Design and Implementation of Efficient Carry Select Adder on Reconfigurable Device Using Novel Logic Formulation

Subha.K1, G.Nusrath Sumaiya2 & S.Murugavalli3
1,2,3Assistant Professor, M.A.M School of engineering, Tamil Nadu, India

Abstract: Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. The logic operations involved in conventional carry select adder (CSLA) and binary to excess-1 converter (BEC)-based CSLA are data dependence and redundant logic operations. In this paper, proposed a new logic formulation for CSLA to eliminate the redundant logic operations present in the conventional CSLA. In the proposed scheme, the carry select (CS) operation is scheduled before the calculation of final-sum, which is different from the conventional approach. The proposed CSLA design has reduced area and delay as compared with BEC-based CSLA. Due to the small carry-output delay, the proposed CSLA design is suitable for Multistage CSLA(SQRT-CSLA). The performance of the CSLA is evaluated by implementing a MAC unit using either conventional, BEC or proposed CSLA in the adder part, for different bit-widths. This work focuses on the performance of CSLA in terms of area, delay and power. The analysis of the result shows that the MAC unit using proposed SQRT-CSLA has less area, delay and power when compared with MAC unit with existing SQRT-CSLAs. The system has been designed efficiently using Verilog HDL codes and simulated using Quartus II 9.1. The hardware implementation is done by using Altera-FPGA.

Index Terms— Carry Select Adder (CSLA), BEC, Multistage CSLA(SQRT-CSLA)

1. Introduction

1.1 Adders

Design of area and power-efficient high-speed data path logic systems are one of the most areas of research in VLSI system design are increasingly used in portable and mobile devices, multistandard wireless receivers, and biomedical instrumentation. An adder is the main component of an arithmetic unit. In electronic applications adders are most widely used. A complex digital signal processing (DSP) system involves several adders. An efficient adder design essentially improves the performance of a complex DSP system. Some other applications of adders are in Multiply – Accumulate (MAC) unit. Adders are also used in multipliers, in high speed integrated circuits and in digital signal processing to execute various algorithms like FFT, IIR and FIR. Therefore, regarding the efficient implementation of an arithmetic unit, the binary adder structures become a very critical hardware unit.

Large number of different circuit architectures with different performance characteristics and widely used in the practice. Although many researches dealing with the binary adder structures have been done, the studies based on their comparative performance analysis are only a few. In this project, qualitative evaluations of the classified binary adder architectures are given. Among the huge member of the adders we wrote VHDL (Hardware Description Language) code for Ripple-carry, Carry-select and Carry-look ahead to emphasize the common performance properties belong to their classes. In the following section, we give a brief description of the studied adder architectures.

Table 1.1: Categorization of adders with respect to delay time and capacity

<table>
<thead>
<tr>
<th>Adder Class Schemes</th>
<th>Delay (T)</th>
<th>Product (A×T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple-carry (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carry select (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carry-look-ahead (3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*1 denotes the LEVEL number

1.1.1 Ripple Carry Adder (RCA)

Ripple carry adder is composed of cascaded full adders for n-bit adder, as shown in figure.1.1. It is constructed by cascading full adder.
blocks in series. The carry out of one stage is fed directly to the carry-in of the next stage. For an n-bit parallel adder it requires n full adders.

![Figure 1.1: 4-Bit Ripple Carry Adder](image)

**Drawbacks**

- Not very efficient when large number bit numbers are used.
- Delay increases linearly with bit length.

