International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 12
August 2016

# FPGA Binary Addition & Carry Tree Adders Using Prefix Computation or Addition

**Mr. M. MAHABOOB BASHA[1]&DADA PEERIAH KALLURI[2]**

[1]Associate Professor Dept. of ECE, Svr Engineering College Nandyal Mail: - mmbfasi@gmail.com

[2]PG-Scholar Dept. of ECE, Svr Engineering College Nandyal

## Abstract

Adders are basic functional units in computer arithmetic. Binary adders are used in microprocessor for addition and subtraction operations as well as for floating point multiplication and division. Therefore adders are fundamental components and improving their performance is one of the major challenges in digital designs. Variable latency adders have been recently proposed in literature. A variable latency adder employs speculation: the exact arithmetic function is replaced with an approximated one that is faster and gives the correct result most of the time, but not always. The approximated adder is augmented with an error detection network that asserts an error signal when speculation fails. Speculative variable latency adders have attracted strong interest thanks to their capability to reduce average delay compared to traditional architectures. This paper proposes a novel variable latency speculative adder based on Han-Carlson parallel-prefix topology that resulted more effective than variable latency Kogge-Stone topology. The paper describes the stages in which variable latency speculative prefix adders can be subdivided and presents a novel error detection network that reduces error probability compared to previous approaches.

**Keywords:** FPGA, Binary addition, Carry tree adders, Prefix computation, Prefix addition.

## 1. Introduction

VLSI binary adders are critically important elements in processor chips, they are used in floating-point arithmetic units, ALUs, and memory addresses program counter update and magnitude comparator [1, 2]. Adders are extensively used as a part of the filter such as DSP lattice filter [3]. Ripple carry adder is the first and most fundamental adder that is capable of performing binary number addition. Since its latency is proportional to the length of its input operands, it is not very useful. To speed up the addition, carry look ahead adder is introduced. Parallel prefix adders provide good results as compared to the conventional adders. The adders with the large complex gates will be too slow for VLSI, so the design is modularized by breaking it into trees of smaller and faster adders which are more readily implemented. For

large adders the delay of passing the carry through the look-ahead stages becomes dominated and therefore tree adders or parallel prefix adders are used. High speed adders depend on the previous carry to generate the present sum. In integer addition any decrease in delay will directly relate to an increase in throughput. In nanometer range, it is very important to develop addition algorithm that provide high performance while reducing power. Parallel prefix adders are suitable for VLSI implementation since they rely on the use of simple cells and maintain regular connection between them. We can define each prefix Structures in terms of logic levels, fan-out and wiring tracks. Zero or more inverters are added to each prefix cell output to minimize the delay based on this model, buffers are individually sized to minimize the delay, buffers are used to minimize the fan-out and loading on gates since high fan-out causes poor performance. we design an extremely fast unreliable adder that produces correct results for the vast majority of input combinations. For brevity, we will call this adder an Almost Correct Adder (ACA).

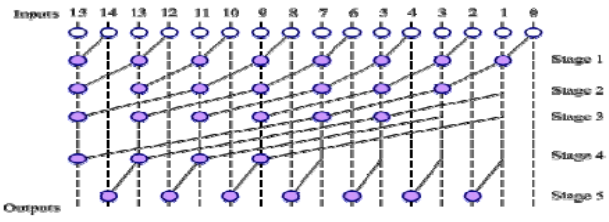



Fig 1. Graph representation of 16-bit Hybrid Han-Carlson Adder.

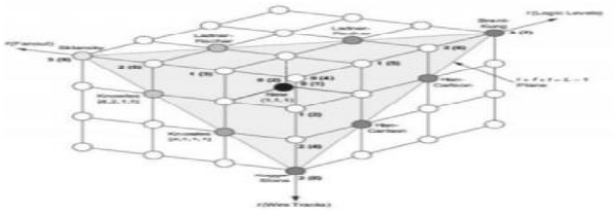


Fig 2. Taxonomy of prefix networks.

