ECE-124 Lab-2 Submission Form – Winter 2018

GROUP NUMBER: 7			[illegible]
SESSION NUMBER: 201			
NAME: (Print)	**UW User ID** **(not Student ID)**	**Signature**	
Partner A: SATISH .B	sboddu	[illegible]	
Partner B: Jiaxi Wang	j949wang	[illegible]	

LAB2 DESIGN DEMO	**Marks Allotted**	**A**	**B**
Seven Segment Display bugs (quantity 3) corrected ?	1	1	1
Operands appear on Digit1 & Digit2 when PB's are OFF ?	1	1	1
Logical Results shown correctly on LEDs[3..0] when PB[2..0] ON ?	1	1	1
Arithmetic results shown on Digits and LED's when PB(3) ON ?	2	2	2
LEDs[7..4] OFF when Arithmetic result Less than or Equal to 1111	2	2	2
DISCUSSION: Describe how you implemented the VHDL coding.	3	3	0
LAB2 DEMO MARK		10	7

LAB2 DESIGN REPORT (see rubric on LEARN for details)	**Marks Allotted**	
Structural VHDL Used in top level VHDL design	2	
Sub-block VHDL files with good Coding Style	2	
Simulation of Logic functions showing the AND,OR,XOR modes	2	
Simulation of Arithmetic functions showing the ADD mode	2	
Total Design Logic Elements Used from Compilation Report	2	
Delay in Report Submission (-1 per day) x number of days:		
LAB2 Report MARK	**Out of 10**	

LAB 2 GRP 7 SESS 201 REPORT

1. Main File:

LogicalStep_Lab2_ top.vhd file:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity LogicalStep_Lab2_top is port (
   clkin_50                          : in      std_logic;
   pb                                : in      std_logic_vector(3 downto 0); -- push buttons
   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);     -- outputs to the Seven Segment LED board
          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;

-- 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
--
-- Pre Defined Vectors (In Lab)
          signal seg7_A            : std_logic_vector(6 downto 0);
          signal seg7_B            : 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 add_inpA             : std_logic_vector(7 downto 0);
        signal add_inpB             : std_logic_vector(7 downto 0);

-- User Defined Vectors (Post Lab)
        signal sum                  : std_logic_vector(7 downto 0);
        signal concate              : std_logic_vector(7 downto 0);
        signal arimhex_A            : std_logic_vector(3 downto 0);
        signal arimhex_B            : std_logic_vector(3 downto 0);
        signal arim_output          :std_logic_vector(7 downto 0);
        signal AND_output           : std_logic_vector(3 downto 0);
        signal OR_output            : std_logic_vector(3 downto 0);
        signal XOR_output           : std_logic_vector(3 downto 0);
        signal conclogic_output : std_logic_vector(7 downto 0);
        signal led_output           : std_logic_vector(7 downto 0);
        signal sevensegerror    : std_logic_vector(7 downto 0);
-- Here the circuit begins
begin
        hex_A <= sw(3 downto 0);
        hex_B <= sw(7 downto 4);

        concate <= hex_A & hex_B;
        add_inpA <= "0000" & hex_A;
        add_inpB <= "0000" & hex_B;
        sum <= std_logic_vector(unsigned(add_inpA)+unsigned(add_inpB));


-- ************************************************************
--LOGIC PART

        with pb(0) select
        AND_output <= hex_A AND hex_B when '0',
                                  "0000"                 when '1';

        with pb(1) select
        OR_output <= hex_A OR hex_B when '0',
                                  "0000"                 when '1';

        with pb(2) select
        XOR_output <= hex_A XOR hex_B when '0',
                                  "0000"                 when '1';


        conclogic_output <= "0000" & AND_output when pb(0) = '0' else
                                                "0000" & OR_output when pb(1) = '0' else
                                                "0000" & XOR_output when pb(2) = '0' else
                                                "00000000";
        with pb(3) select
                led_output <= sum                        when '0',
                                                    conclogic_output        when '1';

        leds (7 downto 0) <= led_output;

-- ***************************************************************
--ARITHMETIC PART

        with pb(3) select
                arim_output     <= concate                  when '1',
                                   sum                      when '0';

        arimhex_A <= arim_output(7 downto 4);
        arimhex_B <= arim_output(3 downto 0);

