IJMTST | Volume: 2 | Issue: 04 | April 2016 | ISSN: 2455-3778

VLSI Implementation of 32-Bit Unsigned Multiplier Using CSLA & CLAA

M.Rama Krishna[1] | P.V.V.Satyanarayana[2] | P.V.R.Naveen Kumar[3] | K.Pawan Kalyan[4]

[1]Assistant Professor, Department of ECE, Ramachandra College of Engineering, Eluru, India.
[2,3,4]B.Tech, Students, Department of ECE, Ramachandra College of Engineering, Eluru, India.

ABSTRACT

In this project we are going to compare the performance of different adders implemented to the multipliers based on area and time needed for calculation. The CLAA based multiplier uses the delay time of 99ns for performing multiplication operation where as in CSLA based multiplier also uses nearly the same delay time for multiplication operation. But the area needed for CLAA multiplier is reduced to 31 % by the CSLA based multiplier to complete the multiplication operation.

KEYWORDS: *CLAA, CSLA, Delay, Area, Array Multiplier, VHDL Modeling & Simulation.*

I. Introduction

To humans, decimal numbers are easy to comprehend and implement for performing arithmetic. However, in digital systems, such as a microprocessor, DSP (Digital Signal Processor) or ASIC (Application-Specific Integrated Circuit), binary numbers are more pragmatic for a given computation. This occurs because binary
values are optimally efficient at representing many values.
Binary adders are one of the most essential logic elements within a digital system. In addition, binary adders are also helpful in units other than Arithmetic Logic Units (ALU), such as multipliers, dividers and memory addressing. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, hence, help improve the performance of the entire system.

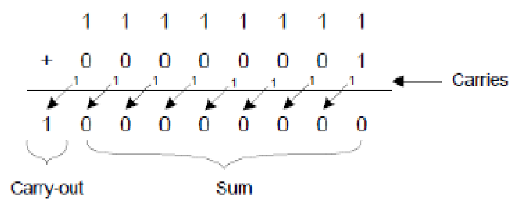


Figure 1 Binary adder example

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. As such, extensive research continues to be focused on improving the power delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Reconfigurable logic such as Field Programmable Gate Arrays (FPGAs) has been gaining in popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for many practical designs involving mobile DSP and telecommunications applications and a significant reduction in development time and cost over Application Specific Integrated Circuit (ASIC) designs.

Ripple-Carry Adders (RCA):

The simplest way of doing binary addition is to connect the carry-out from the previous bit to the next bit's carry-in. Each bit takes carry-in as one of the inputs and outputs sum and carry-out bit and hence the name ripple-carry adder. This type of adders is built by cascading 1-bit full adders. A 4-bit ripple-carry adder is shown in Figure 2.3. Each trapezoidal symbol represents a single-bit full adder. At the top of the figure, the carry is rippled through the adder from C0 to C4.

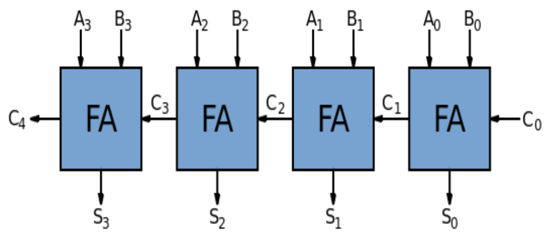


Figure 2 Ripple-Carry Adders.

Carry-Look-ahead Adders (CLA):

The carry-chain can also be accelerated with carry generate/propagate logic. Carry-look ahead adders employ the carry generate/propagate in groups to generate carry for the next block. In other words, digital logic is used to calculate all the carries at once. When building a CLA, a reduced version of full adder, which is called a reduced full adder (RFA) is utilized. Figure 2.4 shows the block diagram for an RFA. The carry generate/propagate signals gi/pi feed to carry-look ahead generator (CLG) for carry inputs to RFA.

