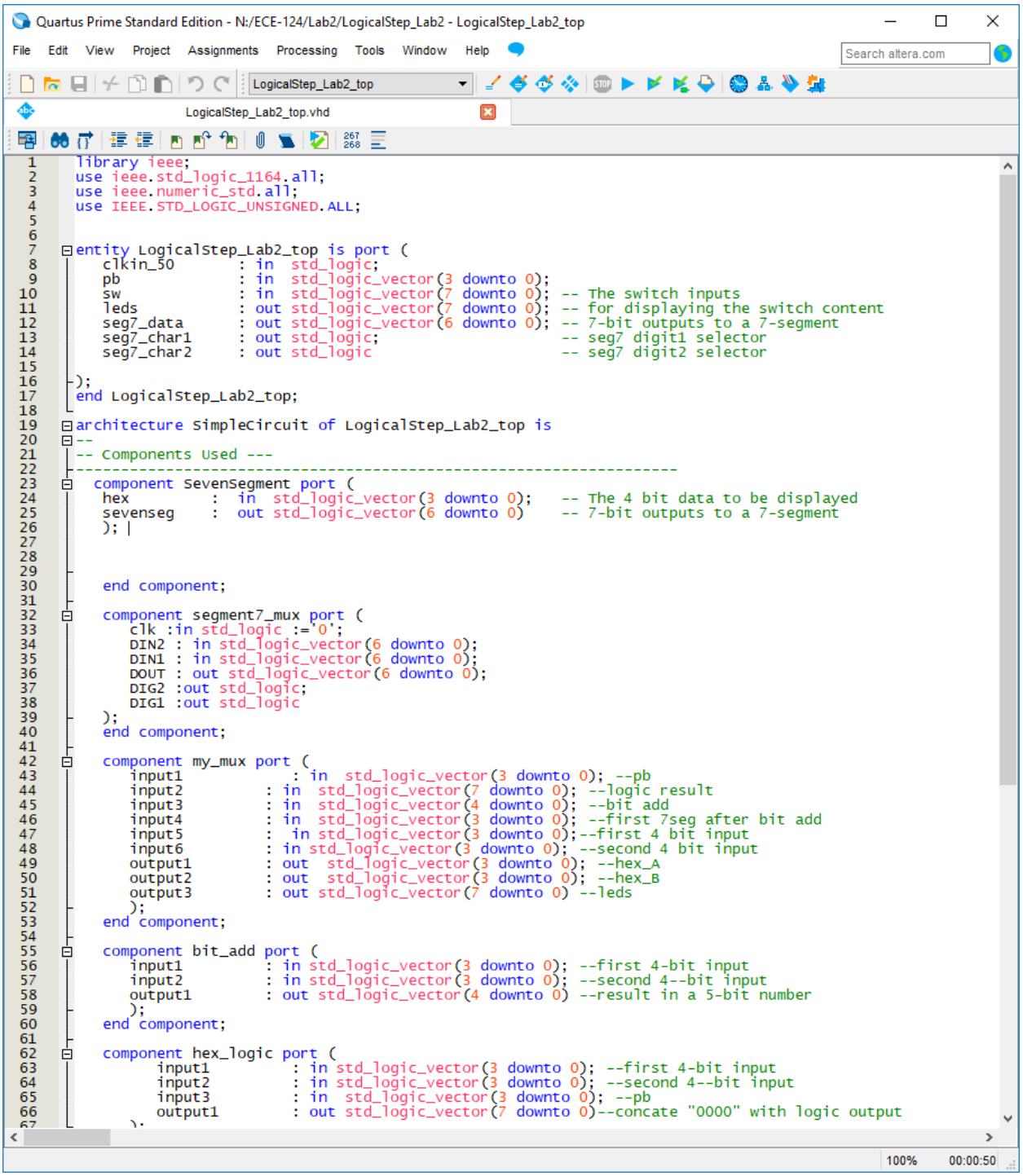
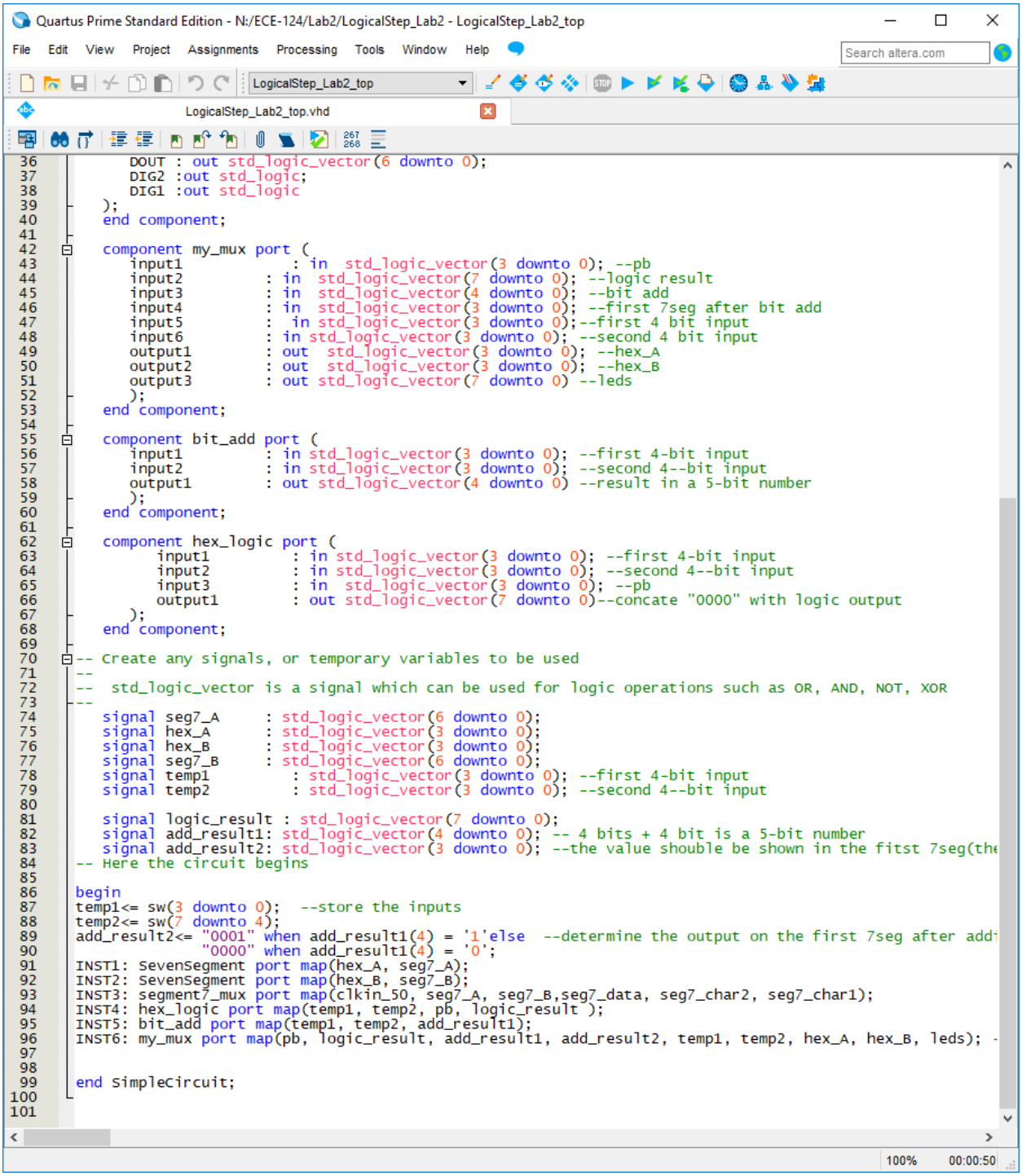
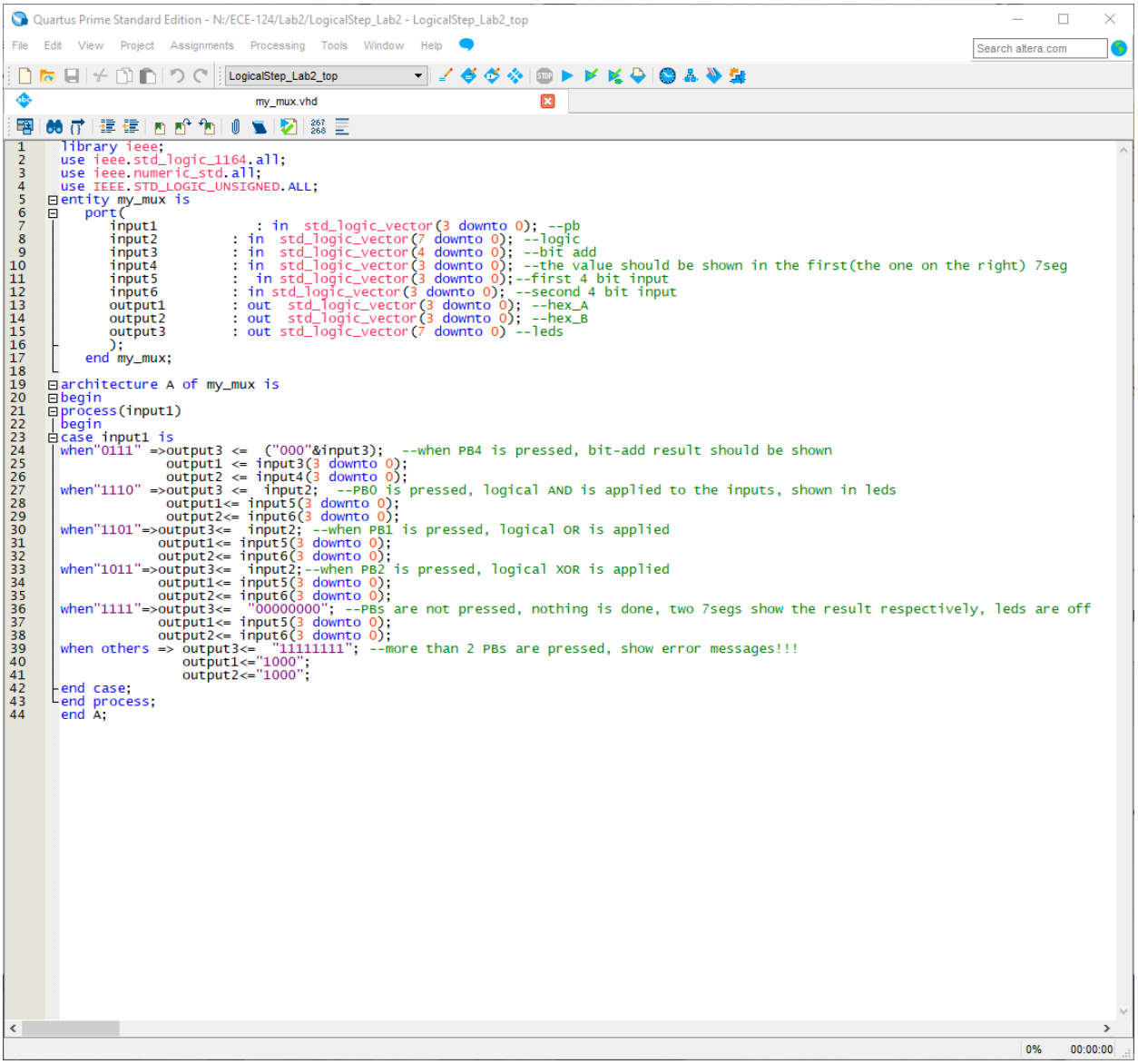
Structural VHDL used in top level VHDL design




```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity LogicalStep_Lab2_top is port (
   clkin_50       : in  std_logic;
   pb             : in  std_logic_vector(3 downto 0);
   sw             : in  std_logic_vector(7 downto 0); -- The switch inputs
   leds           : out std_logic_vector(7 downto 0); -- for displaying the switch content
   seg7_data      : out std_logic_vector(6 downto 0); -- 7-bit outputs to a 7-segment
   seg7_char1     : out std_logic;                    -- seg7 digit1 selector
   seg7_char2     : out std_logic                     -- seg7 digit2 selector

);
end LogicalStep_Lab2_top;

architecture SimpleCircuit of LogicalStep_Lab2_top is
--
-- Components Used ---
------------------------------------------------------------------------
  component SevenSegment port (
   hex         :  in  std_logic_vector(3 downto 0);   -- The 4 bit data to be displayed
   sevenseg    :  out std_logic_vector(6 downto 0)    -- 7-bit outputs to a 7-segment
   );


   end component;

   component segment7_mux port (
      clk :in std_logic :='0';
