A Review on Area-Delay-Power Efficient Carry-Select Adder

Priyanka Agrawal[1], Prof. Vijay Yadav[2], Dr. Rita Jain[3]

[1]*Mtech. Scholar,* [2]*Guide,* [3]*HOD*

Department of Electronics and Communication, LNCT Bhopal

Abstract- Managing power consumption are and delay is an aspect of microprocessor design that is important and growing increasingly more so. In mobile electronics optimizing power consumption is a key factor in increasing battery life, and even in servers and desktop computers power dissipation is an important design constraint. In any form of engineering there are inevitably tradeoffs to be made, and one of the primary tradeoffs facing designers is that of trying to balance performance against power dissipation. The design of high-speed and low-power VLSI architectures need efficient arithmetic processing units, which are optimized for the performance parameters, namely, speed and power consumption. Adders are the key components in general purpose microprocessors and digital signal processors. They also find use in many other functions such as subtraction, multiplication and division. As a result, it is very pertinent that its performance augers well for their speed performance. The simplest form of adder is the ripple-carry adder. An n-bit ripple carry adder consists of n one-bit full adders connected in succession. in these exploration different types of binary adders Ripple carry adder, Carry look-ahead adder, carry select adder and parallel prefix adders has studied. The carry "ripples" from the least significant bit to the most significant bit and hence the name.

Keywords- binary adders, carry select adder, Carry look-ahead adder, Ripple carry adder, CMOS, VLSI.

I. INTRODUCTION

Power dissipation is one of the most important design objectives in integrated circuit, after speed. Digital signal processing (DSP) circuits whose main building block is a Multiplier-Accumulator (MAC) unit. High speed and low power MAC unit is desirable for any DSP processor. This is because speed and throughput rate are always the concerns of DSP system. Due to rapid growth of portable electronic systems like laptop, calculator, mobile etc., and the low power devices have become very important in today world. Low power and high-throughput circuitry design are playing the challenging role for VLSI designer. For real-time signal processing, a high speed and high throughput MAC unit is always a key to achieve a high performance digital signal processing system. A regular MAC unit consists of multipliers and accumulators that contain the sum of the previous consecutive products. The main motivation of this work is to investigate various multiplier and adder architectures which are suitable for implementing Low power, area efficient and high speed MAC unit.

An adder is the main component of an arithmetic unit. A complex digital signal processing (DSP) system involves several adders. An efficient adder design essentially improves the performance of a complex DSP system. A ripple carry adder (RCA) uses a simple design, but carry propagation delay (CPD) is the main concern in this adder. Carry look-ahead and carry select (CS) methods have been suggested to reduce the CPD of adders.

Through choices in such factors as voltage levels, micro architecture, logic style, or gate sizing a designer can make tradeoffs between the two, and seek to find the balance that will best fit the designer's goals. To make these choices designers require information about the consequences of their decisions, and circuit designers use models of varying accuracy and speed to help them forsee the consequences of their choices. An accurate understanding of the causes and severity of the power dissipation within a chip might influence a designer to make certain tradeoffs between speed and power, but a less accurate understanding might lead to tradeoffs that are worse for overall performance. Clearly an accurate representation of how a microchip dissipates power is instrumental to good design. Circuit activity is generally one of the most important factors in energy dissipation. "Arithmetic optimization using carry-save-adders", Carry-save-adder (CSA) is the most often used type of operation In Implementing a fast computation of arithmetic's of register-transfer level design in industry. This work establishes a relationship between the properties of arithmetic computations and several optimizing transformations using CSAs to derive consistently better qualities of results than those of manual implementations. Optimal Allocation of Carry-Save-Adders in Arithmetic Optimization", Carry-save-adder (CSA) is one of the most widely used schemes for fast arithmetic in industry.

Furthermore, for the applications such as the RISC processor design, where single cycle execution of instructions is the key measure of performance of the circuits, use of an efficient adder circuit becomes necessary, to realize efficient system performance.

Additionally, the area is an essential factor which is to be taken into account in the design of fast adders. Towards this end, high-speed, low power and area efficient addition and multiplication have always been a fundamental requirement of high-performance processors and systems. The major speed limitation of adders arises from the huge carry propagation delay encountered in the conventional adder circuits, such as ripple carry adder and carry save adder.