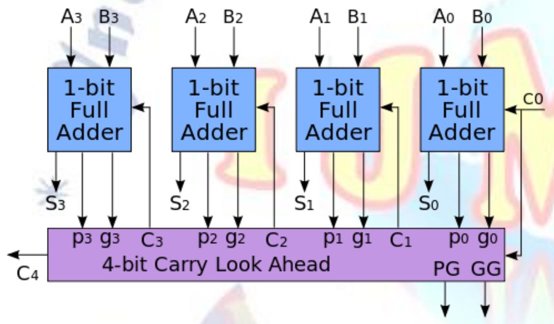


Figure 3 Carry Look Ahead Adder.

Let Gi is the carry generate function and Pi be the carry propagate function, Then we can rewrite the carry function as follows:

Gi = Ai· Bi. (1)

Pi = (Ai xor Bi). (2)

Si = Pi xor Ci. (3)

Ci+l= Gi + Pi.Ci. (4)

Thus, for 4-bit adder, we can compute the carry for all the stages as shown below:

C1 = Go + Po.Co. (5)

C2 = G1 + P1.C1 = G1 + P1.GO + P1.PO.CO (6)

C3 = G2 + P2.C2= G2 + P2.G1 + P2.P1 .GO + P2.P1.PO.CO (7)

C4 = G3 + P3.C3 = G3 + P3.G2 + P3.P2 .G1 + P3.P2.P1.GO + P3.P2.Pl.PO.CO (8)

In general, we can write:

The sum function:

$SUM_i = A_i$ xor B_i xor $C_i = P_i$ xor C_i

The carry function:

Ci+l= Gi + Pi.Ci.

II. Carry Select Adder (CSLA)

The concept of CSLA is to compute alternative results in parallel and subsequently selecting the correct result with single or multiple stage hierarchical techniques. In CSLA both sum and carry bits are calculated for two alternatives Cin=O and 1. Once Cin is delivered, the correct computation is chosen using a mux to produce the desired output. Instead of waiting for Cin to calculate the sum, the sum is correctly output as soon as Cin gets there. The time taken to compute the sum is then avoided which results in good improvement in speed.

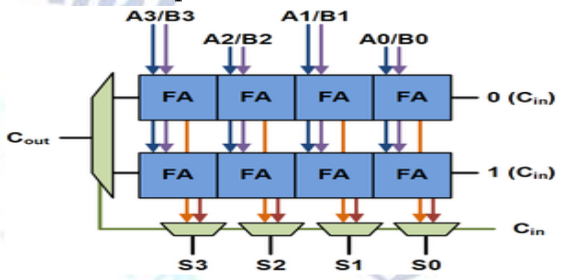


Figure 4 Carry Select Adder.

In general, we can write the algorithm as:

If Carry in =1, then the sum and carry out are given by,

Sum (i) =a (i) xor b (i) xor '1'. (11)

Carry (i+ 1) = (a (i) and b (i)) or (b (i) or a (i)). (12)

If Carry in =0, then the sum and carry out are given by,

Sum (i) = a (i) xor b (i). (13)

Carry (i+ 1) = (a (i) and b (i)). (14)

The sum function:

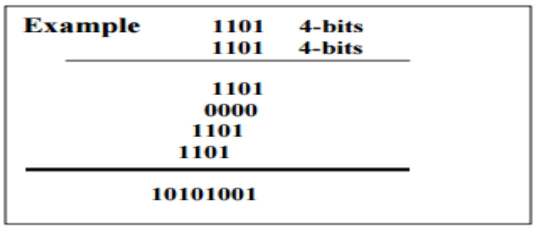
$S_i = C_i S_i^0 + C_i S_i^1$. (15)

The carry function:

$C_{i+1} = C_i C_{i+1}^0 + C_i C_{i+1}^1$. (16)

MULTIPLIER:

