International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.04, Issue.06,
June-2016,
Pages:0401-0406

www.ijvdcs.org

Recursive Approach to the Design of CSLA D-Latch Based Parallel Self Timed Adder

K.SRI HARI[1], GOPI ALOKAM[2], L.CHANDRA SEKHAR[3]

[1]Professor, NRI Institute of Technology, Guntur, Andhra Pradesh, India, Email: ecehod@nriit.ac.in.
[2]PG Scholar, NRI Institute of Technology, Guntur, Andhra Pradesh, India, Email: gopi.alokam443@gmail.com.
[3]Assistant Professor, NRI Institute of Technology, Guntur, Andhra Pradesh, India, Email: chandrasekhar.mtech13@gmail.com.

Abstract: This paper presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multi bit binary addition. The Carry Select Adder is used in many systems to relieve the problem of carry propagation delay which is happen by independently generating multiple carries and to generate the sum then select a carry. Due to uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input However, the CSLA is not time efficient, then by the multiplexers the final sum and carry are selected. The basic idea of this work is to achieve high speed and low power consumption by use Binary to Excess-1 Converter (BEC) instead of RCA in the regular CSLA. At the same time to further reduce the power consumption, a new approach of CSLA with D LATCH is proposed in this project. In the proposed scheme, before the calculation of-final-sum the carry select that is specified as CS operation is scheduled. For logic optimization of Carry selection bit patterns of two anticipating carry words that is corresponding to c_{in} = 0 and 1 and fixed c_{in} bits are used. Using optimized logic units an efficient CSLA design is obtained. The proposed Carry Select Adder design involves significantly less area and power than the recently proposed BEC-based CSLA.

Keywords: CSLA, RCA, BEC, D-LATCH, Self-Timed Adder.

I. INTRODUCTION

Now days the portability of the electronic component have rapid growth, the low power arithmetic circuit has become very important in VLSI industry. In The digital signal processor (DSP) main building block is the Multiplier-Accumulator (MAC) unit. Full Adder used as a part of the MAC unit uses full-adder as part which can significantly influences the efficiency of total system. Full Adder circuit is necessary for low power application due to the reduction in power consumption. The basic operation Carry Select Adder (CSLA) is parallel computation. CSLA generates many carriers and partial sum. Multiplexers select the final sum and carry. In the CSLA architecture, Addition operation usually trembles widely the overall performance of digital systems and a crucial arithmetic function. The adders are most widely used in the electronic applications. In the year 2002, a new concept of adders are come into existence those are called as the hybrid adders and those are used for increases the speed of addition process by Wang et al. the adders gives hybrid carry look-ahead/carry select adders design. In 2008, the new hybrid full adders are used for designing low power multipliers. In digital adders, the speed of addition is mainly based on the propagation delay, it is having the limitation by propagating delay through adder. In VLSI Design one of the most important research is the area and power optimized data path logic systems. The adders are most widely used in electronic system and applications. If we want design multipliers the concept of adders comes in to the picture because the adders are part of the multipliers designs. As we know millions of instructions per second are performed in microprocessors. In the microprocessor device area and power consumption are the most important factors in the designing multipliers and adders. The power consumption and area should low in the microprocessors. Devices like computers Mobile phones, Laptops etc...Those achieve more battery backup. So, a VLSI designer has to make perfect these parameters in a design. These are main constraints, so these are very difficult to achieve .so the constraints have to be made depending on demand or application of the circuit in the industry. the N full adders used in this architecture link together with N bit Ripple carry adder. In the ripple carry adder operation the carry out of previous full adder becomes the input carry for the next full adder. Like that sum and carry calculated at the end of the last full. As carry ripples from one full adder to the other, if the size of the full adder is high then it has a more delay.

