Lab 09
Flip-Flops and Counters

ECE 380-002
University of Alabama

Yichen Huang
Thomas Dillman

2019/11/11

Introduction

In this lab, we mainly focusing on the design two types of flip-flops: one is edge-triggered D flip-flop and another one is JK flip-flop. Two create two components through body diagram files and VHDL code. We also created a BDC counter in two ways those are through VHDL code and another one is use D flip-flop. Finally, we upload these circuits to the DE1 board to test the circuit.

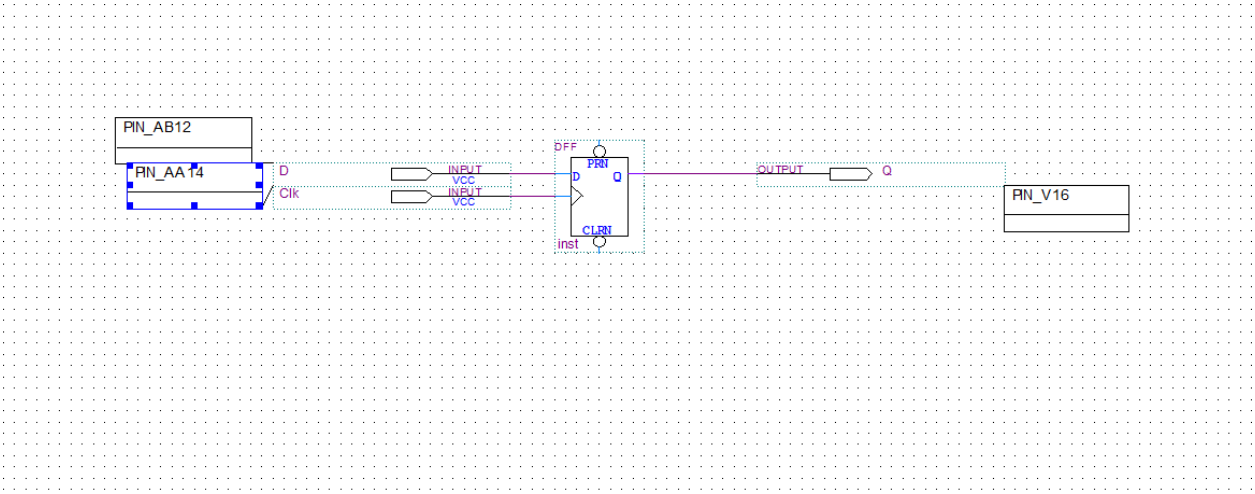
Procedure

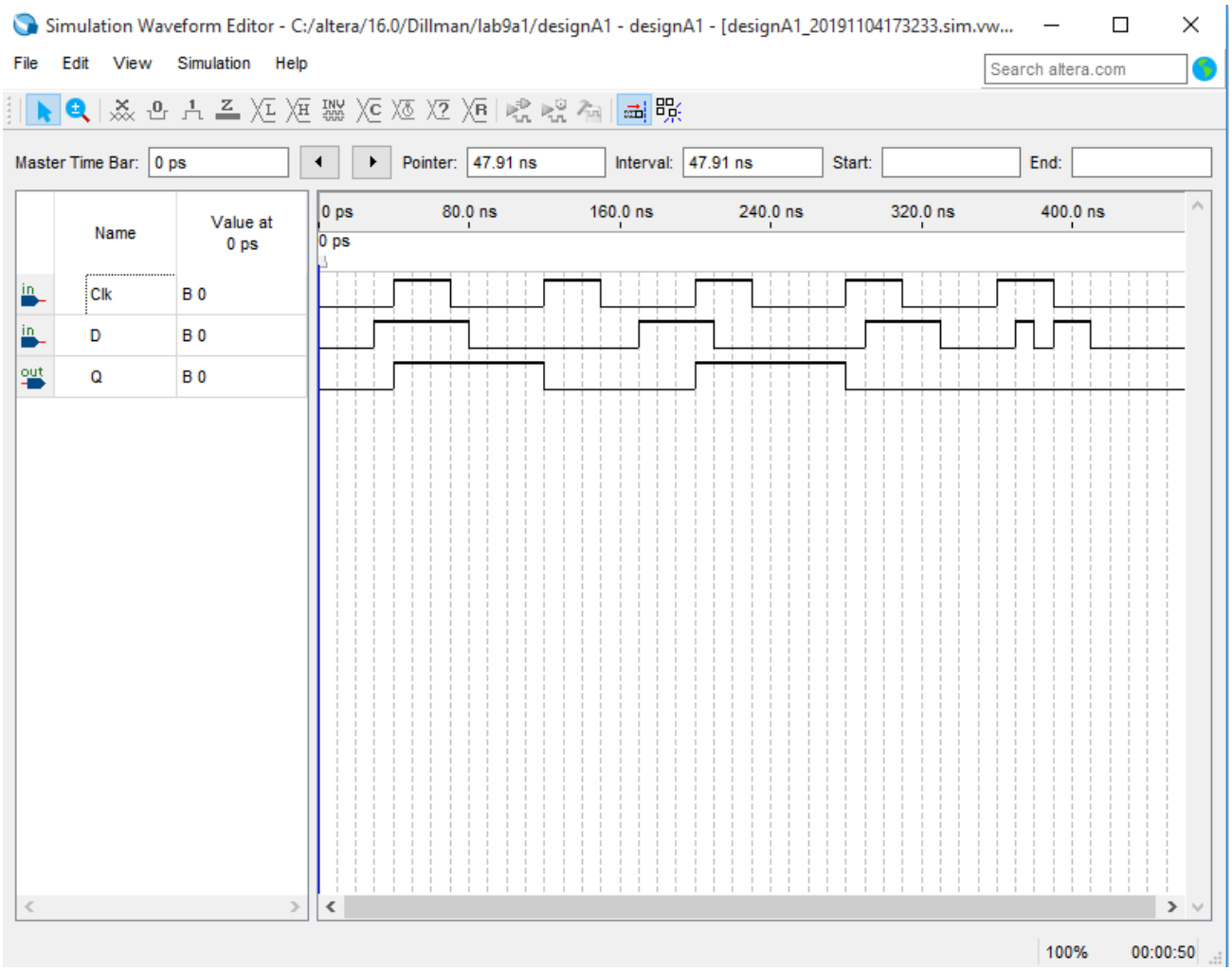
A. Prelab

a. Design A

(1) BDF File

We create a BDF file and implement a D flip-flop; after we compiled, we run the test through waveform file.

