International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

A Carry Save Adder Design

S Subha

Abstract: ***Full adders add three input numbers and give sum and carryout. The full adders which are faster and energy saving is the need of the hour. Simplifications of the Boolean expressions for addition of two n-bit numbers is one method to achieve this. Other method is to use combinational circuits. Various full adders are proposed in literature. Carry save adder is one of them. They are used in many places. The sum and carry of two 32-bit numbers are calculated in this paper. The sum and carry of inputs is calculated as in carry save adder. The carry is added to the sum based on the values of sum bit resulting in final sum and carry. The proposed logic is simulated using Quarturs 2 toolkit. An improvement in area by 16% with time improvement of 8.56% and comparable power consumption is observed for chosen parameters when compared with model proposed in [5].***

Index Terms: ***Area, Boolean logic, Carry save adder, Full adder, Performance***

I. INTRODUCTION

The main parts of computer are memory, central processing unit (CPU) and input/output devices. The arithmetic and logic unit (ALU) and control unit (CU) are parts of the CPU. The arithmetic and logic unit has arithmetic and logic circuits. The arithmetic unit contains integer and floating point units. The four basic arithmetic operations supported by ALU are addition, subtraction, multiplication and division [6]. Usually two's complement addition/subtraction is used for addition and subtraction. Full adder circuits with three inputs and two outputs are used for integer addition. Examples of full adder circuits proposed in literature are ripple carry adder, carry save adder, carry select adder, carry skip adder, parallel prefix adder [1, 2]. In carry save adder, two n-bit numbers are added to give sum and carry. The sum and carry of individual bits are calculated separately. The result is added according to their weights to give the result. This model is called Trad model in this paper. Wallace tree is one method to implement carry save adders [2]. A carry save model with improved area, power and timing is proposed in [4]. This model is called CAS1 in this paper. A method using universal gates to realize carry save adder is proposed in [5]. This model is called CAS2 in this paper. For the CAS1 model an improvement in area by 27% with power saving of 8.5% with timing improvement of 6.8% compared with traditional model Trad is observed [4]. For the CAS2 model, an improvement in area by 33% over Trad, 8.16% over CAS1 model, power saving of 8.65% over Trad model and comparable power consumption with CAS1 model is reported [5]. A timing degradation of 8% over Trad model and 17% over CAS1 model is observed for CAS2 model.

This paper proposes method to perform addition using carry save adder. The input is two 32 bit numbers. The sum and carry of inputs is calculated as in carry save adder. The final answer sum and carry is calculated based on the value of the calculated sum in step 1 and one of the inputs at each stage using modified equations of the sum and carry of the full adder. The proposed model is simulated using Quartus2 Toolkit. An improvement in area by 16% with performance improvement of 8.56% and comparable power consumption is observed for chosen parameters compared with CAS2 model.

Manuscript published on 30 June 2019.
***** Correspondence Author (s)
S Subha, Department of IT and Engineering, SITE, Vellore Institute of Technology, Vellore, T.Nadu, India

II. MATHEMATICAL BACKGROUND

For any full adder circuit given in Table 1 the sum is given as the XOR of the inputs. The carryout is ab or bc or ca for inputs a, b, c [3].

Table I: Full Adder Truth Table

A	B	C	Sum	Carryout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The sum and carry of inputs are calculated as in carry save adder. The carry is added to the sum based on the following logic. Consider two two bit numbers a and b with carryin c. Let a be a[1]a[0] and b be b[1]b[0].

1. Initialize sum to one of inputs say b i.e. sum[1] = b[1] sum [0] =b[0].
2. Initialize carry as sum. Thus c[0] = sum[0] and c[1] = sum[1].
3. If a[0] = 1, sum[0] = !sum[0]. If a[1] = 1, sum[1] = !sum[1].
4. If a[0] = 0 carry[0] = 0. If a[1] = 0 carry[1] = 0.
5. If sum[0] = 0, then sum[0] = c and carry[0] = a[0]
6. If sum[0] = 1 then sum[0] = !c and carry[0] = c
7. Make c = carry[0]. Perform steps 5-6 for bit-1
8. Make cout = carry[1]
9. Stop.

The calculations take advantage of the value of sum bit in i-th stage and calculate the correct value using the carryin. The carryout is made equal to input a if the sum is zero and equal to carryin otherwise.

III. PROPOSED MODEL

Consider the full adder truth table given in Table 1. The carry save adder design 2 is given next.

Algorithm **carry_save_adder_design2** : Given two n-bit numbers this algorithm finds the sum of two numbers. The numbers are stored in array from index 0 to n-1.

