International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

Implementation of Area Effective Carry Select Adders

Narender. J
M.Tech Scohlar (VLSI)
Dept. of ECE
Ashoka Institute of Engineering & Technology

T. Naga Raju
Assitant. Professor
Dept. of ECE
Ashoka Institute of Engineering & Technology

Abstract: **In the design of Integrated circuit area occupancy plays a vital role because of increasing the necessity of portable systems. Carry Select Adder (CSLA) is a fast adder used in data processing processors for performing fast arithmetic functions. From the structure of the CSLA, the scope is reducing the area of CSLA based on the efficient gate-level modification. In this paper 16 bit, 32 bit, 64 bit and 128 bit Regular Linear CSLA, Modified Linear CSLA, Regular Square-root CSLA (SQRT CSLA) and Modified SQRT CSLA architectures have been developed and compared. However, the Regular CSLA is still area-consuming due to the dual Ripple-Carry Adder (RCA) structure. For reducing area, the CSLA can be implemented by using a single RCA and an add-one circuit instead of using dual RCA. Comparing the Regular Linear CSLA with Regular SQRT CSLA, the Regular SQRT CSLA has reduced area as well as comparing the Modified Linear CSLA with Modified SQRT CSLA; the Modified SQRT CSLA has reduced area. The results and analysis show that the Modified Linear CSLA and Modified SQRT CSLA provide better outcomes than the Regular Linear CSLA and Regular SQRT CSLA respectively. This project was aimed for implementing high performance optimized FPGA architecture. Modelsim 10.0c is used for simulating the CSLA and synthesized using Xilinx PlanAhead13.4. Then the implementation is done in Virtex5 FPGA Kit.**

Keywords: **Field Programmable Gate Array (FPGA), efficient, Carry Select Adder (CSLA), Square-root CSLA (SQRTCSLA).**

1. INTRODUCTION

As the scale of integration keeps growing, more and more sophisticated signal processing systems are being implemented on a VLSI chip. These signal processing applications not only demand great computation capacity but also consume considerable amount of energy. While performance and Area remain to be the two major design tolls, power consumption has become a critical concern in today's VLSI system design. The need for low-power VLSI system arises from two main forces. First, with the steady growth of operating frequency and processing capacity per chip, large currents have to be delivered and the heat due to large power consumption must be removed by proper cooling techniques. Second, battery life in portable electronic devices is limited. Low power design directly leads to prolonged operation time in these portable devices.

Addition usually impacts widely the overall performance of digital systems and a crucial arithmetic function. In electronic applications adders are most widely used. Applications where these are used are multipliers, DSP to execute various algorithms like FFT, FIR and IIR. Wherever concept of multiplication comes adders come in to the picture. As we know millions of instructions per second are performed in microprocessors. So, speed of operation is the most important constraint to be considered while designing multipliers. Due to device portability miniaturization of device should be high and power consumption should be low. Devices like Mobile, Laptops etc. require more battery backup.

So, a VLSI designer has to optimize these three parameters in a design. These constraints are very difficult to achieve so depending on demand or application some compromise between constraints has to be made. Ripple carry adders exhibits the most compact design but the slowest in speed. Whereas carry look ahead is the fastest one but consumes more area. Carry select adders act as a compromise between the two adders. In 2002, a new concept of hybrid adders is presented to speed up addition process by Wang et al. that gives hybrid carry look-ahead/carry select adders design. In 2008, low power multipliers based on new hybrid full adders is presented.

DESIGN of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position.

The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum.

Need for Low Power Design: The design of portable devices requires consideration for peak power consumption to ensure reliability and proper operation. However, the time averaged power is often more critical as it is linearly related to the battery life. There are four sources of power dissipation in digital CMOS circuits: switching power, short-circuit power, leakage power and static power. The following equation describes these four components of power:

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

$$P_{avg} = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static} \quad (2.1)$$

$$= \alpha C_L V_{dd} V_s f_{ck} + I_{sc} V_{dd} + I_{leakage} V_{dd} + I_{static} V_{dd} \quad (2.2)$$

Pswitching is the switching power. For a properly designed CMOS circuit, this power component usually dominates, and may account for more than 90% of the total power. α denotes the transition activity factor, which is defined as the average number of power consuming transitions that is made at a node in one clock period. Vs is the voltage swing, where in most cases it is the same as the supply voltage, Vdd. CL is the node capacitance. It can be broken into three components, the gate capacitance, the diffusion capacitance, and the interconnect capacitance. The interconnect capacitance is in general a function of the placement and routing. fck is the frequency of clock. The switching power for static CMOS is derived as follows.