II. CONVENTIONAL ADDITION

1. Serial Addition

Following the steps in the algorithm for serial addition generates the sum one bit at a time, starting from the least significant bit. This is the familiar hand method but done using the binary number system. This is also the fast algorithm for mechanical addition described in the previous chapter. This algorithm was described using Boolean algebra by Shannon [8].

Figure 2.1 shows the result of performing these three steps with two three bit operands. The operands are 3 (011) added to 2 (010) to yield 5 (101). The operands are in large bold face. The carry values are shown underneath the dotted line. The dotted line is used to signify that the carries are generated separately from the original operands, but that they are still added in as though they form a third operand. The solid line separates the sum from the carries. Addition proceeds from the right to the left. The usual implementation of serial addition contains a parallel component since both sums and carries are generated simultaneously in each column.

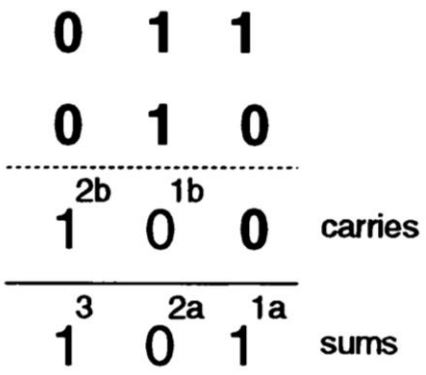

Figure 2.1 Serial Additions Algorithm.

However, the algorithm is serial in the sense that the columns are processed sequentially starting from the least significant bit. The serial steps in figure 2.1 are numbered consecutively, while letters are used to distinguish among the steps that may occur in parallel or in any order.

2. Ripple Carry Addition

Ripple carry addition is similar to serial addition. The principal difference is that ripple carry addition is asynchronous while serial addition is clocked. In ripple carry addition all of the domino style carry chains start simultaneously, as shown for the example case in figure 3.2. Because of the overlapping calculations, signals in the ripple carry adder can transition multiple times1. The vacillation is denoted in the example with crossed out bits.

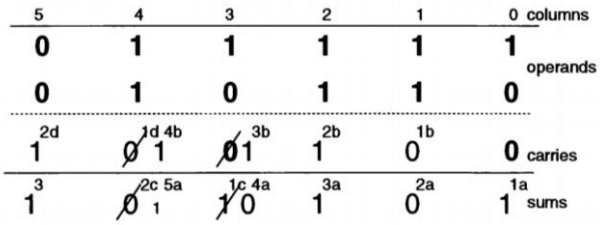

Figure 2.2 Ripple carry Addition Algorithm.

Figure 2.2 shows the salient events for an example add: the generation of two carries, the carries propagating across the adder, and then later one path running over the other in the upper four bits. More specifically, when assuming that the adder starts with all zero carries, the first step in the algorithm produces the sums in columns 0 and 3, and the carries for columns 1 and 4.

3. Carry Select Addition

This method is a divide and conquer strategy which hedges bets on the outcome of intermediate carries. The length of ripple propagate chains must be reduced somehow in order to speed up addition. If we assum at any particular point in the adder that the carry would always be zero, then the adder could be broken at that point into separate parallel processes, and the sum could be completed sooner. However, the assumption would be wrong 50% of the time can hedge bets on the outcome of such a carry calculation by starting two upper half adds, each with a different assumption about the value of the particular carry-out. Figure 3.3 shows an example of adding 01011111 to 01010110. The upper four bits of the add is duplicated, the top version has the carry input bit set, while the bottom version does not. In the first serial step six items are calculated simultaneously. These include the first column sum bits in each of the three additions, and the first column carry-out bits. The three additions continue on independently as in ripple carry addition until the carry emanating from the least significant four bits causes the selection of the correct upper four bits, 1011 in the last step. This carry select addition requires only five serially dependent steps to add 8 bit operands. Ripple carry addition of the same length operands would require 8 serially dependent steps.

The carry select algorithm is then:

(A). Perform three ripple carry adds simultaneously:

i. One add with the lower half bits.
ii. Two adds with the upper half bits.
- One with the carry-in set to one.
- One with the carry-in set to zero.

