International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 20 Issue 2 – FEBRUARY 2016.

A SURVEY ON HAN-CARLSON ADDER WITH EFFICIENT ADDERS

Kaarthik.K, PG scholar,

Dept.of Electronics and communication engineering,
M. Kumarasamy College of engineering,
Karur, Tamilnadu.
kaarthikmkce@gmail.com

Dr .C.Vivek , Associate Professor,

Dept.of Electronics and communication engineering,
M. Kumarasamy College of engineering,
Karur, Tamilnadu
vivekc.phd@gmail.com

***Abstract*— Regular CSLA uses dual Ripple Carry Adder to perform addition operation. Modified CSLA (M-CSLA) uses BEC as one circuit which reduces the area furthermore, such that the total gate count is reduced subsequently. From the architecture of Han Carlson adder it is observed that there is a possibility of reducing the delays further in partial addition components. In this research, we modify CSLA with Han Carlson adders to reduce propagation delay between gates. Our proposed adders are tree structure based and are preferred to speed up the binary additions. This work estimates the performance of proposed design will be better in terms of Logic and route delay. The experimental results will show that the performance of HC with parallel prefix adder is faster and area efficient compared to conventional modified CSLA.**

Keywords-component:, Carry select adder (CSLA), Carry Look-Ahead Adder (CLA), Ripple Carry Adder (RCA), Han Carlson (HC), Binary to excess code (BEC)

I. INTRODUCTION

Addition is a fundamental operation for any digital system, digital signal processing (DSP) or control system. A fast and accurate operation of a digital system is greatly influenced by the performance of the residential adders. Adders are also very important component in digital systems because of their extensive use in basic digital operations such as subtraction, multiplication and division. Hence, improving performance of digital adder would highly advance the execution of binary operations inside a circuit contained those blocks. The performance of a digital circuit block is gauged by analyzing its power dissipation, layout area and its operating speed.

The Carry Select Adder (CSA) provides a compromise between small areas but longer delay Ripple Carry Adder (RCA) and a large area with short delay Carry Look-Ahead Adder (CLA) [1]. In mobile electronics, reducing the area and power consumption are key factors in increasing portability and battery life. Even in the servers and desktop computers, power consumption is an major design constraint. Design of area- and power-efficient high-speed data path logic system are the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time requirement to propagate a carry through the adder. The sum for each bit position in elementary adder is generated sequentially after the previous bit position has been summed and a carry propagated into the next position [3]. Among different types of adders, the CSA is intermediate regarding speed and area [2].

VLSI Integer adders find the applications in Arithmetic and Logic Units (ALU's), microprocessors and memory addressing units. Speed of the adder frequently decides the minimum clock time in a microprocessor. The need for a Parallel Prefix adder is that it is primarily fast on comparison with ripple carry adders. Parallel Prefix adders (PPA) are family of adders derived from the common carry look ahead adders.

These adders are well suited for adders with wider word lengths. PPA circuits uses a tree network to reduce the latency to be O(log2 n) where 'n' represents the number of bits. A three stage process is generally involved in the construction of PPA. The first step involves the creation of generate, complementary skill and propagate signals for all the input operand bits.

$$G_i = A_i \bullet B_i \qquad (1)$$

$$K_i = \overline{A_i + B_i} = \overline{A_i} \bullet \overline{B_i} \qquad (2)$$

$$P_i = \overline{A_i \oplus B_i} \qquad (3)$$

Second step involves the generation of carry signals. In Parallel Prefix Adders, the dot operator ' • ' and the semi-dot operator ' ∘ ' are introduced. The dot operator ' • ' is defined by the equation (4) and the semi-dot operator ' ∘ ' is defined by the equation (5)

$$(g_i, \overline{k_i}) \bullet (g_{i-1}, \overline{k_{i-1}}) = (g_i, + \overline{k_i} g_{i-1}, \overline{k_i} \cdot \overline{k_{i-1}}) \quad (4)$$

$$(g_i, \overline{k_i}) \circ (g_{i-1}, \overline{k_{i-1}}) = (g_i, + \overline{k_i} g_{i-1}) \quad (5)$$

