Ripple Carry Adder and Subtractor

Ramu Pittala
EE18MTECH11027

Aman Agarwal
EE18MTECH11008

April 26, 2019

Contents

1 Introduction

In this project, we are asked to perform the ripple carry addition and subtraction in Verilog using raspberry pi and icoBoard. Hence, we are receiving input bits from Keypad through Arduino and displaying the output in 16 x 2 LCD. A ripple carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded with the carry output from each full adder connected to the carry input of the next full adder in the chain. It is suitable for small bit applications. A ripple carry subtractor is implemented by taking 2s complement of second number and adding to the first number using same ripple carry adder. If the carry turns out to be one in this case, the carry is discarded and the result is a positive number but if the carry turns out to be zero, the final result is the 2s complement of the current result which will always be a negative number.

2 Components

1. Hardware
A) IcoBoard B) Raspberry Pi C) Arduino-Uno D) LCD E) Keypad F) Breadboard G) SD Card H) Potentiometer I) Connecting Wires
2. Software
A) Raspbian OS B) Arduino IDE

3 Theory

1. One-bit full adder: A one-bit full adder is a combinational circuit that forms the arithmetic sum of three bits. It consists of three inputs a, b and C_{in} and two outputs S and C_{out} as illustrated in Fig 1.

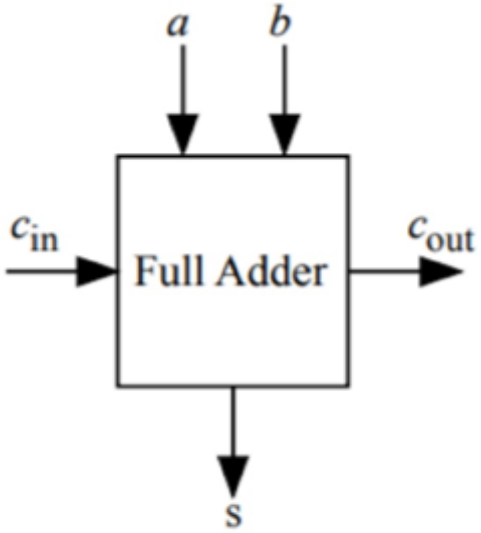


Figure 1: One-bit full adder.

2. The truth table of the full adder is listed in Table 1. The gate implementation of 1-bit full adder is shown in Figure 2. 3. Ripple Carry Adder-Subtractor: A ripple

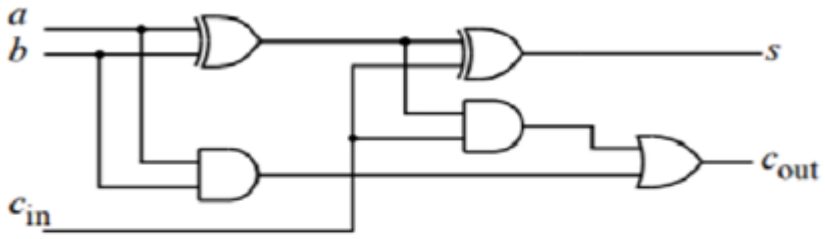


Figure 2: Gate implementation of full adder.

Table 1: Full adder truth table.

a	b	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

carry adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascaded with the carry output from each full adder connected to the carry input of the next full adder in the chain. Figure 3 shows the interconnection of four full adder (FA) circuits to provide a 4-bit ripple carry adder. Notice from Figure 3 that the

input is from the right side because the first cell traditionally represents the least significant bit (LSB). Bits a_0 and b_0 in the figure represent the least significant bits of the numbers to be added. The sum output is represented by the bits $s_0 - s_3$.