-- ***************************************************************
-- COMPONENT HOOKUP
-- generate the seven segment coding

        INST1: SevenSegment port map(arimhex_A, seg7_A);
        INST2: SevenSegment port map(arimhex_B, seg7_B);
        INST3: segment7_mux port map(clkin_50, seg7_A, seg7_B, seg7_data, seg7_char1, seg7_char2);
end SimpleCircuit;
```

2. Sub Files:

SevenSegment.vhd file :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
-----------------------------------------------------------------------
-- 7-segment display driver. It displays a 4-bit number on a 7-segment
-- This is created as an entity so that it can be reused many times easily

entity SevenSegment is 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 SevenSegment;

architecture Structural of SevenSegment is

-- The following statements convert a 4-bit input, called dataIn to a pattern of 7 bits
-- The segment turns on when it is '1' otherwise '0'

begin
  with hex select         --GFEDCBA       3210     -- data in
       sevenseg      <= "0111111" when "0000",    -- [0]
                        "0000110" when "0001",    -- [1]
                        "1011011" when "0010",    -- [2]      +---- a -----+
                        "1001111" when "0011",    -- [3]      |            |
                        "1100110" when "0100",    -- [4]      |            |
                        "1101101" when "0101",    -- [5]      f            b
                        "1111101" when "0110",    -- [6]      |            |
                        "0000111" when "0111",    -- [7]      |            |
                        "1111111" when "1000",    -- [8]      +---- g -----+
                        "1101111" when "1001",    -- [9]      |            |
                        "1110111" when "1010",    -- [A]      |            |
                        "1111100" when "1011",    -- [b]      e            c
                        "1011000" when "1100",    -- [c]      |            |
                        "1011110" when "1101",    -- [d]      |            |
                        "1111001" when "1110",    -- [E]      +---- d -----+
                        "1110001" when "1111",    -- [F]
                        "0000000" when others;    -- [ ]
end Structural;
--------------------------------------------------------------------
```

<u>segment7_mux.vhd</u> file:

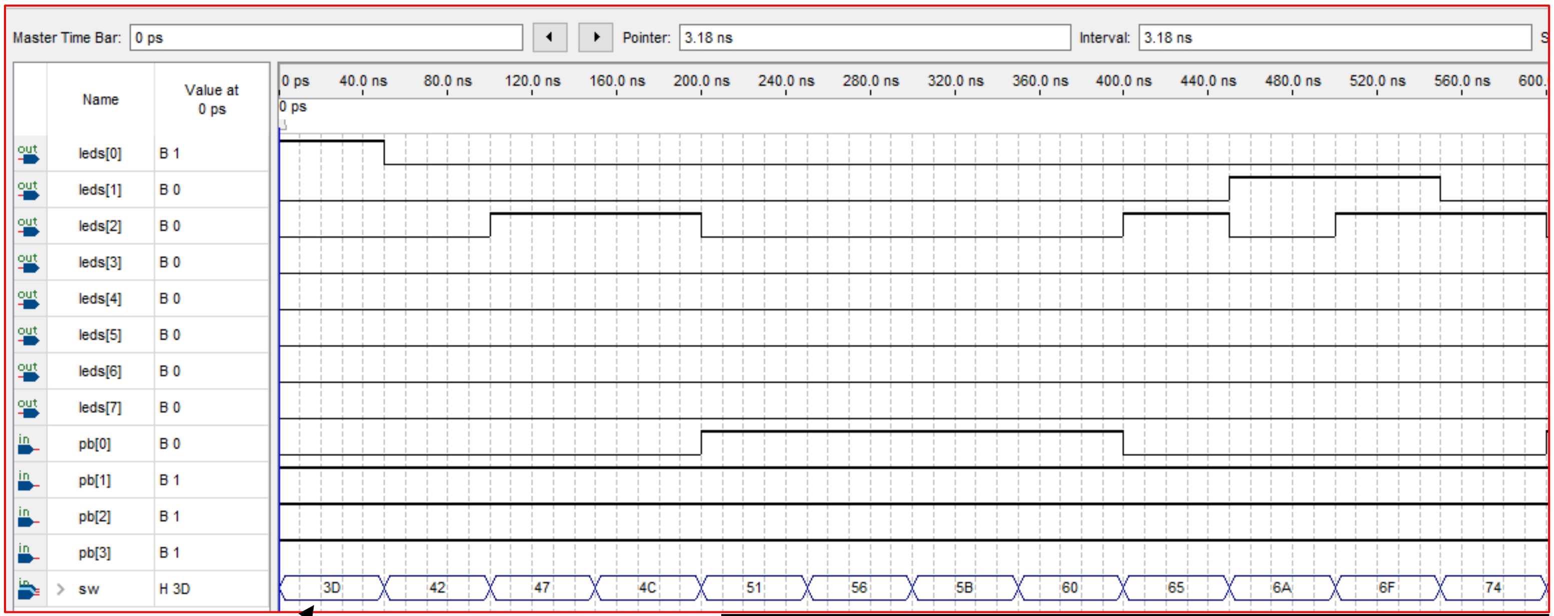
```
--****************************************************************************
--*  Copyright (C) 2016 by Trevor Smouter
--*
--*  All rights reserved.
--*
--*  Redistribution and use in source and binary forms, with or without
--*  modification, are permitted provided that the following conditions
--*  are met:
--*
--*  1. Redistributions of source code must retain the above copyright
--*     notice, this list of conditions and the following disclaimer.
--*  2. Redistributions in binary form must reproduce the above copyright
--*     notice, this list of conditions and the following disclaimer in the
--*     documentation and/or other materials provided with the distribution.
--*  3. Neither the name of the author nor the names of its contributors may
--*     be used to endorse or promote products derived from this software
--*     without specific prior written permission.
--*
--*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
--*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
--*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
--*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
--*  THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
--*  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
--*  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
--*  OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
--*  AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
--*  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
--*  THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
--*  SUCH DAMAGE.
--*
--****************************************************************************
--*  History:
--*
--*  04.02.2016    First Version
--****************************************************************************


-- ***************************************************************************
-- *  Library                                              *
-- ***************************************************************************
```

