ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementations of Low-Power and Area-Efficient Carry Select Adder

B.Sindhurmai
M.Tech Student,
Department of ECE,
KITS for Women's, kodad, T.S, India.

Mr. B. Naresh Reddy
Associate Professor,
Department of ECE,
KITS for Women's, kodad, T.S, India.

ABSTRACT:

Carry select adder (CSLA) is known to be the fastest adder among the conventional adder structures. Due to the rapidly growing mobile industry not only the faster arithmetic unit but also less area and low power arithmetic units are needed. The modified CSLA architecture has developed using Binary to Excess-1 converter (BEC). This paper proposes an efficient method which replaces the BEC using D latch. Experimental results are compared and the result analysis shows that the proposed architecture achieves the three folded advantages in terms of area, delay and power

Keywords:

area efficient, CSLA, low power and BEC

1. INTRODUCTION:

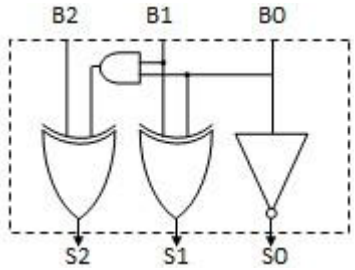
Design of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary 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. The carry-select adder (CSLA) provides a compromise between small area but longer delay ripple carry adder (RCA) and larger area with shorter delay carry look-ahead adder. CSLA uses multiple pairs of ripple carry adder (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 multiplexers. The modified CSLA using BEC has reduced area and power consumption with slight increase in delay. The basic idea of the proposed architecture is that which replaces the BEC by D latch with enable signal. The proposed architecture reduces the area, delay and power. This paper is organized as follows; section III presents the detailed structure and the function of the binary to excess-1 converter logic. Section IV and section V explains the regular and modified CSLA respectively. Section VI deals with the proposed architecture. Results are analyzed in the section VII. Section VIII concludes.

2. METHODOLOGY:

Bedriji 1962 proposes [3] that the problem of carry propagation delay is overcome by independently generating multiple radix carries and using these carries to select between simultaneously generated sums. Akhilash Tyagi 1993 introduces a scheme to generate carry bits with block carryin 1 from the carries of a block with block carryin 0 [8]. Chang and Hsiao 1998 [4] propose that instead of using dual carry ripple adder a carry select adder scheme using an addone circuit to replace one carry ripple adder. Youngioon Kim and Lee Sup Kim 2001 [6] introduces a multiplexer based add one circuit is proposed to reduce the area with negligible speed penalty. Yajuan He et al 2005 proposed an area efficient square root carry select adder scheme based on a new first zero detection logic [5]. Ramkumar et al 2010 proposed a BEC method to reduce the maximum delay of carry propagation in final stage of carry save adder [2]. Ramkumar and Harish 2011 propose [11] BEC technique which is a simple and efficient gate level modification to significantly reduce the area and power of square root CSLA.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Padma Devi et al 2010 proposed [7] modified carry select adder designed in different stages which reduces the area and power consumption.



A few weeks ago, probably due to my recent Arduino and D-CPU obsessions, I started thinking about with this topic: How do modern computer CPUs add numbers? I took classes on this in school, so I had a basic understanding, but the more I thought about it, the more I realized that my ideas about how this would scale up to 64-bit computers would be too slow to actually work.I started digging around, and even though wikipedia is usually exhaustive (and often inscrutable) about obscure topics, I had reached the edge of the internet. Only context-less names like "Kogge-Stone" and unexplained box diagrams greeted me. I had to do actual research of the 20th-century kind.So come with me over the precipice and learn — in great detail — how to add numbers!I'm going to start out as if you've never taken a class in computer engineering. If you're familiar with the basics of binary addition, skip below to get to the good stuff.

Adding in binary:

For big numbers, addition by hand means starting on the rightmost digit, adding all the digits in the column, and then writing down the units digit and carrying the tens over. In the example below, 8 plus 4 is 12, so we carry the 1, which I've indicated with a precious tiny blue 1 over the left column:

```
 1
 482
+345
----
 827
```

We memorize this in school, but the reason it works is that each column is the same power of ten: 8 tens plus 4 tens is 12 tens. And 12 tens is really 1 hundred and 2 tens, so the 1 hundred is shifted/carried over to the hundreds column.

