Design of High Speed 32-Bit Data Processing using CSLA
Anshu Kumari¹, Kamal Niwaria²

ABSTRACT
Modern applications demand extremely low power and fast speed in computer architectures for battery-operated devices like Laptop and others. In this work, the main focus is on the low power consumption and provides high speed to the processors. Low-power and high speed circuits are becoming more desirable due to growing portable device markets and they are also becoming more applicable today in processors. The selection behind the carry select adder is that it is very much efficient in terms of delay. The main focus in this work is to improve the speed of the 32-bit processor and in this case the carry select adder is the better choice. The approach used here is to implement these pipelines in a manner that only one pipeline will be activated through which the carry is propagating.

Keywords--- CSLA, 32-bit, Speed data

I. INTRODUCTION
The design of high speed, low power and as well as minimum area adder architecture has been the main concern of many Very Large Scale Integration (VLSI) researchers and this resulted in a large number of adder architectures. These various available architectures provide the capacity to obtain the gain more accurate and deep instinctive understanding of adder and thus suggest various implementations. Here we are going to discuss about the requirement of adder in processors and will also discuss about different adder architectures. This Design uses a simple and efficient gate-level modification to significantly reduce the area and power of the Carry Select Adder (CSLA). Based on this modification 16-, 32, and 64-bit Square-Root Carry Select Adder (SQRT CSLA) architecture have been developed and compared with the regular SQRT CSLA architecture. The proposed Work has reduced area, power and delay as compared with carry propagation delay by independently generating multiple carries and then select a carry to generate the sum. In this thesis, the main focus is on the low power consumption and provides high speed to the processors. In processors the main concern is an adder which is required in ALU to perform arithmetic operations as well as to decode and fetch addresses in the memory and from the memory.

A pipelining technique is using here to reduce the power consumption and provide high speed of data processing. The carry select adder comes under the category of conditional sum adder as it works over conditional statements. Conditional sum adder means which works on some condition. In this sum and carry are calculated by assuming input carry as 1 and 0 prior the input carry comes. When actual carry input arrives in system, the actual calculated values of sum and carry are selected using multiplexer.

Background
Now a day’s design of area and speed in efficient high speed data rate is commonly used research area in VLSI design. The demand of high data rate and less energy required with less area and high speed VLSI device is growing rapidly. Carry Select Adder is one of the fastest adders which are using in many processing unit. It is found that CSLA is one of the fastest adders to perform the arithmetic functions with high speed. From the structure of the carry select adder, it has been observed that there is scope for reducing the area requirement, power consumption and as well as delay in the CSLA.

Role of Adder in Data Processing Processors
Adders often appear in the arithmetic logic unit, integer execution unit and sometimes in the address generation path. The requirement of more optimum adder designs for a modern microprocessor initiated in this research. If a floating-point unit is present they appear in the significant adder, at the base of multiplier array, and in the divider. Smaller adders also appear in the exponent manipulation circuitry for multiplication and division. The identification of an appropriate adder generator is a highly influence tool for creating an efficient design. Comparators and incrementers are also forms of adders, and they appear in various places. The design requirements for adders are vary for different applications. For example in some cases it is necessary for the execution of unit, adder to be very flexible. The Intel execution unit adder was designed to
naturally produce carries on 8, 16, or 32 bit boundaries since these are the native data types \[12\] for that architecture. The x86(processors) architecture requires a four operand adder for address generation. Often instruction used to generate the address show addition to be the most common arithmetic instruction.

However, there are many stages are involved in the execution of any instruction, therefore the importance of the evaluation time of the adder circuit, which dominates only the execution stage, is not always obvious\[3, 4\].

In all but the least highly complicated processors, instructions are pipelined through fetch, decode, and execution stages. In modern processors which are available today there is also fine grain parallelism in the form of multiple execution units which run in parallel. Hence, the focus of the processor is to keep the execute units busy doing useful work, and to the extent this is possible, the speed of the execution units is important. An efficient operating system that allocates sufficient resources instruction level parallelism and locality in the code a large enough decode window to take advantage of the parallelism large enough caches to take advantage of locality successful branch prediction to decrease the penalty of branches low penalty on misprediction of branches register renaming to remove false dependencies out of order issue and execution so that installed instructions do not stop the pipe. Result forwarding to reduce latency in dependent sequences an instruction set (architecture) that facilitates the above goals. These items place a large burden on the operating system, the compiler, the memory system, and the decoder. When this burden is met – performance may be strongly determined by adder latency. That is to say, when instructions can be provided at a maximum rate with maximum parallelism, execution speed is determined soley by serially dependent instruction steps for which add instructions are common.

