International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-8, June 2020

64bit Hybrid Adder for ALU Design Applications

Hema Singaravelan, Ravi S.

Abstract: ***The Arithmetic Logic Unit is an important component of any Central Processing Unit. An improvement of the speed, area, and power consumption of an ALU directly promotes the performance of the system. Thus, optimization of the ALU design is necessary and for this reason several common adders such as the ripple carry adder, etc. and a proposed model of a 64bit hybrid adder were designed, and a comparative analysis of their performance was studied. The proposed hybrid adder was developed using an 8bit Ripple Carry adder that evaluates the LSB followed by a Carry skip adder block consisting of a 4bit Carry Skip Adder, an 8bit Carry Skip, another 8bit Carry Skip, followed by a 4bit Carry Skip Adder, and finally the MSB is calculated by a 32bit Carry Select Adder. The adders were designed in Verilog on ModelSim-Altera 10.1d (Quartus II 13.0sp1) and later the schematic was obtained on Genus Synthesis (RTL Compiler) of Cadence for ASIC design using 45nm technology. Each adder showed some advantages, but the proposed hybrid adder optimized all aspects of the model while increasing the speed of the device.***

Keywords – ALU, Ripple Carry Adder, Carry Look Ahead Adder, Carry Save Adder, Carry Select Adder, Carry Skip Adder, Hybrid Adder, ASIC, Binary Multiplier, Divider, Control Unit, Delay, Area, Power Consumption, Optimization, Cadence Genus Synthesis (RC)

I. INTRODUCTION

An Arithmetic Logic Unit, abbreviated as ALU, is a combinational digital electronic circuit integrated into systems to perform arithmetic and bitwise operations on operands, that are either binary numbers or integers.

A mathematician, John von Neumann, first proposed the concept of an ALU in 1945 in a report on the foundations for a new computer called the EDVAC. They were called 'math units' at the time and they performs simple math operations, such as, addition. Over time the need and necessity of ALU in computers increased and with Moore's law, it became feasible to build wider ALU's on advanced microprocessors. They are now being used to process more data through the simple math functions that are implemented on them as in the case of manipulating 16bit words.Thus, it became a fundamental building block of several computing circuits, such as, Central Processing Unit, Graphic Processing Units, Floating Point Units, etc. These blocks may comprise of one or more ALU circuits. Recent research have mentioned the elaborate methods to reduce delay, power and area by compromising either one of those three parameters.

Revised Manuscript Received on June 30, 2020.
***** Correspondence Author
Hema Singaravelan*, School of Electronics, Vellore Institute of Technology, Vellore, India. Email: hemasingaravelan@gmail.com
Ravi S., School of Electronics, Vellore Institute of Technology, Vellore, India. Email: msravi@vit.ac.in

Also some of them have explored the usage of hybrid adders[3] to obtain optimized outputs with respect to delay, area and power consumption. Thus, an implementation of a hybrid adder performing binary addition to improve of speed, power consumption and area in an attempt to optimize the results.

II. OBJECTIVE AND METHODOLOGY

A. Objectives

Objective of the experiment is to design a 64 bit Hybrid adder that improves on the efficiency of the other existing adder circuits by,

- Improving the delay at which the output of the adder circuit is obtained from the time at which the input is applied to it, when in comparison with the other adder circuits
- Improving on the area of the circuit layout in comparison with a few of the standard adders
- Improving on the power consumption of the circuit in comparison with other proposed models

B. Methodology

Optimization is achieved on the basic full adder that comprises the complex 64bit adder circuits and a final proposed model for 64bit addition is achieved using the concept of hybrid adders, that are in fact a combination of the common adders to achieve optimization of area, power dissipation, and speed.

Also the performance of 64bit Ripple Carry, Carry Look Ahead, Carry Save, Carry Select, and Carry Skip Adders[4] were studied to conduct a comparative analysis of the performance of those adders against the proposed model to prove that the model indeed is optimized and to also look for any scope for improvement in the proposed model. These studies were conducted by producing the Verilog code, the TCL file, for the simulation on Cadence Genus Synthesis (RTL Compiler). The studies are done with reference to some published articles.

III. DESIGN APPROACH AND DETAILS OF IMPLEMENTATION

Designing of basic adders such as ripple carry adder, carry look ahead adder, carry select adder, carry save adder, and carry skip adder to add two 4bit numbers with a carry input in Verilog was accomplished first to study the functionality of the adders before pursuing higher bit additions using the same.

