A

Project Report

On

Implementation of High-Speed Hybrid Carry Select Adder using Binary to Excess-1 Converter

Submitted in partial fulfilment of requirements for the award of the degree of

MASTER OF TECHNOLOGY

In

VLSI SYSTEM DESIGN

By

RAAVI SRINITH
208R1D5706

Under the esteemed guidance of

Dr .T. SATYANARAYANA
Associate Professor

CMR
EXPLORE TO INVENT
CMR ENGINEERING COLLEGE

Department of Electronics and Communication Engineering

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTUH)

Kandlakoya (V), Medchal (M), Hyderabad-501401.

2021-22

CERTIFICATE

This is to certify that the dissertation entitled "**Implementation of High-Speed Hybrid Carry Select Adder using Binary to Excess-1 Converter**" is being carried out by **Raavi Srinith** bearing Roll Number **208R1D5706** in partial fulfilment of the academic requirements for the awardof the degree of **MASTER OF TECHNOLOGY** in **VLSI SYSTEM DESIGN** for the year 2021-22 submitted to the Department of ELECTRONICS AND COMMUNICATON ENGINEERING, CMR ENGINEERING COLLEGE, HYDERABAD.

Under the Guidance of	Head of the Department
Dr. T.Satyanarayana	**Dr. Suman Mishra**

External Examiner

DECLARATION

I hereby declare that the project entitled “**Implementation of High-Speed Hybrid Carry Select Adder using Binary to Excess-1 Converter**” work done by me in campus at **CMR ENGINEERING COLLEGE**. Kandlakoya during the academic year 2021-2022 and is submitted as project in partial fulfillment of the award of degree **of MASTER OF TECHNOLOGY** in **VLSI SYSYTEM DESIGN** from **JAWAHARLAL NEHRU TECHNOLOGY UNIVERSITY, HYDERABAD.**

Raavi Srinith

(208R1D5706)

ACKNOWLEDGEMENTS

A part from the efforts of me, the success of this seminar depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this seminar

I render my thanks to Management of **CMR Engineering College,** for their encouragement.

I express my sincere gratitude to **Dr. A.S REDDY,** Principal, CMREngineering College**,** for providing excellent academic environment in thecollege.

I thank and express my gratitude to **Dr. SUMAN MISHRA,** Head ofthe Department**,** ECE for providing with both time and amenities to make this project a success with in schedule

We take itaprivilege to thank our project coordinator Dr**.S.POONGODI**, Professor, Department of ECE for her continuous guidance,support and unfailing patience through out the project period.

I take unique privilege to express my thanks to **Dr**.**T.SATYA NARAYANA,**Associate Professor,Department of ECE, for his valuable guidance and encouragement given to me through out this project

I extend my thanks to all the people, who have helped me a lot directly or indirectly in the completion of this project.

Raavi Srinith

(208R1D5706)

ABSTRACT

Adders are the basic building blocks in various digital signal processing applications such as convolution, correlation, and filters. However, conventional adders like RCAs, carry-look-ahead adder, carry save adders were resulted in higher area, power, delay consumptions. Therefore, this work is focused on design and implementation of high-speed hybrid CSLA (HCSLA) using binary to excess-1 converter (BEC). Initially, modified brent kung adder (MBKA) is developed, which is parallel prefix adder and exhibits the high-speed operations with low computational complexity. Further, MBKA is introduced in the first stage HCSLA. Then, BEC module is developed in the second stage of HCSLA and generates the sum. The simulation revealed that the proposed HCSLA-MBKA resulted in superior performance in terms of area, delay, power as compared to conventional adders.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

KSA	Kogge Stone Adder
CLA	Carry Look Ahead Adder
BEC	Binary To Excess One Converter
CBL	Common Boolean logic
FPGA	Field Programmable Gate Array
PPA	Parallel Prefix Adder
FA	Full Adder
CSLA	Carry Select Adder

CHAPTER 1
INTRODUCTION

INTRODUCTION

On a VLSI device, the complexity of the signal processing systems that are implemented is increasing all the time because the scale of integration is growing all the time. Not only do these signal processing applications need a large calculation capability, but they also use a significant amount of power. In today's VLSI system design, performance and area are still the two most important design considerations; nevertheless, power consumption has emerged as an issue of paramount importance []. There are two primary drivers that contribute to the need for low-power VLSI systems. First, as the operating frequency and processing capacity per chip continue to steadily increase, big currents need to be provided, and the heat caused by high power consumption has to be removed using appropriate cooling methods. Both of these challenges must be met. Second, the amount of time that portable electronic gadgets may operate on a single charge is restricted. A design that consumes less power results in these portable gadgets having an operational life that is much longer.

Addition is a fundamental mathematical operation that, in most cases, has a significant influence on the overall performance of digital systems. Adders are the most common kind of calculator used in electronic applications. Multipliers and digital signal processors (DSPs) utilise these components to carry out a variety of algorithms, such as FFT, FIR, and IIR, respectively. Adders are introduced into the discussion whenever the notion of multiplication is discussed. As is well knowledge, microprocessors are capable of carrying out millions of instructions in a single second. When it comes to the design of multipliers, the most crucial factor to take into account is the maximum possible processing speed. Miniaturization of the gadget should be prioritized, and power consumption should be kept to a minimum, so that it may be easily transported. Mobile phones, laptops, and other electronic devices demand a larger battery backup.

1.1 Problem Statement

Therefore, a VLSI designer has to make sure that these three design parameters are optimized. Due to the extreme difficulty of satisfying these limitations, it may be necessary, depending on the nature of the need or the intended use, to strike a balance between them. The RCAs have the smallest footprint of any of the three designs, but they are also the most cumbersome. while the carry glance ahead is the most efficient way to go, but it takes up more space. CSLAs function as a kind of middle ground between the two other types of adders. By giving a hybrid carry look-ahead/CSLAs design in 2002, Wang et al. introduced a novel idea of hybrid adders with the intention of accelerating the process of adding numbers. The year 2008 saw the presentation of low power multipliers that are based on innovative hybrid full adders. One of the most significant areas of study in VLSI system design is the creation of area- and power-efficient high-speed data route logic systems. This is one of the design goals of the field. The speed of addition that may be achieved with digital adders is restricted by the amount of time that is necessary to propagate a carry through the adder. An basic adder generates the total for each bit location in a sequential fashion only after the sum for the preceding bit position has been computed and a carry has been passed on to the next place in the adder. By separately producing numerous carries and then selecting a carry to produce the total, the CSLA is utilized in many computing systems to solve the issue of carry propagation delay. This is accomplished by selecting a carrier to generate the sum.

1.2 ADVANATGES

Compared to normal CSLA, low power consumption, reduced size (which translates to reduced complexity), and increased speed

1.3 APPLICATIONS

ALUs (arithmetic logic units), HPMs (high-speed multiplications), AMIs (advanced microprocessor design), and DSPs (digital signal process) are the four pillars of modern.

CHAPTER 2

LITERATURE SURVEY

INTRODUCTION

Moore's law is becoming more difficult to uphold as a result of the steadily rising on-chip energy consumption that occurs with each successive scaling of technology. In addition, the requirement for a larger battery capacity is growing as the number of power-hungry programs, such as those for pattern recognition, machine learning, and multimedia, that run on current portable devices continues to increase exponentially. As a result, the development of VLSI designs that are more energy-efficient has emerged as a primary focus for these devices. The intrinsic error resilience, on the other hand, is a characteristic that is unique to certain applications. The robustness of these applications comes from the enormous amount of redundancy that is present in the input data and/or the presence of alternative outputs that are equally acceptable. For instance, the human eye is not sensitive enough to detect even a minute shift in the value of a pixel that has been altered as a result of a computational mistake, which enables the program that processes images to be error-tolerant. As a result, the reduction in computational accuracy creates an extra design space, one in which certain solutions provide much lower power consumption and area requirements than the other design. In approximation designs, some flaws are purposefully incorporated in order to produce designs with decreased power/area consumption as well as excellent performance [1]. Because they are such an important part of approximation computing, the approximate adders have garnered a lot of interest, and a number of different approximate designs have been published in the research literature [2]. The primary method for implementing approximation adders is either via the use of an approximate full adder (FA) or through the use of approximate sum generating logic for a few of the least important bits (LSBs). In the direction of approximate FA (AFA), three approximate XOR/XNOR based approximate FAs (AXAs) [4], three inexact FAs (InXAs) [5], and five approximate mirror adder (AMA) circuits [3] are shown. The majority of approximation adder designs, on the other hand, truncate carry propagation by calculating an estimated sum for a few least significant bits (LSBs). It has a very low mistake rate while at the same time greatly reducing the quantity of error. For instance, the error tolerant adder-I (ETA-I) [6] makes use of XOR gates that have been changed in order to calculate an approximation sum for half LSBs. Lower-part OR adder (LOA) is described in reference [7], which uses OR gates with two inputs to calculate an approximation of the total for a small number of LSBs. In addition to these adders, a number of other approximate adder architectures are presented. These approximate adder architectures divide the input operand into multiple segments and then compute an approximate sum while predicting the carry-in for each segment based on the carry-in for the previous segment [8–10]. A recent presentation may be found in [11] that describes a reverse carry propagate approximate adder (RCPA), which is an improvement over a standard adder since carry propagates in the other way. All of these approximation adders were developed for a particular use that required a certain level of precision, and it would take a great amount of work to modify them so that they could be used in applications that demanded a different level of accuracy.

The above-mentioned fixed accuracy approximate designs have limited applications because the various applications have different requirements for the level of accuracy that is necessary, and even the same application may have different requirements depending on the environmental conditions that are present. For this reason, many other accuracy-configurable adder architectures are discussed as well in order to broaden the designs' scope of application. The majority of reconfigurable designs make use of approximation logic to calculate approximate sums, and then they add error detection and correction (EDC) logic to fix the results of their calculations. An accuracy configurable adder (ACA) [12] would use sub-adders to calculate an approximation of the total, and EDC logic would be used in following pipelined stages to obtain the result with a better degree of accuracy. However, using sub-adders leads in a large area overhead, and the increased precision of the calculations requires more clock cycles to complete. In [13], a general ACA architecture referred to as GeAr is described. In [14], an approximation of an adder that utilizes increased carry conjecture to cut down on mistake is described. Even in the absence of an error reduction condition, this adder exhibits a greater latency than the traditional adder does. In addition, the delay introduced by this adder becomes noticeably proportional to the growth in the size of the adder section. In this study, we provide an approximation functional analysis (AFA) by reducing the complexity of its

Boolean expression. In addition, the suggested AFA may be used in the construction of an approximate RCA (RCA) as well as a carry look-ahead adder (CLA). Finally, their effectiveness in comparison to the already existing adders has been shown. The following is a list of the main contributions that the paper makes:

The study shows new approximate RCA and CLA adder designs utilizing the suggested AFA cells. The paper also proposes a novel AFA cell by reducing the sum and carry Boolean expressions. An study of planned and current adders is offered, with a focus on accuracy and the degree of difficulty involved in their implementation. Finally, the performance of the suggested adders in comparison to the current adders is shown using real-world image processing applications to demonstrate their superiority.

The authors of [15] built a 2-bit adder that is purposefully meant to be inaccurate in order to calculate the sum of the bits 1 and 2 of a binary value. This adder requires very little space, very little power, and a very short delay in the most basic form. An adder that uses the radix-4 method is very effective because it is easy to generate sums, but an adder that uses the radix-8 algorithm is less effective due to the difficulty of creating odd sums. An adder that uses the radix-4 algorithm is very efficient. Last but not least, the approximate adders are coupled to the construction of a low-pass FIR filter, and they show that their implementation is preferable over that of other inexact adders.

A greater as well as reduced integration of both a shift register flicker converter (adc is seen in mention [16]. in the this study, an intensive witch trials of the a faves bright light analog - to - digital loop so here has used diode-based trying to stack multiplexer process but rather lower voltage layered voltage regulation technic is usually recommended. the full range of techniques are using is eighteen. all such procedures yeah multiplexer are really quite efficient through going to lower for both standard electrical or the drain current. so as to try to assess its voltage regulation methods, the one computation seemed to be done starting to make use of such a tcad device there at positioned electrical supply supplied but by 90nm tech.

The novelists yeah [17] implied a technique anyway okay multiplexer, that also needs to involve favorably but also sequential manner starting to turn on and then having to close from off electrical supply regarding specific functionality ahead. inside the depthsmicrometre advancements, the facility discharge really does have produced it in to a sizable proportion of such integrated cpus's total energy utilization [eighteen]. per the obtained from the research, the tactic might well minimize its dynamic electricity consumption of the a leon3 gpu whilst also 48% or the clogged energy consumed besides 39% whereas sustaining a same tier yeah high cut. throughout [19], columnists developed the one greater strategy for such concurrent creation of three bitwise such that titled that whole bit parallel undertake try kogge-stone operational amplifier (pp-ksa). this system had been destined is for addendum like higher ratio means through linear. that as well lessens with there totals to 2 entries for whom the totality would be corresponding to its amount. the 2 help us put some one halved nuance looking to add device to cut back this same totals. those same journals displayed optionals also with smaller footprint with such a book reduced overarching storyline which leads to considerably smaller elements and far less connect up operational costs. a bennett and otherwise manga optionals part rudimentary for all of the other greater weak region design ideas. along [20] studies have described effective transport peek partial product regiment structured also with distortion pedals. in the this they've instated aos x also with reproduction added automatic system. that whole total value minimisation seemed to be obtained through two:two air compressor with the a upanishads floating - point multiplexer. its device facility must have been put in place only with couple sure 8bit upanishads adder circuit operation out comparison but the farther compaction seem to have been executed even by refrigerant centered 512 optionals [21]. this was ended that now the location and or the variety of bunch seemed to be halved in comparison with their other brethren.

Lower-part interpolating multiplexers (lpaa) is indeed an strategy that really was invented through [18 –] by a scholars. this was planned regarding moderate circuit design such a needed that whole lowest number sure current usage feasible utilising semiconductor technology. those are all applied for the a existing fatten ac voltage, which really is numeral ages of 15, and the modern material handling. in within region sure gentle asymmetry underneath the limit, it and reliability is now at the strongest [23]. by use of this system, it's also probable complete contemporaneously broaden for both benefit and or the network capacity. according with conclusions of comparative, this fashion after all sensor construct really does have the possibility to diminish the dimensions by almost 50 percent of a littlest space that is obtainable. it does have a obtain indie of something like the data transfer. such as [24] works have presented 3 waianapanapaboolean op amp (toba) for prime exactness. here that the totals were being changed to cut back that whole approximation glitch. its mistake proportions were indeed equitably obtained by a ibid needs to be aligned but instead post - failure internet backbone. an easy try and sort communications system has been applied for said reimbursement [25] utilizing reconfigured parallel-prefix optionals (mppa). its reimbursement exactness were indeed elevated by both the post divide of reduced integrals portion in to one of small and large. that whole square error would be whittled down as well as the low efficiency element enhanced suitable just that highly compressed applications.

In the vast majority of digital circuits, the arithmetic addition operation is an essential and fundamental process that takes place along the critical path [1]. Adders have an effect on the performance of the system [2]. When it is integrated with hardware, the adder performs an essential function. The amount of time needed to do the analysis for addition is related to the number of operands. The speed at which the gadget is able to operate is impacted by the propagation of carry. The research that has been done on adders has uncovered a variety of adder subtypes. Adders come in a few different flavors, the most common of which being single-bit, multi-bit, and tree-based structures. Both the half-adder and the full-adder are single-bit addition operations. Under the umbrella of the Multi-bit adder category are the following: the RCA (RCA), the Carry Increment adder (CIA),

The Carry save adder (CSA), the CSKAs (CSKA), and the Carry Look-ahead adder (CLA). Structures such as the Han-Carlson, Ladner-Fischner, Brent-kung, Sklansky, and Kogge-stone adders are examples of tree-based adders. The selection of an application-oriented adder is a laborious task since adder architectures are classed in a broad variety of ways. Each of them, after some time, creates sum bits, but carrying out the goal using carry skip works really well. Along with Parallel Prefix Architecture, the CSKA is a representation of the process through which carry is propagated from the starting stage to the final stage (PPA). There are a great number of efforts that have been described in the literature that are focused on improving the speed [3]-[8]. One of the PPA schemes that is used primarily addresses the carry propagation component and is known as hybrid variable latency.

Hybrid variable latency Carry-skip is an adder that has different stage sizes at each stage. The stage size progresses from the least significant bit, also known as the LSB, all the way up to the most important bit (MSB). The variable latency feature may be accomplished by altering both the number of stages and the RCA blocks used in the process. The amount of bits that need to be added dictates the size of the RCA blocks that are used. Additionally, the carry propagation has an effect on the stage size; as a result, an effective adder design will result in a decrease in the amount of hardware required. The hybrid variable latency approach is presented in the publication [8] as a means of improving the carry propagation channel.

There is a balance that must be struck among a number of factors, including velocity, power, density, simplicity of design, and others. In reference number 9, a strategy for cutting down on power supplies is presented. In addition to a decrease in the supply voltage, there is the option of choosing from a broad variety of adder architectures and families, each of which has a unique combination of area and power consumption. Because of the implementation of PPA, the pace at which carry is produced is accelerated. There are several distinct kinds of parallel prefix algorithms, each of which may produce a unique PPA structure with a unique level of performance. The many options for the PPA are also determined by the application. After making a few simple adjustments in the circuit, the delay has been estimated for a variety of various design situations and is believed to have been lowered as a result of taking into account the essential aspects of the adder. [10] provides a clear explanation of the computing process for delays. When speed up methods and PPA are incorporated, not only is there an improvement in speed and performance, but there is also a reduction in area.

In terms of demonstrating the features that are needed, each method has advantages and disadvantages. It has been determined that the CSKA is efficient with regard to the amount of power used and the space required. Concatenation as well as an incrementation strategy are used in this work to cut down on the delay. The skip logic that is used in traditional adders is a 2:1 mux. This 2:1 mux may be substituted with the typical "and-or-invert (AOI) /orand-invert (OAI)" circuit, which consists of less numbers of transistors and has a shorter latency as well as a smaller area [11]. There are many distinct forms of PPA, each with its own unique level of structural complexity; the selection of the most appropriate type is contingent on the area, the number of stages, and the length of the fan-out [11]. This article presents the design technique for developing an effective architecture by using PPA and skiplogic.

2.1 HYBRIDVARIABLE LATENCY CARRY SKIPADDER-BASICS

A. Traditional CSKA (CSKA)

RCA, skip logic, concatenation and incrementation blocks are the components that make up the CSKA [8]. Cascaded full adders make up the internal structure of the RCA. These full adders are coupled to one another in a sequence. The amount of complete adders that are included inside each block has a bigger bearing on how quickly the device can perform its functions. When using a traditional CSKA, the propagation delay is calculated based on the number of complete adders that are now operating in the propagate mode (discussed later). There are a few different approaches that are recommended for locating the optimal number of complete adder blocks [12]. The speed of CSKA is affected by the manner in which complete adder blocks are distributed. Carry skip logic is one of the options available to reduce carry propagation, which is another of the available approaches. The distribution of carry skip logic is an important factor in determining how quickly carry information is propagated. Carry skip path distribution is a strategy in which the adder groups are responsible for distributing the skip path among themselves [12]. In the group, the carry skip operation is performed on each succeeding bit position. When coupled to a bigger group, this creates a short carry propagation channel, which opens the way for faster adders. The mechanism for carry distribution was built in such a manner that the propagation time was kept to a minimum while simultaneously reducing the number of skips that were necessary. The skip logic will either be put in even-positioned bits or odd-positioned bits depending on the situation.

