Lab 05
Adders/Subtractor

ECE 380-002
University of Alabama

Yichen Huang
Thomas Dillman

2019/09/30

Introduction

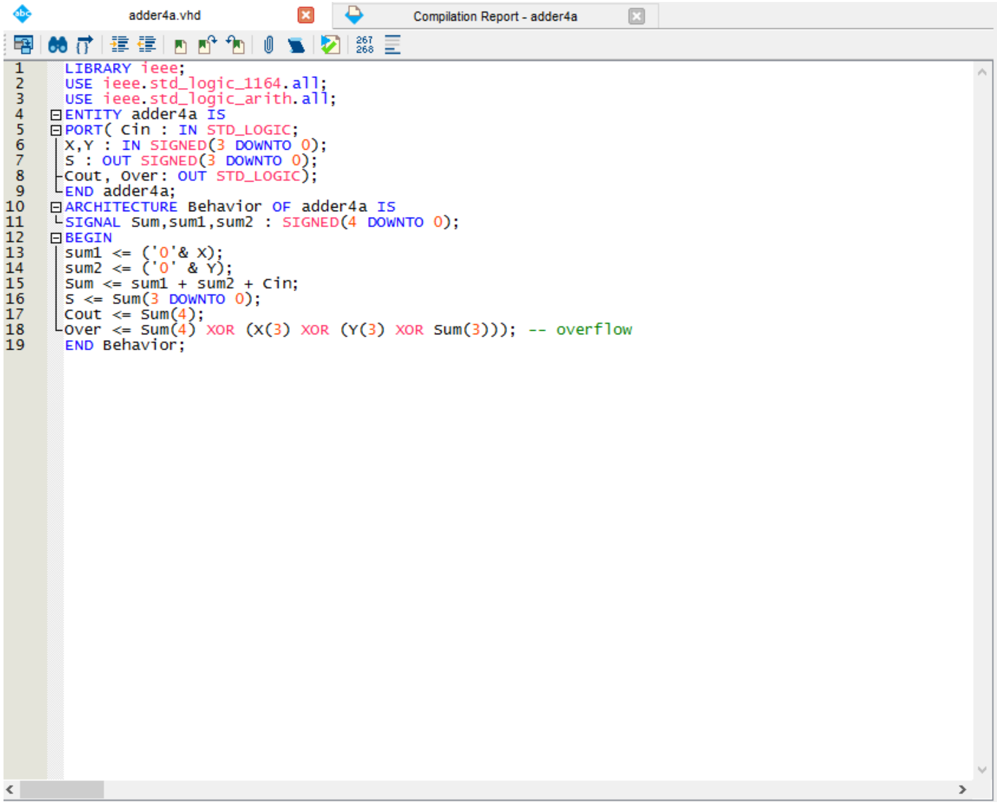
In this lab, we used the Quartus II software package to design and simulate behavioral adder, ripple carry adder, and LPM_ADD_SUB implementation. We also printed the Mega function schematic or VHDL files for Designs A, B, and C.

