IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 10, Issue 06, January 2022)

An Area Efficient Hierarchical Carry Save Algorithm (HCSA) using VHDL

Krishan Kumar Safi
kkrishan040@gmail.com

Ankit Pandit
panditankit48@gmail.com

Ashish Chouhan
ashish.chouhan87@gmail.com

Abstract: **The carry-save adder diverts the carry towards LSB of partial product and in order to decrease the final bits of the adder, CSA generates LSBs. This helps in implementing the pipeline approach which in turn improves the performance. In this proposed scheme the main motive is to remove the independent accumulate stage which creates a huge delay and merge it with the partial product's compression stage. This paper proposed a Hybrid carry-save algorithm for a 32-bit adder with optimized area and speed. The proposed system is developed in VHDL and synthesized in Xilinx.**

Keywords: **CSA, LSB, VHDL, XILINX, etc.**

I. INTRODUCTION

The use of FPGAs has spread very strongly to new fields of application, such as high performance computing, financial computing, cryptography, etc. These new applications require a much higher precision than those implemented in traditional DSP blocks. In addition to the representations in fixed point or floating point, it is common to find implementations today in FPGAs that use floating point representations with double precision or representations in decimal floating point. In these cases operand sizes of 54 bits and larger are used.

The implementation of arithmetic operations with large operands increases the critical path of the signals in the adders when the carry chain implemented by the CPA adders in the FPGAs is used, which reduces the performance. In the implementation of ASIC, redundant arithmetic, especially carry-save, has been used to increase performance for large operands, or a large number of operands. The arithmetic carry-save allows the critical path traveled by the signals in the adders to be practically independent of the operand size.

The use of the carry-save representation in FPGAs was discarded until a few years ago due to several reasons. First of all because of the small size of operands that were used in typical FPGA applications. Secondly, the propagated carry adders (CPA) that they had at the factory offered good performance because the logic of the carry chain had been optimized. Finally, the excessive consumption of area by the synthesis tools when mapping units that work in carry-save. However, in recent years, several works have shown the benefits in the performance, the use of carrysave adders, when the size of the operands grows and it is also possible to efficiently map the carry-save adders in real FPGAs with a very small area increase [1].

Therefore, it is worth studying the implementation of adders and multipliers in FPGAs using redundant arithmetic in special arithmetic carry-save, since it is a technique widely used to improve the performance of the adders in the implementations in dedicated ASIC circuits.

II. CARRY SAVE ADDER

The redundant representation of numbers is used to reduce the summing time, limiting the path of the carry chain to several bits. In this way the time used for the realization of the sum does not depend on the number of bits of the operands. The most common representations are carry-save (CSA) and signed-digit (SD). The carry-save arithmetic is widely used when it is required to add a high number of operands. All the adders previously seen, except the CPA, are designed to accelerate the hauling process and alleviate the delay in obtaining the result of the sum. That is, their design allowed them to work in parallel internally. The adder with carry storage or Carry Save Adder (CSA) follows a different philosophy. It does not aim to achieve sums using parallel internal operations, but to be the parallel processing unit for more complex arithmetic circuits. In this line, it cannot be considered that the CSA [2] is an adder circuit by itself, since it is unable to return a sum word and an output carry bit in response to the sum of two operands and an input carry.

Therefore, it is neither a complete adder nor a half adder. The Carry Save Adder has three words of input and two words of output. One of the answers is the sum of each of the i-th bits of the three operands separately. While the other is composed of each of the carry bits generated in those sums. To obtain the complete result, the word of partial sum has to be combined with the word of carry associated

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 10, Issue 06, January 2022)

to said sum, spreading the produced carries. This last operation is carried out in an adder with carry propagation. As stated above, this type of adder is used in special cases. It is used when you want to add more than two operands (a method widely used in multiplier circuits), since they allow you to delay the haul operation until the end. Thus, its usefulness will not be seen clearly until it explains the acceleration of multiplication [3].

The redundant representation of numbers is used to reduce the summing time, limiting the path of the carry chain to several bits. In this way the time used for the realization of the sum does not depend on the number of bits of the operands. The most common representations are carry-save (CS) and signed-digit (SD). The arithmetic carry-save (CSA) is widely used when it is required to add a high number of operands and in the internal operations of the multipliers [4].

The basic CSA adder performs the sum of three operands using an array of full one-bit adders, but without connecting the carry string, as shown in Figure 1. The result is a redundant number in CS representation that is composed of a sum word (S) and carry word (C).