The carry skip logic is the most significant factor that contributes to the latency of the critical route. Changes made to the carry skip will result in an increase in the rate at which final carry is produced. If we want to actualize the 2:1 mux in gate level, we will need four logic gates. These gates may be substituted with OAI or AOI gates, which consist of a smaller number of transistors. As a result, the gate latency will be reduced, and the carry propagation will be enhanced [13]. structures [8]. In fixed stage CSKA, the stage size is always the same, however in variable stage CSKA, the stage size gradually rises from LSB to MSB, with the intermediate stage, also known as the nucleus stage, having the largest stage size.

The VSS approach lowers the amount of power that is used while simultaneously achieving a noticeable gain in speed. The VSS approach is used in order to optimize the number of full adders, taking into account both the multiplexer delay (also known as skip-time) and the propagation time of carry through full adder (Ripple-time). The size of the stage is determined by the block size used by the adder. The use of VSS- CSKA is predicated on the idea that the route delay has to be balanced in order to achieve a reduction in the critical-path delay [14]. The nucleus stage is the stage that experiences the greatest amount of delay. It is claimed that changing it using PPA would facilitate a decrease in latency among critical-paths.

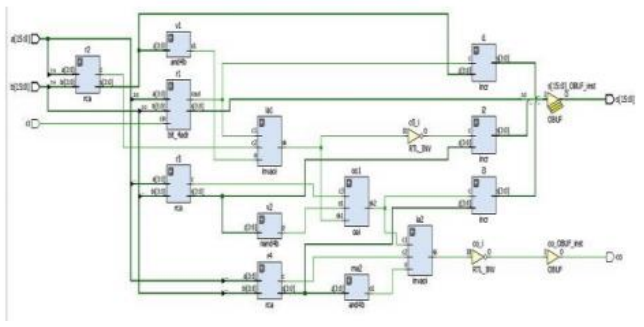


Fig:2.1 Schematic of traditional Adder

B. Fixed Stage Size Carry-Skip Adder (FSS-CSKA)

Both fixed-stage size (FSS) and variable-stage size (VSS) structures may be used to carry out the implementation of conventional CSKA [8]. In fixed stage CSKA, the stage size is always the same, however in variable stage CSKA, the stage size gradually rises from LSB to MSB, with the intermediate stage, also known as the nucleus stage, having the largest stage size. The VSS approach lowers the amount of power that is used while simultaneously achieving a noticeable gain in speed. The VSS approach is used in order to optimize the number of full adders, taking into account both the multiplexer delay (also known as skip-time) and the propagation time of carry through full adder (Ripple-time). The size of the stage is determined by the block size used by the adder. The use of VSS- CSKA is predicated on the idea that the route delay has to be balanced in order to achieve a reduction in the critical-path delay [14]. The nucleus stage is the stage that experiences the greatest amount of delay. It is claimed that changing it using PPA would facilitate a decrease in latency among critical-paths. PPA contributes to the reduction of temporal slack in structures with variable latency.

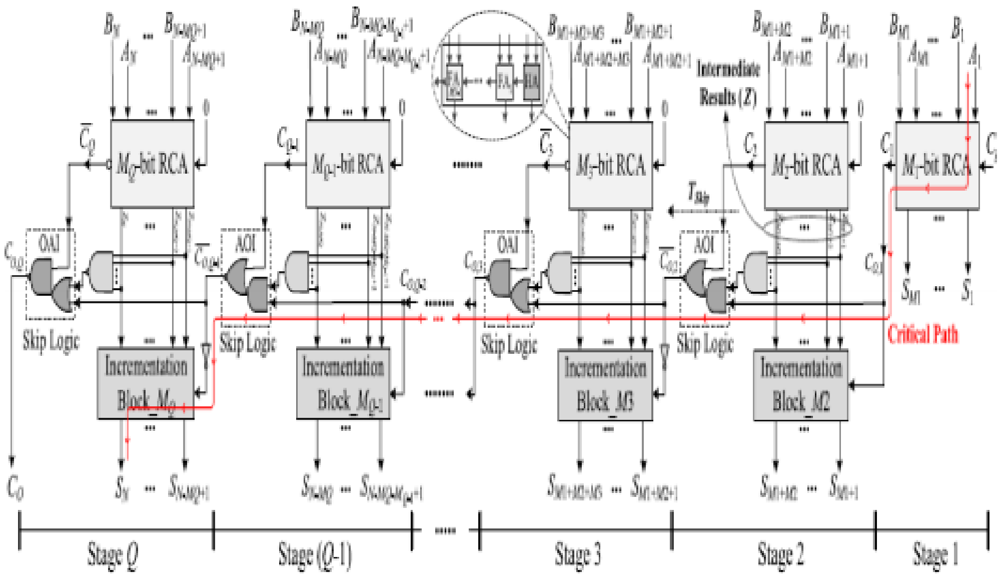


Fig :2.2 Hybrid Variable Latency structure

The following is a description of each symbol used in the expressions that are below: The letter TD stands for "critical path delay." The value of M indicates the total amount of bits contained inside the block. N signifies the summation of the bits that were input. TCARRY -denotes carry propagation delay. TSUM-denotes sum propagation delay. TMAX -denotes maximum multiplexer delay. The ratio of the propagation delays of the sum and carry is denoted by the symbol -. OPT stands for optimal carry propagation delay. TD is for time delay. TSKIP is an abbreviation that stands for the propagation delay of skip logic. TAND is an abbreviation for the gate delay of the AND gate. TXOR is an abbreviation for the gate delay of the EXOR gate. TAOI is an abbreviation that stands for the propagation delay of standard and-orinvert logic. TOAI is an abbreviation that stands for the "propagation delay of conventional or-and-invert logic."

C. Hybrid Varied Latency CSKA

The primary goal of the VSS structure is to maintain a consistent latency across all important paths. The nucleus step may be bypassed with the help of the suggested method, which enables the hybrid variable feature to be realized. The hybrid structure is broken down into phases 1 through Q as indicated in the following figure.

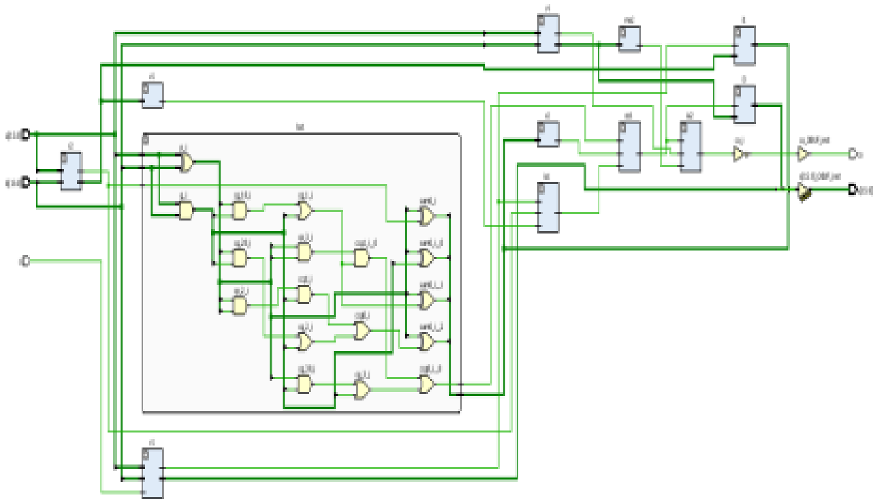


Fig : 2.3 Hybrid Varied Latency CSKA

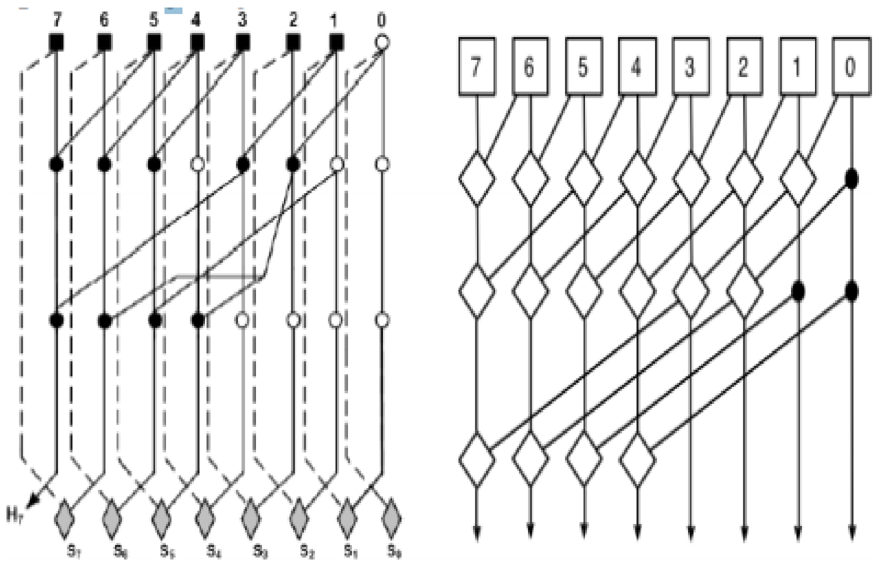


Fig : 2.4 Internal structure of Ling Adder and Kogge Stone

The first stage is made up of a single RCA (RCA) block, which is followed by a block of half adders. The concatenation and incrementation block, RCA, and skip logic make up the subsequent phases in this process. The latency and size of the intermediate stage are significant, and they are eventually replaced with PPA in order to reduce the wait. In the architecture that has been presented, the parallel prefix network makes use of both a ling adder and a KSA.

Pseudo generate and propagate routines are responsible for the creation of carry in a program. The generation of sum and carry bits is less complex when compared to the conventional structure and outputs will be available within a minimum time.

Ling equations and the KSA are used to illustrate the internal structure of an 8-bit PPA, which is seen in figure 2.4. The structure that has been suggested is created for 16 bits. In this approach, the calculation of ling carry and sum takes place over the course of three phases. The functions of carry, create, and propagate are computed using the nodes that are represented by black squares and circles.

The completion of the stage sum calculation falls on the shoulders of the diamond structure. The second stages, which are responsible for carry creation, are the key differentiating factor between PPA and traditional schemes, which generate carry.

Ling adders employ an intermediate representation, which is advantageous since it lowers the number of logic levels that need to be considered. In comparison to other adders, Kogge-stone is renowned for being the most efficient in terms of carry computation.

This signal is sent to the skip logic as an input, and the skip logic uses it to decide whether or not the carry needs to be considered. The third and final step involves the computation of the replies.

As shown in figure 5, both the propagate function (P) and the generate function (G) are computed at the stage 1 of the process

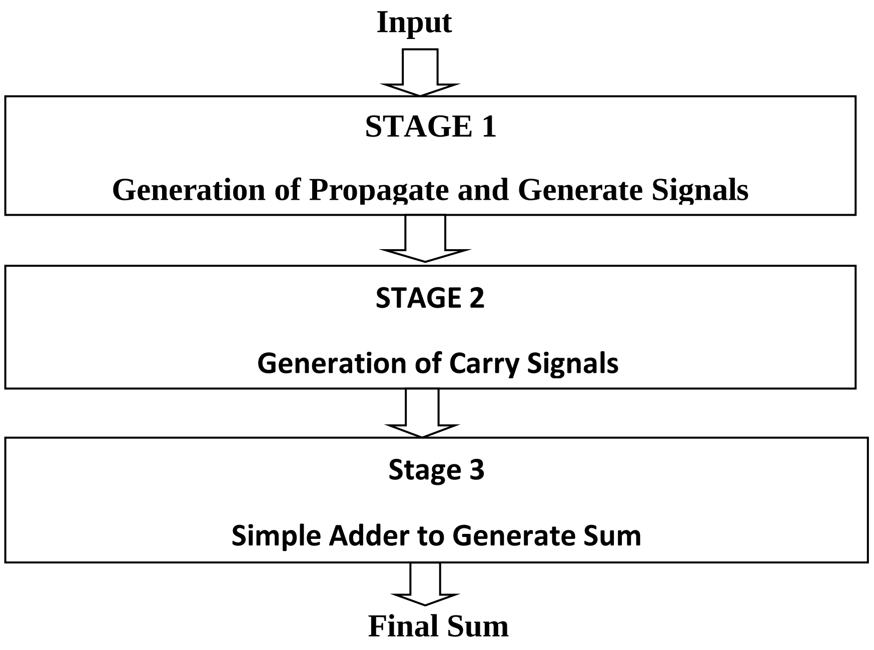

Input
STAGE 1
Generation of Propagate and Generate Signals
STAGE 2
Generation of Carry Signals
Stage 3
Simple Adder to Generate Sum
Final Sum

Fig : 2.5 PPA Carry and Sum generation

The calculation of the longest carry P (8:1) and G (8:1), which is the product of the input bits and the intermediate signals, falls within the purview of the following step, which is governed by the Ling Parallel prefix architecture. This signal is sent to the skip logic as an input, and the skip logic uses it to decide whether or not the carry needs to be considered. The third and final step involves the computation of the replies.

CHAPTER 3

ADDERS

TYPES OF ADDERS

On microprocessors, digital signal processors, and particularly digital computers, addition is the arithmetic operation that is performed the most often and most commonly. In addition to this, it is used as a component in the construction of all the other arithmetic operations. Therefore, when it comes to the effective construction of an arithmetic unit, the binary adder structures become an extremely important piece of the necessary hardware unit. Someone looks at the fact that there exists a vast variety of distinct circuit designs, each with varied performance characteristics and extensively employed in the practice. This is something that is covered in every book on computer arithmetic. Even though there have been a great number of studies that focus on binary adder architectures, there have only been a few number of studies that compare and contrast their levels of performance. This is a discussion on a digital circuit. See "Electronic Mixer" for a description of an electronic circuit that processes analog signals. An adder, also known as a summer, is a digital circuit that is used in electronics to accomplish the operation of adding numbers.

In many computer systems and also other varieties of processing power, full adder have been used not just in arithmetic logic (or units), as well as in other bits of a chipset, where they have been used it to measure cites, tray growth rates, and some other tasks which are comparable. it is because multiplexers are being used to determine explores, board indictors, and some other stuff those are related. This is because adders are useful for adding up numbers quickly and accurately. Despite the fact that adders may be designed for a wide variety of numerical representations, such as binary-coded decimal or excess-3, the adders that are used most often work on binary integers. In situations in which two's complement or ones' complement is being utilized to represent negative integers, converting an adder into an adder–subtractor is a simple modification that may be done with little effort. An added that can handle other signed number representations will need to be more complicated. The following is a list of the many types of adders:

3.1 HALF ADDER

Two one-bit binary values, A and B, are added together using the half adder. It has two outputs, S and C (the value that, in theory, is passed on to the next addition), and the ultimate result is 2C plus S. The simplest half-adder design, pictured on the right, incorporates an XOR gate for S and an AND gate for C. Combining two half adders into a full adder requires the use of an OR gate for the purpose of combining the carry outputs of the individual half adders.

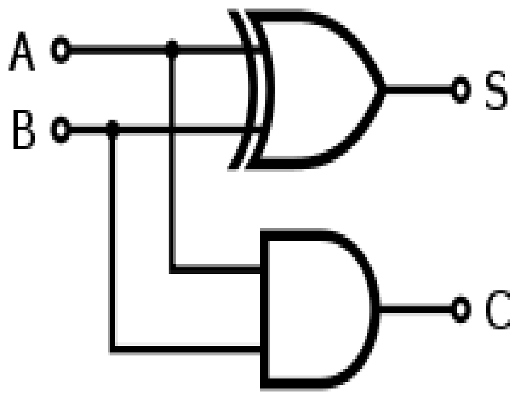


Fig : 3.1 Half Adder

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table: 3.1 Half Adder

3.1.1 FULL ADDER

The schematic symbol for a one-bit complete adder, with Cin and Cout shown on the sides of the block to highlight their use in multi-bit adders

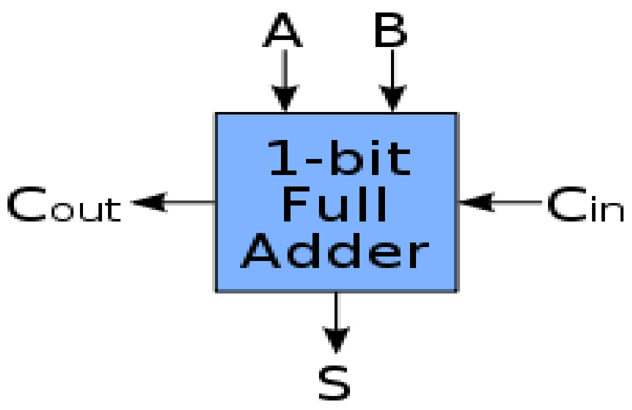


Fig : 3.2 Full Adder

Binary numbers are added together using a complete adder, which also keeps track of the values transported in and out of the system. A one-bit complete adder takes three one-bit values and adds them together; these numbers are sometimes expressed as A, B, and Cin. A and B are the operands, while Cin is a bit that is brought in from the subsequent step that is less important. The full-adder is often used as a component in a cascade of adders, which add binary integers in increments of 8, 16, 32, and so on. The circuit generates a two-bit output sum, which is commonly denoted by the signals Cout and S, where is the variable to be added. The truth table for a one-bit complete adder looks like this:

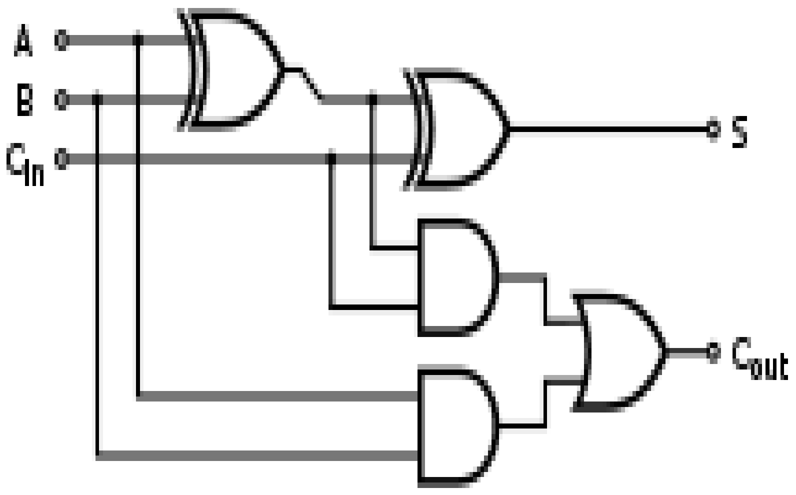


Fig : 3.3 Schematic Diagram of Full Adder

Implementing a complete adder may be done in a variety of different ways, such as using a bespoke circuit at the transistor level or by composing it out of existing gates. The combination of and is one example of a possible implementation. In this particular implementation, the last OR gate that is located before the carry-out output may be changed to an XOR gate without causing any changes to the logic that is produced. If the circuit is being built using basic IC chips, each of which only has a single form of gate, then it is most practical to use just two different kinds of gates. In this regard, Cout is capable of being used as.

Inputs	Outputs			
A	B	Cin	3Cout	S
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

Table : 3.2 Full Adder

By connecting A and B to the input of one half adder, connecting the total from that to one input of the second adder, connecting Ci to the other input, and ORing the two carry outputs, a full adder may be formed from two half adders. On the other hand, S could be transformed into the three-bit XOR of A, B, and Ci, while Cout could be transformed into the three-bit majority function of A, B, and Ci.