Errors were avoided as significant knowledge on how to obtain the structural code for all the adders was gained and the higher bit additions were done using the simpler modules created initially to study their performance.

The simple adder circuits with their descriptions is given below[5][6],

A. Full adder

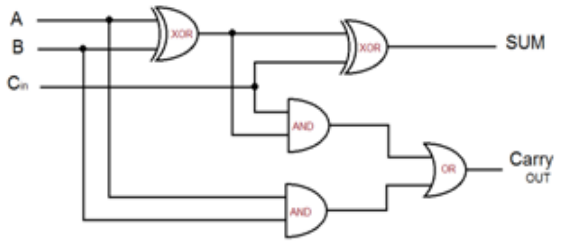

Fig. 1: Full Adder

Fig. 1 is the conventional adder[1] circuit that is applied to most applications as a standard design for the same due to the increased efficiency compared to the one obtained through the K-map of the full adder. The carry out is obtained after a three level (gate) processing.

An improved design was implemented in this project to increase the speed of the circuit and the diagram of the same is as given below in Fig. 2.

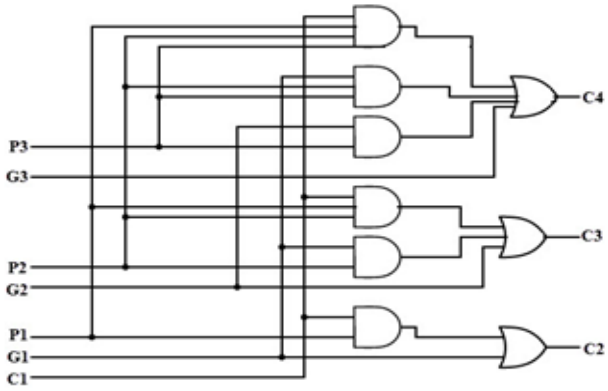

Fig. 2:4 bit Carry Look Ahead Adder

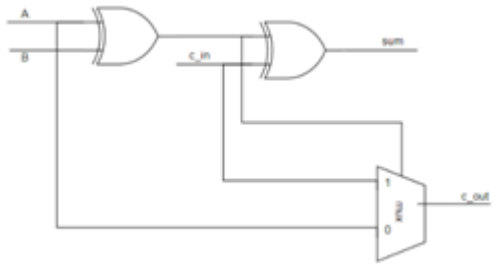

Fig. 3:Full Adder with MUX

The design shown in Fig. 2 [7] significantly reduces the area of the full adder and also promotes a chance to attempt at a two level path to obtain the carry[2]. This architecture for a full adder has been implemented in all the complex adder circuits as a basic unit. This is then followed by the implementation of 4bit adders to study the working mechanism of these standard models.

B. Ripple Carry Adder

The ripple carry adder shown in Fig. 3 is the basic of all adders that follows the simple rule of addition, that is to pass the carry on to the next step to produce the necessary output. Although the area and power consumption of this model is the least the main drawback of this model is the high amount of time that it utilizes to produce the carry

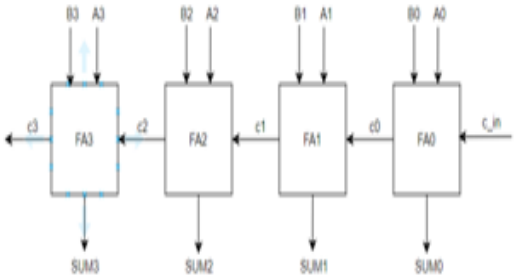

Fig. 4:4 bit Ripple Carry Adder

output since each full adder module of the model needs to wait for the previous model in order to begin its computation and push the carry to the next level. This was addressed in the Carry Look Ahead adder.

C. Carry Look Ahead Adder

The carry look ahead adder improves on the delay by calculating the propagate and the generate using a half adder and supplied to the carry look ahead block as shown in Fig. 4 to calculate the carry at each step. The generates and the propagates are the carry and sum of a half adder, respectively. The carry look ahead block is designed using the formulae in Eq.(1) to (4),

$$C_1 = G_0 + P_0.C_0, \quad (1)$$

$$C_2 = G_1 + P_0.C_1, \quad (2)$$

$$C_3 = G_2 + P_0.C_2, \quad (3)$$

$$C_4 = G_3 + P_0.C_3 \quad (4)$$

Where, C_i is the calculated carry, G_i is the generate which is the **and** operation on each set of corresponding bit inputs and P_i is the propagate which is the **xor/or** operation on each set of corresponding bit inputs, calculated with respect to bit i. Then this along with the propagates at each stage is then fed to the adder or the XOR block to obtain the sum. This increases the area of the model but performs extraordinarily in terms of speed over the Ripple Carry Adder. Several attempts to reduce the computational complexity and increase the speed of adders were thus made and one other such adder is the Carry Save Adder.

