Common adder topologies- Ripple-Carry, Carry-Look-Ahead, Carry-Select and Kogge-Stone

Aditya Ashvin
Department of Electronics and Communication Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri, India
adityaashvin2@gmail.com

Gouri S
Department of Electronics and Communication Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri, India
gourisuja00@gmail.com

Kuruba Aparna
Department of Electronics and Communication Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri, India
aparnakurba07@gmail.com

Singamsetty Chandu Priya
Department of Electronics and Communication Engineering,
Amrita Vishwa Vidyapeetham,
Amritapuri, India
chandupriya.2802@gmail.com

***Abstract*— Adders are fundamental building blocks and sometimes constitute a part of the critical path. Adders play an important role in a variety of arithmetic and logical processes. In Electronics, Adder may be a digital circuit that performs the addition of a particular number of bits. The complexity of the circuit and its operation are compared by a general operation of adding a particular number of bits. In this paper, we design, implement and analyse four different common adder topologies such as Ripple-Carry, Carry-Look-Ahead, Carry-Select and Kogge-Stone. Further, a testing plan (input vectors and expected outputs) is proposed to prove complete functionality and to work out the worst-case delay of your adders.**

Keywords*— *Ripple carry, carry look ahead, carry select, Kogge-stone

I. Introduction

In every VLSI design, the adder is a fundamental building block when arithmetic calculations are involved. Adders are digital circuits used in almost every digital system. Since adders constitute to one of the basic elements in most digital systems used in modern times, optimum adders can have effect in reducing the area, power and increasing the speed of the system. The usage of adders reduces the number of transistors being used in a circuit. Works are that specialize in the configuration of FPGA to optimize for a spread of constrains like area and computational time in ANN.

In modern day world, digital circuitry is entirely based on its performance and the adders play a very crucial role in enhancing the performance of the digital system. The area used, power consumption and delay are the parameters that determine how good a digital system are.

It is quite obvious that a single adder cannot be used to meet all the requirements. So, the adders are selected for various applications based on the constraints each application holds with it. For instance, the use of microprocessors and microcontrollers are increasing intensively. One of the key elements in these microprocessors and microcontrollers is the multipliers. It is known fact that multiplication is repeated addition and thus the importance of adders in multipliers. So, the performance of multipliers depends on adders and successively the general performance of the processor system depends on adder performance.

The usage of various adders depends on the appliance as mentioned above. So, an adder which consumes more area may not be considered in a system in which size is a constraint. Similarly, if the user can bear the power and area consumption complex adders with high speed can be used in such situations. It is clear that the most parameters for and adder is that the area it occupies, the facility it consumes and therefore the speed at which the computation takes place

The foremost necessity of any digital circuit is:

- low power consumption
- low power dissipation
- low area
- High speed.

Comparison among the 4-bit adders has been done to select an appropriate adder.

- Ripple-Carry
- Carry-Look-Ahead

- Carry-Select
- Kogge-Stone.

II. Different types of adders

A. Ripple Carry Adder

The ripple carry adder is the most straightforward adder available, but is additionally the slowest adder having O(n) area and O(n) delay, where n denotes the operand size in bits. The worst-case delay of the simple ripple carry adder as shown in the figure, is when a carry signal transition ripples through all stages of adder chain from the least significant bit to the foremost significant bit, which may be approximated by:

$$t = (n-1)t_c + t_s \quad [1]$$

Here, the delay through the carry stage of a full adder is denoted by t_c , and the delay to compute the sum of the last stage is denoted by t_s. The delay of ripple carry adder is linearly proportional to n, where n denotes the amount of bits. Therefore, the performance of the ripple carry adder is limited when n grows bigger. The advantages of the ripple carry adder include low power consumption also as compact layout giving smaller chip area.