      DIN2 : in std_logic_vector(6 downto 0);
      DIN1 : in std_logic_vector(6 downto 0);
      DOUT : out std_logic_vector(6 downto 0);
      DIG2 :out std_logic;
      DIG1 :out std_logic
   );
   end component;

   component my_mux port (
      input1          : in  std_logic_vector(3 downto 0); --pb
      input2          : in  std_logic_vector(7 downto 0); --logic result
      input3          : in  std_logic_vector(4 downto 0); --bit add
      input4          : in  std_logic_vector(3 downto 0); --first 7seg after bit add
      input5          :  in std_logic_vector(3 downto 0);--first 4 bit input
      input6          : in std_logic_vector(3 downto 0); --second 4 bit input
      output1         : out  std_logic_vector(3 downto 0); --hex_A
      output2         : out  std_logic_vector(3 downto 0); --hex_B
      output3         : out std_logic_vector(7 downto 0) --leds
      );
   end component;

   component bit_add port (
      input1          : in std_logic_vector(3 downto 0); --first 4-bit input
      input2          : in std_logic_vector(3 downto 0); --second 4--bit input
      output1         : out std_logic_vector(4 downto 0) --result in a 5-bit number
      );
   end component;

   component hex_logic port (
         input1          : in std_logic_vector(3 downto 0); --first 4-bit input
         input2          : in std_logic_vector(3 downto 0); --second 4--bit input
         input3          : in  std_logic_vector(3 downto 0); --pb
         output1         : out std_logic_vector(7 downto 0)--concate "0000" with logic output
      );
```

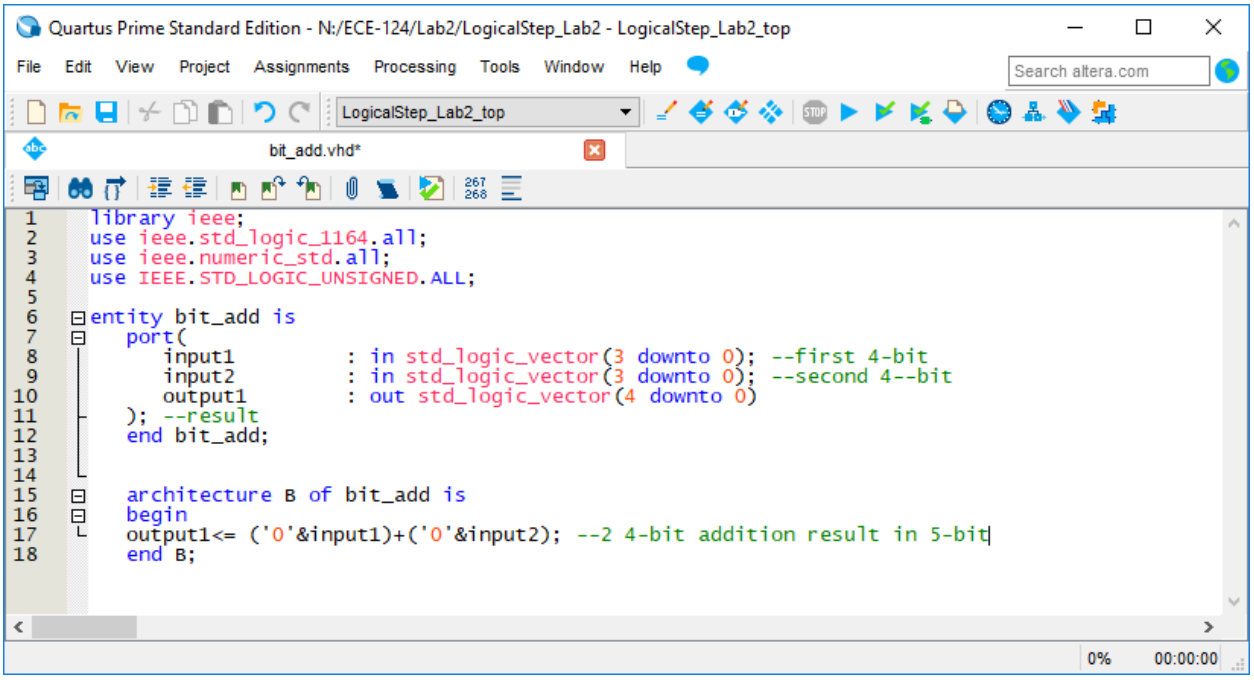

Quartus Prime Standard Edition - N:/ECE-124/Lab2/LogicalStep_Lab2 - LogicalStep_Lab2_top
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
LogicalStep_Lab2_top
LogicalStep_Lab2_top.vhd
DOUT : out std_logic_vector(6 downto 0);
DIG2 :out std_logic;
DIG1 :out std_logic
);
end component;
component my_mux port (
input1 : in std_logic_vector(3 downto 0); --pb
input2 : in std_logic_vector(7 downto 0); --logic result
input3 : in std_logic_vector(4 downto 0); --bit add
input4 : in std_logic_vector(3 downto 0); --first 7seg after bit add
input5 : in std_logic_vector(3 downto 0);--first 4 bit input
input6 : in std_logic_vector(3 downto 0); --second 4 bit input
output1 : out std_logic_vector(3 downto 0); --hex_A
output2 : out std_logic_vector(3 downto 0); --hex_B
output3 : out std_logic_vector(7 downto 0) --leds
);
end component;
component bit_add port (
input1 : in std_logic_vector(3 downto 0); --first 4-bit input
