Lab 06
Adder Design Tradeoffs

ECE 380-002
University of Alabama

Yichen Huang
Thomas Dillman

2019/10/7

Introduction

In this lab, we use the Quartus prime software package to design and simulate one 4-bit subtractor and one 4-bit adder. The requirements for the lab consist of completing Quartus Prime designs, downloading designs to the Altera DE 1 board, printing circuit diagrams, VHDL files, simulating results, and the laboratory report.

Procedure

A. Prelab

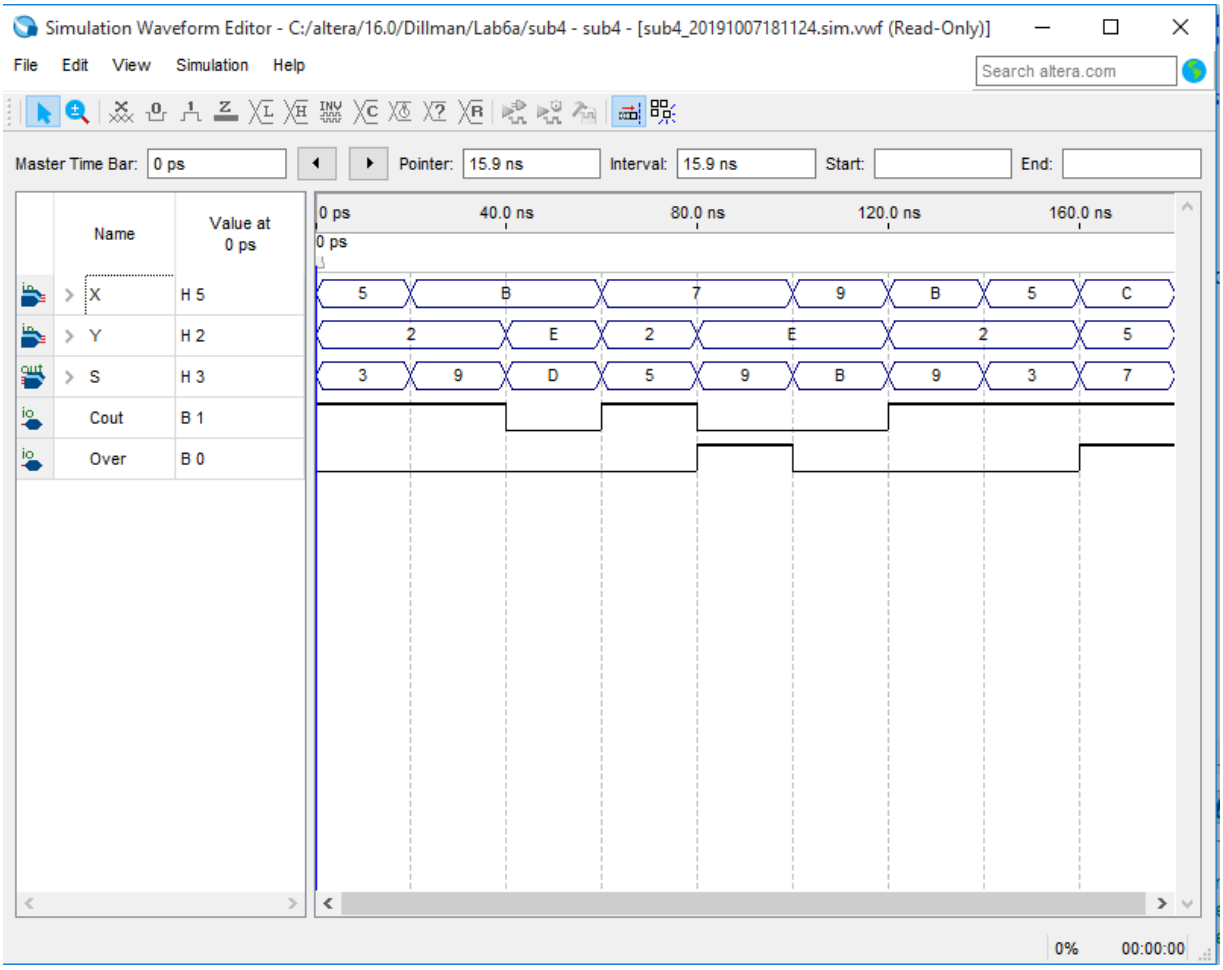
a. Design A

In the design A, we created a 4-bit subtractor using structure VHDL. The circuit has two 4-bit data input A and B, a 4-bit output S, a carry-out bit and an overflow flag. We create the circuit by using two VHDL files, fulladd.vhd and sub4b.vhd. After coding, we compile the project and run the simulation by test vectors.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all ;
ENTITY sub4 IS
PORT (
X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
S : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
Cout, Over: INOUT STD_LOGIC ) ;
END sub4 ;
ARCHITECTURE Structure OF sub4 IS
SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
SIGNAL Cin : STD_LOGIC;
COMPONENT fulladd
PORT ( Cin, x, y: IN STD_LOGIC ;
s, Cout: OUT STD_LOGIC ) ;
END COMPONENT;
BEGIN
Cin <= '1';
stage0: fulladd PORT MAP ( Cin, X(0), Not Y(0), S(0), C(1) ) ;
stage1: fulladd PORT MAP ( C(1), X(1), Not Y(1), S(1), C(2) ) ;
stage2: fulladd PORT MAP ( C(2), X(2), Not Y(2), S(2), C(3) ) ;
stage3: fulladd PORT MAP ( C(3), X(3), Not Y(3), S(3), Cout) ;
Over <= C(3) XOR Cout;
END Structure ;
```

fulladd.vhd | sub4.vhd | Compilation Report - sub4

