Implementation of Carry Select Adder for Low-Power and Area-Efficient Architecture

**BATHINI SUJITHA REDDY**¹, **M.L.RAVICHANDRA**²

¹PG Scholar, Dept of ECE, Netaji Institute of Engineering & Technology, Toopranpet, Nalgonda, AP-INDIA, E-mail: sujithareddy26@gmail.com.

²Assoc Prof, Dept of ECE, Netaji Institute of Engineering & Technology, Toopranpet, Nalgonda, AP-INDIA.

**Abstract:** Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. From the structure of the CSLA, it is clear that there is scope for reducing the area and power consumption in the CSLA. This work uses a simple and efficient gate-level modification to significantly reduce the area and power of the CSLA. Based on this modification 8-, 16-, 32-, 64-b, 128-b square-root CSLA (SQRT CSLA) architecture have been developed and compared with the regular SQRT CSLA architecture. The proposed design has reduced area and power as compared with the regular SQRT CSLA with only a slight increase in the delay. This work evaluates the performance of the proposed designs in terms of delay, area, power, and their products by hand with logical effort and through custom design and layout in 0.18- m CMOS process technology. The results analysis shows that the proposed CSLA structure is better than the regular SQRT CSLA.

**Keywords:** Application-specific integrated circuit (ASIC), area-efficient, CSLA, low power.

**I. INTRODUCTION**

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 [2]. 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 [3]–[5]. The main advantage of this BEC logic comes from the lesser number of logic gates than the n-bit Full Adder (FA) structure. The details of the BEC logic are discussed in Section III. This brief is structured as follows. Section II deals with the Area-Efficient Carry Select. Section III presents the Simulation Comparisons. The SQRT CSLA has been chosen for comparison with the proposed design as it has a more balanced delay, and requires lower power and area [6], [7]. The delay and area evaluation methodology of the regular and modified SQRT CSLA are presented in Sections IV. The ASIC implementation details and results are analyzed. Finally, the work is concluded in Section V.
II. AREA-EFFICIENT CARRY SELECT ADDER

The carry ripple adder is constructed by cascading each single-bit full-adder [2]. In the carry ripple adder, each full-adder starts its computation till previous carry-out signal is ready. Therefore, the critical path delay in a carry ripple adder is determined by its carry-out propagation path. The critical path is N-bit carry propagation path in the full-adders. As the bit number N increases, the delay time of carry ripple adder will increase accordingly in a linear way.

In order to improve the shortcoming of carry ripple adder to remove the linear dependency between computation delay time and input word length, carry select adder is presented [3]. The carry select adder divides the carry ripple adder into M parts, while each part consists of a duplicated (N/M)-bit carry ripple adder pair. This duplicated carry ripple adder pair is to anticipate both possible carry input values, where one carry ripple adder is calculated as carry input value is logic “0” and another carry ripple adder is calculated as carry input value is logic “1”. When the actual carry input is ready, either the result of carry “0” path or the result of carry “1” path is selected by the multiplexer according to its carry input value. To anticipate both possible carry input values in advance, the start of each M part carry ripple adder pair no longer need to wait for the coming of previous carry input. As a result, each M part carry ripple adder pair in the carry select adder can compute in parallel. In this way, the critical path of N-bit adder can be greatly reduced and it is shown in fig.4.

Fig. 3. 4-b BEC with 8:4 mux.

In the conventional N-bit carry ripple adder design, the critical path is N-bit carry propagation path plus one summation generation stage. Alternatively, the critical path is (N/M)-bit carry propagation path plus M stage multiplexer with one summation generation stage in the N-bit carry select adder. Since M is much smaller than N and delay in the multiplexer is smaller than that in the full adder, the computation delay in the carry select adder is much shorter than that in the carry ripple adder. However, implementing the adder with duplicated carry generation circuit costs almost twice hardware and twice power consumption as compared with the carry ripple adder.

Therefore, in this paper, we proposed an area-efficient carry select adder by sharing the common Boolean logic term to remove the duplicated adder cells in the conventional carry select adder. In this way, we can save many transistor counts and achieve a lower PDP. As compared with the conventional carry select adder, our speed is a little slower since the parallel path in our design is shorter. However, we can achieve lower area, lower power consumption, and lower PDP. As compared with the carry ripple adder, our speed can be faster because some of the parallel architecture in the conventional carry select adder is retained. The delay time in our proposed adder design is also proportional to the bit number N; however, the delay time of multiplexer is shorter than that of full adder. Consequently, our area-efficient adder can perform with nearly the same transistor count, nearly the same power consumption, but with faster speed and lower PDP as compared with the carry ripple adder.

