Design of 128- bit Kogge-Stone Low Power Parallel Prefix VLSI Adder for High Speed Arithmetic Circuits

P.Annapurna Bai, M.Vijaya Laxmi

Abstract - Parallel Prefix adders have been one of the most notable among several designs proposed in the past. The advantage of utilizing the flexibility in implementing the three structures based upon throughput requirements. Due to continuing integrating intensity and the growing needs of portable devices, low-power and high-performance designs are of prime importance. The classical parallel prefix adder structures presented in the literature over the years optimize for logic depth, area, and fan-out and interconnect count of logic circuits. In this paper, a new architecture for performing 128-bit Parallel Prefix addition is proposed. In this proposed system kogge-stone adder which is one of types of parallel prefix adder is used. Kogge-stone is the fastest adder because of its minimum fan-out. The proposed 128-bit prefix adder is compared with classical adders of same bit width in terms of power, delay. The results reveal that the proposed 128-bit Parallel Prefix adder has the least power delay product when compared with its peer existing adder structures (ripple carry adder, carry save adders). Simulation results are verified using Xilinx 14.3 software.

Key Words: dot operator, power delay product, kogge-stone, carry save adder, fan-out.

I. INTRODUCTION

VLSI Integer adders find applications in Arithmetic and Logic units (ALU's), microprocessors and memory addressing units. Speed of the adder often decides the minimum clock cycle time in a microprocessor. The need for a Parallel Prefix adder is that it is primarily fast when compared with ripple carry adders. *Parallel prefix adders have been established as the most efficient circuits for binary addition. Their regular structure and fast performance makes them particularly attractive for VLSI implementation*. The classical parallel prefix adder structures presented in the literature over the years optimize for logic depth, area, and fan-out and interconnect count of logic circuits. The need for a Parallel Prefix adder is that it is primarily fast when compared with ripple carry adders. Parallel Prefix adders (PPA) are family of adders derived from the commonly known carry look ahead adders. These adders are best suited for adders with wider word lengths. PPA circuits use a tree network to reduce the latency to *O*(log2 *n*) where 'n' represents the number of bits. Parallel Prefix Adders (PPA) is variations of the well-known carry look ahead adder (CLA).

Manuscript published on 30 August 2013.
* Correspondence Author (s)

Mrs. M.Vijayalaxmi, M.E., Associate Professor in ECE department of Sree kalahasteeswara Institute of Technology, Srikalahasti, Andhra Pradesh, India.
P.Annapurna Bai, Master of Technology (M.Tech) in DECS at Sreekalahasteeswara Institute of Technology, Srikalahasti, Andhra Pradesh, India.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

The difference between a CLA and a PPA lies in the second stage which is responsible for the generation of the carry signals of the binary addition. A parallel Prefix Addition is generally a three step process. The first step involves the creation of generate (gi) and propagate (pi) signals for the input operand bits. The second step involves the generation of carry signals and finally a simple adder to generate sum. The three stage structure of carry look ahead adder and parallel prefix adder is shown in fig 1.

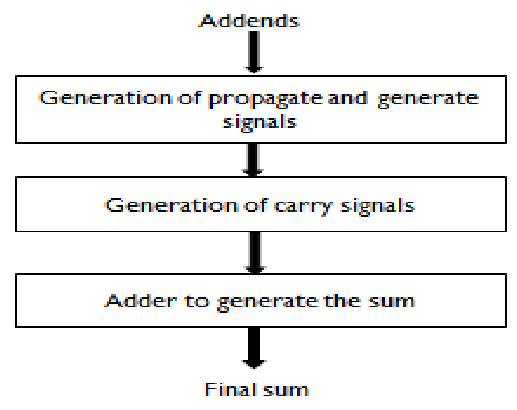

Addends
Generation of propagate and generate signals
Generation of carry signals
Adder to generate the sum
Final sum

Figure: 1 three stage structure of the carry look ahead and parallel prefix adder.