II. REGULAR CSLA

The CSLA has two units as shown in below one is the sum and carry generator unit that is simply define as SCG and the other one is define as the sum and carry selection unit. To the critical path the SCG unit consumes most of the logic resources of Carry Select Adder and significantly contributes. For efficient implementation of the Sum and Carry Generation unit different logic designs have been suggested. Based Carry Select Adders of by suitable logic expressions for the SCG unit of conventional and binary excess-1 converters (BEC) we made a study of the logic designs suggested. To identify redundant logic

operations and data dependence for the main objective of this study. Accordingly, based on their data dependence we remove all redundant logic operations and sequence logic operations.

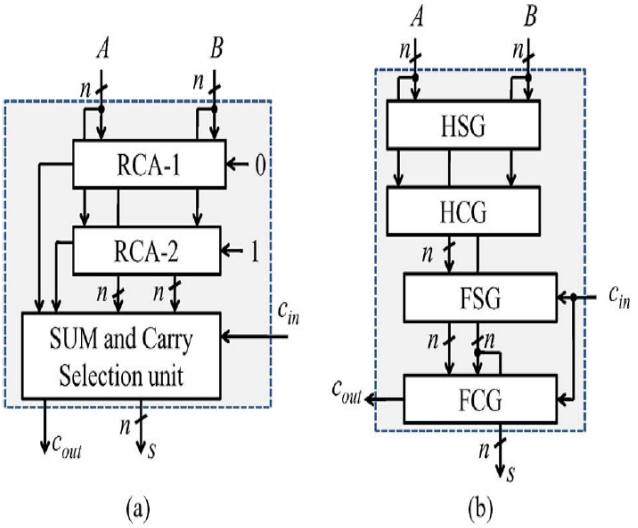

Fig. 1. (a) Conventional Carry Select Adder (b) The logic operations of the RCA is shown in split form.

As shown in Fig. 1(a), the SCG unit of the conventional Carry Select Adder is composed of two n-bit RCAs that is ripple carry adders , where HSG, HCG, FSG, and FCG represent half-sum generation, half-carry generation, full-sum generation, and full-carry generation, respectively. Where n is the adder bit-width. The logic operation of the n-bit RCA is performed in four stages. Those are specified as below.

- half-sum generation which is specified as HSG
- half-carry generation which is specified HCG
- full-sum generation which is specified as FSG and
- full carry generation which is specified as FCG.

For example if two n-bit operands are added in the conventional CSLA, then RCA-1 and RCA-2 generate n-bit sum defined as s0 and s1 and output-carry which are as defined as $c_{out}^0$and c_{out}^1 are the corresponding to carry input as c_{in}= 0 and c_{in} = 1, respectively. Whereas the logic expressions of the RCA of 1 and the RCA of 2 of the SCG unit of the n-bit CSLA are given as

$$s_0^0(i) = A(i) \oplus B(i)\ c_0^0(i) = A(i) \cdot B(i) \quad \textbf{(1a)}$$

$$s_1^0(i) = s_0^0(i) \oplus c_1^0(i-1) \quad \textbf{(1b)}$$

$$c_1^0(i) = c_0^0(i) + s_0^0(i) \cdot c_1^0(i-1)\ c_{out}^0 = c1^0(n-1) \quad \textbf{(1c)}$$

$$s_0^1(i) = A(i) \oplus B(i)\ c_0^1(i) = A(i) \cdot B(i) \quad \textbf{(2a)}$$

$$s_1^1(i) = s_0^1(i) \oplus c_1^1(i-1) \quad \textbf{(2b)}$$

$$c_1^1(i) = c_0^1(i) + s_0^1(i) \cdot c_1^1(i-1)\ c_{out}^1 = c_1^1(n-1) \quad \textbf{(2c)}$$

As shown in Fig. 2, the RCA calculates n-bit sum $s_1^0$and c_{out}^0 corresponding to c_{in} = 0. The BEC unit receives s_1^0 and c_{out}^0 from the RCA and generates (n + 1)-bit excess-1 code. The most significant bit (MSB) of BEC represents c_{out}^1, in which n least significant bits (LSBs) represent s_1^1 in the logic expressions. The structure of the BEC based CSLA is as shown in Fig. 2.

