Lab no 02: Simulate 2-to-1 Multiplexer & Implement 7-Segment Decoder

The purpose of this Lab is to learn to:

1) Simulate 2-to-1 multiplexer on ModelSim and verify multiplexer function using testbench. In this lab, you will build the multiplexer using logic gates (refer to Lab 01).
2) Implement the seven-segment decoder on FPGA. You will write the logic equation for each segment using Verilog bitwise operators.

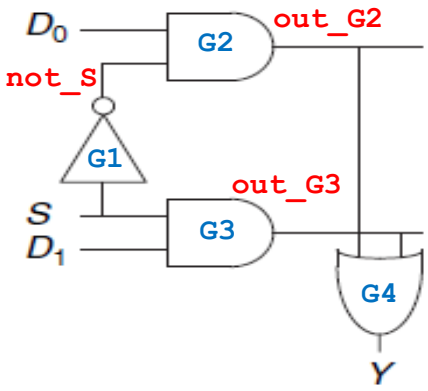
   Objective: a seven-segment run on FPGA (Here).

   Refer to assignment 2, to review the logic equations of the seven-segment decoder.

Parts: -

1. Simulate 2-1 multiplexer using logic gates.
2. Implement the seven-segment decoder using logic equations and run it on the FPGA.

Part 1. Simulate 2-1 multiplexer using logic gates (Lab01).



The multiplexer chooses between the two data inputs based on the select:

if S = 0, Y =D0, and if S = 1, Y =D1.

Verilog code for multiplexer 2-to-1

```
// ------------------- Mux 2-1 ---------------------//
module mux_2_1(S,D0,D1,Y);
input S, D0, D1;
output Y;
wire not_S, out_G2, out_G3;
notGate G1(S, not_S);
andGate G2(D0, not_S, out_G2);
andGate G3(D1, S, out_G3);
orGate G4(out_G2, out_G3, Y);
endmodule
// ------------------- AND Gate ---------------------//
module andGate (a,b,c);
input a, b;
output c;
assign c = a & b;
endmodule
```

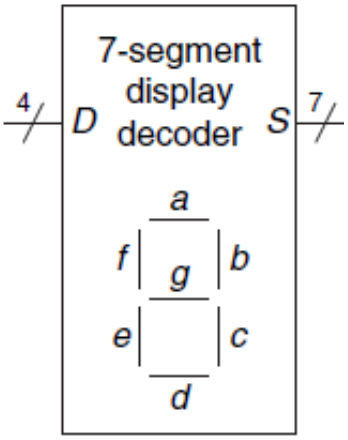

```
// ------------------- Or Gate ---------------------//
module orGate (a,b,c);
input a, b;
output c;
assign c = a | b;
endmodule
// ------------------- Not Gate ---------------------//
module notGate (a,c);
input a;
output c;
assign c = ~a;
endmodule
```

Testbench of the multiplexer 2-to-1

```
module mux_tb;
  reg s, d0, d1;
  wire y;
  mux_2_1 mux_dut(s,d0,d1,y);
initial
  begin
    s= 0; d0 = 0; d1 =0;
    #10 s=0; d0=0; d1=1; // select d0=0
    #10 s=0; d0=1; d1=0; // select d0=1
    #10 s=1; d0=0; d1=1; // select d1=1
    #10 s=1; d0=1; d1=0; // select d1=0
  end
endmodule
```

Part 2. Implement the seven-segment decoder using logic equations.

A seven-segment display decoder takes a 4-bit data input A, B, C, D and produces seven outputs to control light-emitting diodes to display a digit from 0 to 9. The seven outputs are often called segments a through g, as defined in Figure.



The truth table of the seven-segment decoder:

Digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

The logic equations of the seven-segment decoder

$$a = A + C + BD + \overline{B}\,\overline{D}$$

$$b = \overline{B} + \overline{C}\,\overline{D} + CD$$

$$c = B + \overline{C} + D$$

$$d = \overline{B}\,\overline{D} + C\,\overline{D} + B\,\overline{C}\,D + \overline{B}\,C + A$$

$$e = \overline{B}\,\overline{D} + C\,\overline{D}$$

$$f = A + \overline{C}\,\overline{D} + B\,\overline{C} + B\,\overline{D}$$

$$g = A + B\,\overline{C} + \overline{B}\,C + C\,\overline{D}$$

- **Bitwise operators**

&	Bitwise and / reduction and
|	Bitwise or / reduction or
^	Bitwise xor / reduction xor
~	Bitwise not