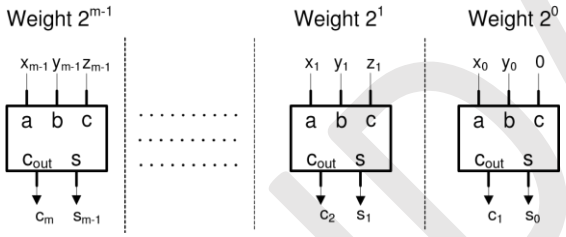

Figure 1 Sum-carry-save of m bits

Therefore, the sum of three operands X, Y, Z of n-bits is represented by two numbers C and S.

$$X + Y + Z = C + S \quad (1)$$

The numbers C and S of n-bits are obtained without propagation of carry with the delay of a single full adder. This representation is called redundant, since many combinations of C and S produce the same number. In Figure 1, by way of example, it can be seen that adding two different numbers that give the same result in conventional arithmetic produce two different combinations of C and S. Therefore, until the sum C and S are not made you will have a number in the conventional representation. For this it will be necessary to use a conventional adder that adds two operands and results in a single number. While all operations are performed in CSA arithmetic, it is not necessary to perform the conversion. It is also possible to perform the conversion "on-the-fly" [5,6], although it increases the consumption of resources since it is necessary to perform a parallel computation.

This circuit CS from another point of view makes the reduction of three binary numbers to two binary numbers, so it is called compressor [3: 2] or counter [3: 2]. If you want to add two CS numbers you need a reduction of 4 to 2. This can be done using two compressors [3: 2] as shown in Figure 2. This circuit is called a compressor [4: 2]. In this case, the computation time for two numbers of n digits CS is that of two complete adders. Note how the carry propagation is cut since the propagated carry of the first level is connected to the input of the full adder of the second level of the next weight bit.

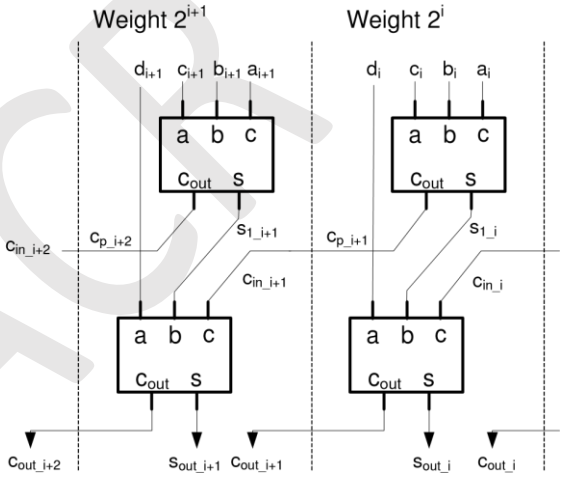

Figure 2: A [4:2] Compressor constructed from two [3:2] compressors [7]

Finally, comment that you can add m operands of one bit and produce a word of sum S and another word of carry C. In this way you have compressors [5:2] that perform a reduction of five bits to two, compressors [6:2] that performs a reduction of 6 bits of input to two and in general compressors [p: t] that performs the sum of p input bits and produces an output of t bits.

One of the disadvantages of the CS representation is that the number of bits involved in the sum is double. To reduce the hardware an alternative is the use of a higher radix, which will be one of the options proposed in the literature and will also be proposed as one of the implementation alternatives that will be shown in this work, for optimization in FPGAs.

The use of the CS representation is especially useful in algorithms that require many intermediate sums since all the sums can be carried out in CS representation and where the conversion to conventional representation does not consume the

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 10, Issue 06, January 2022)

time saved in the CS representation. This is the case of accumulation operations, sum of multioperand, multiplication, division, square root, etc.

CS arithmetic is usually used for the sum of multioperands, especially in multipliers, because the sum of the partial products must be performed [4,5]. To carry it out, compressor shafts are created (not to be confused with compressors as a circuit) that generate the output in arithmetic redundant CS. The sum S and the carry C are finally added to produce a conventional output.

The rest of the work is centered on the carry-save adders, since the extension to the signed-digit representation can easily be achieved by inverting certain inputs and outputs in the compressors, as demonstrated in [8].

One of the disadvantages of the CS representation is that the number of bits involved in the sum is double. To reduce hardware an alternative is the use of a superior radix, as proposed by [9], for optimization in FPGAs.