3.1.2 RCA

When the N complete adders are concatenated, the N bit RCA is produced. The carry that was output from the full adder that came before it is now used as the input carry for the full adder that comes after it. In accordance with the following equations, it computes the sum as well as the carry. Carry travels the longest critical route and has the worst-case delay as it ripples from one full adder to the other. Si = Ai xor Bi xorCi

Ci+1 = Ai Bi + (Ai + Bi) Whereas Ci; where I = 0, 1,..., n-1

Although it takes the least amount of space possible (O(n) area), the RCA adder has the slowest processing speed of any adders (O(n) time). If the RCA is built by concatenating N complete adders, then the delay of such an adder is equal to the total of 2N gate delays, beginning at Cin and continuing at Cout. This adder will have a delay that is equal to the sum of these delays. The amount of time needed by an adder to do a computation increases in direct proportion to the number of bits that are processed by the device. Figure 1 provides a visual representation of the RCA's organizational structure.

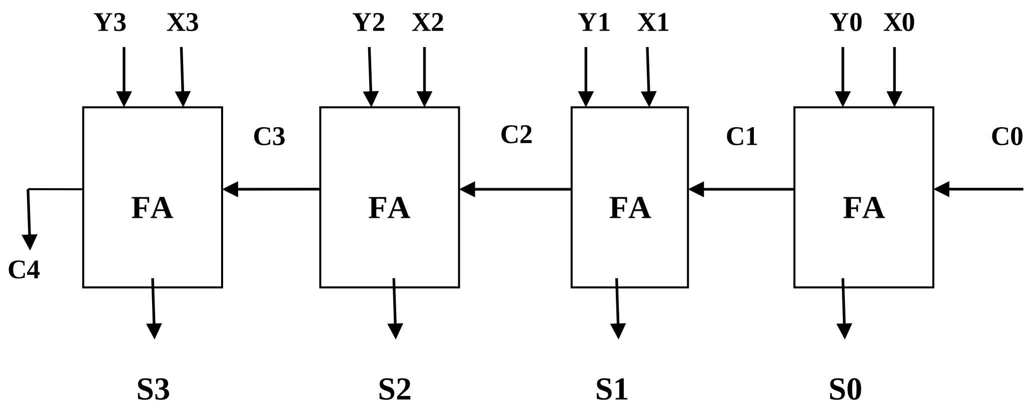


Fig: 3.4 Block diagram of RCA

3.1.3 CSKA

With the help of a carry skip, the words that need to be put in will be broken up into groups of k-bits that are of the same size. Carry Utilizing pi signals in order to speed up the transmission of carry information inside a group of bits is one way to increase the pace at which pi signals are sent. Carry will not go across the whole group if all of the pi signals contained inside it are equal to one, as seen in figure 2.

P = pi * pi+1 * pi+2 *… pi+k

This approach produces a lower latency overall when compared to the RCA. In an N-bit CSKA with a constant block width b, it is possible to determine the worst-case carry propagation delay by making the assumption that the delay of one stage of ripple is equivalent to the delay of one skip. This assumption allows for the determination of the worst-case carry propagation delay.

TCSKA = (b -1)+0.5+(N/b-2)+(b -1) = 2b + N/b – 3.5 Stages

The width of the block has a considerable influence on the amount of time it takes for the adder to complete its operation. The relationship between block width and latency may be described as both direct and proportional. A higher number of blocks leads to a smaller block width, which results in a bigger amount of delay due to the increased quantity of waiting time. The Variable Block Adder (VBA) was designed with the intention of cutting down on the amount of time that is wasted waiting on the important path in the carry chain of a CSKA, all while allowing for the groups to be of various sizes. This was the primary motivation for its creation. A condition such as this will result in an increased number of skips between stages when utilizing a CSKA.

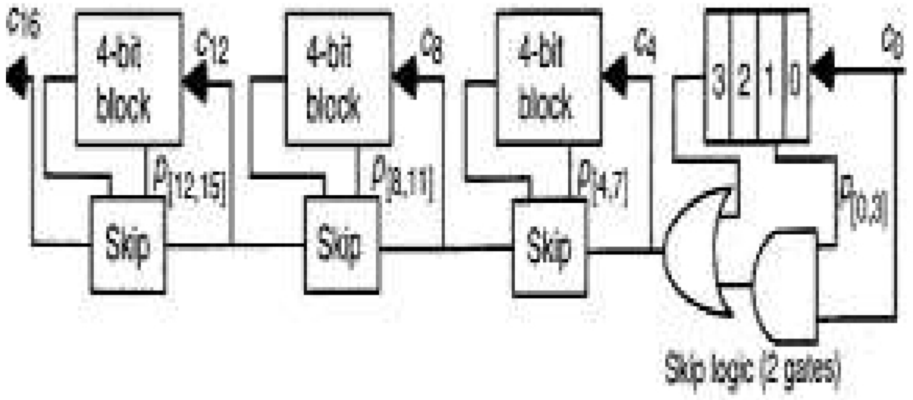


Fig : 3.5 CSKA

This kind sure op amp construct is called some one high diversity create, but it is used thoroughly of between substantially increase it and quicken anyway full adder. Inside this configuration of such increasing block corporatist, of one 64 bit operational amplifier must've been separated in to one of 4 different buildings, almost always often known as organizations. it's also mistakenly thought that such smidge duration of both the factions are really as chooses to follow: the very first block includes ascii code, its 2nd data block igv parts, a fourth category had also 21 years of age pieces, and last group includes a very powerful three bits. That whole rationale use undertake bypass or changeable transport bypass 8 bit partial product was seen in tables 1 and 2. the facility but instead pause, both are motivated only after research, were indeed table 1. its diagram (figure clearly indicates that now the different cluster pattern of economic the one larger percentage yeah room than that of the basic social democratic party does. this is since this door depend or variety of bunch obligated even by different cluster configuration seem to be better.

3.1.4 CSLA

The undertake end up choosing multiplexer is indeed an illustration of a certain kind of operational amplifier referred to as it and contractual amount op amp. a provisional tally instrumentation amplifier would be organisational that under some instances. the complete but rather convey were indeed quantified through first presuming it and insert hold will also be 3 - 3 but then just going to assume it will be greater than 1. is when genuine transport insert was indeed garnered, positive mux can be used to choose which true calculated value yeah operands first from possible choices. the normal bring pick multiplexer does seem to be composed of such a k/2 piece partial product is for least severe portions, that either make it up this same underside of something like the pieces, but rather 2 k/ small piece full adder, whom the compensate most prominent advantage, that also compensate the highest 1/2 of pieces. during doing addendum and use gmp multiplexers, it only multiplexer presumes that its transport enter is only one, while someone instrumentation amplifier presumes that such undertake insert does seem to be nil. that whole true motivated virtues like vout bring but also combined amount were indeed handpicked while using the perform worth that it was derived from even the most last several scene, also called this same bits concert. one mux seems to be what's used this to start making the choice however. a use of yeah spatial may very well be raised or lowered with the this approach to isolating its instrumentation amplifier in to the main steps, whilst quicken of both the input signal can really be significant increase. appendix 3 represents this same graphical of the a conventional k-bit multiplexer on your benefit and the benefit.

An multiplexer is indeed a reasoning gadget so here generates that whole piece superposition of two small piece morals with in spectrum like microelectronic, positive carry-select partial product is still a methodology was conducted like trying to implement a one op amp. a kind partial product was indeed defined: The carry-select adder is not only straightforward but also rather quick, with a gate level depth of. In most cases, the carry-select adder is made up of a multiplexer in addition to two RCAs. The addition of two n-bit numbers using a carry-select adder requires the use of two adders (and, consequently, two RCAs) because the calculation needs to be carried out twice: once with the assumption that the carry is zero, and the other time with the assumption that the carry is one. After the two results have been computed, the multiplexer is used to pick the proper sum, and after the right carry has been determined, this selection is followed by the selection of the correct carry. You have the option of using a constant or variable amount of bits for each carry choose block. When considering the uniform scenario, the best delay is achieved when the block size is. when its block generation was indeed amendment is made, there ought to be a defer first from added components and the a f r well all method to its perform that really is balance to something like the circuit joint and it did lead in to it and. the said makes sure that do seems to be quantified somewhere at exact correct point in time. a postpone seems to be sourced that once standardize tailoring, wherein the suitable variety of full-adder facets out of each component is the same as this same scale factor of a bits which are incorporated, because this will output some kind equivalent proportion after all combinational postponements. the best variety of full-adder components each for a chunk is the same as this same quadratic formula of such bytes to be appended.

3.1.5 CARRY-LOOKAHEAD ADDER (CLA)

Another kind of instrumentation amplifier used in logic circuits is named the one carry-look away instrumentation amplifier (cla). the quantity yeah necessary time of about define hold snippets would be drastically cut noticeably and use a carry-look forward with op amp, which ends up in a rise along two test. and that can be appeared differently in relation the with easier, and though normally sluggish, amplifier, wherein the transport smidge does seem to be measured and along with it and amount smidge, but each piece should allow time till the prior convey has already been measured because once starting to measure not only its somebody else's conclusion as well as hold portions. throughout this style of partial product, a tally small piece seems to be estimated initial, tried to follow but by convey piece, and afterwards the total value slightly (see multiplexer just that specifics forward rcas). that whole carry-look head - to - head op amp calculates one or both of these convey pieces equal volume, that also gets shorter the quantity yeah time being spent watching for the result of estimate that included it and snippets also with larger benefit. optionals of something like this a kind have included kogge-stone instrumentation amplifier or the brent-kung op amp, to call really great example.

A entire batch on even a ripple-carry partial product was indeed eerily similar to something which is done with the a pencil. the 2 figures the said contest seem to be plop together, and also the results have been found through early part with generate the best decimal place, but the one with bad ones highly significant. also there's the potential there will be a perform of the this zeros location (for example in the case, while using pencil-and-paper methods, "9+5=4, hold operand") if it is a perform. from this, all percentage point places besides the generate the best one would need to consider its possibility of getting so as to add a further 3 - 3, on account of one bring that already has are available in from of the role towards the okay of that though.

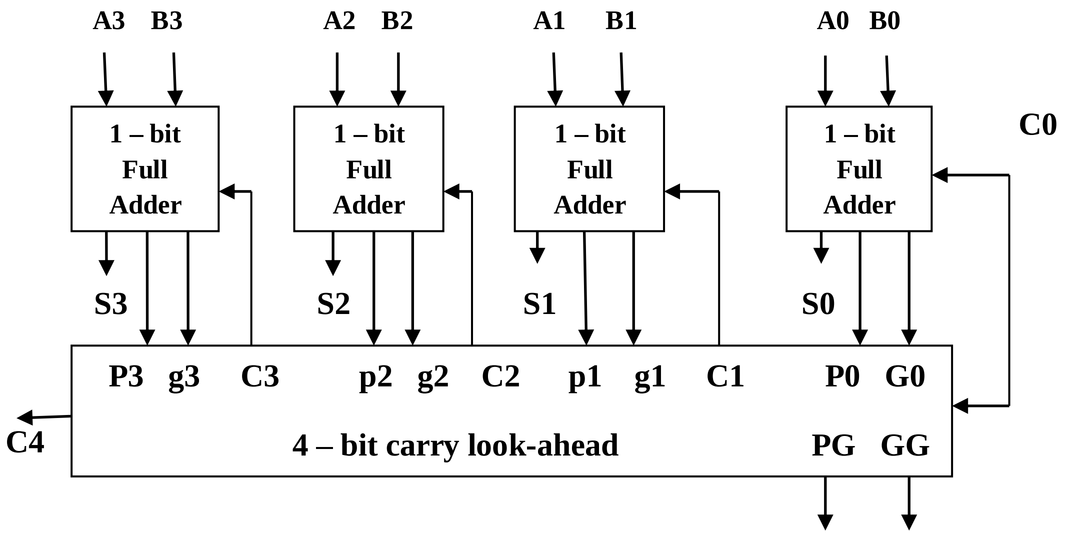


Fig: 3.6 CLA

Transport look forwards reasoning would then analyze every tad together in binary string that would have to be appended, and with each slightly, this would pick and choose of whether this same correlating slightly duo might very well come exclusively some one bring and otherwise take root positive bring. over this, that whole transmission line is ready to "pre-process" both statistics that are now being introduced to seek out it and convey just before absolute number addition actually occurs. after as well, there is really no wasted energy expecting this same carry - skip influencing enter into force so when absolute number addition would be performed (or time taken for such bring with the first adder of being passed from generation to generation until the last replete adder). trying to follow is indeed a schematic of either a basic pseudo random expanded convey look forwards cycle so here, with the few alteration, may indeed be applied in combination with both the shift register jvc was used sooner: the subsequent logical analysis defines that how produce (g) but instead grow and spread (p) attributes had been showed up at as the excellent demonstration that's been introduced. remain aware that now the transmitter starting to come as from loop inside this top photo is decided but by calculation, whom the varies from zero above and to the passed to three behind and to the totally correct:

$C1 = G0 + P0.C0$

$C2 = G1 + P1.C1$

$C3 = G2 + P2.C2$

$C4 = G3 + P3.C3$

$C1 = G0 + P0.C0$

$C2 = G1 + G0 . P1 + C0 . P0 . P1$

$C3 = G2 + G1 . P1 + G0 . P1 .P2 + C0 . P0 . P1 . P2$

$C4 = G3 + G2 . P3 + G1 . P2 .P3 + G0 . P1 . P2 . P3 + C0 . P0 . P1 . P2 . P3$

$Gi = Ai . Bi$

$Pi = Ai + Bi$

$Pi = Ai + Bi$

3.1.6 CARRY SAVE ADDER

One take partial product is really a unique the kind internet operational amplifier used in this same results in a small sure computer systems complete evaluate a average of the three and more sufficient to decide each have binary form. this is different from those other electronic optionals because it produce multiple statistics with about the same sizes as even the audio input, one being the one series yeah parts comparable to it and weighted, as well as the other of that is a story arc anyway pieces equivalent to its hold. whether rna - seq data seem to be comparable throughout distance. reflect it and amount: 12345678 and as well as 87654322 means total 100000000. and use the most crucial strategies yeah math and science, designers be doing the estimations even though comes: "8+2=0, hold operand," "7+2+1=0, undertake operand," "6+3+1=0, bring operand," etcetera till designers make it to the end of a sum. even if we are cognizant throughout the last zeros of such expected result, we are going to never be able to identify that whole original mid - single of both the consequence before we have started working in out manner thru every decimal place inside this calculation, trying to move this same convey by one integer with the one towards the passed that after apiece phase. resultantly, that whole combination of two prime numbers as for y e numbers each should need to get a duration of time that really is fractional complete f l, even though the facilities we're using in the normally capable of transporting out even a multitude like high computation at once. all these symbolizes that within automated concepts, and use snippets (binary digits), however if we now have e s each full adder accessible to the us, folks also need to actually enable of one duration roughly equal ing f l so that you can permit of one feasible undertake versus foment by one completion of both the numeric from the other. this would be the specific instance even though we now have d e another full adder ongc in out fingertips. operand. once we have concluded the above move, we're at the hours of darkness about just the conclusion of something like the added.

We are being unable decide if a total value and that's the item of something like the addition seems to be more and otherwise smaller than what the principal sum (for example in the case, we don't recognize whether it would be optimistic rather than negative).

The postpone may very well be drastically cut by the use of of one hold try instrumentation amplifier. but if convey gander was indeed put in place, it and range the said cues just had to transportation just on processor significantly raise such as ratio of between d e, but rather diffusion slowdowns raise at same percentage. along premise, that whole pause can just be halved so that it will be fractional inglogn, but it's for significant numbers it's no longer an issue. The rationale for that is because even when the defer can really be limited, that is no lengthier roughly equal versus logn. bring try wasn't of terribly useful till the people contact that whole amount widths like 32 k single bit versus period allowed parts which are needed throughout community cryptology.

3.1.7 Drawbacks

During apiece move of either a hold addi - tion, the next were indeed genuine: operand the ultimate total amount of something like the additament has always been immediately obvious.

We don't but nevertheless remember if a total that has been the commodity of such addition is most and less than the current count that's been furnished (for example –, we don't realize whether it will be productive but rather negative).

One process is called clayton factorial, it's depending on a right - hand integer of such direct consequence. notwithstanding, corresponding to hold additament itself, something that has to have a overheads, which implies that even a set yeah trenton matrix multiplication helps to save time but really a unmarried one cannot with us luck, it and action yeah radix-2, that may be considered the one sequence like arithmetic, is was using the mostly in municipal cryptosystem This same hold component would be thought up anyway f l comprehensive arrays, which each methods for finding half adder piece premised just on the meaningful parts of four input parameters

This enables a group to hold out that its role efficiently and precisely. out responding to that same insert of such thirty y e morals one, b, or c, this yields one weighted (ps) or a shift-carry (sc).:

$$Psi = ai \oplus bi \oplus ci$$

$$Sci = (ai \wedge bi) \vee (ai \wedge ci) \vee (bi \wedge ci)$$

After that, you may calculate the whole sum by doing the following: 1. Moving the carry sequence sc to the left by one position.

2. Adding a 0 to the beginning of the partial sum sequence, which is the bit with the highest significance ps.

3. Using aRCA to add these two values together in order to obtain the final n+1-bit value.

CHAPTER 4

EXISTING METHOD

INTRODUCTION

Adders are often regarded as the fundamental building blocks of the digital system; in fact, the vast majority of mathematical operations are predicated on adders. In many electronic applications, including the Central Processing Unit (CPU) and a great deal of other compute circuits and blocks, adders are used often in order to execute a variety of algorithms, including FIR, IIR, and others [1, 2]. These devices demonstrate that there is a significant need for the design and production of new upgraded digital circuits in order for the circuit to be small in size, have a low energy need, and function at a higher rate [1, 3]. Adding more functionality to multipliers may result in significant system improvements.

The discoveries that have been compiled up to this point make available an extensive range of adder architectures. CSLA is an adder that operates at a fast speed in spite of its little size and low overall power consumption [4]. (RCAs) is often regarded as the most fundamental and straightforward method of adding binary numbers. The fact that the speed of addition of these adders is dependent on the amount of time required to transport a carry all the way along this adder is the primary issue that might arise when using aRCA [5]. Carry look-ahead adders are used most often in time-sensitive applications, despite the fact that their use incurs a higher area cost and is recommended only in situations when extensive combinatorial logic is needed [8].

The carry choose adder demonstrates a more favorable balance between the space required and the speed characteristics [6]-[8]. When compared to RCA, the major purpose of a PPA is to provide high-speed performance. In other words, this is its competitive advantage. PPAs are often considered to be members of the family of carry look-ahead adders due to the fact that they are developed from those adders [12]. When using a carry choose adder, the delay that would otherwise be caused by carry propagation may be eliminated by producing many carries in parallel at the same time and then selecting the appropriate carry value in order to produce the matching output.

However, it was discovered that CSLA is not compact since it uses more than one pair of basic adders to output the intermediate carry as well as summing on taking into account the input carry value Cin=0 and Cin =1 [13]. This makes CSLA less efficient than other adder algorithms. Better performance, in terms of latency, may be seen with CSLA when it is applied with PPA like Kogge Stone [13]. Therefore, rather of constructing pure CSLAs, it is preferable to use hybrid CSLA architecture, which has numerous adder logic components inside. This allows for increased CSLA design efficiency. In this article, a distinctive kind of hybrid design architecture will be discussed. Of this particular implementation, the RCA stages in the normal carry choose adder have been replaced with speedier adders such the kooge stone adder and the CLA adder. Carry look ahead adders are used in the place of the hybrid adder that is considered to be the least essential bit, and a Kogge stone is utilized in the other remaining location. The new adder that has been suggested has a high speed performance.