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


-- ***************************************************************************
-- *  Entity                                                 *
-- ***************************************************************************

entity segment7_mux is
  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 entity segment7_mux;

-- ****************************************************************************
-- *  Architecture                                                 *
-- ****************************************************************************

architecture syn of segment7_mux is

           signal toggle                        : std_logic;
           signal DOUT_TEMP                     : std_logic_vector(6 downto 0);

begin

  ---------------------------------------------
  -- Register File
  ---------------------------------------------



           clk_proc:process(CLK)
           variable COUNT                       :unsigned(10 downto 0) := "00000000000";
           begin
                     if (rising_edge(CLK)) then
                               COUNT := COUNT + 1;
                     else
                               COUNT := COUNT;
                     end if;
           toggle <= COUNT(10);
           end process clk_proc;
           DIG1 <= NOT toggle;
           DIG2 <= toggle;
```

```
            DOUT_TEMP(0) <= (DIN2(0)) WHEN (toggle = '1')            ELSE (DIN1(0));
            DOUT_TEMP(1) <= (DIN2(1)) WHEN (toggle = '1')            ELSE (DIN1(1));
            DOUT_TEMP(2) <= (DIN2(2)) WHEN (toggle = '1')            ELSE (DIN1(2));
            DOUT_TEMP(3) <= (DIN2(3)) WHEN (toggle = '1')            ELSE (DIN1(3));
            DOUT_TEMP(4) <= (DIN2(4)) WHEN (toggle = '1')            ELSE (DIN1(4));
            DOUT_TEMP(5) <= (DIN2(5)) WHEN (toggle = '1')            ELSE (DIN1(5));
            DOUT_TEMP(6) <= (DIN2(6)) WHEN (toggle = '1')            ELSE (DIN1(6));
--          DOUT_TEMP(7) <= (DIN2(7)) WHEN (toggle = '1')            ELSE (DIN1(7));