Multiplication involves the generation of partial products, one for each digit in the multiplier, as in Figure .These partial products are then summed to produce the final product. The multiplication of two n-bit binary integers results in a product of up to 2n bits length.



```
Example      1101    4-bits
             1101    4-bits
        ----------------------
             1101
            0000
           1101
          1101
     ---------------------------
          10101001
```

On comparison with the carry look-ahead adder (CLAA) based multiplier the area of calculation of

Volume: 2 | Issue: 04 | April 2016 | ISSN: 2455-3778

the different carry select adder (CSLA) based multiplier is smaller and better with nearly same delay time. Here we are dealing with the comparison in the bit range of n*n (32*32) as input and 2n (64) bit output.

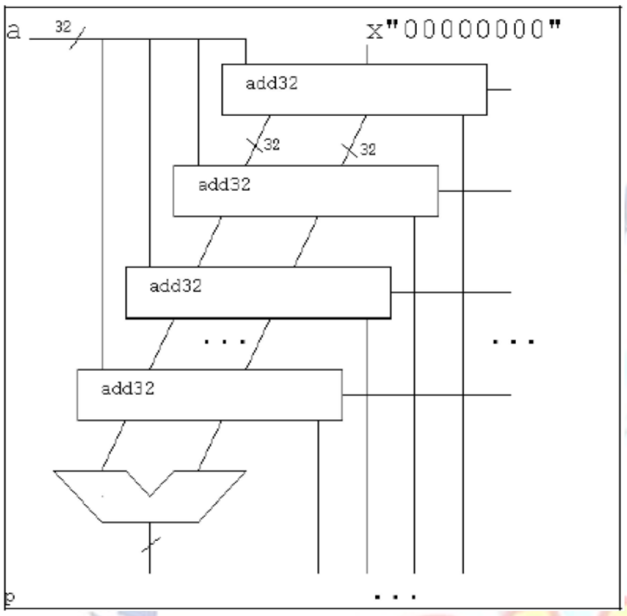


Figure 5 Multiplier.

III. Simulation Results

The simulation results of CLAA and CSLA based multipliers are as shown in below figures

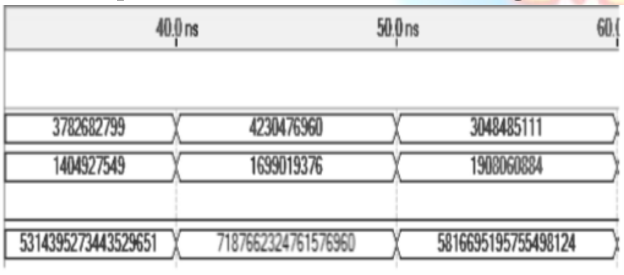


Figure 6 Waveform for a CLAA based Multiplier.

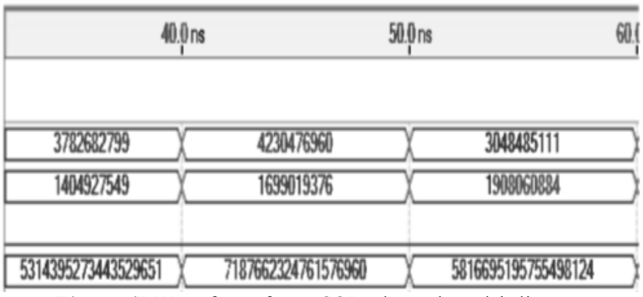
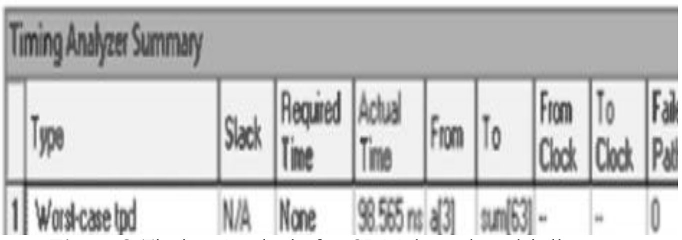


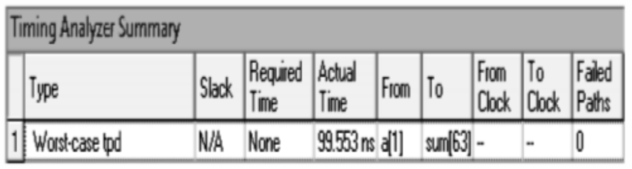
Figure 7 Waveform for a CSLA based multiplier.