**Carry Select Adder**

The carry-select adder can be simply define as a combination of two Ripple Carry Adders (RCA), one to generate the sum for carry input \(C_{in}=1\) and the other to generate the sum for carry input \(C_{in}=0\) and a multiplexer stage is connected to choose the correct sum output in which the selection line is taken as the actual carry propagation. Adding two n-bit data with a carry-select adder is done with two ripple carry adders in order to perform the calculation twice, one time with the assumption of the carry being zero and the other assuming one. After that the two results are calculated, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known.

![Fig.1.1 Carry Select Adder][36]
operations inside a circuit compromised of such blocks. The performance of a digital circuit block is gauged by analyzing its power dissipation, layout area and its operating speed. The implementation techniques of several types of carry select adders and study their characteristics and performance are given below:

**Conventional Square-root Carry Select Adder (CSCSA)**

The conventional square-root carry select adder is designed by equalizing the delay through two carry chains. CSLA uses multiple pairs of ripple carry adder to generate partial sum and carry by considering carry input \( C_{in}=0 \) and \( C_{in}=1 \), then the final sum and carry are selected by multiplexers. The conventional carry select adder is shown in fig 1.2.

![Fig. 1.2 Conventional SQRT Carry Select Adder][5]

**Modified Square-root Carry Select Adder (MSCSA)**

The structure of the 16-bit SQRT CSLA using BEC for RCA with \( C_{in} = 1 \) to optimize the area and power. The structure is split into five stages in which they compute the sum of 16-bits.

As stated above the main idea of modified SQRT CSLA is to use BEC instead of the RCA with \( C_{in}=1 \) in order to reduce the area and power consumption of the regular CSLA. To replace the n-bit RCA, an \( n+1 \) bit BEC is required. The basic function of the CSLA is obtained by using the 4-bit BEC together with the mux. One input of the 8:4 mux get as input \( (B_3, B_2, B_1, \text{ and } B_0) \) and another input of the mux is the BEC output. This produces the two possible partial results in parallel and the mux is used to select either the BEC output or the direct inputs according to the control signal \( C_{in} \). The importance of the BEC logic stems from the large silicon area reduction when the CSLA with large number of bits are designed. The modified square root carry select adder is shown in fig 1.3.

![Fig. 1.3 Modified SQRT CSLA][5]

**II. LITERATURE REVIEW**

R.Priya and J.Senthil Kumar “Enhanced Area Efficient Architecture for 128-bit Modified CSLA”, 2013 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2013] 978-1-4673-4922-2113©2013 IEEE Page(s): 989-992, the basic idea discuss here is for reducing the area, the CSLA can be implemented by using a single RCA and an add-one circuit instead of using dual RCA. It proposed a simple methodology to diminish the area of CSLA architecture. The reduced number of gates of this work offers the great improvement in the reduction of area. Totally from the result analysis the 128-bit Modified SQRT CSLA has less area as compared with Regular Linear CSLA, Regular SQRT CSLA and Modified Linear CSLA. The area of the proposed design shows a decrease for 16-bit, 32-bit, 64-bit and 128-bit sizes which show the success of the method and not a simple tradeoff of delay for area. The Modified CSLA architecture is therefore, simple, low area, and efficient for VLSI hardware implementation. This work has been developed using Verilog-Hardware Description Language (VHDL). It was simulated using Modelsim Altera 10.0c and synthesized using Xilinx PlanAhead 13.4. This design was further implemented in Virtex5 kit. The results given in this paper shows that the Modified Linear CSLA and Modified SQRT CSLA provide better outcomes than the Regular Linear CSLA and Regular SQRT CSLA respectively[1].