**Delay**

Delay from Carry-in to Carry-out is more important than from A to carry-out or carry-in to SUM, because the carry-propagation chain will determine the latency of the whole circuit for a Ripple-Carry adder. Considering the above worst-case signal propagation path we can thus write the following equation. For a k-bit RCA worst case path delay is

\[
T_{RCA-k \text{ bit}} = T_{FA}(x_0, y_0, c_0) + (k-2) \cdot T_{FA}(C_{in}, C_i) + T_{FA}(C_{in}, S_{k-1})
\]

**Logic Equations**

\[
\begin{align*}
& g_i = a_i \cdot b_i \quad (1.1) \\
& p = a_i \oplus b_i \quad (1.2) \\
& S_i = p \oplus c_i \quad (1.3) \\
& C_{i+1} = g_i + p \cdot c_i \quad (1.4)
\end{align*}
\]

**Complexity and Delay for N-Bit RCA Structure**

\[
\begin{align*}
& A_{RCA} = O(n) = 7n \quad (1.5) \\
& T_{RCA} = O(n) = 2n \quad (1.6)
\end{align*}
\]

**1.1.2 Carry Look Ahead Adders (CLA)**

Carry Look Ahead Adder can produce carries faster due to carry bits generated in parallel by an additional circuitry whenever inputs change. This technique uses carry bypass logic to speed up the carry propagation.

![Figure 1.2: 4-Bit CLA Logic Equations](image)

**Logic Equations**

\[
\begin{align*}
& p_i = a_i + b_i \quad (1.7) \\
& g_i = a_i \cdot b_i \quad (1.8) \\
& S_i = a_i \oplus b_i \oplus c_i \quad (1.9) \\
& C_{i+1} = g_i + p_i \cdot c_i \quad (2.0)
\end{align*}
\]

As increase the no of bits in the Carry Look Ahead adders, the complexity increases because the no. of gates in the expression \( C_{i+1} \) increases. So practically its not desirable to use the traditional CLA shown above because it increase the Space required and the power too.

**1.1.3 Carry Select Adders (CSLA)**

In Carry select adder scheme, blocks of bits are added in two ways: one assuming a carry-in of 0 and the other with a carry-in of 1. This results in two precomputed sum and carry-out signal pairs \((s_{i+k}, c_i, s_{i+1-k}, c_i)\), later as the block’s true carry-in \((c_k)\) becomes known, the correct signal pairs are selected. Generally multiplexers are used to propagate carries.
Because of multiplexers larger area is required.

- Have a lesser delay than Ripple Carry Adders (half delay of RCA).
- Hence we always go for Carry Select Adder while working with smaller no of bits.

**Logic Equations**

\[
 s_{i-1:k} = c_k s_{i-1:k}^0 + c_k s_{i-1:k}^1. \tag{2.1}
\]

\[
 c_i = c_k c_{i-1}^0 + c_k c_{i-1}^1. \tag{2.2}
\]

**Complexity and Delay for N-Bit CSLA Structure**

\[
 A_{CSLA} = O(n) = 14n \tag{2.3}
\]

\[
 T_{CSLA} = O\left(n^{1/2}\right) = 2.8n^{1/2} \tag{2.4}
\]

### Table 1.2 Theoretical Comparison of Area Occupied (Ax)

<table>
<thead>
<tr>
<th>Adder Name</th>
<th>Complex (Ax)</th>
<th>Area for n-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry Adder (RCA)</td>
<td>O(n)</td>
<td>7n</td>
</tr>
<tr>
<td>Carry Look Ahead Adder (CLA)</td>
<td>O(n)</td>
<td>4n</td>
</tr>
<tr>
<td>Carry Select Adder (CSLA)</td>
<td>O(n)</td>
<td>14n</td>
</tr>
</tbody>
</table>

### Table 1.3 Comparison Of Time Required.

<table>
<thead>
<tr>
<th>Adder Name</th>
<th>Complex (A)</th>
<th>Delay for 8-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ripple Carry Adder (RCA)</td>
<td>O(n)</td>
<td>20.8 ns</td>
</tr>
<tr>
<td>Carry Select Adder (CSLA)</td>
<td>O(n^{1/2})</td>
<td>12.8 ns</td>
</tr>
<tr>
<td>Carry Look Ahead Adder (CLA)</td>
<td>O(log n)</td>
<td>17.6 ns</td>
</tr>
</tbody>
</table>

### 1.1.5 Comparison among Adders

As can be seen above stated the theoretical comparison of AREA required and both the theoretical and simulated value of TIME required. The values stated above are the values for 8-bit adders. So analyzing the above facts reached at the following conclusions about different adders and intelligent use of them in different circumstances according to the SPACE TIME trade-off. The results can be summarized as follows.