During the low to high output transition, the path from Vdd to the output node is con-ducting to charge CL. Hence, the energy provided by the supply source is

$$E = \int_0^\infty V_{dd} I(t) dt \quad (2.3)$$

where $I(t) = \frac{V_s}{R} e^{-t/RC_L}$ is the current drawn from the supply. Here, R is the resistance of the path between the Vdd and the output node. Therefore, the energy can be rewritten as

$$E = C_L V_{dd} V_s \quad (2.4)$$

During the high to low transition, no energy is supplied by the source. Hence, the average power consumed during one clock cycle is

$$P = \frac{E_{percycle}}{T} = C_L V_{dd} V_s f_{ck} \quad (2.5)$$

Eq. (2.4) and Eq. (2.5) estimate the energy and the power of a single gate only. From a system point of view, α is used to account for the actual number of gates switching at a point in time.

Pshortcircuit is the short-circuit power. It is a type of dynamic power and is typically much smaller than Pswitching. Isc is known as the direct-path short circuit current. It refers to the conducting current from power supply directly to ground when both the NMOS and PMOS transistors are simultaneously active during switching. Pleakage is the leakage power. Ileakage refers to the leakage current. It is primarily determined by fabrication technology considerations and originates from two sources. The first is the reverse leakage current of the parasitic drain-/source-substrate diodes. This current is in the order of a few femtoamperes per diode, which translates into a few microwatts of power for a million transistors. The second source is the sub threshold current of MOSFETs, which is in the order of a few nanoamperes. For a million transistors, the total subthreshold leakage current results in a few milliwatts of power. Pstatic is the static power and Istatic is static current. This current arises from circuits that have a constant source of current between the power supplies such as bias circuitries, pseudo-NMOS logic families. For CMOS logic family, power is dissipated only when the circuits switch, with no static power consumption.

Energy is independent of the clock frequency. Reducing the frequency will lower the power consumption but will not change the energy required to perform a given operation, as depicted by Eq. (2.4) and Eq. (2.5). It is important to note that the battery life is determined by energy consumption, whereas the heat dissipation considerations are related to the power consumption.

There are four factors that influence the power dissipation of CMOS circuits. They are technology, circuit design style, architecture, and algorithm. The challenge of meeting the contradicting goals of high performance and low power system operation has motivated the development of low power process technologies and the scaling of device feature sizes.

Design considerations for low power should be carried out in all steps in the design hierarchy, namely 1) Fundamental, 2) material, 3) device, 4) circuit, and 5) system.

Low Voltage: Power consumption is linearly proportional to voltage swing (Vs) and supply voltage (Vdd) as indicated in Eq. (2.5). For most CMOS logic families, the swing is typically rail-to-rail. Hence, power consumption is also said to be proportional to the square of the supply voltage, Vdd. Therefore, lowering the Vdd is an efficient approach to reduce both energy and power, presuming that the signal voltage swing can be freely chosen. This is, however, at the expense of the delay of circuits. The delay, td, can be shown to be proportional to $V_{dd}/(V_{dd} - V_T)^\gamma$.The exponent γ is between 1 and 2. It tends to be closer to 1 for MOS transistors that are in deep sub-micrometer region, where carrier velocity saturation may occur. γ increases toward 2 for longer channel transistors.

The current technology trends are to reduce feature size and lower supply voltage. Lowering Vdd leads to increased circuit delays and therefore lower functional throughput. Smaller feature size, however, reduces gate delay, as it is inversely proportional to the square of the effective channel length of the devices. In addition, thinner gate oxides impose voltage limitation for reliability reasons. Hence, the supply voltage must be lowered for smaller geometries. The net effect is that circuit performance improves as CMOS technologies scale down, despite of the Vdd reduction. Therefore, the new technology has made it possible to fulfill the contradicting requirements of low-power and high throughput.

The various techniques that are currently used to scale the supply voltage include optimizing the technology and device reliability, trading off area for low power in architecture driven approach, and exploiting the concurrency possibility in algorithmic transformations. Hence, the voltage scaling is limited by the threshold voltage Vth.

In applications such as digital processing, where the throughput is of more concern than the speed, architecture can

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

be designed to reduce the supply voltage at the expense of speed without throughput degradation. Hence, the performance of the system can be maintained.

2. ADDERS

2.1. RIPPLE CARRY ADDERS (RCA)

Concatenating the N full adders forms N bit Ripple carry adder. In this carry out of previous full adder becomes the input carry for the next full adder. It calculates sum and carry according to the following equations. As carry ripples from one full adder to the other, it traverses longest critical path and exhibits worst-case delay. Si = Ai xor Bi xorCi

Ci+1 = Ai Bi + (Ai + Bi) Ci; where i = 0, 1… n-1

RCA is the slowest in all adders (O (n) time) but it is very compact in size (O (n) area). If the ripple carry adder is implemented by concatenating N full adders, the delay of such an adder is 2N gate delays from Cin to Cout. The delay of adder increases linearly with increase in number of bits. Block diagram of RCA is shown in figure 1.