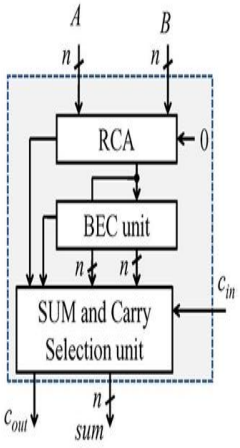

Fig.2. Structure of the BEC based CSLA; n is the input operand bit-width of the RCA are the same as those given in 1(a)- 1(c).

The logic expression of the BEC unit of the n-bit BEC based CSLA are given as

$$s_1^1(0) = s_1^0(0)\ c_1^1(0) = s_1^0(0) \quad \textbf{(3a)}$$

$$s_1^1(i) = s_1^0(i) \oplus c_1^1(i-1) \quad \textbf{(3b)}$$

$$c_1^1(i) = s_1^0(i) \cdot c_1^1(i-1) \quad \textbf{(3c)}$$

$$c_{out}^1 = c_1^0(n-1) \oplus c_1^1(n-1) \quad \textbf{(3d)}$$

for $1 \le i \le n-1$. We can find from (1a)–(1c) and (3a)–(3d) that, in the case of the BEC-based CSLA, c_1^1 depends on s_1^0, which otherwise has no dependence on $s1^0$ in the case of the conventional CSLA. The BEC method therefore increases data dependence in the CSLA. We have considered logic expressions of the conventional CSLA and made a further study on the data dependence to find an optimized logic expression for the CSLA.

III. PROPOSED ADDER DESIGN

The proposed self-timed carry select adder is based on the logic formulation given in (4a)–(4g). It is having one full sum generation (FSG) unit, one half sum generation (HSG) unit, one Carry Generation unit, and one Carry Sum unit. The carry generation unit is composed of two CGs (CG_0 and CG_1) corresponding to input-carry '0' and '1'. The half sum generation takes two n-bit operands to generate half-sum word s0 of width n-bit and half-carry word c0 of width n bits. Both CG_0 and CG_1 receive s_0 and c0 from the HSG unit and generate two n-bit full-carry words $c_1^0$and c_1^1 corresponding to input-carry '0' and '1', respectively. The CS unit selects one final carry word from the two carry words available at its input line using the control signal c_{in}. It selects c0 1 when c_{in} = 0; otherwise, it selects c_1^1. The CS unit can be implemented using an n-bit 2-to-l MUX. However, we find from the truth table of the CS unit that carry words c_1^0 and c_1^1 follow a specific bit pattern. If $c_1^0(i)$ = '1', then $c_1^1(i)$=1, irrespective of $s_0(i)$ and

$c_0(i)$, for $0 \leq i \leq n-1$. This feature is used for logic optimization of the CS unit. The final carry word c is obtained from the CS unit. The MSB of c is sent to output as c_{out}, and (n − 1) LSBs are XORed with (n − 1) most significant bit(MSB) of half-sum (s_0) in the full sum generation(FSG) to obtain (n − 1) MSBs of final-sum (s). The least significant bit(LSB) of s0 is XOR with c_{in} to obtain the LSB of s.

The major speed limitation in any adder is in the production of carries and many authors have considered the addition problem. This logic can be implemented with Carry Select Adder using D-Latch to achieve power-efficient and high-speed data path logic systems. The proposed 64-bit Carry Select Adder compared with the Carry Select Adder (CSLA) and Regular 64-bit Carry Select Adder and also with BEC 64-bit Carry Select Adder. How the goal of fast addition is achieved using BEC together with a multiplexer (mux) is described in Fig.2, one input of the 8:4 mux gets as it input (B3, B2, B1, and B0) and another input of the Mux is the BEC output. This produces the two possible partial product results in parallel and the Muxes are used to select either BEC output or the direct inputs according to the control signal C_{in}. The Boolean expressions of 4-bit BEC are listed below, (Note: functional symbols, ~ NOT, & AND, ^ XOR).

