Compter Architecture Organization - CECS 341

LAB 2: Logic Block Design California State University, Long Beach

Group Members:
1. Katherine Lazo
2. Nguyen Vo
3. Diego Garcia
4. Shawn Anthony

Full Adder 1-bit: An adder that takes in three inputs: A(1-bit), B(1-bit), Carry In(1-bit). It produces two outputs: Sum(1-bit), and Carry Out(1-bit).

Sum = A XOR B XOR Cin

Cout = (A AND B) OR (A AND Cin) OR (B AND Cin)

Full Adder –Truth Table				
Input			Output	
A	B	Carry 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

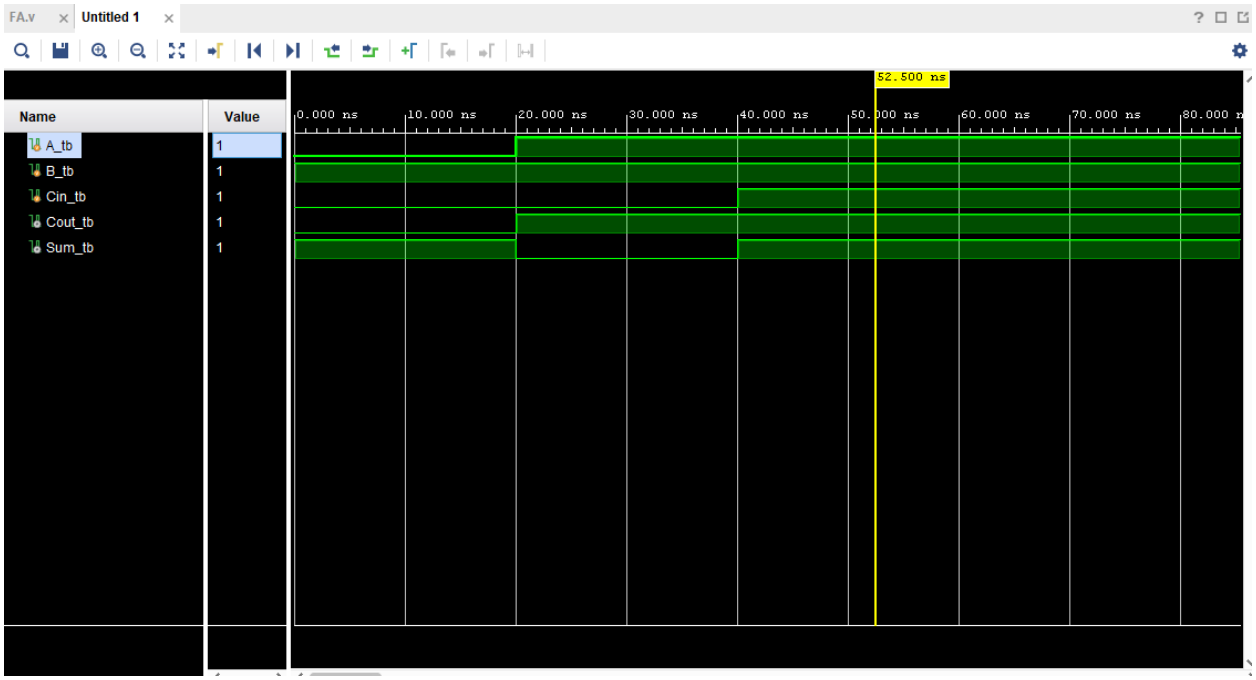
Test Cases:

Test1 (Run for 20ns): A=”0” , B = ”1”, Cin = ’0’

Test2 (Run for 20ns): A=”1” , B = ”1”, Cin = ’0’

Test3 (Run for 20ns): A=”1” , B = ”1”, Cin = ’1’

Waveform:



Conclusion: A 1-bit adder takes in three inputs mentioned above. A represents the first number, B represents the second number, and Cin is the Carry in if there is any. If A, B, or Cin have a value of 1, then Cout, the Carry Out, would result in a 1.