```
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 ;
```



b. Design B

In the design B, we design a 4-bit adder by carry look ahead adders. We implemented the design by dataflow VHDL method to the file. This 4-bit adder have two 4-bit inputs, a 4-bit data outputs, and a bit for carryout. After implementation and compiling process, we run the stimulation, the result is same as we aspect.

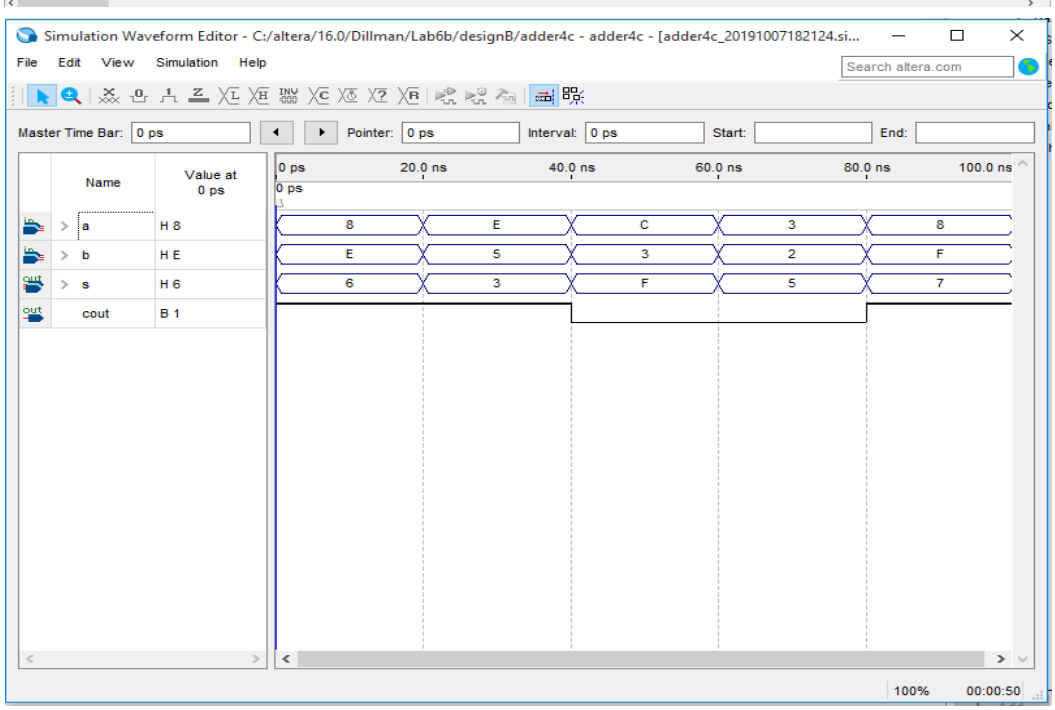
```
LIBRARY ieee;

USE ieee.std_logic_1164.all;

ENTITY adder4c IS
PORT ( a, b: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
s: OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
cout: OUT STD_LOGIC);
END adder4c;

ARCHITECTURE LogicFunc OF adder4c IS
SIGNAL c: STD_LOGIC_VECTOR (4 DOWNTO 0);
SIGNAL p: STD_LOGIC_VECTOR (3 DOWNTO 0);
SIGNAL g: STD_LOGIC_VECTOR (3 DOWNTO 0);
SIGNAL cin: STD_LOGIC;