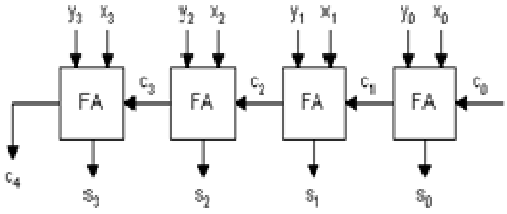


Figure.1. Block diagram of RCA

2.2. CARRY SKIP ADDER (CSKA)

A carry skip divides the words to be added in to groups of equal size of k-bits. Carry Propagate *pi* signals may be used within a group of bits to accelerate the carry propagation. If all the pi signals within the group are pi=1, carry bypasses the entire group as shown in figure 2.

P = pi * pi+1 * pi+2 *… pi+k
In this way delay is reduced as compared to ripple carry adder. The worst-case carry propagation delay in a N-bit carry skip adder with fixed block width *b*, assuming that one stage of ripple has the same delay as one skip, can be derived:

TCSKA = (b -1)+0.5+(N/b-2)+(b -1) = 2b + N/b – 3.5 Stages

Block width tremendously affects the latency of adder. Latency is directly proportional to block width. More number of blocks means block width is less, hence more delay. The idea behind Variable Block Adder (VBA) is to minimize the critical path delay in the carry chain of a carry skip adder, while allowing the groups to take different sizes. In case of carry skip adder, such condition will result in more number of skips between stages.

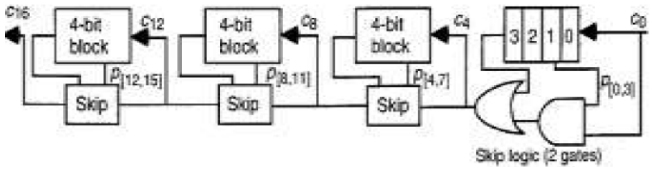


Figure.2. Block diagram of CSKA

Such adder design is called variable block design, which is tremendously used to fasten the speed of adder. In the variable block carry skip adder design we divided a 32-bit adder in to 4 blocks or groups. The bit widths of groups are taken as; First block is of 4 bits, second is of 6 bits, third is 18 bit wide and the last group consist of most significant 4 bits.

Table 1 shows that the logic utilization of carry skip and variable carry skip 32-bit adder. The power and delay, which are obtained also given in the table1. From table it can be observed that variable block design consumes more area as gate count and number of LUT's consumed by variable block design is more than conventional carry skip adder.

2.3. CARRY SELECT ADDER (CSA)

The carry select adder comes in the category of conditional sum adder. Conditional sum adder works on some condition. Sum and carry are calculated by assuming input carry as 1 and 0 prior the input carry comes. When actual carry input arrives, the actual calculated values of sum and carry are selected using a multiplexer. The conventional carry select adder consists of k/2 bit adder for the lower half of the bits i.e. least significant bits and for the upper half i.e. most significant bits (MSB's) two k/ bit adders. In MSB adders one adder assumes carry input as one for performing addition and another assumes carry input as zero. The carry out calculated from the last stage i.e. least significant bit stage is used to select the actual calculated values of output carry and sum. The selection is done by using a multiplexer. This technique of dividing adder in to stages increases the area utilization but addition operation fastens. The block diagram of conventional k bit adder is shown in figure 3.

Basics:

In electronics, a **carry-select adder** is a particular way to implement an adder, which is a logic element that computes the $(n+1)$-bit sum of two n-bit numbers. The carry-select adder is simple but rather fast, having a gate level depth of $O(\sqrt{n})$.

The carry-select adder generally consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders (therefore two ripple carry adders) in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After the two results

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

The number of bits in each carry select block can be uniform, or variable. In the uniform case, the optimal delay occurs for a block size of $\lfloor\sqrt{n}\rfloor$. When variable, the block size should have a delay, from addition inputs A and B to the carry out, equal to that of the multiplexer chain leading into it, so that the carry out is calculated just in time. The $O(\sqrt{n})$ delay is derived from uniform sizing, where the ideal number of full-adder elements per block is equal to the square root of the number of bits being added, since that will yield an equal number of MUX delays.

2.4. CARRY-LOOKAHEAD ADDER (CLA)

A **carry-look ahead adder** (CLA) is a type of adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, *ripple carry adder* for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits (see adder for detail on ripple carry adders). The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits. The Kogge-Stone adder and Brent-Kung adder are examples of this type of adder.