This works the same in binary, but the digits can only ever be 0 or 1, so the biggest number we can add is 1 plus 1. This would be 2, or "10" in binary (1 two and 0 ones), so there's a carry of 1. In fact, if we have a carry, 1 plus 1 with a carried 1 is 3: "11" (1 two and 1 one). That still only carries a 1, which is convenient, because it means the carry can be represented in binary just like every other digit.

```
 1 1
 0110 (6)
+0111 (7)
-----
 1101 (13)
```

So, to add two binary numbers, we just need to add 3 binary digits (one digit from each of the numbers, plus a possible incoming carry), and produce a sum bit and an outgoing carry bit. We can make a logic table for this:

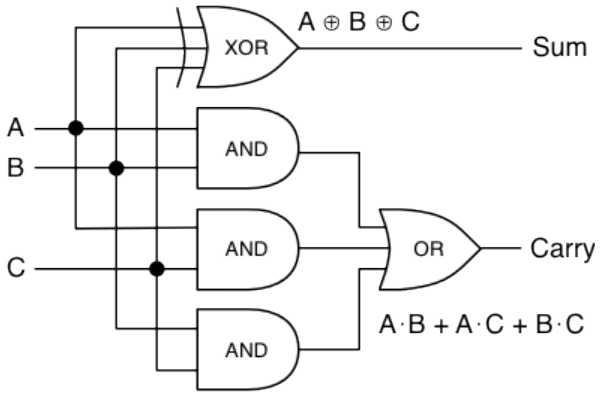
A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum and Carry bits. In logic circuit equations, "+" means OR, means AND, and means XOR. (Programmers usually use "&" to mean AND, and "|" to mean OR, but I think in this case it's important to use the symbols that professional circuit designers use. It gives you a bit more intuition when dealing with logical equations, which will come up later.)One way to think of it is: According to the logic table we just made, the sum should be 1 if there are an odd number of incoming 1s. XOR is the operation that matches odd inputs. And the carry should be 1 if at least two of the incoming digits are 1.

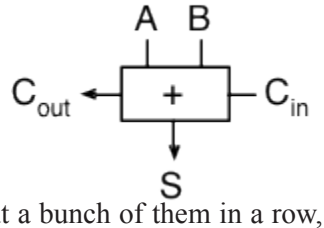
Adding in circuitry:

The most straightforward logic circuit for this is

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

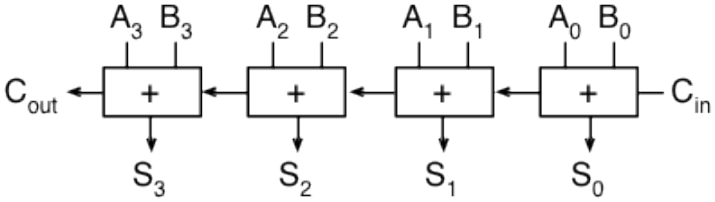


assuming you have a 3-input XOR gate. If you don't, you can just hook two 2-input XOR gates together.Now rename C to Cin, and Carry to Cout, and we have a "full adder" block that can add two binary digits, including an incoming carry, and generate a sum and an outgoing carry.



And if we put a bunch of them in a row, we can add any N-bit numbers together!

Starting along the top, there are four inputs each of A and B, which allows us to add two 4-bit numbers. The rightmost bit, A0, is the "ones", A1 is the "twos", and so on through the "fours" and "eights" (powers of two instead of ten). On the far right, we have a dangling carry-in which we'll just set to zero so that it doesn't matter.



Starting along the top, there are four inputs each of A and B, which allows us to add two 4-bit numbers. The rightmost bit, A0, is the "ones", A1 is the "twos", and so on through the "fours" and "eights" (powers of two instead of ten). On the far right, we have a dangling carry-in which we'll just set to zero so that it doesn't matter. The carry-out from the right-most adder is passed along to the second adder, just like in long addition: any carry from the "ones" is added to the "twos" column. Finally, on the far left, we get an "extra" carry out, because the addition of two 4-bit numbers may require 5 bits.

