VLSI implementation of Kogge Stone Adder Using ZFC for high speed and low area applications

[1]S BABA FARIDDIN, [2]Dr.RAHUL MISHRA
[1]Research Scholar, Dept of ECE, Dr. A.P.J. Abdul Kalam University, Arandia, Indore, M.P, India.
[2]Professor & H.O.D, Dept of ECE, Dr. A.P.J. Abdul Kalam University, Arandia, Indore, M.P, India.

ABSTRACT: **Adder is most important component in digital circuit designs and processors. To improve the performance of digital system and VLSI systems adder is the main component. Ripple carry adder and carry look ahead adder are the conventional methods which are studied in detail manner. By using two N-bit numbers addition operation is performed using ripple carry adder. But the main dis-advantage of the ripple carry adder is when performing the addition operation for some time carry bits have to wait. Because of this delay will be increased. To overcome this kogge stone adder using ZFC logic is introduced. From parallel prefix adder, kogge stone adder is obtained. Therefore in the high performance applications kogge stone adder is most widely used. To occupy less area and to get low delay all the carries are computed in parallel form.**

KEY WORDS: Parallel Prefix Adder, Kogge stone Adder, ZFC (Zero finding logic), Gray Cell and Black Cell.

I. INTRODUCTION

Fastest technologies are developed in present days. In present days, reduction of device size, fast operation and low power consumption are required. The designing of low power VLSI system has more demand in mobile communication. Due to the device designed by designer with high speed, low power consumption and small silicon area, the device is available with low power. ALU (Arithmetic logic unit) and FU (Floating point unit) are the main parts in computations [1]. Logical computations are addition, subtraction, multiplication, division and logical operations are AND, OR, INV and comparison which are processed by Arithmetic logic unit (ALU).

Data path has an important role in digital signal processors and microprocessors because of some characteristics such as power consumption, speed of operation and die-area. The above characteristics depend upon the data path efficiency. Data path contains complex operations are subtraction, addition, division and multiplication [2]. The main important factor is data path performance which is affected by efficient hardware units of complex computations.

In the data path addition is the important executed operation, addition operation contains binary adder to add given numbers. In complex computations such as decimal operations, multiplication and division, adders has important task [3]. To get data path efficiently, the implementation of binary adder should be efficient.

In central processing unit (CPU) crucial element is ALU (Arithmetic logic unit). An adder has important function in ALU and an adder performs not only addition but also performs multiplication, subtraction and decrement/increment. In ALU and general processors to get better performance, efficient adder is needed. From 1950s, for hardware implementation of VLSI arithmetic circuits, research started on efficient adder implementation. In control systems and digital signal processing main operation is the

addition. The properties of system or processor like accuracy and speed depends upon the performance of adder. To execute the addition of numbers, adder is used which is a digital circuit. Different processors and computers contains ALU in which adder is used. To reduce different parameters, different designs have been implemented based on parallel and serial structures. Four elementary operations are performed in binary addition.

The adders can be represented in many forms like BCD (binary coded decimal) and excess-3 code; binary numbers are used in adders to perform the operation. Negative numbers are represented by ones complement or two's complement, for this adder is modified as adder-subtractor. More logic is required to represent signed numbers including basic adder [4-5].

To execute the addition operation, computers contain Arithmetic Logic unit (ALU) in which adders are mostly used. Graphics processing unit (GPU) and central processing unit uses the adders to decrease the redundancy for the graphics applications. In the adders first type is half adder it include two inputs and it provides two outputs such as carry and sum. Next one is full adders which include two inputs with carry input and it provides two outputs such as carry and sum. Both half adder and full adder are used for single bit. To perform multi bit addition, full adders are coupled in parallel. When full adders are coupled in parallel, the sum of each full adder produced as usually, and carry output of each full adder is given to next full adder as carry input and final carry is the carry output.