Procedure

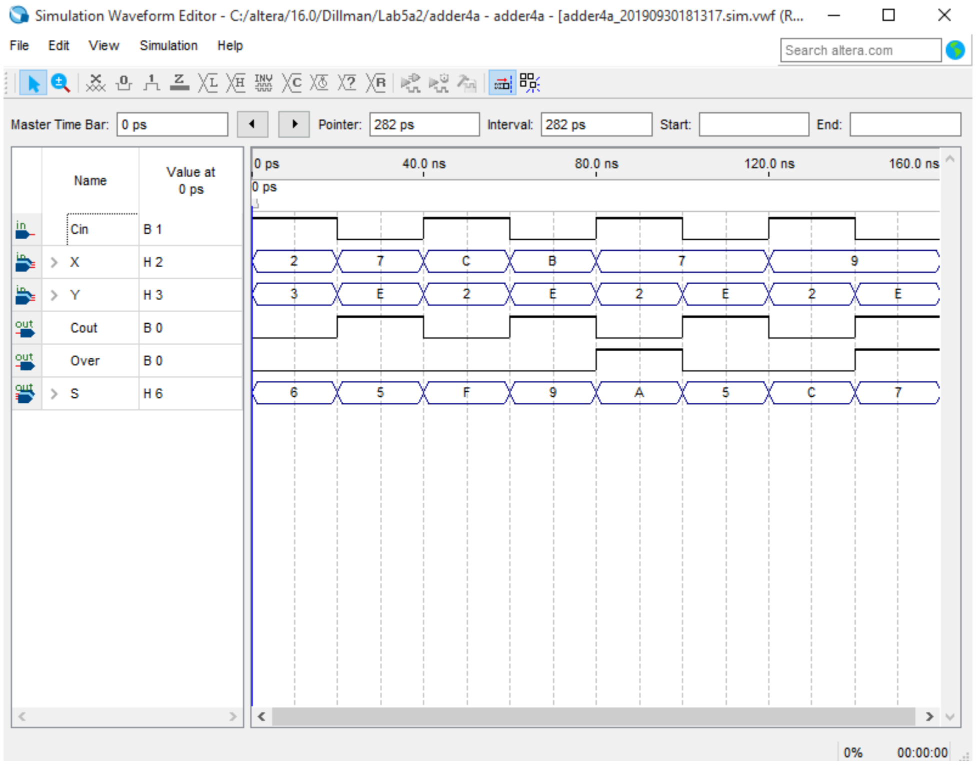
A. Prelab

a. Design A

In the design A, we created a project in the software and make a VHDL file called adder4a.vhd. We type the dataflow VHDL in the file. Then we run the compile, after compiling successfully, we run the test file and run different case in waveform file.



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
ENTITY adder4a IS
PORT( Cin : IN STD_LOGIC;
X,Y : IN SIGNED(3 DOWNTO 0);
S : OUT SIGNED(3 DOWNTO 0);
Cout, Over: OUT STD_LOGIC);
END adder4a;
ARCHITECTURE Behavior OF adder4a IS
SIGNAL Sum,sum1,sum2 : SIGNED(4 DOWNTO 0);
BEGIN
sum1 <= ('0'& X);
sum2 <= ('0' & Y);
Sum <= sum1 + sum2 + Cin;
S <= Sum(3 DOWNTO 0);
Cout <= Sum(4);