## 2. Related Work

Design the Speculative Han-Carlson Adder. It differs from other adder in that it can be used for large word sizes. The proposed design reduces the number of prefix operation by using more number of Brent-Kung stages and lesser number of Kogge-Stone stages. This also reduces the complexity, silicon area and power consumption significantly. Variable latency speculative prefix adders can be subdivided

in five stages: pre-processing, speculative prefix-processing, post-processing, error detection and error correction. The error correction stage is off the critical path, as it has two clock cycles to obtain the exact sum when speculation fails.

Consider the n-bit addition of two numbers: A = an−1, an−2, a0 and B = bn−1, bn−2, b0 resulting in the sum, S = sn−1, sn−2, s0 and a carry, Cout. The first stage in CLA computes the bit generate and bit propagate as follows:

$$g_i = a_i \cdot b_i$$

$$p_i = a_i + b_i, \qquad (1)$$

Where gi is the bit generates and pi is the bit propagate. The schematic of gi and pi using CMOS and transmission gates design style. These are then utilized to compute the final sum and Carry bits, in the last stage as follows:

$$Ci+1 = gi + pi \cdot ci, \qquad (2)$$

Where ·, + and ⊕represent AND,OR, and XOR operations. It is seen that the first and last stages are intrinsically fast because they involve only simple operations on signals local to each bit position. However, intermediate stages embody the long-distance propagation of carries, as a result of which the performance of the adder hinges on this part [10]. These intermediate stages calculate group generate and group propagate to avoid waiting for a ripple which, in turn, reduces the delay. These group generate and propagates are given by

$$Pi{:}\,j = Pi{:}k \cdot Pk{-}1{:}\,j\,,$$

$$Gi{:}\,j = Gi{:}k + Gk{-}1{:}\,j \cdot Pi{:}k. \qquad (3)$$

There are many ways to develop these intermediate stages, the most common being parallel prefix. Many parallel prefix networks have been described in the literature, especially in the context of addition. In this paper, we have used the Kogge-Stone implementation, Hans-Carlson, Sklansky, Brent-Kung implementation of CLA, and Kogge- Stone implementation of Ling adder. PG logic in all adders is generally represented in the form of cells. These diagrams known as cell diagrams will be used to compare a variety of adder architectures in the following sections. Here two cells are used for implementation of all the adders: grey cell and the black cell.

**Han-Carlson Adder**: The Han-Carlson trees are a family of networks between Kogge-Stone and Brent-Kung. The logic performs Kogge-Stone on the odd numbered bits and then uses one more stage to ripple into the even positions.

**Kogge-Stone Adders:** The main difference between KoggeStone adders and other adders is its high performance. It calculates carries corresponding to every bit with the help of group generate and group propagates. In this adder the logic levels are given by log2N, and fan-out is 2.

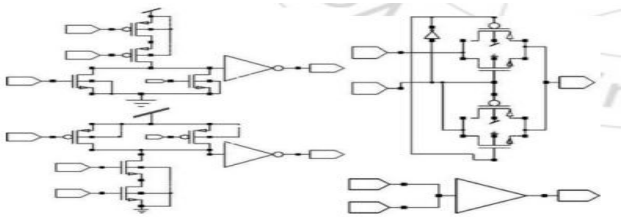


Figure 3: Block generate and propagate (Ling carry) using CMOS and transmission gate.