(B). Pick the correct upper half.

Although the number of series steps for performing this type of addition is approximately half the number for ripple carry addition, constant factors do not change the order of growth, and the performance is still linear:

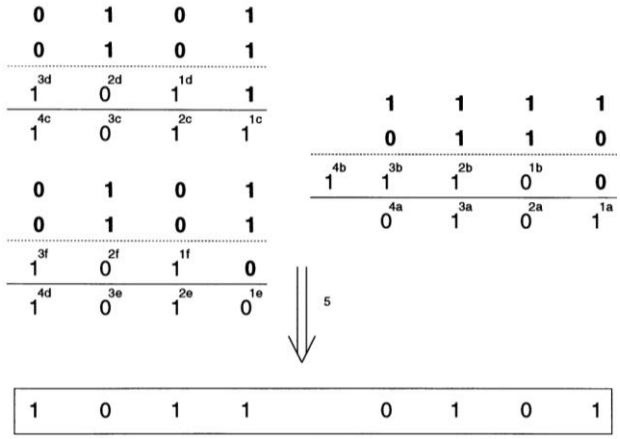

Figure 2.3 Carry Select Addition Algorithm.

4. Ripple Carry Select

Another conventional algorithm for addition is made by breaking ripple carry addition into a larger number of pieces than the two pieces of the carry select method.

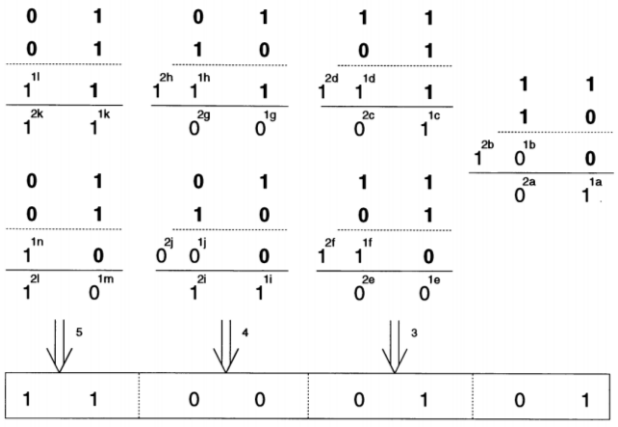

Figure 2.4 A ripple Carry Addition algorithm.

For example, the 8 bit addition from the previous example can be broken into four 2 bit pieces, as shown in figure 2.4. Each of the 2 bit pieces contains duplicated 2 bit adders: one copy adds with an assumed carry-in of one, and the other with an assumed carry-in of zero. The correct pieces are selected in ripple fashion: the bottom 2 bit add selects among the correct second section add, and the correct second section add then provides the correct carry for the third section, which provides the correct carry for the fourth section.

III. PRIOR WORK

B. K. Mohanty and S. K. Patel,[1] In this brief, the logic operations involved in conventional carry select adder (CSLA) and binary to excess-1 converter (BEC)-based CSLA are analyzed to study the data dependence and to identify redundant logic operations. We have eliminated all the redundant logic operations present in the conventional CSLA and proposed a new logic formulation for CSLA. In the proposed scheme, the carry select (CS) operation is scheduled before the calculation of-final-sum, which is different from the conventional approach. Bit patterns of two anticipating carry words (corresponding to cin = 0 and 1) and fixed cin bits are used for logic optimization of CS and generation units. An efficient CSLA design is obtained using optimized logic units. The proposed CSLA design involves significantly less area and delay than the recently proposed BEC-based CSLA. Due to the small carry-output delay, the proposed CSLA design is a good candidate for square-root (SQRT) CSLA. A theoretical estimate shows that the proposed SQRT-CSLA involves nearly 35% less area-delay-product (ADP) than the BEC-based SQRT-CSLA, which is best among the existing SQRT-CSLA designs, on average, for different bit-widths. The application-specified integrated circuit (ASIC) synthesis result shows that the BEC-based SQRT-CSLA design involves 48% more ADP and consumes 50% more energy than the proposed SQRT-CSLA, on average, for different bit-widths.