Over <= Sum(4) XOR (X(3) XOR (Y(3) XOR Sum(3))); -- overflow
END Behavior;
```

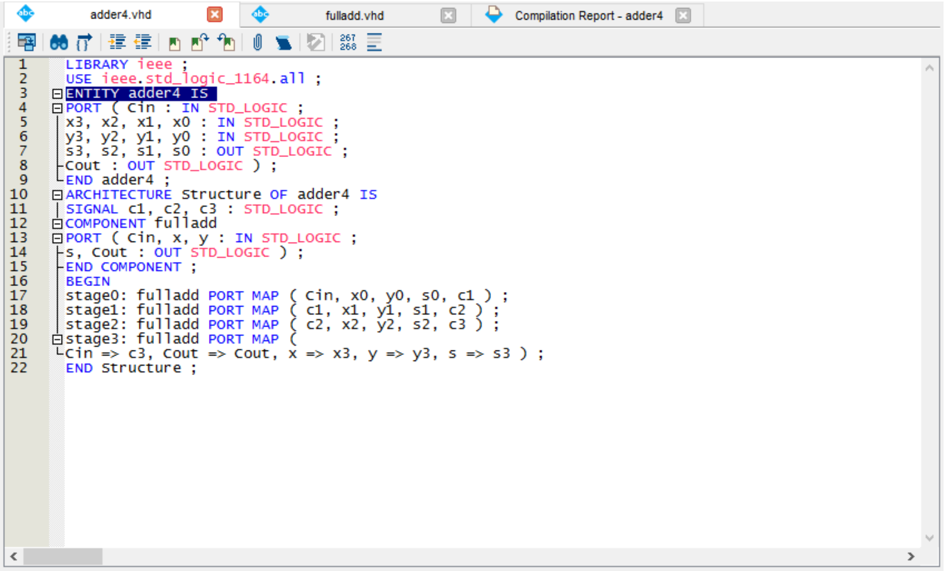

Simulation Waveform Editor - C:/altera/16.0/Dillman/Lab5a2/adder4a - adder4a - [adder4a_20190930181317.sim.vwf (R...

Master Time Bar: 0 ps Pointer: 282 ps Interval: 282 ps Start: End:

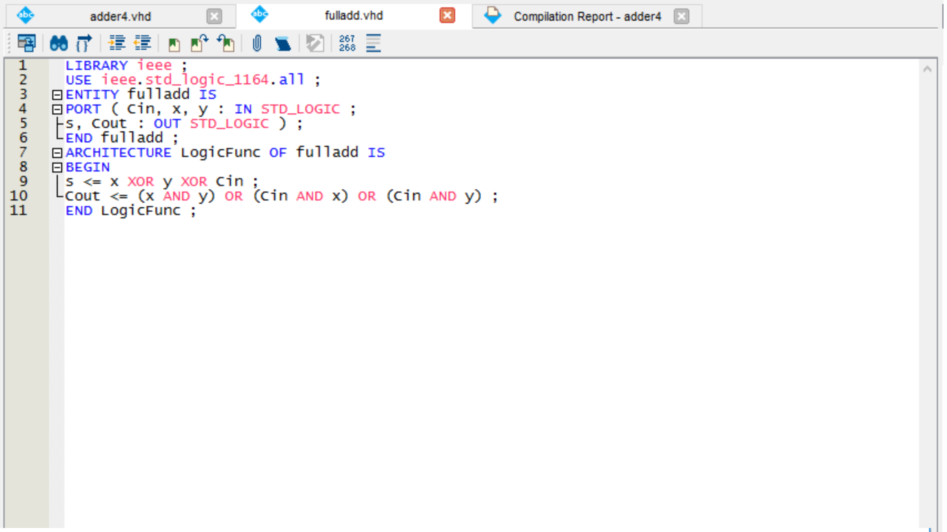
Name	Value at 0 ps
Cin	B 1
X	H 2
Y	H 3
Cout	B 0
Over	B 0
S	H 6

b. Design B

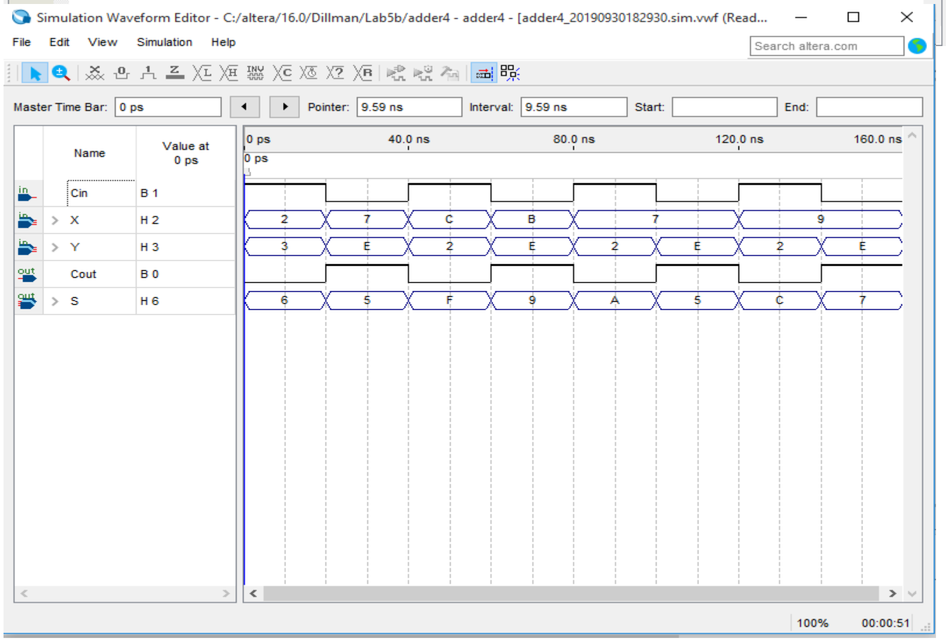
In the design B, we created a new project and implement a 4-bit ripple adder using structural VHDL. We created two files ‘fulladd.vhd’ and ‘adder4.vhd’. After implementation, we compile two files. Finally, we run the test file through waveform file. We group 4 single bits for input X, Y and output S.