Now, instead of utilizing traditional carries, a new of carry, known as Ling carries, is produced where the ith Ling carry in [11] is defined to be

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 12
August 2016

$$ci = Hi \cdot pi, \quad (5)$$

Where

$$Hi = ci + ci-1. \quad (6$$

In this way, each $Hi$ can be in turn represented by

$$Hi = gi + gi-1 + pi-1 \cdot gi-2 + \cdots + pi-1 \cdot pi-2 \cdot pi-3 \cdot \ldots p1g0. \quad (7)$$

We can see from (5) that Ling carries can be calculated much faster than Boolean carry. Consider the case of $c_4$ and $H_4$

$$c4 = g4 + p4 \cdot g3 + p4 \cdot p3 \cdot g2 + p4 \cdot p3 \cdot p2 \cdot g1 + p4 \cdot p3 \cdot p2 \cdot p1 \cdot g0, \quad (8)$$
$$H4 = g4 + g3 + p3 \cdot g2 + p3 \cdot p2 \cdot g1 + p3 \cdot p2 \cdot p1 \cdot g0.$$

If we assume that all input gates have only two inputs, we can see that calculation of c4 requires 5 logic levels, whereas that for H4 requires only four. Although the computation of carry is simplified, calculation of the sum bits using Ling carries is much more complicated. The sum bit, when calculated by using traditional carry, is given to be

$$si = di \oplus ci - 1. \quad (9)$$

Substituting (5) into (9), we get that

$$si = di \oplus pi - 1 \cdot Hi - 1 \quad (10)$$

However, according to [12] the computation of the bits si can be transformed as follows:

$$si = Hi\text{-}1 \cdot di + Hi\text{-}1(di \oplus pi\text{-}1) \quad (11)$$

Equation (11) can be implemented using a multiplexer with Hi− 1 as the select line, which selects either di or (di ⊕ pi− 1). No extra delay is added by Ling carries to compute the sum since the delay generated by the XOR gate is almost equal to that generated by the multiplexer and that the time taken to compute the inputs to the multiplexer is lesser than that taken to compute the Ling carry.. Here, for n-bit addition, Ling carry Hi and Hi+1 is given by

$$Hi = (G*i, P*i-1), (G*i-2, P*i-3), \boxplus \cdots \boxplus (G*0, P*-1),$$
$$Hi+1 = (G*i+1, P*i), (G*i-1, P*i-2) \cdots (G*1, P*0), \quad (12)$$

Where

$$G*I = gi + gi-1,$$
$$P*I = pi \cdot pi-1 \quad (13)$$

$$H3 = g3 + g2 + p2 \cdot g1 + p2 \cdot p1 \cdot g0, \quad (14)$$
$$H4 = g4 + g3 + p3 \cdot g2 + p3 \cdot p2 \cdot g1 + p3 \cdot p2 \cdot p1 \cdot g0.$$

$$H3 = G*3 + P*2 \cdot G*1,$$
$$H4 = G*4 + P*3 \cdot G*2 + P*3 \cdot P*1 \cdot G*0. \quad (15)$$

This can be then further reduced by using the "." operator to

This allows the parallel prefix computation of Ling adders using a separate tree [9] for even and odd indexed positions. Using this methodology, we implemented a 16-bit adder using the Kogge-Stone tree and then utilized that block to develop 32 and 64-bit adders. The gates and blocks used for this implementation were then modified using transmission gates. Cells other than gray and black cell that are used as components in Ling adder.

