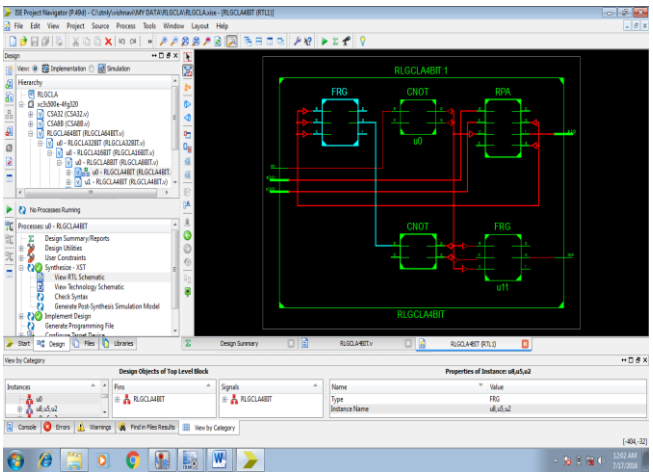


Figure 7.14: RTL schematic of PROPOSED REVERSIBLE CLA4BIT

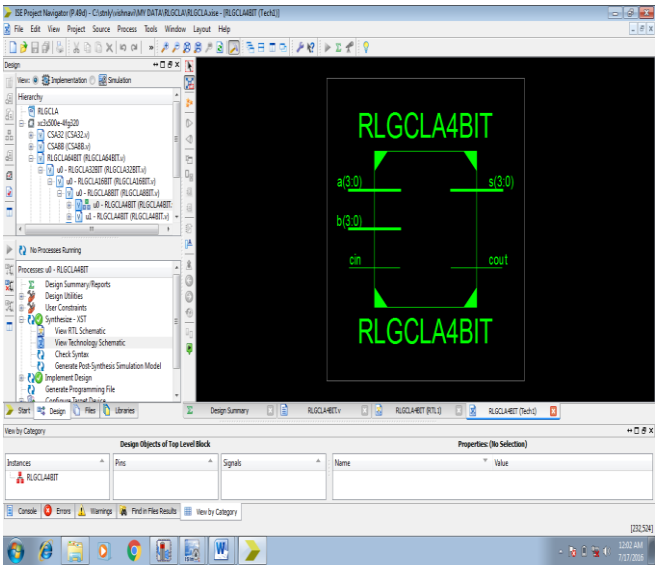


Figure 7.15: Technology schematic of PROPOSED REVERSIBLE CLA4BIT

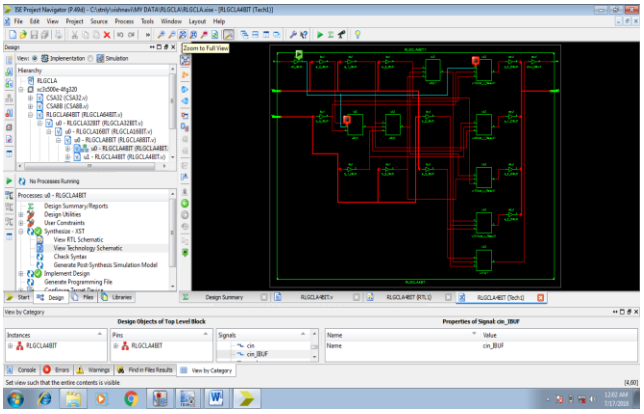


Figure 7.16: Technology schematic of PROPOSED REVERSIBLE CLA4BIT

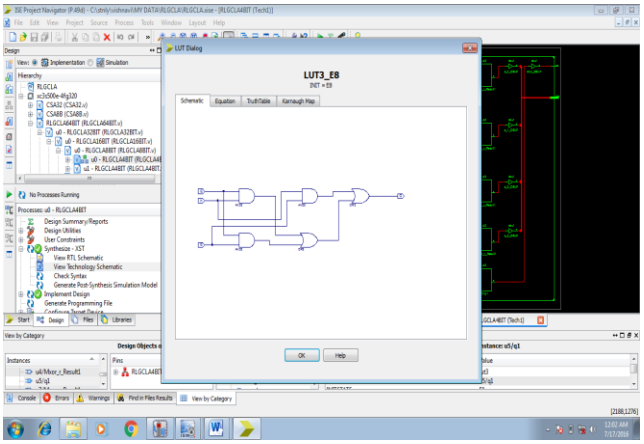
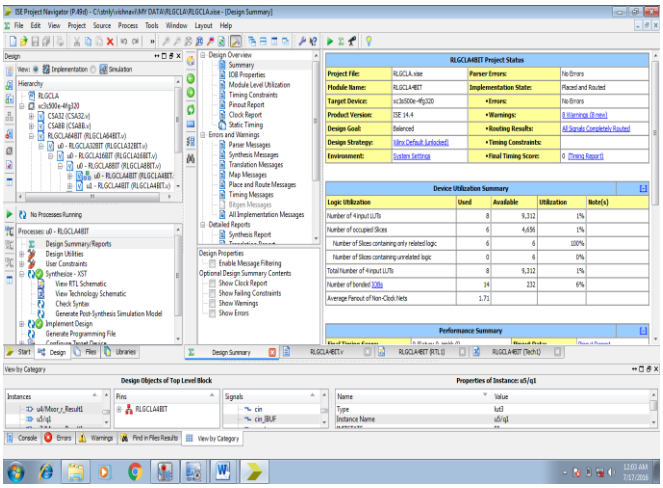


Figure 7.17: Internal block PROPOSED REVERSIBLE CLA4BIT

7.4 Synthesis Report

Table 7-1: Synthesis report of PROPOSED REVERSIBLE CLA4BIT



SIMULATION RESULTS

The corresponding simulation results of the adders are shown below.