B. Carry Look Ahead

A Carry-Look-Ahead Adder (CLA) is an electronic adder that is the successor of the Ripple Carry Adder. The CLA logic uses the concepts of generating and propagating carries and also performs additions quickly, Faster ways are devised by engineers to scale back computation time and to add two binary numbers by using Carry-Look-Ahead Adder Carry-Look-Ahead adder is additionally referred to as a quick adder. In Carry-Look-Ahead Adder the Result of the larger-value bits of the adder is calculated by reducing the wait time and also calculates one or more carry bits before the sum.

C. Carry Select

Carry Select Adder (CSLA) is a fast adder which is commonly used in the digital communication. CSLA independently generates multiple carries and then selects one carry to produce the sum. It can reduce the problem of carry-propagation delay compared to Ripple carry adders because the value of sum is computed parallelly by each RCA pair before the previous stage carry comes. However, the conventional CSLA is not efficient in terms of area as it uses pairs of RCA's to produce the sum and carry by assuming Cin =0 and Cin=1, and the final sum and carry are selected by the multiplexers. Since the additions in CSLA are carried out parallelly and also due to the reduced maximum carry path CSLA is 40% to 90% faster than RCA. But since there exists copy of hardware in each stage, it may cause hike in the amount of power consumption and cost.

D. Kogge Stone

Kogge and Stone (1973) introduced the Kogge-Stone adder method, which has low fan-out and optimal depth but builds complex circuits and consequently a large number of interconnects. The delay of this structure, for high-speed applications, is given by,

$$[\log_2 (n)] \quad [2]$$

and the computational nodes is given by,

$$[n \log_2 (n) - n + 1] \quad [3]$$

the Kogge-Stone structure is quite useful. The recursive doubling algorithm concept is used to overcome the problem of fan-out in this construction. The Kogge-Stone adder also has a standard layout. The parallel prefix form carry look-ahead adder (KSA) is a carry look-ahead adder that uses parallel prefix forms. It is known as the fastest adder in the industry and is widely utilized in high-performance arithmetic circuits. Carry is generated in O (log n) time in KSA Carriers are computed quickly by running them in parallel, however this comes at the cost of more space Figure 5 shows the tree structure of a three-bit Kogge stone adder (KSA). It comprises three processing steps for calculating the sum bits.

III. Design and working of the adders

A. Ripple Carry Adder

Looking into the block diagram, the operands that we used are A and B, the operation to be performed is addition and the results are sum(S) and Carry(C). The first bits of the operands, A0 and B0 are sent to the very first full adder A1, it is assumed that the input carry, Cin is 0. The output generated will be the first bit of sum S0 and the output carry Cout will be rippled to the preceding full adder. Likewise, the second bits of operands are then given to the second full adder, and the third bits are then given to the third full adder and fourth bits are then given to the fourth adder. The sum generated by each full adder will be the corresponding bit of sum and the carry generated by each of the full adder, which will be then rippled to the next full adder as an input carry. The output carry of the operation executed is provided by the last full adder provides us the output carry.

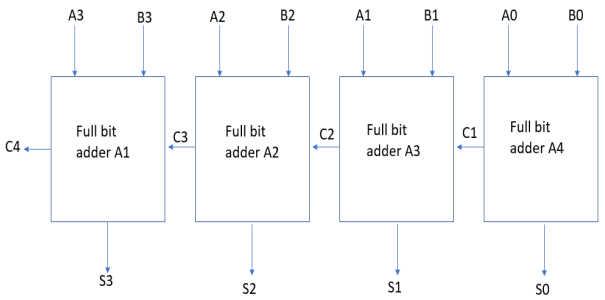
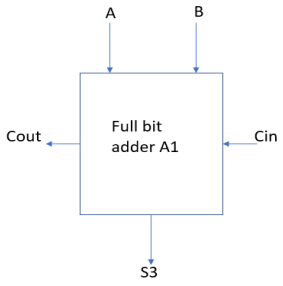


Fig 1. 4 bit Ripple Carry Adder

The layout of ripple carry adder as shown in the figure is simple, which facilitates fast design time. However, the ripple carry adder is relatively slower when the number of stages get increased. This is because, each full adder must have to wait for the carry bit to be calculated from the previous full adder.