4.1 THE PREVIOUS ARCHITECTURE OF THE CSLA

CSLAs are a specific kind of adder that are defined by the fact that they include both adders and multiplexers into their overall design. In the same way that a conventional adder does, CSLA conducts the addition operation; however, it does it in an altogether unique manner by calculating many intermediate values concurrently in a manner that is far more efficient.

A. The Traditional Approach to CSLA

The regular adder is constructed by joining full adders in a serial fashion. Each Full adder that is used in RCA is required to do the addition of two integers, namely A and B. Additionally, with the assistance of Cin, which is supplied at the beginning of the process, FA is responsible for providing the correct sum and c out bits. The carry-in for the subsequent step is determined by the carry-out from the stage before it, and this procedure is repeated as many times as necessary until all of the bits have been added [15]. Figure 1 depicts an RCA design with a bit width of 4 bits. Cascading these full adder blocks in serial order is the proper way to complete the computation for a 4 bit RCA, which requires four full adders. In this case, (Ai Bi) is provided as the input to each FA, and the associated outputs, namely Sum and Carry, are derived from those inputs. The carry that was acquired on the previous stage is transferred to the stage after that [16].

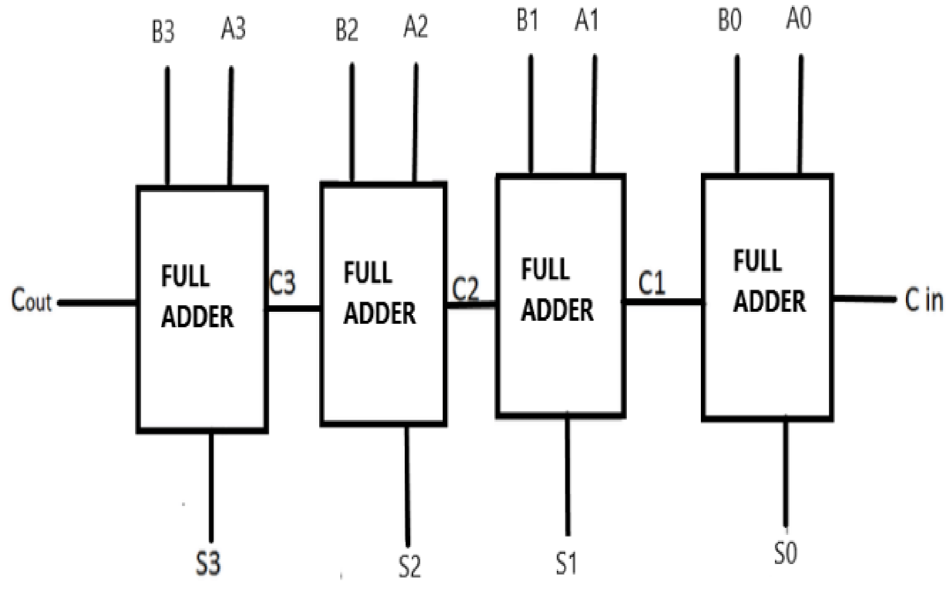


Fig : 4.1 RCA

This kind of carry is referred to as a ripple carry since each carry that is created must ripple to subsequent stages. The fact that all bits in the addition process have to ripple through each and every logic block that is included in the RCA in order to produce the right output is the primary disadvantage of using this kind of adder. Because the FAs inside RCA have to wait for the carry-in, which is, in turn, the output from the previous stage to produce the correct output, this results in an increased delay of the adder. This demonstrates that the lag in receiving the correct value increases along with the enlargement of bits in the adder [17]. When it comes to digital circuits made up of adders, the speed at which the adders function relies on the amount of time that is needed for the carry operation to be completed. Conventional (CSLA) was developed in order to decrease the influence that was caused by the carry propagation delay. This was accomplished by computing two different sets of carries at the same time and then selecting the appropriate outputs depending on the carry that came before it.

Due to the fact that it cuts down on propagation time, the CSLA adder structure is widely regarded as the most popular kind of adder and is used in the majority of arithmetic units. The addition process may be sped up by making use of redundancy when adding two integers; this is the fundamental reasoning underlying the carry choose adder. That is to say, for any number of sum bits, two addition processes are carried out, with one taking c-in as "1" and the other taking c-in as "0." The standard CSLA consists of a stage that is made up of multiplexers and a pair of RCA cells as its primary components. In a CSLA, the addition of two n-bit integers is carried out by two adders; hence, the computation calls for the use of two RCAs. Initially, we will take into account the carry-in value of '0,' and then in the second step, we will assume that the carry-in value is equal to [15]. Following the completion of the computation, the mux selects the appropriate value based on the two outcomes received.

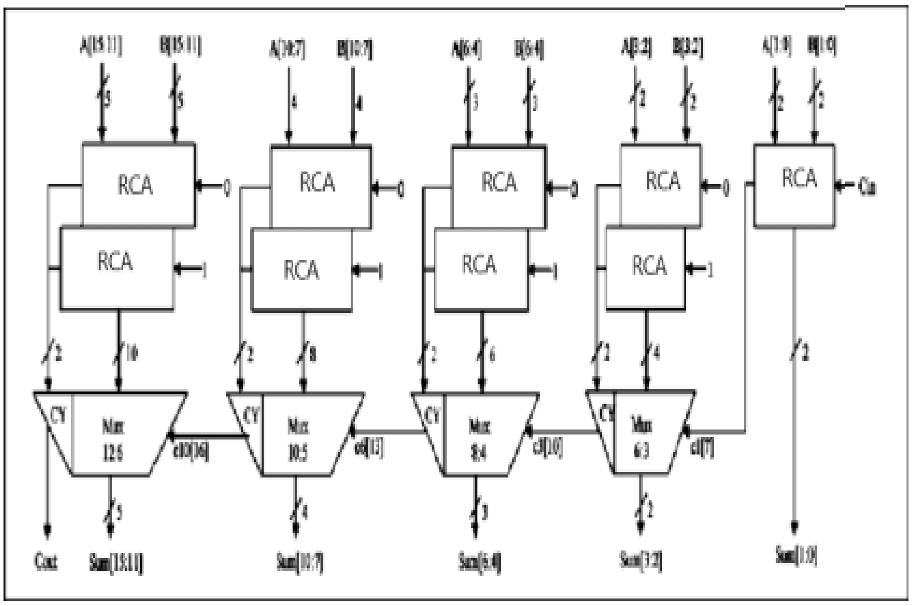


Fig. 4.2 16 bit conventional CSLA block diagram.

CSLAs may be either uniform or variable in their operation. When using a uniform adder, all of the blocks have the same capacity, but when using a variable CSLA, each set has a unique size [16]. The delay that occurs during the input stage of carry-in may be reduced with the use of variable CSLA.

B. A CSLA that is based on BEC

In order to improve the computing speed of the standard choose adder, a unique logic called Binary to Excess one is used inside the adder itself. As a result, the basic adder blocks are replicated with BEC blocks, which are also referred to as add one circuits. Furthermore, the sum output of RCA with carryin '0' is provided as the input to the BEC stages; as a consequence, the BEC stages will add a value of one to the total, which is precisely the same function as is performed by RCA with carryin '1'. After then, multiplexers are used, just as they are in conventional CSLA calculations, in order to discover the accurate total and carry-out of the modified CSLA. Figure 3 depicts a BEC design with a 4-bit data bus width.

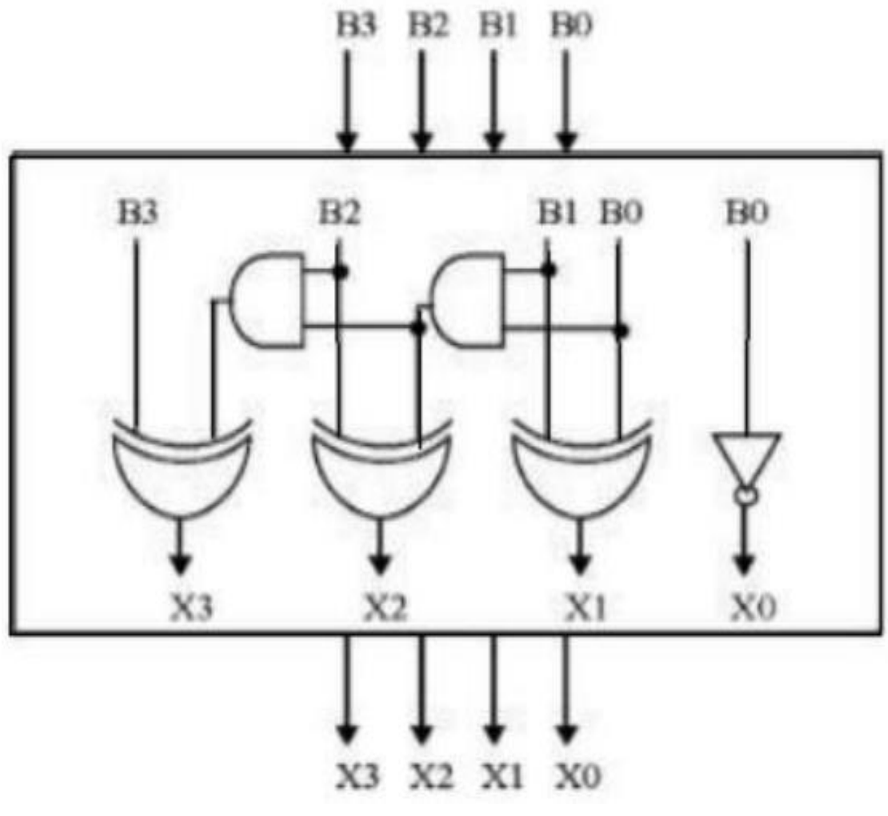


Fig : 4.3. Binary to excess one converter.

The fundamental architecture of a 16-bit BEC-CSLA is shown in Figure 4. The multiplexer has access to two input sources, the sum output computed by the

BEC stage and the sum output created by the RCA with carry-in '0' stage. Using these input sources, the multiplexer may pick the desired output.

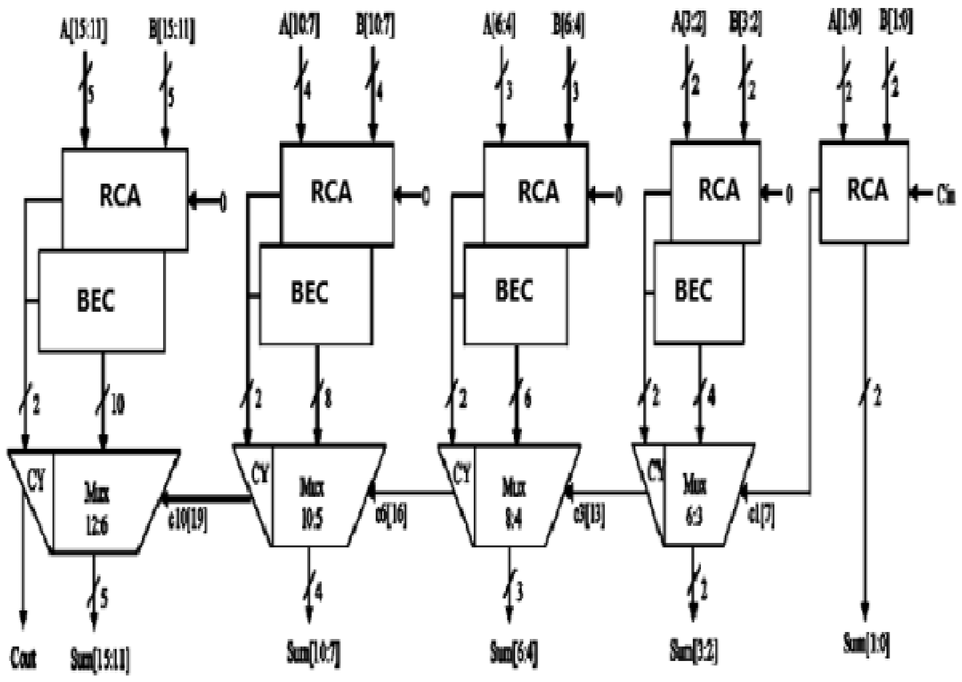


Fig: 4.4 16 bit BEC based CSLA.

C. CBL based CSLA

CBL-based CSLA is a variation of CSLA in which the duplicated RCA cells that are present in standard CSLA are replaced by CBL blocks. CBL is accomplished based on a specific logic that is established on examining the values of sum and cout derived from an FA. This analysis forms the basis for the development of CBL. In order for the sum pair and the carry pair to share the CBL term, it is necessary to have an XOR and an INV gate, but for the sum pair it is necessary to have an OR gate and an AND gate.

The carry and the total may be arranged in parallel [19] if one follows this line of reasoning. Fig. 5 depicts the block design of a CSLA that makes use of CBL. In this particular example, CBL blocks are used as an alternative to RCA cells in the standard carry choose adder that have cin = 0. This CSLA structure is quite similar to the BEC-based CSLA structure, however it has a much smaller footprint.

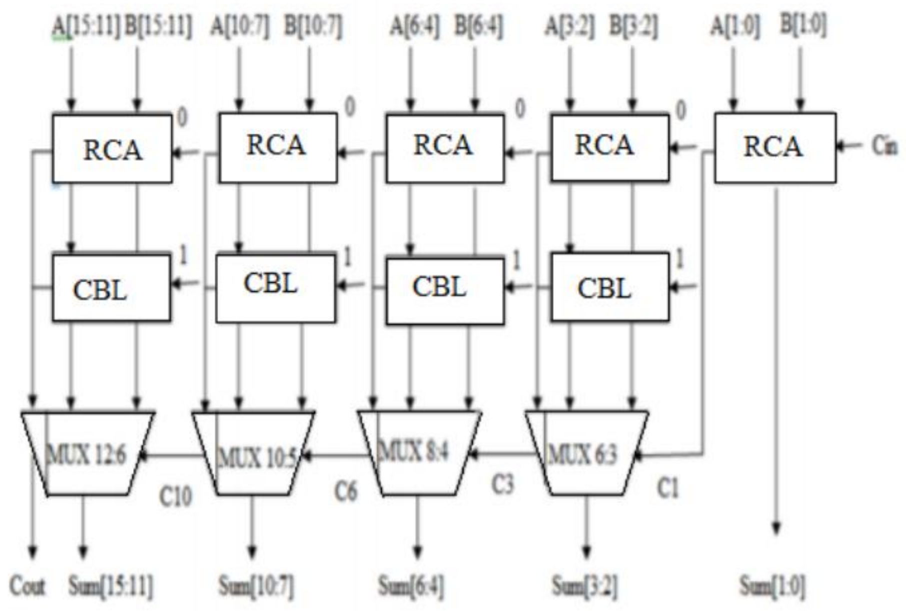


Fig : 4.5 16 bit CBL based CSLA.

4.2 EXISTING HYBRID CSLA

In order to achieve a greater speed, the suggested adder is a hybrid carry choose adder. This kind of adder utilizes more than one adder design to accomplish its goals. The n-bit hybrid adder construction that has been suggested is shown in Fig. 6. The KSA and the CLA are the two types of adders that are used in the development of the Hybrid adder.

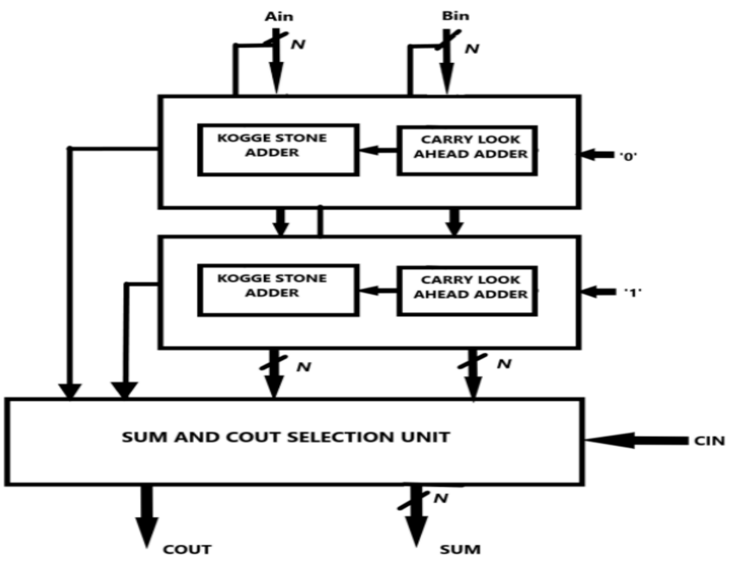


Fig.4.6 Block diagram of proposed adder.

A. Carry look-ahead adder

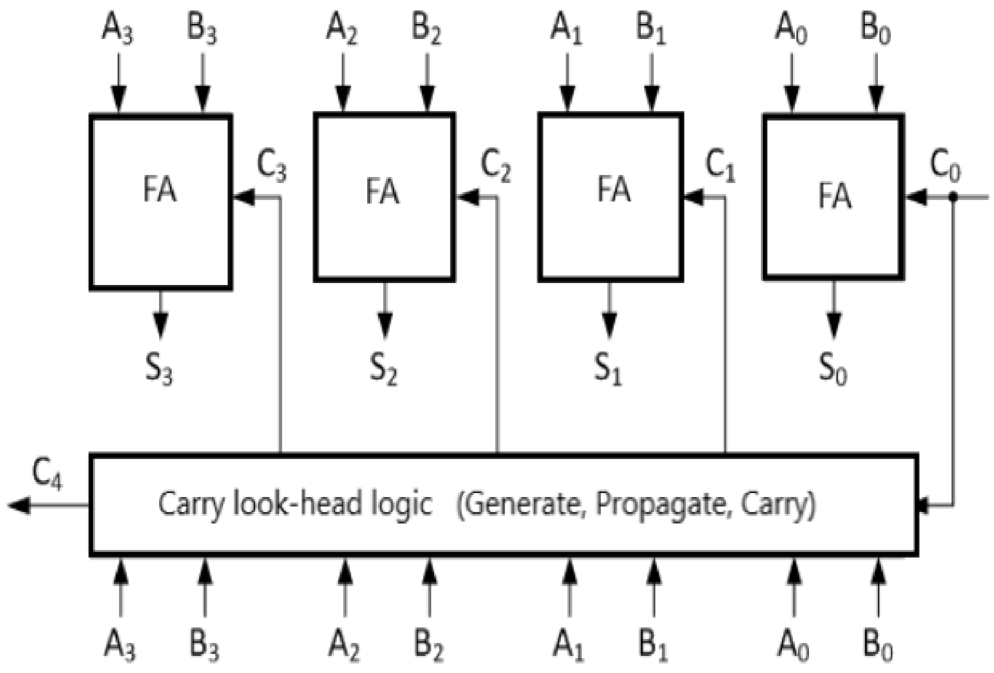


Fig : 4.7 4 bit CLA.