In the above equation, ' • ' operator is applied on two pair of bits $(g_i, \overline{k_i})$ and $(g_{i-1}, \overline{K_{i-1}})$. These bits represent, generate and propagate signals used by addition. The output of the operator is a new pair of bits which is once again combined using a dot operator ' • ' or semi-dot operator ' ∘ ' with another pairs of bits. This procedural use of dot operator ' • ' and semi-dot operator ' ∘ ' creates a prefix tree network which ultimately ends in generation of all carry signals. In the final step, the sum bits of the adders are generated with the propagate signals of operand bits and the preceding stage carry bit using a xor gate. The semi-dot operator ' ∘ ' will be obtained as last computation node in each column of the prefix graph structures, where it is essential to compute only generate term, whose value is the carry generated from that bit to the succeeding bit.

II. REQUIREMENTS

A. Design

We propose a high speed Carry Select Adder by replacing Ripple Carry Adder with parallel prefix adder. Adders are the basic building blocks in digital integrated circuit based designs. Ripple Carry Adder (RCA) is usually preferred for addition of two multi-bit numbers as these RCA offer fast design time among all types of adders. However RCAs are slowest adder as every full adder must wait until the carry is generated from previous full adder. On the other hand, Carry Look Ahead (CLA) adder are faster adder, but they required more area. The Carry Select Adder is a compromise on between the RCA and CLA in terms of area and delay. CSLA is designed by using dual RCA: due to this arrangement the area and delay are concerned factors. It clears that there is a scope for reducing delay in such arrangement. In this research, we have implemented CSLA with parallel prefix adders.

Parallel prefix adders are tree based structure and are preferred to speed up the binary additions. This process estimates the performance of proposed design in terms of logic and route delay. The experimental results show the performance of CSLA with parallel prefix adder is fast and area efficient compared to conventional modified CSLA.

B. Functionality

In addition to the final deadline, each section of the project was given separate deadlines to ensure each design group was making sufficient progress throughout the semester. The first deadline required us to turn in the ADD, OR, PASS A, 8:1 MUX functions, as well as an arbitrary function that we chose on our own, and the second design review required the ADD, SUB, SHIFT, ALU, in/out connectivity, and registers working. Since we had already finished those parts previously, the final report does not cover those individual components, but it does require that our ALU be able to complete each function and demonstrate its correctness.

The total list of functions that our ALU must complete is listed in Table 1.

Table 1. Required ALU functions

ALU Function	Description
NOP	Do nothing – No change at Out
ADD	Out = A + B
SUB	Out = A - B
SHIFT	Out = A << B
AND	Out = A & B
OR	Out = A | B
Pass A	Out = A
Arbitrary	Out = <function of your choice>

C. Metric

A single full-adder is capable to add two one-bit numbers and an input carry. In order to add binary numbers which is more than one bit, the full-adders must be employed in addition. A n-bit parallel can be constructed using number of full adder circuits connected in parallel.

The parallel adder is ripple carry adder in which the carry output of each full-adder stage is connected with the carry input of the next higher-order stage. Therefore the sum and carry output of any stage cannot be produced until the input carry occurs; this leads to time delay in the addition process. The delay is known as carry propagation delay.

The ripple carry adder is constructed by cascading full adders (FA) blocks in series. One full adder is responsible for the addition of two binary digits at any stage of the ripple carry. The carryout of one stage is fed directly to the carry-in of the next stage. Even though this is a simple adder and can be used to add unrestricted bit length numbers, it is however not very efficient when large bit numbers are used. One of the most serious drawbacks of this adder is that the delay increases linearly with the bit length.

One method of speeding up the process by eliminating the inter stage carry delay is called carry look-ahead addition. This method utilizes logic gates to look at the lower order bits of the augends and addends to see if a higher-order carry is to be generated.

The use of one half-adder or one full-adder are great for add up two binary numbers with a length of one bit each, but when the computer needs to add up two binary numbers with a longer length, there are several ways of doing this. The fastest way is to use the Parallel Binary Adder. The Parallel Binary Adder uses one half-adder, along with one or more full adder.