BEGIN
cin <= '0';
G1: FOR i IN 0 TO 3 GENERATE
p(i) <= a(i) XOR b(i);
g(i) <= a(i) AND b(i);
s(i) <= p(i) XOR c(i);
END GENERATE;
c(0) <= cin;
c(1) <= (cin AND p(0)) OR g(0);
c(2) <= (cin AND p(0) AND p(1)) OR (g(0) AND p(1)) OR g(1);
c(3) <= (cin AND p(0) AND p(1) AND p(2)) OR (g(0) AND p(1) AND p(2)) OR (g(1) AND p(2)) OR g(2);
c(4) <= (cin AND p(0) AND p(1) AND p(2) AND p(3)) OR (g(0) AND p(1) AND p(2) AND p(3)) OR (g(1) AND p(2) AND p(3)) OR (g(2) AND p(3)) OR g(3);
cout <= c(4);
END LogicFunc;
```



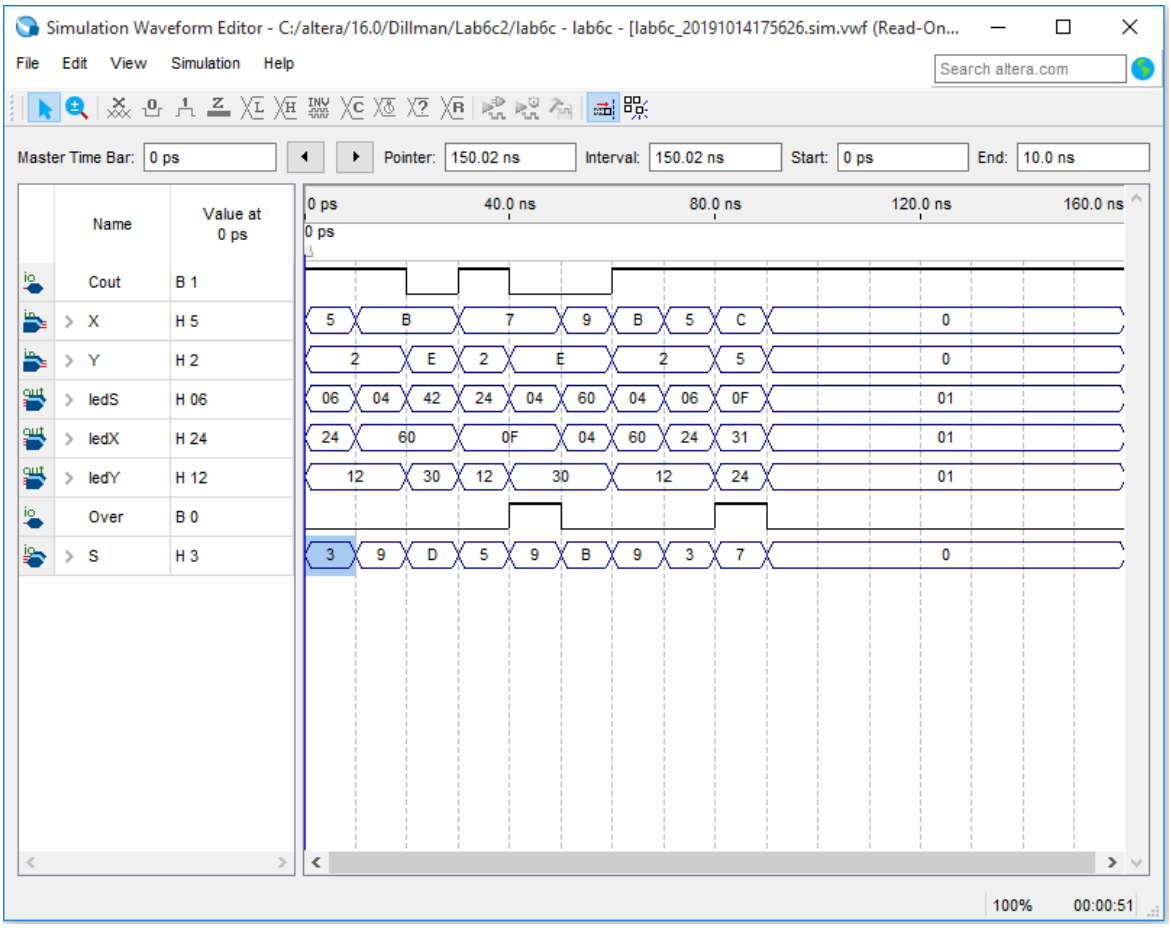
c. Design C

In the design C, we use structure VHDL code to display seven segment LED to display 2 input and 1 output.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all ;
ENTITY lab6c IS
PORT (
 X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
 S : INOUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
 Cout, Over: INOUT STD_LOGIC ;
 ledX, ledY, ledS : OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
END lab6c;
ARCHITECTURE Structure OF lab6c IS
 SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
 SIGNAL Cin : STD_LOGIC;
COMPONENT fulladd
PORT ( Cin, x, y: IN STD_LOGIC ;
 s, Cout: OUT STD_LOGIC ) ;
END COMPONENT;
COMPONENT DesignC
PORT ( h: IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
 leds: OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
END COMPONENT;
 BEGIN
 Cin <= '1';
 stage0: fulladd PORT MAP ( Cin, X(0), Not Y(0), S(0), C(1) ) ;
 stage1: fulladd PORT MAP ( C(1), X(1), Not Y(1), S(1), C(2) ) ;
 stage2: fulladd PORT MAP ( C(2), X(2), Not Y(2), S(2), C(3) ) ;
 stage3: fulladd PORT MAP ( C(3), X(3), Not Y(3), S(3), Cout) ;
 Over <= C(3) XOR Cout;
 stage4: DesignC PORT MAP (X, ledX);
 stage5: DesignC PORT MAP (Y, ledY);
 stage6: DesignC PORT MAP (S, ledS);
END Structure ;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all ;
ENTITY DesignC IS
PORT ( h: IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
 leds: OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
END DesignC ;
ARCHITECTURE Behavior OF DesignC IS
BEGIN
PROCESS (h)
 BEGIN
CASE h IS --abcdefg
 WHEN "0000" => leds<= "0000001" ;--0
 WHEN "0001" => leds<= "1001111" ;--1
 WHEN "0010" => leds<= "0010010" ;--2
 WHEN "0011" => leds<= "0000110" ;--3
 WHEN "0100" => leds<= "1001100" ;--4
 WHEN "0101" => leds<= "0100100" ;--5
 WHEN "0110" => leds<= "0100000" ;--6
 WHEN "0111" => leds<= "0001111" ;--7
 WHEN "1000" => leds<= "0000000" ;--8
 WHEN "1001" => leds<= "0000100" ;--9
 WHEN "1010" => leds<= "0001000" ;--A
 WHEN "1011" => leds<= "1100000" ;--B
 WHEN "1100" => leds<= "0110001" ;--C
 WHEN "1101" => leds<= "1000010" ;--D
 WHEN "1110" => leds<= "0110000" ;--E
 WHEN "1111" => leds<= "0111000" ;--F
 WHEN OTHERS => NULL ;
END CASE ;
END PROCESS ;
END Behavior ;
```

```
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 ;
```