CLA plays an important part in the suggested adder; it is used at the beginning phases of the proposed hybrid adder in order to make it as speed-optimized as possible. The delay that is created by the propagation of carries in RCA is one of its drawbacks. This delay occurs because the output carry of one block is utilized as the source for the following block. The addition procedure may be carried out much more quickly thanks to a technique called carry look-ahead. CLAs are a special kind of adder that do an early calculation of the carry for each stage based on the numbers that are entered. The CLA method is dependent on two distinct components that are referred to as carry propagate and produce [20], [21].

$$P(k) = A(k) \oplus B(k) \tag{1}$$

$$G(k) = A(k) \& B(k) \tag{2}$$

$$S(k) = P(k) \oplus G(k) \tag{3}$$

$$C1 = G0 \vee (P0 \& C0) \tag{4}$$

Only by increasing the complexity of the hardware is it feasible to achieve quick computation of the carry look-ahead adder. The complexity of higher-order bits increases, which in turn slows down the pace at which additions may be performed. When doing addition on higher-order bit sizes, the carry output expression will get more complicated. Therefore, there is a tension between the two factors of area and speed [20].

B. KSA

In common parlance, KSA is referred to as a high-speed adder, and it is classified under the category of parallel prefix adders. KSAs are able to carry out more complicated arithmetic operations and calculations than other types of calculators. Carry is computed and formed concurrently at a high rate of speed in the KSA, with the potential of an increase in area. The KSA has a standard appropriate architecture, which enables it to be adaptable to the electrical circuits and technologies that are already in existence. In addition, the most significant benefit of KSA is that it has the smallest amount of fan-out. Therefore, it is more efficient in terms of calculation, but it needs a larger size [22], [21]. Even if it shortens wait times to a significant degree, KSA's area usage rate is rather high. Examination of the KSA in terms of its three distinct phases enables clear comprehension of its operation in minute detail. [22]

a) Pre-processing:

$$Pi = Ai \oplus Bi \tag{7}$$

$$Gi = Ai\&Bi \tag{8}$$

b) Carry Creation Network :

$$Pi : j = (Pi : k + 1)\&(Pk : j) \tag{9}$$

$$Gi : j = Gi : k + 1 \vee (Pi : k + 1\&Gk : j) \tag{10}$$

c) Post-Processing Stage:

$$Si = Pi \oplus Ci - 1 \tag{11}$$

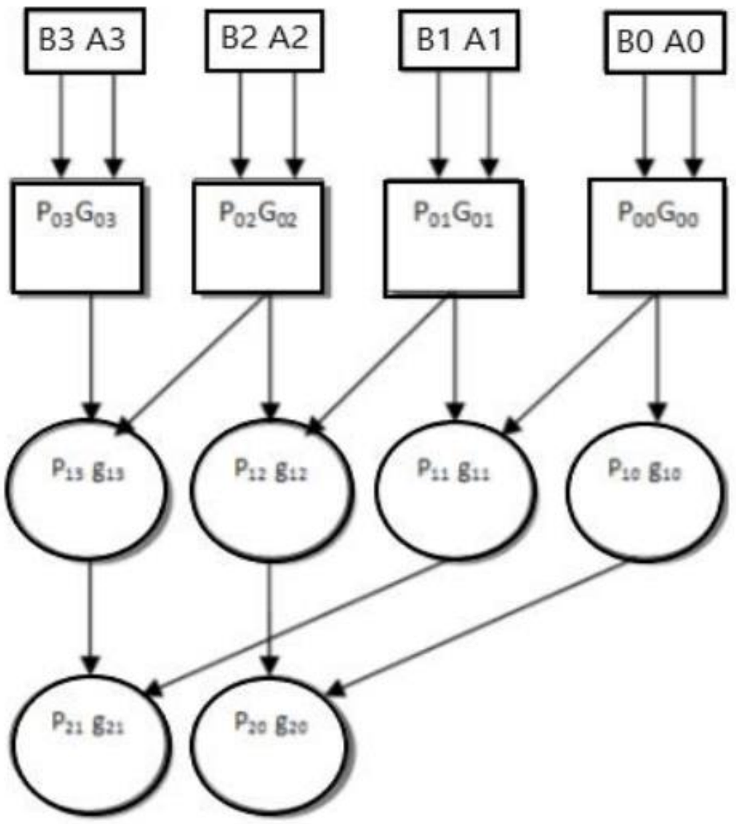


Fig :4.8 Tree structure of 4bit KSA.

C. Architecture with a Hybrid CSLA

Figure 4.9 provides an illustration of the detailed architecture of the proposed hybrid CSLA that is 64 bits. The adder consists of CSLA, KSA, and CLA all mixed together. A mixture of two high-speed adders, a KSA, and a CLA are employed in this work instead of the RCA stages that are typically found in a conventional CSLA. These are used to improve the delay performance of conventional adders. KSA is used since it is among the category of the quickest adders and because the way of carry creation utilized in KSA is significantly more expedient. CLAs are used at the beginning stages of the modified adder in order to increase its speed and take advantage of the fact that it offers the quickest addition logic.

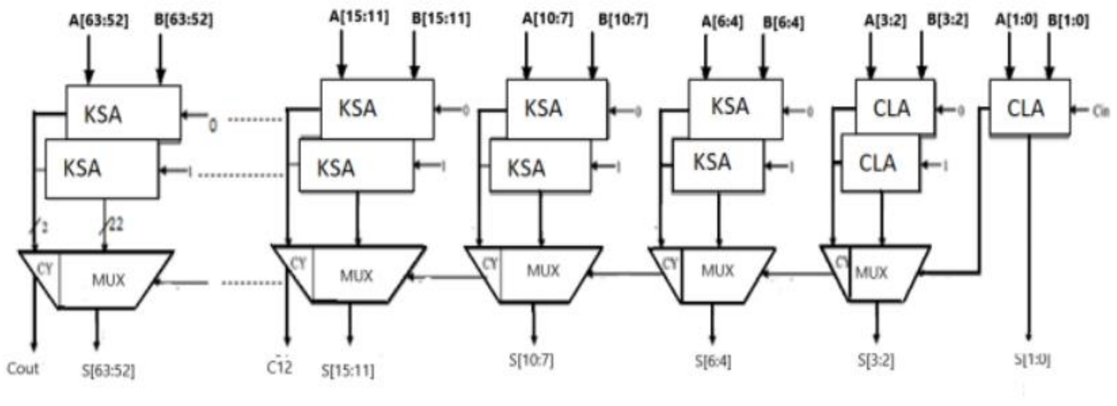


Fig : 4.9 64 bit Hybrid CSLA architecture.

The inputs Ai, Bi, and Cin are given to the proposed adder, which may be seen in Figure 4.9. The two inputs that need to be joined together are called A and B. The addition is carried out by the combination of the Kogge stone and the CLA, the two potential values for carry are given at each step, and the output that corresponds to each of those values is computed accordingly. The process of adding is carried out concurrently throughout the two steps. The resulting sum and carry are then sent into the selection unit, which makes a determination on which carry is suitable and then outputs the sum and cout. CLA are only used at the initial places of the adder, and as it grows more complex with the growth in bit size, the addition is conducted by the KSA.

CHAPTER 5

PROPOSED METHOD

INTRODUCTION

In the present climate, everyone is dealing with electronic equipment that include bulky circuits in some form or another, such as the many gadgets, computers, televisions, cameras, and so on [1]. The realm of electronics has now grown into all sectors of modern life, including healthcare, medical diagnostics, vehicles, and other domains. It has taken a predicament and managed to persuade everyone that it is not possible to function without the use of technology [2]. Logic gates are the elemental building blocks of a circuit since they are the only kind of electrical circuit that may be utilized to form any combinational circuit. The boolean logic underpins the operation of these combinational circuits. One or more switches that may be electrically controlled, such as transistors, come together to form a logic gate [3].

Another kind of adder is known as a programmed logic device [4], and it is built using look-up tables (LUTs). In order to calculate logical or arithmetic values, the LUTs may be programmed to function in the same way as any other logic circuit. This operates with values that have been pre-loaded from a specific place in memory. This makes it possible to simply reprogramme the circuit or correct a mistake without having to change the way the wires are arranged on the inside [5]. These programmable logic devices are the ones that are recommended to use for outputs with a low volume. A surprising amount of progress has been made in the conception of electrical adders over the course of the last few decades. In their earliest forms, adders were built utilizing vacuum tubes and transistors as its primary components. The creation of integrated circuits, in which all of the essential logic gates were incorporated on a single chip rather of being scattered over several chips. The original integrated circuits, often known as IC chips, [6] had a relatively low level of integration and a very small size. In a few instances, the gate count was nothing more than an extremely low number [7]. s

The development of new technologies has enabled designers to fit hundreds of gates onto a single chip, leading to the creation of what are now known as medium scale integration chips.

The technology advanced to the point where it could now integrate on a broad scale and was capable of accommodating thousands of gates [8]. The development of very large scale integration (VLSI) made it possible for the design of circuits on this chip to include more than a thousand thousand transistors. The capacity to include an N-thousandth of a given component's transistors is the measure of its complexity. The specific integrated circuit's (IC) level of complexity grows proportionally with the number of transistors that need to be built inside it [9].

A larger number of the circuit's fundamental components would need to be used in its construction if it were a large circuit with several processing steps. In the present situation, the integrated circuits (ICs) are supposed to be constructed in such a manner that they might take less space, which would almost surely result in lower levels of power consumption. A very large scale integrated circuit (VLSI) would only be considered efficient if it could use fewer transistors [10], shorten the time it takes for signals to propagate, and waste less power. When the circuit's complexity is reduced, it will undoubtedly be able to include a bigger circuit within the confines of the allotted space on the die of an integrated circuit. Therefore, it is possible to achieve compactness, dependability owing to a reduced number of interconnections, and lower power consumption [11]. Due to the intricacy of the circuits, it is currently difficult to manually verify the adders using the breadboards. At this point in time, electronic design automation tools were developed specifically for the aforementioned procedures. The verification process and the development of the layout also required the use of certain computer-aided methods [12].

The creators of the system constructed the gate level adders by hand till the total number of gates became much lower. The verification and design of very large scale integrated circuit (VLSI) adders are both becoming more reliant on computer-aided methods. Additionally, this became common practice for the automated placement of components and the routing of circuit layouts [13].

The creation of logic simulators was required because it was essential to assess the working of these circuits before they were put on chip. As a result of this need, the development of logic simulators was required. The logic simulation was given a key role to play in the design process [14], due to the increasing complexity of the design. The designers were able to address the functional problems in the architecture and sort out the defects, if any, in the design before it was built. Before it was fabricated, the designers sorted out the flaws in the design. The major contributions of this work are as follows:

- Design and implementation of HCSLA-MBKA, which can perform the high-speed variable width addition operations.
- Implementation of MBKA for optimized sum generation with high-speed carry generation with parallel prefix calculations.

5.1 Proposed Model

Because a supplemental adder may produce carry bits in parallel anytime the CSLA's inputs are being changed, the CSLA has the capacity to generate carries at a quicker pace than other adders. The carry propagation is sped up with the help of this method, which makes heavy use of carry bypass logic. Carry look-ahead logic is the kind of logic that allows for the production and propagation of carries. The Carry Look Ahead Logic evaluates each bit in the series of binary bits that are to be added and decides whether or not the chosen bit pair will produce a carry or propagate a carry. This evaluation occurs for each bit in the sequence. The addition of the two integers serves as a precursor to the calculation of the carry, which is the result of that precursor. When the actual addition is conducted, which comes after the pre-processing of bits, there is no need for any waiting time for the ripple carry. This waiting time refers to the amount of time it takes for the carry to be passed on to the next stage Adder from the stage before it. The proposed HCSLA-MBKA with carry generate and propagate is shown in the Figure 1. The HCSLA-MBKA is capable of reducing the propagation delay in the sequential addition. The logic of the HCSLA-MBKA supports for a faster addition.

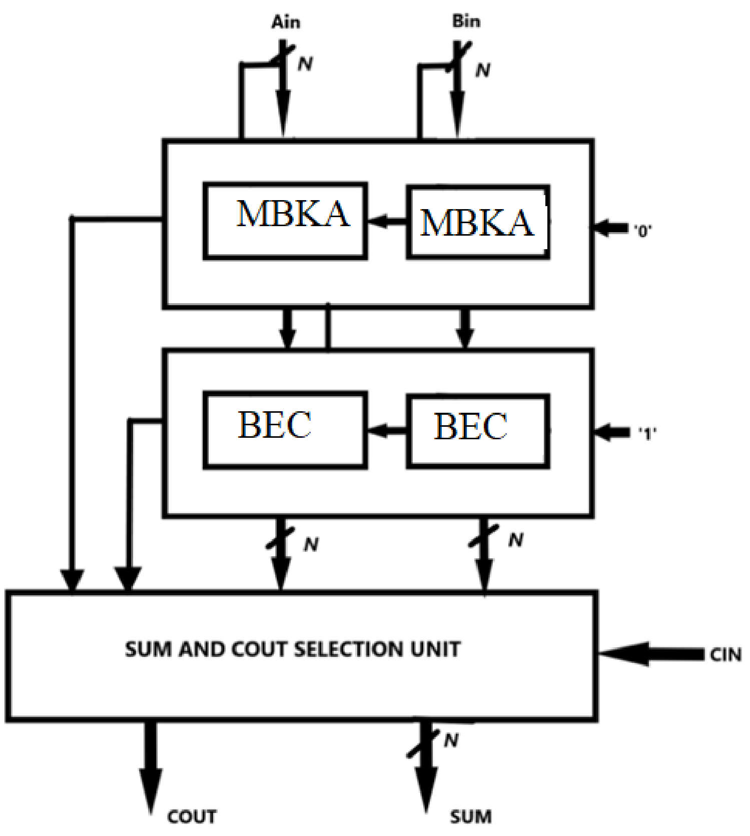


Fig : 5.1 Proposed HCSLA-MBKA Block diagram

Initially, MBKA is introduced, which is parallel prefix adder and exhibits the high-speed operations with low computational complexity. Further, MBKA is introduced in the first stage HCSLA. Then, BEC module is developed in the second stage of HCSLA and generates the sum. First the Carry-generate and Carry-propagate vectors are evaluated.

5.2 Proposed MBKA

Figure 5.2 presents a 16-bit MBKA that has an integrated pipelined design consisting of four stages. Eleven Gray Cells and fourteen Black Cells, in addition to four-stage pipeline registers, are used in the 16-bit pipelined MBKA. The pipelined MBKA functions as an Adder at a very high speed. Pipelined systems are able to be included into any parallel prefix adder topologies because to their adaptability. Although the gray and black cells that make up the various levels of each parallel prefix architecture are always the same, the amount of cells that make up each level is never the same. Pipeline may be done at several levels. The fact that there are four pipelined stages indicates that the clock speed is much faster than that of an analogous adder that does not use pipelines. An adder that does not use pipelining does not have any adder components that allow for the calculations to be synchronized with one another. Every calculation is performed in a manner that does not involve the introduction of any sequential components. In the case of pipelined adders, the greatest permissible pipeline clock rate is determined by taking the reciprocal of the maximum delay. If the overall complicated logic can be split down into a series of smaller, simpler stages that nevertheless need synchronization components, then the maximum frequency can be increased even more.

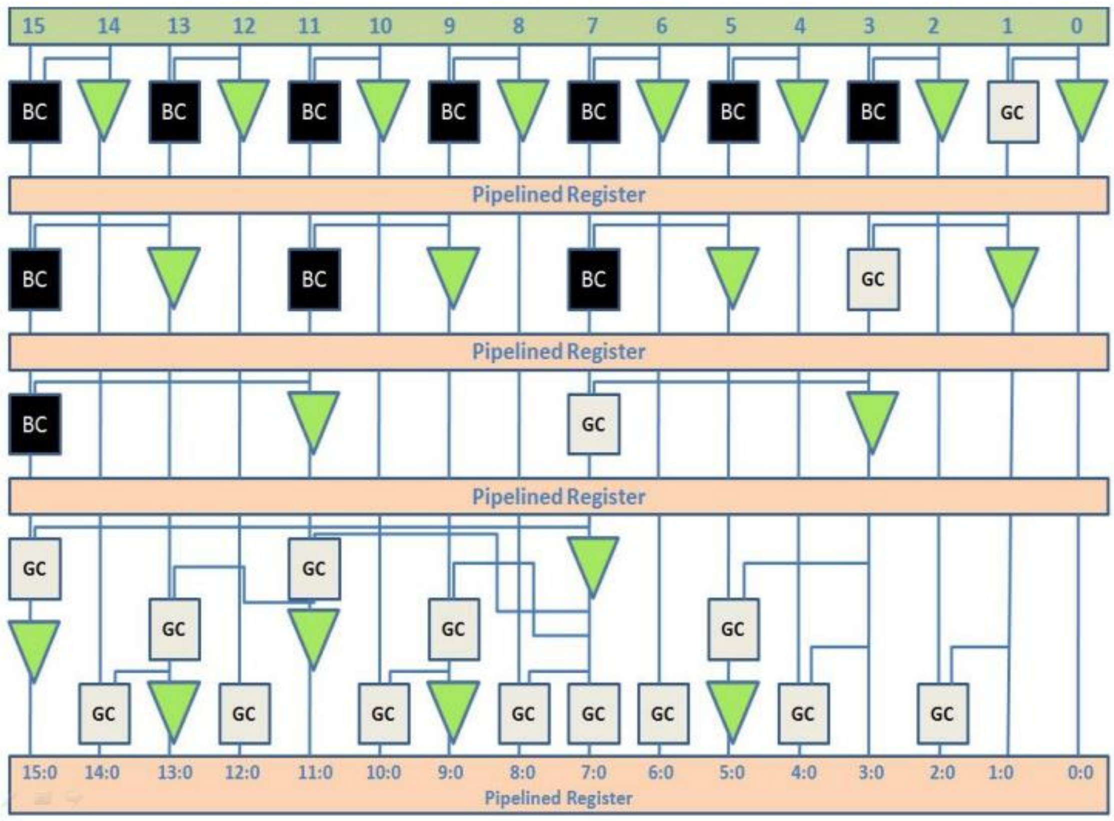


Fig : 5.2 Proposed 16-bit Pipelined MBKA Architecture.

Figure 5.3 shows the black cell (BC) and grey cell (GC) internal operational diagram. The mechanism for carry look-ahead summation can be describes as below:

$$Pi = Ai \oplus Bi \qquad (1)$$

$$Gi = AiBi \qquad (2)$$

$$Si = Ci \oplus Pi \qquad (3)$$

$$C_{i+1} = Gi + PiCi \qquad (4)$$

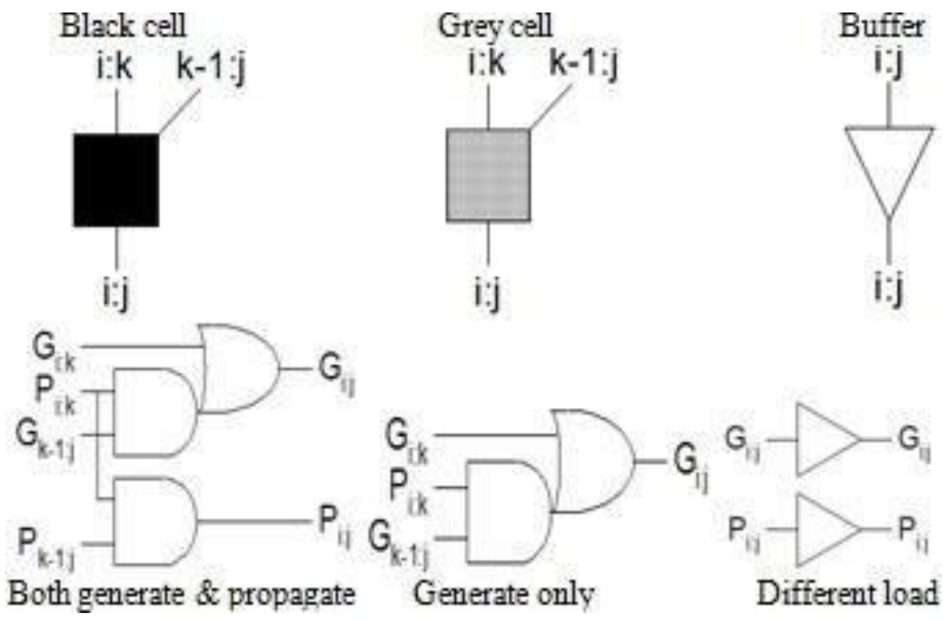


Fig : 5.3 BC and GC of MBKA.