II. PREFIX OPERATORS

Parallel-prefix adders, also known as carry-tree adders, pre-compute the propagate and generate signals. These signals are variously combined using the fundamental carry operator (fco). Beside fundamental carry operator also known as black operator or dot operator as shown in figure 2 ,there is another component called buffer component which translates the generate and propagate signals. The two operators are shown in the figure:

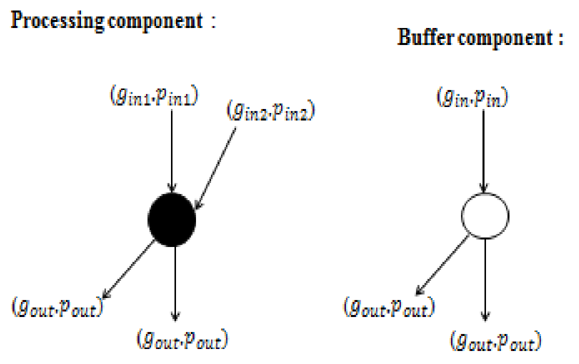


Figure 2. The components used in parallel prefix adders

*Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.*

In this equation, " ● " is applied on two pairs of bits (gin1, pin1) and (gin2, pin2). These pairs represent generate and propagate signals used in the addition. The output of the operator is a new pair of bits generated as described in equation below

$$(g_{out}, p_{out}) = (g_{in1} + p_{in1} \cdot g_{in2}, p_{in1} \cdot p_{in2})$$

The buffer component output is given as

$$(g_{out}, p_{out}) = (g_{in1}, p_{in1})$$

III. EXISTING PARALLEL PREFIX ADDERS

The arrangement of the prefix network specifies the type of the PPA.It is apparent that a key advantage of the tree structured adder is that the critical path due to the carry delay is on the order of log2N for an N-bit wide adder. The arrangements of the prefix network gives rise to various families of adders.There are many parallel-prefix adders that have been invented so far. Among them Brent-kung, Ladner-Fisher, Han-Carlson, Knowles, and Sklansky adders were widely known parallel prefix adders.

There exist various architectures for the carry calculation part. Tradeoffs in these architecture involves

- Area of the Adder
- Its depth
- The fan out of the nodes
- The overall wiring network.

The biggest difference between the full adder and parallel prefix adder is that in the full adder, summation and carry calculation is done in the same one bit block but in the prefix adder, summation and carry calculation are separated from the bit block and all calculation is treated as a whole in the carry graph. The carry graph uses the prefix circuit and this is the origin of the name, "Prefix Adder".

A. Comparisons

The ripple carry adder is a digital circuit which adds two N-bit binary numbers. Full adders are connected in a chain fashion the carry output of first full adder is given as carry input to the second one and so on. This kind of adder is typically known as Ripple Carry Adder because carry ripples to next full adder. Ripple Carry Adder is slowest among all the adders because every full adder must wait till the previous full adder generates the carry bit for its input. The delay is more.

Carry save adder computes the sum of three or more n-bit numbers in binary. The output of the carry save adder is two numbers of same dimensions as inputs one is sum sequence and another is sequence of carry bits. On adding the sequence of both sum and carry we will get the result but the result is not in the form of binary. We have to convert the result. Suppose on adding three number sequences we got the sum and carry sequences like 1101 and 1111 respectively the result of the carry save adder is 2212 we have to multiply the each bit in powers of 2 and add the adjacent bits produces the final result.

```
    1    1    0    1 ……………………… sum
    1    1    1    1 ………………………..carry
_____________________________________________
    2    2    1    2 ……………………..result
    8    4    2    1……………………powers of 2
_____________________________________________
  16 + 8  + 2   +2 =  28………………..final result.
```

The results will produced in a single tick of clock because there is no waiting for another bit entry means no carry propagation which implies reduction in delay and fast when compared to ripple carry adder. One disadvantage is we have to convert the result obtained by adding sum and carry sequences because the result is unambiguous.