II. LITEARTURE SURVEY

Er. Aradhana Raju, Richi Patnaik, Ritto Kurian Babu, Purabi Mahato.et.al [6]

VLSI, in modern day technology has seen extensive use of PPA with a better delay performance. These pre-compute the carries and thus have upper hand over the commonly used Ripple Carry Adder (RCA). Addition has been an indispensible operation in most of the widely used applications. We present the 4,8,16 and 32 bit of different adders-Carry Look Ahead (CLA), Carry Save Adder (CSA), Kogge Stone Adder (KSA), Sparse Kogge Stone Adder (SKSA), Brent Kung Adder (BKA), Sklansky Adder, Lander Fischer Adder (LFA) and Han Carlson Adder (HCA). They have been categorized and ranked as per delay, device utilization and cell usage. These adders are implemented in VHSIC Hardware Description Language (VHDL) using Xilinx Integrated Software Environment (ISE) 9.2i Design Suite.

Sudheer Kumar Yezerla, B Rajendra Naik.et.al [7]

In Very Large Scale Integration (VLSI) designs, Parallel prefix adders (PPA) have the better delay performance. This paper investigates four types of PPA's (Kogge Stone Adder (KSA), Spanning Tree Adder (STA), Brent Kung Adder (BKA) and Sparse Kogge Stone Adder (SKA)). Additionally Ripple Carry Adder (RCA), Carry Look ahead Adder (CLA) and Carry Skip Adder (CSA) are also investigated. These adders are implemented in verilog Hardware Description Language (HDL) using Xilinx Integrated Software Environment (ISE) 13.2 Design Suite. These designs are implemented in Xilinx Spartan 6 Field Programmable Gate Arrays (FPGA). Delay and area are measured using Power analyzer and all these adder's delay, power and area are investigated and compared finally.

Dayu Wang1, Xiaoping Cui2, Xiaojing Wang.et.al [8]

Parallel-prefix computation provides a highly efficient solution to binary addition problem. This paper proposes an advanced design based on parallel-prefix Ling adder. In order to further improve the Ling adder's performance, the pre-processing block and carry propagation block are all optimized to reduce the delay path for sum generation. Experimental results indicate that the proposed adder has an improvement of 21 percent time delay and 25 percent area compared to the traditional Ling adders.

Tso-Bing Juang, Pramod Kumar Meher, Chung-Chun Kuan.et.al [9]
Efficient addition of binary numbers plays a very important role in the design of dedicated as well as general purpose processors for the implementation of arithmetic and logic units, branch decision, and floating-point operations, address generations, etc. Several methods have been reported in the literature for the fast and hardware-efficient realization of binary additions. Among these methods, parallel-prefix addition schemes have received much of attentions, since they provide many design choices for delay/area-efficient implementations and optimization of tradeoffs. In this paper, we have proposed area-efficient approach for the design of parallel-prefix Ling adders. We have achieved the area efficiency by computing the real carries, based on the Ling carries produced by the lower bit positions. Using the proposed method, the number of logic levels can be reduced by one, which leads to reduction of delay as well as significant saving of area complexity of the adder. We have implemented the proposed adders using 0.18μm CMOS technology; and from the synthesis results, we find that our proposed adders could achieve up to 35% saving of area over the previously reported parallel-prefix Ling adders under the same delay constraints.

M.Moghaddam, M. B. Ghaznavi-Ghoushchi.et.al [10]
In this paper a relation between graph energy and electrical energy or power consumption with graph nodes and edges for PPA structures is introduced. The Sklansky PPA is selected as the target PPA and its recursion steps limited. This reduced the internal nodes of the PPA. The reduction of nodes and edges and limiting the recursive stages to maximum 8 steps, resulted in proposed adder I and using special dot and semi-dot operators for graph nodes in proposed adder I led to proposed adder II. The number of transistors, total power and critical delay path of our proposed adder II are reduced about 31%, 29% and 7% versus Sklansky adder respectively. Finally the power-delay-product (PDP) of proposed PPA II comes with about 35% improvement compared with Sklansky adder.