Because a carry (C_{i+1}) is created whenever $G_i = 1$, regardless of the carry that was input, this signal is referred to as the carry Generate signal in this context (C_i). In addition, Pi is referred to as the carry propagate signal because if Pi equals 1, the carry from the input is transferred to the carry of the output, which is denoted by $C_{i+1}. = C\ i$

Ai	Bi	Ci	Ci+1	Condition
0	0	0	0	No Carry Generates
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propagates
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry Propagates
1	1	1	1	

Table : 5.1 Table for the Pi and Gi.

The conditions at which the Carry generate, no carry propagate or no carry generate for the combinations of the input data along with the carry inputs are tabulated in the Table 1, for the HCSLA adders. The Boolean expression for the carry outputs of various stages for a 4-bit block can be written as follows:

$C1 = G0 + P0\,C0$ (5)

$C2 = G1 + P1\,C1 = G1 + P1\,(G0 + P0\,C0) = G1 + P1\,G0 + P1\,P0\,C0$(6)

$C3 = G2 + P2\,C2 = G2 + P2\,G1 + P2\,P1\,G0 + P2\,P1\,P0\,C0$ (7)

$C4 = G3 + P3\,C3 = G3 + P3\,G2 + P3\,P2\,G1 + P3\,P2\,P1\,G0 +$
$P3\,P2\,P1\,P0\,C0$ (8)

CHAPTER 6

Xilinx-ISE

SIMULATION PROCESS

Step 1: CLICK ON NEW PROJECT

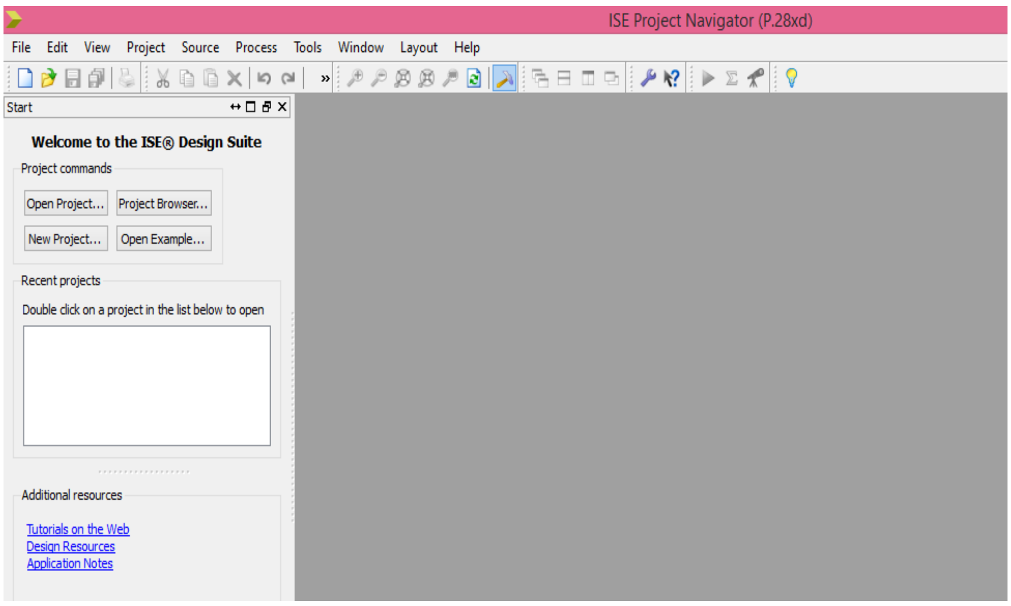
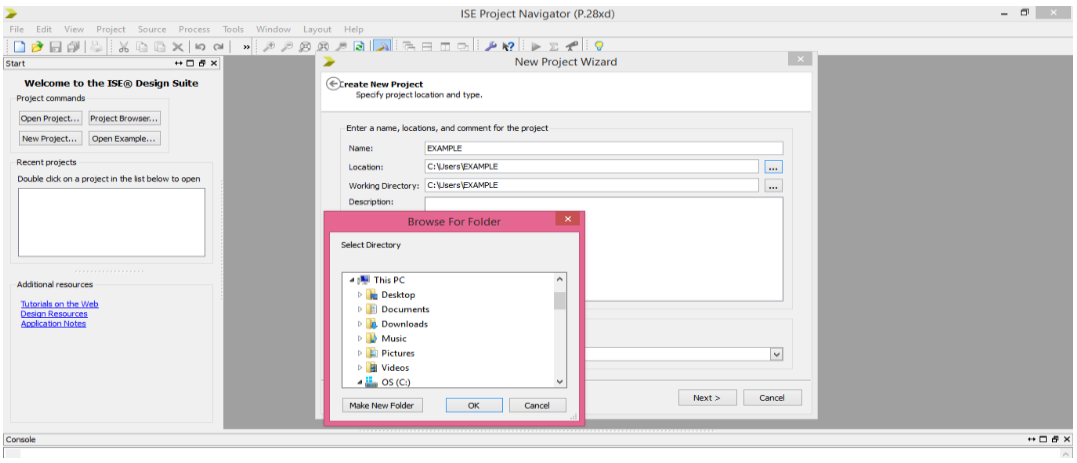

ISE Project Navigator (P.28xd)
File Edit View Project Source Process Tools Window Layout Help
Start
Welcome to the ISE® Design Suite
Project commands
Open Project...
Project Browser...
New Project...
Open Example...
Recent projects
Double click on a project in the list below to open
Additional resources
Tutorials on the Web
Design Resources
Application Notes

Fig 6.1Click on new project

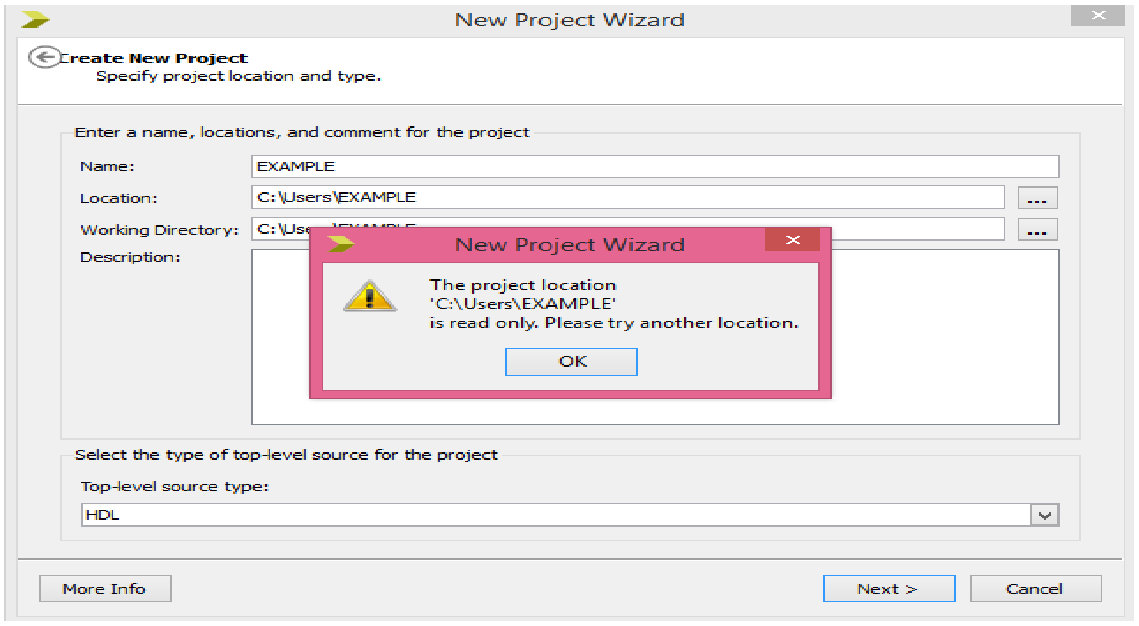
Step 2: GIVE THE PROJECT NAME AND SELECT LOCATION (WRITABLE)



Name: EXAMPLE
Location: C:\Users\EXAMPLE
Working Directory: C:\Users\EXAMPLE
Description:

Fig 6.2 GIVE THE PROJECT NAME and SELECT LOCATION

Step 3: CLICK ON NEXT and NEXT



New Project Wizard

Create New Project
Specify project location and type.

Enter a name, locations, and comment for the project

Name:	EXAMPLE
Location:	C:\Users\EXAMPLE
Working Directory:	C:\Use…
Description:	

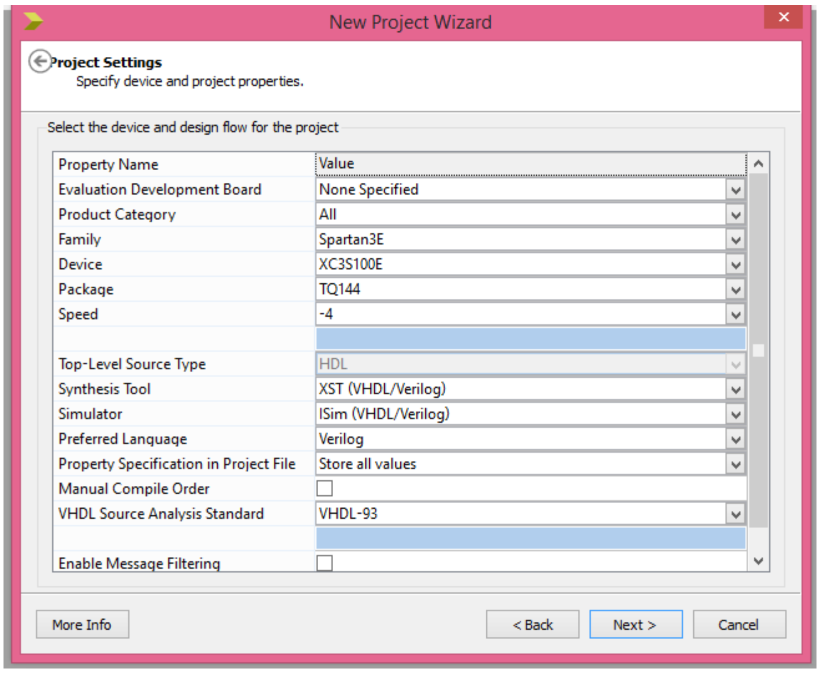
New Project Wizard

The project location
'C:\Users\EXAMPLE'
is read only. Please try another location.

Select the type of top-level source for the project

Top-level source type: HDL

Fig 6.3 CLICK ON NEXT and NEXT



New Project Wizard

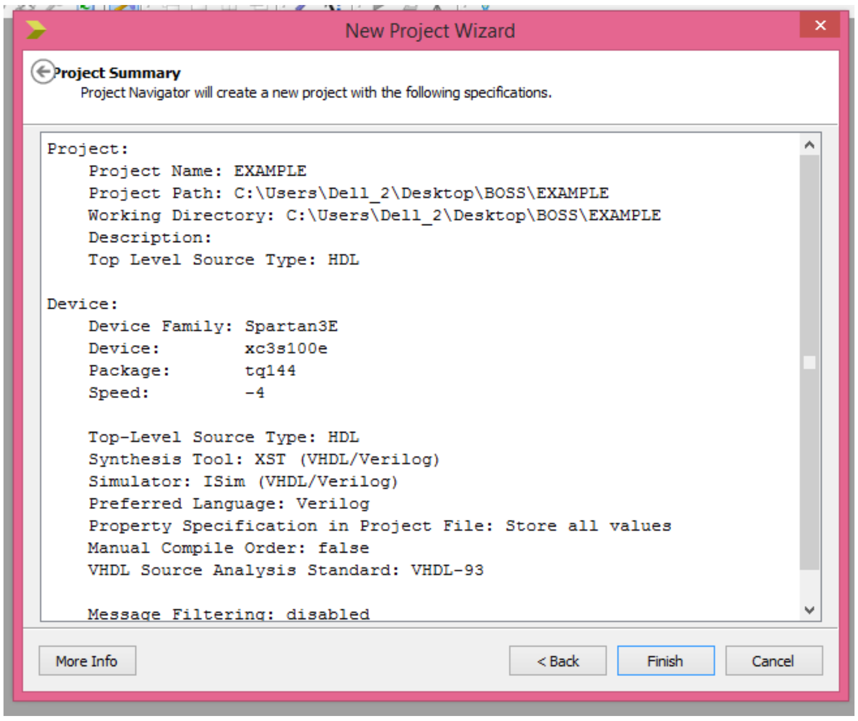
Project Settings
Specify device and project properties.

Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	TQ144
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Fig 6.4 CLICK ON NEXT and NEXT