$$X0 = \sim B0$$

$$X1 = B0 \wedge B1$$

$$X2 = B2 \wedge (B0 \,\&\, B1)$$

$$X3 = B3 \wedge (B0 \,\&\, B1 \,\&\, B2)$$

The AND, OR, and Inverter (AOI) implementation of an2:1 MUX, FA are shown in below. The gates between the dotted lines are performing the operations in parallel and the numeric representation of each gate indicates how much delay is provided. AND, OR, and Inverter are the basic gates for designing any digital circuits so the these are consider for the delay and area , those are having equal delay and area of 1-unit. If we add more number of gates through longest path it gives maximum delay. The counting of AOI Gates a required for each logic block to decide the area evalution. Based on this approach, the CSLA adder blocks of 2:1 mux, Half Adder (HA), and FA are evaluated as shown in Fig.3.

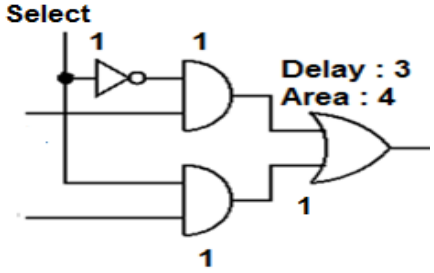

Fig.3. Delay and Area evaluation of an 2:1 Mux gate.

The bottom-line is that clock management is becoming increasingly difficult and solving it in today's high-speed complex system-on-chip designs appears to be a complex and costly affair. The second major problem faced by designers is power dissipation, which is a very important metric that has gained significance with the phenomenal growth of portable electronics. For mobile electronic applications, the average power consumption has become the most critical design concern. For maximum efficiency, all gates in the system should be performing useful work. However in synchronous systems, this is not usually the case. Consequently, synchronous systems tend to consume more power than necessary. Many gates switch unnecessarily since they are connected to the clock and not because they have to process new input data The recent demonstration of the potential advantages of the world's first 8-bit physically flexible asynchronous microprocessor design over a synchronous flexible version in terms of power and noise by Karachi et al. from Seiko Epson's Technology Platform Research Centre ,phase handshaking and quasi-delay-insensitive design style, endorses the future of self timed design techniques for even unconventional electronics as shown in Fig.4.

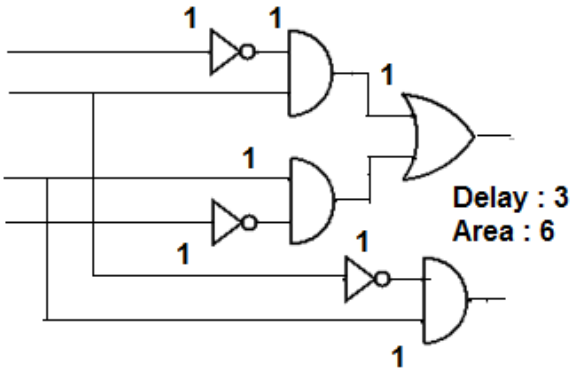

Fig.4. Delay and Area evaluation of a Full Adder.

Asynchronous circuits assume that signals are binary but the notion that time is not discrete. An asynchronous system is one in which there is no global synchronization within the system; subsystems within the system are synchronized locally by the communication protocols between them. The results produced by the subsystems in an asynchronous system can be consumed by other subsystems as soon as they are generated without having to wait for a global clock tick. Moreover in asynchronous systems, a sub-system can easily be replaced by another subsystem with the same functionality but with different performance, but this is not a straightforward task in case of a synchronous system as the clock period might have to be recomputed. An asynchronous system stage that involves request/acknowledge handshake (signal exchange) signaling protocol.

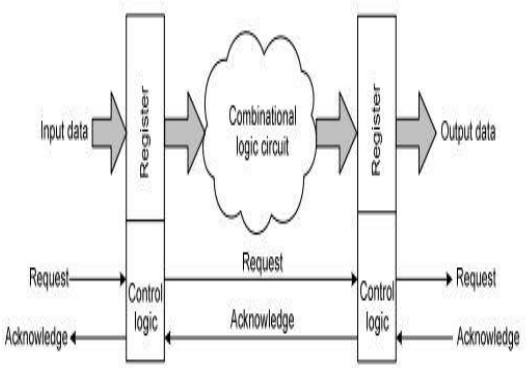

Fig.5. Register Information flow.