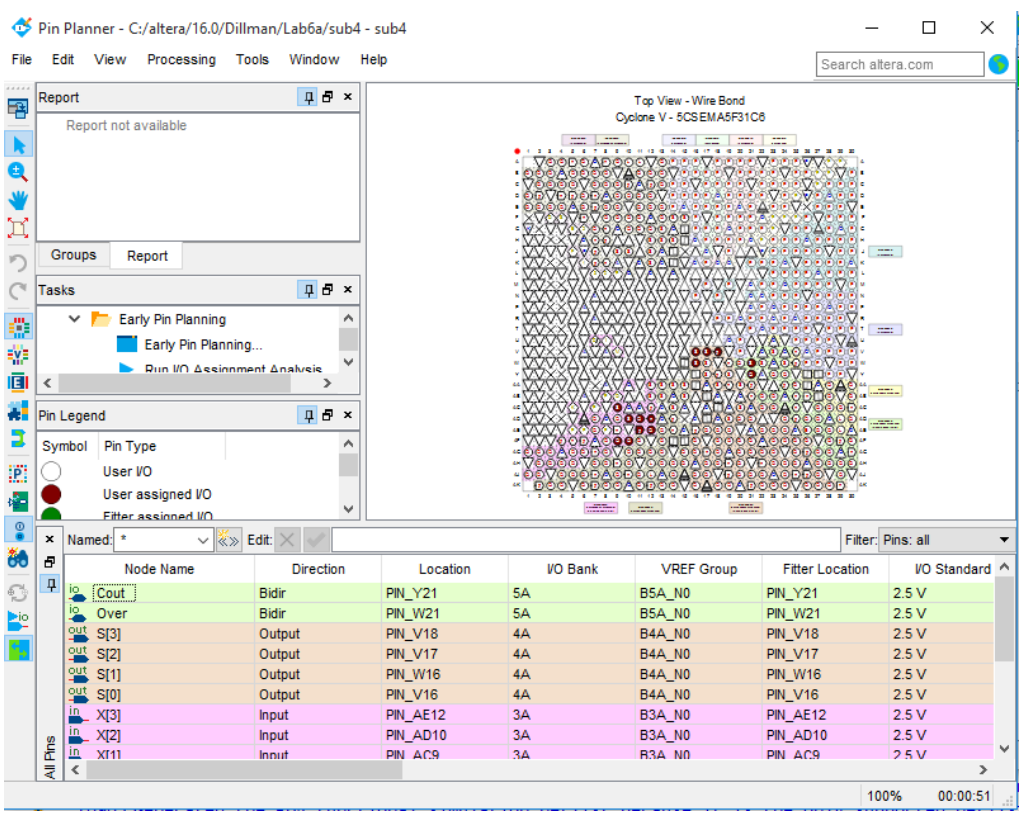
d. Testing vectors

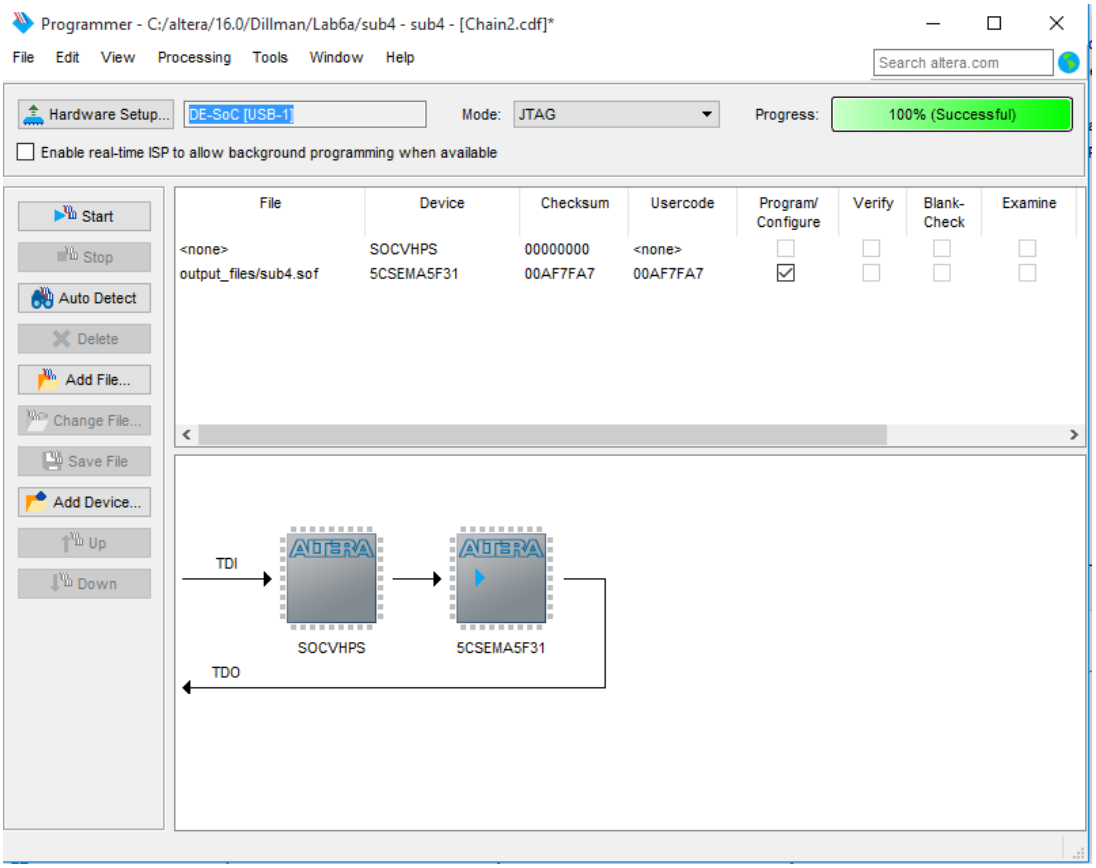
For the testing part, this lab is focusing on 4-bits adder circuit and 4-bits subtractor. In general, we have three outputs: a 4-bits output for addition or subtraction, a carry-out bit and a bit indicate overflow. For the test case, I will choose the case which will lead a correct 4-bits output with a