



# Design and Implementation of different types of efficient parallel prefix adders

Mr. Deepak Raj<sup>1</sup>, Mrs. Sahana K Adyanthaya<sup>2</sup>, Prof. Praveen J<sup>3</sup>, Prof. Raghavendra Rao R<sup>4</sup>

M. Tech 2<sup>nd</sup> year, VLSI Design and Embedded Systems, Dept. of ECE, AIET, Mijar, Moodbidri, Karnataka, India<sup>1</sup>

Assistant Professor, Dept of ECE, AIET Mijar, Moodbidri, Karnataka, India<sup>2</sup>

Sr. Associate Professor, Dept of ECE, AIET Mijar, Moodbidri, Karnataka, India<sup>3,4</sup>

**Abstract:** In Very Large Scale Integration (VLSI) designs, Parallel prefix adders (PPA) have the better delay performance. A parallel prefix adder involves the execution of the operation in parallel which can be obtained by segmentation into smaller pieces. The binary addition is the basic arithmetic operation in digital circuits and it became essential in most of the digital systems including Arithmetic and Logic Unit (ALU), microprocessors and Digital Signal Processing (DSP). At present, the research continues on increasing the adder's delay performance. In this paper the investigation of four types of PPA's (Kogge Stone Adder (KSA), Spanning Tree Adder (STA), Brent Kung Adder (BKA) and Sparse Kogge Stone Adder (SKA)) is done. Additionally Ripple Carry Adder (RCA), Carry Lookahead 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) 13.4 Design Suite. The area, delay and power consumed by all types of PPA's are analyzed. The area of the adder design are given in terms of Look Up Tables (LUT's) and Input Output bounds (IOB's). The adder designs are implemented and delay, power and area of all the adders are investigated.

**Keywords:** PPA, RCA, CLA, SKA, KSA, BKA, STAFFPGA, Delay, Power, Area

## I. INTRODUCTION

Binary adders are one of the most basic and widely used arithmetic operations in modern integrated circuits. They tend to play a critical role in determining the performance of the design. Arithmetic operations are the regular common operations in digital integrated circuits. The simplest circuit adds, subtracts, and multiplies or divides. The computation should be fast and the area consumed by the arithmetic units should be small. These are the two basic requirements for any adder [2].

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently in parallel. There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling. As power consumption and consequently heat generation by computers has become a major concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors. Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with core and multi-processor computers having multiple processing elements within a single machine. Area and Time consumed by the circuit are the basic and important

requirements. Numbers can be represented in digital circuits in various ways. Hence, developing efficient adder architecture is crucial to improving the efficiency of the design. Generally ripple carry adder is used for binary addition. After the design of ripple carry adder several techniques are used for the computation of parallel adders. Carry look ahead adders are based on parallel prefix computation which gives better performance than ripple carry adder. After many years of research, focus is on improving the delay performance of the adder. As such, extensive research continues to be focused on improving the delay performance of the adder. Next, Brent and Kung (BK) designed parallel prefix-computation graph in an area-optimal way and the Kogge-stone (KS) architecture is optimized for timing.

The binary addition is the basic arithmetic operation in digital circuits and it became essential in most of the digital systems including Arithmetic and Logic Unit (ALU), microprocessors and Digital Signal Processing (DSP). At present, the research continues on increasing the adder's delay performance. In many practical applications like mobile and telecommunications, the Speed and power performance improvement in FPGAs is better than microprocessor and DSP's based solutions. Additionally, power is also an important aspect in growing trend of mobile electronics, which makes large-scale use of DSP functions. Because of the Programmability, structure of configurable logic blocks (CLB) and programming



interconnects in FPGAs, Parallel prefix adders have better performance [1].

## II. LITERATURE REVIEW

Sudheer Kumar Yezerla et al.[1] did research on, “Design and Estimation of delay, power and area for Parallel Prefix Adders”, where the design was implemented using Xilinx Virtex 5 FPGA and the adder delays were estimated using Agilent 1692A Logic Analyzer. The 16-bit SKA computes the carries with Black Cells (BC’s) and Grey Cells (GC’s) and terminates with a 4-bit RCA.

The 16-bit STA used also terminates with RCA and uses BC’s and GC’s and Full Adders but has difference in the interconnection between them. The 16-bit KSA designed used 16 BC’s and 15 GC’s with less delay compared to SKA and STA. The 16-bit BKA used 14 BC’s and 11 GC’s which is less compared to KSA and hence has less architecture and occupies less space than KSA. The delay measurements show that SKA and BKA have almost same delay where as STA has better delay results. The efficiency has increased for delay up to 5.77% for RCA and for KSA has improved by 19.28%.