## 3. Experimental Results

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 12
August 2016

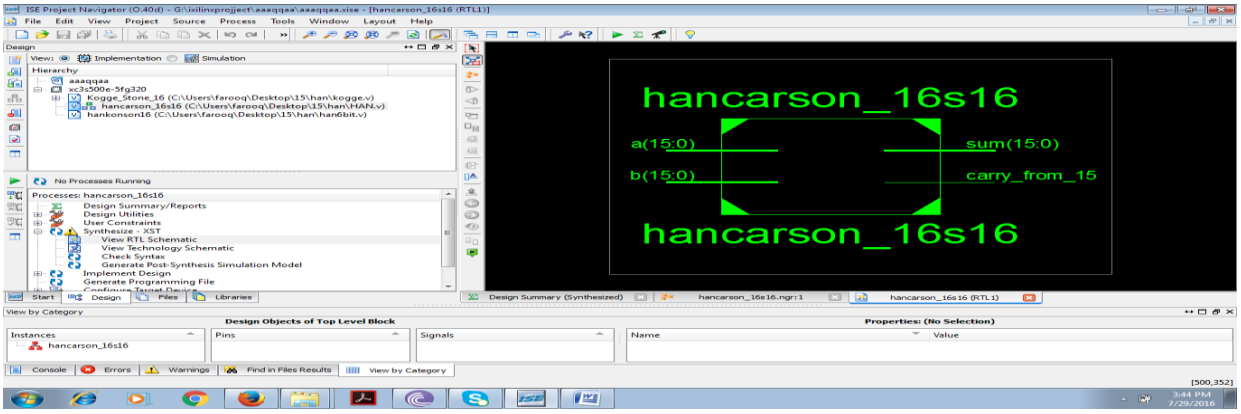
Verilog descriptions of the proposed variable latency speculative adders, and of their non-speculative counterpart. It is not easy to compare performances (in terms of power, speed, and area) of different designs, since they strongly depend on timing constraint used during synthesis.

### 3.1 The Optimal K Choice

Comparison between variable latency adder and the nonspeculative Han-Carlson topology reveal that variable latency adders allow to reduce the minimum achievable delay. For instance, in the 64 bit case, the minimum achievable delay is about 280 ps for the non-speculative adder and reduces up to 225 ps in the variable latency architecture.

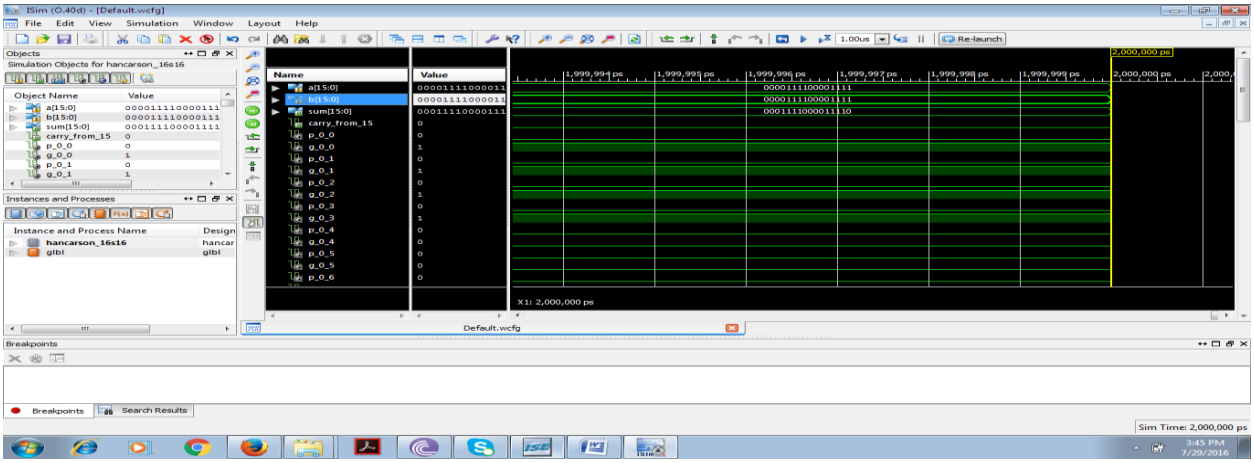
### 3.2 Comparison with Kogge-Stone Variable Latency Speculative Adder:

Fig. 7 shows the comparison between proposed speculative adder and Kogge-Stone one. Also in this case, we report the performance of non-speculative adders, in order to identify the region where the speculative approach is effective. As an example, focusing on 64-bit adders, for lower than 350 ps, the proposed Han-Carlson speculative adder is the best choice in terms of silicon area and power consumption. Moreover, it allows reducing the minimum achievable to 225 ps, with a 18% improvement respect to Kogge-Stone no speculative adder and a 11% improvement respect to KoggeStone speculative adder. For , proposed speculative adders offer 45% area reduction and 35% power saving compared to Kogge-Stone non-speculative adder.