1. Start
2. The sum is one of the inputs say b
3. The carry is sum
4. The sum is calculated based on other input a. If other input is one the sum is negated else it is b
5. If the other input a is zero, the carry is set to zero else it is b i.e. sum in step 1. The cin is the carryin in least significant bit (LSB). It is zero initially. For the LSB the logic is

if (sum == 1'b0) sum = cin and cout = a[0]
else if (sum == 1'b1) sum = !cin; cout = cin;

6. For all other bits, cin is the carry out of previous bit.

 The logic for these bits is given next.

if (sum == 1'b0) sum = cin and cout = a[i]
else if (sum == 1'b1) sum = !cin; cout = cin;

7. The final carryout is calculated as follows.

if (cin == 1'b0) then cout = carryout(n-1) else cout = !carryout (n-1)

8. Stop

In the proposed model, the XOR equation of the sum and the carryout equation for full adder are calculated by initializing the sum to one of the inputs and carry to one of the inputs. The final sum is based on the carryin value at each bit and final carryout is based on the carryin value. This is the logic given in the proposed algorithm.

Example: Consider two four bit numbers a= 1010 and b=1101. The steps in the algorithm are given below to calculate the sum of a and b.

1. Sum = b = 1101.
2. Carry = sum = 1101
3. Inspect a. If a[i] is one the sum[i] is negated. A array is denoted by a[3..0]. We find that a[3] and a[1] is one.. Hence the sum is 0111.
4. If a[i] is zero, carry is zero else it is b. We find that a[0] and a[2] are zero. Hence carry is set to 1000
5. For cin in position zero, it is zero. From step 3 sum[0] = 1. Hence sum[0] =! cin = 1 and cout = a[0] = 0
6. For i=1, sum[1] = 1. The cin = 0. Hence sum[1] = !cin = 1 and cout = cin = 0.
7. For I = 2 sum[2] = 1, cin = 0. Hence sum = !cin = 1 and cout = cin = 0.
8. For i=3 sum[3] = 0 cin = 0. Hence sum = cin = 0 and cout = a[3] = 1.
9. The final answer is 10111.

IV. SIMULATIONS

The proposed model in section 3 is simulated using Quartus 2 Toolkit. The input is two 32 bit numbers. The model is compared with CAS2. The simulation parameters are given in Table 2.

Table II Simulation parameters

Parameter	**Value**
Processor family	Cyclone II
Package	FBGA
Pin count	484
Speed grade	Fastest
Target Device	Auto select by Fitter

The simulation results are shown in Table 3

Table III Simulation Results

paramete	prop	cas2	%improve
area	75/14448	90/14448	16.66667
timing	8.173ns	8.939ns	8.569191
power	78.01mW	78.27mW	0.332183

As seen from Table 3 an area improvement of 16% with timing improvement of 8.56% and comparable power is observed for the proposed model compared with CAS2 model.

V. CONCLUSION

A carry save adder with improved area and performance is presented in this paper. The proposed model calculates the carry and sum of inputs as in carry save adder. The final sum and carry are calculated using the calculated sum value and carry in value. The proposed model reduces the number of Boolean logic gates compared with model presented in CAS2. The proposed model is compared with model CAS2. An area improvement of 16% with 8.56% timing improvement and comparable power is observed for the proposed model compared with CAS2 model.

REFERENCES

[1] Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, 2000
[2] Israel Koren, Computer Arithmetic Algorithms, Prentice Hall, NJ 1993
[3] Morris Mano, Digital Logic and Computer Design, Prentice Hall, 1979
[4] S Subha, An Improved Carry Save adder design, International Journal of Advanced Science and Research (IJASR), Vol. 3, No. 1, January 2018, pp. 01-02
[5] S Subha, A Power Saving Carry Save Algorithm, Proceedings of ICICES 2018(IEEE to be published IEEE conference ?), March 21-22, 2018
[6] William Stallings, Computer Organization and Architecture Eight Edition, Pearson Prentice Hall, 2010

AUTHORS PROFILE

S. Subha has done her Ph.D in computer Engineering in caches from Santa Clara University, CA, USA. Her research interests are in processor cache memories, computerarithmetic. She has teaching experience of seventeen years, software industry experience of six years. She is presently working in Vellore Institute of Technology, Vellore, India. She has successfully guided /co-guided four research scholars in area of computer architecture, parallel processing, cloud computing at VIT, Vellore. She has authored/co-authored fifty journal papers in international journals, thirty nine international conference publications. She has worked as reviewer of international journals for past five years