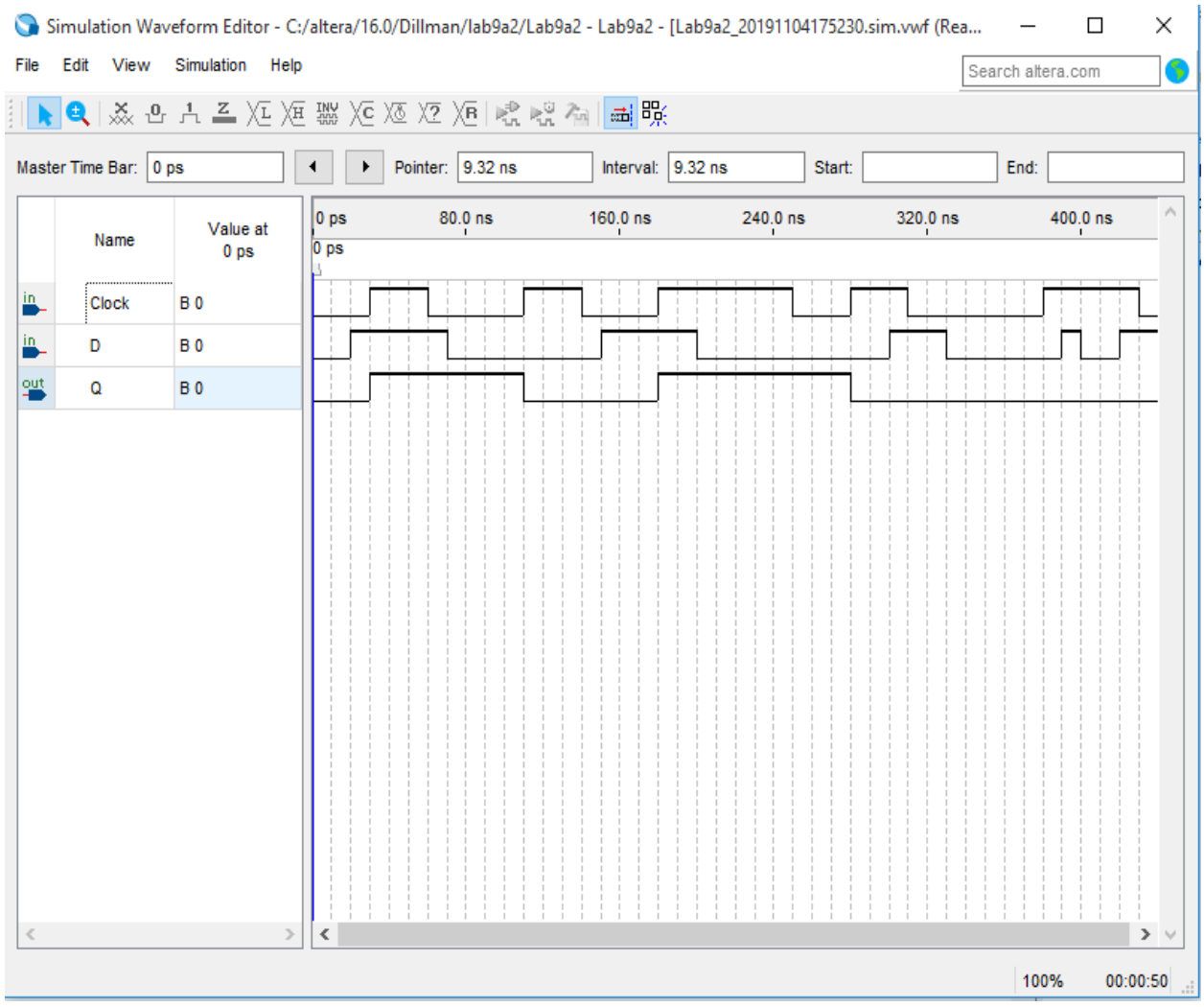




(2) VHDL File

We create D flip-flop through VHDL code. After we compile the file, we run the waveform file to test our design.

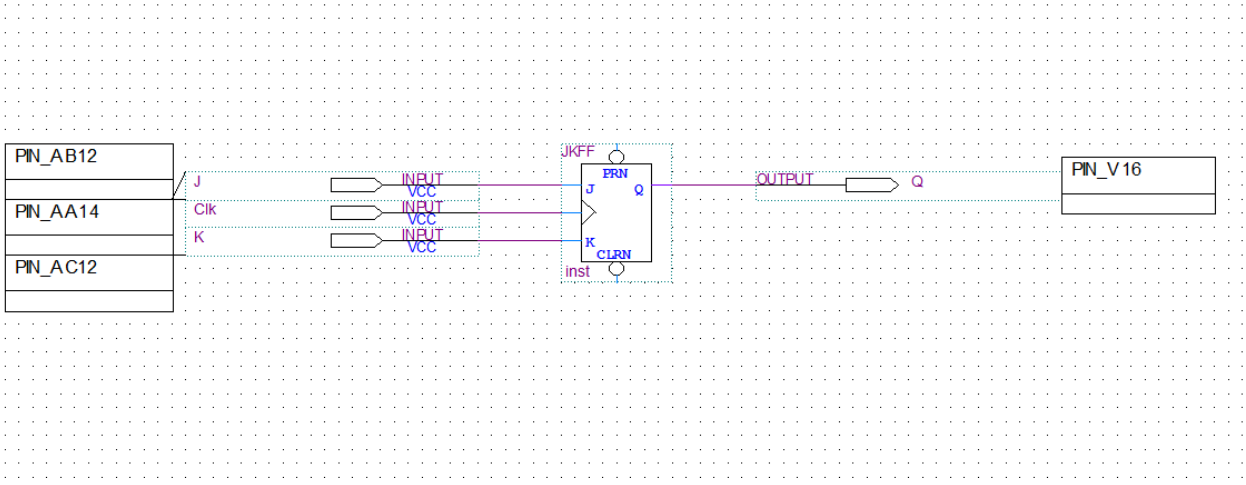
```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;
3   ENTITY Lab9a2 IS
4   PORT ( D, Clock : IN STD_LOGIC ;
5   Q : OUT STD_LOGIC) ;
6   END Lab9a2 ;
7   ARCHITECTURE Behavior OF Lab9a2 IS
8   BEGIN
9   PROCESS ( Clock )
10  BEGIN
11  IF Clock'EVENT AND Clock = '1' THEN
12  Q <= D ;
13  END IF ;
14  END PROCESS ;
15  END Behavior ;
```

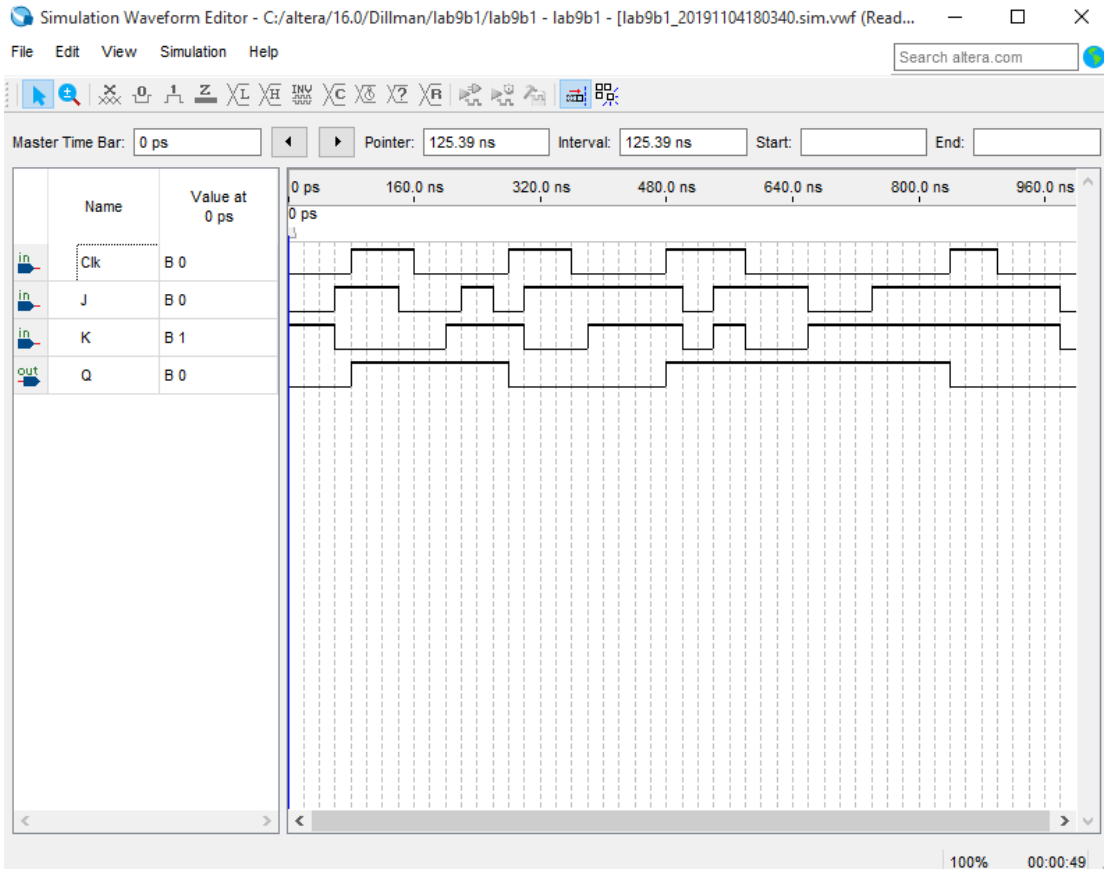


b. Design B

(1) BDF File

We create the JK flip-flop through BDF file. After we compile the file, we run the waveform file to test it.