The total number of adder needed depends on the length of the largest of the two binary numbers that are to be added. For example, if we need to add up the binary numbers 1011 and 1, we would need four adder in total, because the length of the larger number is four, by keeping this in mind, here is a demonstration of how a four-bit parallel binary adder works, by 1101 and 1011 as the two numbers to add:

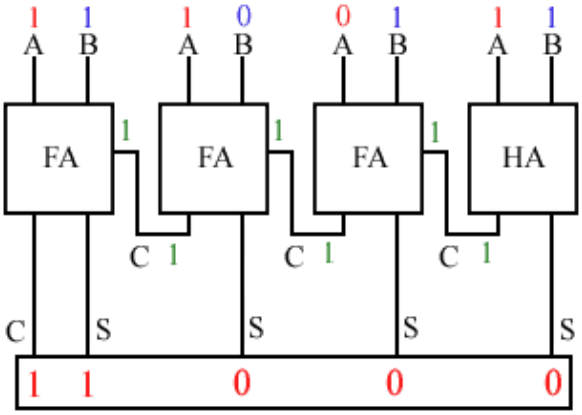

Fig 1. Ripple-Carry Adder

When we add with the computer, it adds from right to left. Just like when we add without the computer, in the parallel binary adder is a step by step list, Fig.1 showing you what happens in the parallel Binary Adder.

D. Specification

Each of the three metrics have been specifically stated how they will be evaluated. Active power is measured for one computation per cycle at the highest frequency achievable by the design for a specific series of inputs, which PICo will supply at the second design review. The *delay* is the worst case access delay. The *area* is the sum of the widths of transistors used in the design."

In addition, our design is assumed to interface with pads that connect to the outside world with all inputs valid .5 FO4 delay before the rising edge of the clock, and hold for 1 FO4 delay after the rising edge of the clock. Therefore, we assumed that the clock is an ideal signal driven through a static CMOS buffer.

III. DESIGN

During our design process, we encountered several design decisions that we had to make to reduce our overall metric. We made our original designs for each sub-circuit when they were due for the design reviews, however, we did not take into account metric decisions then.

When it came time to reduce the overall delay, power, and cost of our design, we went to each individual sub circuit and evaluated how we could reduce the specifications for that sub circuit, in an effort to reduce the overall sub circuit. For many part of our processor, we chose to use new designs, such as new Adders, reduced the number of gates where we could, and sized our transistors proportionally in order to be the most efficient.

Images of our designs can be found attached as Appendices to this document

A. Adder

The most important decision in our Digital Signal Processor was choosing the adder, as its delay would be significantly greater than any other function, and so would be the determining factor in our maximum speed. In all cases, the adder was converted to an adder/ subtractor by adding an inverter, a 2:1 multiplexer, and a select line. This line was determined to be 0 for add, and 1 for subtract. This line also was the carry in for the entire adder, which gave a 2's complement version of B when subtract was selected.

1) Ripple Carry Adder : Originally we used a ripple-carry adder, which gave us a large delay of 11ns, but was simple to implement by chaining together full Adders. The full adders were designed using the mirror adder pattern as Fig. 3rather than the full static CMOS design. The large delay was a result of each full adder having to wait for the carry bit to be calculated from the previous full adder, and as a result, a large number with 16 bits would take a long time to fully calculate. Because of the large delay, we searched for faster adders to increase speed as in Fig.2.

