

Area Efficient and High Speed ALU Using 64 Bit Modified Square Root Carry Select Adder

¹N. Pranavi.,M.Tech, S.V. Engineering College for Women

²P.Anil Kumar, Research Scholar, S V University College of Engineering

Abstract: In digital computer, an Arithmetic logic unit (ALU) is a crucial combinational circuit that executes arithmetic and logical functions. Parallel adder in ALU plays an essential role, however the speed of addition is limited by the time required to transmit a carry through the adder. For area-efficient and high performance computer applications, 64-Bit modified Square Root Carry Select Adder (SQRT CSLA) based ALU is proposed. This paper delivers the design and implementation of 64-Bit modified SQRT CSLA based ALU. Compares it with the regular SQRT CSLA in terms of area and speed. The design entry can be described in Verilog .For simulate and synthesize use Altera QUARTUS-II 9.1.The results analysis shows that the proposed SQRT CSLA structure is better than the regular SQRT CSLA.

I. INTRODUCTION

Reduced area and high performance data path logic systems are the important areas of research in VLSI system design. High-speed addition and multiplication has always been a basic requirement of high-performance processors and systems. In digital adders, the speed of addition is limited by the propagation delay of a carry through the adder. The sum for each bit position in an regular adder is produced sequentially only after the previous bit position has been summed and a carry propagated into the next position. Some adder designs available (Ripple Carry Adder, Carry Look Ahead Adder, Carry Save Adder, Carry Skip Adder) which have its own advantages and disadvantages. Many times speed limitation in any adder is in the generation of carries and many authors considered the addition problem. To solve The carry propagation delay CSLA is developed which drastically reduces the area and delay to a great extent.

The CSLA is used in many computational systems design to mitigate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. It uses independent ripple carry adders (for $C_{in}=0$ and $C_{in}=1$) to produce the resultant sum. However, the Regular CSLA is area and speed inefficient because it requires multiple pairs of Ripple Carry Adders (RCA) to produce partial sum and carry by considering carry input. The final sum and carry are selected by the multiplexers. Due to the use of two independent RCA the area will increase which results an increase in delay. To surmount the above problem, the basic idea of the proposed work is to use n-bit binary to excess-1 code converters (BEC) to enhance the speed of addition. This logic can be replaced in RCA for $C_{in}=1$ to further improves the speed and thus reduces the delay. Using Binary to Excess-1 Converter (BEC) instead of RCA in the regular CSLA will establish lower area, delay which speeds up the addition operation. The main benefit of this BEC logic comes from the lesser number of logic gates than the Full Adder (FA) structure because the number of gates used will be reduced.

The basic idea is to use Binary to Excess-1 Converter (BEC) in the regular CSLA to get lower area and increased speed of operation. This logic is replaced in RCA with $C_{in}=1$. This logic can be implemented for different bits which are used in the modified design. The main use of this BEC[3] logic comes from the fact that it uses fewer number of logic gates than the n-bit Full Adder (FA) structure. As stated above the main idea of this work is to use BEC instead of the RCA with $C_{in}=1$ in order to reduce the area and increase the speed of operation in the regular CSLA to obtain modified CSLA. To substitute the n-bit RCA, an n+1 bit BEC logic is needed. The logic diagram of a 6-bit BEC are shown in Figure1