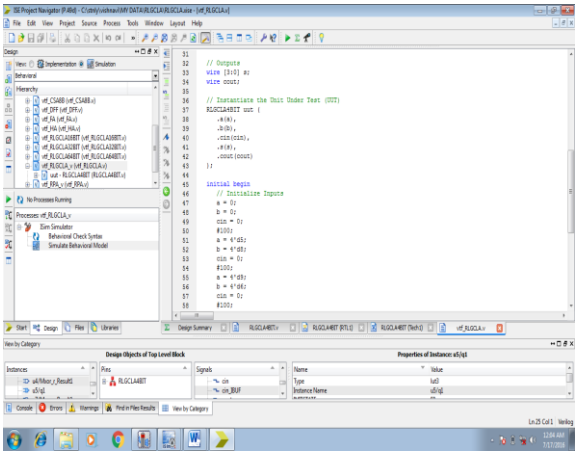


Figure 8-1: Test Bench for PROPOSED REVERSIBLE CLA4BIT

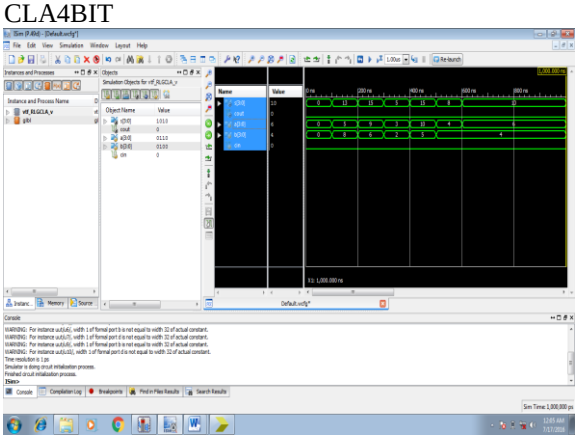


Figure 8-2: Simulated output for PROPOSED REVERSIBLE CLA4BIT

CONCLUSION AND FUTURE SCOPE

This paper presented a novel design methodology of a reversible nbit carry look-ahead adder (RCLA) circuit using dynamic programming, where n is the number of bits. An efficient algorithm was proposed to design a compact low power reversible carry lookahead adder. The carry look-ahead adder was constructed in two steps: At first, a reversible partial adder was developed to produce the carry propagation, carry generation and summation signals of the inputs. Secondly, a reversible circuit for generating the carry output of RCLA was used. We also found that the proposed reversible carry look-ahead adder circuit is much faster than the existing ones [13-16]. In addition, we show that the proposed reversible carry look-ahead adder is constructed with the optimum number of reversible gates, quantum cost, garbage outputs, quantum gate calculation complexity and delay using Xilinx ISE14.4. In this paper, we present an explicit scheme for executing elementary arithmetic operation. As adders are the basic and one of the most important components of a reversible arithmetic unit, we believe that the implementation of our design can certainly improve the currently available reversible systems [3], [4], [19].

VI. FUTURE WORK

Learning and designing a compact reversible carry skip adder using reversible logic gates and using dynamic programming. To get efficient outcome for addition we are using new technique Reversible partial adder for carry skip

adder. We are extending this work for designing of Vedic multiplier. Multiplication is one of the fundamental block in almost all the arithmetic logic units. This Vedic multiplication is mainly used in the fields of the Digital Signal Processing (DSP) and also in so many applications like Fast Fourier Transform, convolution, filtering and microprocessor applications. In most of the DSP algorithms multiplier is one of the key component and hence a high speed and area efficient multiplier is needed and multiplication time is also one of the predominant factor for DSP algorithms. The ancient mathematical techniques like Vedic mathematics used to reduce the computational time such that it can increases speed and also requires less hardware.

REFERENCES

[1] C. H. Bennett and G. Brassard. Quantum cryptography: Public-key distribution and coin tossing. In Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing, pp. 175-179, Bangalore, India, 1984. IEEE Press.

[2] A. Dixit, V. Kapse, Arithmetic & logic unit (ALU) design using reversible control unit, International Journal of Engineering and Innovative Technology (IJEIT) 1(June (6)) (2012).

[3] H.V.R. Aradhya, B.V.P. Kumar, K.N. Muralidhara, Design of control unit for low power ALU using reversible logic, International Journal of Scientific & Engineering Research 2 (September (9)) (2011).

[4] A. Peres, "Reversible logic and quantum computers," Phys. Rev. A, vol. 32, pp. 3266–3276, Dec 1985.

[5] E. Fredkin, T. Toffoli, "Conservative logic," International Journal of Theoretical Physics, vol. 21, no. 3–4, pp. 219–253, 1982.

[6] M. Nielsen and I. Chuang, "Quantum Computation and Quantum Information," Cambridge Univ. Press, 2000.