4(a)

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 12
August 2016



4(b)

Fig. 4(a). Area report of Han-Carlson Adder, 4(b). error correction report of Han- Carlson Adder

## 4. Conclusion

The variable latency Han-Carlson parallel prefix speculative adder for high-speed application is proposed. A new, more accurate, error detection network is introduced, which allows reducing the error probability compared to the other approaches. Compared with traditional, non-speculative, adders, our analysis demonstrates that variable latency Han-Carlson adders show sensible improvements when the highest speed is required; otherwise the burden imposed by error detection and error correction stages overwhelms any advantage. This can be used in various applications like digital signal processing, satellite and mobile phones. The 16 Bit Existing and Proposed adders are implemented. In the future work, proposed architecture will be converted into 32 bit Adders and analyzes the area and speed. This adder will be done by using our verilog and the Adder design will implemented into FPGA

## 5. References

[1] O. J. Bedrij, .Carry-select adder,. IRE Trans. Electron. Comput., pp. 340.346, June 1962.

[2] R. P. Brent and H. T. Kung, .A regular layout for parallel adders,. IEEE Trans.Computers, vol. C-

31, no. 3, pp. 260.264, Mar. 1982.

[3] J. Chen and J. E. Stine, .Optimization of bipartite memory systems for multiplicative divide and square root,. 48th IEEE International Midwest Symp. Circuits and Systems, vol. 2, pp. 1458.1461, 2005.

[4] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in Proc. Design, Autom., Test Eur. Conf. Exhib, (DATE'09), Apr. 2009, pp. 664–669.

[5] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Proc. Design, Autom., Test Eur. Conf. Exhib.(DATE '12), Mar. 2012, pp. 1257–1262.

[6] D. Goldberg, .What every computer scientist should know about floating point arithmetic,. ACM Computing surveys, vol. 23, no. 1, pp. 5.48, 1991.

[7] T. Han and D. Carlson, .Fast area-efficient VLS Adders,. in Proc. 8th Symp. Comp. Arith., Sept. 1987, pp. 49.56.

[8] D. Harris, .A taxonomy of parallel prefix networks, in Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, Nov. 2003, pp. 2213.2217.

[9] S. Knowles, .A family of adders,. in Proc. 15th IEEE Symp. Comp. Arith., June 2001, pp. 277.281.

[10] P. Kogge and H. Stone, .A parallel algorithm for the efficient solution of a general class of recurrence relations,. IEEE Trans. Computers, vol. C-22, no. 8, pp. 786.793, Aug. 1973.

[11] I. Koren, Computer Arithmetic Algorithms. Natick, MA, USA: A K Peters, 2002.

[12] H. Ling, .High speed binary adder,. IBM Journal of Research and Development, vol. 25, no. 3, pp.156.166, 1981.

[13] R. Ladner and M. Fischer, .Parallel prefix Computation,.J. ACM, vol. 27, no. 4, pp. 831.838,

Oct. 1980.

[14] J. Sklansky, .Conditional sum addition logic,. IRE Trans. Electron. Comput., pp. 226. 231, June

1960.

[15] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion,"

IEE Proc. Comput. Digit. Tech., vol. 143, no. 5, pp. 301–307, Sep. 1996.

[16] V. G. Oklobdzija, B. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, Energy delay estimation technique for high-performance microprocessor VLSI adders,. Proc. 16th IEEE

Symp. Computer Arithmetic (ARITH-16'03), p. 272, June 2003.

[17] A. K. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in Proc. Design, Autom., Test Eur. (DATE '08), Mar. 2008, pp. 1250–1255.

[18] S. Winograd, .On the time required to perform addition,.J. ACM, vol. 12, no. 2,pp. 277.285, 1965.

[19] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. thesis, Swiss Federal Institute of Technology, (ETH) Zurich, Zurich, Switzerland, 1998, HartungGorre Verlag.