D. Carry Save Adder

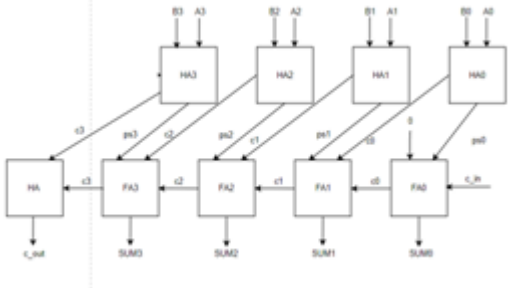

Fig. 5:4 bit Carry Save Adder

The carry save adder implements the method of calculating the partial carry and partial sum values at each stage to which is then, after left shifting the partial carry outputs by one place and padded with a zero, are added together using a ripple carry adder as shown in Fig. 5. The formulae to calculate the partial carries and partial sums are given below in Eq.(5) and Eq.(6),

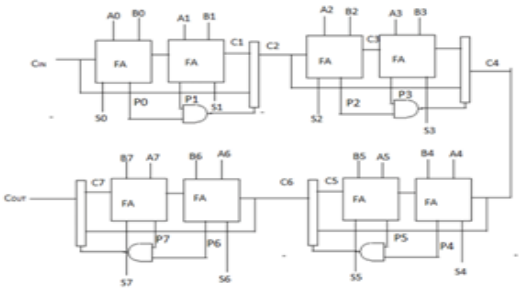

Fig. 6:8 bit Carry Skip Adder

$$ps_i = a_i \oplus b_i \oplus c_i, \quad (5)$$

$$sc_i = (a_i \wedge b_i) \vee (a_i \wedge c_i) \vee (b_i \wedge c_i) \quad (6)$$

Where, ps is the partial sum and pc is the partial carry at bit i= 0, 1, 2,..Although this model is efficient in the case of small bit additions, it consumes high area and power at a small reduction in delay. Also the addition results are still not known at once, and the result of the addition is not known to be larger or smaller than a given number in the case of subtraction without the implementation of a comparator.

E. Carry Select Adder

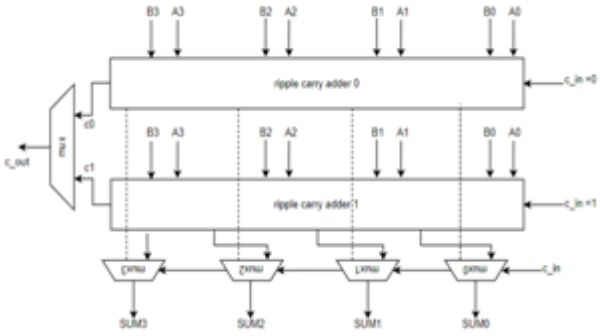

Fig. 7:4 bit Carry Select Adder

This adder also addresses improvisation of performance through increase in output production. This model aims at decreasing the delay in obtaining the output when there is a delay in the circuitry that precedes the adder. It consists of two Ripple Carry Adders, that compute the output each under an assumed carry, either 0 or 1, while it waits to confirm the actual carry to the circuit. Thus, when the carry arrives at the input, a series of MUXs pick the required output as per the value of that carry. This is depicted in Fig. 6. The major drawbacks of this model are increased power consumption, increased area, and in the case where it is the initial block, it also increases the delay of the circuit. Also when implemented with other blocks it must be ensured that the delay before the carry select block has a delay at least in comparison with the delay of the Ripple Carry Adders of this adder.

F. Carry Skip Adder

Another attempt at increasing the performance of adders over the ripple carry adder is carry skip adder that improves on the delay of ripple carry adders with much less effort compared to the other adder designs. The carry skip adder performs best when higher bits are produced through sliced inputs supplied to smaller carry skip blocks that are concatenated.