Senthilkumar.A and Kousalya devi.A, “VLSI Implementation of an Efficient Carry Select Adder Architecture”, International Journal Of Advance Research In Science And Engineering IJARSE, Vol. No.2, Issue No.4, April, 2013 ISSN-2319-8354(E), the idea provided here is to use modified carry select adder one RCA \( C_{in}=1 \) is replaced by BEC. BEC design consists of AND, XOR and NOT gates as its structure. In this structure the XOR gate will be replaced by MUX with NOT gate. The least
significant bit of the input is given to NOT gate and it is given as control signal to the MUX for the next input value. The LSB and the next immediate bit is provided as input to the AND gate where its corresponding output value is given as a control signal to the Multiplexer. Depending upon the control signal it will produce the sum value. If it is 0, then its output is same as the input otherwise it produces its complement value. The operation of XOR is same as that of the MUX with NOT gate. The proposed CSLA design reduces the area and power by replacing the gates in BEC design. The proposed CSLA consumes less power and area compared to the modified CSLA. The adders are designed using Very High Speed Integration Hardware Description Language (VHDL), Xilinx Project Navigator 8.1i is used as a synthesis tool and ModelSim Integrated System Environment (ISE) III 6.2C for simulation. Thus the result analysis shows that the proposed CSLA is better compared to the conventional CSLA. [2]

L. Mugilvannan and S. Ramasamy, “Low-Power And Area-Efficient Carry Select Adder Using Modified Bec-1 Converter”, International Journal of Computer Applications in Engineering Sciences Special Issue on National Conference on Information and Communication (NCIC’13) Volume III, Special Issue, August 2013 ISSN: 2231-4946, the structure of the proposed 16-b SQRT CSLA using transistor level modified BEC for RCA with cin=1 to optimize the area and power. They again split the structure into five groups. It has transistor level design of 3-b BEC and 6:3 Mux. The Transistor level modified 3-b BEC is used in CSLA for reducing the power and area of the adder. The total number of P-type Metal Oxide Semiconductor (PMOS) and N-type Metal Oxide Semiconductor transistor (NMOS) in the 3-b BEC and MUX are 34 and 34 Similarly the transistor level modified 4-b BEC and 8:4 Mux, 5-b BEC and 10:5 Mux ,6-b BEC and 12:6 Mux has been designed. [5]

B. Ramkumar and Harish M Kittur “Low-Power And Area-Efficient Carry Select Adder”, IEEE transactions on Very Large Scale Integration (VLSI) systems, VOL. 20, NO. 2, FEBRUARY 2012, the approach described here is to reduce the area and power consumption of SQRT CSLA architecture by minimizing the number of gates required to implement the whole architecture. In this work the authors utilizes the BEC instead of RCA for propagation of carry-1 and this helps in reducing the area to a greater extent. Thus the modified CSLA proves to be greatly efficient in terms of area and power requirement but this modified CSLA has to compromise over speed. The design proposed in this paper has been developed using Verilog-HDL and synthesized in Cadence RTL compiler using typical libraries of TSMC 0.18 um technology. The synthesized Verilog netlist and their respective design constraints file (SDC) are imported to Cadence SoC Encounter and are used to generate automated layout from standard cells and placement and routing [7].

T. Ratna Mala, R. Vinay Kumar, T. Chandra Kala, “Design and Verification of Area Efficient High-Speed Carry Select Adder”, T. Ratna Mala, et al, International Journal of Research in Computer and Communication technology, IJRCCCT, ISSN 2278-5841, Vol 1, Issue 6, November 2012, A simple approach is proposed to reduce the area of SQRT CSLA architecture. The reduced number of gates of this work offers the great advantage in the reduction of area. The compared results show that the modified SQRT CSLA has a delay (only 3.76%), but the area of the 128-bit modified SQRT CSLA are significantly. The area-delay product of the proposed design show a decrease for 16 and 128-bit sizes which indicates the success of the method and not a mere trade-off of delay for power and area. The modified CSLA architecture is therefore, low area, simple and efficient for VLSI hardware implementation.

Problem formulations

The basic problem which has been analyzed is flow of data rate as in this thesis, working is on high data rate and compact size. The other problem which occurs during the implementation of this project is in a part of energy consumption because the concept has been implemented here is to reduce the power consumption in such a way that the carry propagation takes place through only one pipeline whereas others will remain quite. In this way that power can be save which is waisting while all the pipeline architectures were activated all the time. Thus, the requirement of power can be reduced. But the problem is how to propagate the carry in such a way and generate the partial sum. To rectify this problem the concept of leading edge and trailing edge has been implemented. This concept enable the particular pipeline when the leading edge triggers while disable the pipeline when trailing edge occurs.

