ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

An optimized Design of Reversible Carry Look-Ahead Adder Using RPA

P.Anil Kumar
Assistant Professor,
Dept of ECE,
Malla Reddy College of Engineering.

Navanandula Praveen Kumar
Student,
Dept of ECE,
Malla Reddy College of Engineering.

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 Xilinx ISE14.4, 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 Computing; Carry Look-Ahead Adder.

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.

A general frame-work for analyzing the problems is provided by dynamic programming. Within this frame-work, a variety of optimization techniques can be employed to solve particular aspects of a more general formulation [2]. The adder circuit is one of the most significant components of a reversible processor that determines its throughput, as it is used in the reversible ALU, the floating-point unit, and address generation for cache or memory accesses [3-4]. In this paper, we present a new vision into the addition process. The proposed method of addition is based on a dynamic programming and we construct a compact and low power reversible n-bit carry look-ahead adder circuit using the proposed method.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

II. BASIC DEFINITIONS AND PROPERTIES OF DIFFERENT REVERSIBLE GATES

In this section, we have presented the basic definitions of reversible logic and an overview on few reversible gates which are relevant with this research work.

A. Reversible Gate:

In a reversible gate, the number of inputs is equal to the number of outputs and there is a unique mapping between the input and output vectors [5]. This phenomenon ensures the recovery of lost bits. There are various kinds of reversible gates like Feynman gate [6], Peres gate [7], Fredkin gate [8] etc. to perform different kinds of operations. Fig. 1 shows the block diagram of Peres Gate and its input and output mapping.

B. Garbage Output:

In a reversible gate, the number of outputs must be equal to the number of inputs. As a consequence, there are usually some outputs which are not further required and remains unused called garbage outputs [3-5]. For example, if we want to perform an Ex-OR operation (ab) using a Peres gate (shown in Fig. 1), the gate produces p and r as two garbage outputs.

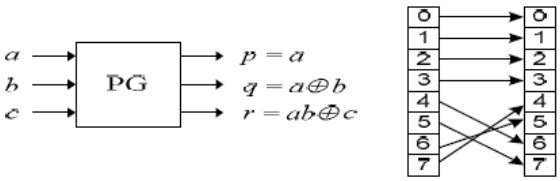

Fig. 1 33 Peres Gate and Its Corresponding Input Output Mapping.

C. Quantum Cost:

The quantum cost is the total number of quantum gates in a given reversible circuit, where the reversible gates are substituting by a cascade of basic quantum gates [9]. The most used basic quantum gates are the NOT gate (a single quantum bit is inverted), the controlled-NOT (CNOT) gate (the target quantum bit is inverted if the single control quantum bit is 1), the controlled-V gate (also known as a square root of NOT, since two consecutive V operations are equivalent to an inversion), and the controlled-V+ gate (which performs the inverse operation of the V gate and thus, it is also a square root of NOT gate) [9].The quantum cost of Fredkin gate in Fig. 2(b) is five. In this figure, V is a square root of NOT gate and V+ is its hermitian.

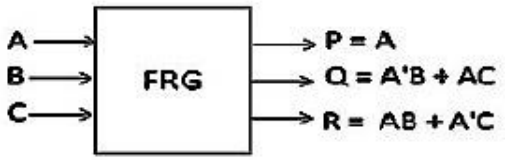

Fig. 2(a) Block Diagram of Fredkin Gate.

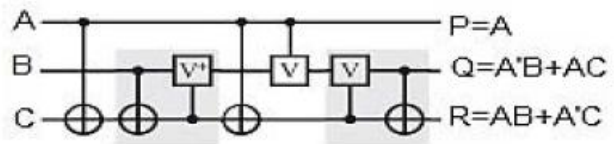

Fig. 2(b) Quantum Representation of Fredkin Gate

D. Delay:

In a logic circuit, the delay can be defined by the maximum number of reversible gates in a path from the input line to the output line. This definition is defined by the following two suppositions [5]: Firstly, the computation time of each gate requires unit time which means all gates in the given circuit will take the same amount of time for internal logic operations. Secondly, before the starting of the computation, all inputs of the circuit are known. The computation time for each gate is obtained using DSCH 3.5 [10] and delay is calculated as the sum of delays of each gate through the path that causes maximum delay where the internal structure and each operation of the gate are known before starting the calculation. As semiconductor technology is marching towards lower sized chips, the interconnect delay is playing an important role [11] [12]. The interconnect delay (tid) can be calculated as:

$$t_{id} = (3.56 \times K \times L^2 \times \rho \times \epsilon)/(\lambda^2 \times n)$$

Here K = Die-electric constant of metal, L = Maximum Interconnect Wire length, = Wire Resistivity, =Permitivity, = Least thickness value, n =

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

no. of lines broke for calculation for long lines. In this paper, we compute the delay as the sum of delays of all gates as well as the interconnection delays of the circuit. E. Hardware Complexity of Reversible Gates Hardware Complexity of a circuit is the total logical operation performed by the circuit. Total logical calculation is the count of the Ex-OR (), AND () and NOT (d) in the output expressions [5]. For example, a Peres gate requires two Ex-OR and one AND operations. So, the total hardware complexity or logical calculation of a Peres gate is 2 + (shown in Fig. 1).