A ripple-carry adder works in the same way as pencil-and-paper methods of addition. Starting at the rightmost (least significant) digit position, the two corresponding digits are added and a result obtained. It is also possible that there may be a carry out of this digit position (for example, in pencil-and-paper methods, "9+5=4, carry 1"). Accordingly all digit positions other than the rightmost need to take into account the possibility of having to add an extra 1, from a carry that has come in from the next position to the right.

This means that no digit position can have an absolutely final value until it has been established whether or not a carry is coming in from the right. Moreover, if the sum without a carry is 9 (in pencil-and-paper methods) or 1 (in binary arithmetic), it is not even possible to tell whether or not a given digit position is going to pass on a carry to the position on its left. At worst, when a whole sequence of sums comes to ...99999999... (in decimal) or ...11111111... (in binary), nothing can be deduced at all until the value of the carry coming in from the right is known, and that carry is then propagated to the left, one step at a time, as each digit position evaluated "9+1=0, carry 1" or "1+1=0, carry 1". It is the "rippling" of the carry from right to left that gives a ripple-carry adder its name, and its slowness. When adding 32-bit integers, for instance, allowance has to be made for the possibility that a carry could have to ripple through every one of the 32 one-bit adders.

Carry look ahead depends on two things:

Calculating, for each digit position, whether that position is going to propagate a carry if one comes in from the right.

Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

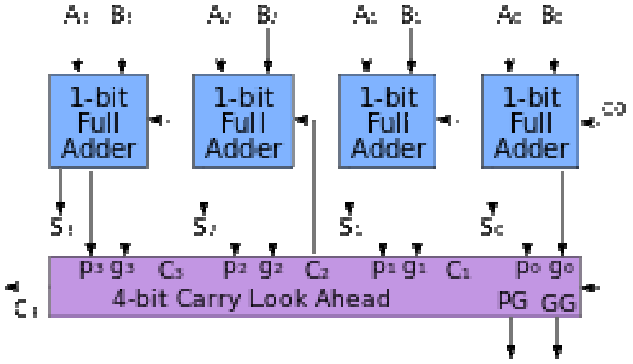


Figure.3. Block diagram of CLA

For each bit in a binary sequence to be added, the Carry Look Ahead Logic will determine 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. Then, when the actual addition is performed, there is no delay from waiting for the ripple carry effect (or time it takes for the carry from the first Full Adder to be passed down to the last Full Adder). Below is a simple 4-bit generalized Carry Look Ahead circuit that combines with the 4-bit Ripple Carry Adder we used above with some slight adjustments:

For the example provided, the logic for the generate (g) and propagate (p) values are given below. Note that the numeric value determines the signal from the circuit above, starting from 0 on the far left to 3 on the far right:

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1$$
$$C_3 = G_2 + P_2 \cdot C_2$$
$$C_4 = G_3 + P_3 \cdot C_3$$

Substituting C_1 into C_2, then C_2 into C_3, then C_3 into C_4 yields the expanded equations:

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + G_0 \cdot P_1 + C_0 \cdot P_0 \cdot P_1$$
$$C_3 = G_2 + G_1 \cdot P_2 + G_0 \cdot P_1 \cdot P_2 + C_0 \cdot P_0 \cdot P_1 \cdot P_2$$
$$C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$$

To determine whether a bit pair will generate a carry, the following logic works:

$$G_i = A_i \cdot B_i$$

To determine whether a bit pair will propagate a carry, either of the following logic statements work:

$$P_i = A_i \oplus B_i$$
$$P_i = A_i + B_i$$

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

2.5. CARRY SAVE ADDER

A **carry-save adder** is a type of digital adder, used in computer micro architecture to compute the sum of three or more *n*-bit numbers in binary. It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence of partial sum bits and another which is a sequence of carry bits. Consider the sum: 12345678+87654322=100000000.

Using basic arithmetic, we calculate right to left, "8+2=0, carry 1", "7+2+1=0, carry 1", "6+3+1=0, carry 1", and so on to the end of the sum. Although we know the last digit of the result at once, we cannot know the first digit until we have gone through every digit in the calculation, passing the carry from each digit to the one on its left. Thus adding two *n*-digit numbers has to take a time proportional to *n*, even if the machinery we are using would otherwise be capable of performing many calculations simultaneously.

In electronic terms, using bits (binary digits), this means that even if we have *n* one-bit adders at our disposal, we still have to allow a time proportional to *n* to allow a possible carry to propagate from one end of the number to the other. Until we have done this,

1. We do not know the result of the addition.
2. We do not know whether the result of the addition is larger or smaller than a given number (for instance, we do not know whether it is positive or negative).