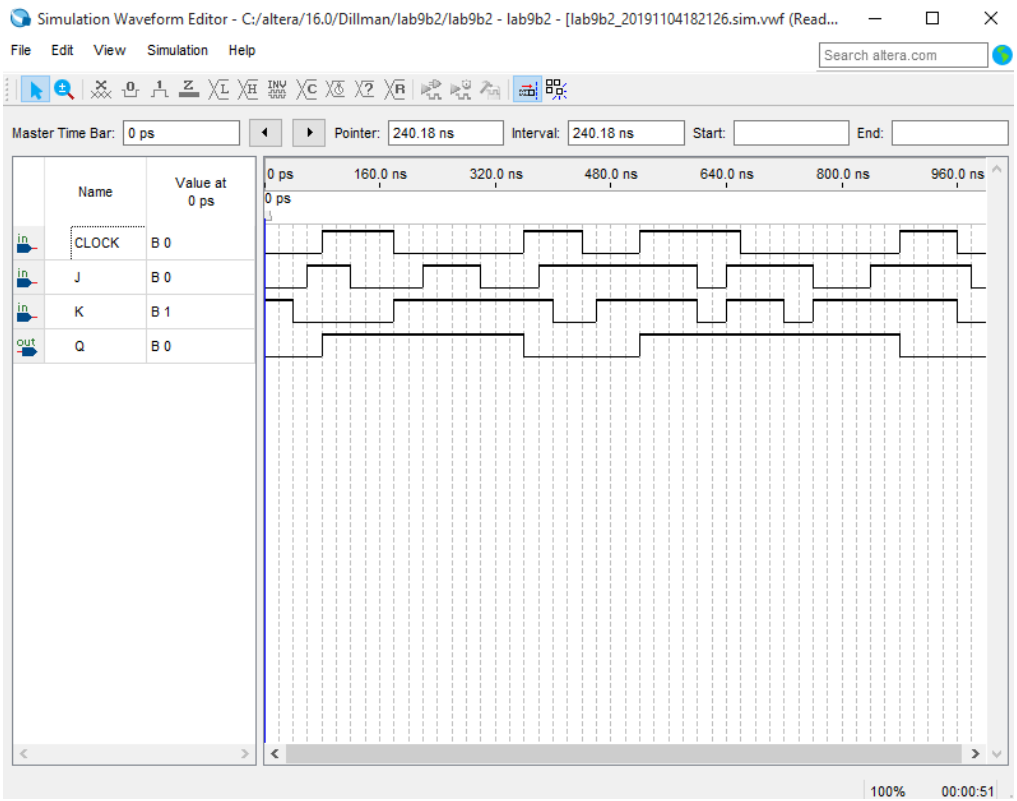




(2) VHDL File

We create the JK flip-flop through VHDL code and then we test the file through the waveform file.

```
 1   library ieee;
 2   use ieee. std_logic_1164.all;
 3   use ieee. std_logic_arith.all;
 4   use ieee. std_logic_unsigned.all;
 5
 6   entity lab9b2 is
 7   PORT( J,K,CLOCK: in std_logic;
 8   Q : out std_logic);
 9   end lab9b2;
10
11   Architecture behavioral of lab9b2 is
12   begin
13   PROCESS(CLOCK)
14   variable TMP: std_logic;
15   begin
16   if(CLOCK='1' and CLOCK'EVENT) then
17   if(J='0' and K='0')then
18   TMP:=TMP;
19   elsif(J='1' and K='1')then
20   TMP:= not TMP;
21   elsif(J='0' and K='1')then
22   TMP:='0';
23   else
24   TMP:='1';
25   end if;
26   end if;
27   Q<=TMP;
28   end PROCESS;
29   end behavioral;
```



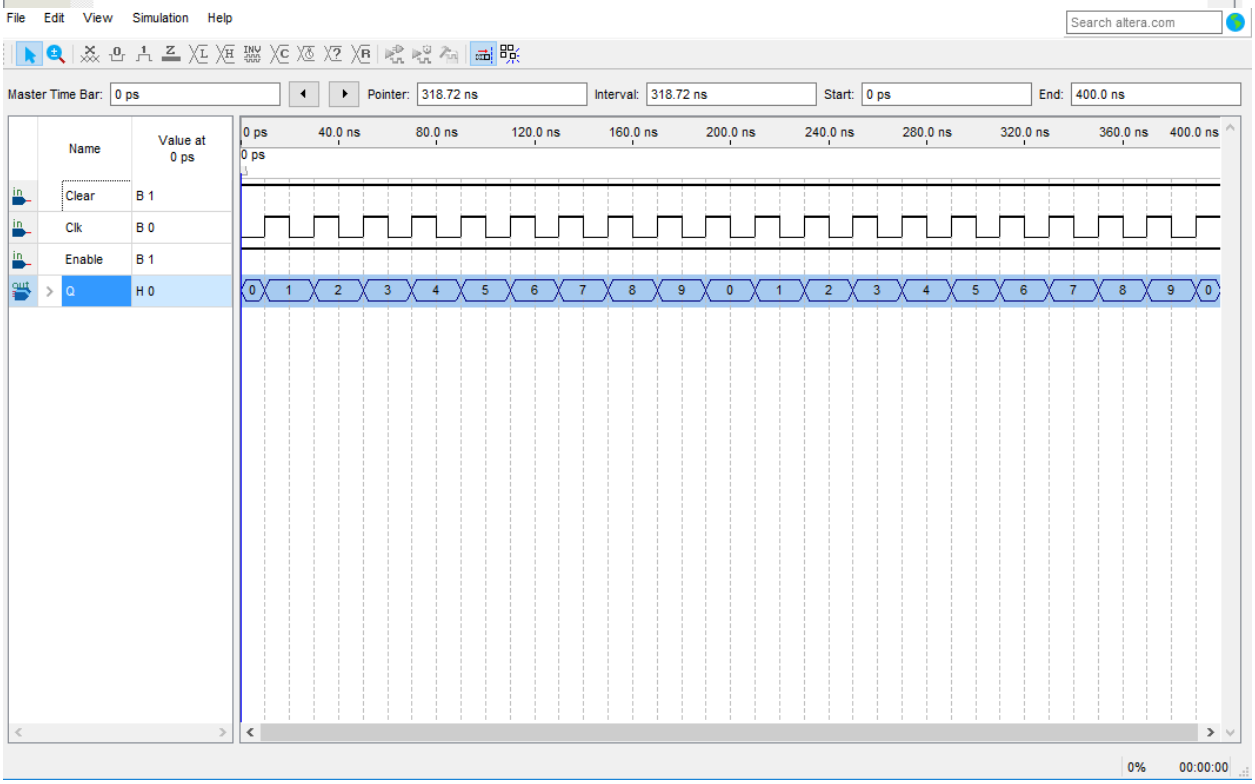
c. Design C

(1) VHDL

In this design, we implement a BCD counter through VHDL code.