Regarding the circuit area complexity in the adder architectures, the ripple-carry adder (RCA) in the first class is the most efficient one, but the carry select adder (CSLA) in the fourth class with highest complexity is the least efficient one.

Considering the circuit delay time, Carry Select Adder (CSLA) is the fastest one for every n-bit length, so has the shortest delay. Otherwise, Ripple Carry Adder (RCA) is the slowest one, due to the long carry propagation.

Carry look ahead adder reduces the delay compared to ripple carry adder but as the adder size
increases complexity increases also, fan in to the gates increases. DeLay of the circuit depends upon, how many gates are connected in series and delay of a gate depends upon no of fan ins, so the later term dominates as adder size increases in carry look ahead adder.

Defined a term **Area-Delay Product** which gave the clear picture of the space-time trade-off. It is worthy to note that while consider all the adders discussed above **Ripple Carry adders** and **Carry Select Adders** are the two sides of the spectrum. Because, while Ripple Carry Adders have a smaller area and lesser speed, in contrast to which Carry Select adders have high speed (nearly twice the speed of Ripple Carry Adders) and occupy a larger area. Hence among the three, **Carry Select Adder** has the least Time.

### 1.2 Multiplier

Multiplication is a computation intensive and the core operation in many algorithms used in scientific computations. The computational complexity of algorithm used in digital signal processing (DSP) has gradually increased. As a result fast and efficient parallel multipliers are required for general purpose digital signal processors (DSPs) as well as application specific architectures for DSP.

Generally as we know multiplication goes in two basic steps, Partial product generation and partial addition. In multiplication the addition stage is in critical path that is it takes long delay and also it requires large hardware resources to compute product from partial product.

Figure 1.4-16bit Multiplier.

In particular, if the processing has to be performed under real time conditions, such algorithms have to deal with high throughput rates. In many cases implementation of DSP algorithm demands using application specific integrated circuits (ASICs). This is especially required for image processing applications. Since development costs for ASICs are high, algorithms should be verified and optimized before implementation.

However, with recent advancements in very large scale integration (VLSI) technology, hardware implementation has become a desirable alternative. Significant speedup in computation time can be achieved by assigning computation intensive tasks to hardware and by exploiting the parallelism in algorithms. Recently, field programmable gate arrays (FPGAs) have emerged as a platform of choice for efficient hardware implementation of computation intensive algorithms. FPGAs enable a high degree of parallelism and can achieve orders of magnitude speedup over general purpose processors (GPPs). This is a result of increasing embedded resources available on FPGA. FPGA have the benefit of hardware speed and the flexibility of software. The three main factors that play an important role in FPGA based design are the targeted FPGA architecture, electronic design automation (EDA) tools and design techniques employed at the algorithmic level using hardware description languages. In FPGAs, the choice of the optimum multiplier involves three key factors: area, propagation delay and reconfiguration time. Therefore, FPGA has become viable technology and an attractive alternative to ASICs.

### 1.3 Applications
- DSP
- Multipliers
- Multimedia.

### 2. Proposed Method

#### 2.1 proposed Description

In the proposed scheme, the carry select (CS) operation is scheduled before the calculation of final-sum, which is different from the conventional approach. Bit patterns of two anticipating carry words (corresponding to cin = 0 and 1) and fixed cin bits are used for logic optimization of CS and generation units. An efficient CSLA design is obtained using optimized logic units. The proposed CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry-output delay, the proposed CSLA design is for Single stage CSLA. A theoretical estimate shows that the proposed CSLA involves nearly 26% less area–delay–product (ADP) than the BEC-based CSLA, which is best among the existing CSLA designs, for 64bit-widths.
2.2 Logic Formulation