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Fai Pat
1	Worst-case tpd	N/A	None	98.565 ns	a[3]	sum[63]	--	--	0

Figure8 Timing Analysis for CLAA based multiplier.



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	99.553 ns	a[1]	sum[63]	--	--	0

Figure 9 Timing Analysis for CSLA based Multiplier

IV. Analysis Table

In this analysis table shown in figurel4, the delay time is nearly same, the area and the area delay product of CSLA based multiplier is reduced to 31 % when compared to CLAA based multiplier.

Multiplier type	Delay(ns)	Area	Delay Area Product
CLAA Based Multiplier	98.5	2957 Logic Cells	2912645
CSLA Based Multiplier	99.5	2039 Logic Cells	2028805

V. Conclusion and Future Work

A design and implementation of a VHDL-based 32bit unsigned multiplier with CLAA and CSLA was presented. VHDL, a Very High Speed Integrated Circuit Hardware Description Language, was used to model and simulate our multiplier. Using CSLA improves the overall performance of the multiplier. Thus a 31 % area delay product reduction is possible with the use of the CSLA based 32 bit unsigned parallel multiplier than CLAA based 32 bit unsigned parallel multiplier.

This 32 bit multiplier can be further extended to 64 bit multiplier and 128 bit multiplier using the proposed method for multiplication operation can be done as future work.

References

[1] Muhammad Ali Akbar and Jeong-A Lee, *Senior Member, IEEE*Comments on “Self-Checking Carry-Select Adder Design Based onTwo-Rail Encoding”IEEE transactions on circuits and systems : regular papers, vol. 61, no. 7, July 2014

[2] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, “Self-checking carryselectadder design based on two-rail encoding,” *IEEE Trans. CircuitsSyst. I, Reg. Papers*, vol. 54, no. 12, pp. 2696–2705, Dec. 2007.

[3] M. Alioto, G. Palumbo, and M. Poli, “Optimized design of parallelcarry-select adders,” *Integration, the VLSI J.*, vol. 44, no. 1, pp. 62–74,Jan. 2011.

[4] H. Belgacem, K. Chiraz, and T. Rached, “A novel differentialXOR-based self-checking adder,” *Int. J. Electron.*, vol. 99, no. 9, pp.1239–1261, Apr. 2012.

[5] Y. S.Wang, M. H. Hsieh, J. C.-M. Li, and C. C.-P. Chen, “An at-speedtest technique for high-speed high-order adder by a 6.4-GHz 64-bitdomino adder

example," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol.59, no. 8, pp. 1644–1655, Aug. 2012.

[6] Sertbas and R.S. Ozbey, "A performance analysis of classified binary adder architectures and the VHDL simulations", J Elect. Electron. Eng., Istanbul, Turkey, vol. 4, pp. 1025-1030,2004.

[7] O. J. Bedrij, "Carry-select adder," *IRE Trans. Electron. Comput.*, pp.340–344, 1962.

[8] B. Ramkumar, H.M. Kittur, and P. M. Kannan, "ASIC implementationof modified faster carry save adder," *Eur. J. Sci. Res.*, vol. 42, no. 1, pp.53–58, 2010.

[9] T. Y. Ceiang and M. J. Hsiao, "Carry-select adder using single ripplecarry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.

[10] 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.

[11] J. M. Rabaey, *Digtal Integrated Circuits—A Design Perspective*.Upper Saddle River, NJ: Prentice-Hall, 2001.