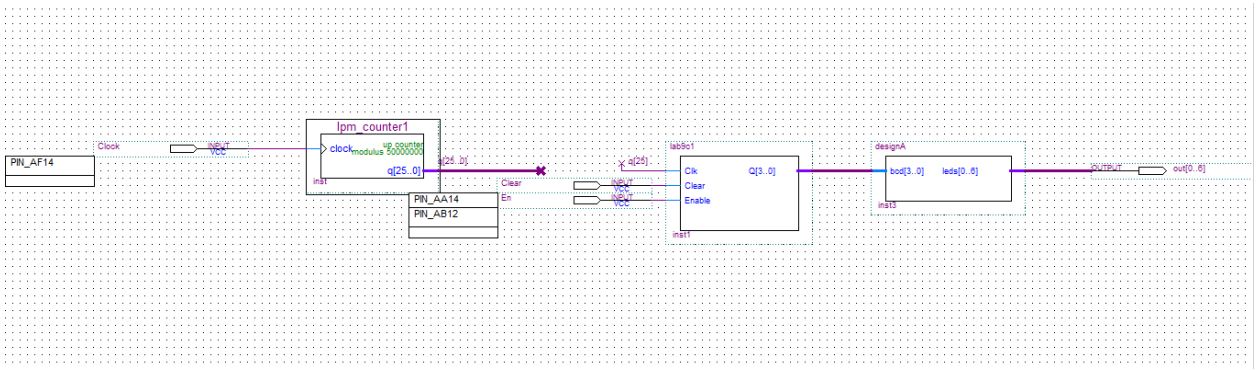
In this design, we have three input: Enable, Reset and CLK and 4-bits output Q.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity lab9c1 is
port(Clk: in std_logic; Clear: in std_logic;Enable : in std_logic;
Q : out std_logic_vector(3 downto 0));
end lab9c1;
architecture RTL of lab9c1 is
signal Count: std_logic_vector(3 downto 0);
begin
lab9c1: process(Clk,Clear)
begin
if (Clear = '0')then
Count <= (others => '0');
elsif(Clk'event and Clk = '1')then
if(Enable = '1')then
if (Count = "1001")then
Count <= (others => '0');
else
Count <= Count + 1;
end if;
end if;
end if;
end process lab9c1;
Q <= Count;
end RTL;
```



(2) BCD Counter with LED display

In this design we create a BEF file and implement the symbols from previous design and lab2 design. We then design a code converter from lab8.



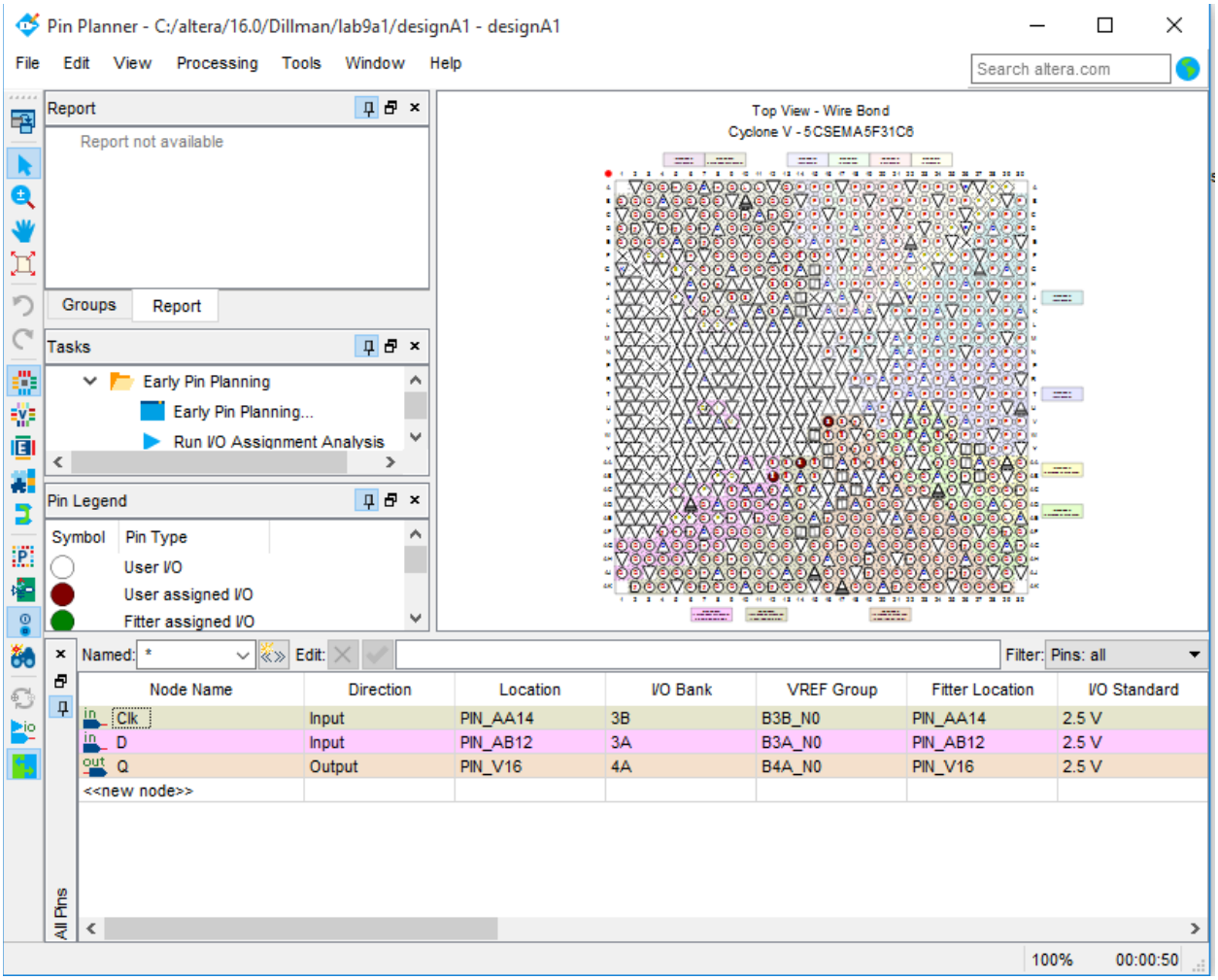

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY designA IS
PORT ( bcd: IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
 leds: OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
END designA ;
ARCHITECTURE Behavior OF designA IS
BEGIN
PROCESS ( bcd )
 BEGIN
CASE bcd IS -- abcdefg
 WHEN "0000" => leds <= "0000001";
 WHEN "0001" => leds <= "1001111";
 WHEN "0010" => leds <= "0010010";
 WHEN "0011" => leds <= "0000110";
 WHEN "0100" => leds <= "1001100";
 WHEN "0101" => leds <= "0100100";
 WHEN "0110" => leds <= "0100000";
 WHEN "0111" => leds <= "0001111";
 WHEN "1000" => leds <= "0000000";
 WHEN "1001" => leds <= "0001100";
 WHEN "1010" => leds <= "0100000";
 WHEN "1011" => leds <= "0100100";
 WHEN "1100" => leds <= "1001100";
 WHEN "1101" => leds <= "0000110";
 WHEN "1110" => leds <= "0010010";
 WHEN "1111" => leds <= "1001111";
 WHEN OTHERS => leds <= "-------";
END CASE ;
END PROCESS ;
END Behavior;
```