The use of the CS representation is especially useful in algorithms that require many intermediate sums since all the sums can be carried out in CS representation and where the conversion to conventional representation does not consume the time saved in the CS representation. For example, in operations of convolution, accumulation, sum of multioperand, multiplication, division, square root, etc.

III. HIERARCHICAL CARRY SAVE ALGORITHM (HCSA)

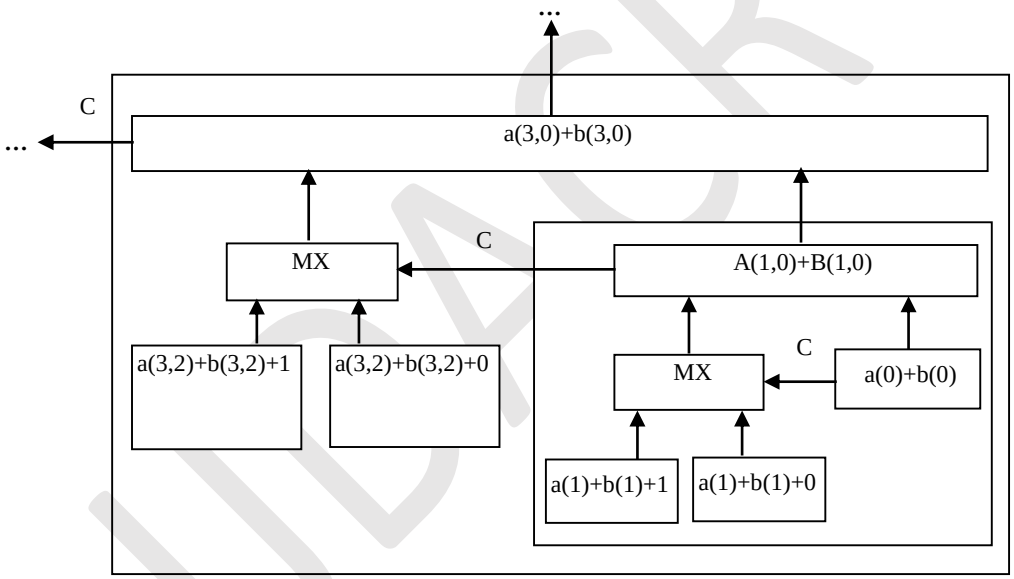

Figure 3: Hierarchical Carry save algorithm

From figure 3 above block diagram explains the hierarchical implementation of carry save algorithm operand bits are spliced into singe bit, then two bit, three bit and so on according to their data size .for an example if the data set is 4 bit then at the start single bit full adder is considered with carry '0' and carry '1' for second bit of operand A and B. The outcome of this full adder (CSA) is sampled to multiplexer the selection of multiplexer will be decided with the addition of first bit (LSB) of operand A and B. The outcome after Mux is further given to next 2 bit full adder ,and again the outcome of these fu2 bit full adder given to next 4 bit full adder, hence according to data size it keep on making hierarchy.

IV. SIMULATION RESULTS

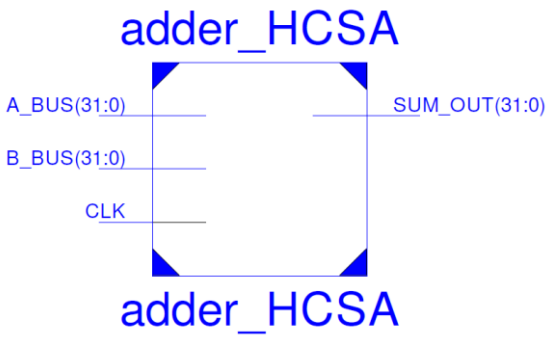

Figure 4: Pin diagram for HCSA adder

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 10, Issue 06, January 2022)