carryout bit and without carryout bit(Like B - 2 and B – E) and other case which will lead overflow with carryout bit and without carryout bit (Like 7 – E and C – 5)

B. During the lab

a. Design A

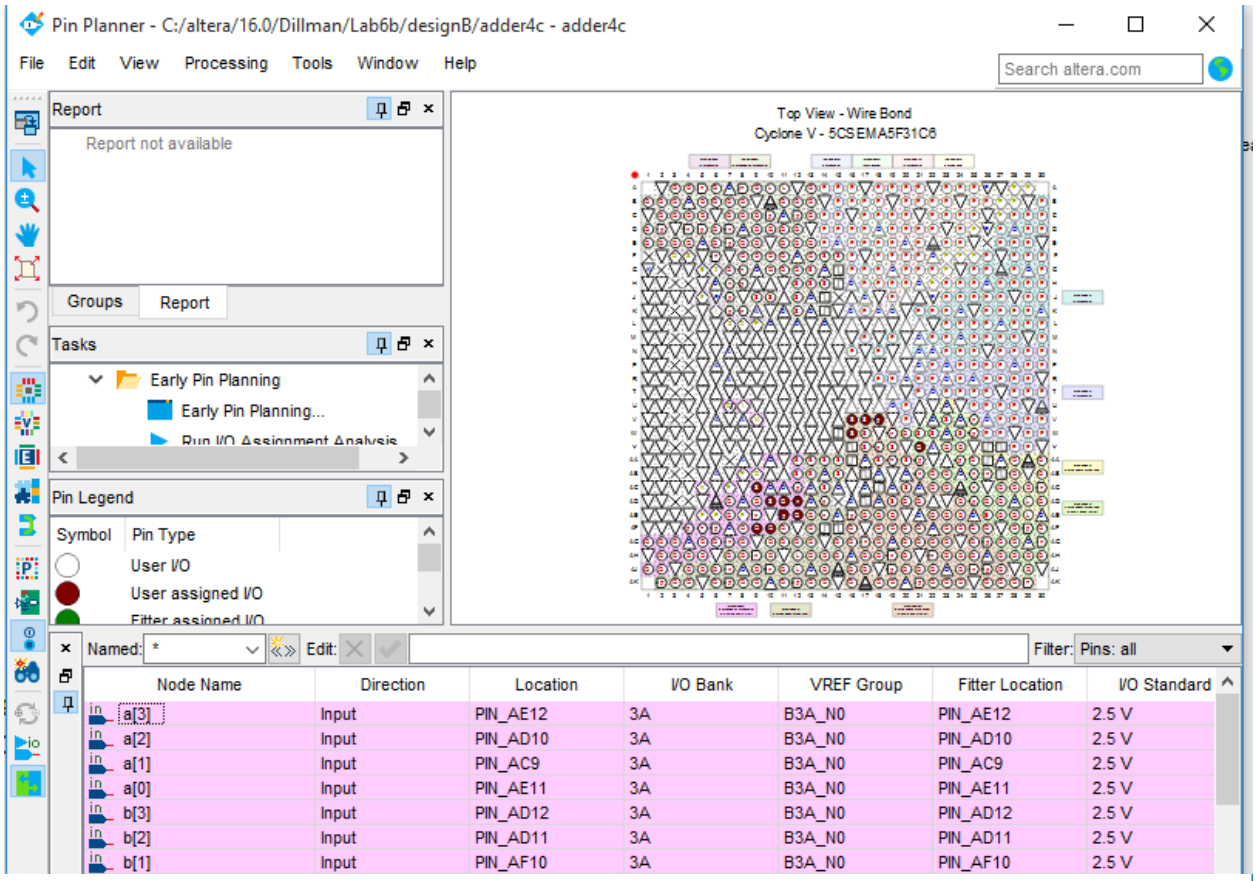
In lab for design A, we set the switches on the board and compiled again. After success, we set the DE1 board and do the test again, the result is same as simulated result.