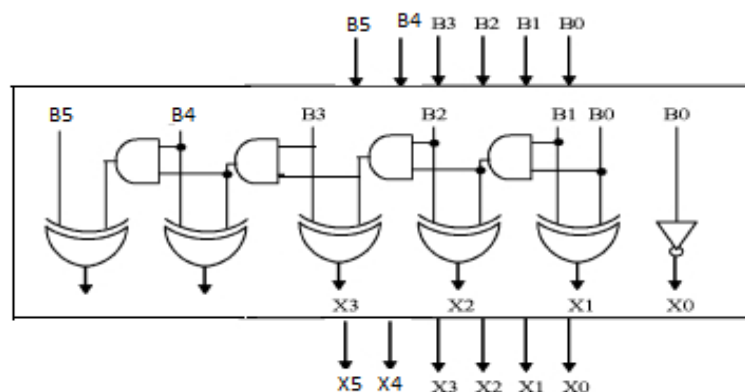


Figure 1: 6-binary to excess-1 converter

The Boolean expressions for the 6-bit BEC logic are expressed below

$$X0 = \sim B0$$

$$X1 = B0 \wedge B1$$

$$X2 = B2 \wedge (B0 \wedge B1).$$

$$X3 = B3 \wedge (B0 \wedge B1 \wedge B2).$$

$$X4 = B4 \wedge (B0 \wedge B1 \wedge B2 \wedge B3).$$

$$X5 = B5 \wedge (B0 \wedge B1 \wedge B2 \wedge B3 \wedge B4).$$

II. EXISTED METHOD

Figure2 shows the Regular structure of 64-bit Sqrt CSLA. It includes many ripple carry adders of different sizes which are divided into groups. Group 0 contains 2-bit RCA[2] which consist only one ripple carry adder which adds the input bits and the input carry and results to sum [1:0] and the carry out. The carry out of the Group 0 which acts as the selection input to multiplexer which is in group 1, selects the result from the related RCA (Cin=0) or RCA (Cin=1). Similarly the rest of the groups will be selected depending on the Cout from the previous groups.

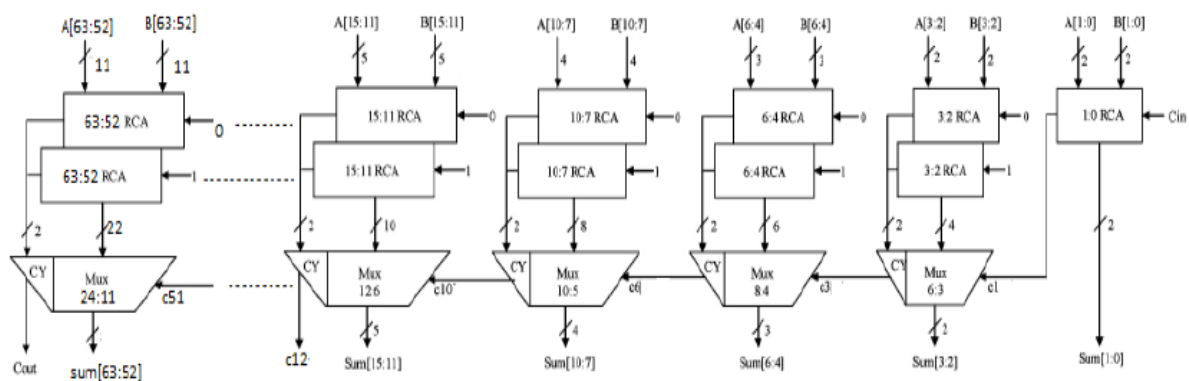


Figure2: Architecture of 64-Bit Regular Sqrt CSLA

In Regular CSLA[1], there is only one RCA to perform the addition of the least significant bits [1:0]. The remaining bits the addition is performed with two RCAs corresponding to the one assuming a carry-in of 0, the other a carry-in of 1 within a group. In a group, there are two RCAs that receive the same data inputs but different Cin. The upper adder has a carry-in of 0, the lower adder a carry-in of 1. The actual Cin from the preceding sector selects one of the two RCAs. That is, as shown in the Fig.2, if the carry-in is 0, the sum and carry-out of the upper RCA is selected, and if the carry-in is 1, the sum and carry-out of the lower RCA is selected. For this Regular CSLA architecture[4,5], the implementation code, for the Full Adders and Multiplexers of different sizes (6:3, 8:4, 10:5 up to 24:11) were designed initially. The regular 64-bit implemented by calling the ripple carry adders and all multiplexers.

III. ARCHITECTURE OF PROPOSED MODIFIED 64-BIT Sqrt CSLA

This architecture is same as regular 64-bit Sqrt CSLA, the only thing to change is that, we replace RCA with Cin=1 among the two available RCAs in a group with a BEC. This BEC posses similar operation that it can perform as if it replaced RCA with Cin=1. Fig 3 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 different groups of variable sizes of bits with each group having the ripple carry adders, BEC and concern multiplexer. As shown in the Fig.3, 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 multiplexer 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 multiplexer is earlier than the sum of RCA[6,7] and BEC. For remaining groups the selection input arrival is later than the RCA and BEC. Thus, the sum1 and c1 (output from multiplexer) are depending on multiplexer and results computed by RCA and BEC respectively. The sum2 depends on c1 and multiplexer. For the remaining parts the arrival time of multiplexer selection input is always greater than the arrival time of data inputs from the BEC's. Thus, the delay of the remaining multiplexer depends on the arrival time of multiplexer selection input and the multiplexer delay. In this Modified CSLA architecture, the implementation code for Full Adder and Multiplexers of 6:3, 8:4, 10:5, 12:6, 12:6, 22:11, 34:17 and 13:17 were designed. The design used different BEC with sizes were 3,4,5,6,6,,11,17, and 17 respectively and corresponding ripple carry adder were then designed with 2, 2,3, 4, 5,5,10,16 and 16 bit respectively