The proposed logic formulation for the CSLA is given as

\[ s_0(i) = A(i) \land B(i) \]  
(2.1)

\[ C_0(i) = A(i) \land B(i) \]  
(2.2)

\[ C_{10}(i) = C_{10}(i-1) \land s_0(i) + c_0(i) \text{ for } (c_{10}(0) = 0) \]  
(2.3)

\[ c_0(i) = c_0(i-1) \text{ if } (c_{1n} = 0) \]  
(2.4)

\[ c_1(i) = c_1(i-1) \text{ if } (c_{1n} = 1) \]  
(2.5)

\[ c_{out} = c(n-1) \]  
(2.6)

\[ s(i) = s(i) \land c_{1n} = s_0(i) \land c(i-1) \]  
(2.7)

The proposed CSLA is based on the logic formulation given in (2.1)–(2.7), and its structure is shown in Fig. 2.1.

Figure 2.1 Proposed carry select adder designs.

It consists of one HSG unit, one FSG unit, one CG unit, and one CS unit. The CG unit is composed of two CGs (CG0 and CG1) corresponding to input-carry ‘0’ and ‘1’.

2.2.1 HSG Unit

The HSG receives two \( n \)-bit operands (\( A \) and \( B \)) and generate half-sum word \( s0 \) and half-carry word \( c0 \) of width \( n \) bits each. The logic diagram is shown below.

Figure 2.1(b) Gate-level optimized design of (CG0) for input-carry = 0.

2.2.2 CG0 and CG1 Unit

CG0 and CG1 receive \( s0 \) and \( c0 \) from the HSG unit and generate two \( n \)-bit full-carry words \( c01 \) and \( c11 \) corresponding to input-carry ‘0’ and ‘1’, respectively. The logic circuits of CG0 and CG1 are optimized to take advantage of the fixed input-carry bits. The optimized designs of CG0 and CG1 are shown in Fig. 2.1(b) and (c), respectively.

Figure 2.1(c) Gate-level optimized design of (CG1) for input-carry = 1.
2.2.3 CS Unit

The CS unit selects one final carry word from the two carry words available at its input line using the control signal cin. It selects c01 when cin = 0; otherwise, it selects c11. The CS unit can be implemented using an n-bit 2-to-1 MUX. However, we find from the truth table of the CS unit that carry words c01 and c11 follow a specific bit pattern. If c01(i) = ‘1’, then c11(i) = 1, irrespective of s0(i) and c0(i), for 0 ≤ i ≤ n − 1. This feature is used for logic optimization of the CS unit. The optimized design of the CS unit is shown in Fig. 2.1(d), which is composed of n AND–OR gates. The final carry word c is obtained from the CS unit.

2.2.4 FSG Unit

The MSB of c is sent to output as cout, and (n − 1) LSBs are XORed with (n − 1) MSBs of half sum (s0) in the FSG [shown in Fig. 2.1(e)] to obtain (n − 1) MSBs of final sum (s). The LSB of s0 is XORed with cin to obtain the LSB of s.

2.3 MAC Unit Operation

The Multiplier-Accumulator (MAC) operation is the key operation not only in DSP applications but also in multimedia information processing and various other applications. As mentioned above, MAC unit consist of multiplier, adder and register/accumulator. In this project, used 64 bit multiplier. The MAC inputs are obtained from the memory location and given to the multiplier block. This will be useful in 64 bit digital signal processor. The input which is being fed from the memory location is 64 bit. When the input is given to the multiplier it starts computing value for the given 64 bit input and hence the output will be 128 bits. The multiplier output is given as the input to three different adders which performs addition. The function of the MAC unit is given by following equation

\[ F = \sum P_j Q_j \]  

(2.8)

The output of three adders is 129 bit i.e. one bit is for the carry (128 bits+ 1 bit). Then, the output is given to the accumulator register. The accumulator register used in this design is Parallel in Parallel out (PIPO). Since the bits are huge and also three CSLA adders produces all the output values in parallel, PIPO register is used where the input bits are taken in parallel and output is taken in parallel. The output of the accumulator register is taken out or fed back as one of the input to different three CSLA adders. The figure shows the basic architecture of MAC unit.