III. BACKGROUND STUDY OF PREVIOUS DESIGNS

Draper et al. proposed a logarithmic depth quantum carry look ahead adder [13] in 2006, which demonstrates two versions of addition namely in place and out-of-place method of addition for n-bit numbers, but the design [13] is not optimized in terms of gates, garbage outputs, quantum cost and delay. Thapliyal et al. proposed a novel methodology for reversible carry look-ahead adder [14] in 2013. In this work [14], they presented improved designs of both in-place and out-of-place designs of reversible carry look-ahead adder proposed in [13]. They utilized the properties of the reversible Peres gate and the TR gate to optimize the design [14].

Though the paper considered a reversible design of carry look-ahead adder, it was actually the representation of carry lookahead adder using quantum gates. As the existing design [14] had a huge number of gates and complex circuit structure, it affected severely on the adder circuit in terms of number of gates, quantum cost, garbages and delay. The theoretical explanation and algorithm of the design [14] are also absent in this work. Tiwari et al. proposed an optimized carry look-ahead BCD adder using reversible logic [15] in 2011. Here, the proposed circuit can add only two 4-bit binary variables. This design [15] is not optimized in terms of gates, garbage outputs, quantum cost and delay.

Chakrabarti et al. proposed a design of quantum adder circuits and evaluating their error performance [16] in 2008. In this paper, they proposed the design of quantum adder circuits using CNOT and Ck NOT gates only which had a huge effect on quantum cost and delay, but the number of garbages was reduced due to an efficient algorithm for reduction of garbages. Moreover, the circuit [15-16] is not generalized and the design procedure of the circuit is also not presented in the paper.

IV. PROPOSED METHOD FOR CARRY GENERATION OF CARRY LOOK-AHEAD ADDERS

A problem f into a set of other problems can be decomposed using dynamic programming [17], 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 proposed 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

$$C_{out} = C_{i+1} = A_i . B_i . \overline{C_i} + (A_i + B_i) . C_i \quad \ldots \quad (1)$$

The above expression can also be represented as:

$$C_{i+1} = P_i G^*_{i-1} + G_i . \overline{G^*_{i-1}} \quad \ldots \quad (2)$$

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

$$C_{i+1} = \begin{cases} C_{i+1} & ; i = 0 \\ P_i G^*_{i-1} + G_i . \overline{G^*_{i-1}} & ; 1 \le i \le n-1 \text{ and } G^*_{i-1} = P_{i-1} G^*_{i-2} + G_{i-1} . \overline{G^*_{i-2}} \\ C_{out} & ; i = n \end{cases}$$

Applying this to a 4-bit adder, we have:

$C_0 = 0$
$C_1 = P_0 C_0 + G_0 \overline{C_0} = P_0 .0 + G_0 .1 = G_0 = G^*_0$
$C_2 = P_1 G^*_0 + G_1 \overline{G^*_0} = G^*_1$
$C_3 = P_2 G^*_1 + G_2 \overline{G^*_1} = G^*_2$
$C_4 = P_3 G^*_2 + G_3 \overline{G^*_2} = G^*_3$

The sum signal can be calculated as follows:

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$$

In Table I, 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 proposed carry generation method is much less than the existing carry generation method. 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 I. Comparison among the Proposed and the Existing Carry Generation Techniques in terms of Number of Operations.

No. of Bits	Existing* [13-16]	Proposed
4	30	24
8	60	48
16	120	96
32	240	192

*As all existing methods consider the same technique, the number of operations is the same.