Verilog code for Decoder to 7 segments

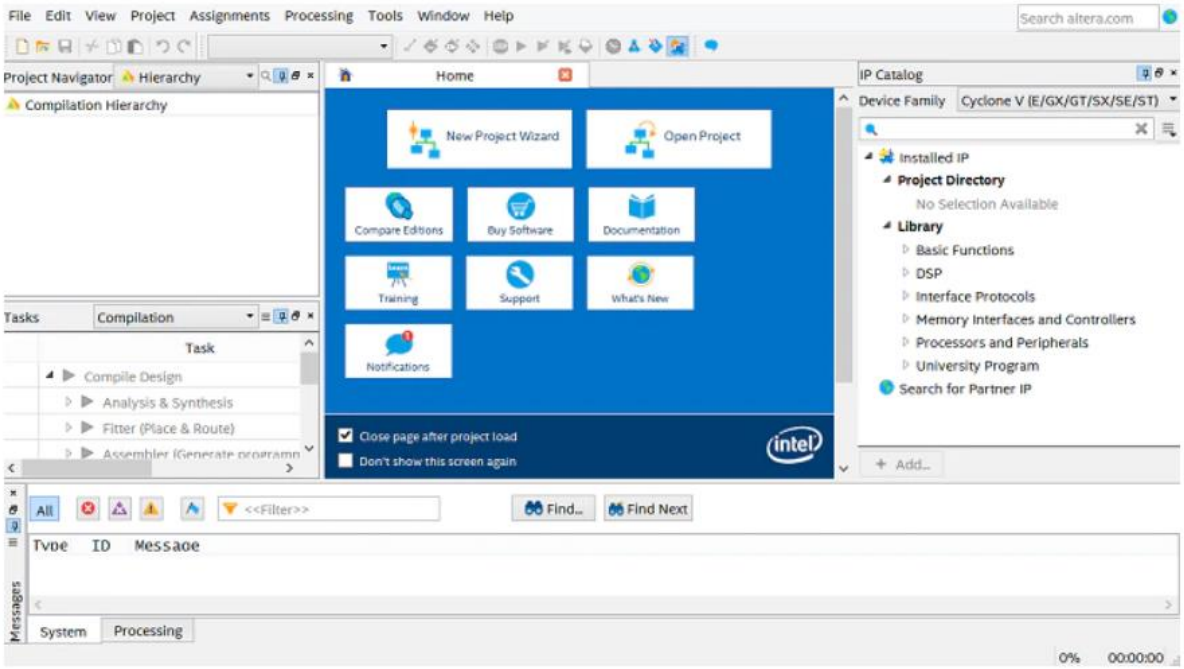
```
module decoder_7seg (A, B, C, D, led_a, led_b, led_c,
led_d, led_e, led_f, led_g);
      input A, B, C, D;
      output led_a, led_b, led_c, led_d, led_e, led_f,
             led_g;

      assign led_a = ~(A | C | B&D | ~B&~D);
      assign led_b = ~(~B | ~C&~D | C&D);
      assign led_c = ~(B | ~C | D);
      assign led_d = ~(~B&~D | C&~D | B&~C&D | ~B&C |A);
      assign led_e = ~(~B&~D | C&~D);
      assign led_f = ~(A | ~C&~D | B&~C | B&~D);
      assign led_g = ~(A | B&~C | ~B&C | C&~D);
endmodule
```

Run the seven-segment decoder on FPGA.

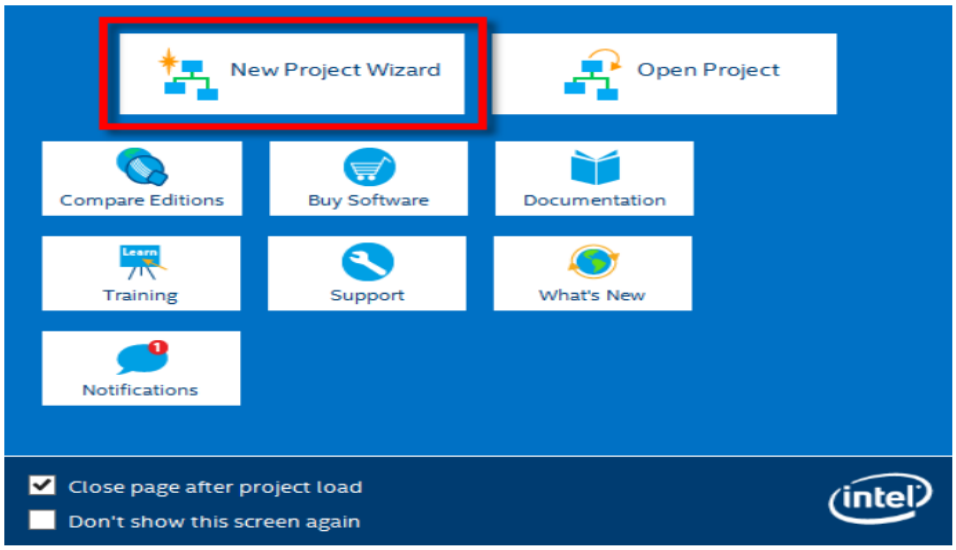
We will use a DE-10lite kit, Altera MAX 10 based FPGA board, check here. Check DE10-lite user manual (Here).
You will use **Quartus** to program the FPGA.