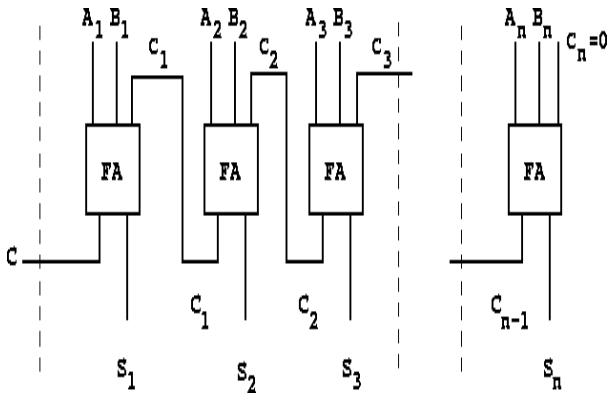

Fig 2. Ripple-Carry Adder

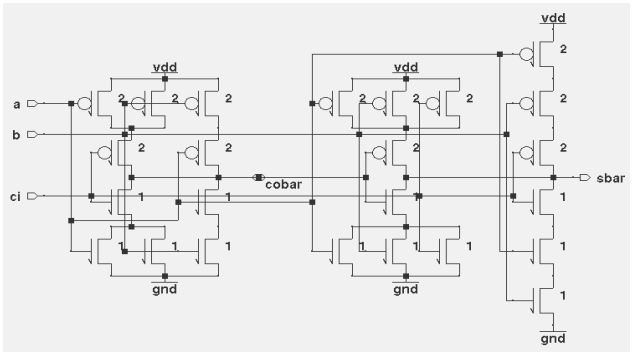

Fig 3. Mirror Full Adder

2) Carry Look-Ahead Adder: Carry Look-Ahead adders were designed to reduce overall computational time by using propagate and generate signals for each bit position, based on whether a carry is propagated through to the next bit.

The carry lookahead adder (CLA) solves the carry delay problem by calculating the carry signals in advance, based on the input signals. It is based on the fact that a carry signal will be generated in two cases: (1) when both bits and a_i are b_i 1, or (2) when one of the two bits is 1 and the carry-in is 1.

This sequence of adding from the least significant bit and propagating the carry bit ahead reduces the overall delay, but is more complex than the Ripple-Carry adder, and also uses more transistors overall. Due to its complexity and size, as well as the possibility of a Manchester adder see Fig.4, we decided not to use a carry look-ahead adder.

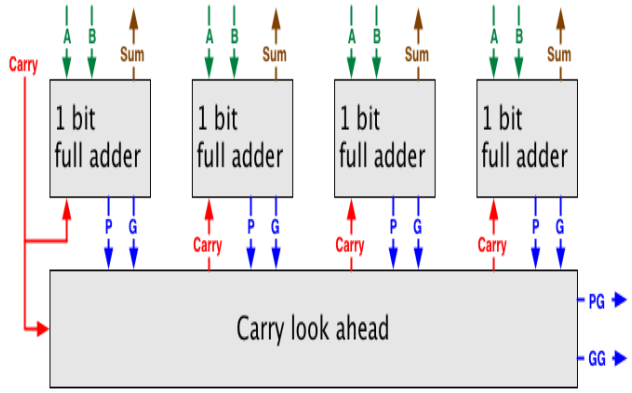

Fig 4. Carry Look Ahead Adder

3) Carry Select Adder: In the carry select adder, there are two full adders, each of which takes a different preset carry-in bit. The sums and carry-out bits that are produced are then selected by the carry-out from the previous stage.

One of the earliest logarithmic time adder designs is based on the conditional - sum addition algorithm. In this scheme, blocks of bits are added in two ways: assuming an incoming carry of 0 or of 1, with the correct outputs selected later as the block's true carry-in becomes known. This is one of the speed-up techniques that is used in order to reduce the latency of carry propagation as seen with the ripple-carry adder.

Basically, the adder will add the sum with and without a carry from the previous stage, and will then use a multiplexer to determine which sum is the correct one, depending on whether or not there was a carry. The basic design for the carry select adder is shown in Fig.5.

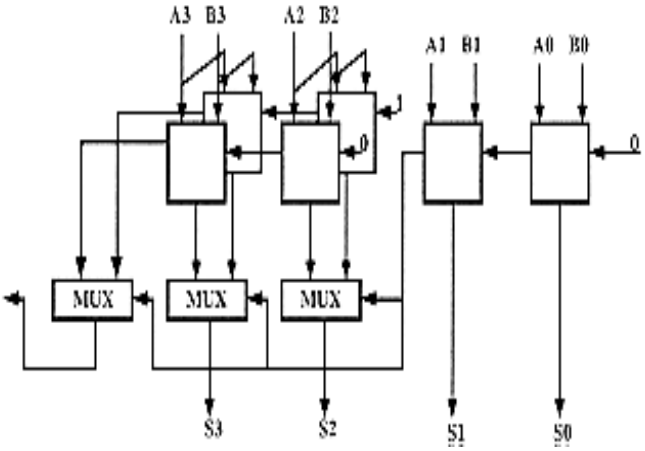

Fig 5. Carry Select Adder

The Carry Select adder was more efficient than the Ripple-Carry Adder, with a delay of 8ns. It was more difficult to implement, but since we had already finished our 2:1 multiplexers, it wasn't too difficult. The major problem with the carry select adder was the size, well over double that of the ripple carry adder. This was too large a price to pay in cost to get only a 3ns increase in speed.