```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY adder4 IS
PORT ( Cin : IN STD_LOGIC ;
x3, x2, x1, x0 : IN STD_LOGIC ;
y3, y2, y1, y0 : IN STD_LOGIC ;
s3, s2, s1, s0 : OUT STD_LOGIC ;
Cout : OUT STD_LOGIC ) ;
END adder4 ;
ARCHITECTURE Structure OF adder4 IS
SIGNAL c1, c2, c3 : STD_LOGIC ;
COMPONENT fulladd
PORT ( Cin, x, y : IN STD_LOGIC ;
s, Cout : OUT STD_LOGIC ) ;
END COMPONENT ;
BEGIN
stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
stage3: fulladd PORT MAP (
Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
END Structure ;
```



```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY fulladd IS
PORT ( Cin, x, y : IN STD_LOGIC ;
s, Cout : OUT STD_LOGIC ) ;
END fulladd ;
ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
s <= x XOR y XOR Cin ;
Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

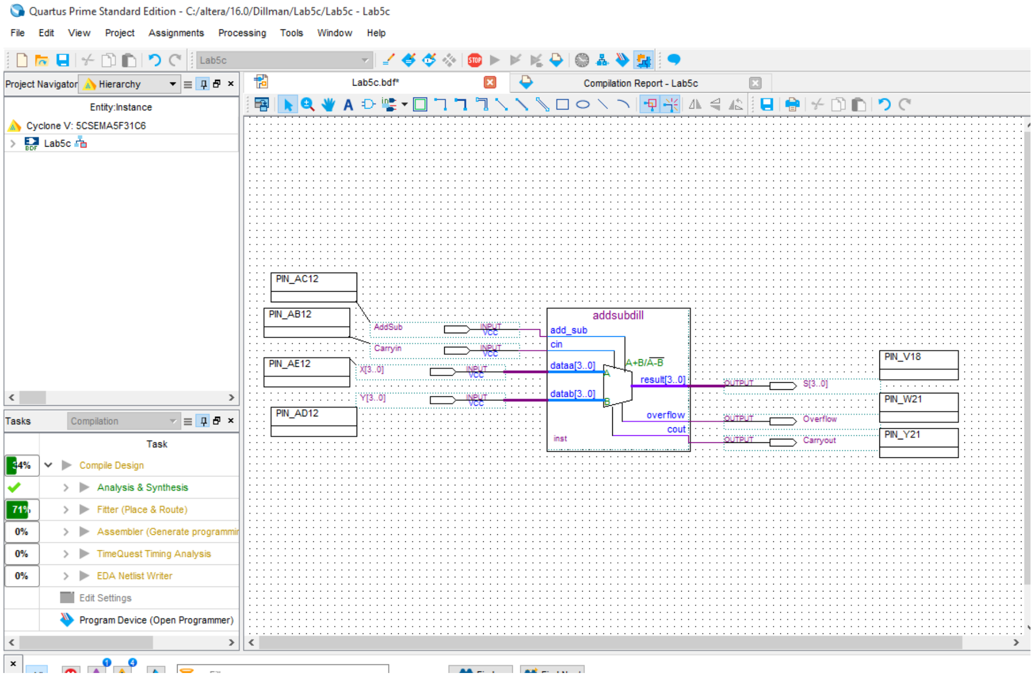

Simulation Waveform Editor - C:/altera/16.0/Dillman/Lab5b/adder4 - adder4 - [adder4_20190930182930.sim.vwf (Read...

Master Time Bar: 0 ps Pointer: 9.59 ns Interval: 9.59 ns Start: End:

Name	Value at 0 ps
Cin	B 1
X	H 2
Y	H 3
Cout	B 0
S	H 6

c. Design C

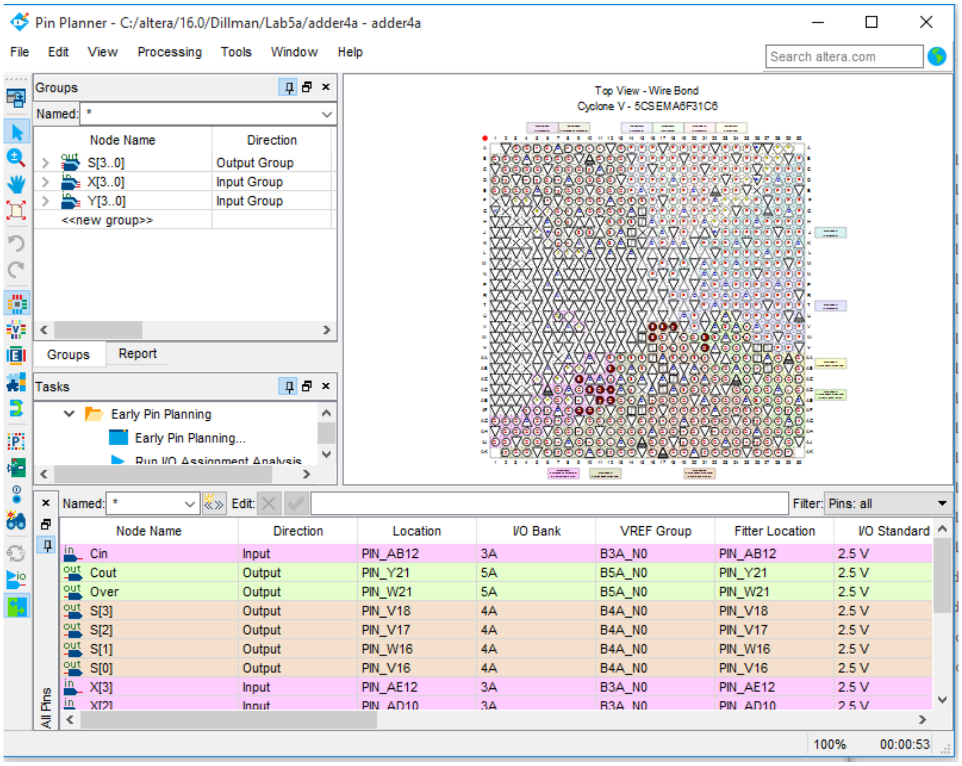
In the design C, we implement a 4-bit adder using the LPM_ADD_SUUB module in the schematic file ‘adder4c.bdf’. The circuit has 2 4-bits data input, a carry-in bit, a add or sub control input and a 4-bit data output, a carry-out output and an overflow output. After we finish the schematic, we run the compiling process.