Quartus – Seven Segment Decoder Project Steps

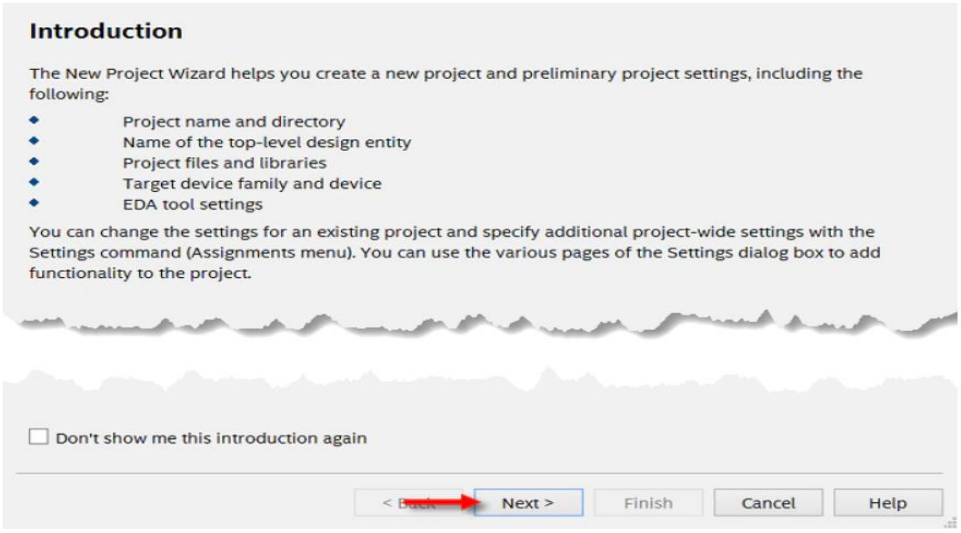
Step 1: Open Quartus.



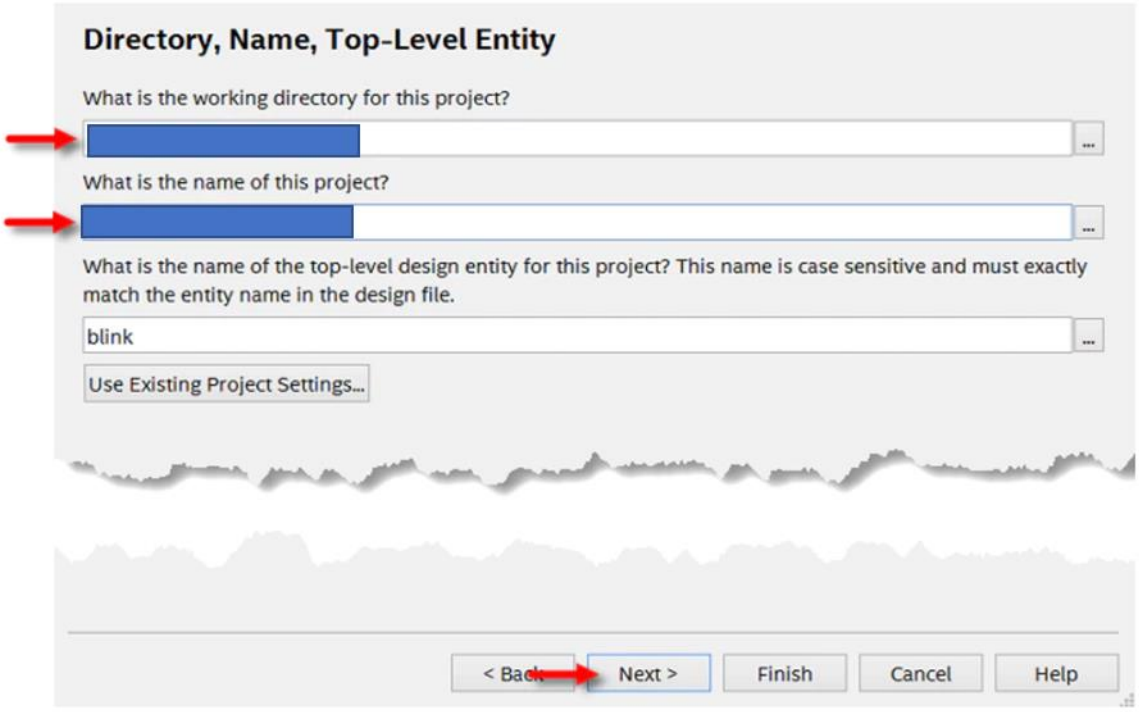
Step 2: Open a New Project Wizard.



