Journal of Environmental Science, Computer Science and Engineering & Technology

An International Peer Review E-3 Journal of Sciences and Technology

Available online atwww.jecet.org

Section C: Engineering & Technology

Review Article

Analysis of 16 & 32 Bit Kogge Stone Adder Using Xilinx Tool

Jasbir Kaur[1] and Pawan Kumar[2]

[1]E& ECE Department, PEC University of Technology, Chandigarh, India

[2]ME Student, VLSI, PEC University of Technology, Chandigarh, India

Received: 30 May 2014; **Revised:** 21 June 2014; **Accepted:** 28 June 2014

Abstract: An important component of digital computers is adders. Adders are used in many different parts of the digital computer. They are not only used in the Arithmetic Logic Unit (ALU) but also in address calculation. Adders are also used in multipliers and other functional units. One of the Most Prominent adders In VLSI Industry is Parallel Prefix adders[1] .The Parallel Prefix Adder (PPA) is one of the fastest types of adder that had been created and developed. One of the common types of parallel prefix adder is Kogge Stone adder. By using the Xilinx 14.1 software, the designs for Kogge Stone adders were developed. In this paper, Different bit of Kogge Stone adder structures can be used to execute addition and is widely used in the industry for high performance arithmetic circuits. This paper focuses on the implementation and simulation of 16-bit and 32 bit Kogge adder based on Verilog code and compared for their performance in Xilinx. Hence, this paper is significant in showing which of the adder being tested perform better in terms of computational delay based on different sizes of bits.

Keywords: Addition, Kogge Stone Adder (KSA), Parallel Prefix adders (PPA), Simulation, Xilinx.

INTRODUCTION

A Parallel Prefix Adder (PPA) is equivalent to the CLA adder. The two differ in the way their carry generation block is implemented Parallel prefix adders are fastest adders and these are used for high performance arithmetic circuits in industries. The meaning of Parallel Prefix Operation is:-

Prefix: The outcome of the operation depends on the initial inputs.

Parallel: Involves the execution of an operation in parallel. This is done by segmentation into smaller pieces that are computed in parallel.

Operation: Any arbitrary primitive operator "°" that is associative is parallelizable It is fast because the processing is accomplished in a parallel fashion.

For a large number of applications, the speed critical computation block includes adders: either as stand-alone blocks or integrated N bit multiplier architectures. Due to this, specialized speed optimized adder architectures are required for high performance systems. The Main Component is the arithmetic Unit of ALU. In ALU Most of the Calculations in Multiplier is done with the help of adder. PPA can be divided into three main parts,

Pre-possessing stage: In this stage we compute, generate and propagate signals to each pair of inputs A and B. These signals are given by the logic equations (1) & (2):

Pi=Ai xor Bi.............................. (1)

Gi = Ai and Bi............................ (2)

Carry generation network in this stage we compute carries corresponding to each bit. Execution of these operations is carried out in parallel [5]. After the computation of carries in parallel they are segmented into smaller pieces. It uses carry propagate and generate as intermediate signals which are given by the logic equations (3(i)& 3(ii)):

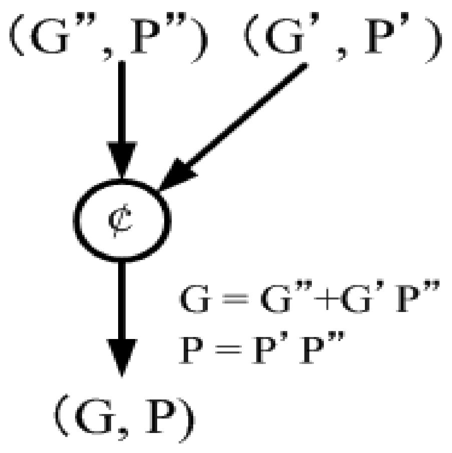

Fig. 1: Carry Operator

The operations involved in this figure are given as:

P=P' and P''............................ (3(i))

G= (P' and G'') or G'................ (3(ii))

Post processing: This is the final step to compute the summation of

Input bits. It is common for all adders and the sum

Bits are computed by logic equation (5) :-

Ci-1=(Pi and Cin) or Gi.................. (4)

Si=Pi xor Ci-1............................... (5)

Carry look-ahead adder (CLA): The main idea behind carry look-ahead addition is an attempt to generate all incoming carries in parallel and avoid waiting until the correct carry propagates from the stage (FA) of the adder where it has been generated. The carry, Ci+1, produced at the ith stage is given as follows:

$$c_{i+1} = x_i y_i + (x_i \oplus y_i) c_i \, .$$

The equation can be interpreted as stating that there is a carry either if one is generated at that stage or if one is propagated from the preceding stage. In other words, a carry is generated if both operand bits are 1, and an incoming carry is propagated if one of the operand bits is 1 and the other is 0. Therefore, let Gi and Pi denote the generation and propagation at the ith stage, we have:

$$G_i = x_i y_i ,$$
$$P_i = x_i \oplus y_i ,$$
$$c_{i+1} = G_i + P_i c_i ,$$

for operand bit xi and yi and carry-in ci. These expressions allow us to calculate all the carries in parallel from the operands. For example, the carries for a 4-bit adder are given as

$$c_0 = c_0 ,$$
$$c_1 = G_0 + c_0 P_0 ,$$
$$c_2 = G_1 + G_0 P_1 + c_0 P_0 P_1 ,$$
$$c_3 = G_2 + G_1 P_2 + G_0 P_1 P_2 + c_0 P_0 P_1 P_2 ,$$
$$c_4 = G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + c_0 P_0 P_1 P_2 P_3 .$$