The other problem is how to reduce the area requirement because here dual ripple carry architecture has been using which acquires more area. This problem is overcome by implementing the carry propagation in a non-linear fashion, so that number of stages has been reduces. Thus, the proposed 32-bit CSLA designed here reduces energy consumption and provide high data rate. It also reduces area of carry select adder.

Proposed Methodology

The methodology has been implemented in this research work is utilizing the basic concept of the CSLA using the dual RCA. The simple design flow has been shown in fig.4.1. But the simple concept is implementing here in a different manner. Here the sum is generating linearly but the carry is propagating non-linearly.

Design Flow of CSLA

The basic design flow of the data is shown here. It is providing us the information that how this design has been working. So, the first block is representing the setup block which is signify the initial setup time require to initiate the process. If the block consists of a 4 full adder
then that block will add the bits from k to k+3. Then the 0-
carry propagation block and 1-carry propagation blocks
will provide the SUM-0 and SUM-1 respectively. Then the
simple MUX stage is provided to select the correct sum.
Here the selection operation is much faster than time to
compute either of the two possible vectors. Implementation of each of these blocks has been done
through HDL level optimization to obtain better
performance versus efficiency ratio.

![Basic Block Diagram of Proposed CSLA](image)

Basic Block Diagram of Proposed CSLA

In the proposed CSLA, each of these sections is
composed of two 4-bits ripple-carry adders. This is
referred as linear expansion. The delay of n-bit carry select
adder based on m-bit CLA blocks can be given by the
following equation when using constant carry number
blocks. The Basic block diagram of CSLA using 4-bit RCA
is shown in fig.4.2.

\[ T = t_{setup} + m \cdot t_{carry} + \frac{n}{m} \cdot t_{mux} + t_{sum} \]

And by the following equation when using successively
incremented carry number blocks respectively.

\[ T = t_{setup} + m \cdot t_{carry} + (2n)^{1/2} \cdot t_{mux} + t_{sum} \]

4-bit Linear CSLA

Carry select Adder is a better choice especially in
the case of Carry delay. As in a ripple-carry adder, every
full adder cell has to wait for the incoming carry before an
outgoing carry can be generated. This dependency can be
eliminated by pre-calculating i.e. by taking both possible
values of the carry input and evaluating the result for both
possibilities in advance. Once the real value of the
incoming carry is known, the correct result is easily
selected with a simple multiplexer stage. The
implementation of this idea is called the linear carry select
adder and the diagram of the first four bits of the adder is
shown in fig 4.3.

![Fig.4.2 Basic block diagram of CSLA using 4-bit RCA](image)

Fig.4.2 Basic block diagram of CSLA using 4-bit RCA

4-bit Linear CSLA

Carry select Adder is a better choice especially in
the case of Carry delay. As in a ripple-carry adder, every
full adder cell has to wait for the incoming carry before an
outgoing carry can be generated. This dependency can be
eliminated by pre-calculating i.e. by taking both possible
values of the carry input and evaluating the result for both
possibilities in advance. Once the real value of the
incoming carry is known, the correct result is easily
selected with a simple multiplexer stage. The
implementation of this idea is called the linear carry select
adder and the diagram of the first four bits of the adder is
shown in fig 4.3.

![Fig.4.3 Diagram of first 4-bit linear CSLA](image)

Fig.4.3 Diagram of first 4-bit linear CSLA

In the above diagram all the inputs are given at a
time to both the Ci =0 and the Ci =1 carry logic. The carry
circuits generate the appropriate carryouts and depending
upon the original carry input the appropriate values are
selected from the multiplexer and fed to the sum circuit.
This is the basic 4-bit carry select adder. Thus for
implementing the higher order bit the carry out from the fifth stage multiplexer passes as the carry in for the next 4-bit, while the inputs are given at the same time. It is clear that the delay is reduced to a large extent by performing the carry calculations before hand, but the disadvantage is that the hardware over head of the carry select adder is restricted to an additional carry path and a multiplexer.