Example 1 :The Carry generation techniques with a carry output 0 for the proposed method and the existing method are described using the following example: By adding 1101 and 0010, we get 1111. We generate carry for these numbers using the existing and proposed methods:

Here, A3 A2 A1 A0 and B3 B2 B1 B0
1 1 0 1 0 0 1 0
Now, P3 = A3 + B3= 1+0 = 1 ; G3 = A3 . B3 = 1.0 = 0
P2 = A2 + B2= 1+0 = 1 ; G2 = A2 . B2 = 1.0 = 0
P1 = A1 + B1= 0+1 = 1 ; G1 = A1 . B1 = 0.1 = 0
P0 = A0 + B0= 1+0 = 1 ; G0 = A0 . B0 = 1.0 = 0
In the existing methods [14-15], the carry is generated as follows:
C0 = 0
C1= G0+P0 .C0 = 0 + 1.0 = 0
C2= G1+ P1.C1= G1+ P1.G0+P1.P0.C0 = 0+1.0+1.1.0 = 0
C3= G2+P2.G1+P2.P1.G0+P2.P1.P0.C0
= 0+1.0+1.1.0+1.1.1.0 = 0
C4 = G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.C0
= 0+1.0+1.1.0+1.1.1.0+1.1.1.1.0 = 0

In the proposed method, the carry is generated as follows:

$C_0 = 0$
$C_1 = P_0 .C_0 + G_0 . \overline{C_0} = G^*_0 = 1.0 + 0.1 = 0$
$C_2 = P_1 .G^*_0 + G_1 . \overline{G^*_0} = G^*_1 = 1.0+0.1 = 0$
$C_3 = P_2 .G^*_1 + G_2 . \overline{G^*_1} = G^*_2 = 1.0+0.1 = 0$
$C_4 = P_3 .G^*_2 + G_3 . \overline{G^*_2} = G^*_3 = 1.0+0.1 = 0$

Here, we can see that, both methods generate the same carries, i.e., 0. Thus, the proposed method generates the accurate carry output with the minimum number of operations as it requires 24 operations while the existing method requires 30 operations.

Example 2 The Carry generation techniques with a carry output "1" for the proposed method and the existing method are described using the following example:

By adding1101 and 0011, we get 10000. We generate carry for these numbers using the existing and proposed methods:

Here, A3 A2 A1 A0 and B3 B2 B1 B0
1 1 0 1 0 0 1 1

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Now, P3 = A3 + B3 = 1+0 = 1 ; G3 = A3 . B3 = 1.0 = 0
P2 = A2 + B2= 1+0 = 1 ; G2 = A2 . B2 = 1.0 = 0
P1 = A1 + B1 = 0+1 = 1 ; G1 = A1 . B1 = 0.1 = 0
P0 = A0 + B0= 1+1 = 1 ; G0 = A0 . B0 = 1.1 = 1

In the existing method [14-15], the carry is generated as follows:
C0 = 0
C1 = G0+P0 .C0 = 1 + 1.0 = 1
C2 = G1+ P1.C1= G1+ P1.G0+P1.P0.C0 = 0+1.1+1.1.0 = 1
C3 = G2+P2.G1+P2.P1.G0+P2.P1.P0.C0
= 0+1.0+1.1.1+1.1.1.0 = 1
C4 = G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.C0
= 0+1.0+1.1.0+1.1.1.1+ 1.1.1.1.0 = 1
In the proposed method, the carry is generated as follows:

$C_0 = 0$
$C_1 = P_0.C_0 + G_0.\overline{C_0} = G^*_0 = 1.0 + 1.1 = 1$
$C_2 = P_1.G^*_0 + G_1.\overline{G^*_0} = G^*_1 = 1.1 + 0.0 = 1$
$C_3 = P_2.G^*_1 + G_2.\overline{G^*_1} = G^*_2 = 1.1 + 0.0 = 1$
$C_4 = P_3.G^*_2 + G_3.\overline{G^*_2} = G^*_3 = 1.1 + 0.0 = 1$

Here, we can see that, both methods generate the same carries, i.e., 1. Thus, the proposed method generates the accurate carry output with the minimum number of operations as it requires 24 operations while the existing method requires 30 operations.

Algorithm I: Proposed 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, F_i
4. Set, $F_i.a = A_i$; $F_i.b = B_i$; $F_i.c = 0$; $F_i.d = 0$
5. Output $Sum_i = F_i.r$
6. **End For**
7. **For** $i = 0$ to $(n-1)$ **do**
8. $P_i = A_i \oplus B_i$
9. $G_i = A_i.B_i$
10. **End For**
11. Set *Carry* = Generate_Carry(C_0, P_0, G_0, P_1, G_1, …, P_{n-1}, G_{n-1}) [Algorithm II]
12. **End**

Algorithm II: Carry Generation Algorithm for Generate_Carry( ) Procedure

Input: $C_0, P (P_0, P_1, …, P_{n-1}), G (G_0, G_1, …, G_{n-1})$
Output: ***Carry (1-bit)***
1. **Begin**
2. **For** $i = 1$ to n do
3. $C_i = P_{i-1}.G^*_{i-2} + G_{i-1}.\overline{G^*_{i-2}}$ where, $G^*_{i-2} = P_{i-2}.G^*_{i-3} + G_{i-2}.\overline{G^*_{i-3}}$
4. **End For**
5. Set, *Carry* = C_n
6. **End**

This method of addition can be used for both classical irreversible) and reversible adder circuits. In the proposed method, there are only O(log2 n) layers and each layer has O(1) transmission delay. So, the time complexity of the proposed method is O(log2n). Now, we design a reversible carry look-ahead adder circuit using the proposed method which is shown in Section V.