The adder works in the fashion that the propagates are obtained from full adders are supplied to an and gate who's output determines whether the output carry is equal to the input carry or the output carry of the ripple carry block through a mux.

The drawbacks of this model is that the adder block must be sliced into smaller blocks of carry skip adders, as shown in Fig. 7, to obtain the output quickly in the case of higher bit additions as waiting for the propagate at each step would then resemble that of the Ripple Carry Adder's wait for carries. Also delays of previous blocks must be held in mind before optimizing each carry skip block of the adder.

G. Proposed Hybrid adder

The proposed model for a 64bit hybrid adder was then formulated and the Verilog code of the same was obtained. It consists of an 8bit Ripple Carry adder the begins the evaluation of the LSB followed by a Carry skip block consisting of a 4bit Carry Skip Adder, an 8bit Carry Skip, another 8bit Carry Skip, followed by another 4bit Carry Skip Adder, and finally this is followed by a 32bit Carry Select Adder as shown in Fig. 8.

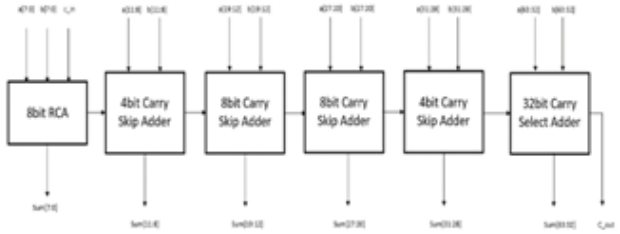

Fig. 8: 64bit Proposed Hybrid Adder design

This design works under the principle of increasing the speed of calculation of the MSB by increasing the speed of propagation of carry though the Carry Skip block and also ensuring the delay before the Carry Select block allows it to compute the necessary outputs and wait for the incoming carry to produce 32 bits of the output at the same instant.

All of the above was designed in Verilog and then simulated on Cadence Genus synthesis tool to obtain the circuit layout, critical path delay, total power consumption, total area, and slack.

This data was then compared with the other standard 64 bit adder outputs and the resulting conclusions were drawn that will be discussed later on.

IV. RESULTS AND DISCUSSION

After the Verilog code for the above adders were developed, the same was applied to Cadence Genus Synthesis(RTL Compiler) and the results displayed in table I were obtained,

TABLE I. RESULTS OBTAINED AFTER RTL SYNTHESIS

Adders (64 bit)	Area (μm^2)	Power (nW)	CP delay(ps)	Slack (ps)
RCA	1565	337284.458	10416	284
CLA	1609	342465.458	8738	1962
Carry Select Adder	2217	419434.068	10463	237
Carry Save Adder	1806	365270.767	10614	86
Carry Skip Adder	1695	350885.655	10697	3
Proposed Hybrid Adder	1919	380649.056	6631	4069

The clock period used for the above designs was set at 10.7ns at the sdc file.

Therefore, from the above result it can be observed the CP (Critical Path) delay and the slack of the design suggest that the proposed design performs better than the rest of the designs simulated alongside the proposed adder for comparison of results. This model had increased the speed of execution by 24.11% of decrease of critical path delay in the proposed model in comparison with a 64 bit Carry Look Ahead Adder while the increase in area is 19.2% and power consumption is increased by 11.14%. It can also be noted that the area and power consumptions in correlation with the increasing complexity and speed of the adder did stays relatively achievable in a physical design of the proposed added. The circuit diagram as produced by RTL compiler for the proposed adder is given in Fig. 9.

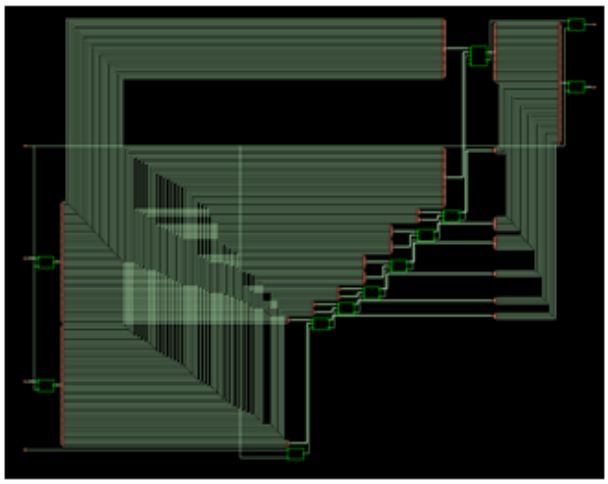

Fig. 9:Proposed adder circuit

V. LIMITATIONS AND SCOPE FOR IMPROVEMENT

The limitation of the 64bit proposed adder is that it can only be reduced to 24bit addition and also efficiency reduces with increase in the number of bits thus a better format in applying the component adders must be devised.