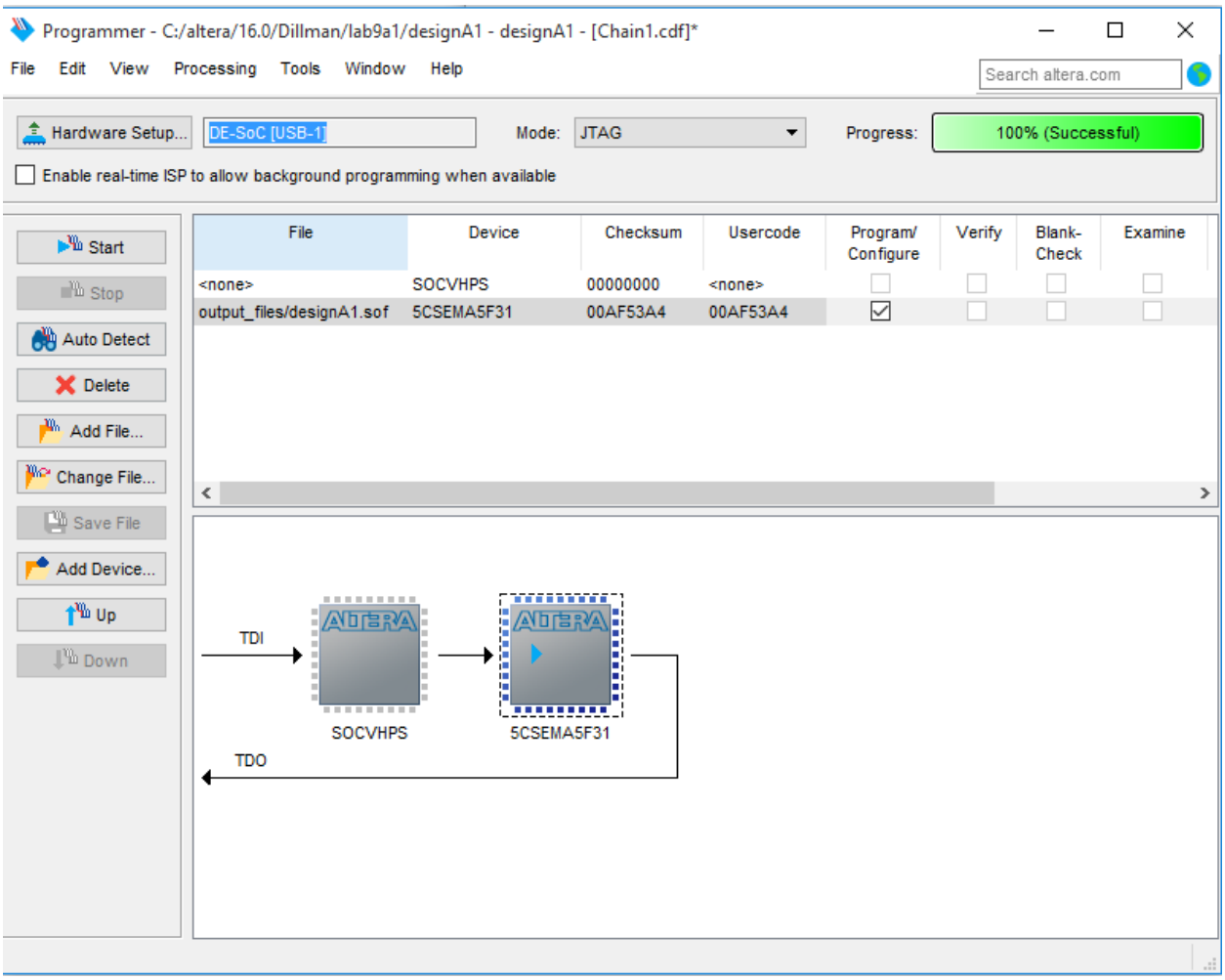
B. During the lab

a. Design A

(1)Part I

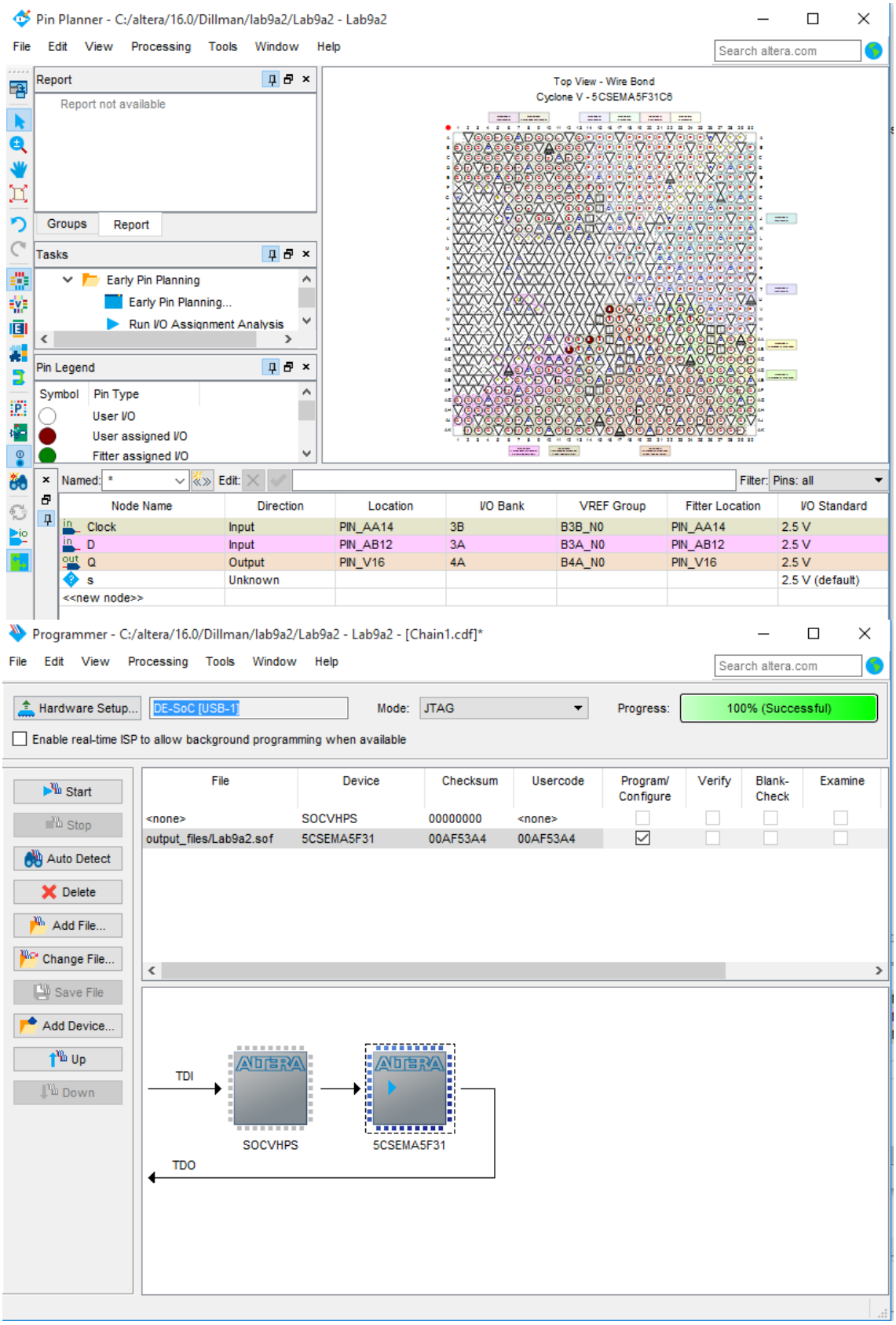
In this part we assign the input and output to the pin planner then we compiled again. Finally, we upload the design to the board, and we do the test on the board.