The equations of the well-known CLA adder can be formulated as a parallel prefix problem by employing a special operator "°".This operator is associative hence it can be implemented in a parallel fashion

Kogge Stone Adder: Kogge-stone adder is a parallel prefix form of Carry Look-ahead Adder. Kogge-Stone adder can be represented as a parallel prefix graph consisting of carry operator nodes. The time required to generate carry signals in this prefix adder is *o* (log n). It is the fastest adder with focus on design time and is the common choice for high performance adders in industry. The better performance of Kogge-Stone adder is because of its minimum logic depth and bounded fan-out. It is the common design for high-performance adders in industry. It has a lower fan-out at each stage, which increases performance .An example of an 8 -bit Kogge–Stone adder is shown in **Fig.2**

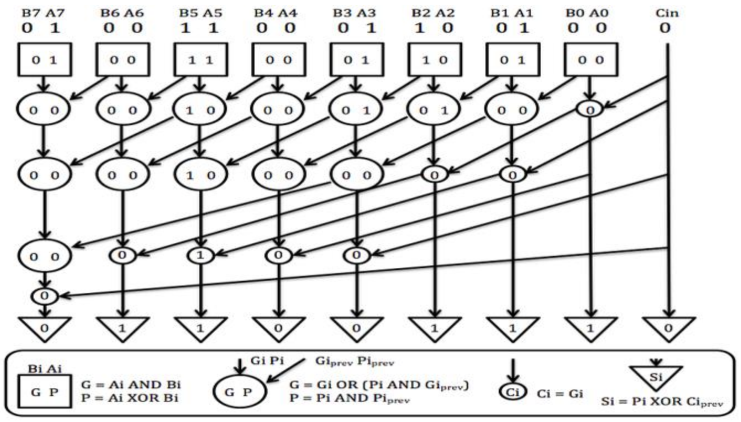

Fig. 2: 8 bit Kogge Stone adder

IV. IMPLEMENTATION

The programming environment for implementing the circuit is based on Verilog HDL. In implementation of this design, Prefix adder is designed for 8-bits using Data Flow Style of modelling and is implemented. The adder is divided into 3 sections i.e. upper, middle and lower sections. The Comparison Different Bit of the KSA adder is given in **Table 1**

Table- 1: Comparison of Different Parameters

Adders	No. of Slice LUTS used	Logic Level	Delay (ns)
16 bit KSA	48	8	12.84
32 bit KSA	134	8	11.26