V. PROPOSED DESIGN OF A COMPACT REVERSIBLE CARRY LOOK-AHEAD ADDER

In this section, we propose a new method for designing a reversible n-bit carry look-ahead adder circuit. To implement our proposed addition algorithm, carry generation circuitis needed. At first, we propose a new reversible partial adder named RPA to produce carry propagation (P) signal, carry generation (G) signal and sum (S) and then, we present a reversible 4-bit carry look-ahead adder circuit. Finally, we 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. A. Proposed Reversible Partial Adder In this subsection, a new 44 reversible partial adder, namely RPA, is proposed. The input vector, Ivand the output vector, Ovof theproposed circuit are as follows:

$$I_v = \{A, B, C, D\}; \text{ and}$$
$$O_v = \{P = A, Q = A \oplus B, R = A \oplus B \oplus C, S = AB \oplus D\}$$

Fig. 3(a) shows the diagram of the proposed 44 RPA. The quantum cost of RPA is five which is shown in Fig. 3(b). The corresponding truth table of the circuit is shown in Table II.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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.

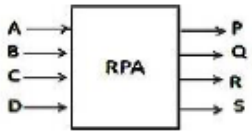

Fig. 3 (a) Block Diagram of Reversible Partial Adder, when D = 0.

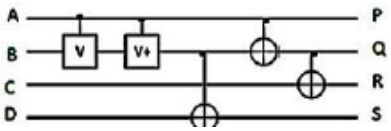

Fig. 3(b) Quantum Realization of Reversible Partial Adder, when D = 0.

Table II. Reversibility of the 44 Reversible Partial Adder Circuit.

Inputs				Outputs			
A	B	C	D	P	Q	R	S
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	0	0
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

Algorithm III: Proposed Algorithm for Construction of a Reversible Partial Adder

Input: *A, B, C_{in}*
Output: *Sum(S), Carry Propagation(P), Carry Generation(G)*
1. Begin
2. Take an RPA circuit, (a, b, c, d)⇔(p, q, r, s)
   a. Set, RPA.a = *A*; RPA.b = *B*; RPA.c = C_{in}; RPA.d = 0;
   b. Output *S* = RPA.r
   c. Output *P* = RPA.q
   d. Output *G* = RPA.s
3. End

B. Comparison of Proposed 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 [14] and [15]. 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 [18].

Table III. Comparison of Partial Adders Obtained by Different Methods.

Methods	Quantum Cost	Garbage Outputs
Proposed	5	1
Existing [14]*	9	3
Existing [15]*	15	2

* Added the additional quantum cost and garbages for generating carry propagation and carry generation outputs.

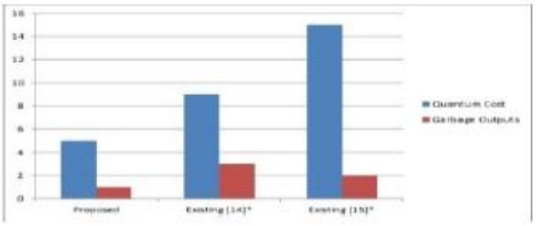

Fig. 4.Comparison among the Proposed and Existing Partial Adders with respect to Quantum Cost and Garbage Outputs.

C. 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. At first, we generate the carry generation (G) signals, carry propagation (P) signals and sum (S) using proposed RPA circuits.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Then, these signals are sent to another reversible circuit named Fredkin gate (FRG) to produce the carry output signals. Fig. 5 shows the design of the proposed 4-bit reversible carry look-ahead adder circuit.

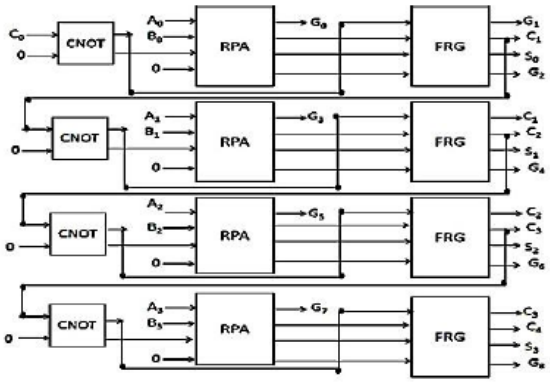

Fig. 5 A 4-Bit Reversible Carry Look-Ahead Adder Circuit.