(2)Part II

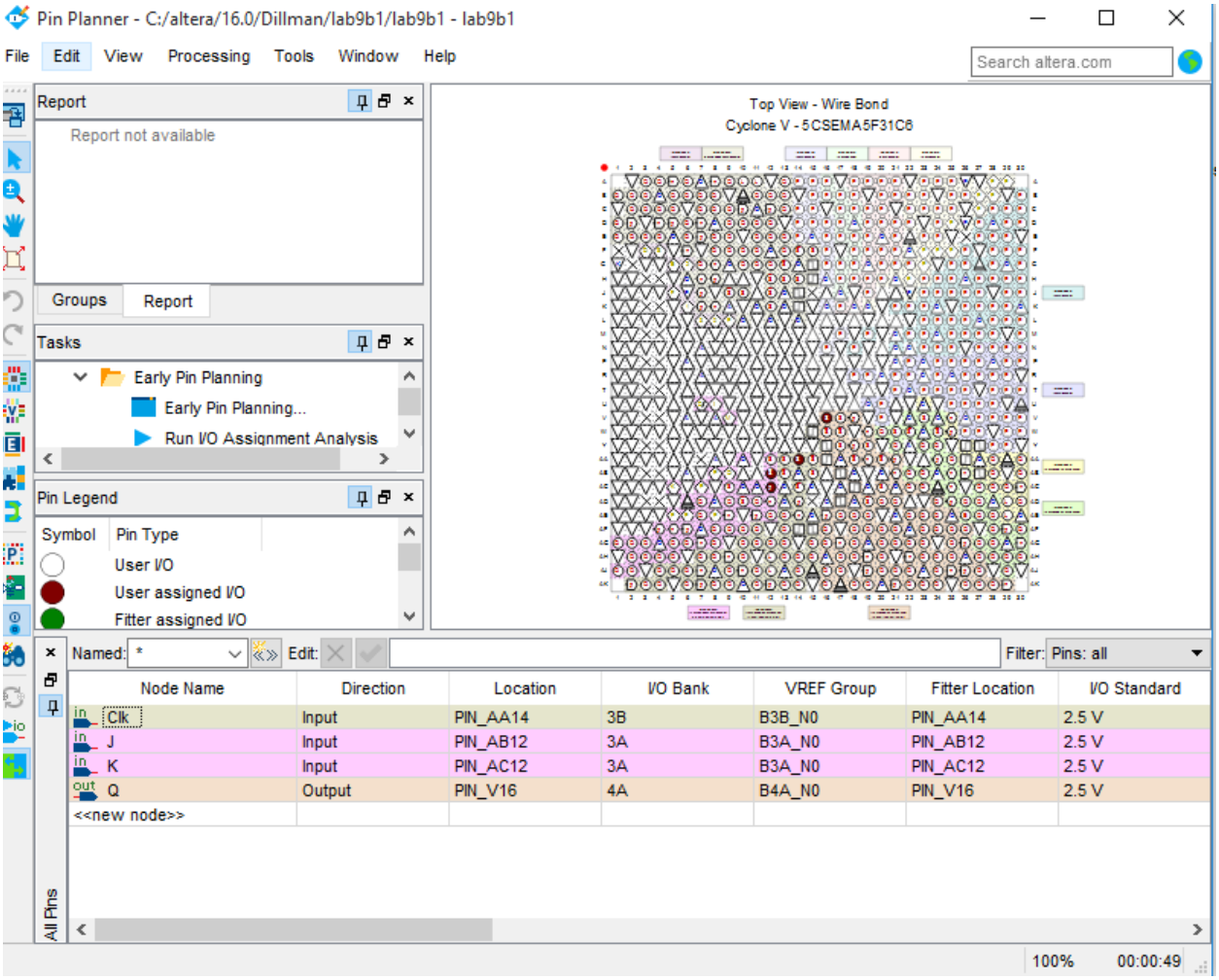
In this part we repeat the same procedure in the previous part.

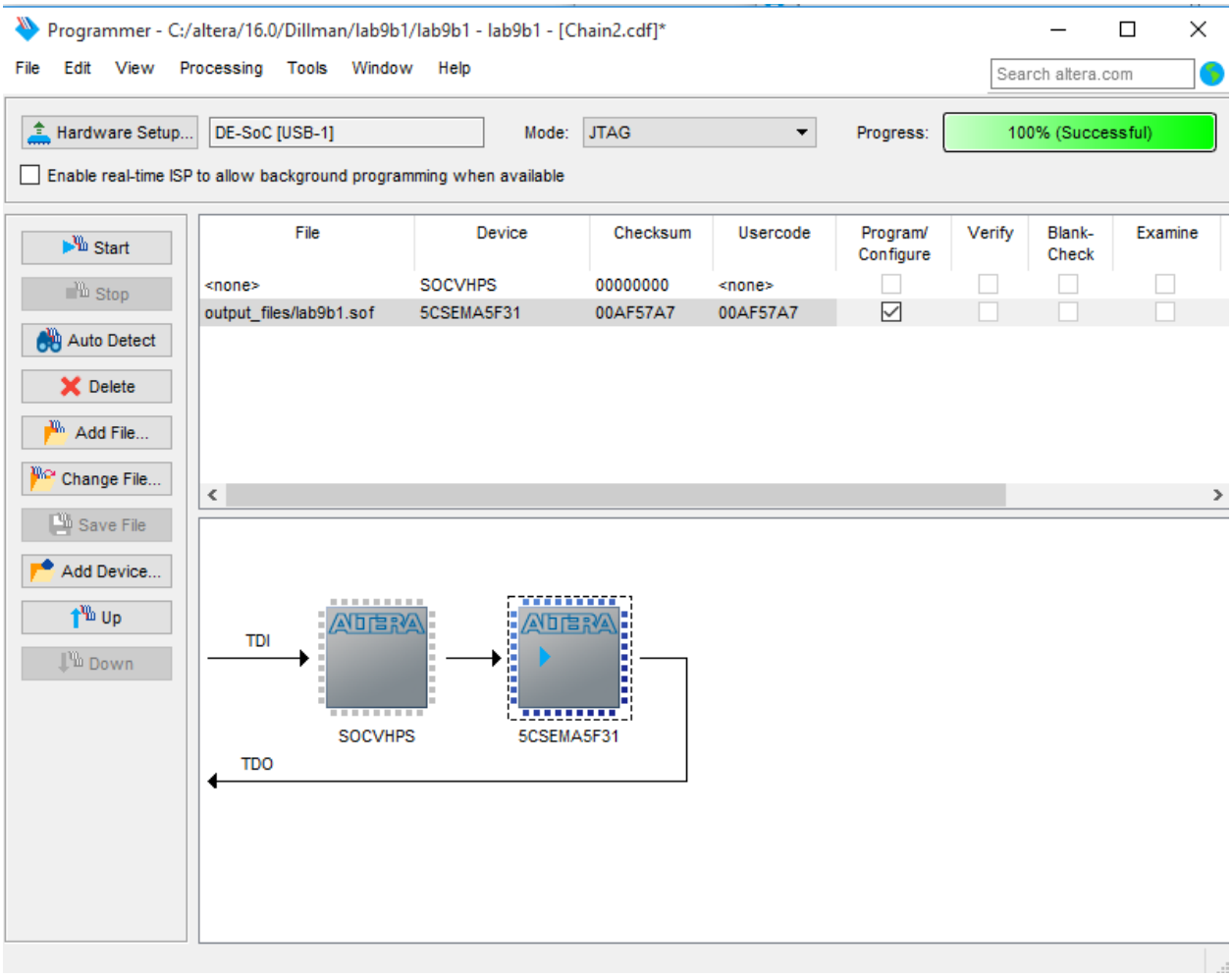

Pin Planner - C:/altera/16.0/Dillman/lab9a2/Lab9a2 - Lab9a2
File Edit View Processing Tools Window Help
Search altera.com
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning...
Run I/O Assignment Analysis
Pin Legend
Symbol Pin Type
User I/O
User assigned I/O
Fitter assigned 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
Clock Input PIN_AA14 3B B3B_N0 PIN_AA14 2.5 V
D Input PIN_AB12 3A B3A_N0 PIN_AB12 2.5 V
Q Output PIN_V16 4A B4A_N0 PIN_V16 2.5 V
s Unknown 2.5 V (default)
<<new node>>
Programmer - C:/altera/16.0/Dillman/lab9a2/Lab9a2 - Lab9a2 - [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/Lab9a2.sof 5CSEMA5F31 00AF53A4 00AF53A4
TDI
SOCVHPS
5CSEMA5F31
TDO