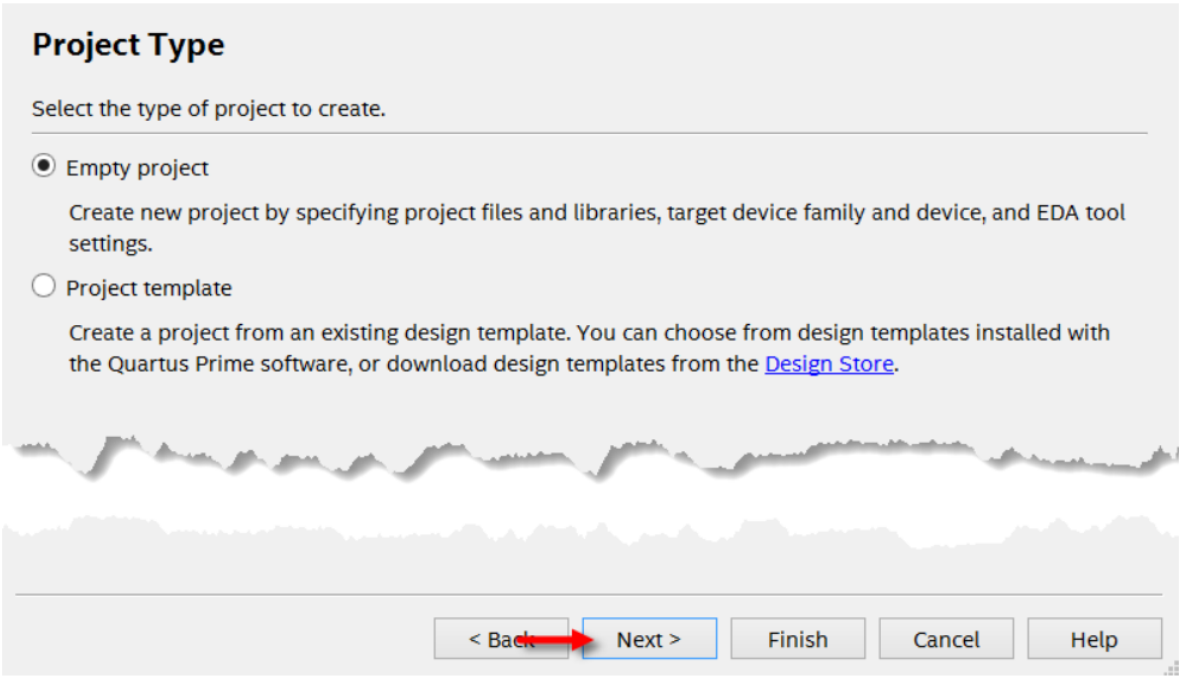
Step 3: Select Next



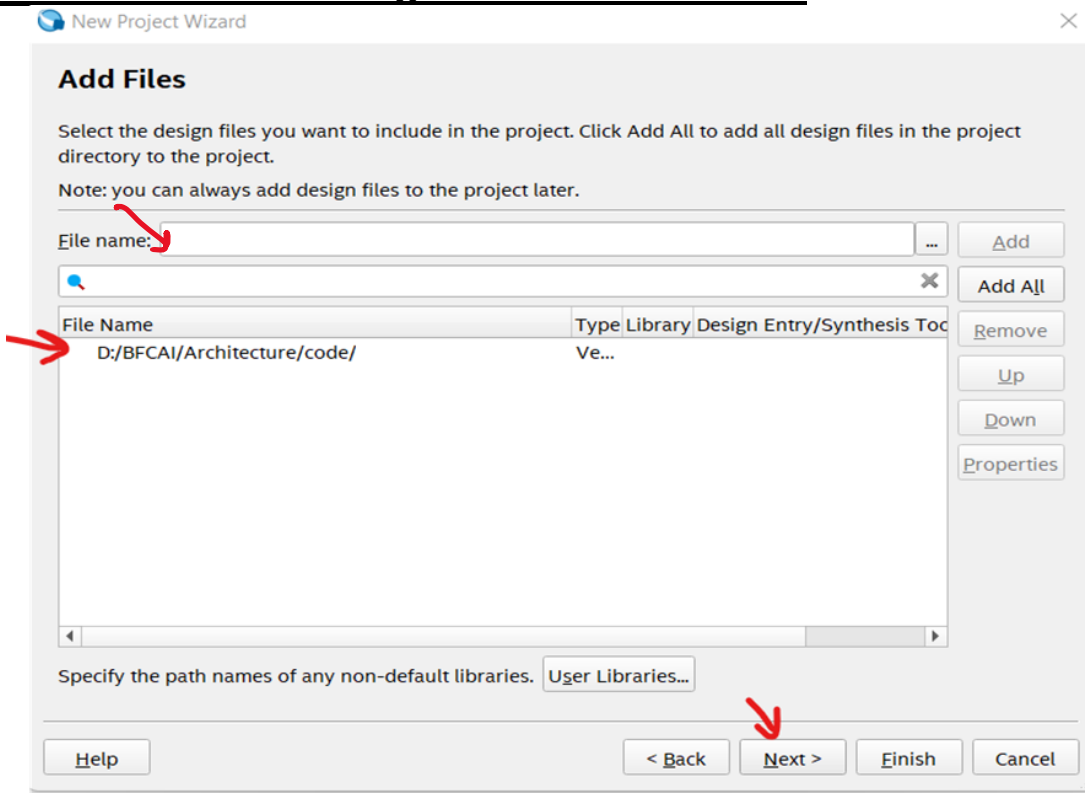
Step 4: Choose a directory to put your project under. you can place it wherever you want. Name the project as the name of the top-level module. You will name it **"decoder_7seg",** Select Next.