However, robust asynchronous systems embed the request information within the data wires and are usually referred to as self-timed systems. Self-timed systems are characterized by the absence of any timing reference to which all the operations are synchronized – being in stark contrast to synchronous systems where all operations are synchronized to the global clock signal. Binary addition is the single most important operation that a processor performs as shown in Fig.5. Most of the adders have been designed for synchronous circuits even though there is a strong interest in clock less asynchronous processors/circuits. Asynchronous circuits do not assume any quantization of time. Therefore, they hold great potential for logic design as they are free from several problems of clocked (synchronous) circuits. In principle, logic flow in asynchronous circuits is controlled by a request-acknowledgment handshaking protocol to establish a pipeline in the absence of clocks. Explicit handshaking blocks for small elements, such as bit adders, are expensive. Therefore, it is implicitly and efficiently managed using dual-rail carry propagation in adders. A valid dual-rail carry output also provides acknowledgment from a single-bit adder block. Thus, asynchronous adders are either based on full dual-rail encoding of all signals (more formally using null convention logic that uses symbolically correct logic instead of Boolean logic) or pipelined operation using single-rail data encoding and dual-rail carry representation for acknowledgments. While these constructs add robustness to circuit designs, they also introduce significant overhead to the average case performance benefits of asynchronous adders. Therefore, a more efficient alternative approach is worthy of consideration that can address these problems.

This brief presents an asynchronous parallel self-timed adder (PASTA) using the algorithm originally proposed in. The design of PASTA is regular and uses half-adders (HAs) along with multiplexers requiring minimal interconnections. Thus, it is suitable for VLSI implementation. The design works in a parallel manner for independent carry chain blocks. The implementation in this brief is unique as it employs feedback through XOR logic gates to constitute a single-rail cyclic asynchronous sequential adder. Cyclic circuits can be more resource efficient than their acyclic counterparts. On the other hand, wave pipelining (or maximal rate pipelining) is a technique that can apply pipelined inputs before the outputs are stabilized. The proposed circuit manages automatic single-rail pipelining of the carry inputs separated by propagation and inertial delays of the gates in the circuit path. Thus, it is effectively a single-rail wave-pipelined approach and quite different from conventional pipelined adders using dual-rail encoding to implicitly represent the pipelining of carry signals. The remainder of this brief is organized as follows. There are a myriad designs of binary adders and we focus here on asynchronous self-timed adders. Self-timed refers to logic circuits that depend on and/or engineer timing assumptions for the correct operation. Self-timed adders have the potential to run faster averaged for dynamic data, as early completion sensing can avoid the need for the worst case bundled delay mechanism of synchronous circuits. They can be further classified as follows.

A. Pipelined Adders Using Single-Rail Data Encoding

The asynchronous Req/Ack handshake can be used to enable the adder block as well as to establish the flow of carry signals. In most of the cases, a dual-rail carry convention is used for internal bitwise flow of carry outputs. These dual-rail signals can represent more than two logic values (invalid, 0, 1), and therefore can be used to generate bit-level acknowledgment when a bit operation is completed. Final completion is sensed when all bit Ack signals are received (high). The carry-completion sensing adder is an example of a pipelined adder, which uses full adder (FA) functional blocks adapted for dual-rail carry as shown in Fig.6. On the other hand, a speculative completion adder is proposed in. It uses so-called abort logic and early completion to select the proper completion response from a number of fixed delay lines. However, the abort logic implementation is expensive due to high fan-in requirements.

