EE210 - Digital System Design Lab Report-1

Adarsh Santoria
B21176

***Abstract*—This report demonstrates the operation of 4-bit Ripple Carry Adder (RCA) and Subtractor. Addition is carried out by Digital Logic Gates, whereas Subtraction is simulating addition of two binary numbers using the concept of Twos Complement . Verilog language, which is a Hardware Descriptive Language (HDL) , is used to describe the low level hardware, on Xilinx Vivado.**

I. Introduction

A N-bit Ripple Carry Adder is a digital circuit which is used to add two N-bit numbers plus a carry-in bit, resulting in an N-bit sum and a carry-out bit. 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. Let's first examine each module one by one.

A. Full Adder

Half adder is a combinational logic circuit with two inputs and two outputs. The half adder circuit is designed to add two single bit binary number A and B. It is the basic building block for addition of two single bit numbers. Full adder is developed to overcome the drawback of Half Adder circuit. It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit. The full adder circuit and it's truth table for is shown in Fig 1 and Fig 2 respectively. From the truth table, the equation for sum 'S' and carry 'Cout' comes out as follows:

$$S = a \bigoplus b \bigoplus c \tag{1}$$

and

$$C = a \cdot b + b \cdot c + c \cdot a. \tag{2}$$

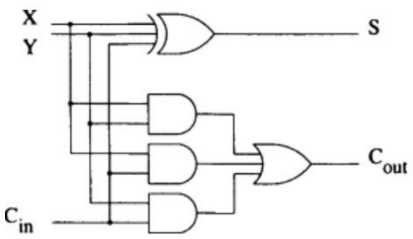


Fig. 1. Logic circuit of Full Adder

B. Full Subtractor

Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit. The disadvantage of a half subtractor is overcome by full subtractor. The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output. The full subtractor circuit and it's truth table for is shown in Fig 3 and Fig 4 respectively. From the truth table, the equation for difference 'D' and borrow 'Bo' comes out as follows:

$$D = a \bigoplus b \bigoplus c \tag{3}$$

and

$$Bo = \overline{a} \cdot b + c \cdot \overline{(a \bigoplus b)}. \tag{4}$$

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
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

Fig. 2. Truth Table of Full Adder

Inputs			Outputs	
A	B	$Borrow_{in}$	Diff	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fig. 4. Truth Table of Full Subractor

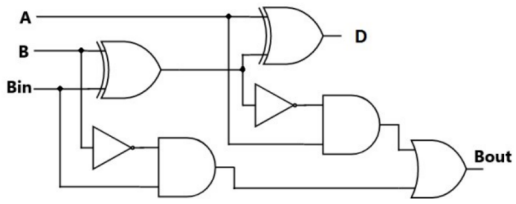


Fig. 3. Logic circuit of Full Subtractor

C. *Ripple Carry Adder cum Subtractor*

The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder. When Cin = 0, the circuit is an adder, and when Cin = 1, the circuit becomes a subtractor. Each exclusive-OR gate receives input Cin and one of the inputs of B. This can be derived as A xor 0 is A itself where A xor 1 is A's compliment. Fig. 3 shows us how a 4 bit RCA circuit can be designed using full adder modules. Formula to get a better view:

$$b \bigoplus c \quad = \quad \overline{b} \cdot c + b \cdot \overline{c} \tag{5}$$

Substraction operation is carried out with the following strategy:

$$A - B \quad = \quad A + (-B) \tag{6}$$

or

$$A - B \quad = \quad A + 2's complement of B \tag{7}$$

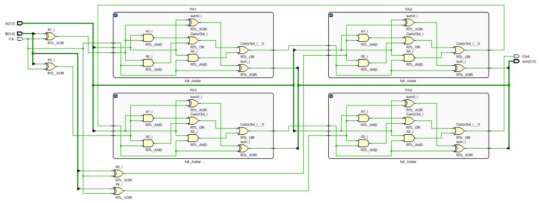
or

$$A - B \quad = \quad A + 1's complement of B + 1 \tag{8}$$

II. SYNTHESIS AND SIMULATIONS

REFERENCES

[1] Schematic Diagram of 4-bit Ripple Carry Adder cum Subtractor



[2] Test Bench Waveform

[3] Design Source Code

```
`timescale 1ns / 1ps

module full_Adder(sum , Cout , a ,b, Cin);
output sum , Cout ;
input a , b, Cin;
wire w1 ,w2 ,w3 ;
assign sum = a^b^Cin;
and A1(w1,a,b);
and A2(w2,b,Cin);
and A3(w3,a,Cin);
or CarryOut( Cout,w1 , w2 ,w3 );
endmodule

module Bit_ACS(sum,Cout,A,B,Cin);
output [3:0] sum  ;
output Cout ;
input [3:0] A,B;
input Cin;
wire w1,w2,w3,w4,w5,w6,w7;
xor X1(w4,B[0],Cin);
xor X2(w5,B[1],Cin);
xor X3(w6,B[2],Cin);
xor X4(w7,B[3],Cin);

full_Adder FA1( sum[0], w1 , A[0] , w4, Cin);
full_Adder FA2( sum[1], w2 , A[1] , w5 , w1);
full_Adder FA3( sum[2], w3 , A[2] , w6 , w2);
full_Adder FA4( sum[3] ,Cout , A[3], w7 , w3);
endmodule
```

[4] Test Bench Code

```
`timescale 1ns / 1ps

module Bit_ACS_tb();
reg [3:0]A1;
reg [3:0]B1;
reg Cin1;
wire [3:0]sum1;
wire Cout1;

Bit_ACS bit_ACS( sum1 ,Cout1,A1,B1,Cin1);
initial begin
A1[0]=1;A1[1]=0;A1[2]=0;A1[3]=0;B1[0]=1;B1[1]=0;B1[2]=0;B1[3]=0;Cin1=0;
#10
A1[0]=1;A1[1]=0;A1[2]=0;A1[3]=1;B1[0]=1;B1[1]=0;B1[2]=0;B1[3]=1;Cin1=0;
#10
A1[0]=1;A1[1]=0;A1[2]=1;A1[3]=1;B1[0]=0;B1[1]=0;B1[2]=1;B1[3]=1;Cin1=0;
#10
A1[0]=0;A1[1]=1;A1[2]=1;A1[3]=0;B1[0]=0;B1[1]=1;B1[2]=1;B1[3]=0;Cin1=0;
#10
A1[0]=1;A1[1]=1;A1[2]=1;A1[3]=1;B1[0]=1;B1[1]=1;B1[2]=1;B1[3]=1;Cin1=0;
#10
A1[0]=1;A1[1]=0;A1[2]=0;A1[3]=0;B1[0]=1;B1[1]=0;B1[2]=0;B1[3]=0;Cin1=1;
#10
A1[0]=1;A1[1]=0;A1[2]=0;A1[3]=1;B1[0]=1;B1[1]=0;B1[2]=0;B1[3]=1;Cin1=1;
#10
A1[0]=1;A1[1]=0;A1[2]=1;A1[3]=1;B1[0]=0;B1[1]=0;B1[2]=1;B1[3]=1;Cin1=1;
#10
A1[0]=0;A1[1]=1;A1[2]=1;A1[3]=0;B1[0]=0;B1[1]=1;B1[2]=1;B1[3]=0;Cin1=1;
#10
A1[0]=1;A1[1]=1;A1[2]=1;A1[3]=1;B1[0]=1;B1[1]=1;B1[2]=1;B1[3]=1;Cin1=1;
#10
$stop;
end
endmodule
```

III. CONCLUSION

We can make N-bit Adder and Subtractor using N 1-bit Adder and Subtractor respectively using some sequential order of bits manipulation or gates.

REFERENCES

[1] Digital Design by Morris Mano