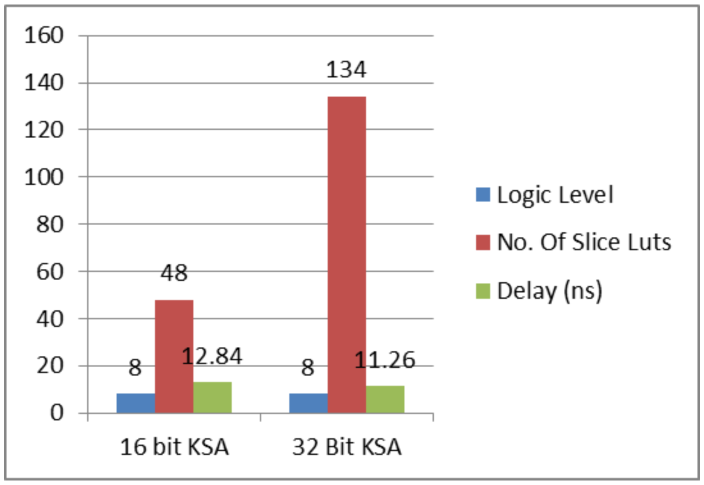

Fig. 3: Comparison of Logic Level, Slice LUTS & Delay of Different Bit KSA

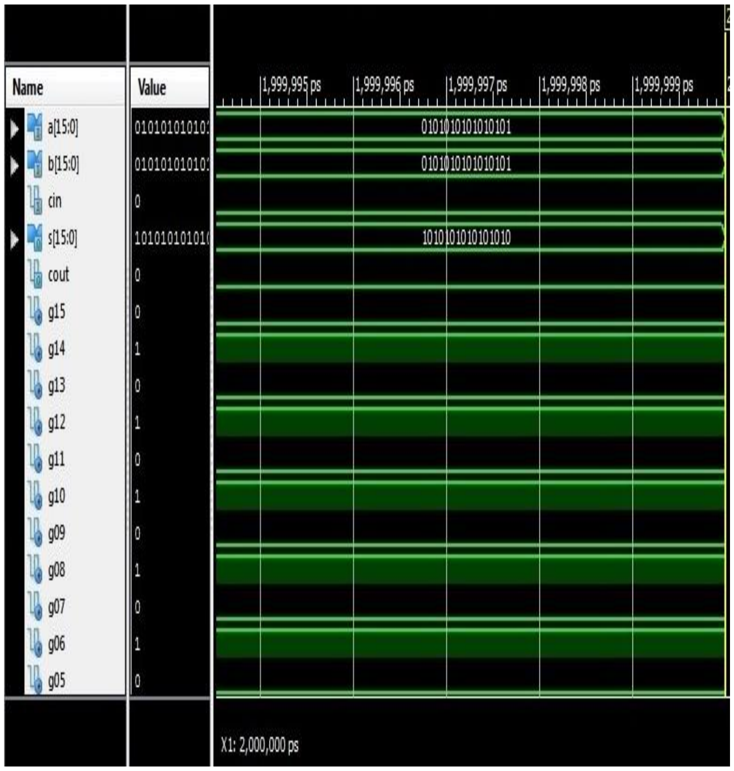

Fig. 4: Simulation Result of 16 bit Kogge Stone adder

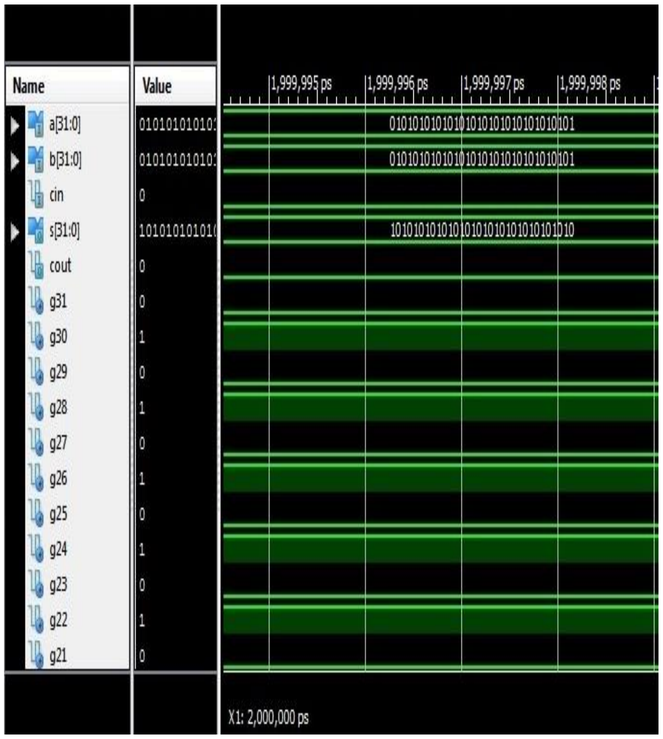

Fig. 5: Simulation Results of 32 bit Kogge Stone adder

CONCLUSION

Simulation results in this project come in the form of Register Transfer Level (RTL) diagram, Functional vector waveform outcome and classic timing analysis. Functional vector waveform outcome are produced by selecting random bit values and add up to produce the sum and carry bits. Timing analysis can be obtained by viewing the summary of the classic timing analysis after compiling the whole project. The simulations are done by using the functions that is included in the Xilinx 14.1 design software. Simulation analysis is prepared by viewing the results from the simulated Verilog code. Analysis of the simulation is performed once the desired simulation outcome is obtained [8]. The analysis of the (PPA) Parallel Prefix Kogge Stone adder is conducted by viewing the time delay produced different bits of KSA adders in performing bits addition finally, the PPA comparison will be made once all two simulation results are analysed. (KSA) will be compared at this stage and will be conducted in its bit category. The comparisons will be based on the computational speed or also known as time propagation delay.

REFERENCES

1. P.M. Kogge and H.S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,” IEEE Trans. Computers, vol. 22, no. 8, pp. 786-792, Aug. 1973.
2. D. Harris, “A Taxonomy of Parallel Prefix Networks,” in Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–7, 2003
3. J.M. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits,A Design Perspective, 2nd Prentice Hall, Englewood Cliffs, NJ, 2002.
4. B. Parhami, Computer Computer Arithmetic Algorithm and Hardware Designs, Oxford University Press, 2000.
5. R. P. Brent and H. T. Kung, “A regular layout for parallel adders,” IEEE Trans. Comput., vol. C-31, pp.260-264, 1982.
6. N.H.E.Weste and D.Harris, CMOS VLSI Design, 4thedition, Pearson–Addison-Wesley, 2011.
7. K. Vitoroulis and A. J. Al-Khalili, “Performance of Parallel Prefix Adders Implemented with Technology,” IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007.
8. Y. Choi, “Parallel Prefix Adder Design,” Proc. 17thIEEE Symposium on Computer Arithmetic, pp 90-98, 27th June 2005.

***Corresponding Author: Pawan Kumar**

ME Student, VLSI, PEC University of Technology, Chandigarh, India