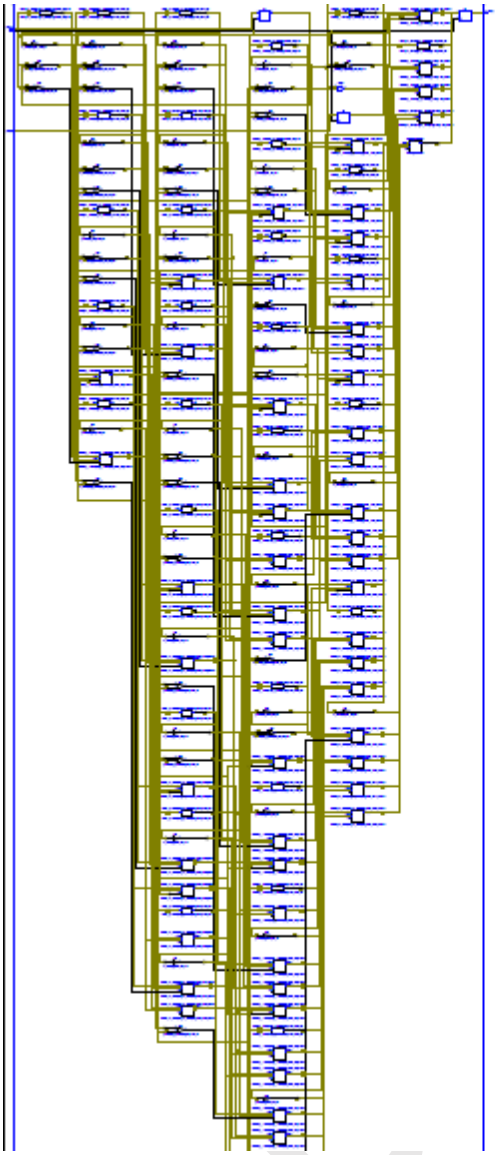

Figure 5: RTL schematic for HCSA adder

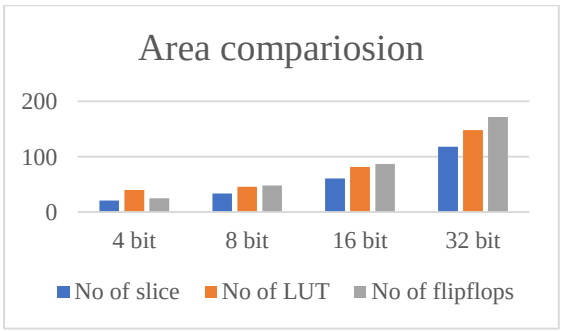

Figure 6: Area optimization optimized modified adder

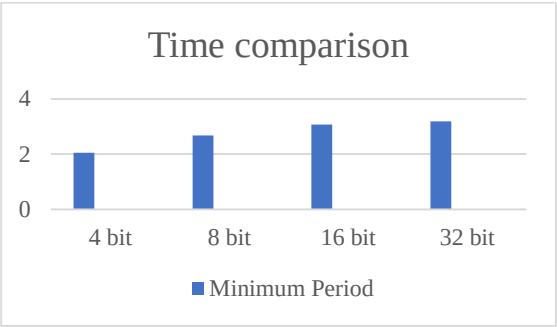

Figure 7: Time comparison modified adder

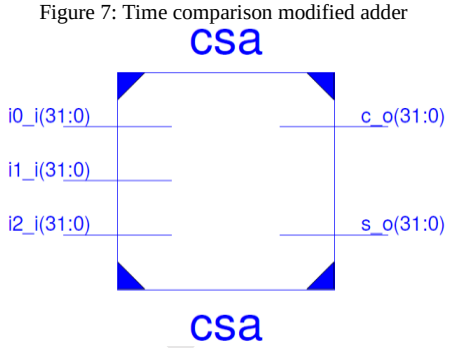

Figure 8: Pin diagram for CSA adder

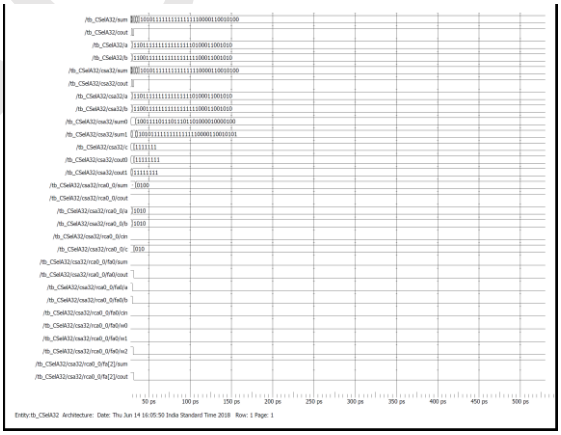

Figure 9: Simulation waveform for carry save adder

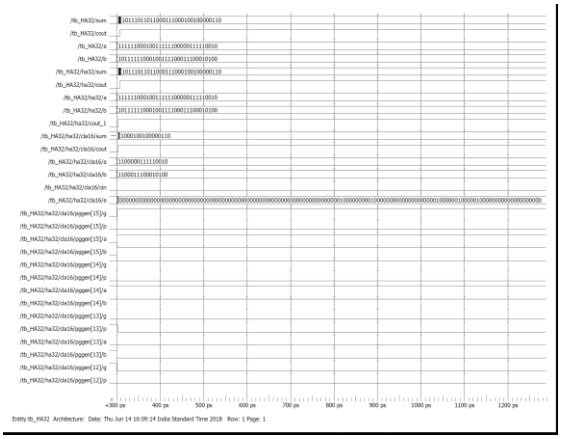

Figure 10: Simulation waveform for HA32

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 10, Issue 06, January 2022)

Table 1: Device utilization summary

Device utilization summary	32 bit CLA Adder[10]	32 bit RCA Adder	32 bit CSA Adder
Number of slices	110	123	105
Number of LUT Flip Flop pairs used	192	195	195
Number of bonded IOBs:	227	238	153
Time	23.47 ns	26.43 ns	14.34 ns

Table 2: Device utilization summary

Device utilization summary	32 bit HCSA Adder	32 bit CSA Adder
Number of slices	94	105
Number of LUT Flip Flop pairs used	96	195
Number of bonded IOBs:	148	153
Time	3.65 ns	14.34 ns

Table 1 and table 2 demonstrate the slice used in 32 bit adder architecture where CLA consumes 110 slice ,RCA has maximum number of slice consumption that is 123.CSA architecture consumes lesser then RCA and CLA which is 105.The proposed structure gives better results in terms of slice consumption in given FPGA resource i.e. 94.

V. CONCLUSION

Proposed hybrid Adder claims lesser area as compared with different types of adders are explored in this chapter. RCA, CSA, CLA The outcome of hierarchal CSA based adder has given better area and speed then traditional CSA. Minimum Number of slices 105 is achieve as compared with other architecture. The proposed work can be used in multiplier and MAC architecture in future work.

REFERENCE

[1] Javali, R. A., Nayak, R. J., Mhetar, A. M., & Lakkannavar, M. C. (2014, November). Design of high speed carry save adder using carry lookahead adder. In *International Conference on Circuits, Communication, Control and Computing* (pp. 33-36). IEEE.