B. During the lab

a. Design A

In the project of designA, we set the switches on the DE1 board. After setting, we recompile again. Then we upload the project to the DE1 board, after configuration, we test the case on the board again, the result is same in the prelab.

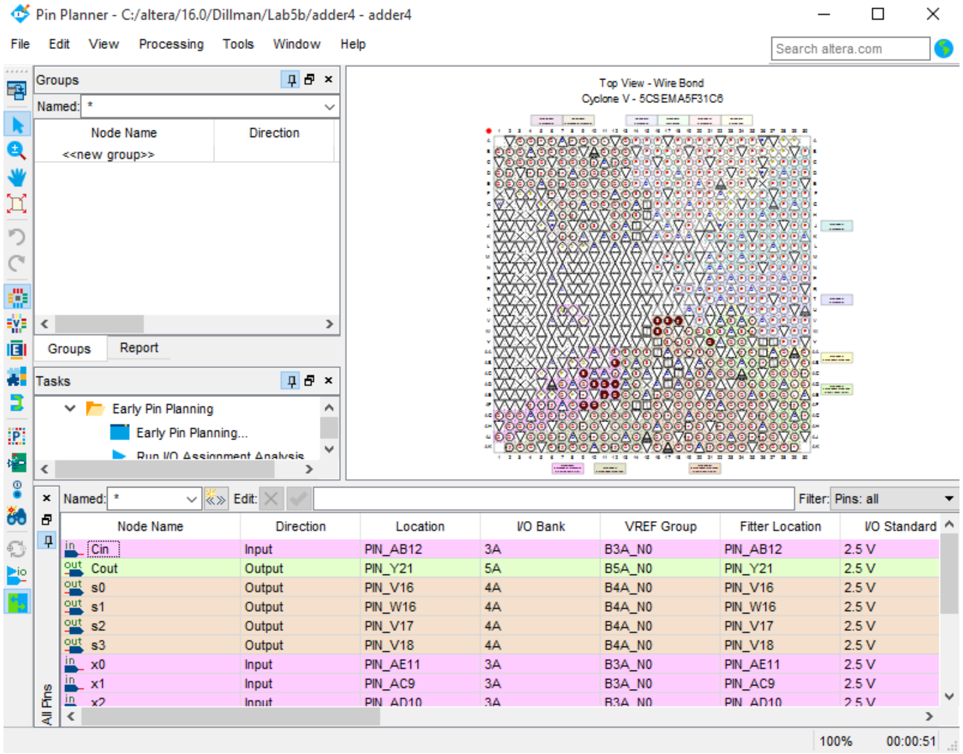


Node Name	Direction
S[3..0]	Output Group
X[3..0]	Input Group
Y[3..0]	Input Group
<<new group>>	

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
Cin	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
Cout	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	2.5 V
Over	Output	PIN_W21	5A	B5A_N0	PIN_W21	2.5 V
S[3]	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V
S[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V
S[1]	Output	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V
S[0]	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
X[3]	Input	PIN_AE12	3A	B3A_N0	PIN_AE12	2.5 V
X[2]	Input	PIN_AD10	3A	B3A_N0	PIN_AD10	2.5 V

b. Design B

In the design B, we repeat the process in design A again; after setting the DE1 board, we compile the project again and upload the file to the DE1 board. Finally, we upload the file to the DE1 board. After setting, we stimulate the test again on the board by turn on or turn off the switch. The result is same as the prelab stimulation.