4) Manchester Carry Adder : The Manchester Carry Chain is a variation of the Carry Look-Ahead adder, but instead uses shared transistor logic to lower the overall transistor count. The Manchester Carry Adder consists of cascading chains of Manchester Carry chains, which is broken down in order to reduce the number of series-propagate transistors, resulting a great reduction in delay as the number of transistors in series is reduced. As with the Carry Look-Ahead adder, it was too complex to be used in this design is shown in Fig.6.

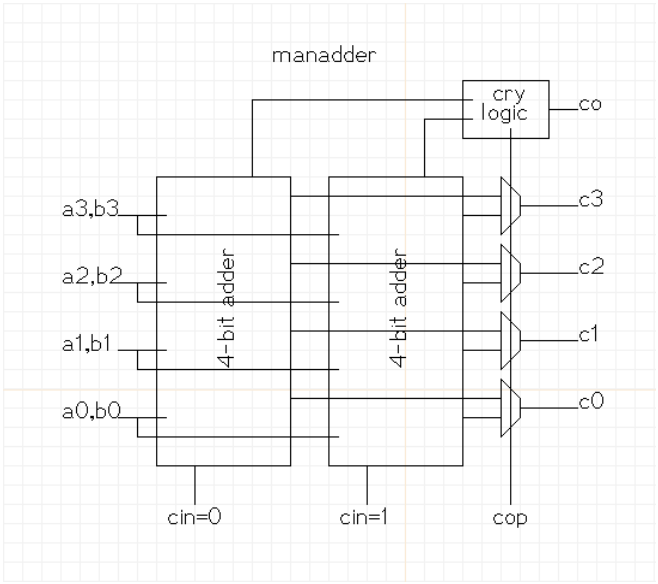

Fig 6. Manchester Carry Adder

B. *Transmission Gates*

Some sub-circuits of our design also make use of transmission and pass gates, depending on the situation. We decided to use pass gates in certain instances because it reduced our overall number of transistors required, which meant less power and cost. Their was also a slight reduction when we used pass gates in most cases, though we could only use these gates when the next input was buffered, in order to restore the signal to its full power is shown in Fig.7.

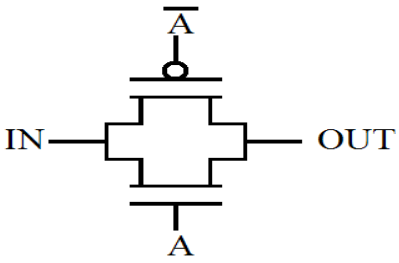

Fig 7. Transmission Gate

C. *8:1 Multiplexer*

For our 8:1 Multiplexer, we originally used four 2:1 multiplexers combined with a 4:1 multiplexer, which worked well. Our 4:1 multiplexers were created out of two 2:1 multiplexers, combined with another 2:1 multiplexer to selected between all 4 inputs, as shown in Fig.8.

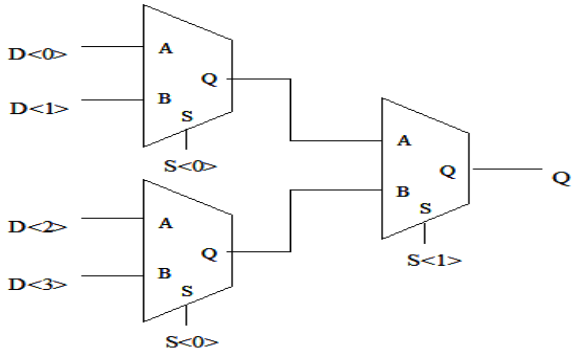

Fig 8. 4:1 Multiplexer

However, we realized that we could implement the same function using two 4:1 multiplexers combined with a 2:1 multiplexer to meet the requirement as well, and it would also offer a better delay, with fewer transistors.