input2 : in std_logic_vector(3 downto 0); --second 4--bit input
output1 : out std_logic_vector(4 downto 0) --result in a 5-bit number
);
end component;
component hex_logic port (
input1 : in std_logic_vector(3 downto 0); --first 4-bit input
input2 : in std_logic_vector(3 downto 0); --second 4--bit input
input3 : in std_logic_vector(3 downto 0); --pb
output1 : out std_logic_vector(7 downto 0)--concate "0000" with logic output
);
end component;
-- Create any signals, or temporary variables to be used
--
-- std_logic_vector is a signal which can be used for logic operations such as OR, AND, NOT, XOR
--
signal seg7_A : std_logic_vector(6 downto 0);
signal hex_A : std_logic_vector(3 downto 0);
signal hex_B : std_logic_vector(3 downto 0);
signal seg7_B : std_logic_vector(6 downto 0);
signal temp1 : std_logic_vector(3 downto 0); --first 4-bit input
signal temp2 : std_logic_vector(3 downto 0); --second 4--bit input
signal logic_result : std_logic_vector(7 downto 0);
signal add_result1: std_logic_vector(4 downto 0); -- 4 bits + 4 bit is a 5-bit number
signal add_result2: std_logic_vector(3 downto 0); --the value shouble be shown in the fitst 7seg(the
-- Here the circuit begins
begin
temp1<= sw(3 downto 0); --store the inputs
temp2<= sw(7 downto 4);