III. KOGGE STONE ADDER USING ZFC

The below figure (1) shows the architecture of Kogge stone adder using ZFC. In this we use mainly black cell and grey cell and parallel carry architecture. The entire addition process is performed in three stages. The pre-processing stage generates the propagate and generate signals. The carry generation stage is controlled by the intermediate signals. At last the post processing stage gives output as sum and carry.

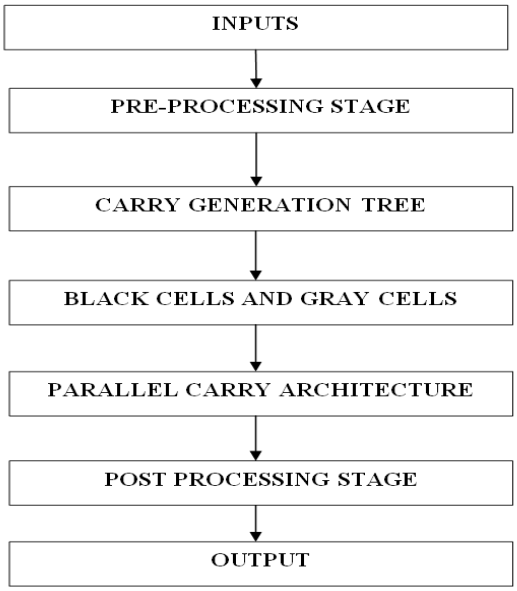

Fig. 1: STRUCTURE OF KOGGE STONE ADDER

Three stages are presented in the Kogge Stone Adder (KSA) is explained in detail manner.

1. Pre-Processing Stage:

In this stage, propagate signals and generate signals are manipulated to pair of each inputs A and B. Propagate signal and generate signal are represented as

$$Pi = Ai \text{ XOR } Bi$$
$$Gi = Ai \text{ AND } Bi$$

2. Carry Generation Network:

In carry generation stage the calculation is performed based on the bits and carries obtained. The entire operation is performed in the form of parallel. Generate and propagate signals are obtained from the intermediate signals. The below equations shows the propagate and generate signals:

$$Pi:j = Pi:k \text{ AND } Pk-1:j$$
$$Gi:j = Gi:k \text{ OR } (Pi:k \text{ AND } Gk-1:j)$$

Black/gray cells implement the given two equations, which will be usually used in the following discussion on prefix trees.

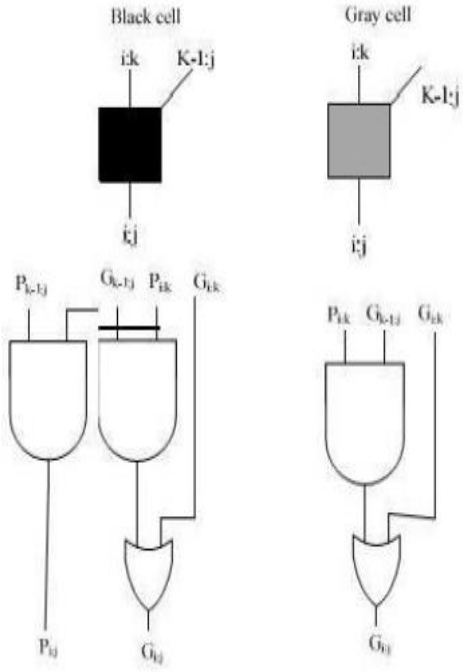

Fig. 2: GRAY CELL & BLACK CELL

3. Post Processing Stage:

In post processing stage the calculation is performed based on the input bits. From post processing stage sum and carry is generated. The below equations shows the sum and carry equations:

$$Ci = (Pi \text{ AND } Cin) \text{ OR } Gi$$
$$Si = Pi \text{ XOR } Ci\text{-}1$$

In applications of high speed circuits, very useful adder is Kogge-Stone adder. Kogge-Stone adder is designed based on the power and area.