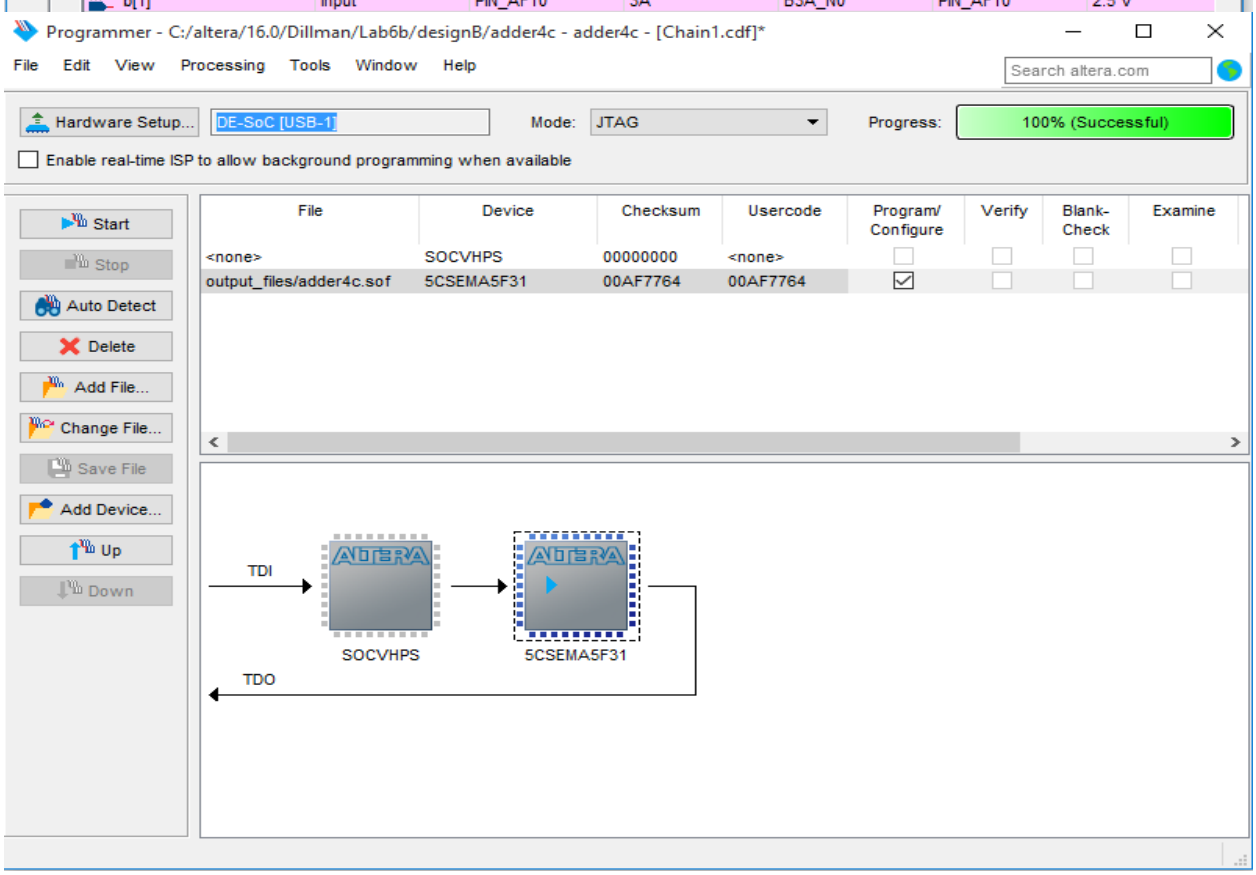




b. Design B

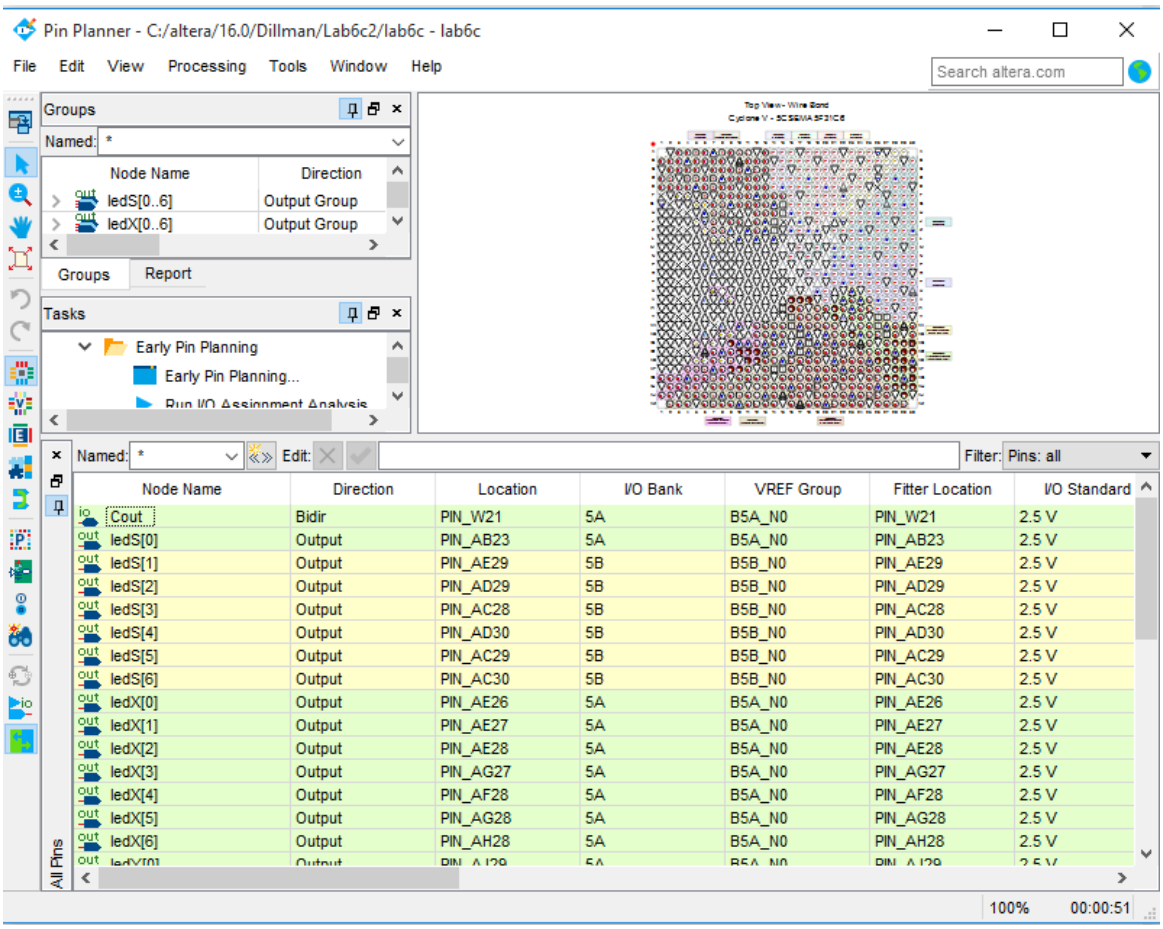
In the design B, we repeat the same process in design A. The result is also same as we expected and simulated result.


Pin Planner - C:/altera/16.0/Dillman/Lab6b/designB/adder4c - adder4c
File Edit View Processing Tools Window Help
Search altera.com
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning...
Pin Legend
Symbol Pin Type
User I/O
User assigned I/O
Top View - Wire Bond
Cyclone V - 5CSEMA5F31C6
Named: *
Edit:
Filter: Pins: all
Node Name Direction Location I/O Bank VREF Group Fitter Location I/O Standard
a[3] Input PIN_AE12 3A B3A_N0 PIN_AE12 2.5 V
a[2] Input PIN_AD10 3A B3A_N0 PIN_AD10 2.5 V
a[1] Input PIN_AC9 3A B3A_N0 PIN_AC9 2.5 V
a[0] Input PIN_AE11 3A B3A_N0 PIN_AE11 2.5 V
b[3] Input PIN_AD12 3A B3A_N0 PIN_AD12 2.5 V
b[2] Input PIN_AD11 3A B3A_N0 PIN_AD11 2.5 V
b[1] Input PIN_AF10 3A B3A_N0 PIN_AF10 2.5 V