Full Adder 4-bit: An adder that takes in three inputs: A(4-bits), B(4-bits), and Carry In(1-bit). It produces two outputs: Sum(4-bits), and Carry Out(1-bit).

Sum = A + B + Cin

Cout = A + B + Cin

Test Cases:

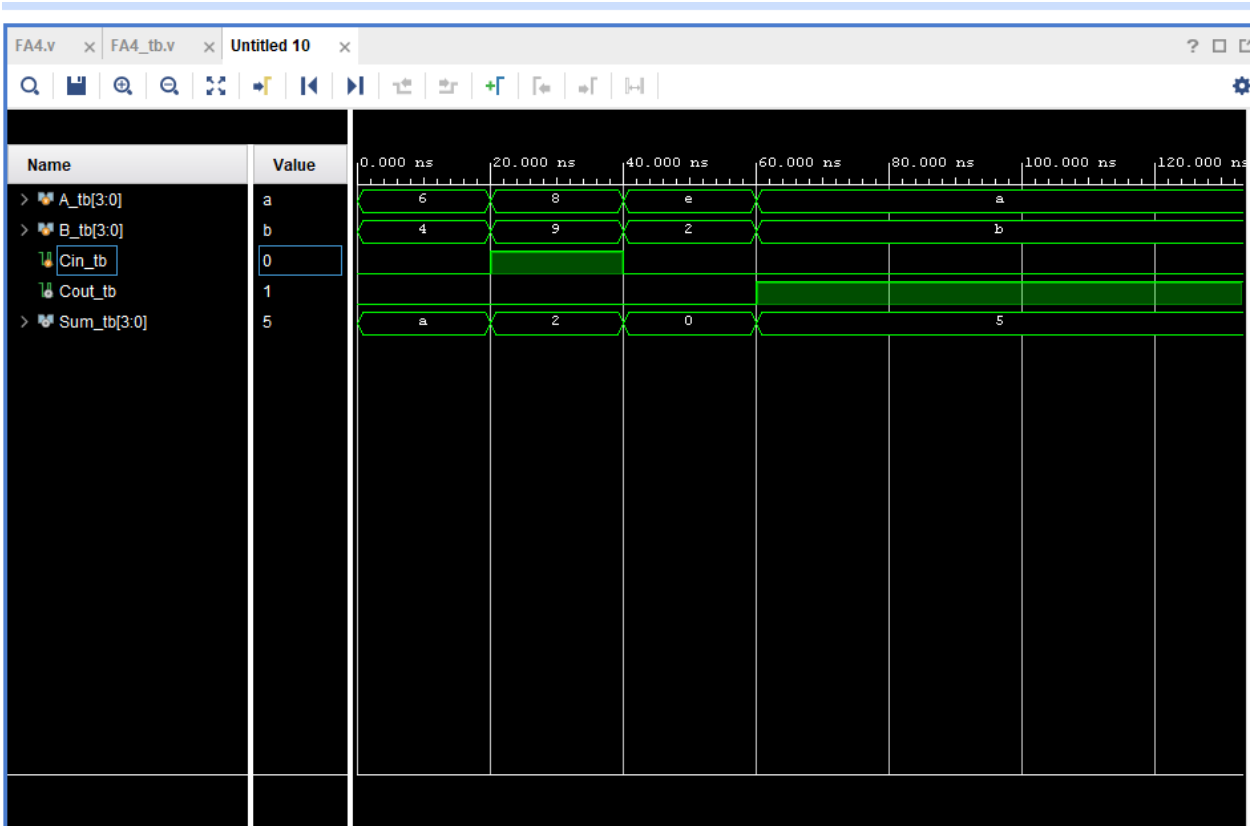
Test1 (Run for 20ns): A=”0110” , B = ”0100”, Cin = ’0’

Test2 (Run for 20ns): A=”1000” , B = ”1001”, Cin = ’1’

Test3 (Run for 20ns): A=”1110” , B = ”0010”, Cin = ’0’

Test4 (Run for 20ns): A=”1010” , B = ”1011”, Cin = ’0’

Waveform:



Conclusion: A 4-bit adder takes in three inputs mentioned above. Similarly to the 1-bit adder, the 4-bit adder adds A,B, and Cin. However, A and B are now in 4-bits instead of one bit. Cin still remains with 1-bit. A represents the first number, B represents the second number, and Cin is the Carry in if there is any. If A, B, or Cin have a value of 1, then Cout, the Carry Out, would result in a 1.