Structure delay= $\log_2 n$

Number of computation nodes= $[(n)(\log_2 n)-n+1]$

The fan-out problem is recognised by KSA and recursive doubling algorithm is introduced. Idem potency property is used in KSA which controls the fan-out. Cost is increased due to number of lateral wires at every stage. Here massive overlap is occurred between prefix sub-terms.

The implementation of KSA occupies more area. At every stage, fan-out is less in Kogge-Stone adder; it increases performance for CMOS process nodes. In Kogge-Stone adder, the problem is wiring congestion. There is no wiring congestion in Lynch–Swartzlander design, it provides lower fan-out and it is small. Manchester carry chain implementations support by process nodes.

IV. RESULTS

The below figure (3) shows the RTL schematic of kogge stone adder using ZFC. Basically, RTL schematic is the combination of Inputs and outputs. Here a and b are the inputs which consists of 32 bits and sum and carry are the outputs which consist of 32 bits. Here c0 is the additional carry.

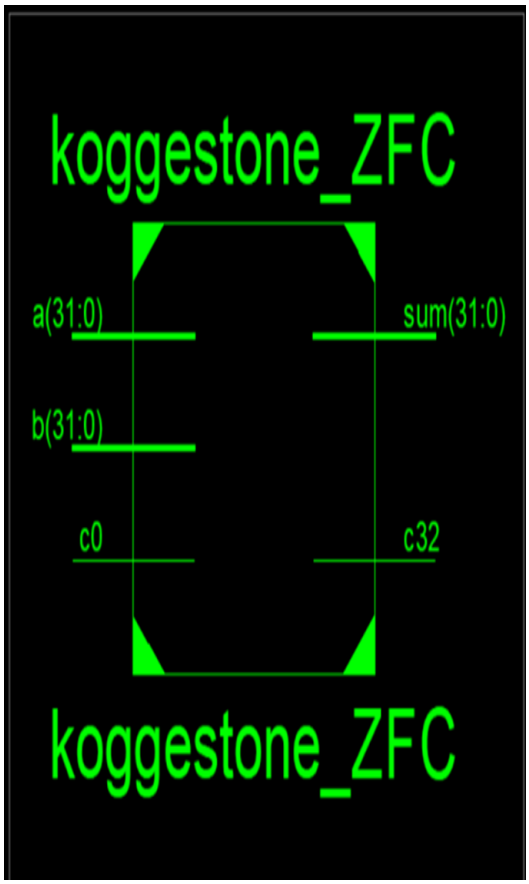

Fig. 3: RTL SCHEMATIC OF KOGGE STONE ADDER USING ZFC

The below figure (4) shows the RTL schematic of Kogge Stone Adder using ZFC. Here RTL is nothing but register transfer level. The technology schematic is the combination of look up tables, truth tables, equations and K-Map. The look up tables in this system is arranged as key value pairs. The truth table will define the logic of the function. K-MAP will define the logic expression from the values given.