A carry look-ahead adder can reduce the delay. In principle the delay can be reduced so that it is proportional to log*n*, but for large numbers this is no longer the case, because even when carry look-ahead is implemented, the distances that signals have to travel on the chip increase in proportion to *n*, and propagation delays increase at the same rate. Once we get to the 512-bit to 2048-bit number sizes that are required in public-key cryptography, carry look-ahead is not of much help

Montgomery multiplication, which depends on the rightmost digit of the result, is one solution; though rather like carry-save addition itself, it carries a fixed overhead , so that a sequence of Montgomery multiplications saves time but a single one does not. Fortunately exponentiation, which is effectively a sequence of multiplications, is the most common operation in public-key cryptography.

The carry-save unit consists of *n* full adders, each of which computes a single sum and carry bit based solely on the corresponding bits of the three input numbers. Given the three *n* - bit numbers **a**, **b**, and **c**, it produces a partial sum **ps** and a shift-carry **sc**:

$$ps_i = a_i \oplus b_i \oplus c_i$$
$$sc_i = (a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i)$$

The entire sum can then be computed by:

1. Shifting the carry sequence **sc** left by one place.
2. Appending a 0 to the front (most significant bit) of the partial sum sequence **ps**.
3. Using a ripple carry adder to add these two together and produce the resulting *n* + 1-bit value.

3. METHODOLOGY

3.1 CARRY SELECT ADDER

DESIGN of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated in to the next position .The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. However ,the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input cin=0 and cin=1, then the final sum and carry are selected by the multiplexers (mux).The basic idea of this work is to use Binary to Excess-1 Converter(BEC) instead of RCA with cin=1 in the regular CSLA to achieve lower area and power consumption The main advantage of this BEC logic comes from the lesser number of logic gates than the n-bitFull Adder (FA) structure.

The carry select adder comes in the category of conditional sum adder. Conditional sum adder works on some condition. Sum and carry are calculated by assuming input carry as 1 and 0 prior the input carry comes. When actual carry input arrives, the actual calculated values of sum and carry are selected using a multiplexer. The conventional carry select adder consists of k/2 bit adder for the lower half of the bits i.e. least significant bits and for the upper half i.e. most significant bits (MSB's) two k/ bit adders. In MSB adders one adder assumes carry input as one for performing addition and another assumes carry input as zero. The carry out calculated from the last stage i.e. least significant bit stage is used to select the actual calculated values of output carry and sum. The selection is done by using a multiplexer. This technique of dividing adder in to stages increases the area utilization but addition operation fastens.

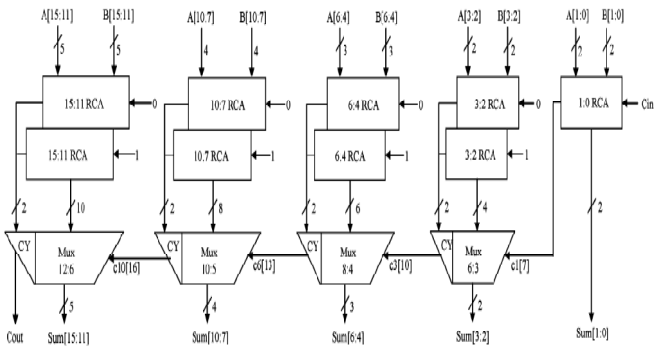


Fig.4. Regular 16-b SQRT CSLA.

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

3.2. MULTIPLEXER

In electronics, a multiplexer (or MUX) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line.[1] A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output.[2] Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth.[1] A multiplexer is also called a data selector. They are used in CCTV, and almost every business that has CCTV fitted, will own one of these.

An electronic multiplexer makes it possible for several signals to share one device or resource, for example one A/D converter or one communication line, instead of having one device per input signal. On the other hand, a demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. A multiplexer is often used with a complementary demultiplexer on the receiving end. An electronic multiplexer can be considered as a multiple-input, single-output switch, and a demultiplexer as a single-input, multiple-output switch.[3] The schematic symbol for a multiplexer is an isosceles trapezoid with the longer parallel side containing the input pins and the short parallel side containing the output pin.[4] The schematic on the right shows a 2-to-1 multiplexer on the left and an equivalent switch on the right. The wire connects the desired input to the output.

In digital circuit design, the selector wires are of digital value. In the case of a 2-to-1 multiplexer, a logic value of 0 would connect I_0 to the output while a logic value of 1 would connect I_1 to the output. In larger multiplexers, the number of selector pins is equal to $\lceil \log_2(n) \rceil$ where n is the number of inputs.

For example, 9 to 16 inputs would require no fewer than 4 selector pins and 17 to 32 inputs would require no fewer than 5 selector pins. The binary value expressed on these selector pins determines the selected input pin.