CH. Chimpiraiah et al.[2] work on, “An efficient architecture for Parallel Adders” presents an efficient structure for parallel adders with fast performance which are particularly attractive for VLSI implementations. The simulation and synthesis was done in Xilinx ISE Foundation 9.2i software.

The proposed design was the combination of KSA and BKA and the result is compared with KSA. The area given in terms of slices was reduced from 81 to 63 and the delay from 12.27ns to 11.04ns. The number of logic levels was reduced from 8 to 7 compared to KSA which gives better power performance. V. Krishna Kumari et al. [3] researched on, “Designing and Characterization of Kogge Stone, Sparse Kogge stone, Spanning tree and BrentkungAdders”, and the results were obtained using modelsim 6.4b, Xilinx ISE 10.1i for simulation and synthesis.

The practical issues while testing the adders were analyzed and a circuit to test the adders is designed. The objective of the testing adder circuit was to find the worst path and the input causing it was combination which is stored in ROM.

A.N. Jayanthi et al. [4] work on, “Comparison of performance of high speed VLSI adders”, proposed 16-bit and 64-bit adder design for all the adders and the comparison was made in terms of delay.

The 16-bit Ling adder design proposed has delay reduced to half compared to the 16-bit RCA, but the circuit has limitations on fan-in. The KSA adder scheme reduces logical fan-out in each node but increases the wiring

within the circuit which in turn will have high power consumption. The delay of the 64-bit KSA is 48.064ns and of the Ling adder is 51.841ns. They concluded that several architectures are often close to the optimum performance for the given design environment.

The result obtained shows that the given methodologies lead to the best design at all technological nodes. Neil Burgess [5] in the paper, “Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI”, presented new high-radix ripple carry adder based on Ling’s addition technique. The critical path introduced inverting CMOS cell per stage along carry in and carry out. The fastest of them matches speed of 16-bit prefix adder for only 63% of the area.

Mangesh B Kondalkar et al. [6] work on, “Improved Fault Tolerant Sparse KOGGE Stone ADDER”, proposed a method for error correction due to inherent redundancy in the carry tree and also error detection was possible. The prior work had only error correction which used Kogge Stone configuration. Several enhancements are introduced in the design; the error recovery time is reduced by using a 16-bit register, error correction due to fault in multiple ripple carry adders is included which improves the reliability of the circuit.

Triple Mode redundancy Ripple carry (TMR-RC) is simple and effective method of fault detection and correction but it increases the area overhead and almost results in tripling of the associated power dissipation. A Sparse Kogge-Stone adder which is fully fault tolerant in its lower half (i.e., in the ripple carry adders) was proposed.

The addition of register reduces the error recovery time and also makes it possible to detect and correct the error in multiple RCA.

## III. IMPLEMENTATION OF PARALLEL PREFIX ADDER STRUCTURE

### 3.1. Parallel Prefix Adder Structure

Parallel-prefix structures are found to be common in high performance adders because the delay is logarithmically proportional to the adder width. PPA’s basically consists of 3 stages

- Pre computation
- Prefix stage
- Final computation

The Parallel Prefix adder structure is shown in Fig 3.1.

### 3.2 Implementation stages of PPA

PPA’s basically consists of 3 stages:

#### Pre-Computation Stage

In pre computation stage, propagates and generates are computed for the given inputs using the given equations (1) and (2).

$$P_i = A_i \wedge B_i \quad (1)$$

$$G_i = A_i \cdot B_i \quad (2)$$

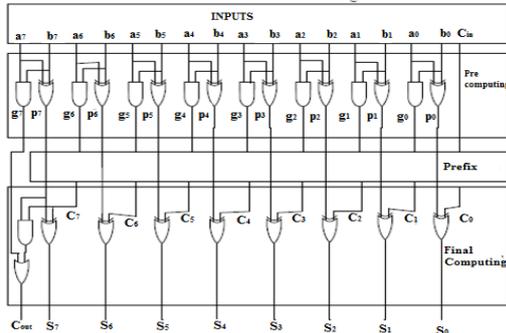


Fig 3.1: Parallel-Prefix structure with carry save notation<sup>[1]</sup>.

**Prefix Stage**

In the prefix stage, group generate/propagate signals are computed at each bit using the given equations. The black cell generates the ordered pair and the grey cell generates only the left signal. The fundamental carry operator is denoted by the symbol “o”. The generate and propagate signals are combined using the fundamental operator as shown in equation (3).

$$(g_L, P_L) o (g_R, P_R) = (g_L + P_L \cdot g_R, P_L \cdot g_R) \quad (3)$$

The black cell and grey cell logic definition are shown in Fig 3.2.