L. Mugilvannan and S. Ramasamy,[2] Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. From the structure of the CSLA, it is clear that there is scope for reducing the area and power consumption in the CSLA. This work uses a simple and efficient transistorlevel modification in BEC-1 converter to significantly reduce the area and power of the CSLA. Based on this modification 16-b square-root CSLA (SQRT CSLA) architecture have been developed and compared with the SQRT CSLA architecture using ordinary BEC-1 converter. The proposed design has reduced area and power as compared with the SQRT CSLA using ordinary BEC-1 converter with only a slight increase in the delay. This work evaluates the performance of the proposed designs in terms of delay, area, and power by hand with logical effort and through Cadence Virtuoso. The results analysis shows that the

proposed CSLA structure is better than the SQRT CSLA with ordinary BEC-1 converter.

L. Suri, D. Lamba, K. Kritarth and G. Sharma,[3] Highly-increasing requirement for mobile and several electronic devices want the use of VLSI circuits which are highly power efficient. The most primitive arithmetic operation in processors is addition and the adder is the most highly used arithmetic component of the processor. Carry Select Adder (CSA) is one of the fastest adders and the structure of the CSA shows that there is a possibility for increasing its efficiency by reducing the power dissipation and area in the CSA. This research work presents power and delay analysis of various adders and proposed a 32-bit CSA that is implemented using Hybrid PTL/CMOS logic style. This work evaluates and analyses the performance of the proposed designs in terms of area, delay, power, and their products in 90nm CMOS process technology. The results analysis is showing that the proposed CSA structure shows better result in terms of area, power and PDP (Power Delay Product) than the others.

A. Grover and N. Grover, [4] This article proposed an area-efficient carry select adder by sharing the common Boolean logic term. Representation of the circuit in summation operation need one XOR gate and one Inverter, while carry out can be represented using one AND gate and an inverter. Using the multiplexer, we are able to select the correct output results according to the carry input signal. In this way, the transistor count in a 32-bit carry select adder can be greatly reduced from 1947 to 960 and the carry select adder performs with a faster speed as compare to the carry ripple adder. Two different design styles using one multiplexer and two multiplexers has also been considered in terms of number of transistors, average power consumption, propagation delay at sum and at carry output.

A. Ramakrishna Reddy and M. Parvathi,[5] Most of the VLSI applications, such as DSP, image and video processing, and microprocessors use carry select adder (CSLA) for arithmetic functions. From the structure of regular SQRT CSLA, still there is possibility to obtain better design in which optimization of area, power are to be major concentrations along with high speed performance. One of the existing solutions used in SQRT CSLA is replacement of second level RCA by BEC. Though increases the performance, very less percentage of improvement in reduction of area and power dissipation. And also the existing adder with BEC technique is not suitable for low power applications. Hence this work proposes Special Hardware using Multiplexers (SHM) design in place of second level RCA. It is observed from the results that the area and power dissipation are reduced at comparable percentages with respect to the RCA and BEC techniques. When SHM is used at the second level of second block in 16-bit SQRT CSLA, observed that area is reduced by 13.5% and power dissipation is reduced by 6.4%. This proposed logic is designed in transistor level using 0.12μm technology in the Micro wind tool.

M. A. Akbar and J. A. Lee, [6] The common design problem in various approaches for self-checking adders is the fault propagation due to carry. Such a fault can misguide the system to detect the particular faulty module. In this work, we proposed a self-checking Carry Select Adder (CSA) with fault localization ability. Our scheme can provide minimum area overhead for self-recovery process because instead of replacing the whole system we can now replace the particular faulty modules. The proposed self-checking CSA consumes 12% less area with equal performance as compared to the previously proposed self-checking CSA approach.