B. Delay Insensitive Adders Using Dual-Rail Encoding

Delay insensitive (DI) adders are asynchronous adders that assert bundling constraints or DI operations. Therefore, they can correctly operate in presence of bounded but unknown gate and wire delays. There are many variants of DI adders, such as DI ripple carry adder (DIRCA) and DI carry look-ahead adder (DICLA). DI adders use dual-rail encoding and are assumed to increase complexity as shown in Fig.7. Though dual-rail encoding doubles the wire complexity, they can still be used to produce circuits nearly as efficient as that of the single-rail variants using dynamic logic or n-MOS only designs. Similar to CLA, the DICLA defines carry propagate, generate, and kill equations in terms of dual-rail encoding. They do not connect the carry signals in a chain but rather organize them in a hierarchical tree. Thus, they can potentially operate faster when there is long carry chain. A further optimization is provided from the observation that dual-rail encoding logic can benefit from settling of either the 0 or 1 path. Dual-rail logic need not wait for both paths to be evaluated. Thus, it is possible to further speed up the carry look-ahead circuitry to send carry-generate/carry-kill signals to any level in the tree. This is elaborated in [8] and referred as DICLA with speedup circuitry (DICLASP).

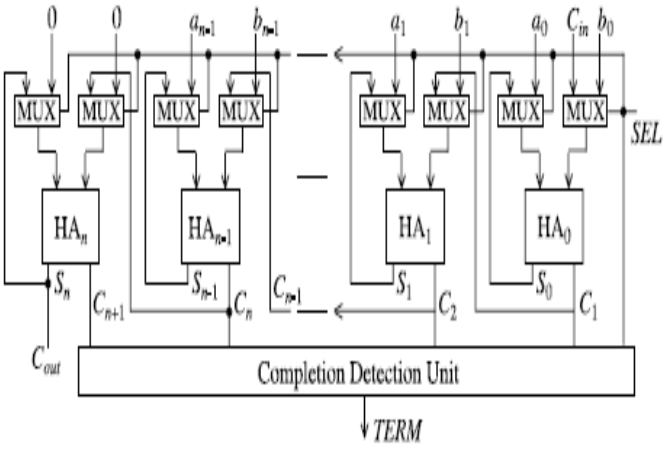

Fig. 6. General block diagram of PASTA.

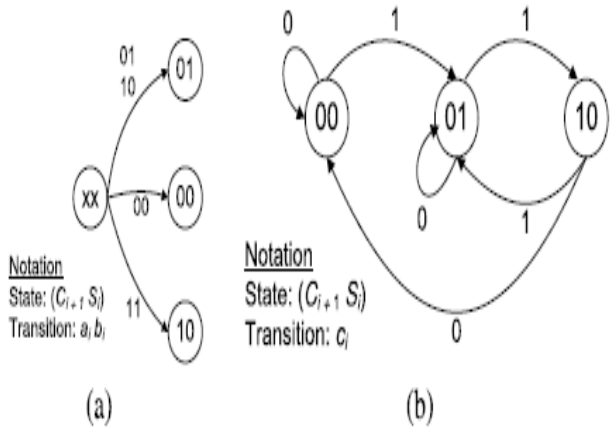

Fig. 7. State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

IV. CSLA USING D-LATCH LOGIC

This method replaces the BEC circuit by D-latch. Latches are used to store 1-bit binary information. The latch is one of the sequential circuits so Their outputs are depends on the present inputs and previous inputs.

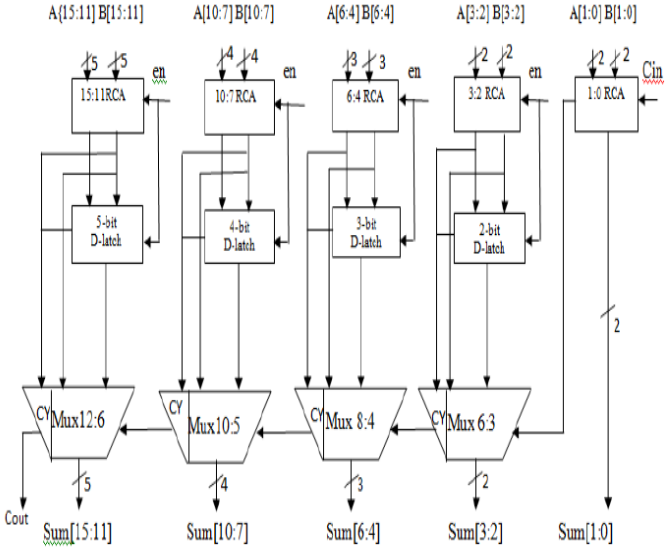

Fig.8. 16-Bit CSLA with D-Latch Architecture.