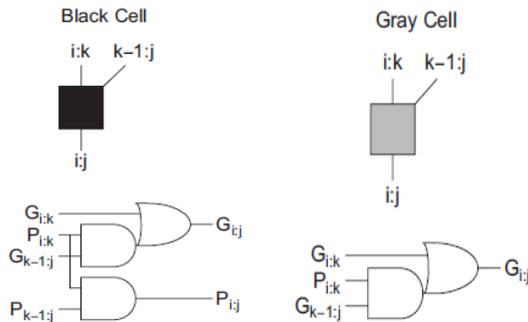


Fig 3.2: Black and Grey Cell logic definition.

From the black and grey cell logic definition, the generate and propagate signals are given by equations (4) and (5).

$$G_{i:k} = G_{ij} + P_{ij} \cdot G_{j-1:k} \quad (4)$$

$$P_{i:k} = P_{ij} \cdot P_{j-1:k} \quad (5)$$

More practically, the equations (4) and (5) can be expressed using a symbol “o” denoted by Brent and Kung. Its function is exactly the same as that of a black cell and is given in (6). The “o” operation will help make the rules of building prefix structures [7].

$$G_{i:k} : P_{i:k} = (G_{ij}, P_{ij}) o (G_{j-1:k}, P_{j-1:k}) \quad (6)$$

**Final Computation**

In the final computation, the sum and carryout are the final output. These are given by equations (7) and (8), where “-1” is the position of carry-input.

The generate/propagate signals can be grouped in different fashion to get the same correct carries. Based on different ways of grouping the generate/propagate signals, different prefix architectures can be created.

$$S_i = P_i \cdot G_{i-1:-1} \quad (7)$$

$$C_{out} = G_{n:-1} \quad (8)$$

**IV. DIFFERENT TYPES OF PPA'S**

**4.1 16-Bit Kogge Stone Adder**

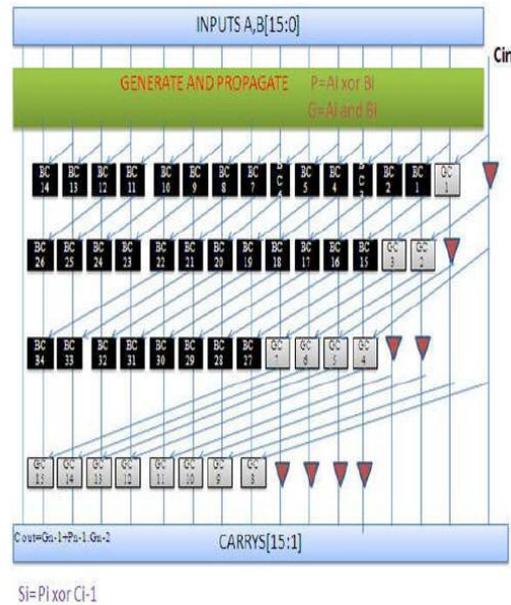


Fig 4.1 16-bit Kogge-Stone Adder<sup>[1]</sup>

Kogge-Stone adder is one among the parallel prefix adders. This has regular layout which makes them favored adder in electronic technology. It has the minimum fan-out. A 16 bit Kogge stone adder is shown in the figure 4.1. The maximum fan-out is 2 in all the logic levels for all width Kogge-stone prefix trees. The key of building any prefix tree is to implement the structure according to the grey cell and black cell logic. The number of stages for a Kogge stone adder is calculated by log2 power N. no of cells is calculated as N (log2 power N – 1) +1. It consists of 34 BC's and 15 GC's.

**4.2 16-bit Sparse Kogge-Stone Adder**

The Sparse Kogge stone adder consists of several small ripple carry adders on its lower part, a carry tree is on its upper part. It terminates with ripple carry adders. Number of carries generated is less in this adder compared to Kogge stone adder.

The function of grey cells and black cells is same as discussed in previous sections. Figure 4.2 shows the block diagram of 16 bit Sparse Kogge Stone adder. Like the sparse Kogge-Stone adder, this design terminates with a 4-bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the Performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders.