b. Design B

(1)Part I

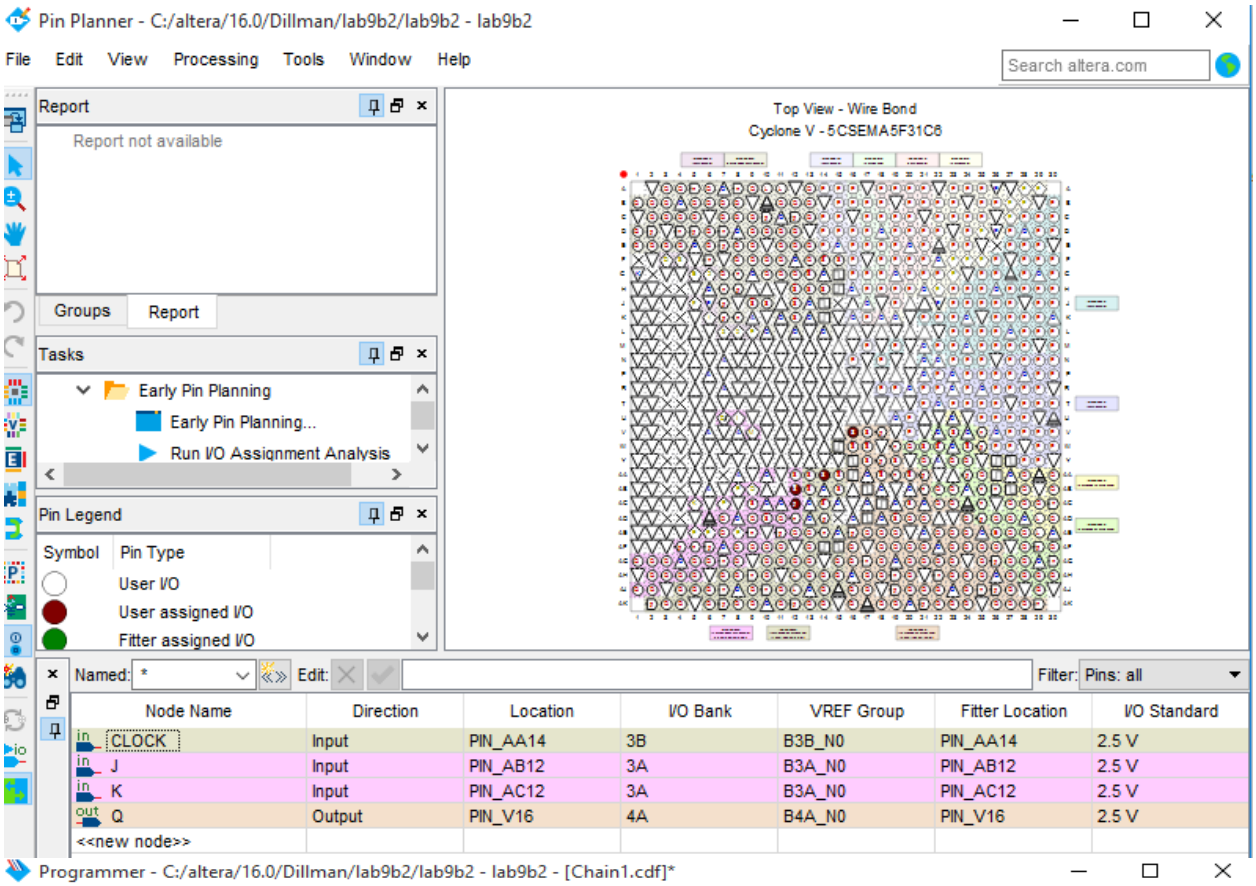
In this part, we assign input and output of JK flip-flop to the pin. Then we compiled the file again. Finally, we upload the design to the upload to DE1 board and run the test.