This can be avoided by the parallel prefix adders and speed will be increased than carry save adders. There are different types of parallel prefix adders. The Sklansky adder presents a minimum depth prefix network at the cost of increased fan-out for certain computation nodes. The algorithm invented by Kogge-Stone has both optimal depth and low fan-out but produces massively complex circuit realizations and also account for large number of interconnects. Brent-Kung adder has the merit of minimal number of computation nodes, which yields in reduced area but structure has maximum depth which yields slight increase in latency when compared with other structures. The Han-Carlson adder combines Brent-Kung and Kogge-Stone structures to achieve a balance between logic depths and interconnect count. Knowles presented a class of logarithmic adders with minimum depth by allowing the fan-out to grow. Ladner and Fischer proposed a general method to construct a prefix network with slightly higher depth when compared with Sklansky topology but achieved some merit by reducing the maximum fan-out for computation nodes in the critical path. Related work on PPA literature such as Ling adder, achieve improved performance gains by changing the equation of the dot operator '•'. Taxonomy of classical Prefix Parallel Adders based on fan-out, interconnect count and depth.

In the proposed system kogge-stone adder is used because it has minimum fan out of 1 at each node (implies faster performance) and low depth (less calculation time).

B. Kogge-stone adder

The Kogge–Stone adder is a parallel prefix form carry look-ahead adder. It generates the carry signals in O(log n) time, and is widely considered the fastest adder design possible. It is the common design for high-performance adders in industry. It takes more area to implement than the Brent–Kung adder, but has a lower fan-out at each stage, which increases performance. Wiring congestion is often a problem for Kogge–Stone adders as well.

An example of a 4-bit Kogge–Stone adder is shown to the right. Each vertical stage produces a "propagate" and a "generate" bit, as shown. The culminating generate bits (the carries) are produced in the last stage (vertically), and these bits are XOR'd with the initial propagate after the input (the square boxes) to produce the sum bits. E.g., the first (least-significant) sum bit is calculated by XORing the propagate in the farthest-right square box (a "1") with the carry-in (a "0"), producing a "1". The second bit is calculated by XORing the propagate in second box from the right (a "0") with C0 (a "0"), producing a "0".

Inputs: A = 1001 and B = 1100

Outputs: sum =1010

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

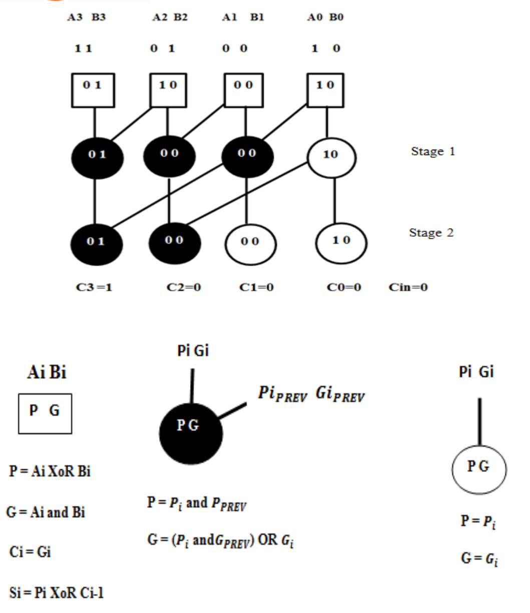


Fig 3: 4 bit kogge-stone adder

In a 4 bit adder like the one shown in the picture to the right, there are 5 outputs. Below is the expansion:
S0 = (A0 XOR B0) XOR C_{IN}
S1 = (A1 XOR B1) XOR (A0 AND B0)
S2 = (A2 XOR B2) XOR (((A1 XOR B1) AND (A0 AND B0)) OR (A1 AND B1))
S3 = (A3 XOR B3) XOR ((((A2 XOR B2) AND (A1 XOR B1)) AND (A0 AND B0)) OR (((A2 XOR B2) AND (A1 AND B1)) OR (A2 AND B2)))
S4 = (A4 XOR B4) XOR ((((A3 XOR B3) AND (A2 XOR B2)) AND (A1 AND B1)) OR (((A3 XOR B3) AND (A2 AND B2)) OR (A3 AND B3)))