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, we 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. Here, the Algorithm IV is based on Algorithm I and Algorithm II.

Algorithm IV: Algorithm for Construction of an *n-Bit* Reversible Carry Look-Ahead Adder Circuit

Input: $(x_0, x_1, x_2, \ldots, x_{n-1}), (y_0, y_1, y_2, \ldots, y_{n-1})$
Output: $(s_0, s_1, s_2, \ldots, s_{n-1}), (c_1, c_2, \ldots, c_n)$

1. Begin
2. For i:=0 to (n-1) do
3. Apply CNOT gate to make a copy of c_i
4. Apply RPA circuit where
5. Input:= $\{x_i, y_i, c_i, 0\}$ and Output:= $\{x_i, p_i, s_i, g_i\}$
6. Apply Fredkin Gate where
7. Input:= $\{c_i, p_i, g_i\}$ and Output:= $\{c_i, c_{i+1}, G\}$
8. End Loop
9. End

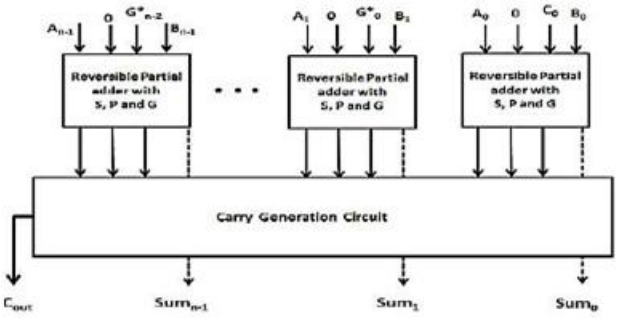

Fig. 6 Block Diagram of an n-Bit Reversible Carry Look-Ahead Adder Circuit.

D. Complexity of the Proposed n-Bit Reversible Carry LookAhead Adder

The complexities of a reversible circuit in terms of reversible gates, quantum cost, garbage outputs and delay etc. are very important as they have a great impact during the physical implementation [5]. In this section, we present the different complexities of an n-bit Reversible Carry Look-Ahead Adder (RCLA) Circuit.

Lemma 1: An n-bit reversible carry look-ahead adder (RCLA) circuit requires 3n gates, where n is the number of bits.

Proof: We prove the above statement by method of contradiction. Suppose, a reversible n-bit carry look-ahead adder (RCLA) circuit doesn't require 3n gates, where n is the number of bits. In proposed RCLA circuit, n numbers of carry propagation, carry generation and summation signals are generated. To generate these signals, one reversible partial adder (RPA) circuit is needed.The number of gates for these signalscan be modeled using RPA circuit (GRPA) as below:

GRPA = n . . . (1)
Fredkin gates (FRG) are required to generate carry signals. For each carrysignals, one RPA circuit is needed. The number of gates to generate carrysignals (GFRG) is as below:
GFRG = n . . . (2)

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

In addition, n numbers of CNOT gates (GCNOT) are required for copyingoperation. So, the total number of gates of the RCLA circuit (GRCLA) can be modeled asfollows:

GRCLA = GRPA + GFRG + GCNOT = n + n + n = 3n
This contradicts the supposition that a reversible n-bit carry look-ahead adder (RCLA) circuit doesn't require 3n gates. Hence, the supposition is false and the Lemma 1 is true and this completes the proof.
Lemma 2: An n-bit reversible carry look-ahead adder (RCLA) circuit requires 11n quantum cost, where n is the number of bits.

Proof: We prove the above statement by method of contradiction. Suppose, a reversible n-bit carry look-ahead adder (RCLA) circuit doesn't require 11n quantum cost, where n is the number of bits. In proposed RCLA circuit, n numbers of carry propagation, carry generation and summation signals are generated. To generate these signals, one reversible partial adder (RPA) circuit is needed.

The quantum cost for these signals canbe modeled using RPA circuit (QCRPA) as below:
QCRPA = 5_n = 5n . . . (3)
Fredkin gates (FRG) are required to generate carry signals. For each carrysignals, one RPA circuit is needed. The quantum cost to generate carry signals(QCFRG) is as below:
QCFRG =5_n = 5n . . . (4)
In addition, n numbers of CNOT gates (QCCNOT) are required for copyingoperation.
So, the total quantum cost of the RCLA circuit (QCRCLA) can be modeled asfollows:
QCRCLA = QCRPA + QCFRG + QCCNOT = 5n + 5n + n = 11n

This contradicts the supposition that a reversible n-bit carry look-ahead adder (RCLA) circuit doesn't require 11n quantum cost. Hence, the supposition is false and the Lemma 2 is true and this completes the proof.