The gate delay of the ripple carry adder can be easily calculated by inspecting the full adder circuit. Each full adder requires up to 3 levels of logic. In a simple 32-bit ripple carry adder, there are 32 full adders and therefore, the

critical path in the worst-case delay is 3 [from input to carry in first adder] + 31 * 2 [for carry propagation in later adders)] which equals to 65 gate delays. The fact that ripple-carry adder gets very slow when there is a need to add more bits is considered as the biggest disadvantage of it.



Inputs			Outputs	
A	B	Cin	Cout	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Fig 2. 1 bit Ripple Carry Adder

In the ripple carry adder shown in the figure1, the output of the first sum does not result until an input carry is given to the adder. Similarly, the second sum output also does not result until the previous carry is propagated and so on. The final sum and the carry outputs will not be resulted until all the previous adder carries are produced which may also result in delay. To cutback the delay in a ripple carry adder, the carry look ahead adder is implemented.

B. Carry Look Ahead

The inputs of the Adder are A, B, Cin. We get the carry output as 1 when:

- One among the A or B is 1 and Cin is 1 or
- Both A and B are 1.

Carry-Look-Ahead Adder have two concepts generate and propagate carries. Generate and Propagate Carries are denoted by Gi and Pi variables respectively.
We has two cases to get the Carry Output as 1:

- When an Input carry is given, then an Output Carry is propagated. So, the mathematical expression of Pi can be represented as:

$$Pi = Ai \oplus Bi \quad [4]$$

- when both inputs, A and B, are high, regardless the value of the input carry then an Output Carry is generated. We will call this output carry as Gi. Thus, we will mathematically express Gi as:

$$Gi = Ai \cdot Pi \quad [5]$$

Usually, for a Full Adder we have

$$Sum = A \oplus B \oplus C_i \quad [6]$$

$$Carry = C_i (A+B) + AB \quad [7]$$

The equations of Full Adders are often represented in terms of Carry Propagate (Pi) and Carry Generate (Gi) are:

$$Sum = Pi \oplus C_i \quad [8]$$

$$Carry = Gi + Pi \cdot C_i \quad [9]$$

Output carry C1, C2, C3, and C4 can be calculated using the above derived equations as:

$$C1 = (C_{in} \cdot \mathbf{P0}) + \mathbf{G0} \quad [10]$$

$$C2 = (C1 \cdot P1) + G1 = (((C_{in} \cdot P0) + G0) \cdot P1) + G1$$

$$= (C_{in} \cdot \mathbf{P0} \cdot \mathbf{P1}) + (\mathbf{G0} \cdot \mathbf{P1}) + \mathbf{G1} \quad [11]$$

$$C3 = (C2 \cdot P2) + G2 = (((C1 \cdot P1) + G1) \cdot P2) + G2$$

$$= \mathbf{G2} + (\mathbf{P2} \cdot \mathbf{G1}) + (\mathbf{P2} \cdot \mathbf{P1} \cdot \mathbf{G0}) + (\mathbf{P2} \cdot \mathbf{P1} \cdot \mathbf{P0} \cdot C_{in}) \quad [12]$$

$$C4 = (C3 \cdot P3) + G3$$

$$= (C_{in} \cdot \mathbf{P0} \cdot \mathbf{P1} \cdot \mathbf{P2} \cdot \mathbf{P3}) + (\mathbf{P3} \cdot \mathbf{P2} \cdot \mathbf{P1} \cdot \mathbf{G0}) + (\mathbf{P3} \cdot \mathbf{P2} \cdot \mathbf{G1}) + (\mathbf{G2} \cdot \mathbf{P3}) + \mathbf{G3} \quad [13]$$