IV.PROPOSED 128 BIT KOGGE-STONE ADDER

128- Bit kogge-stone adder is shown in the figure. In this architecture a two 128 bits are added using a kogge-stone adder. The total number of stages present in the 128 bit kogge-stone adder is 7 stages (log *n*), where *n*= 128 the total number of input bits.

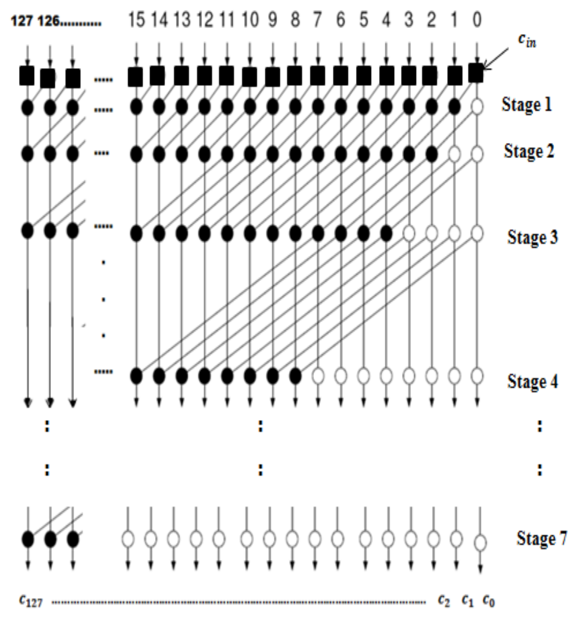
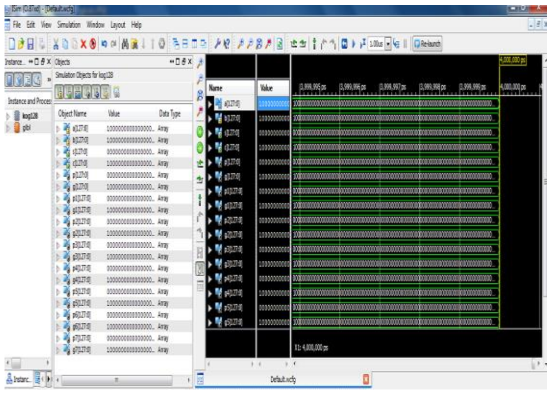


Fig 5: 128 bit kogge-stone adder's carry generation stage block diagram

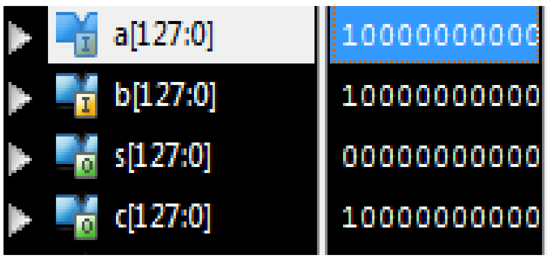
In the figure, the kogge-stone adder operation is shown for 128 bit but not fully because of clumsiness. From LSB to MSB we can observe that in each stage the black nodes are reducing or shifted to 2^{l-1} , (where l = stage number) horizontally and the reduced black nodes are inserted with white nodes. The black nodes are reduced finally to $\frac{n}{2}$ (i.e., in the final stage).