A 2-to-1 multiplexer has a Boolean equation where A and B are the two inputs, S is the selector input, and Z is the output:

$$Z = (A \cdot \overline{S}) + (B \cdot S)$$

This truth table shows that when $S=0$ then $Z=A$ but when $S=1$ then $Z=B$. A straightforward realization of this 2-to-1 multiplexer would need 2 AND gates, an OR gate, and a NOT gate.

Larger multiplexers are also common and, as stated above, require $\lceil \log_2(n) \rceil$ selector pins for n inputs. Other common sizes are 4-to-1, 8-to-1, and 16-to-1. Since digital logic uses binary values, powers of 2 are used (4, 8, 16) to maximally control a number of inputs for the given number of selector inputs.

The boolean equation for a 4-to-1 multiplexer is:

$$F = (A \cdot \overline{S_0} \cdot \overline{S_1}) + (B \cdot S_0 \cdot \overline{S_1}) + (C \cdot \overline{S_0} \cdot S_1) + (D \cdot S_0 \cdot S_1)$$

3.3. MODIFIED REGULAR CSLA USING BEC

The main idea of this work is to use BEC instead of the RCA with Cin=1in order to reduce the area and power consumption of the regular CSLA. To replace the n bit RCA, an n+1 bit BEC is required. A structure and the function table of a 4 bit BEC are show in Fig and Table II respectively.

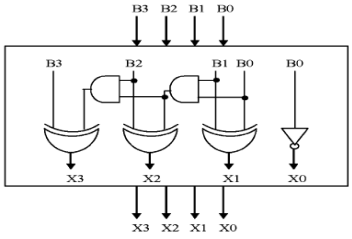


Figure.5. 4-bit BEC

TABLE II
FUNCTION TABLE OF THE 4-b BEC

B[3:0]	X[3:0]
0000	0001
0001	0010
⋮	⋮
1110	1111
1111	0000

Fig. above illustrates how the basic function of the CSLA is obtained by using 4-bit BEC together with the mux. One input of the 8:4 mux gets as it input((B3, B2, B1, and B0) and another input of the mux is the BEC output. This produces the two possible partial results in parallel and the mux is used to select either the BEC output or the direct inputs according to the control signal Cin. The importance of the BEC logic stems from the large silicon area reduction when the CSLA with large number of bits are designed. The Boolean expressions of the 4-bit BEC is listed as (note the functional symbols NOT, & XOR)

$$X0 = \sim B0$$
$$X1 = B0 \hat{} B1$$
$$X2 = B2 \hat{} (B0 \,\&\, B1)$$
$$X3 = B3 \hat{} (B0 \,\&\, B1 \,\&\, B2).$$

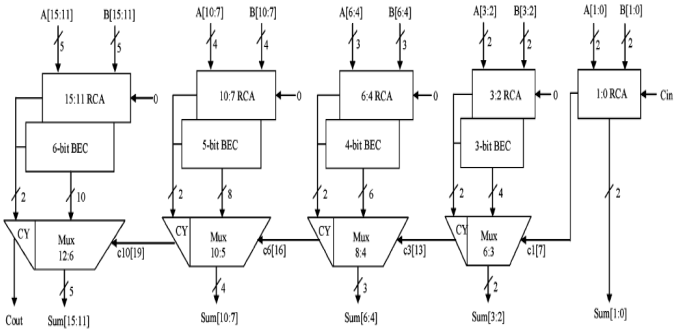


Figure.6. Modified SQRT Carry Select Adder The parallel RCA with Cin=1 is replaced with BEC

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

3.4. DELAY AND AREA EVALUATION OF MODIFIED CSLA

The structure of the proposed 16-b SQRT CSLA using BEC for RCA with Cin=1 to optimize the area and power is shown in above fig. We again split the structure into five groups. The delay and area estimation of each group are shown in Fig below. The steps leading to the evaluation are given here.