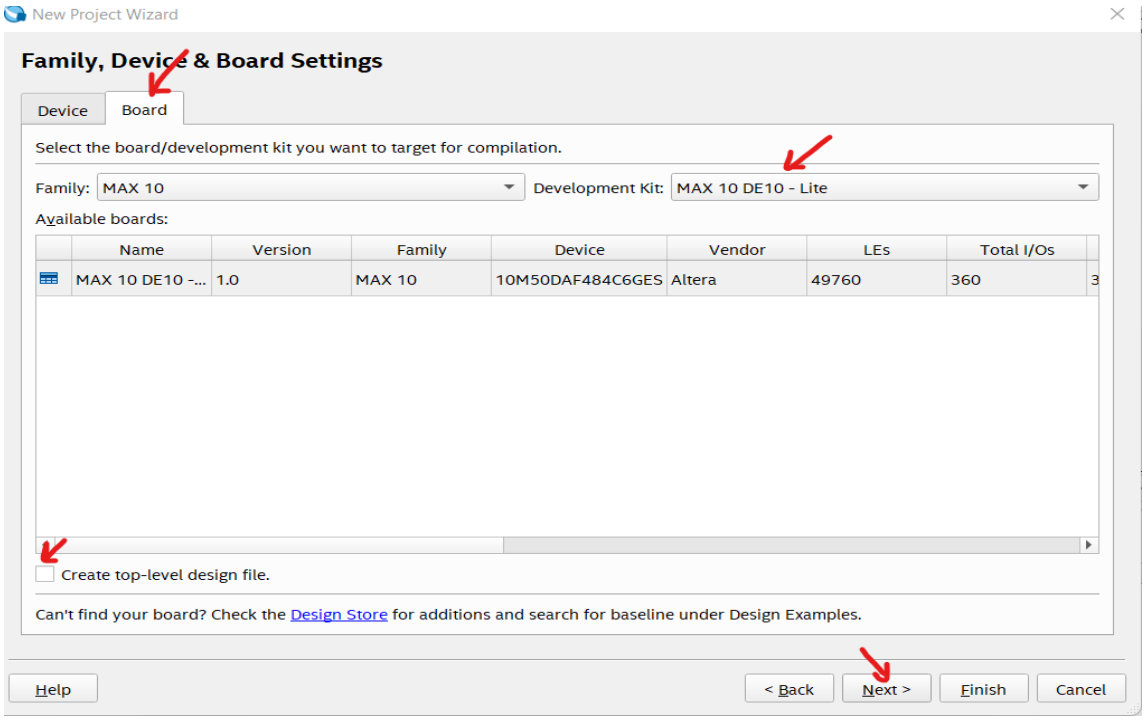
Step5: Select Empty Project, and then click Next.



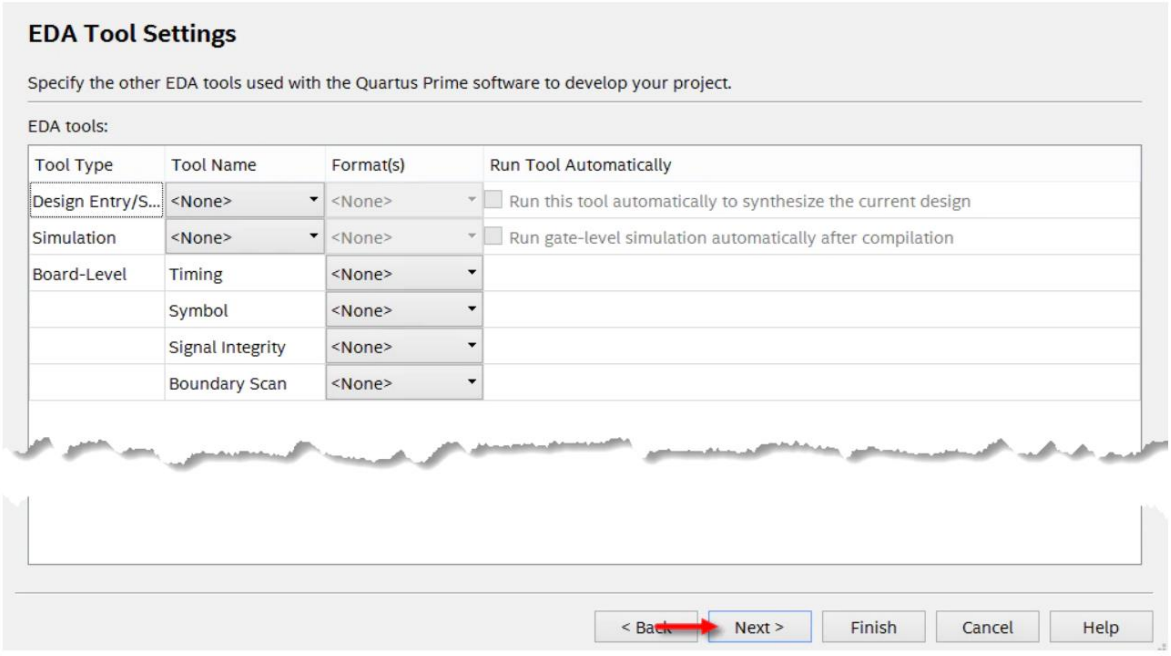
Step 6: If You want to add any files here. **Add the seven-segment decoder Verilog file And Click Next**.