There is no standard ratio of applying the proposed adder in below 24bit addition scenarios, which is a limitation.

Other standard adders may also be substituted in the designed adder to produce more efficient adders so as to improve on area and power consumption.

Other blocks of an ALU may be implemented using the hybrid adder as building blocks such as, subtractors or multipliers. Substituting the adders of the multipliers[8][9] with this proposed design will enhance the speed of the design and it a potential progress made in designing an ALU using the proposed model.

VI. CONCLUSION

From the results it can be seen that the adders- carry look Ahead, and proposed have a significantly reduced delay in comparison with ripple carry adder.

The carry save adder when created without breaking them into simpler block for computational advantage does not perform better than a ripple carry adder in adding two 64bit numbers. On the other hand simpler blocks of the carry skip adder cascaded together to compute a 64 bit output perform better than a ripple carry adder of the same bit size. It also must be noted that although theoretically feasible the carry skip adder cannot be designed for 64bit addition even when reduced to simpler locks (4bit carry skip adders-cascaded) with the same clock as applied to the other adders since the slack is less than 50ps which indicates that even if the model is designed, chances are positive that the output might be calculated past the given clock time interval. The proposed adder performs better than all other adders in a 64bit setup in terms of speed of output production. . This model had increased the speed of execution by 24.11% of decrease of critical path delay in the proposed model in comparison with a 64 bit Carry Look Ahead Adder, which was the fastest of all the adders designed in for this study.

ACKNOWLEDGMENT

Expressing gratitude to VIT for providing such an opportunity to access advanced tools in the labs and gaining profound knowledge through experimentation. A special thanks to various publications and cited authors who have made helped materialize this experiment by spreading the knowledge and work.

REFERENCES

1. Padmanabhan Balasubramanian, Nikos Mastorakis, “High Speed Gate Level Synchronous Full Adder Designs”,in WSEAS Transactions on Circuits and Systems, 2009, pp. 8(2):290-300
2. Ramesh Boda, M.Charitha, B.Yakub, R.Sayanna, “Multiplexer-Based Design of Adders/Subtractors and Logic Gates for Low Power VLSI Applications”, in IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), 2015, Volume 5, Issue 6, Ver. II, pp. 59-66
3. Habib Ghasemizadeh, Akbar Ghasemizadeh Tamar, Khayrollah Hadidi, Abdollah Khoei, Pourya Hoseini, “High speed area reduced 64-bit static hybrid carry-lookahead/carry-select adder”, in 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2011), 2011
4. Rashmi Rahul Kulkarni, “Comparison among Different Adders”, in OSR Journal of VLSI and Signal Processing (IOSR-JVSP), 2015, Volume 5, Issue 6, Ver. I, pp. 01-06
5. Jasbir Kaur, Lalit Sood, “Comparison Between Various Types of Adder Topologies”, in IJCST, 2015, Vol. 6, Issue 1

6. Shubham Sarkar, Sujan Sarkar, Jishan Mehedi, "Comparison of Various Adders and their VLSI Implementation", IEEE International Conference on Computer Communication and Informatics (ICCCI), 2018
7. Ravi S., Shaji Nair, Rajeev Narayan, H. M. Kittur, "Low Power and Efficient Dadda Multiplier", Research Journal of Applied Sciences, Engineering and Technology, 2015, pp. 9(1):53-57
8. Shahzad Asif, Yinan Kong, "Low-Area Wallace Multiplier", Hindawi Publishing Corporation, 2014
9. Sudhir Bussa, Ajaykumar Rao, Aayush Rastogi, "Design of Binary Multiplier Using Adders", International Journal of Electrical and Electronics Research, 2016, Vol. 4, Issue 1, pp: 169-173

AUTHORS PROFILE

Hema Singaravelan, School of Electronics, Vellore Institute of Technology, Vellore, India. Email: hemasingaravelan@gmail.com

Ravi S., School of Electronics, Vellore Institute of Technology, Vellore, India. Email: msravi@vit.ac.in