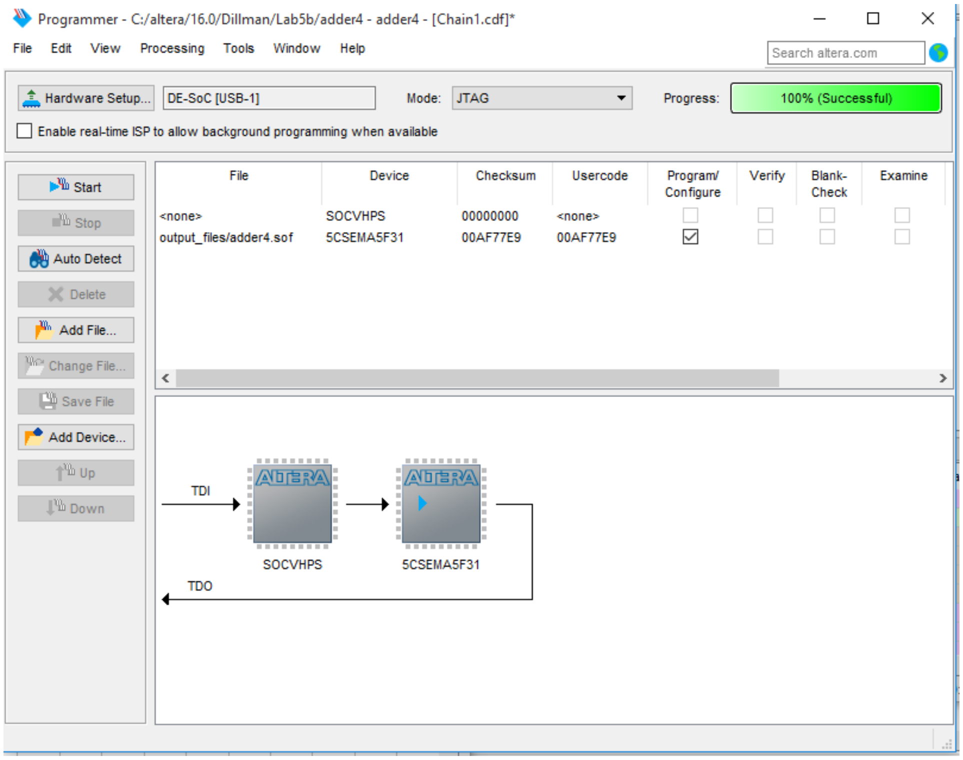


Pin Planner - C:/altera/16.0/Dillman/Lab5b/adder4 - adder4

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
Cin	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
Cout	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	2.5 V
s0	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
s1	Output	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V
s2	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V
s3	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V
x0	Input	PIN_AE11	3A	B3A_N0	PIN_AE11	2.5 V
x1	Input	PIN_AC9	3A	B3A_N0	PIN_AC9	2.5 V
x2	Input	PIN_AD10	3A	B3A_N0	PIN_AD10	2.5 V

100% 00:00:51



Programmer - C:/altera/16.0/Dillman/Lab5b/adder4 - adder4 - [Chain1.cdf]*

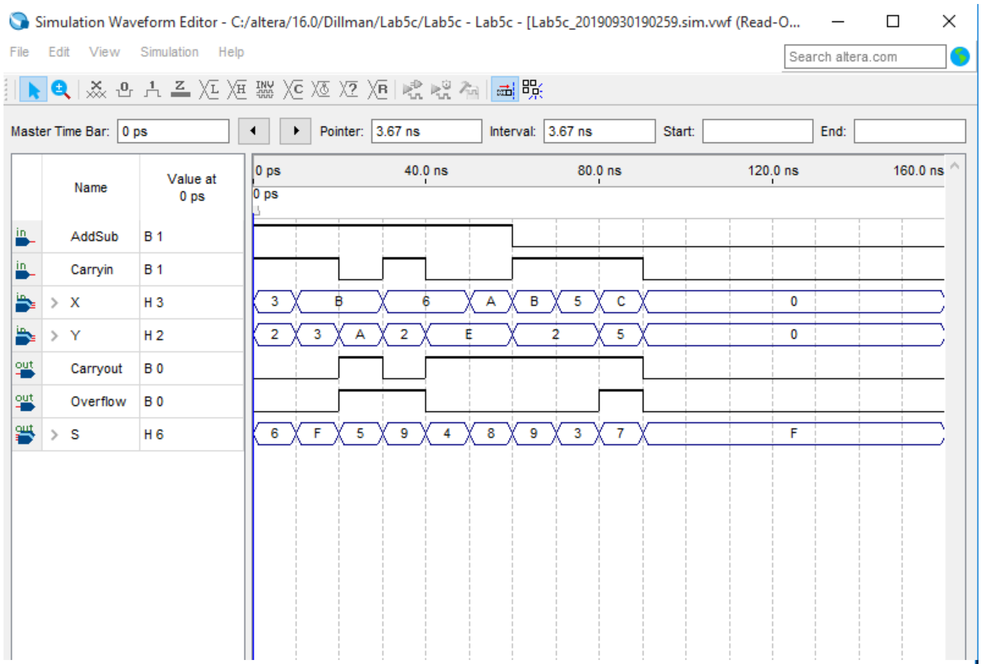
Hardware Setup... DE-SoC [USB-1] Mode: JTAG Progress: 100% (Successful)