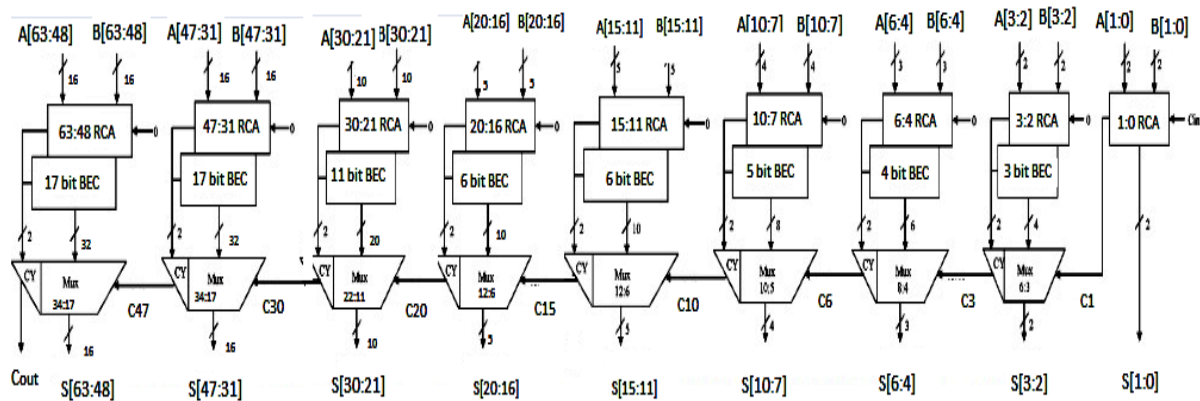


Figure3: Architecture of 64-Bit Modified Sqrt CSLA

IV. SIMULATION RESULTS

The implemented design in this work has been simulated using Verilog-HDL. The adders and ALU are designed and simulated using Altera QUARTUS-II 9.1. All the V files (Regular and Modified) are also simulated and synthesized in Altera QUARTUS-II 9.1[8] and corresponding results are compared.

Table 1 shows the comparison of regular and modified CSLA of various bits which includes Delay and area comparisons. From the table it is clear that the delay and combinational ALUTs (area) decreases for 64-bit modified method when compared with regular method. 64 bit ALU also gives better results if it uses 64 bit modified Sqrt CSLA.

Table 1: Comparison between Different Regular and Modified Sqrt Adder

S No	Sqrt Adder		Combinational ALTUs	Dedicated logic	DSP Block 9 bit elements	Time Delay(ns)
1	32 Bit	Regular	84	-	-	15.669
		Modified	66	-	-	12.355
2	64 Bit	Regular	140	-	-	23.325
		Modified	132	-	-	13.592
3	64Bit ALU	Modified	354	64	24	13.819

Flow Status	Successful - Mon Jun 19 17:25:47 2017
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SQRTCSLA
Top-level Entity Name	sqrt_adder2_64
Family	Stratix II
Met timing requirements	Yes
Logic utilization	1 %
Combinational ALUTs	140 / 12,480 (1 %)
Dedicated logic registers	0 / 12,480 (0 %)
Total registers	0
Total pins	194 / 343 (57 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	0 / 96 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Figure4: Design summary for proposed 64 bit square root adder