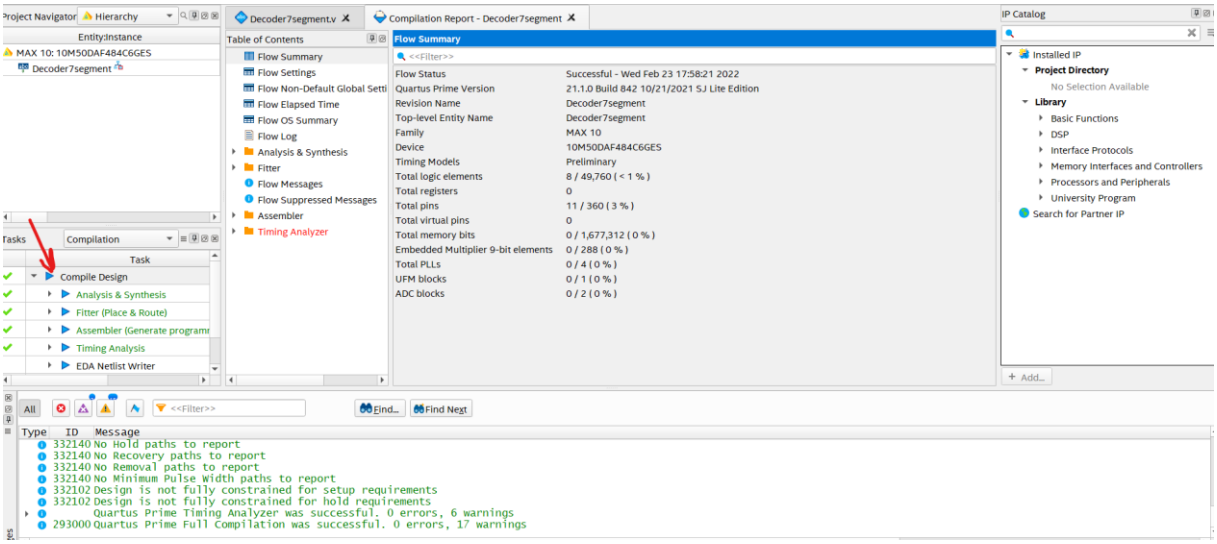
Step7: Board Settings, select our board **DE10-lite**, **unmark “create top-level design file”**



Step 8: EDA Tool Settings then finish.

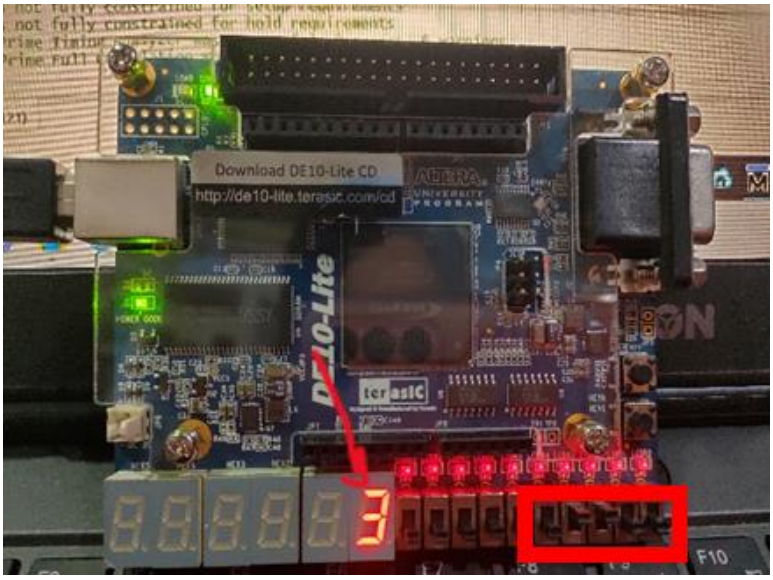


Step 9: compile the design



Step 10: Pin assignment on FPGA

In this step, you will assign inputs (A,B,C,D) to the switches, And the outputs (led_a, led_b, led_c, led_d, led_e, led_f, led_g) to the first segment display.



To assign pins, refer to DE10-lite FPGA user manual.

User-Defined Slide Switch Section

Figure 3-15 Connections between the slide switches and MAX 10 FPGA

Table 3-4 Pin Assignment of Slide Switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTL

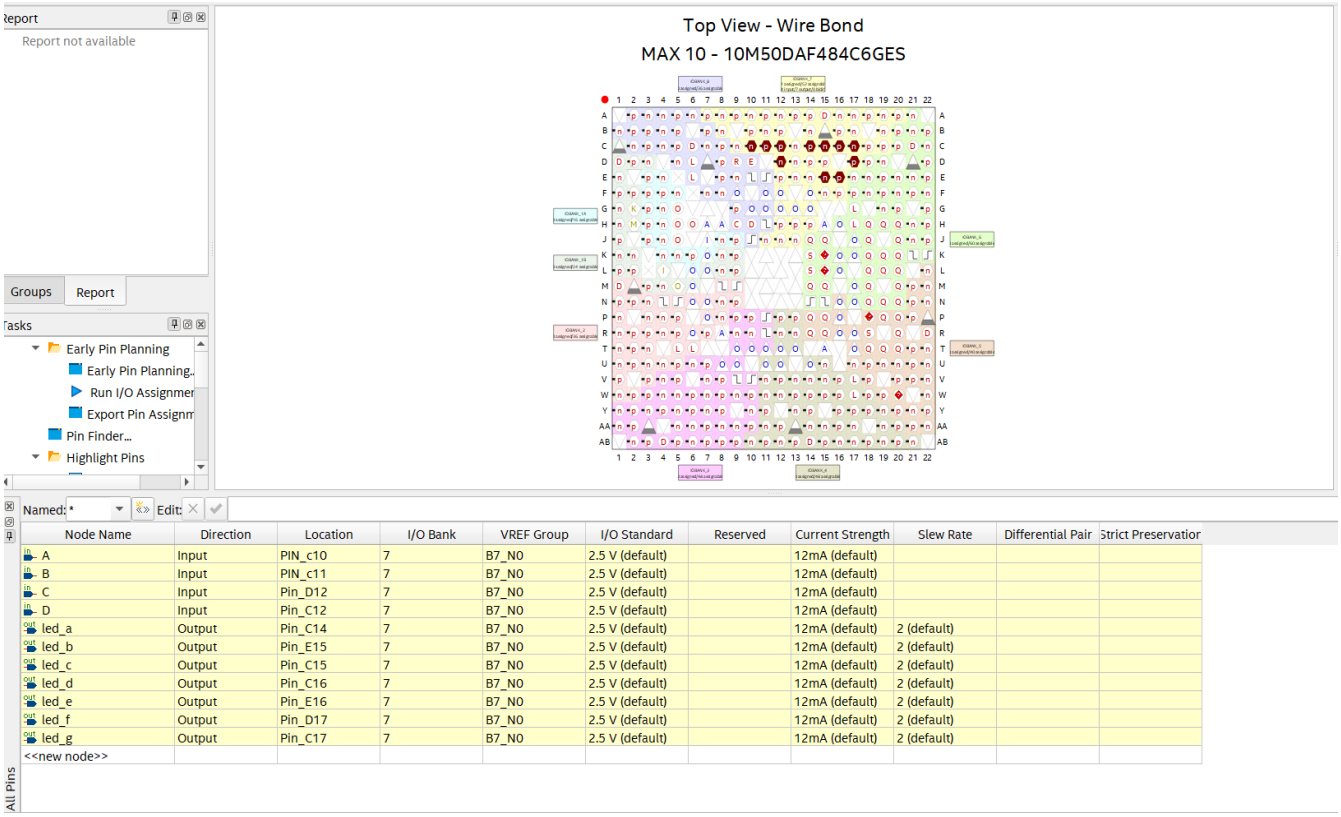
7-segment displays section

Figure 3-17 Connections between the 7-segment display HEX0 and the MAX 10 FPGA

Table 3-6 Pin Assignment of 7-segment Displays

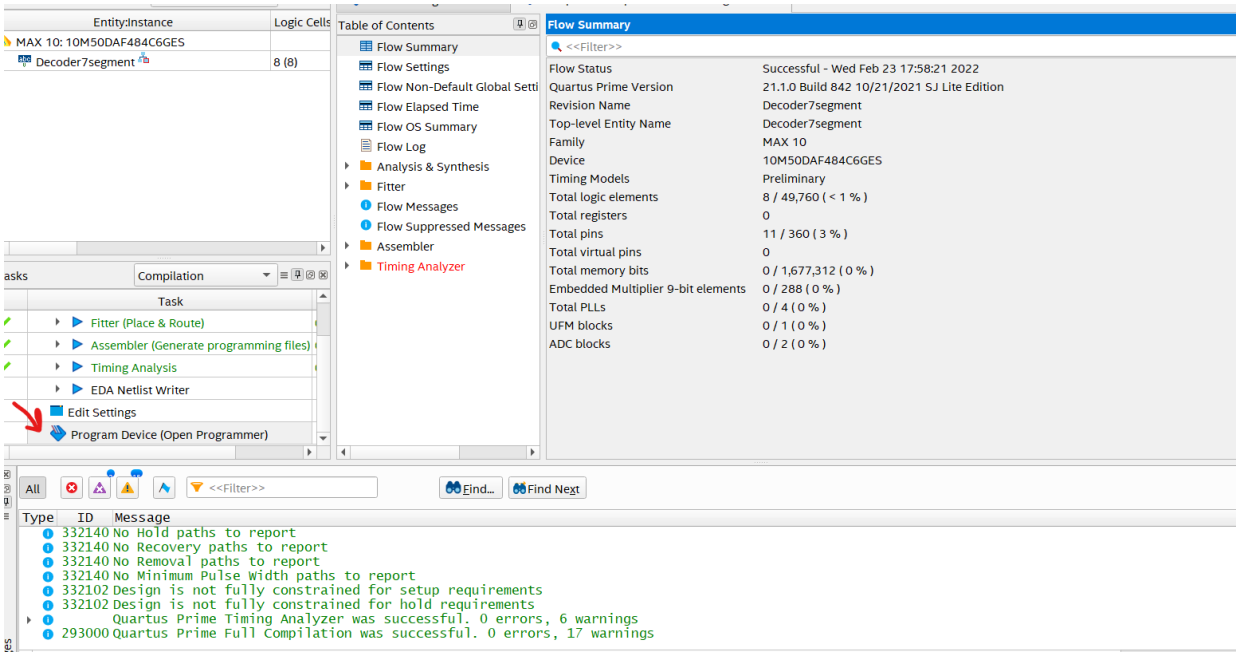
Signal Name	FPGA Pin No.	Description	I/O Standard
HEX00	PIN_C14	Seven Segment Digit 0[0]	3.3-V LVTTL
HEX01	PIN_E15	Seven Segment Digit 0[1]	3.3-V LVTTL
HEX02	PIN_C15	Seven Segment Digit 0[2]	3.3-V LVTTL
HEX03	PIN_C16	Seven Segment Digit 0[3]	3.3-V LVTTL
HEX04	PIN_E16	Seven Segment Digit 0[4]	3.3-V LVTTL
HEX05	PIN_D17	Seven Segment Digit 0[5]	3.3-V LVTTL
HEX06	PIN_C17	Seven Segment Digit 0[6]	3.3-V LVTTL
HEX07	PIN_D15	Seven Segment Digit 0[7], DP	3.3-V LVTTL

Assign pins on Quartus, open the assignment tab, click on pin planner, and assign pins as figure below.