III. IMPLEMENTATION OF NON-LINEAR CARRY PROPAGATION

The actual modification which has been made in this proposed CSLA is to use a method which gives more optimum results is to apportion the adder non-linearly. Here, as the computation of 32-bits has been implementing. Thus, to minimize the number of stages require for the computation can reduced only and only if the variable size ripple carry adders are implement. Fig 4.4 shows 32-bit carry select adder design.

For example to design a 32-bit Carry-Select Adder one can use 6 stages of adders with sizes: 4, 4, 5, 6, 7, 6 = 32 bits. Each stage computes a partial sum; Ripple adders can be used for stage adders.

Internal Structure of Non-linear CSLA

Here the internal structure is shown of 32-bit CSLA is given which is showing that how the carry is propagating non-linearly and the partial sums are generated. As it is already stated that the 32-bits are apportioned in variable sizes i.e. not in a fixed manner, but in unequal sizes so the 32-bits are split up in 4, 4, 5, 6, 7 and 6. It is clear that there are 6 stages which are used to implement the sum of 4-bits, 4-bits, 5-bits, 6-bits, 7-bits and 6-bits. All these stages are implemented in the form of branching pipeline.

The branching pipeline architecture is another concept which is implemented here to reduce the requirement of power consumption. Here, by keenly observe the structure we come to know that at one time carry will propagate through one stage only.

The inputs are given to the 1st stage then it will generate the partial and an output carry and this output carry will be forwarded to II’nd stage and as soon as this stage generate the carry output it will provided to the next stage. From this discussion it is very much clear that the carry propagation is taking place only through one stage at a time.

Therefore, the concept which has been utilizing in this work is to activate only that pipeline through which carry is propagating while cut-off the others. This concepts proves to be very helpful in the reduction of consumption of power, thus the power loss has been minimized, otherwise in the previous SQRT CSLA all the pipelines were activated at all the time and thus the power requirement by every pipeline stage will be cumulatively added and thus it is obviously greater than the power requirement of this work. The internal structure of 32-bits is shown in Fig 4.5.

In such a way, this research work reduces the power consumption from the previous work as well as increases the speed because it requires less number of stages to compute the sum of 32-bits.

IV. RESULT

The minimum delay provided by this CSLA is 0.598ns In this System, a fully parallel processing architecture has been implemented successfully to compute the sum of 32-bits.

Let us also look at the macro statistics, i.e. the number of registers, flip-flops and multiplexer required. To implement the whole design, there are 96 number of registers are required that mean it need 96 flip-flops amd in total 57 multiplexers has been used. In which 31 multiplexer are of 2-bit 2:1 Mux, 15 are of 3-bit 2:1 Mux, 7 are of 5-bit 2:1 Mux, 3 are of 9-bit 2:1 Mux and in the last one multiplexer of 17-bit 2:1 Mux are required.

In a synthesis results, there are shown the top level block diagram, RTL view and the technological view of the implemented CSLA.

In the top level block diagram a green block shows the block diagram of 32-bit carry select adder which consist of two buses A<31:0> and B<31:0> and clock pulse and sum output is obtain of 32-bit. The RTL view shows the green blocks which signify various hardware blocks and the red lines signify wires and in the technological view green blocks represent Look Up Tables (LUT’s), Carry logic , Input Output Blocks (IOB) and
other technology specific component and red wires indicate the interconnects.

V. CONCLUSION

Here the conclusion is obtained that this proposed structure of CSLA is better than all the architectures that have been implemented earlier, because all the three constraints are optimized by using this proposed CSLA. In this project, the entire adder architecture has been implemented using VHDL and simulated using Mentor Graphics tool suite. The architecture is then further evaluated in FPGA using Xilinx ISE Design Suite. Power has been reduced by 23.7% to that of the previous work due to implementation of stages in branching pipeline. Delay has reduced to 89.09% because the carry propagation is routed non-linearly therefore number of stages has been reduced. Area is comparatively similar but still less than by 0.07%. This thesis has dealt with fundamental concepts of addition and optimization. Here we are presented a number of interesting results. Thus, the obtain results are much better in comparison to previous work in all the respects.

REFERENCES