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
<none>	SOCVHPS	00000000	<none>	□	□	□	□
output_files/adder4.sof	5CSEMA5F31	00AF77E9	00AF77E9	☑	□	□	□

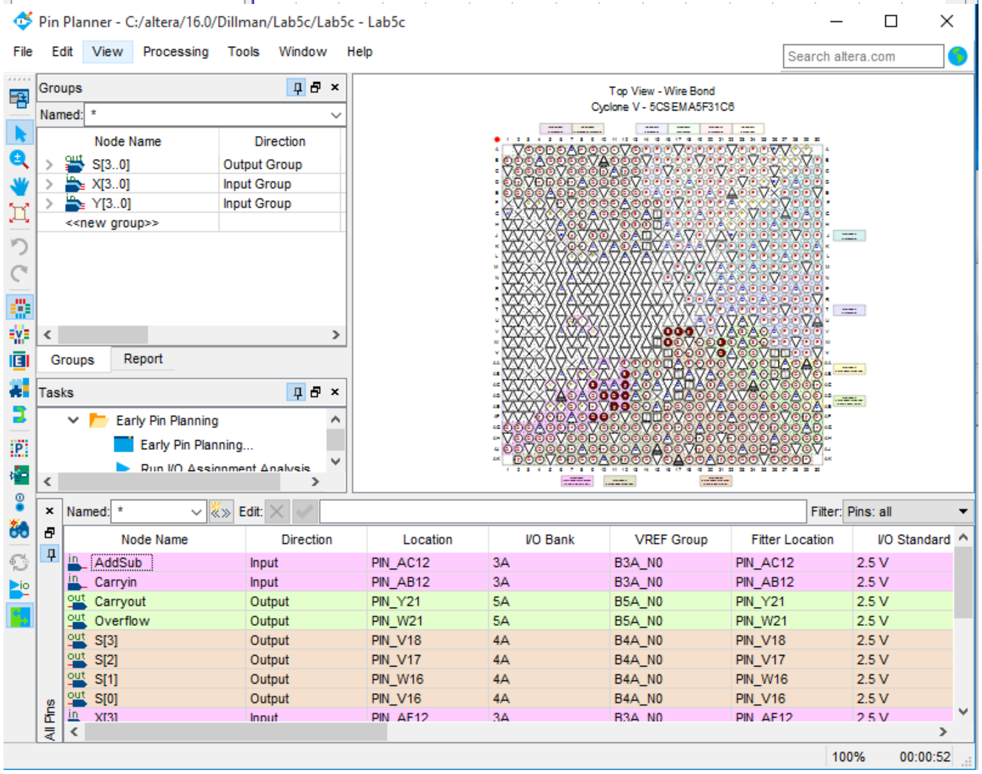
c. Design C

In design C, we firstly run the stimulation by the waveform file. We repeat the setting DE1 board process same as Design A and B. Then, we run the compiling process again and upload the file to the DE1 board. After configuration, we do the testing process. The result is same as the result of waveform file.


Simulation Waveform Editor - C:/altera/16.0/Dillman/Lab5c/Lab5c - Lab5c - [Lab5c_20190930190259.sim.vwf (Read-O...

Master Time Bar: 0 ps Pointer: 3.67 ns Interval: 3.67 ns Start: End:

Name	Value at 0 ps
AddSub	B 1
Carryin	B 1
X	H 3
Y	H 2
Carryout	B 0
Overflow	B 0
S	H 6