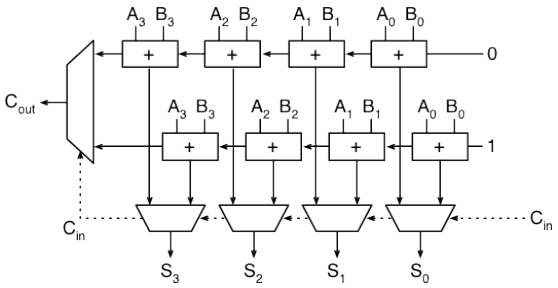
Normally this is considered an "overflow", but the carry-out bit is stored in some kind of status register by every CPU that I know of. It just usually can't be accessed from C or any other language directly, so it gets lost.

Adding in slow-motion:

But here's where the problems come in. Imagine setting up 64 of those adders in a chain, so you could add two 64-bit numbers together. How long would it take? The circuit diagram above shows that each sum goes through one or two gates, and each carry-out goes through two. And the carry-out of one adder becomes the carry-in for the next one. So to generate the entire sum and the final carry-out bit, we need to go through 64 2 = 128 gates.Uh oh.Spoiler alert: No CPU has time to wait for 128 gates to flip in sequence, so no CPU actually adds this way. The problem is that the carry bit needs to "ripple" across each bit, and will only scale linearly with the number of bits being added. We'll need some way to break out of linearity.

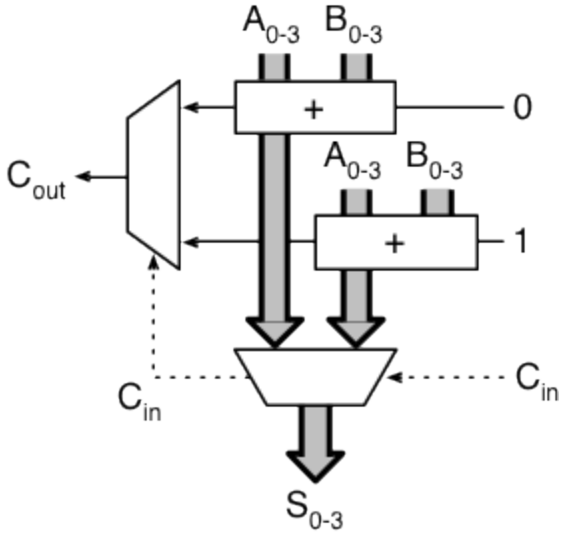
3. IMPLEMENTATION:
Carry-select adder:

The trick that seems most obvious to me — and the only one I thought of before doing research — was apparently invented in 1960 by Sklansky. If you're willing to add more circuitry in exchange for speed, you can put two adders in parallel. One computes the sum with a carry-in of 0, and the other computes with a carry-in of 1. When the real carry-in signal arrives, it selects which addition to use. Here's an example of a 4-bit carry-select adder:

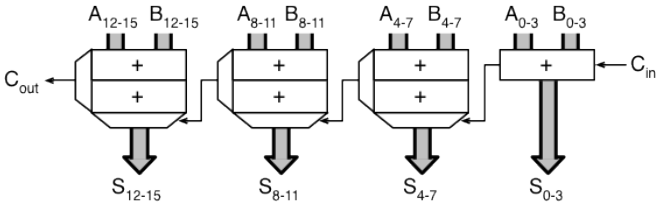


The weird rhombus-shapes are multiplexers, or "mux" for short. A mux takes two inputs and selects one or the other, based on a control signal. In this case, each mux uses the carry-in signal to determine which adder output to use, for each of the four sum bits (along the bottom), and the carry-out bit (on the left).The diagram gets simpler if we make a shortcut box for a series of connected adder units, and draw each group of 4 input or output bits as a thick gray bus:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