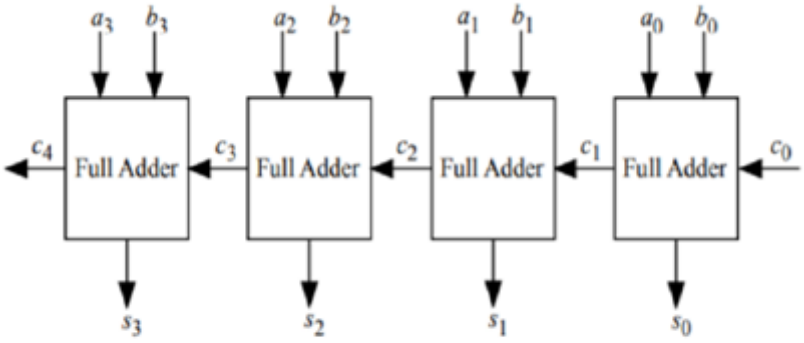
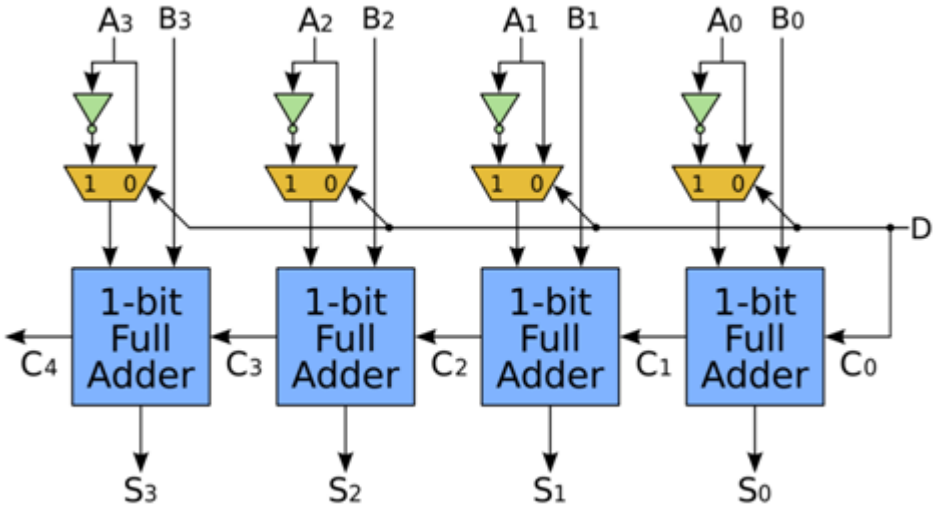


Figure 3: 4-bit full adder.

Clearly, here $c_0 = 0$ for performing 4-bit addition and for performing 4-bit subtraction, $c_0 = 1$ with one input as b_0, b_1, b_2, b_3 and another input as output of the NOT gate when each of a_0, a_1, a_2 and a_3 is passed through it. This is also called 2s complement addition. If carry c_4 turns out to be 1, result is a positive number and this carry c_4 is discarded. But if c_4 is 0, the final result is the 2s complement of the current result which will be a negative number. Hence both the addition and subtraction operation can be viewed in same figure as shown below:



4 Algorithm

1. Addition :
S[0] = $X[0]\hat{}Y[0]$;
b5 = $X[0]\&Y[0]$;
w1 = $X[1]\hat{}Y[1]$;
S[1] = $w1\hat{}b5$;
w2 = $w1\&b5$;
w3 = $X[1]\&Y[1]$;
b6 = $w2|w3$;
w4 = $X[2]\hat{}Y[2]$;
S[2] = $w4\hat{}b6$;
w5 = $w4\&b6$;
w6 = $X[2]\&Y[2]$;
b2 = $w5|w6$;
w7 = $X[3]\hat{}Y[3]$;
S[3] = $w7\hat{}b2$;
w8 = $w7\&b2$;
w9 = $X[3]\&Y[3]$;
V = $w8|w9$;

2. Subtraction:
h0= $Y[0]$;
b0= $X[0]\hat{}h0$;
b9= $1'b1$;
A0= $b9\hat{}b0$;
b5= $X[0]\&h0$;
b6= $b9\&b0$;
b7= $b5|b6$;

h1= $\sim Y[1]$;
w1= $X[1]\hat{}h1$;
A1= $w1\hat{}b7$;
w2= $w1\&b7$;
w3= $X[1]\&h1$;
b2= $w2|w3$;

h2= $\sim Y[2]$;
w4= $X[2]\hat{}h2$;
A2= $w4\hat{}b2$;
w5= $w4\&b2$;
w6= $X[2]\&h2$;
b3= w5—w6;

h3= $\sim Y[3]$;
w7= $X[3]\hat{}h3$;
A3= $w7\hat{}b3$;
w8= $w7\&b3$;
w9= $X[3]\&h3$;
b8= $w8|w9$;

if($b8 == 1'b0$)
begin

D0 = $\sim A0$;
D1 = $\sim A1$;
D2 = $\sim A2$;
D3 = $\sim A3$;
S[0] = $b9 \hat{} D0$;
c1 = $b9 \& D0$;
S[1] = $c1 \hat{} D1$;
c2 = $c1 \& D1$;
S[2] = $c2 \hat{} D2$;
c3 = $c2 \& D2$;
S[3]= $c3 \hat{} D3$;

end

else if($b8 == 1'b1$)
begin
S[0] = A0;
S[1] = A1;
S[2] = A2;
S[3] = A3;
end

5 Conlusion

Addition and Subtraction operation was implemented successfully.