III. SIMULATION COMPARISONS

We compare the circuit performance with three different architectures, 32-bit carry ripple adder, 32-bit carry select adder, and 32-bit area-efficient carry select adder that is proposed in this paper. As for the transistor count, the transistor count of our proposed area-efficient carry select adder could be reduced to be very close to that of carry ripple adder; however, the transistor count in the conventional carry select adder is nearly double as compared with the proposed design. This result shows that sharing common Boolean logic term could indeed achieve a superior performance in aspect of transistor count.

The area-efficient carry select adder can also achieve an outstanding performance in power consumption. Power consumption can be greatly saved in our proposed area-efficient carry select adder because we only need one XOR gate and one INV gate in each summation operation as well as one AND gate and one OR gate in each carry-out operation after logic simplification and sharing partial circuit. Because of hardware sharing, we can also significantly reduce the occurring chance of glitch. Besides, the improvement of power sharing can be more obvious as the input bit number increases. We simulated the power consumption in the proposed area-efficient adder and the conventional carry select adder with 4, 8, 16, and 32-bit respectively in tsmc 0.18um CMOS technology. The power consumption difference...
Implementation of Carry Select Adder for Low-Power and Area-Efficient Architecture

between these two designs is small in the case of 4-bit input word length. Since the conventional carry select adder consists of the duplicated adder cells to prepare both the possible output values for the corresponding carry input values in advance. It not only needs larger hardware area, but also generates more glitch signals because of propagation path difference.

Therefore, as the input bit number increases, the slope of power consumption increase in the conventional carry select adder would be larger than that in our proposed design. As the input bit number of the conventional carry select adder increases to 32-bit, the power consumption in the conventional carry select adder will be 3.3 times larger than that in our proposed area-efficient carry select adder.

The conventional carry select adder performs better in terms of speed. The delay of our proposed design increases slightly because of logic circuit sharing sacrifices the length of parallel path. However, the proposed area-efficient carry select adder retains partial parallel computation architecture as the conventional carry select adder design; the delay increment of the proposed design is similar to that in the conventional design as the input bit number increases. We also simulated the delay performance in the proposed area-efficient adder and conventional carry select adder with 4, 8, 16, and 32-bit respectively as given in fig.5 and fig.6.

Fig.5. The 16-Bit Carry Select Adder Is Divided By The Carry Ripple Adder Into 4 Parts, While Each Part Consists Of A Duplicated 4-Bit Carry Ripple Adder Pair.

The delay difference existing between these two designs is mainly come from the length difference in their parallel paths. In the conventional carry select adder, it divided N bits into M blocks; however, our proposed design divided every single bit as individual block. In other words, we still retain N blocks in the N bits adder. Such arrangement will lead to some speed sacrifice. We further analyze the Power-Delay-Product as shown.

Fig.6. 5-Bit Carry Select Adder

The proposed area-efficient carry select adder is constructed by sharing the common Boolean logic term in summation generation. We can find out that the PDP of our proposed design is smaller as compare with the conventional carry select adder and carry ripple adder design. The difference of PDP between these three designs is small in the case of the smaller input bit number.

However, as the input bits increases, the slope of power consumption increment in the conventional carry select adder would be larger than that of the proposed design. Our proposed design can compute the addition function more efficiently by means of logic circuit sharing and partial parallel computation architecture retaining; therefore, the power saving ratio in our design would be much higher than the ratio of speed sacrifice. Simplifying the carry select adder through logic simplification and partial logic circuit sharing can make the carry select adder more area-efficient and more power-efficient. The performance index of transistor count, power, delay, and PDP are summarized.

As compared with the carry ripple adder, operation speed in our proposed carry select adder can be much faster; however, transistor count and power consumption only increase slightly. In the case of a 32-bit adder, the transistor count in the carry ripple adder is 896. The transistor count in our proposed area-efficient carry select adder is 960, which only increases 7%. However, the transistor count in the conventional carry select adder is 1974, which increases more than twice. In terms of power consumption, we can save much power through removal of redundant logic and redundant signal switching by means of sharing common Boolean logic term.
As compared with the conventional carry select adder, we can save 70% power. Relative to the carry ripple adder, we only increase 2% power. As a result, our proposed area-efficient carry select adder can perform the lowest PDP, which is only 60% of conventional carry select adder and 66% of carry ripple adder, respectively.