Step 4: CLICK ON FINISH


New Project Wizard
Project Summary
Project Navigator will create a new project with the following specifications.
More Info
< Back
Finish
Cancel

```
Project:
    Project Name: EXAMPLE
    Project Path: C:\Users\Dell_2\Desktop\BOSS\EXAMPLE
    Working Directory: C:\Users\Dell_2\Desktop\BOSS\EXAMPLE
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan3E
    Device:        xc3s100e
    Package:       tq144
    Speed:         -4

    Top-Level Source Type: HDL
    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: Verilog
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

Fig 6.5 Click on finish

Step 5: CLICK ON CHIP (XC…) then MOUSE RIGHT CLICK then CLICK ON ADD SOURCE

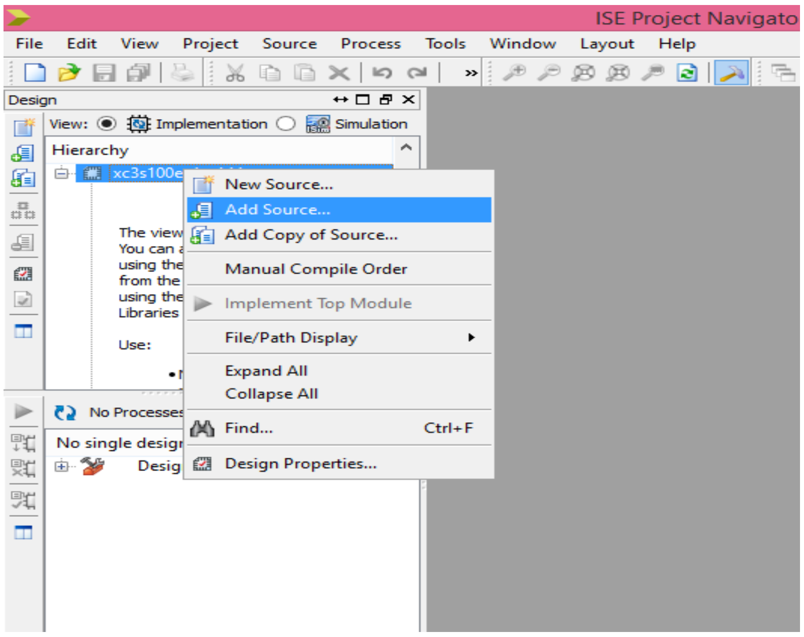
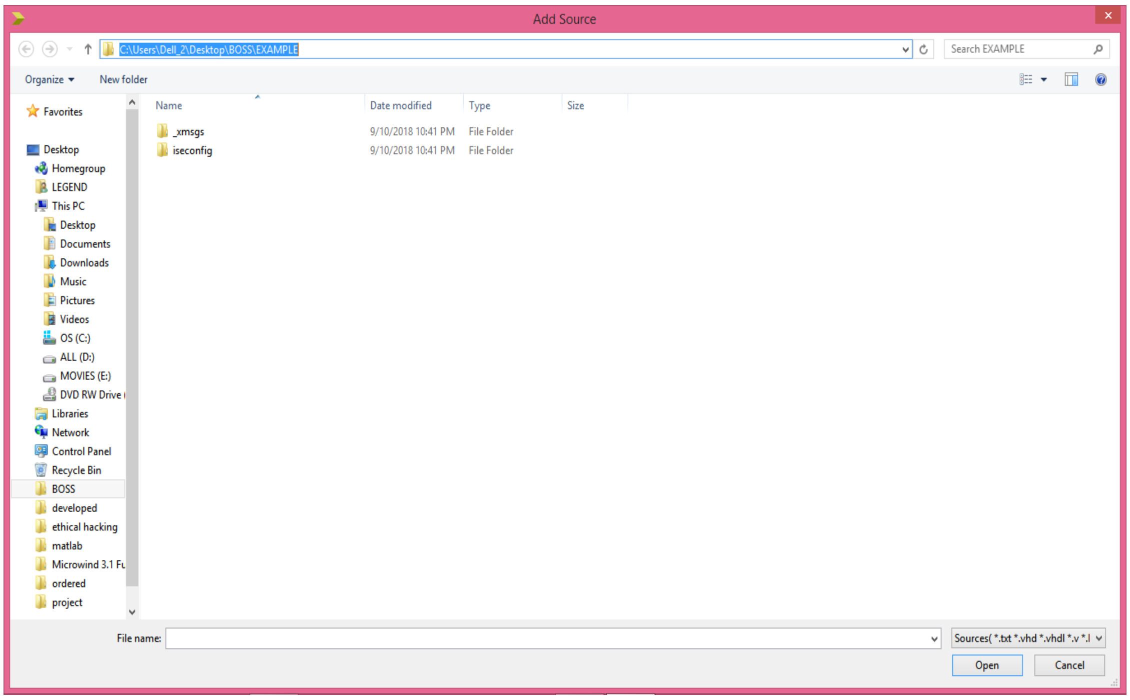


Fig 6.6 CLICK ON CHIP (XC…) then MOUSE RIGHT CLICK then CLICK ON ADD SOURCE

Step 6: SELECT THE CODE LOCATION GIVEN BY DEVELOPER AND ADD CODE

(Note ALL FILES) AND CLICK OPEN



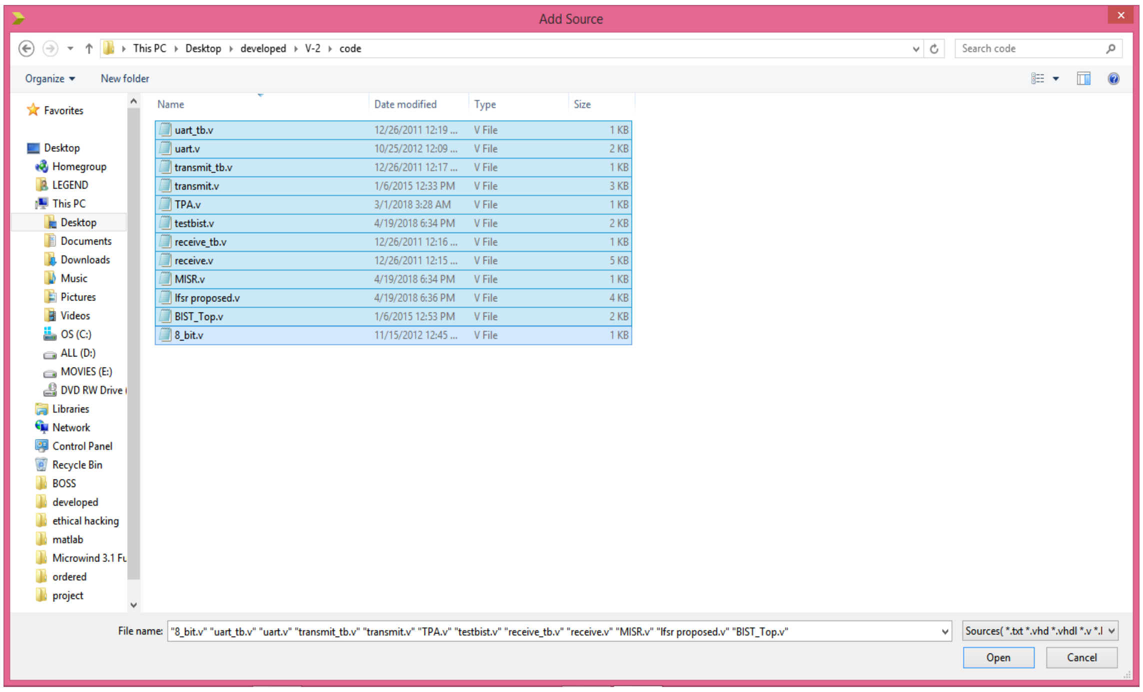
C:\Users\Dell_2\Desktop\BOSS\EXAMPLE

Name	Date modified	Type	Size
_xmsgs	9/10/2018 10:41 PM	File Folder	
iseconfig	9/10/2018 10:41 PM	File Folder	

File name:

Sources(*.txt *.vhd *.vhdl *.v *.l

Fig 6.7 Select the code location given by developer and add code



This PC › Desktop › developed › V-2 › code

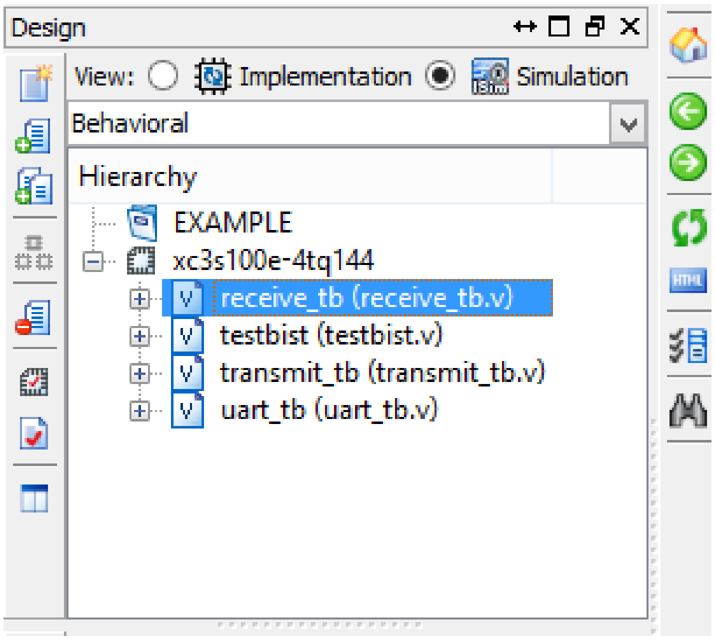
Name	Date modified	Type	Size
uart_tb.v	12/26/2011 12:19 ...	V File	1 KB
uart.v	10/25/2012 12:09 ...	V File	2 KB
transmit_tb.v	12/26/2011 12:17 ...	V File	1 KB
transmit.v	1/6/2015 12:33 PM	V File	3 KB
TPA.v	3/1/2018 3:28 AM	V File	1 KB
testbist.v	4/19/2018 6:34 PM	V File	2 KB
receive_tb.v	12/26/2011 12:16 ...	V File	1 KB
receive.v	12/26/2011 12:15 ...	V File	5 KB
MISR.v	4/19/2018 6:34 PM	V File	1 KB
lfsr proposed.v	4/19/2018 6:36 PM	V File	4 KB
BIST_Top.v	1/6/2015 12:53 PM	V File	2 KB
8_bit.v	11/15/2012 12:45 ...	V File	1 KB

File name: "8_bit.v" "uart_tb.v" "uart.v" "transmit_tb.v" "transmit.v" "TPA.v" "testbist.v" "receive_tb.v" "receive.v" "MISR.v" "lfsr proposed.v" "BIST_Top.v"

Sources(*.txt *.vhd *.vhdl *.v *.l

Fig 6.8 Select the code location given by developer and add code

Step 7: SELECT THE SIMULATION and select files to RUN


Design

View: ○ Implementation ● Simulation

Behavioral

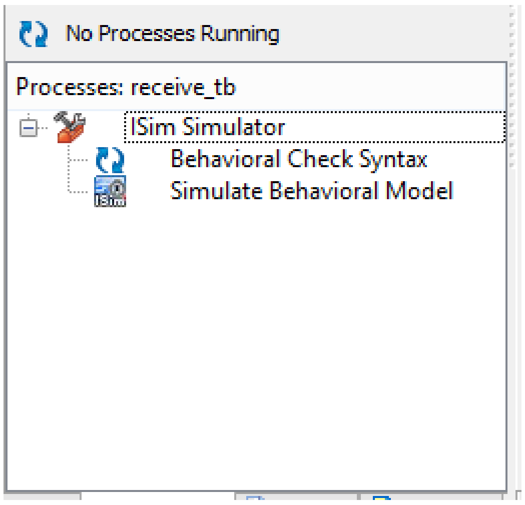
Hierarchy

EXAMPLE
xc3s100e-4tq144
receive_tb (receive_tb.v)
testbist (testbist.v)
transmit_tb (transmit_tb.v)
uart_tb (uart_tb.v)

Fig 6.9 SELECT THE SIMULATION and select files to RUN

Step 8: SELECT ISIM SIMULATOR and SIMULATE BEHAVIORAL MODEL

If no errors isim window will open


No Processes Running

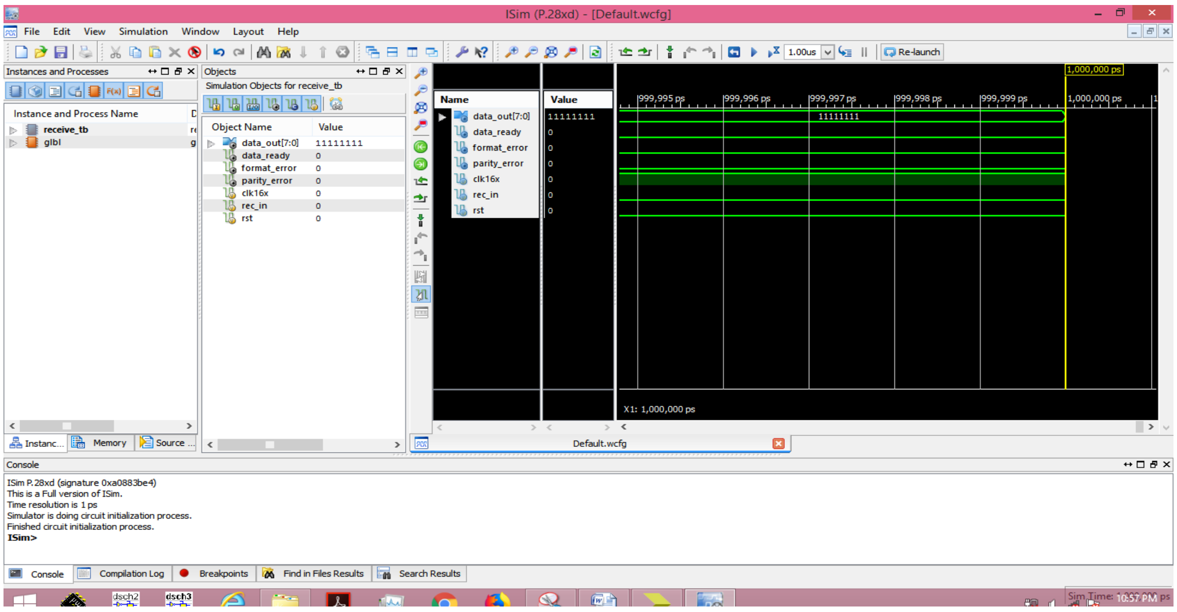
Processes: receive_tb

ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model

Fig 6.10 SELECT ISIM SIMULATOR and SIMULATE BEHAVIORAL MODEL

Step 9: ISIM WINDOW

select zoom to full view



```
Object Name        Value
data_out[7:0]      11111111
data_ready         0
format_error       0
parity_error       0
clk16x             0
rec_in             0
rst                0
```

```
ISim P.28xd (signature 0xa0883be4)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

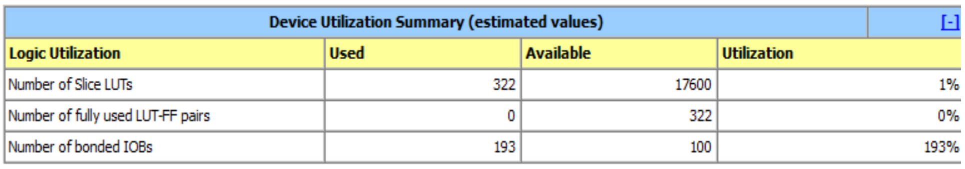
Fig 6.11 Select zoom to full view

CHAPTER 7

SIMULATION RESULT

Xilinx ISE software was used to create all of the HCSLA-MBKA designs. This software programmed gives two types of outputs: simulation and synthesis. The simulation results provide a thorough examination of the HCSLA-MBKA architecture in terms of input and output byte level combinations. Decoding procedure approximated simply by applying numerous combinations of inputs and monitoring various outputs through simulated study of encoding correctness. The use of area in relation to the transistor count will be accomplished as a result of the synthesis findings. In addition, a time summary will be obtained with regard to various path delays, and a power summary will be prepared utilizing the static and dynamic power consumption.

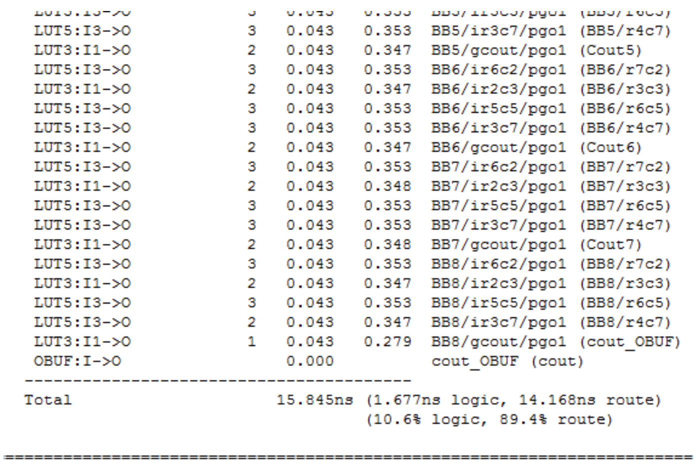
7.1 Existing Results



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	322	17600	1%
Number of fully used LUT-FF pairs	0	322	0%
Number of bonded IOBs	193	100	193%

Fig : 7.1 Existing Results

7.1.1 Design summary

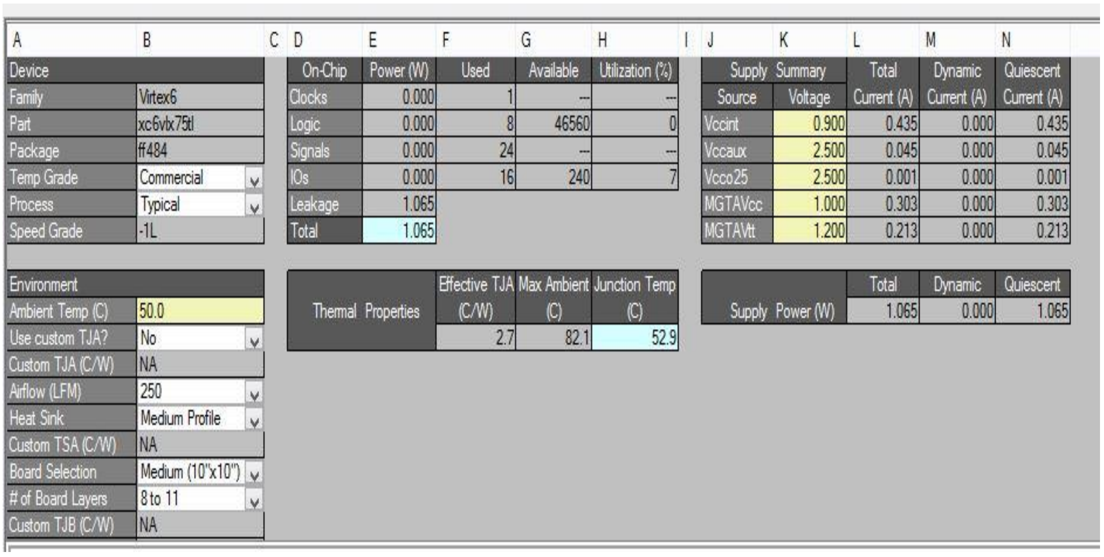


```
 LUT5:I3->O            3   0.043   0.353  BB5/ir5c5/pgo1 (BB5/r6c5)
 LUT5:I3->O            3   0.043   0.353  BB5/ir3c7/pgo1 (BB5/r4c7)
 LUT3:I1->O            2   0.043   0.347  BB5/gcout/pgo1 (Cout5)
 LUT5:I3->O            3   0.043   0.353  BB6/ir6c2/pgo1 (BB6/r7c2)
 LUT3:I1->O            2   0.043   0.347  BB6/ir2c3/pgo1 (BB6/r3c3)
 LUT5:I3->O            3   0.043   0.353  BB6/ir5c5/pgo1 (BB6/r6c5)
 LUT5:I3->O            3   0.043   0.353  BB6/ir3c7/pgo1 (BB6/r4c7)
 LUT3:I1->O            2   0.043   0.347  BB6/gcout/pgo1 (Cout6)
 LUT5:I3->O            3   0.043   0.353  BB7/ir6c2/pgo1 (BB7/r7c2)
 LUT3:I1->O            2   0.043   0.348  BB7/ir2c3/pgo1 (BB7/r3c3)
 LUT5:I3->O            3   0.043   0.353  BB7/ir5c5/pgo1 (BB7/r6c5)
 LUT5:I3->O            3   0.043   0.353  BB7/ir3c7/pgo1 (BB7/r4c7)
 LUT3:I1->O            2   0.043   0.348  BB7/gcout/pgo1 (Cout7)
 LUT5:I3->O            3   0.043   0.353  BB8/ir6c2/pgo1 (BB8/r7c2)
 LUT3:I1->O            2   0.043   0.347  BB8/ir2c3/pgo1 (BB8/r3c3)
 LUT5:I3->O            3   0.043   0.353  BB8/ir5c5/pgo1 (BB8/r6c5)
 LUT5:I3->O            2   0.043   0.347  BB8/ir3c7/pgo1 (BB8/r4c7)
 LUT3:I1->O            1   0.043   0.279  BB8/gcout/pgo1 (cout_OBUF)
 OBUF:I->O                 0.000          cout_OBUF (cout)
----------------------------------------
Total                     15.845ns (1.677ns logic, 14.168ns route)
                                   (10.6% logic, 89.4% route)

=========================================================================
```

Fig : 7.2 Design summary

7.1.2 Time summary



Device	
Family	Virtex6
Part	xc6vlx75tl
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-1L

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	8	46560	0
Signals	0.000	24	---	---
IOs	0.000	16	240	7
Leakage	1.065			
Total	1.065			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	82.1	52.9

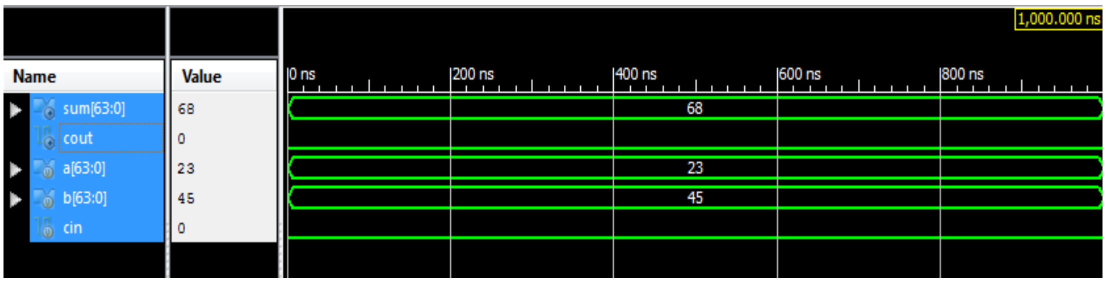
Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	0.900	0.435	0.000	0.435
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

	Total	Dynamic	Quiescent
Supply Power (W)	1.065	0.000	1.065

Fig 7.3 Time summary

7.2 Existing Method Power

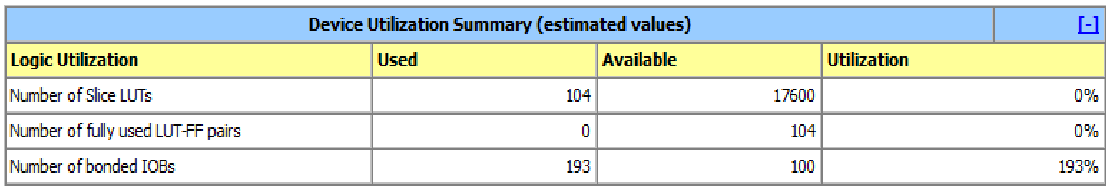
7.2.1Proposed Results



Name	Value
sum[63:0]	68
cout	0
a[63:0]	23
b[63:0]	45
cin	0

Fig : 7.4 Simulation outcome.

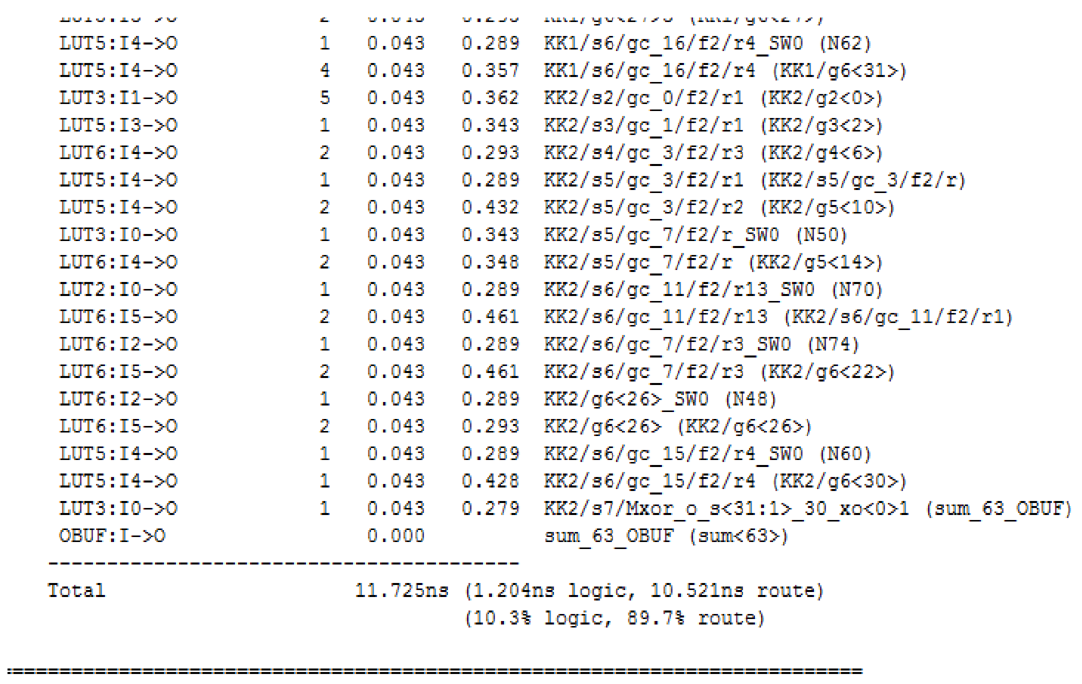
Figure 7.1 shows the simulation results of proposed HCSLA-MBKA. Here, a, b are the 64-bit input ports, cin is the one-bit input port. Further, sum is the 64-bit output port and cout is the one-bit output port.