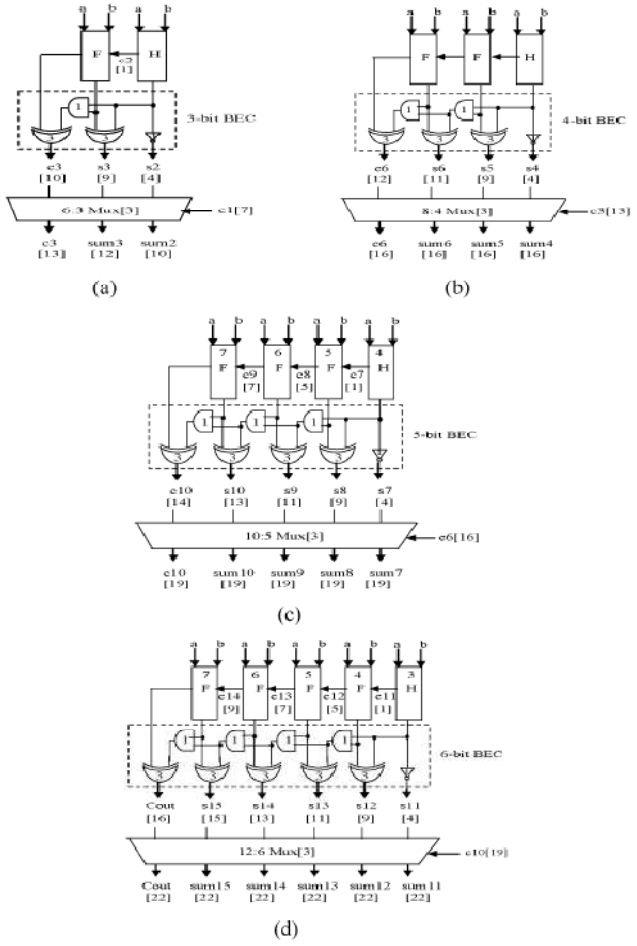


Figure.7. Delay and area evaluation of modified SQRT CSLA: (a) group2, (b) group3, (c) group4, and (d) group5. H is a Half Adder.

The group2 has one 2-b RCA which has 1 FA and 1 HA for Cin=0 Instead of another 2-b RCA with Cin=1 a 3-b BEC is used which adds one to the output from 2-b RCA. Based on the consideration of delay values of Table I, the arrival time of selection input C1 [t=7] of 6:3 mux is earlier than the s3 [t=9] and C3[t=10] and later than theS2[t=4]. Thus, the sum3 and finalC3 (output from mux) are depending on S3 and mux and partial C3 (input to mux) and mux, respectively. The sum2 depends on C1 and mux.

$$\text{Gate count} = 43\ (FA + HA + Mux + BEC)$$
$$FA = 13(1 * 13)$$
$$HA = 6(1 * 6)$$
$$AND = 1$$
$$NOT = 1$$
$$XOR = 10(2 * 5)$$
$$Mux = 12(3 * 4).$$

1) The area count of group2 is determined as follows:

3) Similarly, the estimated maximum delay and area of the other groups of the modified SQRT CSLA are evaluated and listed in Table IV.

TABLE IV
DELAY AND AREA COUNT OF MODIFIED SQRT CSLA

Group	Delay	Area
Group2	13	43
Group3	16	61
Group4	19	84
Group5	22	107

3.5. ARCHITECTURE OF MODIFIED 64-BIT SQRT CSLA

This architecture is similar to regular 64-bit SQRT CSLA, the only change is that, we replace RCA with Cin=1 among the two available RCAs in a group with a BEC. This BEC has a feature that it can perform the similar operation as that of the replaced RCA with Cin=1. Fig 4 shows the Modified block diagram of 64-bit SQRT CSLA. The number of bits required for BEC logic is 1 bit more than the RCA bits. The modified block diagram is also divided into various groups of variable sizes of bits with each group having the ripple carry adders, BEC and corresponding mux . As shown in the Fig.below, Group 0 contain one RCA only which is having input of lower significant bit and carry in bit and produces result of sum[1:0] and carry out which is acting as mux selection line for the next group, similarly the procedure continues for higher groups but they includes BEC logic instead of RCA with Cin=1.Based on the consideration of delay values, the arrival time of selection input C1 of 8:3 mux is earlier than the sum of RCA and BEC. For remaining groups the selection input arrival is later than the RCA and BEC. Thus, the sum1 and c1(output from mux) are depending on mux and results computed by RCA and BEC respectively.

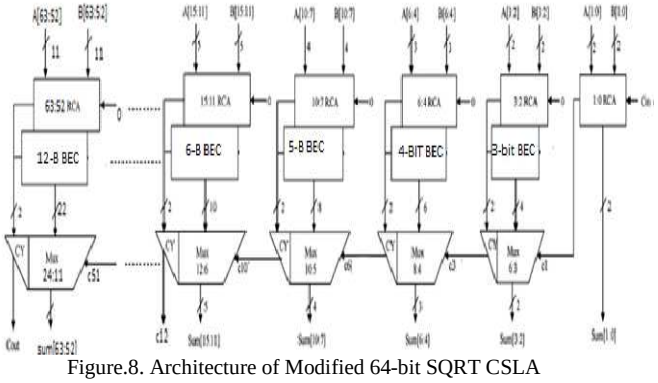


Figure.8. Architecture of Modified 64-bit SQRT CSLA

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 10, October 2014)