Example 3: When n = 4, the total quantum cost of a 4-bit RCLA circuit is (114) = 44. Lemma 3: A n-bit reversible carry look-ahead adder (RCLA) circuit requires (7n+5n +2nd) logical calculation or hardware complexity, where n is the number of bits, is the Ex-OR gate calculation complexity, is the AND gate calculation complexity, d is the NOT gate calculation complexity.

Proof: We prove the above statement by method of contradiction. Suppose, a reversible n-bit reversible carry look-ahead adder doesn't require (7n+5n +2nd) logical calculation or hardware complexity, where n is the number of bits, is the Ex-OR gate calculation complexity, is the AND gate calculation complexity, d is the NOT gate calculation complexity. The hardware complexities of RPA, FRG and CNOT gates are (4+),(2+4 +2d) and , respectively. According to Lemma 5.1, the hardware complexity of reversible carry look-ahead adder (HRCLA) can be modeled as below:

HRCLA = (4_+_) n + (2_+4_+2d) n + n_ = 7n_+5n_+2nd
Therefore, the total hardware complexity of the n-bit reversible carry lookaheadadder can be calculated as below:
HM = 7n_+5n_+2nd
This contradicts the supposition that a reversible n-bit carry look-ahead adder doesn't require 7n+5n +2nd logical calculation or hardware complexity. Hence, the supposition is false and the Lemma 3 is true and this completes the proof.

Example 4: When n = 4, the total hardware complexity of a 4-bit RCLA circuit is (74)σ+(54)Ω+(24)d = 28σ +20Ω +8d.

Lemma 4: An n-bit reversible carry look-ahead adder (RCLA) circuit generates 2n+1 garbage outputs, where n is the number of bits.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Proof: We prove the above statement by mathematical induction. As the proposed reversible partial adder (RPA) circuit is used for carry propagation carry generation and summation signals, thus, it produces only one garbage output. The Fredkin gate is used for generating carry signals, so, it produces only one garbage outputs except the first Fredkin gate which produce two garbage outputs. The CNOT gate doesn't produce any garbage output. So, the total number of garbage outputs generated by a 2-bit RCLA circuit (GRCLA2) is, GRCLA2 = GRPA2 + GFRG2 + GCNOT2 = 2+3+0 = 5.

Thus, the statement holds for base case n=2. Assume that, the statement holds for n=m. So, an m-bit RCLA can be realized with 2m+1 garbage outputs, where m is the number of bits. A (m+1)-bit RCLA circuit produces (m+1) garbage outputs from RPA circuits and ((m+1)+1) garbage outputs from Fredkin gates . As a result, the total garbage outputs produces by a (m+1)-bit RCLA circuit is 2(m+1)+1, where m+1 is the number of bits. Thus, the statement holds for n=m+1. Therefore, an n-bit RCLA circuit generates 2n+1garbage outputs.

Example 5: When n = 4, the total number of garbage output of a 4-bit RCLA circuit is (24 +1) = 9. Lemma 5: An n-bit reversible carry look-ahead adder (RCLA) circuit requires 0.48n ns gate level delay, where n is the number of bits and ns is the unit of measuring delay.

Proof: We prove the above statement by method of contradiction. Suppose, a reversible n-bit carry look-ahead adder doesn't require 0.48n ns delay, where n is the number of bits and ns is the unit of measuring delay. From Lemma 1, we find that an n-bit RCLA circuit consists of n number of reversible partial adders (RPA), n number of Fredkin gates and n number of CNOT gates.
DRCLA = DRPA + DFRG + DCNOT
= (0.15n+0.23n+0.1n) ns = 0.48n ns

This contradicts the supposition that a reversible n-bit carry look-ahead adder doesn't require 0.48n ns delay. Hence, the supposition is false and the Lemma 5 is true and this completes the proof.

E. Comparative Study:

The comparative study shows that the proposed carry look-ahead adder requires less number of reversible gates, quantum cost, garbage outputs and delay when the number of bits of addition is increased. This means that the proposed adder is scalable. Tables IV-VI show number of reversible gates, quantum cost and garbage outputs required for 4, 8, 16, 32, 64 and 128 bits reversible carry look-ahead adders. Table VII depicts the above mentioned parameters and hardware complexity of the reversible carry look-ahead adder in terms of n, where n is the number of bits of addition. The performance analysis between the proposed and the existing designs [13-16] of reversible carry look-ahead adder circuit is shown in Table VIII. However, in Table VI the method in [16] performs a little bit better in terms of number of garbages than the proposed and the existing methods [13-15], where the garbages are calculated as per Definition II(B).