[2] Singh, R. P. P., Kumar, P., & Singh, B. (2009). Performance analysis of 32-bit array multiplier with a carry save adder and with a carry-look-ahead adder. *International Journal of Recent Trends in Engineering*, *2*(6), 83.

[3] Erniyazov, S., & Jeon, J. C. (2019). Carry save adder and carry look ahead adder using inverter chain based coplanar QCA full adder for low energy dissipation. *Microelectronic Engineering*, *211*, 37-43.

[4] Vamsi, A. K., Kumar, N. U., Sindhuri, K. B., & Teja, G. S. C. (2018, December). A systematic delay and power dominant carry save adder design. In *2018 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 359-362). IEEE.

[5] Priyadarshini, K. M., Ravindran, R. E., & Nanda, I. (2020). A novel two level edge activated carry save adder for high speed processors. *International Journal of Advanced Computer Science and Applications*, *11*(4), 487-493.

[6] Mendez, T., & Nayak, S. G. (2022). Design and Analysis of an Iterative Carry Save Adder-based Power-Efficient Multiplier. *Iranian Journal of Electrical and Electronic Engineering*, *18*(1), 2238-2238.

[7] Hameed, A. S., & Kathem, M. J. (2021). High speed modified carry save adder using a structure of multiplexers. *International Journal of Electrical and Computer Engineering*, *11*(2), 1591.

[8] Javali, R. A., Nayak, R. J., Mhetar, A. M., & Lakkannavar, M. C. (2014, November). Design of high speed carry save adder using carry lookahead adder. In *International Conference on Circuits, Communication, Control and Computing* (pp. 33-36). IEEE.

[9] Bennet, B., & Maflin, S. (2015, March). Modified energy efficient carry save adder. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]* (pp. 1-4). IEEE.

[10] Mitra, A., Bakshi, A., Sharma, B., & Didwania, N. (2015). Design of a High Speed Adder. *International Journal of Scientific & Engineering Research*, *6*(4), 918-921.