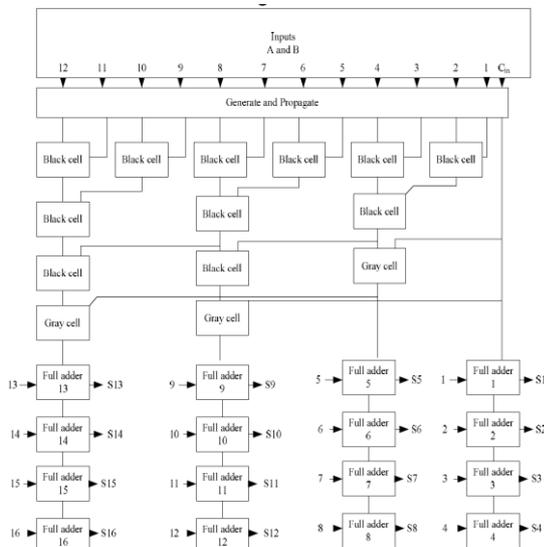


Fig 4.2 16-bit Sparse Kogge-Stone Adder<sup>[1]</sup>

### 4.3 16-bit Spanning Tree Adder

Another carry-tree adder known as the spanning tree carry-lookahead (CLA) adder shown in Fig 4.3 is also examined. Like the sparse Kogge-Stone adder, this design terminates with a 4-bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders. Also of interest for the spanning-tree CLA is its testability features. It also uses the black cells and gray cells and full adder blocks like sparse kogge-stone adders but the difference is the interconnection between them.

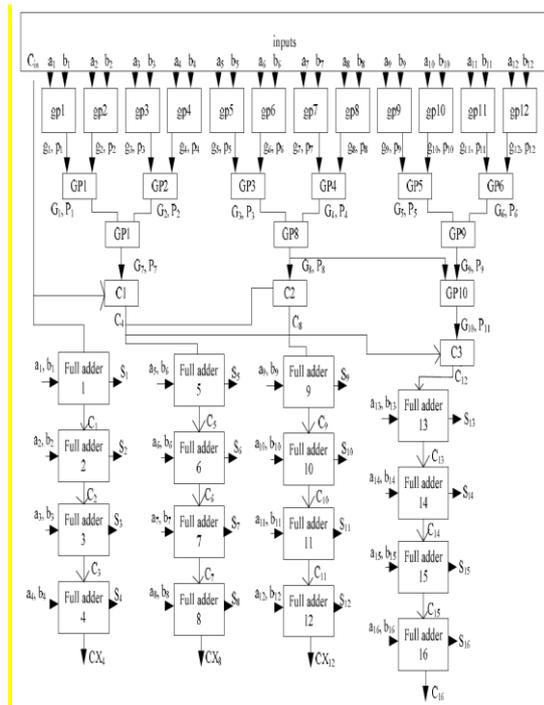


Fig 4.3 16-bit Spanning Tree Adder<sup>[1]</sup>

### 4.4 16-bit Brent Kung Adder

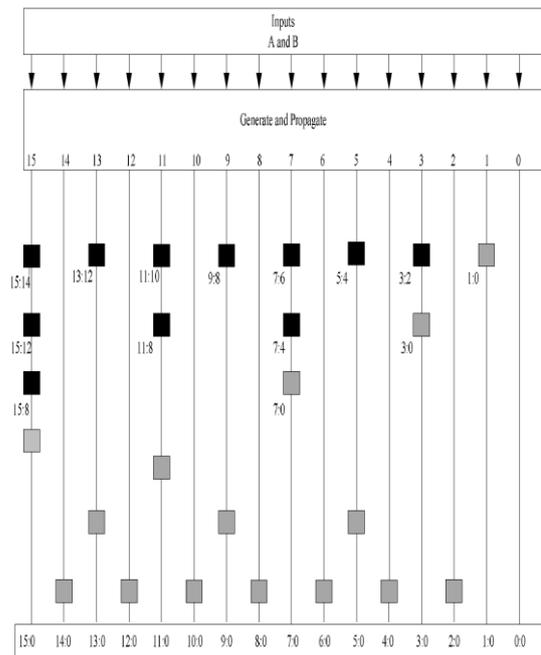


Fig 4.4 16-bit Brent Kung Adder<sup>[1]</sup>

Another carry tree known as BKA which also uses BC's and GC's but less than the KSA. So it takes less area to implement than KSA. The 16 bit BKA uses 14 BC's and 11 GC's but kogge stone uses 36 BC's and 15 GC's. So BKA has less architecture and occupies less area than KSA. The 16 bit BKA is shown in the below figure 4.4. BKA occupies less area than the other 3 adders called SKA, KSA, STA. This adder uses limited number of propagate and generate cells than the other 3 adders. It takes less area to implement than the KSA and has less wiring congestion. The operation of the 16 bit Brent Kung adder is given below [8]. This adder uses less BC's and GC's than Kogge-Stone adder and has the better delay performance.