Multiplexer 2:1:

A 2:1 multiplexer has two (1-bit) data inputs, D0 and D1, and one selector (S) input that chooses between them. If S is 0, D0 is the output. If S is 1, D1 is the output. The logic for output:

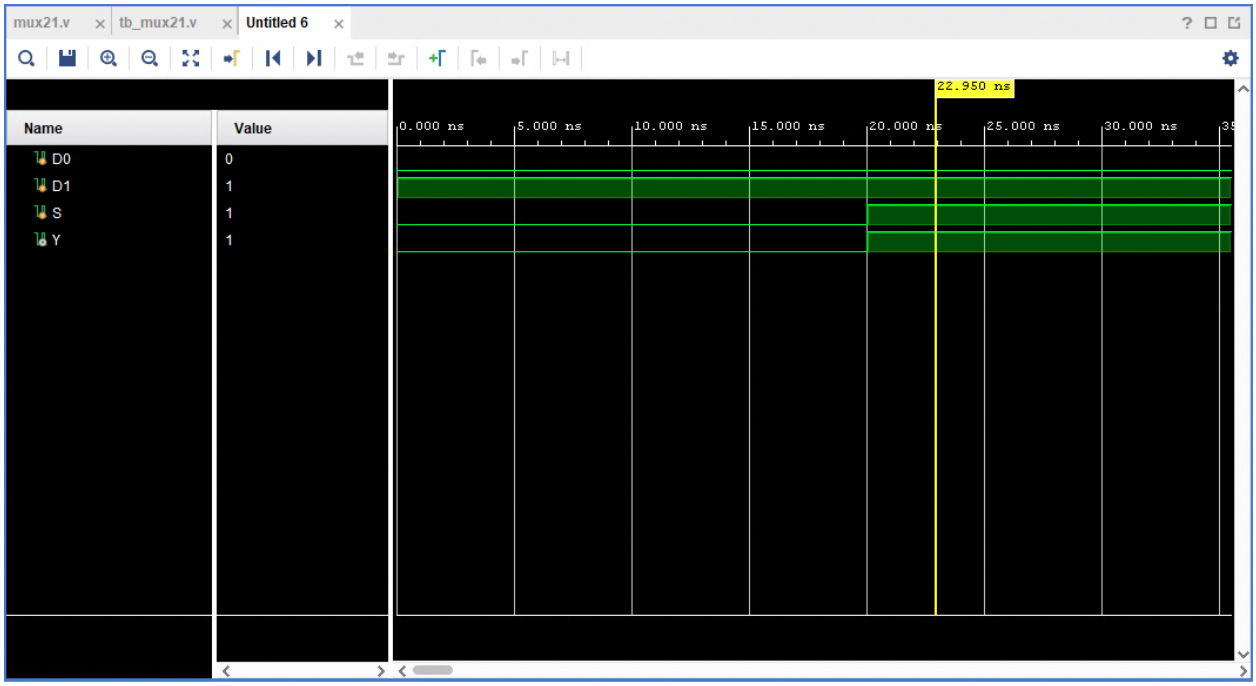
Y = (!S & D0) | (S & D1)

Test Cases:

Test1 (Run for 20ns): D0=’0’ , D1 = ’1’, S = ”0”

Test2 (Run for 20ns): D0=’0’ , D1 = ’1’, S = ”1”

Waveform:



Conclusion:

The mux takes the three inputs (including the selector) and produces the correct output based on the logic:

test1 (Run for 20ns): D0=’0’ , D1 = ’1’, S = ”0”

Y=1

test2 (Run for 20ns): D0=’0’ , D1 = ’1’, S = ”1”

Y=1

MUX 2:1	Truth Table		
S	D0	D1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1