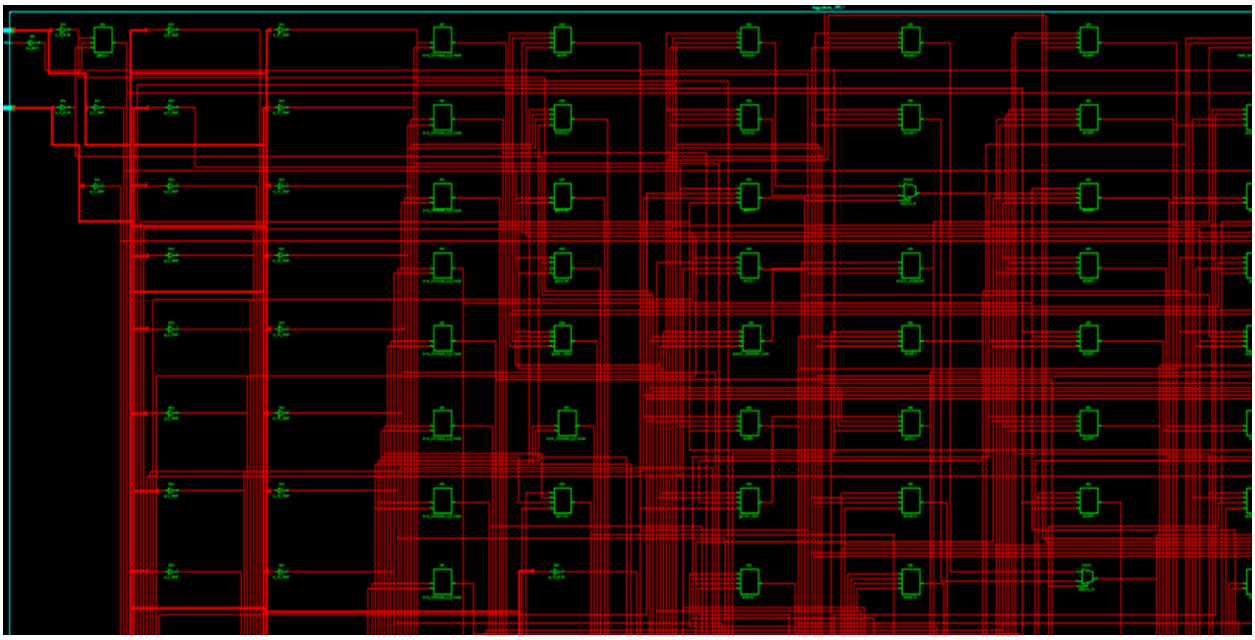

Fig. 4: TECHNOLOGY SCHEMATIC OF KOGGE STONE ADDER USING ZFC

The below figure (5) shows the output waveform of Kogge Stone Adder using ZFC. Here for given outputs corresponding outputs are obtained. Here the input a is given as "01010110101010100101010010001010" and input b is given as "01101010101010100101010101010101" and the obtained output (i.e) sum and carry is obtained as "11000001010101001010100111100000" and "0".

Fig. 5: OUTPUT WAVEFORM OF KOGGE STONE ADDER USING ZFC

The table 1 shows the delay of Kogge Stone Adder using ZFC. Here the delay is divided into three forms they are total delay, route delay and logic delay.

Table 1: RESULT OF KOGGE STONE ADDER USING ZFC

S.NO	PARAMETER	EXISTED VALUE	PROPOSED VALUE
1	Total Delay	36.936 ns	5.535 ns
2	Route Delay	14.295 ns	5.062ns
3	Logic Delay	22.641 ns	0.473 ns