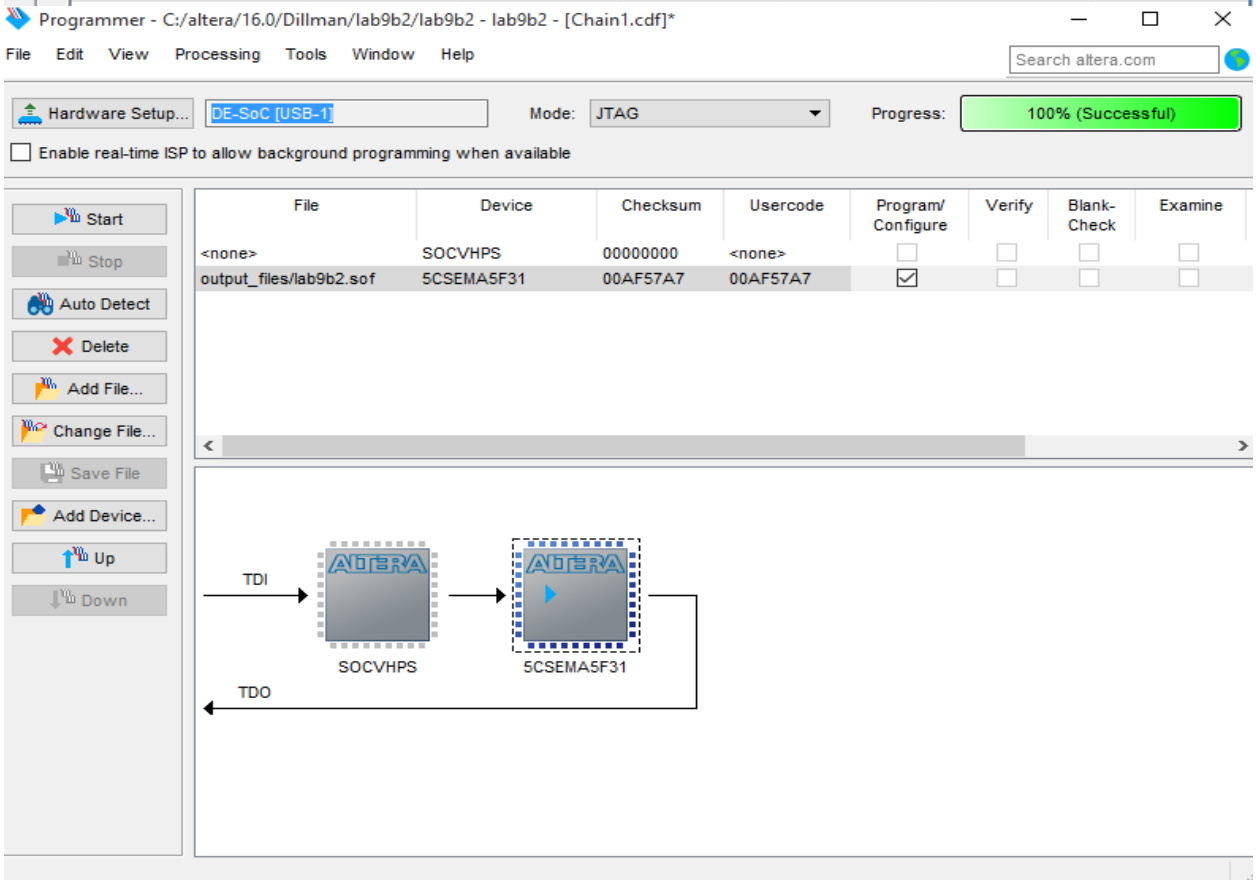




(2)Part II

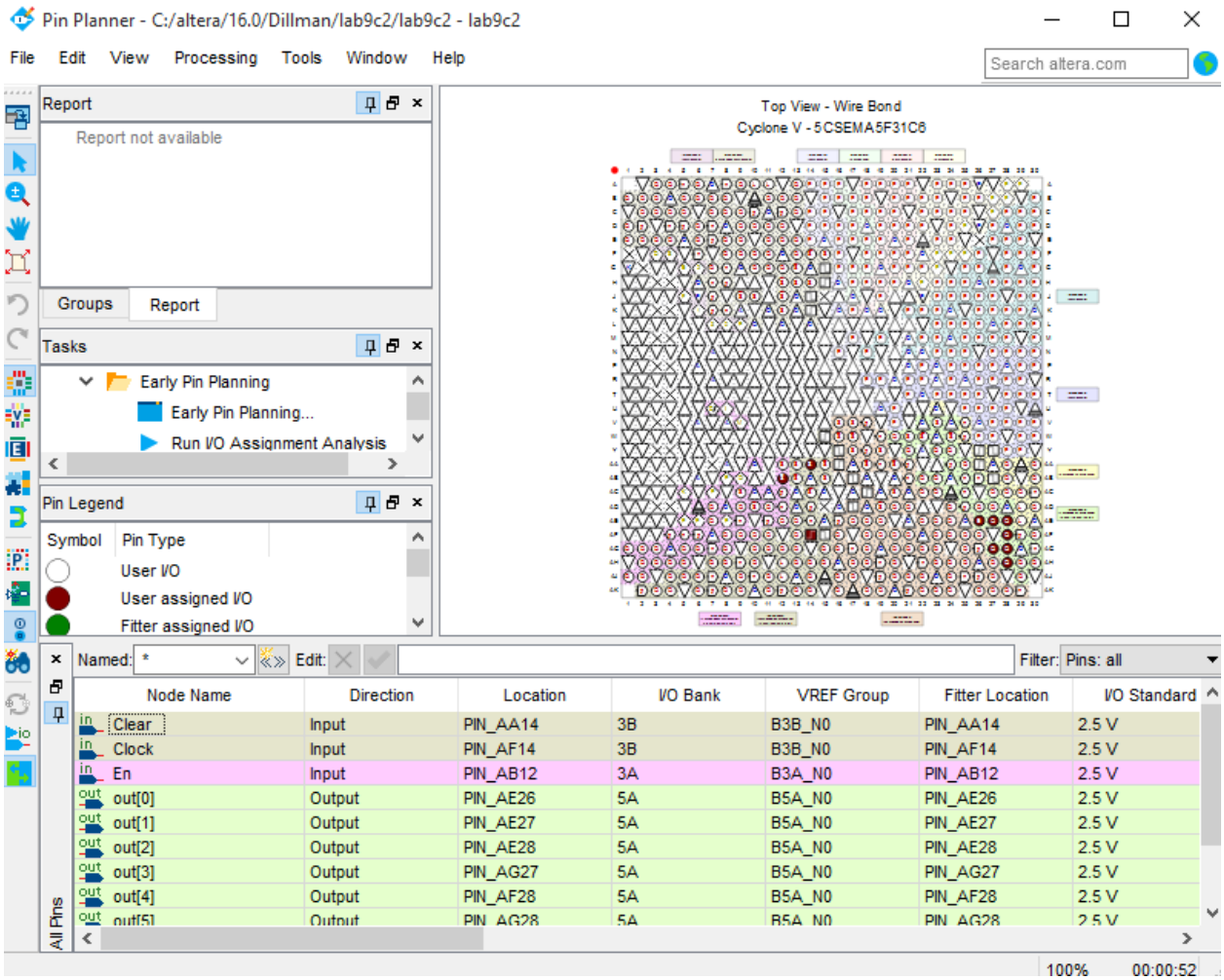
We do the same procedure in the previous part.

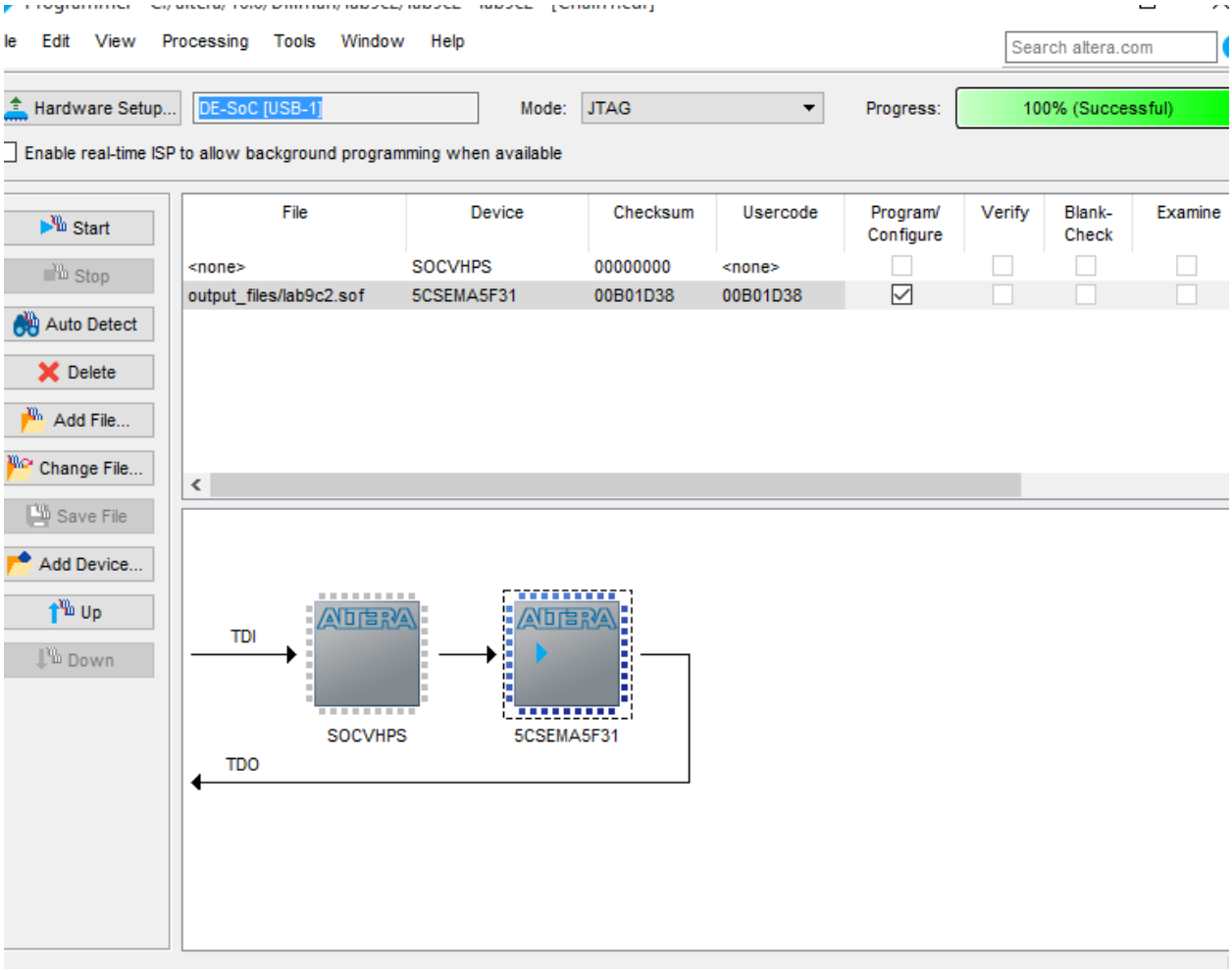

Pin Planner - C:/altera/16.0/Dillman/lab9b2/lab9b2 - lab9b2
File Edit View Processing Tools Window Help
Search altera.com
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning...
Run I/O Assignment Analysis
Pin Legend
Symbol Pin Type
User I/O
User assigned I/O
Fitter assigned 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
CLOCK Input PIN_AA14 3B B3B_N0 PIN_AA14 2.5 V
J Input PIN_AB12 3A B3A_N0 PIN_AB12 2.5 V
K Input PIN_AC12 3A B3A_N0 PIN_AC12 2.5 V
Q Output PIN_V16 4A B4A_N0 PIN_V16 2.5 V
<<new node>>


Programmer - C:/altera/16.0/Dillman/lab9b2/lab9b2 - lab9b2 - [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/lab9b2.sof 5CSEMA5F31 00AF57A7 00AF57A7
TDI
SOCVHPS
5CSEMA5F31
TDO

c. Design C

For the design C, we assigned the input and output then compiled the file again. After we upload the design to the board, we start test.





Result

All the designs are tested. The result is same as we expected.

Conclusion

- Connect 1 Hz clock signal with the four-bit counter. And use the seven-segment LED to display the counting numbers, 0 to 9.

Pre-Lab (30%)	**Score**	**TA initial**
30% Designs		

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

Lab Grade (100%)	
Bonus Points:	