D. *Parallel Prefix Adder*

Parallel Prefix Adder (PPA) is very useful in today's world of technology because of its implementation in Very Large Scale Integration (VLSI) chips. The VLSI chips rely heavily on fast and reliable arithmetic computation. These contributions can be provided by PPA. There are many types of PPA such as Brent Kung , Kogge Stone , Ladner Fisher , Hans Carlson and Knowles.
For the purpose of this research, only Han Carlson adder will be investigated with the other types of adders.

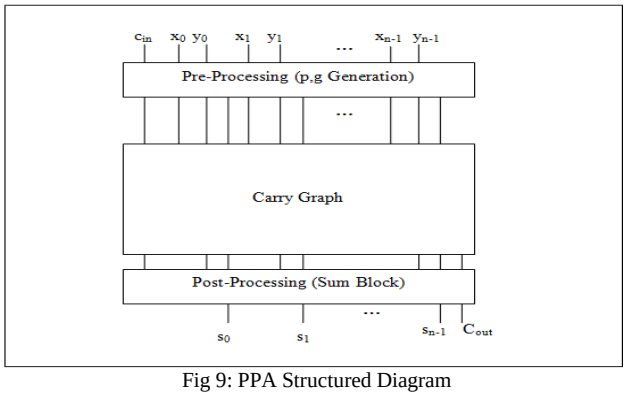

Fig 9: PPA Structured Diagram

The design file has to be analyzed from Fig.9, synthesis and compile before it can be simulated. Simulation results in this project come in the form of Register Transfer Level (RTL) diagram, functional vector waveform outcome and classic timing analysis. The RTL design can be obtained by using the RTL viewer based on the Netlist viewer. Functional vector waveform outcome are produced by selecting random bit values and add up to produce the sum and carry bits. Timing analysis can be obtained by viewing the summary of the classic timing analysis after compiling the whole project. The
simulations are done by using the functions.

Simulation analysis is prepared by viewing the results from the simulated VHDL source code. Analysis of the simulation is performed once the desired simulation outcome is obtained. Simulation results show the classic timing analysis, RTL schematic diagram and also vector waveform outcome of the simulated designs. The analysis of the PPA is conducted by viewing the time delay produced by Han Carlson adders in performing bits addition as displayed in Fig.11.

Finally, the PPA comparison will be made once all six simulation results are analyzed. Han Carlson adder and the various efficient adders will be compared at this stage and will be conducted in its bit category. The comparisons will be based on the computational speed or also known as time propagation delay and area (cost) from Fig.10.