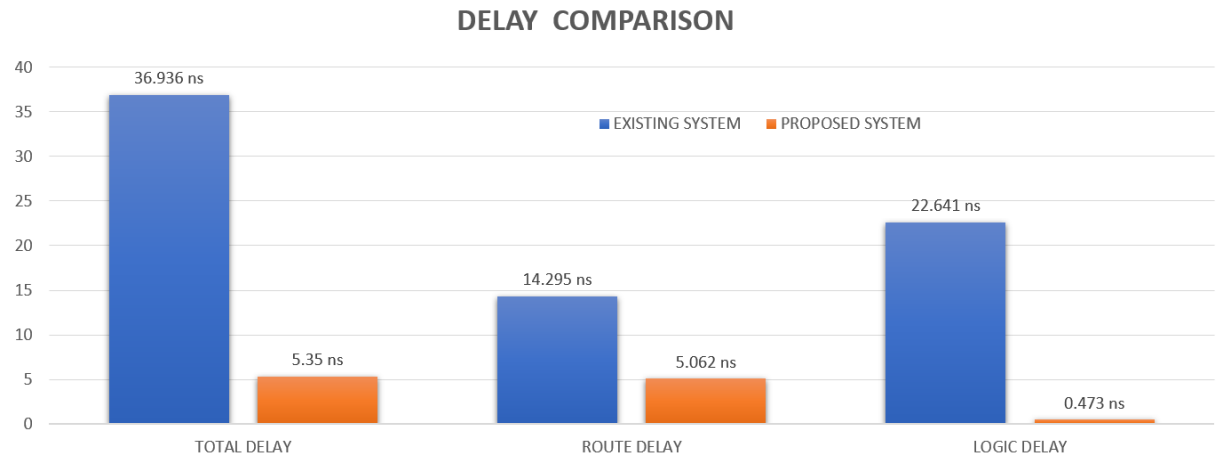

Fig. 6: DELAY COMPARISON

The above figure (6) shows the comparison graph of delay. In this total delay, route delay and logic delay are presented. Compared to the existing system, the delay is reduced in proposed system. Hence the kogge stone adder using effective results compared to others.

V. CONCLUSION

VLSI implementation of Kogge Stone Adder Using ZFC for high speed and low area applications was implemented in this paper. There are three stages in entire structure. In first stage signals of generate and propagate are obtained. In second stage carries are obtained. In last stage sum and carry are obtained. 5.35 ns is the total delay of KSA using ZFC, 5.062 ns is the route delay obtained for KSA using ZFC. 0.473 ns is the logic delay obtained for KSA using ZFC. By using KSA using ZFC effective and efficient results are obtained.

VI. REFERENCES

[1] U Penchalaiah, Siva Kumar VG, "Design of High-Speed and Energy-Efficient Parallel Prefix Kogge Stone Adder", 2018 IEEE Conference.
[2] Aung Myo San , Alexey N. Yakunin , "Reducing the Hardware Complexity of a Parallel Prefix Adder", 978-1-5386-4340-2/18/$31.00©2018 IEEE.
[3] Bhavani Koyada,1 N. Meghana,2 Md. Omair Jaleel3 and Praneet Raj Jeripotula4, "A Comparative Study on Adders", 978-1-5090-4442-9/17/$31.00 c 2017 IEEE.
[4] S.Daphni, K.S. Vijula Grace, " A REVIEW ANALYSIS OF PARALLEL PREFIX ADDERS FOR BETTER PERFORMNCE IN VLSI APPLICATIONS", 978-1-5090-6480-9/17/$31.00©2017 IEEE.

[5] Shaheen Khan , Zainul Abdin Jaffery, " Parallel-prefix modulo adders: A Review", 978-1-5386-4318-1/17/$31.00 ©2017 IEEE.
[6] Er. Aradhana Raju, Richi Patnaik, Ritto Kurian Babu, Purabi Mahato , "Parallel Prefix Adders- A Comparative Study For Fastest Response", 2016 IEEE Conference.
[7] Sudheer Kumar Yezerla , B Rajendra Naik, "Design and Estimation of delay, power and area for Parallel prefix adders", 978-1-4799-2291-8/14/$31.00 ©2014 IEEE.
[8] Dayu Wang1, Xiaoping Cui2, Xiaojing Wang, "Optimized design of Parallel Prefix Ling Adder", 978-1-4577-0321-8/11/$26.00 ©2011 IEEE.
[9] Tso-Bing Juang, Pramod Kumar Meher, Chung-Chun Kuan, "Area-Efficient Parallel-Prefix Ling Adders", 978-1-4244-7456-1/10/$26.00 ©2010 IEEE.
[10] M.Moghaddam, M. B. Ghaznavi-Ghoushchi,"A New Low-Power, Low-area, Parallel Prefix Sklansky Adder with Reduced Inter-Stage Connections Complexity", 2011 IEEE Conference.
[11] Nagaraja Revanna, Earl E. Swartzlander, Jr., " Memristor Adder Design", 978-1-5386-7392-8/18/$31.00 ©2003 IEEE.
[12] Pavan Kumar.M.O.V #1, Kiran.M, " Design Of Optimal Fast Adder", 978-1-5386-7392-8/18/$31.00 ©2002 IEEE.