**IV. ASIC IMPLEMENTATION RESULTS**

The design proposed in this paper has been developed using Verilog-HDL and synthesized in Cadence RTL compiler using typical libraries of TSMC 0.18 um technology. The synthesized Verilog net list and their respective design constraints file (SDC) are imported to Cadence SoC Encounter and are used to generate automated layout from standard cells and placement and routing [8]. Parasitic extraction is performed using Encounter’s Native RC extraction tool and the extracted...
Implementation of Carry Select Adder for Low-Power and Area-Efficient Architecture

TABLE I
Delay and Area Count of Modified Sqrt CSLA

<table>
<thead>
<tr>
<th>Group</th>
<th>Delay (ns)</th>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group2</td>
<td>13</td>
<td>43</td>
</tr>
<tr>
<td>Group3</td>
<td>16</td>
<td>61</td>
</tr>
<tr>
<td>Group4</td>
<td>19</td>
<td>84</td>
</tr>
<tr>
<td>Group5</td>
<td>22</td>
<td>107</td>
</tr>
</tbody>
</table>

Table II exhibits the simulation results of both the CSLA structures in terms of delay, area and power. The area indicates the total cell area of the design and the total power is sum of the leakage power, internal power and switching power. The percentage reduction in the cell area, total power, power-delay product and the area–delay product as function of the bit size are shown in Fig. 8(a). Also plotted is the percentage delay overhead in Fig. 8(b).

It is clear that the area of the 8-, 16-, 32-, 64- and 128-b proposed SQRT CSLA is reduced by 9.7%, 15%, 16.7%, and 17.4%, respectively. The total power consumed shows a similar trend of increasing reduction in power consumption 7.6%, 10.56%, 13.63%, and 15.46 % with the bit size. Interestingly, the delay overhead also exhibits a similarly decreasing trend with bit size. The delay overhead for the 8, 16, and 32-b is 14%, 9.8%, and 6.7% respectively, whereas for the 64-b it reduces to only 3.76%. The power–delay product of the proposed 8-b is higher than that of the regular SQRT CSLA by 5.2% and the area-delay product is lower by 2.9%. However, the power-delay product of the proposed 16-b SQRT CSLA reduces by 1.76% and for the 32-b and 64-b by as much as 8.18%, and 12.28% respectively. Similarly the area-

parasitic RC (SPF format) is back annotated to Common Timing Engine in Encounter platform for static timing analysis. For each word size of the adder, the same value changed dump (VCD) file is generated for all possible input conditions and imported the same to Cadence Encounter Power Analysis to perform the power simulations. The similar design flow is followed for both the regular and modified SQRT CSLA.
delay product of the proposed design for 16-, 32-, and 64-bit is also reduced by 6.7%, 11%, and 14.4% respectively.

Fig. 10. Synthesis Report for 128 bit Regular CSLA

Fig. 11. Simulation Result for 128-bit modified CSLA

Fig. 12. Simulation Result for 128-bit Regular CSLA
Fig. 13. Synthesis Report for 128 bit Modified CSLA

Fig. 14. Device Utilisation Summary for 128 bit Regular CSLA

Fig. 15. Device Utilisation Summary for 128 bit Regular CSLA
V. CONCLUSION
A simple approach is proposed in this paper to reduce the area and power of SQRT CSLA architecture. The reduced number of gates of this work offers the great advantage in the reduction of area and also the total power. The compared results show that the modified SQRT CSLA has a slightly larger delay (only 3.76%), but the area and power of the 128-b modified SQRT CSLA are significantly reduced by 17.4% and 15.4% respectively. The power-delay product and also the area-delay product of the proposed design show a decrease for 16-, 32-, 64-, 128-b sizes which indicates the success of the method and not a mere tradeoff of delay for power and area. The modified CSLA architecture is therefore, low area, low power, simple and efficient for VLSI hardware implementation. It would be interesting to test the design of the modified 128-b SQRT CSLA.

VI. REFERENCES