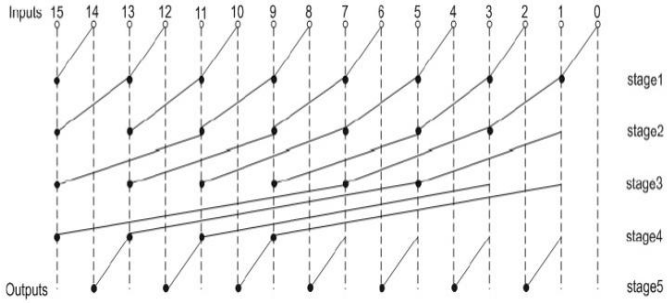

Fig 10. Han Carlson adder

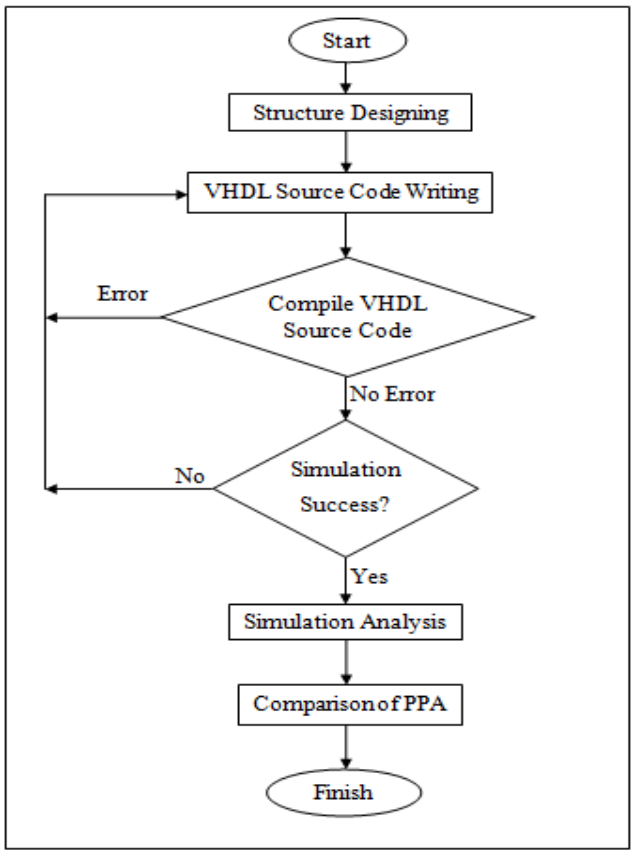

Fig 11. System design Flow chart

IV. SURVEY RESULTS

We tested all of our components using simulation through the modelsim 6.2, the area, power and delay of the various adders are been compared and obtained the outputs in the form of chart. In this obtained output the power distribution of the Han Carlson adder is higher when compared with the all other adders which could be highly efficient for an IC chip to provide the results. The delay is also an major term to be concerned while the execution of results, if the delay is higher, then the processing speed will be noticed as less even it is an efficient adder the delay is higher it is not been considered by anybody for their work. The delay is also be a major factor , in Han Carlson adder it lower when compared with the other types of adders which could be a major advantage for our adder.

The area of an IC chip is been decided by the number of gates used in our project to obtain the expected result. In the Han Carlson adder the number of gates used is less , if the number of gates is reduced the area will also been gets reduced in chip. It can able use the special technique called as the folding transformation technique which could be highly useful to obtain the result with less number of gates where in the other type of adders this technique is not applicable, So in area wise also the Han Carlson adder is highly efficient and produce the results

without speculation see the Performance analysis from Fig.12.

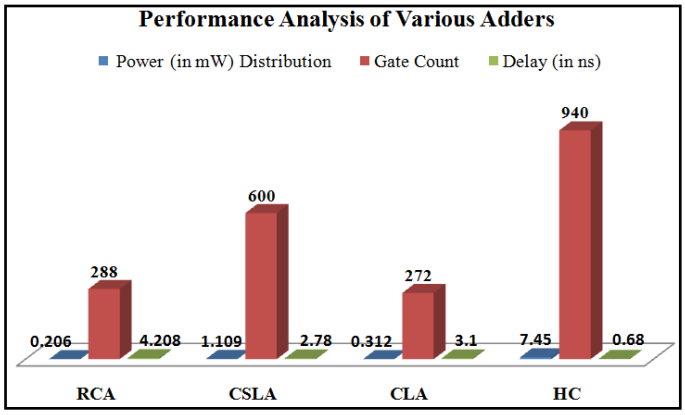

Fig 12. Performance Analysis of Adders

V. ACKNOWLEDGEMENTS

Our thanks to M.Kumarasamy college of Engineering for offering us the opportunity to do this wonderful project, and to Dr. V. Kavitha for her Guidance to do the survey.

VI. REFERENCES

[1] Bender, Ryan (April 17, 2000). A Simulator for Digital Circuits. Massachusetts Institute of Technology. Retrieved on April 28, 2008 from http://mitpress.mit.edu/sicp/full_text/sicp/book/node64.html

[2] Alan, Elay (2007). Hierarchal Schematics and Simulation Within Cadence. University of California at Berkley. Retrieved on April 28, 2008 from http://bwrc.eecs.berkeley.edu/Classes/ICDesign/EE141_f07/CadenceLabs/hierarchy/hierarchy.htm

[3] Lin, Charles (2003). Half Adders, Full Adders, Ripple Carry Adders. University of Maryland. Retrieved April 28, 2008 from http://www.cs.umd.edu/class/sum2003/cmsc311/Note/Comb/adder.html

[4] Mlynek, D. Design of VLSI Systems. EPFL. Retrieved on April 28, 2008 from http://lsiwww.epfl.ch/LSI2001/teaching/webcourse/ch06/ch06.html

[5] Lie, Sean. (2002). Carry Select Adder Details. Retrieved April 28, 2008 from http://www.slie.ca/projects/6.371/webpage/cryseladderdetails.html