add_result2<= "0001" when add_result1(4) = '1'else --determine the output on the first 7seg after add
"0000" when add_result1(4) = '0';
INST1: SevenSegment port map(hex_A, seg7_A);
INST2: SevenSegment port map(hex_B, seg7_B);
INST3: segment7_mux port map(clkin_50, seg7_A, seg7_B,seg7_data, seg7_char2, seg7_char1);
INST4: hex_logic port map(temp1, temp2, pb, logic_result );
INST5: bit_add port map(temp1, temp2, add_result1);
INST6: my_mux port map(pb, logic_result, add_result1, add_result2, temp1, temp2, hex_A, hex_B, leds);
end SimpleCircuit;
100%
00:00:50

Sub-block my_mux

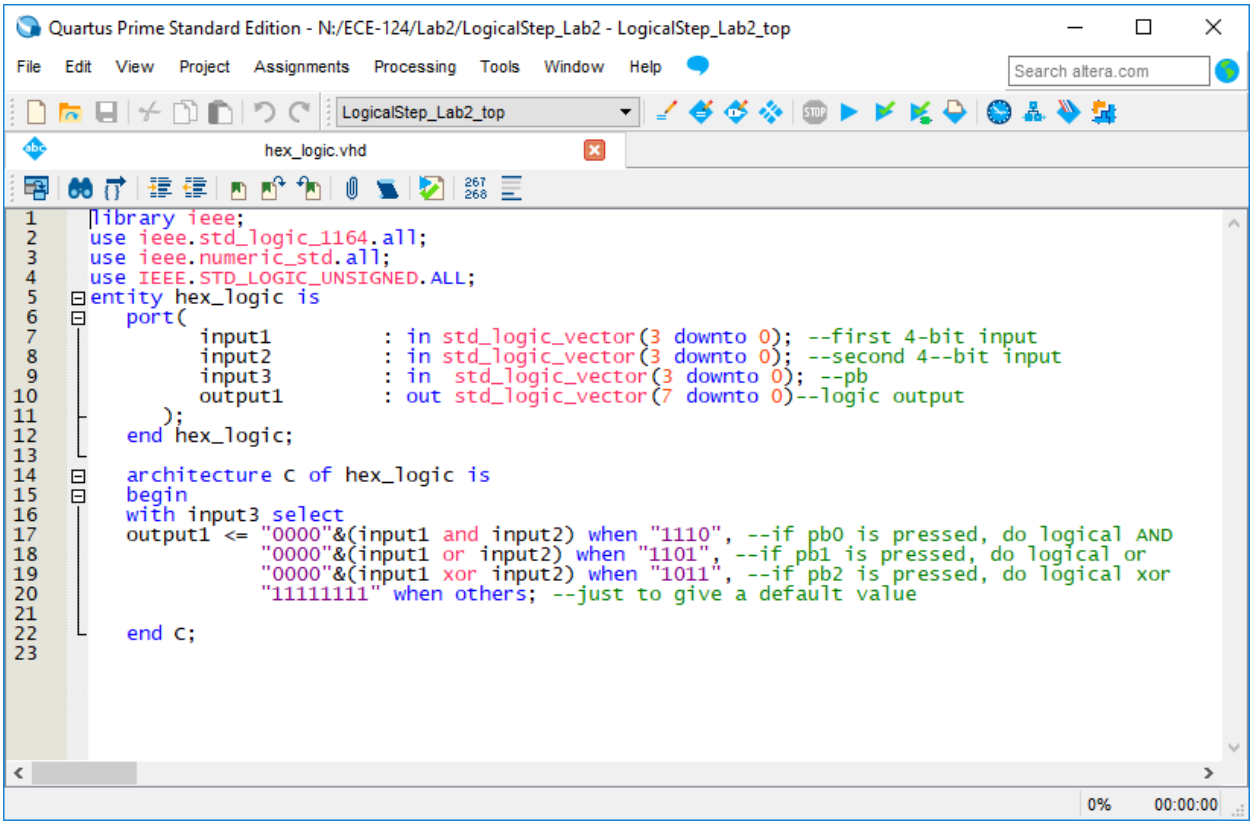



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity my_mux is
    port(
        input1             : in  std_logic_vector(3 downto 0); --pb
        input2          : in  std_logic_vector(7 downto 0); --logic
        input3          : in  std_logic_vector(4 downto 0); --bit add
        input4          : in  std_logic_vector(3 downto 0); --the value should be shown in the first(the one on the right) 7seg
        input5          :  in std_logic_vector(3 downto 0);--first 4 bit input
        input6          : in std_logic_vector(3 downto 0); --second 4 bit input
        output1         : out  std_logic_vector(3 downto 0); --hex_A
        output2         : out  std_logic_vector(3 downto 0); --hex_B
        output3         : out std_logic_vector(7 downto 0) --leds
        );
    end my_mux;

architecture A of my_mux is
begin
process(input1)
 begin
case input1 is
 when"0111" =>output3 <=  ("000"&input3);  --when PB4 is pressed, bit-add result should be shown
              output1 <= input3(3 downto 0);
              output2 <= input4(3 downto 0);
 when"1110" =>output3 <=  input2;  --PB0 is pressed, logical AND is applied to the inputs, shown in leds
              output1<= input5(3 downto 0);
              output2<= input6(3 downto 0);
 when"1101"=>output3<=  input2; --when PB1 is pressed, logical OR is applied
             output1<= input5(3 downto 0);
             output2<= input6(3 downto 0);
 when"1011"=>output3<=  input2;--when PB2 is pressed, logical XOR is applied
             output1<= input5(3 downto 0);
             output2<= input6(3 downto 0);
 when"1111"=>output3<=  "00000000"; --PBs are not pressed, nothing is done, two 7segs show the result respectively, leds are off
             output1<= input5(3 downto 0);
             output2<= input6(3 downto 0);
 when others => output3<=  "11111111"; --more than 2 PBs are pressed, show error messages!!!