Now, for example, to compute the sum of two 16-bit numbers, we can split each number into four chunks of four bits each, and let each of these 4-bit chunks add in parallel. When the adders are finished, the carry-out bit from the lowest (rightmost) adder is used to select which adder's result to use for the next four bits, and then that selected carry-out is used to select the next adder's result, and so on. Simplifying the diagram a bit more, it looks like:



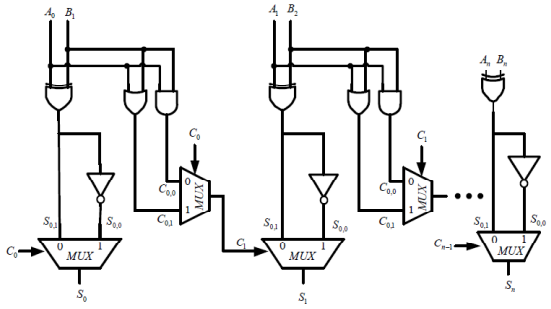
If we assume a mux takes as long as a logic gate, then this circuit can compute a 16-bit addition in 2 4 + 4 = 12 gate delays: 8 for all the adders to finish, and 4 for the muxs to ripple the carry bits across. For a 64-bit adder, it would take 24 delays, because it would have 16 muxes instead of 4. Going from 128 to 24 is a great start, and it only cost us a little less than twice as many gates!We can fuss with this and make it a little faster. The leftmost adder unit waits a long time to get its incoming carry bit, and the first 75% of the time is spent waiting for the first adder to finish. If we compute only one bit at a time on the right, then two, then three, and so on as it goes left, we can shave off a few more.

4. DISCUSSION:

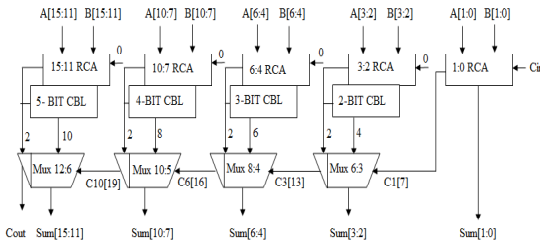
To remove the duplicate adder cells in the conventional CSLA, an area efficient SQRT CSLA is proposed by sharing Common Boolean Logic (CBL) term. While analysing the truth table of single bit full adder, results show that the output of summation signal as carry-in signal is logic "0" isinverse signal of itself as carry-in signal is logic "1". It is illustrated by red circles in Table II. To share the Common Boolean Logic term, we only need to implement a XOR gate and one INV gate to generate the summation pair. And to generate the carry pair, we need to implement one OR gate and one AND gate. In this way, the summation and carry circuits can be kept parallel.

Cin	A	B	S0	C0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

This method replaces the Binary to Excess-1 converter add one circuit by common Boolean logic. As compared with modified SQRT CSLA, the proposed structure is little bit faster. Internal structure of proposed CSLA is shown in Fig.



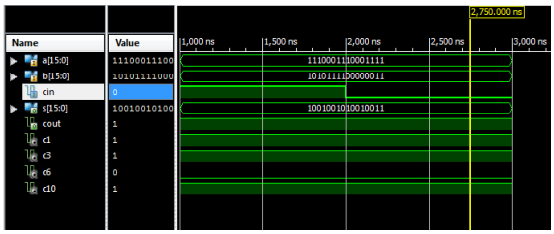
In the proposed SQRT CSLA, the transistor count is tradeoff with the speed in order to achieve lower powerdelay product. Thus the proposed SQRT CSLA using CBL is better than all the other designed adders. Fig. 9 shows the block diagram of Proposed SQRT CSLA.