Table IV. Comparison between Proposed and Existing Reversible Carry Look-AheadAdder in terms of Reversible Gates.

No. of Bits	Existing [13]	Existing [14]	Existing [15]	Existing [16]	Proposed
4	29	25	32	22	12
8	85	71	122	42	24
16	203	169	242	82	48
32	451	385	482	162	96
64	893	761	962	322	192
128	1969	1711	1922	642	384

Table V. Comparison between Proposed and Existing Reversible Carry Look-AheadAdder in terms of Quantum Cost.

No. of Bits	Existing [13]	Existing [14]	Existing [15]	Existing [16]	Proposed
4	85	72	100	998	44
8	261	231	200	4146	88
16	609	600	400	10442	176
32	1515	1463	800	23034	352
64	3258	3009	1600	48218	704
128	6801	6290	3200	98586	1408

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Table VI. Comparison between Proposed and Existing Reversible Carry Look-AheadAdder in terms of Garbage Outputs.

No. of Bits	Existing [13]	Existing [14]	Existing [15]	Existing [16]	Proposed
4	10	10	18	8	9
8	20	20	36	16	18
16	40	40	72	32	36
32	80	80	144	64	72
64	160	160	288	128	144
128	320	320	576	256	288

Table VII. Comparative Results of Different Reversible n-bit Carry Look-Ahead Adder Circuits.

	Existing [13]	Existing [14]	Existing [15]	Existing [16]	Proposed
Number of Gates	16n-3n(n)-3n(n-1)-3log n-3log(n-1)-14	8n-3n(n)-3logn-4	(15n+4)/2	5n+2	3n
Quantum Cost	56n-15n(n)-15n(n-1)-15log n-15 log(n-1)-42	52n-15n(n)-15n(n-1)-15logn-15log(n-1)	31n-24	787n-2150	11n
Garbage Outputs	(5n)/2	(8n)/2	(9n)/2	2n	2n+1
Delay	2.1n-0.45n(n)-0.45n(n-1)-0.45log n-0.45log(n-1)-1.75	0.95n-0.45n(n)-0.45logn-0.35	1.825n+0.2	0.8n+0.6	0.48n
Hardware Complexity	(4n-3n(n)-3logn)(α+β)+(n-1)2α+β)+(n-1)α+(2n-2)d	(10n-3n(n)-3n(n-1)-3log n-3log(n-1)-7) (α+β)+ (4n-5) α+ (2n-2)d	(n+1)(2α+β)+(4n-1)(α+β)+(2n+2)α+n(4/5α+2β)+n(4/2α+β)	2n+(3n+2)(α+β)	7nα+5nβ+2nd

Here, _ = Ex-OR gate calculation complexity, _ = AND gate calculation complexity, d = NOT gate calculation complexity and ns = unit of measuring delay

Table VIII. Performance Analysis of a 128-Bit Reversible Carry Look-Ahead

Performance	Number of Gates	Quantum Cost	Garbage Output	Delay (ns)
Proposed Design	384	1408	288	61.44
Existing Design [13]	1969	6801	320	150.40
Improvement (%) with respect to [13]	80.79	79.29	10	59.14
Existing Design [14]	1711	6290	320	62.80
Improvement (%) with respect to [14]	77.55	77.61	10	2.16
Existing Design [15]	1922	3200	576	233.8
Improvement (%) with respect to [15]	80.02	56	50	73.72
Existing Design [16]	642	98586	256	103
Improvement (%) with respect to [16]	40.18	98.57	-12.5	40.34

ns= unit of measuring delay

VI. SIMULATION RESULTS:

All the synthesis and simulation results are performed using Verilog HDL. Theproposed circuit has been simulated using Xilinx ISE14.4. The simulation results are shown below figures. Figures. Shows the simulation of the 128-bit reversible carry look-ahead adder circuit and ensures the correctness of the proposed adder.