                output1<="1000";
                output2<="1000";
end case;
end process;
 end A;
```

Sub-block bit_add

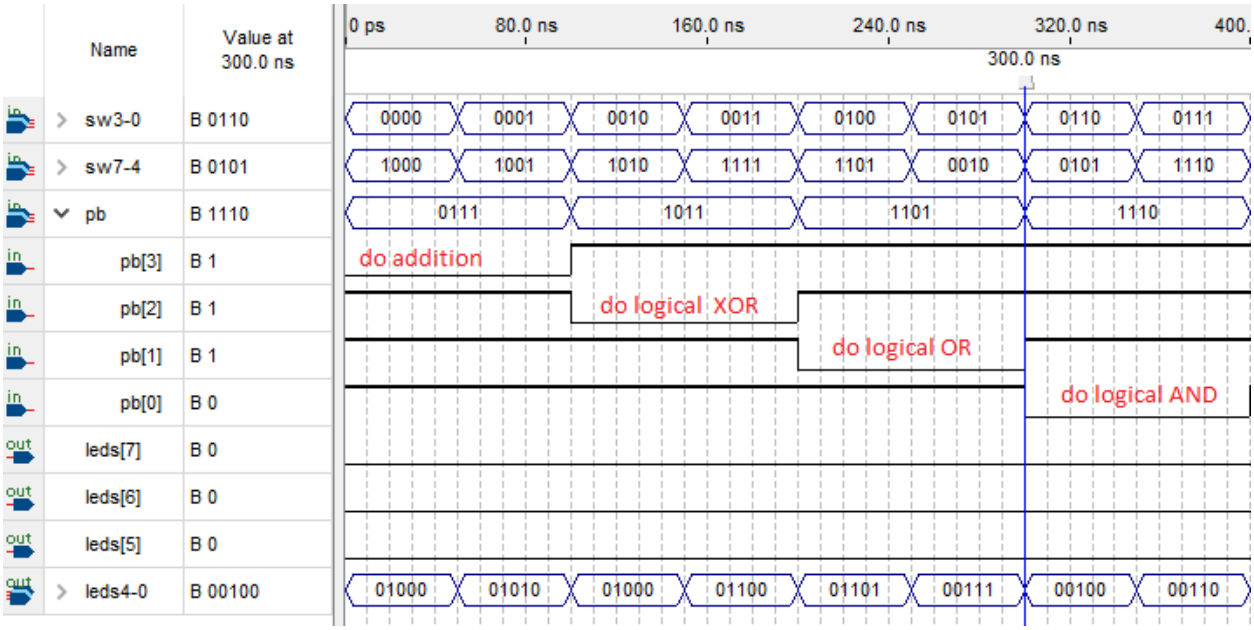


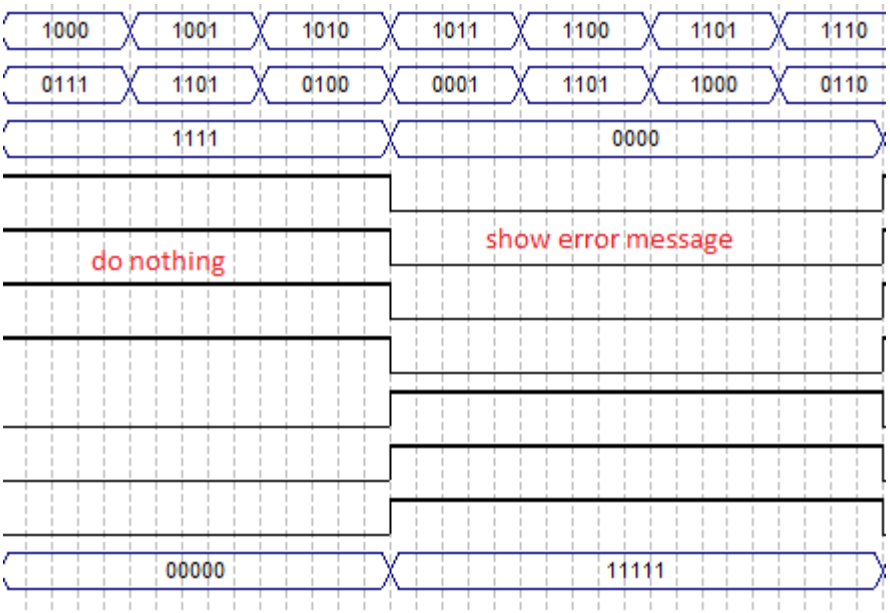
Sub-block hex_logic




```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5  entity hex_logic is
6     port(
7           input1         : in std_logic_vector(3 downto 0); --first 4-bit input
8           input2         : in std_logic_vector(3 downto 0); --second 4--bit input
9           input3         : in  std_logic_vector(3 downto 0); --pb
10          output1        : out std_logic_vector(7 downto 0)--logic output
11       );
12    end hex_logic;
13
14    architecture c of hex_logic is
15    begin
16    with input3 select
17    output1 <= "0000"&(input1 and input2) when "1110", --if pb0 is pressed, do logical AND
18               "0000"&(input1 or input2) when "1101", --if pb1 is pressed, do logical or
19               "0000"&(input1 xor input2) when "1011", --if pb2 is pressed, do logical xor
20               "11111111" when others; --just to give a default value
21
22    end c;
23
```

Simulation


Name
Value at 300.0 ns
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
300.0 ns
sw3-0
B 0110