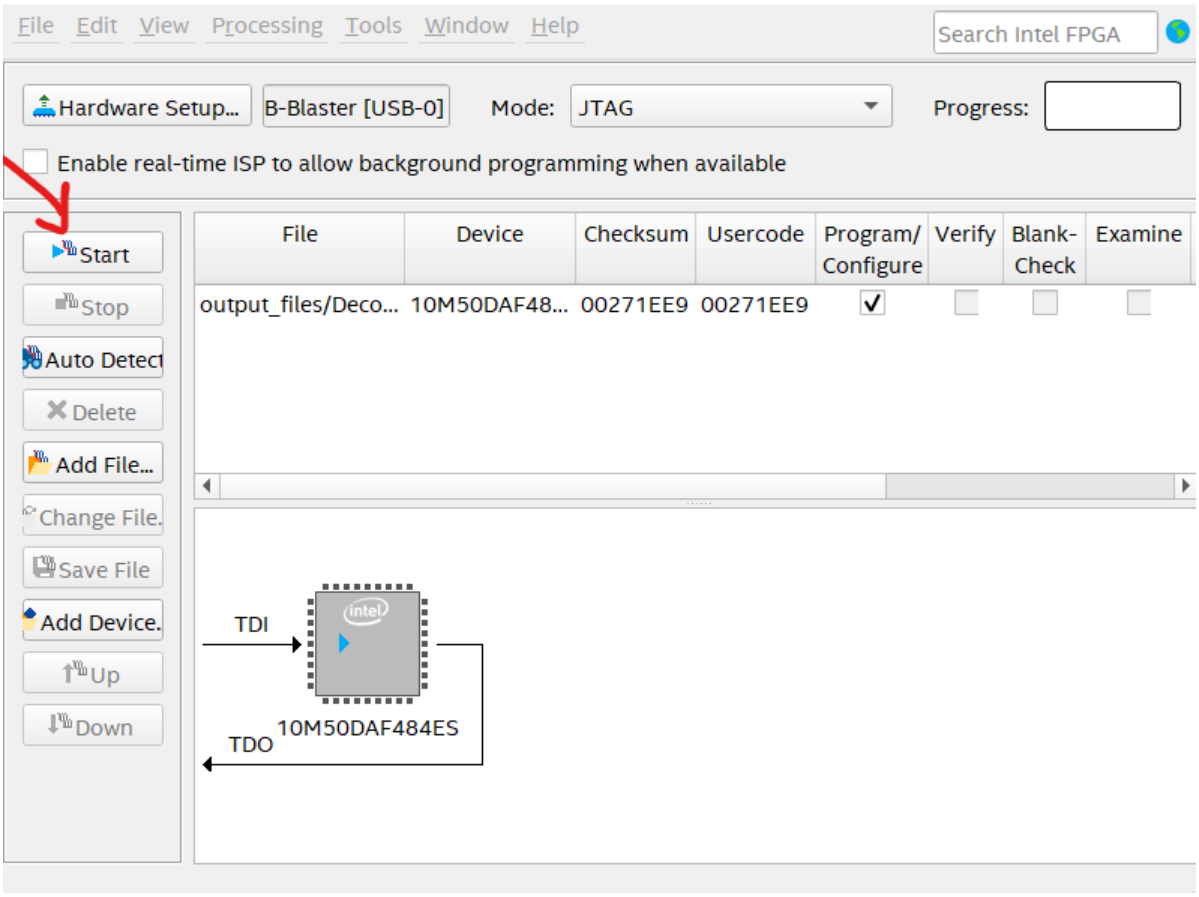


Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
A	Input	PIN_c10	7	B7_N0	2.5 V (default)		12mA (default)			
B	Input	PIN_c11	7	B7_N0	2.5 V (default)		12mA (default)			
C	Input	Pin_D12	7	B7_N0	2.5 V (default)		12mA (default)			
D	Input	Pin_C12	7	B7_N0	2.5 V (default)		12mA (default)			
led_a	Output	Pin_C14	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_b	Output	Pin_E15	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_c	Output	Pin_C15	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_d	Output	Pin_C16	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_e	Output	Pin_E16	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_f	Output	Pin_D17	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
led_g	Output	Pin_C17	7	B7_N0	2.5 V (default)		12mA (default)	2 (default)		
<<new node>>										

Step 11: Compile all project after pin assignment, like step 9. Program the FPGA.
Step 12: load the program to the FPGA.



Press “Start”
Hint: you may have a problem with the FPGA driver. Check the “device manager” and update the USB driver.



😊😊😊