V. SIMULATION RESULTS

A. proposed 128 bit parallel prefix adders



In the proposed system we are adding two 128 bits. In the simulation results adding the inputs a and b, the two 128 bit numbers and the results are stored in outputs sum(s) and carry(c). On adding two n bit numbers we get the sum as n+1bit considering carry.



```
a[127:0]   10000000000
b[127:0]   10000000000
s[127:0]   00000000000
c[127:0]   10000000000
```

Here we are forcing the 128 bit value 100000…..0 into 'a' and another 128 bit value 10000000…..0 into 'b'. on adding these two inputs we get the sum value of 128 bit in 's' as 00000….0 with carry as 10000….0 .

Table 1: Comparison of power and delay

Adder	Delay (ns)	Power(mW)	Power-delay product $(\times 10^{-9})$
Ripple carry	15.293	5241.56	80.15
Carry-save	12.257	5365.17	65.76
Kogge-stone	10.584	3062.17	32.4

The comparison of kogge-stone adder, ripple carry adder and carry save adder for 128 bits is given in the table in terms of power and delay. The power delay product is less when compared among the adders.

VI. CONCLUSION

A new efficient structure for multiple (n inputs) m-bit addition is proposed which is based on adding n inputs bit by bit from LSB to MSB in a manner that each stage generates carries which should be given just to the next higher significant stage, i.e. the operation of each stage is dependent only to a lower significant stage. This structure results in an extreme reduction in hardware. The hardware is reduced since it is not proportional to the number of bits of inputs. The circuit is implemented for adding 128-bit inputs. The results show that our methodology of addition performs the addition with a least delay and power when comparable to ripple and carry save adder at a reduced cost, and high speed.

ACKNOWLEDGEMENT

P.Annapurna bai, would like to thank Mrs. M.Vijaya laxmi M.E.,ECE Department who had been guiding throughout the project and supporting me in giving technical ideas about the paper and motivating me to complete the work efficiently and successfully.

REFERENCES

1. P.Ramanathan, P.T.Vanathi, "Novel Power Delay Optimized 32-bitParallel Prefix Adder for High Speed Computing", International Journal of Recent Trends in Engineering, Vol 2, No. 6, November 2009.
2. R. Zimmermann, Binary Adder Architectures for Cell-Based VLSI and their Synthesis, ETH Dissertation 12480, Swiss Federal Institute of Technology, 1997.
3. David Harris, "A Taxonomy of parallel prefix networks," Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers Pacific Grove, California, pp.2213-2217, November 2003.
4. Knowles, "A family of adders", Proceedings of the 15th IEEE Symposium on Computer Arithmetic. Vail, Colorado, pp.277-281, June2001.
5. .Ramanathan, P.T.Vanathi, "A Novel Logarithmic Prefix Adder with Minimized Power Delay Product", Journal of Scientific & Industrial Research, Vol. 69, January 2010, pp. 17-20.
6. R. Ladner and M. Fischer, "Parallel prefix computation," Journal of ACM. La Jolla, CA, vol.27, no.4, pp. 831-838, October 1980.
7. Andrew Beaumont-Smith and Cheng-Chew Lim, "Parallel Prefix Adder Design", Department of Electrical and Electronic Engineering, the University of Adelaide, 2001.
8. J. Sklansky, "Conditional sum addition logic," IRE Transactions on Electronic computers. New York, vol. EC- 9, pp. 226-231, June 1960.
9. P.Kogge and H.Stone, "A parallel algorithm for the efficient solution of a general class of recurrence relations," IEEE Transactions on Computers, vol. C-22, no.8, pp.786-793, August 1973.

Mrs. M.Vijayalaxmi,M.E., is currently working as an Associate Professor in ECE department of Sree kalahasteeswara Institute of Technology,Srikalahasti.her research areas are wireless communications,VLSI signal processing and 6 international journals published.

P.Annapurna bai completed her B.tech in Electronics and Communication Engineering from Gokula Krishna college of Engineering, sullurpet, Nellore, Andhra Pradesh, India in 2011.she is pursuing her Master of Technology (M.Tech) in DECS at Sreekalahasteeswara Institute of Technology, Srikalahasti, and Andhra Pradesh, India. Her interest includes digital design, VLSI testing.