S. Parmar and K. P. Singh,[7] The work describes the power and area efficient carry select adder (CSA). Firstly, CSA is one of the fastest adders used in many data-processing systems to perform fast arithmetic operations. Secondly, CSA is intermediate between small areas but longer delay Ripple Carry Adder (RCA) and a larger area with shorter delay carry look-ahead adder. Third, there is still scope to reduce area in CSA by introduction of some add-one scheme. In Modified Carry Select Adder (MCSA) design, single RCA and BEC are used instead of dual RCAs to reduce area and power consumption with small speed penalty. The reason for area reduction is that, the number of logic gates used to design a BEC is less than the number of logic gates used for a RCA design. Thus, importance of BEC logic comes from the large silicon area reduction when designing MCSA for large number of bits. MCSA architectures are designed for 8-bit, 16-bit, 32-bit and 64-bit respectively. The design has been synthesized at 90nm process technology targeting using Xilinx Spartan-3 device. Comparison results of modified CSA with conventional CSA show better results and improvements.

IV. PROBLEM STATEMENT

In a microprocessor or a digital signal processor (DSP), data path plays a prominent role since performance metrics like the die-area, speed of operation, power dissipation etc., depend directly on the efficiency of data-path. As is known, core of the data path involves complex computations like addition, subtraction, multiplication and division, etc. Thus, realizing efficient hardware units for these computations, which directly affect the performance of data path, is of prime importance. The most executed operation in the data path is addition, which requires a binary adder that adds two given numbers. Adders also

play a vital role in more complex computations like multiplication, division and decimal operations. Hence, an efficient implementation of binary adder is crucial to an efficient data path. Relatively significant work has been done in proposing and realizing efficient adder circuits for binary addition.

V. CONCLUSION

For development of efficient adder architectures that address the problems of high fan-out, delay and power consumption. These architectures while having a delay overhead have the advantage of relatively small fan-out and reduced energy consumption overall. Various literature and different algorithms of binary addition like serial addition ripple carry addition carry select addition and ripple carry addition are reviewed. The ripple carry adder is most popular among all the delay of carry select adder and carry look ahead adder is considerably less compared to ripple carry adders. Some new algorithms are also discussed. The need to identify optimum adder designs for a modern microprocessor initiated this study. Adders often appear in the integer execution unit, and sometimes in the address generation path. 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 appear in the exponent manipulation circuitry for multiply and divide.

REFERENCES

[1] B. K. Mohanty and S. K. Patel, "Area–Delay–Power Efficient Carry-Select Adder," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 6, pp. 418-422, June 2014.

[2] L. Mugilvannan and S. Ramasamy, "Low-power and area-efficient carry select adder using modified BEC-1 converter," 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Tiruchengode, 2013, pp. 1-5.

[3] L. Suri, D. Lamba, K. Kritarth and G. Sharma, "High performance and power efficient 32-bit carry select adder using hybrid PTL/CMOS logic style," 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), Kottayam, 2013, pp. 765-768.

[4] A. Grover and N. Grover, "Comparative Analysis: Area-Efficient Carry Select Adders 180 Nm Technology," 2013 7th Asia Modelling Symposium, Hong Kong, 2013, pp. 99-102.

[5] A. Ramakrishna Reddy and M. Parvathi, "Efficient carry select adder using 0.12µm technology for low power applications," 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, 2013, pp. 550-553.

[6] M. A. Akbar and J. A. Lee, "Self-Checking Carry Select Adder with Fault Localization," 2013 Euromicro Conference on Digital System Design, Los Alamitos, CA, 2013, pp. 863-869.

[7] S. Parmar and K. P. Singh, "Design of high speed hybrid carry select adder," 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013, pp. 1656-1663.

[8] N. Sloane and A. D. Wyner, eds., Claude Elwood Shannon Collected Works. IEEE Press, 1993. [1] Y. He, C. H. Chang, and J. Gu, “An area-efficient 64-bit square root carry- select adder for low power application,” in Proc. IEEE Int. Symp. Circuits Syst., 2005, vol. 4, pp. 4082-4085.

[9] B. Ramkumar and H. M. Kittur, “Low-power and area-efficient carry-select adder,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 2, pp. 371-375, Feb. 2012.

[10] I.-C. Wey, C.-C. Ho, Y.-S. Lin, and C. C. Peng, “An area-efficient carry select adder design by sharing the common Boolean logic term,” in Proc. IMECS, 2012, pp. 1-4.

[11] S. Manju and V. Sornagopal, “An efficient SQRT architecture of carry select adder design by common Boolean logic,” in Proc. VLSI ICEVENT, 2013, pp. 1-5.

[12] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.