![Figure 2.2 MAC unit Architecture.](image-url)
2.4 Performance Comparison

2.4.1. Area–Delay Estimation Method

Consider all the gates to be made of 2-input AND, 2-input OR, and inverter (AOI). A 2-input XOR is composed of 2 AND, 1 OR, and 2 NOT gates. The area and delay of the 2-input AND, 2-input OR, and NOT gates (shown in Table 2.1) are taken from the Synopsys Armenia Educational Department (SAED) 90-nm standard cell library datasheet for theoretical estimation. The delays of a design are calculated using the following relations:

\[ A = a \cdot N_a + r \cdot N_o + i \cdot N_i \]  \hspace{1cm} (2.9)
\[ T = \text{na} \cdot T_a + \text{no} \cdot T_o + \text{ni} \cdot T_i \]  \hspace{1cm} (2.10)

Where \((N_a, N_o, N_i)\) and \((\text{na}, \text{no}, \text{ni})\), respectively, represent the (AND, OR, NOT) gate counts of the total design and its critical path. \((a, r, i)\) and \((Ta, To, Ti)\), respectively, represent the area and delay of one (AND, OR, NOT) gate. We have calculated the (AOI) gate counts of each design for area and delay estimation. Using (5a) and (5b), the area and delay of each design are calculated from the AOI gate counts \((N_a, N_o, N_i)\), \((\text{na}, \text{no}, \text{ni})\), and the cell details of Table 2.1.

2.4.2 Single-Stage CSLA

The general expression to calculate the AOI gate counts of the \(n\)-bit proposed BEC-based CSLA and conventional CSLA are given in Table 2.1 of single stage design. Calculated the AOI gate counts on the critical path of the proposed \(n\)-bit CSLA and CSLAs and used those AOI gate counts in (3.9) to find an expression for delay of final-sum and output-carry in the unit of \(T_i\) (NOT gate delay). The delay of the \(n\)-bit single-stage CSLA for comparison. For further analysis of the critical path of the proposed CSLA, the delay of each intermediate and output signals of the proposed \(n\)-bit CSLA design of Fig. 2.1 is shown.

Using the expressions of AOI gate details of Table 2.1, we have estimated the area and delay complexities of the proposed CSLA and the existing CSLA including the conventional one for input bit-widths. For the single-stage CSLA, the input-carry delay is assumed to be \(t = 0\) and the delay of final-sum \((fs)\) represents the adder delay.

The proposed CSLA involves nearly 27% less area and 1.3% less output power than that of conventional CSLA. Consequently, the conventional CSLA of involves 24% higher ADP and 13% higher PDP than the proposed CSLA, for 64-bit-width. Compared with the BEC-based CSLA, the proposed CSLA design has marginally less ADP. However, in the BEC-based CSLA, delay increases at a much higher rate than the proposed CSLA design for higher bit widths. Compared with the BEC based CSLA, the proposed CSLA involves 1.6% more delay, but it involves nearly 26% less ADP and nearly 3% less PDP due to less area complexity. Power consumption is more in proposed single-stage CSLA when it compared with the existing CSLAs, to reduce the power consumption go for multi-stage CSLA.

2.4.3 Multistage CSLA (SQRT-CSLA)

The multipath carry propagation feature of the CSLA is fully exploited in the SQRT-CSLA, which is composed of a chain of CSLAs. CSLAs of increasing size are used in the SQRT-CSLA to extract the maximum concurrence in the carry propagation path. Using the SQRT-CSLA design, large-size adders are implemented with significantly less delay than a single-stage CSLA of same size. However, carry propagation delay between the CSLA stages of SQRT-CSLA is critical for the overall adder delay. Due to early generation of output-carry with multipath carry propagation feature, the proposed CSLA design is more suitable than the existing CSLA designs for area–delay efficient implementation of SQRT-CSLA. A 16-bit SQRT-CSLA design using the proposed CSLA is shown in Fig. 2.3, where the 2-bit RCA, 2-bit CSLA, 3-bit CSLA, 4-bit CSLA, and 5-bit CSLA are used. Considered the cascaded configuration for different bit-widths. To demonstrate the advantage of the proposed CSLA design in SQRT-CSLA, estimated the area and delay of SQRT-CSLA using the proposed CSLA design and the BEC-based CSLA for bit-widths 16, 32, 64, 128.
2.4.4 Implementation of Mac Unit Using Conventional CSLA

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√n.