Pin Planner - C:/alter/16.0/Dillman/Lab5c/Lab5c - Lab5c

Groups

Named: *

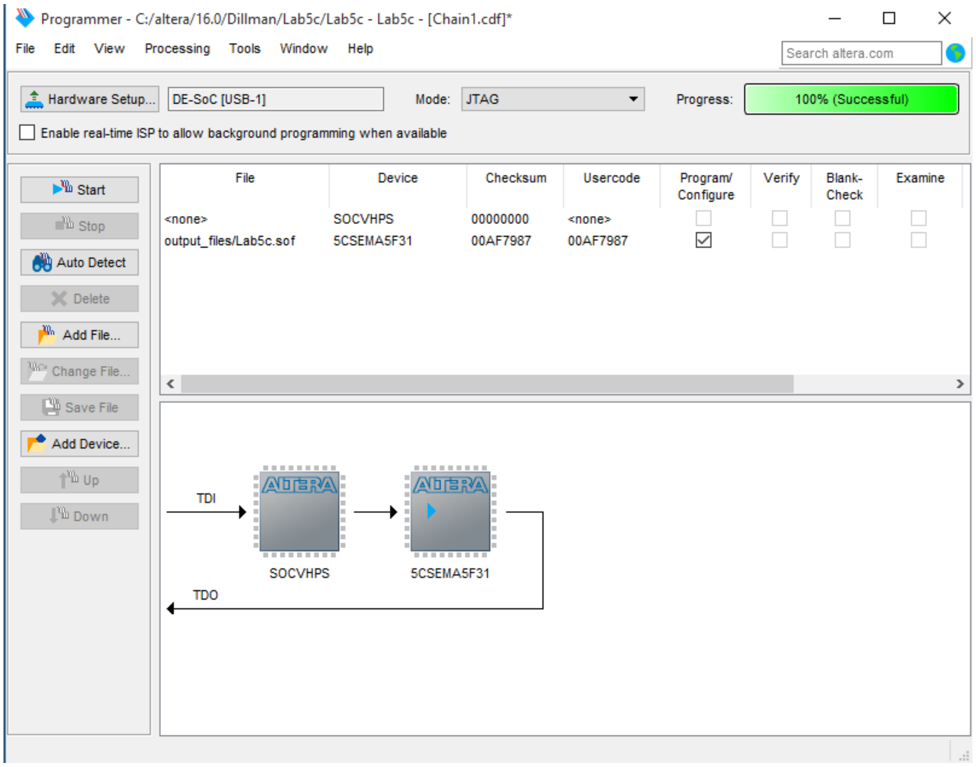
Node Name	Direction
S[3..0]	Output Group
X[3..0]	Input Group
Y[3..0]	Input Group
<<new group>>	

Tasks

- Early Pin Planning
  - Early Pin Planning...
  - Run I/O Assignment Analysis

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
AddSub	Input	PIN_AC12	3A	B3A_N0	PIN_AC12	2.5 V
Carryin	Input	PIN_AB12	3A	B3A_N0	PIN_AB12	2.5 V
Carryout	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	2.5 V
Overflow	Output	PIN_W21	5A	B5A_N0	PIN_W21	2.5 V
S[3]	Output	PIN_V18	4A	B4A_N0	PIN_V18	2.5 V
S[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	2.5 V
S[1]	Output	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V
S[0]	Output	PIN_V16	4A	B4A_N0	PIN_V16	2.5 V
X[3]	Input	PIN_AF12	3A	B3A_N0	PIN_AF12	2.5 V



Programmer - C:/altera/16.0/Dillman/Lab5c/Lab5c - Lab5c - [Chain1.cdf]*

Hardware Setup... DE-SoC [USB-1]

Mode: JTAG

Progress: 100% (Successful)

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
<none>	SOCVHPS	00000000	<none>	□	□	□	□
output_files/Lab5c.sof	5CSEMA5F31	00AF7987	00AF7987	☑	□	□	□

Result

The result of design A and B are same as the waveform result in the prelab. For the design C, the final result is also same as the waveform file result.

6

Input	*Add_Sub*	Add	add	add	add	add	add	sub	sub	Sub
Input	*Cin*	1	1	0	1	0	0	1	1	1
Input	X	0x3	0xB	0xB	0x6	0x6	0xA	0xB	0x5	0xC
Input	Y	0x2	0x3	0xA	0x2	0xE	0xE	0x2	0x2	0x5
Output	S	6	F	5	9	4	8	9	3	7
Output	*Cout*	0	0	1	0	1	1	1	1	1
Output	*Over*	0	0	1	1	0	0	0	0	1

Download your design to the Cyclone® V 5CSEMA5F31C6 device on the DE1 board. Use inputs SW[9..6] (SW[6] is the LSB) for the X input and SW[5..2] (SW[2] is the LSB) for the Y input. Use SW[0] for Cin. Use LEDs LEDG[3..0] (LEDG[0] is the LSB) for the circuit output S. Use LEDs LEDR[9] and LEDR[8] for Cout and Over, respectively.

By using the created test vectors, verify that Design C is operational on the DE1 board. Receive TA initials when you complete verifications.

Conclusion

In this lab, we learned two different ways to build an adder in VHDL file. We also understand the method to test the adder. Furthermore, we also learned how to insert ADD_SUB Gate to the schematic file.

145

Pre-Lab (30%)	**Score**	**TA initial**
Design A (15%)		
Design B (15%)		

Report (70%)	
10% Introduction	
10% Procedures	
20% Results	
30% Conclusions	

Lab Grade	