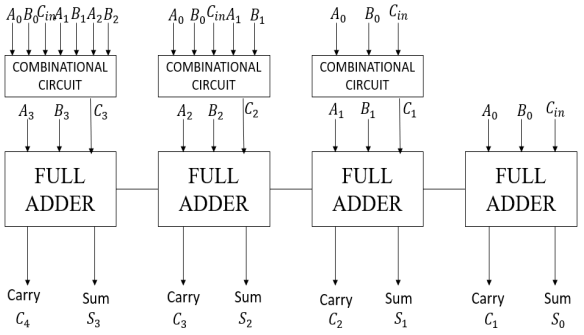


Fig 3. 4-bit Carry Look Ahead Adder

A	B	Cin	Sum	Carry	Condition
0	0	0	0	0	No Carry Generated
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	Carry Propagated
1	0	0	1	0	No Carry
1	0	1	0	1	Carry Propagated
1	1	0	0	1	Carry Generated
1	1	1	1	1	

Fig 4. Truth table for full adder

We can see that there's no dependency on any intermediate Carry values in any of the equations. On solving the equations, we see that Sum and Output Carry values can be calculated by taking only the input Carry C_{in}.
This resolves the problem faced by the Ripple Carry Adder's dependency on the intermediate carry values. Thus, the whole operation of carry look ahead Adder works faster for higher-order bits, in comparison to the Ripple Carry Adder. This is often the rationale why the CLA Adder is additionally called a Fast Adder.

C. Carry Select Adder

A carry select adder has two sections, each performing two additions parallelly, the first one assumes Cin to be 0 and the other assumes Cin as 1. A four-bit carry select adder basically consists of two Ripple Carry Adders and a multiplexer. When both the results are calculated (with Cin=0 and Cin=1), the correct sum and carry are selected with the multiplexer. If the previous carry is logic '0' then the multiplexer selects the result of carry "0" path or if the previous carry is logic '1' then the result of carry "1" path is selected by the multiplexer i.e., actual carry is utilized to pick the sum and carry using a multiplexer.

The operands say, X and Y are given to the first set of full adders with Cin=0 and the same operands are given to the next set of full adders with Cin=1. Here each set of 4 full adders act as a RCA. The output from the two RCA's, the sum bits for Cin=0 and Cin=1 are given as input to a 2x1 multiplexer. Here the input carry provided by the user will act as the selection line.

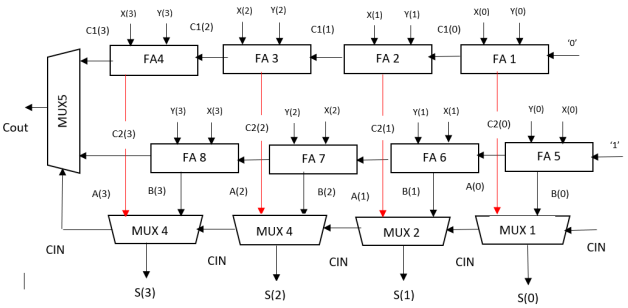


Fig 5. 4 bit Carry Select Adder

There are two types of carry select adders: Uniform sized adders and variable sized adders. The adder is called uniform sized adder or linear Carry Select Adder if the bit length is equally divided. It is called variable sized adder or SQRT Carry Select Adder if the bit length is unequally divided. The variable sized adder has one RCA and a binary to excess-one converter instead of two Ripple Carry adders

D. Kogge Stone Adder

Pre-processing: During A and B, this phase entails computing the generate and propagate signals corresponding to each pair of bits. These signals are given by the logic equations:

pi=Ai xor Bi [14]

gi = Ai and Bi [15]

Carry a look-ahead network: This block sets KSA apart from other adders and is the driving cause behind its outstanding performance. Calculating the carriers that correspond to each bit is part of this phase. It employs group propagate and generate as intermediary signals, as defined by the following logic equations:

Pi :j = Pi:k+1 and Pk :j and [16]

Gi :j = Gi:k+1 or (Pi:k+1 and Gk :j [17]