The carry-select adder generally consists of two ripple carry adders and a multiplexer, shown in figure 2.5. 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 are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

2.4.5 Implementation of Mac Unit Using BEC based CSLA

The conventional CSLA is not area efficient and delay for the operation is more because it uses multiple set of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input and then the final sum and carry are selected by the multiplexers.

To avoid this problem, the regular CSLA structure is modified using n-bit Binary to Excess-1 code converter (BEC) to improve the speed of operation. To improve the speed of operation use the Binary to Excess-1 Converter (BEC) instead of RCA with Cin=1 in the regular CSLA to achieve less delay. A structure and the function table of a 4-bit BEC function of the CSLA is obtained by using the 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 performance of the CSLA is evaluated by implementing an MAC unit using BEC based CSLA shown in figure 2.6.

![Image](image1.png)

Figure 2.7(a) 4bit BEC, 2.7(b) n-bit BEC based CSLA.

2.4.6 Implementation of Mac Unit Using Proposed CSLA

The proposed CSLA is based on the logic formulation and its structure is shown in Fig. 2.1(a). It consists of one HSG unit, one FSG unit, one CG unit, and one CS unit. The CG unit is composed of two CGs (CG0 and CG1) corresponding to input-carry ‘0’ and ‘1’. The HSG receives two n-bit operands (A and B) and generate half-sum word s0 and half-carry word c0 of width n bits each. Both CG0 and CG1 receive s0 and c0 from the HSG unit and generate two n-bit full-carry words c10 and c11 corresponding to input-carry ‘0’ and ‘1’, respectively. The logic diagram of the HSG unit is shown in Fig. 2.1(b). The logic circuits of CG0 and CG1 are optimized to take advantage of the fixed input-carry bits. The optimized designs of CG0 and CG1 are shown in Fig. 2.1(c) and (d), respectively. The CS unit selects one final carry word from the two carry words available at its input line using the control signal cin. It selects c10 when cin = 0; otherwise, it selects c11. The CS unit can be implemented using an n-bit 2-to-1 MUX.

![Image](image2.png)

Figure 2.8 MAC unit using Proposed CSLA.

However, find from the truth table of the CS unit that carry words c10 and c11 follow a specific bit pattern. If c10(i) = 1, then c11(i) = 1, irrespective of s0(i) and c0(i), for 0 ≤ i ≤ n - 1. This feature is used for logic optimization of the CS unit. The optimized design of the CS unit is shown in Fig. 3.1(e), which is composed of n AND–OR gates. The final carry word c is obtained from the CS unit. The MSB of c is sent to output as cout, and (n - 1) LSBs are XORed with (n - 1) MSBs of half-sum (s0) in the FSG [shown in Fig. 2.1(f)] to obtain (n - 1) MSBs of final-sum (s). The LSB of s0 is XORed with cin to obtain the LSB of s. The performance of the CSLA is evaluated by implementing an MAC unit using Proposed CSLA shown in figure 2.8.

3. Simulation and Result Analysis

3.1 SIMULATION RESULTS OF MULTISTAGE (SQRT) CSLA

The design is develop using Verilog HDL and then simulation and synthesis the results for different CSLAs using Quartus II 9.1 and the family utilize is Stratix III, device utilize is EP3SL50F484C2. Area, Delay and Power
Calculations have found for Different Multi stage CSLAs using Quartus II 9.1.

Figure 3.1 Simulation output of 16-bit Conventional SQRT-CSLA.

Figure 3.2 Simulation output of 16-bit BEC based SQRT-CSLA.

Figure 4.3 Simulation output of 16-bit Proposed SQRT-CSLA.