Fig.7: RTL schematic of 128-bit reversible carry look-ahead adder

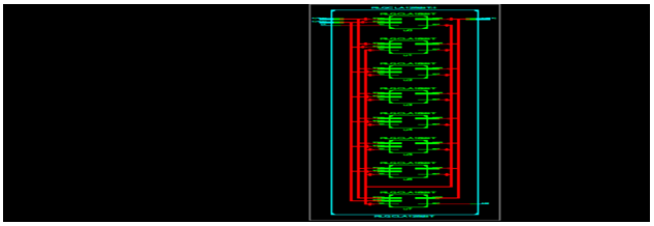

Fig.8: RTL sub schematic of 128-bit reversible carry look-ahead adder

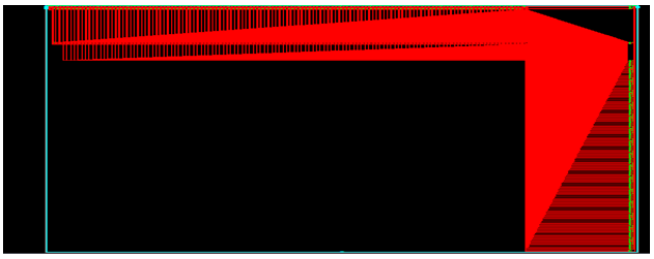

Fig.9: Technology schematic of 128-bit reversible carry look-ahead adder

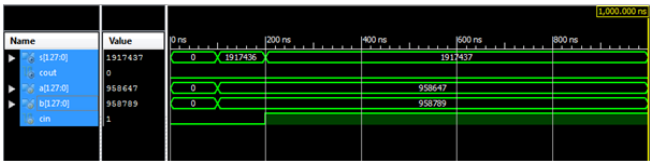

Fig.10: Simulation of 128-bit reversible carry look-ahead adder

VII.CONCLUSIONS:

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 look ahead adder. The carry look-ahead adder was constructed in two steps:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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].

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] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts, 1990.

[3] 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).

[4] 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).

[5] Biswas, A.K., Hasan, M.M., Chowdhury, A.R., and Babu, H.M.H.: 'Efficient approaches for designing reversible binary coded decimal adders', 440 Microelectron. J. 39, 12, 1693- 1703, 2008.

[6] R. Feynman, "Quantum mechanical computers," Foundations of Physics, vol. 16, pp. 507–531, 1986, 10.1007/BF01886518. [Online]. Available: http://dx.doi.org/10.1007/BF01886518.

[7] A. Peres, "Reversible logic and quantum computers," Phys. Rev. A, vol. 32, pp. 3266–3276, Dec 1985. [Online]. Available: http://- link.aps.org/doi/10.1103/PhysRevA.32.3266.

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

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

[10] Microwind - DSCH - digital schematic editor. Microwind.[Online]. Available: http://www.microwind.net/dsch.php.

[11] Man Lung Mui, Kaustav Banerjee, Senior Member, IEEE, and AmitMehrotra, A Global Interconnect Optimization Scheme for Nanometer Scale VLSI With Implications for Latency, Bandwidth, and Power Dissipation, IEEE TRANSACTIONS ON ELECTRON DEVICES, VOL. 51, No. 2, FEBRUARY 2004, pp. 195-203

[12] Prof. Krishna Saraswat, Interconnect Scaling, Stanford University Lecture Slide, URL: http:// www.stanford.edu / class /ee311 /NOTES /InterconnectScalingSlides.pdf

[13] Draper, T. G., Kutin, S. A., Rains, E. M., and Svore, K.M.: A logarithmic-depth quantum carry look-ahead adder. Quantum Information and computation.vol.6 No. 4&5, pp. 351-369 (2006).

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[14] H. Thapliyal, H. V. Jayashree, A. N. Nagamani, H.R. Arabnia, "Progress in Reversible Processor Design: A Novel Methodology for Reversible Carry Look-Ahead Adder", Springer Transactions on Computational Science XVII Lecture Notes in Computer Science Volume 7420, 2013, pp. 73-97.

[15] KanchanTiwari, Amar Khopade, PankajJadhav, "Optimized Carry Look-Ahead BCD Adder Using Reversible Logic", Technology Systems and Management Communications in Computer and Information Science (springer journal) Volume 145, 2011, pp. 260-265.

[16] Chakrabarti, A.,Sur-Kolay, S., "Designing quantum adder circuits and evaluating their error performance", International Conference on Electronic Design (ICED 2008), 1-3 Dec. 2008, pp. 1-6.

[17] Maslov, Dmitri A., "Dynamic programming algorithms as quantum circuits: symmetric function realization", Quantum Information and Computation II. Edited by Donkor, Eric; Pirich, Andrew R.; Brandt, Howard E. Proceedings of the SPIE, Volume 5436, pp. 386-393 (2004).

[18] B. Parhami, Computer arithmetic: algorithms and hardware designs. Oxford, UK: Oxford University Press, 2000.

[19] H.Buhrman, R.Cleve and A.Wigderson. "Quantum vs. Classical Communication and computation," Proceedings of the 30th Annual ACM Symposium on the Theory of Computation, ACM Press, El Paso, start page. 63 (1998).