0000 0001 0010 0011 0100 0101 0110 0111
sw7-4
B 0101
1000 1001 1010 1111 1101 0010 0101 1110
pb
B 1110
0111 1011 1101 1110
pb[3]
B 1
do addition
pb[2]
B 1
do logical XOR
pb[1]
B 1
do logical OR
pb[0]
B 0
do logical AND
leds[7]
B 0
leds[6]
B 0
leds[5]
B 0
leds4-0
B 00100
01000 01010 01000 01100 01101 00111 00100 00110


1000 1001 1010 1011 1100 1101 1110
0111 1101 0100 0001 1101 1000 0110
1111 0000
show error message
do nothing
00000 11111

Compilation Report

| Flow Summary | |
|---|---|
| Flow Status | Successful - Thu Jun 13 09:52:16 2019 |
| Quartus Prime Version | 15.1.0 Build 185 10/21/2015 SJ Standard Edition |
| Revision Name | LogicalStep_Lab2_top |
| Top-level Entity Name | LogicalStep_Lab2_top |
| Family | MAX 10 |
| Device | 10M08SAE144C8G |
| Timing Models | Final |
| Total logic elements | 72 / 8,064 ( < 1 % ) |
| Total combinational functions | 72 / 8,064 ( < 1 % ) |
| Dedicated logic registers | 11 / 8,064 ( < 1 % ) |
| Total registers | 11 |
| Total pins | 30 / 101 ( 30 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 387,072 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 48 ( 0 % ) |
| Total PLLs | 0 / 1 ( 0 % ) |
| UFM blocks | 0 / 1 ( 0 % ) |
| ADC blocks | 0 / 1 ( 0 % ) |