3.2 Result Analysis of Multi-Stage (SQRT) CSLA

After the observation of simulation waveforms, synthesis is performed for calculation of delay and area and thereby the speed and power of the CSLA’s are calculated and a comparison of conventional, BEC and proposed CSLA is made in terms of delay, area and power and listed in the below table.

Table 3.1 Comparison of area, delay and power of existing and proposed for Multi-Stage CSLA.

<table>
<thead>
<tr>
<th></th>
<th>128-bit</th>
<th>64-bit</th>
<th>32-bit</th>
<th>16-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>POWER(mW)</td>
<td>1.15</td>
<td>1.32</td>
<td>1.05</td>
<td>1.76</td>
</tr>
<tr>
<td></td>
<td>0.80</td>
<td>1.35</td>
<td>0.52</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td>0.58</td>
<td>0.51</td>
<td>0.55</td>
<td>0.50</td>
</tr>
<tr>
<td>DELAY(ns)</td>
<td>57.051</td>
<td>38.052</td>
<td>21.434</td>
<td>16.872</td>
</tr>
<tr>
<td>AREA (ALUTs)</td>
<td>128-bit</td>
<td>395</td>
<td>372</td>
<td>285</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>64-bit</td>
<td>188</td>
<td>189</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>32-bit</td>
<td>87</td>
<td>88</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>16-bit</td>
<td>41</td>
<td>37</td>
<td>41</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGN</th>
<th>Conv CSLA</th>
<th>BEC CSLA</th>
<th>Pro CSLA</th>
</tr>
</thead>
</table>

**Table 4.2 Comparison of MAC unit implementation with Conventional, BEC and Proposed CSLA.**

<table>
<thead>
<tr>
<th>POWER(mW)</th>
<th>64-bit</th>
<th>32-bit</th>
<th>16-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>20.46</td>
<td>18.13</td>
<td>19.92</td>
</tr>
<tr>
<td>32-bit</td>
<td>14.55</td>
<td>12.24</td>
<td>15.40</td>
</tr>
<tr>
<td>16-bit</td>
<td>10.61</td>
<td>15.52</td>
<td>10.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DELAY(ns)</th>
<th>64-bit</th>
<th>32-bit</th>
<th>16-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>20.800</td>
<td>9.826</td>
<td>8.026</td>
</tr>
<tr>
<td>32-bit</td>
<td>11.862</td>
<td>7.189</td>
<td>6.796</td>
</tr>
<tr>
<td>16-bit</td>
<td>8.039</td>
<td>5.737</td>
<td>5.258</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AREA(ALUTs)</th>
<th>64-bit</th>
<th>32-bit</th>
<th>16-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit</td>
<td>488</td>
<td>431</td>
<td>376</td>
</tr>
<tr>
<td>32-bit</td>
<td>196</td>
<td>184</td>
<td>180</td>
</tr>
<tr>
<td>16-bit</td>
<td>84</td>
<td>76</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DESIGN</th>
<th>Conv CSLA</th>
<th>BEC CSLA</th>
<th>Pro CSLA</th>
</tr>
</thead>
</table>

### 4. Conclusion

In this paper, analyzed the logic operations involved in the conventional and BEC-based CSLAs are data dependence and redundant logic operations. In this paper, sproposed a new logic formulation for CSLA to eliminate the redundant logic operations present in the conventional CSLA. In the proposed scheme, the CS operation is scheduled before the calculation of final-sum, which is different from the conventional approach. The proposed CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry output delay, the proposed CSLA design is suitable for the SQRT adder. All the three models of SQRT- CSLA are designed and are implemented in Verilog HDL using Quartus II 9.1 tool and the results are compared in terms of area, delay and power. The Proposed CSLA proves to be the High Speed and Low Power CSLA. The performance of this CSLA in terms of area, delay and power is evaluated by implementing an MAC unit by using the CSLA in the adder part and again it proves to be the High Speed and Low Power CSLA. It is also implemented with Altera FPGA. The Proposed CSLA architecture is therefore, high speed, low power and area efficient for VLSI hardware implementation.

### 5. References