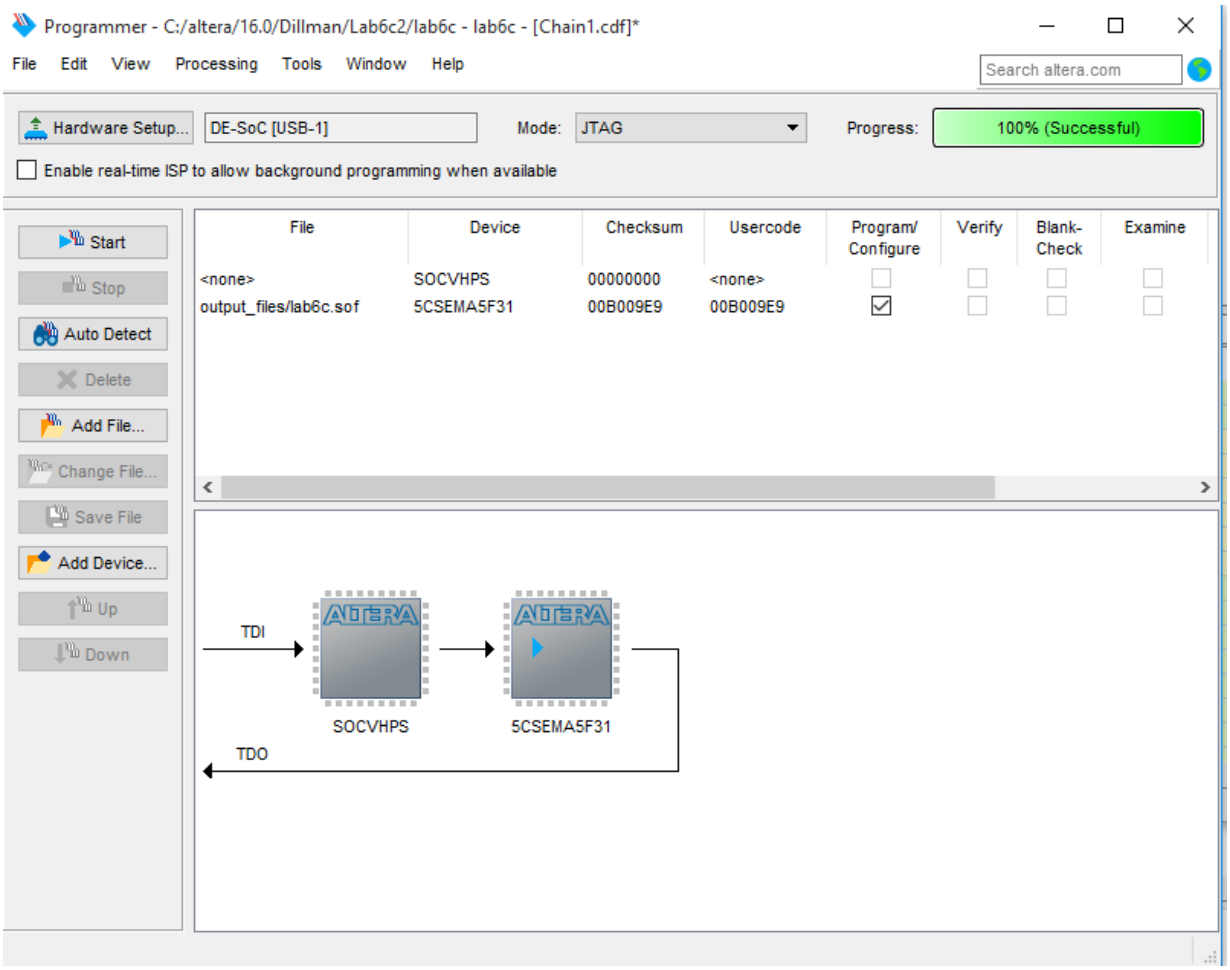

Programmer - C:/altera/16.0/Dillman/Lab6b/designB/adder4c - adder4c - [Chain1.cdf]*
File Edit View Processing Tools Window Help
Search altera.com
Hardware Setup... DE-SoC [USB-1]
Mode: JTAG
Progress: 100% (Successful)
Enable real-time ISP to allow background programming when available
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File Device Checksum Usercode Program/ Configure Verify Blank-Check Examine
<none> SOCVHPS 00000000 <none>
output_files/adder4c.sof 5CSEMA5F31 00AF7764 00AF7764
TDI
SOCVHPS
5CSEMA5F31
TDO

c. Design C

For the design C, we also repeat the same process in design B. After setting, we run the test on the board. The LED display the number of inputs and output. The number is also correct our simulation.



Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard
Cout	Bidir	PIN_W21	5A	B5A_N0	PIN_W21	2.5 V
ledS[0]	Output	PIN_AB23	5A	B5A_N0	PIN_AB23	2.5 V
ledS[1]	Output	PIN_AE29	5B	B5B_N0	PIN_AE29	2.5 V
ledS[2]	Output	PIN_AD29	5B	B5B_N0	PIN_AD29	2.5 V
ledS[3]	Output	PIN_AC28	5B	B5B_N0	PIN_AC28	2.5 V
ledS[4]	Output	PIN_AD30	5B	B5B_N0	PIN_AD30	2.5 V
ledS[5]	Output	PIN_AC29	5B	B5B_N0	PIN_AC29	2.5 V
ledS[6]	Output	PIN_AC30	5B	B5B_N0	PIN_AC30	2.5 V
ledX[0]	Output	PIN_AE26	5A	B5A_N0	PIN_AE26	2.5 V
ledX[1]	Output	PIN_AE27	5A	B5A_N0	PIN_AE27	2.5 V
ledX[2]	Output	PIN_AE28	5A	B5A_N0	PIN_AE28	2.5 V
ledX[3]	Output	PIN_AG27	5A	B5A_N0	PIN_AG27	2.5 V
ledX[4]	Output	PIN_AF28	5A	B5A_N0	PIN_AF28	2.5 V
ledX[5]	Output	PIN_AG28	5A	B5A_N0	PIN_AG28	2.5 V
ledX[6]	Output	PIN_AH28	5A	B5A_N0	PIN_AH28	2.5 V



Result

The result of design A, B and C are the same as the waveform file in the prelab. For the design C the LED display the number of input and output are also correct.

Ms

Pre-Lab (30%)	**Score**	**TA initial**
20% Designs	A	
10% Paragraph	B -	
	B	
Report (70%)		
10% Introduction		
10% Procedures		
20% Results		
30% Conclusions		
Lab Grade (100%)		
Bonus (20 pts)		

Conclusion

In this lab, we learned how to create adder and subtractor in two different ways structure VHDL and dataflow VHDL. We also understand how to create carry-out look ahead adder. In the design C, we learned how to use led to display varibles.