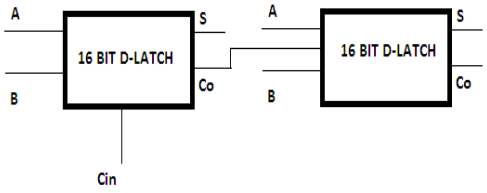

Fig.9. 32-Bit CSLA with D-Latch Architecture.

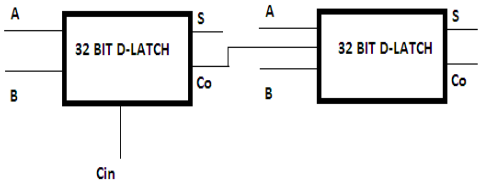

Fig.10.64-Bit CSLA with D-Latch Architecture.

In other words, the latch is level sensitive, so when latch are enabled, the operation of latch is changes according with input signal of the latch. The architecture of proposed 16-bit Carry Select Adder is shown in Fig. 8. In this we are using five different ripple carry adders with different bit size and D-Latch. In this proposed method uses only one adder Instead of using two separate adders in the regular carry select adder(CSLA) to reduce the area, and power consumption. In CSLA Each of the two additions is performed in one clock cycle. In the 16-bit adder Ripple carry adder used in the least significant bit (LSB) which is 2 bit wide. The upper half of the adder is most significant part is 14-bit wide the upper of the adder is works depends on the clock, the input carry performed addition when carry goes high. The carry input is assumed as zero While clock goes low and the sum of adder is stored in adder itself. From the Fig.8 it can understand that latch is used to store the sum and carry for C_{in}=1 and c_{in}=0.Carry out from the previous stage i.e., least significant bit adder is used as control signal for multiplexer to select final output carry and sum of the 16-bit adder. If the actual carry input is one, then computed sum and carry latch is accessed and for carry input zero MSB adder is accessed is the output carry. The architectures of the 16-Bit, 32-Bit and 64-Bit CSLA with D-LATCH are shown in fig 8, fig 9 and 10. The 64-Bit CSLA with D-LATCH Architecture is designed based on the cascading of two 32-Bit Architectures. In the D-LATCH architecture first stage is designed based on Ripple Carry adders and the second stage is designed based on the D-LATCH logic.

A. Working Principle of CSLA Adder Using D Latch

The bits from a and b (i,e a[1:0] and b[1:0]) as inputs to (1:0)RCA along with c_{in} as input. The bits from a and b (i,e a[3:2] and b[3:2]) as inputs to (3:2)RCA along with en. When en=1, the output of the(3:2)RCA is fed as input to the (2 bit) D-Latch and the output of the (2 bit) D-latch follows the input and given as an input to the (6:3)multiplexer. When en=0, the last state of the (2 bit) D input is trapped and held in the latch and therefore the output from the (3:2)RCA is directly given as an input to the (6:3) multiplexer without any delay. Now the (6:3) multiplexer selects the sum bit according to the carry generated from (1:0) RCA which takes c_{in} as input carry and it is the selection bit and the inputs of the (6:3) multiplexer are the outputs obtained when en=1 and 0. The bits from a and b (i,e. a[6:4] and b[6:4]) are given as inputs to (6:4)RCA along with en as carry. When en=1, the output of the(6:4)RCA is fed as input to the (3 bit) D-Latch and the output of the (3 bit) D-latch follows the input and given as an input to the (8:4)multiplexer. When en=0, the last state of the (3 bit) D input is trapped and held in the latch and therefore the output from the (6:4)RCA is directly given as an input to the (8:4) multiplexer without any delay. Now the (8:4) multiplexer selects the sum bit according to the carry generated from (6:3) multiplexer which is the selection bit and the inputs of the (8:4) multiplexer are the outputs obtained when en=1 and 0. The bits from a and b (i,e a[10:7] and b[10:7]) are given as inputs to (10:7)RCA along with en as carry. When en=1, the output of the(10:7)RCA is fed as input to the (4 bit) D-Latch and the output of the (4 bit) D-latch follows the input and given as an input to the (10:5)multiplexer. When en=0, the last state of the (4 bit) D