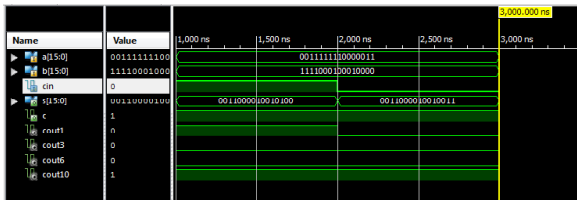
ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

5. Experimental Results:

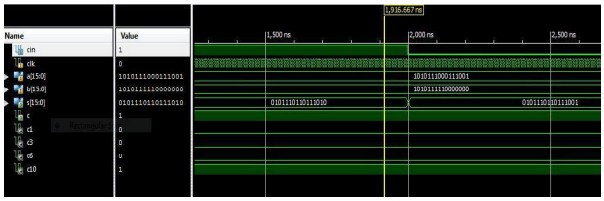
This work has been developed using Xilinx tool. Table III shows the comparison between the various adders like conventional CSLA, Modified CSLA, regular SQRT CSLA, modified SQRT CSLA and proposed SQRT CSLA for 8-bit, 16-bit, 32-bit and 64-bit. The parameters on which they are compared are area, delay and power. Fig. 10 depicts that the proposed SQRT CSLA has less number of gates and hence less area. Fig. 11 shows the adder circuit for delay comparison. The results compared in Fig. 12 shows that the power consumption of proposed SQRT CSLA is reduced. It is clear that power, area and delay of proposed SQRT CSLA for 8-bit, 16-bit, 32-bit and 64-bit is reduced as compared to other adder.



Regular SQRT CSLA Simulation



CSLA using BEC



Modified CSLA using D-Latch

6. Concluding Remarks :

Power, delay and area are the constituent factors in VLSI design that limits the performance of any circuit. This work presents a simple approach to reduce the area, delay and power of CSLA architecture. The conventional carry select adder has the disadvantage of more power consumption and occupying more chip area.

The proposed SQRT CSLA using common Boolean logic has low power, less delay and reduced area than all the other adder structures. It is also little bit faster than all the other adders. In this way, the transistor count of proposed SQRT CSLA is reduced having less area and low power which makes it simple and efficient for VLSI hardware implementations.

7.ACKNOWLEDGMENTS:

I am B.Sindhurmai and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Associate Prof: Mr. B. Naresh Reddy for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

8.REFERENCES:

1] M. C. C¸ avu¸so˘glu. Telesurgery and Surgical Simulation: Design, Modeling, and Evaluation of Haptic Interfaces to Real and Virtual Surgical Environments. PhD thesis, University ofCalifornia, Berkeley, August 2000.

[2] M. C. C¸ avu¸so˘glu, F. Tendick, M. Cohn, and S. S. Sastry. A laparoscopic telesurgical workstation. IEEE Transactions on Robotics and Automation, 15(4):728–739, August 1999.

[3] E. Graves. Vital and Health Statistics. Data f rom the National Health Survey No. 122. U.S. Department of-Health and Human Services, Hyattsville, MD, 1993.

[4] J. W. Hill, P. S. Green, J. F. Jensen, Y. Gorfu, and A. S. Shah. Telepresence surgery demonstration system. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 2302–2307, 1994.

[5] A. J. Madhani. Design of Teleoperated Surgical Instruments for Minimally Invasive Surgery. PhD thesis, Massachusetts Institute ofTechnology, 1998.

[6] A. J. Madhani, G. Niemeyer, and J. K. Salisbury. The black falcon: a teleoperated surgical instrument for minimally invasive surgery. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98), volume 2, pages 936–944, 1998

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[7] J.W.Hill, P. S. Green, J. F. Jensen,Y. Gorfu, and A. S. Shah, "Telepresencesurgery demonstration system," in Proc. IEEE Int. Conf. Robot. Autom.,San Diego, CA, May 1994, vol. 3, pp. 2302–2307.

[8] P. Dario, E. Guglielmelli, B. Allotta, and M. C. Carrozza, "Robotics formedical applications," IEEE Robot. Autom.Mag., vol. 3, no. 3, pp. 44–56,Sep. 1996.

Author's Details:

Ms. B.Sindhurmai MTech student, in M.Tech Student, Dept of CSE in KITS for women's,kodad, T.S, India.

Mr. B. Naresh Reddy working as a Assistant at ECE in KITS for women's,kodad, T.S, IndiaJNTUH Hyderabad. he has 6 years of UG/PG Teaching Experience.

.