The sum2 depends on c1 and mux. For the remaining parts the arrival time of mux selection input is always greater than the arrival time of data inputs from the BEC's. Thus, the delay of the remaining MUX depends on the arrival time of mux

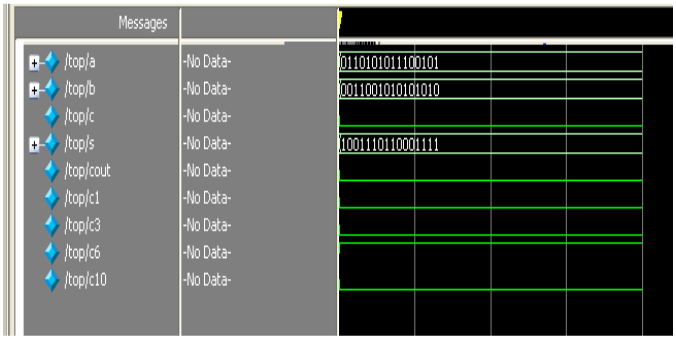
6. CONCLUSION

The implemented design in this work has been simulated using Verilog-HDL (Modelsim). The adders (of various size 16, 32, 64) are designed and simulated using Modelsim. All the Regular and Modified are also simulated in Modelsim and corresponding results are compared. After simulation the different size codes are synthesized using Xilinx ISE 9.1i. The simulated files are imported into the synthesized tool and corresponding values of delay and area are noted. The synthesized reports contain area and delay values for different sized adders. The similar design flow is followed for both the regular and modified SQRT CSLA of different sizes.

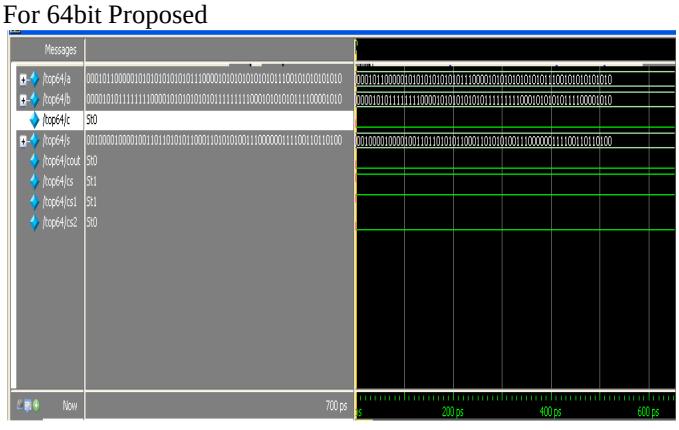
The comparative values of areas shows that the number of LUT will be more for modified method for the 16, 32 and 64. This value decreases gradually for 128 bits. For 256 bits the value almost equal to regular method which will reduces more for still higher order bits. Thus the modified method decreases the delay and also area to a great extent.

7. SIMULATION RESULT

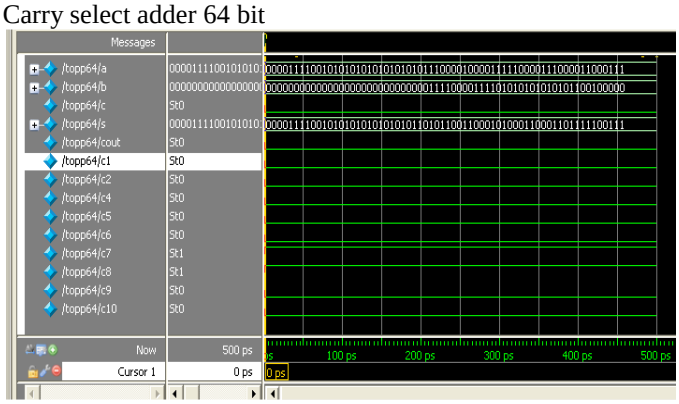
Carry select adder proposed system with 16 bit



For 64bit Proposed



Carry select adder 64 bit



REFERENCES

[1]. O. J. Bedrij, "Carry-select adder," *IRE Trans. Electron. Comput.*, pp. 340–344, 1962.
[2]. B. Ramkumar, H.M. Kittur, and P. M. Kannan, "ASIC implementation of modified faster carry save adder," *Eur. J. Sci. Res.*, vol. 42, no. 1, pp. 53–58, 2010.
[3]. T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripple carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
[4]. Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area," *Electron. Lett.*, vol. 37, no. 10, pp. 614–615, May 2001.
[5]. J. M. Rabaey, *Digtal Integrated Circuits—A Design Perspective*. Upper Saddle River, NJ: Prentice-Hall, 2001.
[6]. Y. He, C. H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for lowpower applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, vol. 4, pp. 4082–4085.
[7]. Cadence, "Encounter user guide," Version 6.2.4, March 2008.