input is trapped and held in the latch and therefore the output from the (10:7)RCA is directly given as an input to the (10:5) multiplexer without any delay. Now the (10:5) multiplexer selects the sum bit according to the carry generated from (8:4) multiplexer which is the selection bit and the inputs of the (10:5) multiplexer are the outputs obtained when en=1 and 0. The bits from a and b (i,e a[15:11] and b[15:11]) are given as inputs to (15:11)RCA along with en as carry. When en=1, the output of the(15:11)RCA is fed as input to the (5 bit) D-Latch and the output of the (5 bit) D-latch follows the input and given as an input to the (12:6)multiplexer. When en=0, the last state of the (5 bit) D input is trapped and held in the latch and therefore the output from the (15:11)RCA is directly given as an input to the (12:6) multiplexer without any delay. Now the (12:6) multiplexer selects the sum bit according to the carry generated from (10:5) multiplexer which is the selection bit and the inputs of the (12:6) multiplexer are the outputs obtained when en=1 and 0.

V. CONCLUSION

Self-timed adder is the most common and often used arithmetic operation on microprocessor, digital signal processor, especially digital computers. Also, it serves as a building block for synthesis all other arithmetic operations. Therefore, regarding the efficient implementation of an arithmetic logic unit, the adder structures become a very critical hardware unit. A D-LATCH based CSLA architecture is proposed in this project to reduce the delay of CSLA architecture than the recently proposed BEC based CSLA architecture. The functionality verification of the design is carried out by using ISE Simulator and the synthesis is also carried out by the XILINX ISE 12.3i.The HDL used for obtaining an RTL schematic and for designing the modules is VERILOG. From the graphs and the tables it is concluded that, the proposed D-LATCH based design is having less delay when compare to the BEC based and RCA based architectures. With the increase in silicon densities, it is becoming feasible for compression systems to be implemented in a single chip. Here we have implemented CSLA using D-latch approach with less Delay. In future, we further reduce Area and Power parameters without the penalty of resources.

VI. REFERENCES

[1] B. Ramkumar and Harish M Kittur, "Low Power and Area Efficient Carry Select Adder" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS-2011.

[2] Bedrij, O. J., (1962), "Carry-select adder," IRE Trans. Electron. Comput., pp.340–344 .

[3] Ceiang ,T. Y. and Hsiao,M. J. ,(Oct. 1998),"Carry-select adder using single ripple carry adder," Electron. Lett., vol. 34, no. 22, pp. 2101– 2103.

[4] Ramkumar,B. , Kittur, H.M. and Kannan ,P. M. ,(2010),"ASIC implementation of modified faster carry save adder," Eur. J. Sci. Res., vol. 42, no. 1,pp.53–58,2010.

[5] J. M. Rabaey, Digtal Integrated Circuits—A Design Perspective. Upper Saddle River, NJ: Prentice-Hall, 2001.

[6] E. Abu-Shama and M. Bayoumi, "A New cell for low power adders," in Proc.Int. Midwest Symp. Circuits and Systems, 1995, pp. 1014–1017.

[7] Y. Kim and L.-S. Kim, "64-bit carry-select adder with reduced area,"Electron. Lett., vol. 37, no. 10, pp. 614–615, May 2001.

[8] Y. He, C. H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," in Proc. IEEE Int. Symp. Circuits Syst., 2005, vol 4, pp.4082-4085.