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	104	17600	0%
Number of fully used LUT-FF pairs	0	104	0%
Number of bonded IOBs	193	100	193%

Table : 7.1 Design summary.

Table 7.1 shows the design (area) summary of proposed method. Here, the proposed method utilizes the low area in terms of slice LUTs i.e., 104 out of available 17600.

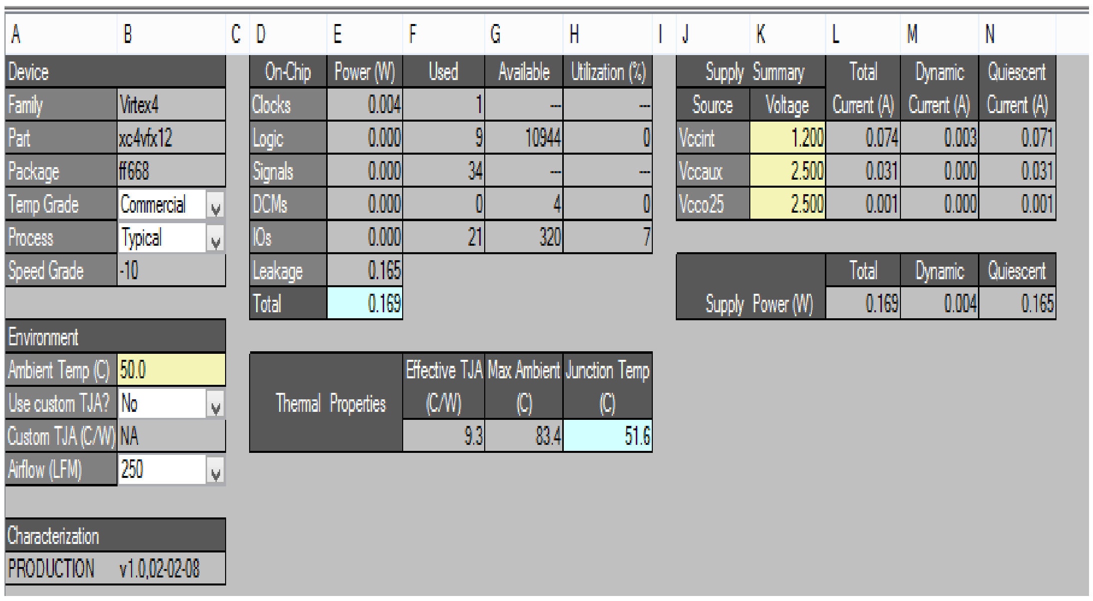


```
  LUT5:I4->O           1   0.043   0.289  KK1/s6/gc_16/f2/r4_SW0 (N62)
  LUT5:I4->O           4   0.043   0.357  KK1/s6/gc_16/f2/r4 (KK1/g6<31>)
  LUT3:I1->O           5   0.043   0.362  KK2/s2/gc_0/f2/r1 (KK2/g2<0>)
  LUT5:I3->O           1   0.043   0.343  KK2/s3/gc_1/f2/r1 (KK2/g3<2>)
  LUT6:I4->O           2   0.043   0.293  KK2/s4/gc_3/f2/r3 (KK2/g4<6>)
  LUT5:I4->O           1   0.043   0.289  KK2/s5/gc_3/f2/r1 (KK2/s5/gc_3/f2/r)
  LUT5:I4->O           2   0.043   0.432  KK2/s5/gc_3/f2/r2 (KK2/g5<10>)
  LUT3:I0->O           1   0.043   0.343  KK2/s5/gc_7/f2/r_SW0 (N50)
  LUT6:I4->O           2   0.043   0.348  KK2/s5/gc_7/f2/r (KK2/g5<14>)
  LUT2:I0->O           1   0.043   0.289  KK2/s6/gc_11/f2/r13_SW0 (N70)
  LUT6:I5->O           2   0.043   0.461  KK2/s6/gc_11/f2/r13 (KK2/s6/gc_11/f2/r1)
  LUT6:I2->O           1   0.043   0.289  KK2/s6/gc_7/f2/r3_SW0 (N74)
  LUT6:I5->O           2   0.043   0.461  KK2/s6/gc_7/f2/r3 (KK2/g6<22>)
  LUT6:I2->O           1   0.043   0.289  KK2/g6<26>_SW0 (N48)
  LUT6:I5->O           2   0.043   0.293  KK2/g6<26> (KK2/g6<26>)
  LUT5:I4->O           1   0.043   0.289  KK2/s6/gc_15/f2/r4_SW0 (N60)
  LUT5:I4->O           1   0.043   0.428  KK2/s6/gc_15/f2/r4 (KK2/g6<30>)
  LUT3:I0->O           1   0.043   0.279  KK2/s7/Mxor_o_s<31:1>_30_xo<0>1 (sum_63_OBUF)
  OBUF:I->O                0.000          sum_63_OBUF (sum<63>)
 ----------------------------------------
 Total                    11.725ns (1.204ns logic, 10.521ns route)
                                   (10.3% logic, 89.7% route)

=========================================================================
```

Fig : 7.5 Time summary

Figure 5 shows the time summary of proposed method. Here, the proposed method consumed total 11.725ns of time delay, where 1.204ns of delay is logical and 10.521ns of delay is route.



Device	
Family	Virtex4
Part	xc4vfx12
Package	ff668
Temp Grade	Commercial
Process	Typical
Speed Grade	-10

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250

Characterization	
PRODUCTION	v1.0,02-02-08

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.004	1	--	--
Logic	0.000	9	10944	0
Signals	0.000	34	--	--
DCMs	0.000	0	4	0
IOs	0.000	21	320	7
Leakage	0.165			
Total	0.169			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	9.3	83.4	51.6

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.074	0.003	0.071
Vccaux	2.500	0.031	0.000	0.031
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.169	0.004	0.165

Fig : 7.6 Power summary

Figure 7.6 shows the power consumption report of proposed HCSLA-MBKA. Here, the proposed method consumed power as 0.169 watts. Table 1 compares the performance evaluation of various HCSLA-MBKA approches. Here, the proposed HCSLA-MBKA resulted in superior (reduced) performance in terms of LUTs, time-delay, and power consumption as compared to conventional approaches such as PP-KSA [19], LPAA [22], and TOBA [24].

Metric	**PP-KSA [19]**	**LPAA [22]**	**TOBA [24]**	**Proposed HCSLA-MBKA**
LUTs	7523	5642	4834	104
Time delay (ns)	11.28	8.284	7.453	6.410
Power consumption (w)	3.49	2.34	1.349	0.165

Table : 7.2 Performance evaluation.

CHAPTER 8
CONCLUSION

The development and application of a BEC-based high-speed HCSLA are the main goals of this work. First, MBKA, a parallel prefix adder with high-speed operations and minimal computational complexity, is developed. Furthermore, the first stage HCSLA introduces MBKA. The second stage of HCSLA then develops the BEC module, which generates the sum. The simulation showed that the proposed HCSLA-MBKA performed better than conventional adders in terms of area, delay, and power. Further, this work can be extended with advanced multipliers, subtractors, dividers, and arithmetic-logic units for supporting the real time applications.

Future Scope

VLSI chips are utilized in portable devices for the processing of compute-intensive multimedia applications. One of the primary requirements for these chips is that they have designs that are efficient with both power and space. This article provides an approximation full adder (AFA) by simplifying the Boolean equations associated with its output in such a way that the amount of error is kept to a minimum despite a large drop in the gate count. In order to accomplish two new approximate ARCA and ACLA designs, the suggested AFA is applied in the modular conventional RCA and CLA. The suggested adders have a minimal level of implementation complexity, a very tiny amount of error, and a very short critical path. On the basis of error/quality measurements and performance metrics, the effectiveness of the proposed adders is assessed both as a standalone adder unit and in the applications themselves. We are able to make this job more extensive by designing a Simple accuracy reconfigurable adder (SARA). When compared to the most recent work that came before it with the same degree of precision, this one has a power/EDP that is substantially lower.

In addition, the area overhead for SARA is far smaller than it was for practically all of the works that came before it. A DAR approach contributes to an additional improvement in the accuracy-power-delay efficiency.

In the context of image processing, we illustrate the effectiveness of our adder via the applications of multiplication circuits as well as DCT computing circuits.

REFERENCES

[1].Reuben, John. " Dinesh Kumar, J. R., and C. Ganesh Babu. "Performance Investigation of a Modified Hybrid Parallel Prefix Adder for Speedy and Lesser Power Computations." *IETE Journal of Research* (2022): 1-18.Design of In-Memory Parallel-Prefix Adders." *Journal of Low Power Electronics and Applications* 11.4 (2021): 45.

[2].Thakur, Garima, Harsh Sohal, and Shruti Jain. "A novel parallel prefix adder for optimized Radix-2 FFT processor." *Multidimensional Systems and Signal Processing* 32.3 (2021): 1041-1063.

[3].Iqbal, Asma, and K. Manjunatha Chari. "Concurrent fault detection and location with minimal overhead in Ling parallel prefix adders with a scheme for fault tolerant Ling prefix adders." *Microelectronics Reliability* 127 (2021): 114375.

[4].Janwadkar, Sudhanshu, and Rasika Dhavse. "Qualitative and quantitative analysis of parallel-prefix adders." *Advances in VLSI and Embedded Systems*. Springer, Singapore, 2021. 71-88.

[5].Vijay, V., Sreevani, M., Rekha, E. M., Moses, K., Pittala, C. S., Shaik, K. S., ... &Vallabhuni, R. R. (2022). A Review On N-Bit Ripple-Carry Adder, Carry-Select Adder And Carry-Skip Adder. *Journal of VLSI circuits and systems*, *4*(01), 27-32.

[6].Janwadkar, Sudhanshu, and Rasika Dhavse. "Qualitative and quantitative analysis of parallel-prefix adders." *Advances in VLSI and Embedded Systems*. Springer, Singapore, 2021. 71-88.

[7].Janwadkar, Sudhanshu, and Rasika Dhavse. "Qualitative and quantitative analysis of parallel-prefix adders." *Advances in VLSI and Embedded Systems*. Springer, Singapore, 2021. 71-88.

[8].Thakur, Garima, Harsh Sohal, and Shruti Jain. "A Novel ASIC-Based Variable Latency Speculative Parallel Prefix Adder for Image Processing Application." *Circuits, Systems, and Signal Processing* 40.11 (2021): 5682-5704.

[9].Roy, Rajarshi, et al. "Prefixrl: Optimization of parallel prefix circuits using deep reinforcement learning." *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021.

[10].Rahimi, M., and M. B. Ghaznavi-Ghoushchi. "A novel generic modulo-2 graph with full set taxonomical conversion to parallel prefix adders." *International Journal of Circuit Theory and Applications* 50.4 (2022): 1143-1159.

[11].Kumar, Ayush, Ankit Kumar, and Deva Nand. "Design and Study of Dadda Multiplier by using 4: 2 Compressors and Parallel Prefix Adders for VLSI Circuit Designs." *2021 2nd International Conference for Emerging Technology (INCET)*. IEEE, 2021.

[12].Sudhkar, J., and E. Jagadeeswara Rao. "Design of Low-Power Parallel Prefix Adder Templates Using Asynchronous Techniques." *Communication and Intelligent Systems*. Springer, Singapore, 2022. 433-445.

[13].Yakunin, A. N., Aung Myo San, and Han Myo Htun. "Improving the Operating Efficiency of a Multibit Binary Parallel-Prefix Adder." *Russian Microelectronics* 50.7 (2021): 491-498.

[14].Gunasekaran, K., et al. "FPGA Based Implementation of Brent Kung Parallel Prefix Adder." *Recent Advances in Internet of Things and Machine Learning*. Springer, Cham, 2022. 191-198.

[15].Rahimi, M., and M. B. Ghaznavi-Ghoushchi. "A fanout-improved Parallel Prefix Adder with full-swing PTL cells and Graded Bit Efficiency." *Microelectronics Journal* 113 (2021): 105086.

[16].Roja, Ms Palagara Vineela, et al. "Implementation of Approximate Multipliers Using Exact 4: 2 Compressors and Parallel Prefix Adder." *Journal of VLSI Design and Signal Processing (e-ISSN: 2581-8449)* 8.2 (2022): 17-32.

[17].Roja, Ms Palagara Vineela, et al. "Implementation of Approximate Multipliers Using Exact 4: 2 Compressors and Parallel Prefix Adder." *Journal of VLSI Design and Signal Processing (e-ISSN: 2581-8449)* 8.2 (2022): 17-32.

[18].Muthuraman, Athappan, and Santha Karuppiah. "A Delay Efficient Hybrid Parallel Prefix Variable Latency CSKA Based Multi-Operand Adder with Optimized 5: 2 Compressor and Skip Logic." *International Journal of Electronics* just-accepted (2022).

[19].TANAKA, Tomoyuki, Christopher L. AYALA, and Nobuyuki YOSHIKAWA. "A 16-bit parallel prefix carry look-ahead Kogge-Stone adder implemented in adiabatic quantum-flux-parametron logic." *IEICE Transactions on Electronics* (2022): 2021SEP0001.

[20].Balasubramanian, P., & Mastorakis, N. E. (2022). High-Speed and Energy-Efficient Carry Look-Ahead Adder. *Journal of Low Power Electronics and Applications*, *12*(3), 46.

[21].Christilda, V., and A. Milton. "Area and delay optimized two step binary adder using carry substitution algorithm for FIR filter." *Analog Integrated Circuits and Signal Processing* 112.3 (2022): 433-441.

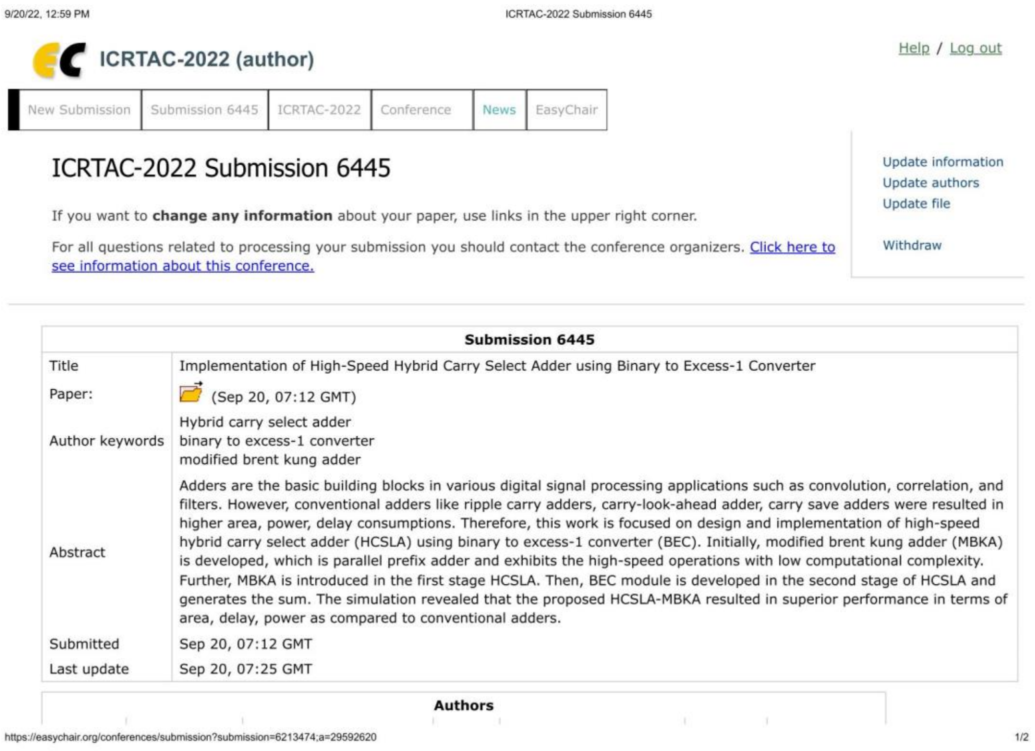
[22].Alan, Tanfer, and Jörg Henkel. "Probability-Driven Evaluation of Lower-Part Approximation Adders." *IEEE Transactions on Circuits and Systems II: Express Briefs* 69.1 (2021): 204-208.

[23].Geethanjali, G. C., et al. "Implementation and Analysis of Wallace Tree Multiplier Using Kogge Stone Adder and Sklansky Adder." *International Journal of Research in Engineering, Science and Management* 5.6 (2022): 45-49.

[24].Azeez, Saba, and Pankaj Rangaree. "FPGA Implementation of High Speed and Area Efficient Three Operand Binary Adder." *International Journal of Advanced Science Computing and Engineering* 3.1 (2021): 10-17.

[25].Reuben, J., &Pechmann, S. (2021). Accelerated addition in resistive ram array using parallel-friendly majority gates. *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, *29*(6), 1108-1121.

ANNEXURE



9/20/22, 12:59 PM ICRTAC-2022 Submission 6445

ICRTAC-2022 (author)

Help / Log out

New Submission | Submission 6445 | ICRTAC-2022 | Conference | News | EasyChair

ICRTAC-2022 Submission 6445

If you want to **change any information** about your paper, use links in the upper right corner.

For all questions related to processing your submission you should contact the conference organizers. Click here to see information about this conference.

Update information
Update authors
Update file

Withdraw

Submission 6445	
Title	Implementation of High-Speed Hybrid Carry Select Adder using Binary to Excess-1 Converter
Paper:	(Sep 20, 07:12 GMT)
Author keywords	Hybrid carry select adder binary to excess-1 converter modified brent kung adder
Abstract	Adders are the basic building blocks in various digital signal processing applications such as convolution, correlation, and filters. However, conventional adders like ripple carry adders, carry-look-ahead adder, carry save adders were resulted in higher area, power, delay consumptions. Therefore, this work is focused on design and implementation of high-speed hybrid carry select adder (HCSLA) using binary to excess-1 converter (BEC). Initially, modified brent kung adder (MBKA) is developed, which is parallel prefix adder and exhibits the high-speed operations with low computational complexity. Further, MBKA is introduced in the first stage HCSLA. Then, BEC module is developed in the second stage of HCSLA and generates the sum. The simulation revealed that the proposed HCSLA-MBKA resulted in superior performance in terms of area, delay, power as compared to conventional adders.
Submitted	Sep 20, 07:12 GMT
Last update	Sep 20, 07:25 GMT

Authors

https://easychair.org/conferences/submission?submission=6213474;a=29592620 1/2