## V. RESULTS

The simulation results are carried out in ModelSim6.3f design suite and is shown in the fig 5.1.

Messages	0	1	2	3	4
Alodge_stone_16_b/c	0	0	0	0	0
Alodge_stone_16_b/y	0	0	0	0	0
Alodge_stone_16_b/cn	0	0	0	0	0
Alodge_stone_16_b/cum	0	0	0	0	0
Alodge_stone_16_b/cout	0	0	0	0	0
Alodge_stone_16_b/cut/cn	0	0	0	0	0
Alodge_stone_16_b/cut/cum	0	0	0	0	0
Alodge_stone_16_b/cut/cy	0	0	0	0	0
Alodge_stone_16_b/cut/cum	0	0	0	0	0
Alodge_stone_16_b/cut/cn	0	0	0	0	0

Fig 5.1: Simulation result of adder

The device utilization summary report from Xilinx 13.2 gives the area occupied by each of the adders in terms of slices and LUT's when used in SPARTAN 3 Field Programmable Gate Array. For 16-bit adder the number of slices occupied is shown in the Table 5.1. The delay for the maximum path in adder design is summarized in Table 5.2.



Design Utilization Summary (SPARTAN 3)			
Adder	No. of Slices	No. of 4 input LUT's	No. of bounded IOB's
RCA	18	31	49
CKA	21	37	50
CLA	18	32	50
KSA	41	72	50
BKA	21	38	50

- [5]. Neil Burgess, ARM Inc, "Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI", 2011 20th IEEE Symposium on Computer Arithmetic, 1063-6889, © 2011 IEEE.
- [6]. Mangesh B Kondalkar, Arunkumar P Chavan, P Narashimaraja, "Improved Fault Tolerant Sparse KOGGE Stone ADDER", International Journal of Computer Applications (0975 – 8887) Volume 75– No.10, August 2013.
- [7]. Niel H EWeste, David Money Harries, "CMOS VLSI DESIGN – A CIRCUITS AND SYSTEMS PERSPECTIVE", fourth edition.
- [8]. R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp. 260-264, 1982.

Table 5.1: Device Utilization Summary of various adders

Adder	Xilinx 13.2 ISE tool Delay(ns)
RCA	27.34
CSA	33.98
CLA	28.741
KSA	18.588
BKA	23.854

Table 5.2: Delays for various adders

## VI. CONCLUSION AND FUTURE WORK

It is shown that the results obtained for Parallel Prefix Adders are better than the serial adders in terms of delay and at the same time there is a trade-off with the area occupied. The results obtained for carry chain adders at higher bit widths (128 to 256 bits) has higher performance when compared to serial adders. Because the adder is often the critical element which determines to a large part the cycle time and power dissipation for many digital signal processing and cryptographically implementations, it would be worthwhile for future FPGA designs to include an optimized carry path to enable tree based adder designs to be optimized for place and routing. Inclusion of a ROM is being done in few adder designs to find the worst case delay combination of the input. Architectures that include fast carry chains and the possible tradeoffs are investigated. In the future, designs for Spanning Tree Adder(STA) and Sparse Kogge Stone Adder (SKA) are done and delay and area occupied are calculated to check performance enhancement keeping serial adders as reference.

## REFERENCES

- [1]. Sudheer Kumar Yezerla, B Rajendra, "Design and Estimation of delay, power and area for Parallel prefix adders", Proceedings of 2014 RAECs UIET Panjab University Chandigarh, 06 - 08 March, 2014, 978-1-4799-2291-8 ©2014 IEEE.
- [2]. CH. Chimpiraiah, E.V Vijay, "AN EFFICIENT ARCHITECTURE FOR PARALLEL ADDERS", International Journal of VLSI and Embedded Systems-IJVES, Vol 03, Issue 04; September-October 2012, ISSN: 2249 – 6556.
- [3]. V.KrishnaKumari, Y.SriChakrapani, "Designing and Characterization of koggestone, Sparse Kogge stone, Spanning tree and Brentkung Adders", International Journal of Modern Engineering Research (IJMER), Vol. 3, Issue. 4, July-august. 2013 pp-2266-2270 ISSN: 2249-6645.
- [4]. A.N. Jayanthi, Dr.C S Ravichandran, "Comparison Of Performance Of High Speed VLSI Adders", International Conference on Current Trends in Engineering and Technology, IEEE – 32107 July 3, 2013© IEEE 2013.