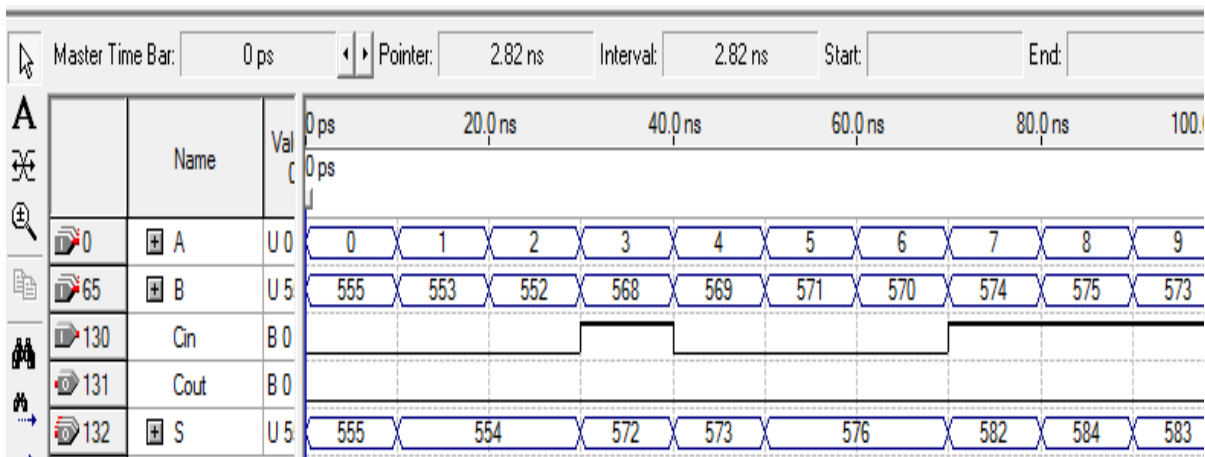


Figure5: Simulation results for proposed 64 bit square root adder

Flow Status	Successful - Mon Jun 19 17:29:39 2017
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SQRTCSLA
Top-level Entity Name	alu4
Family	Stratix II
Met timing requirements	Yes
Logic utilization	4 %
Combinational ALUTs	354 / 12,480 (3 %)
Dedicated logic registers	64 / 12,480 (< 1 %)
Total registers	64
Total pins	198 / 343 (58 %)
Total virtual pins	0
Total block memory bits	0 / 419,328 (0 %)
DSP block 9-bit elements	24 / 96 (25 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP2S15F484C3
Timing Models	Final

Figure6: Design summary for proposed 64 bit ALU

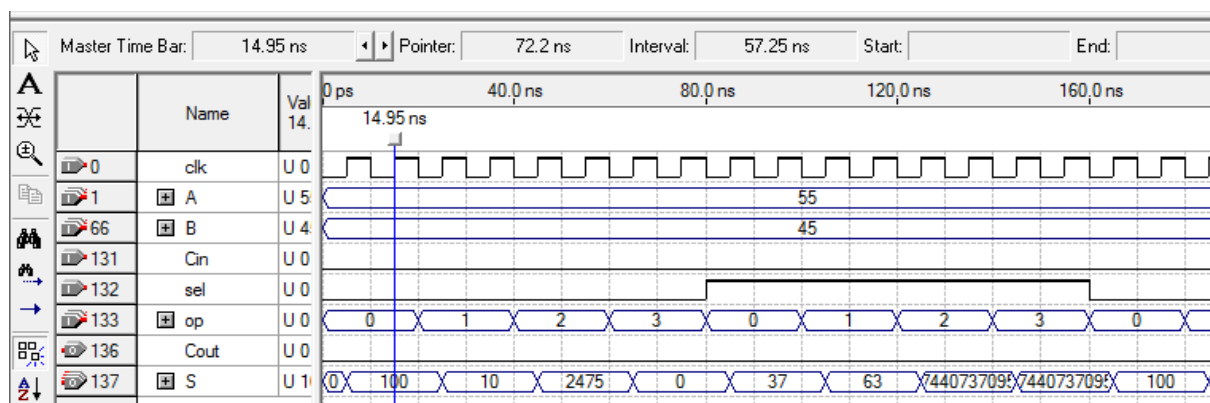


Figure7: Simulation Results for proposed 64 bit ALU

V. ACKNOWLEDGMENT

N.Pranavi would like to thank Mr.B.Ravi Kumar, Research Scholar, ECE Department who had been guiding throughout the project and supporting me in giving technical information about the paper and motivating me to complete the work effectively and successfully.

VI. CONCLUSION

An efficient approach is proposed in this paper to reduce the area and delay of SQR CSLA architecture. The reduction in the number of gates is obtained by simply replacing the RCA with BEC in the structure. The compared results shows that the modified SQR CSLA has a slightly larger area for lower order bits which further reduces for higher order bits. The delay is reduced to a great extent with the modified SQR CSLA. Thus the results shows that using modified method the area and delay will decrease thus leads to good alternative for adder implementation for many processors. The modified CSLA architecture is therefore low area and high speed approaches for VLSI hardware implementation. The results evident that 64 bit modified SQR CSLA is used 140 combinational ALUs and 64 bit ALU is used 354 combinational ALUs and 64 dedicated logic.

REFERENCES

- [1] O. J. Bedrij, "Carry-select," IRE Trans. on Electron. Comput., pp. 340-344, 1962.
- [2] B. Ramkumar and H. M. Kittur, "Low-power and area efficient carry-select adder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 2, pp. 371-375, Feb. 2012.
- [3] M. Morris Mano, Digital Logic And Logic Design., 3rd ed., Pearson Edu. Inc., 2011.
- [4] B. Parahmi, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.
- [5] 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.
- [6] J. M. Rabaey, Digital integrated circuits—A Design perspective. Upper Saddle River, NJ: Prentice-Hall, 2001.
- [7] T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripple carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, 1998
- [8] Ronald J. Tocci, Neal s. Widmer and Gregory L. Moss "Digital systems- principles and applications"