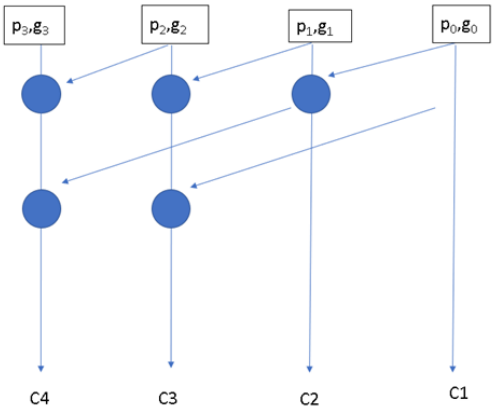


Fig 6. 4 bit Kogge Stone Adder

Post-processing: This is the final step, which is the same for all or any of the adders in this family (carry look-ahead).It entails adding and subtracting bits. The logic below is used to compute some bits.

Si = pi xor Ci-1 [18]

IV. Comparison of the Adders

Ripple Carry Adder uses less area. Carry Look Ahead adder also utilises the same area as Ripple Carry adder. In comparison, the Carry Select adder is not area efficient as it uses multiple RCA's to produce the sum and carry. For delay comparison, it can be observed that maximum time delay is for Ripple Carry adder as it takes more time for execution. In comparison to RCA, Carry Look Ahead adder executes the operation in lesser time. The minimum delay occurs for carry select, as it is computing parallelly. The Kogge Stone Adder, is the observed to be the best among all the adders since it requires less area (which is same as that of RCA) and also requires lesser time than that of RCA.

V. Testing Plan

The simulation process will be done using the Cadence tool. The simulation results will give us information about the operands, operations, and outcomes. The operands here are A and B. They are of 4 bits each and there is also an input carry Cin. The values of operands to be taken for the first case are A = "1010", B = "1001", Cin = '0' for the first 100 ns which results in the sum as S ="1111" and output carry as Co = '0'. The second set of values are A = "1101", B = "0110", Cin = '0' from 100 ns to 200 ns which resulted in the sum bits as S = "0011" and output carry as Co = '1'

A	B	Cin	Sum
1010	1001	0	1111
1101	0110	0	0011

References

[1] B. Koyada, N. Meghana, M. O. Jaleel and P. R. Jeripotula, "A comparative study on adders," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017, pp. 2226-2230, doi: 10.1109/WiSPNET.2017.8300155.

[2] P. L. Thangkhiew, R. Gharpinde, P. V. Chowdhary, K. Datta and I. Sengupta, "Area efficient implementation of ripple carry adder using memristor crossbar arrays," 2016 11th International Design & Test Symposium (IDT), 2016, pp. 142-147, doi: 10.1109/IDT.2016.7843030.

[3] Ananthakrishnan, A. Ajit, A. P.V., K. Haridas, N. M. Nambiar and D. S., "FPGA Based Performance Comparison of Different Basic Adder Topologies with Parallel Processing Adder," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), 2019, pp. 87-92, doi: 10.1109/ICECA.2019.8821925.

[4] B. Harish, K. Sivani and M. S. S. Rukmini, "Design and Performance Comparison among Various types of Adder Topologies," 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 2019, pp. 725-730, doi: 10.1109/ICCMC.2019.8819779.

[5] J. M R and Manjudevi, "Analysis of Special Adders for Digital System," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 253-256, doi: 10.1109/ICOEI51242.2021.9452953.

[6] J. Saini, S. Agarwal and A. Kansal, "Performance, analysis and comparison of digital adders," 2015 International Conference on Advances in Computer Engineering and Applications, 2015, pp. 80-83, doi: 10.1109/ICACEA.2015.7164650.

[7] R.UMA,Vidya Vijayan, M. Mohanapriya , Sharon Paul,Research Scholar, Department of Computer Science, Pondicherry University, Area, Delay and Power Comparison of Adder Topologies

[8] Rishabh Rai and Rajni Parashar, Department of Electronics & Communication Engineering, Ajay Kumar Garg Engineering College "An Efficient Carry Select Adder–A Review"