            DOUT(0) <= '0' WHEN (DOUT_TEMP(0) = '0')       ELSE '1';
            DOUT(1) <= '0' WHEN (DOUT_TEMP(1) = '0')       ELSE 'Z'; --open drain
            DOUT(2) <= '0' WHEN (DOUT_TEMP(2) = '0')       ELSE '1';
            DOUT(3) <= '0' WHEN (DOUT_TEMP(3) = '0')       ELSE '1';
            DOUT(4) <= '0' WHEN (DOUT_TEMP(4) = '0')       ELSE '1';
            DOUT(5) <= '0' WHEN (DOUT_TEMP(5) = '0')       ELSE 'Z'; --open drain
            DOUT(6) <= '0' WHEN (DOUT_TEMP(6) = '0')       ELSE 'Z'; --open drain
--          DOUT(7) <= '0' WHEN (DOUT_TEMP(7) = '0')       ELSE '1';




end architecture syn;
```

Simulations

1. AND Logic Gate Simulation

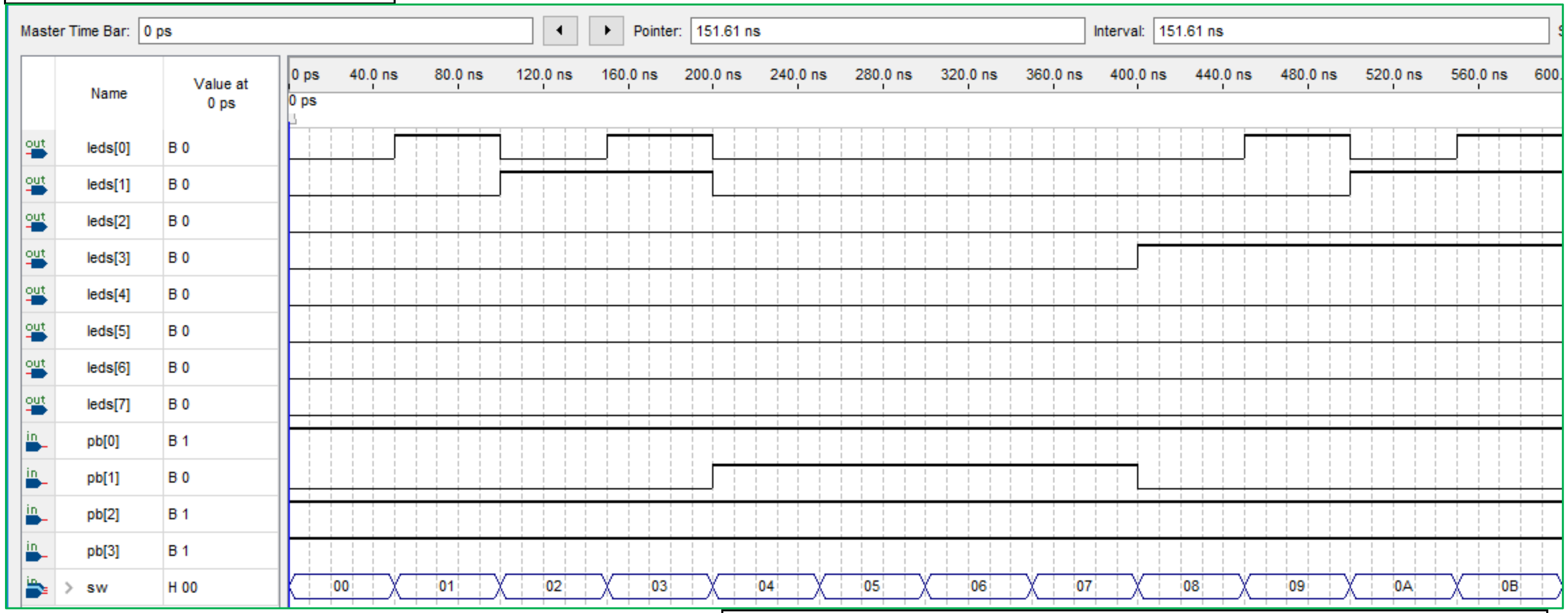

Master Time Bar: 0 ps Pointer: 3.18 ns Interval: 3.18 ns

Name	Value at 0 ps
leds[0]	B 1
leds[1]	B 0
leds[2]	B 0
leds[3]	B 0
leds[4]	B 0
leds[5]	B 0
leds[6]	B 0
leds[7]	B 0
pb[0]	B 0
pb[1]	B 1
pb[2]	B 1
pb[3]	B 1
sw	H 3D

Random Readings have been used to show the functioning of the Gate.

pb[0] is varying while the rest of the push buttons are set to ‘OFF’

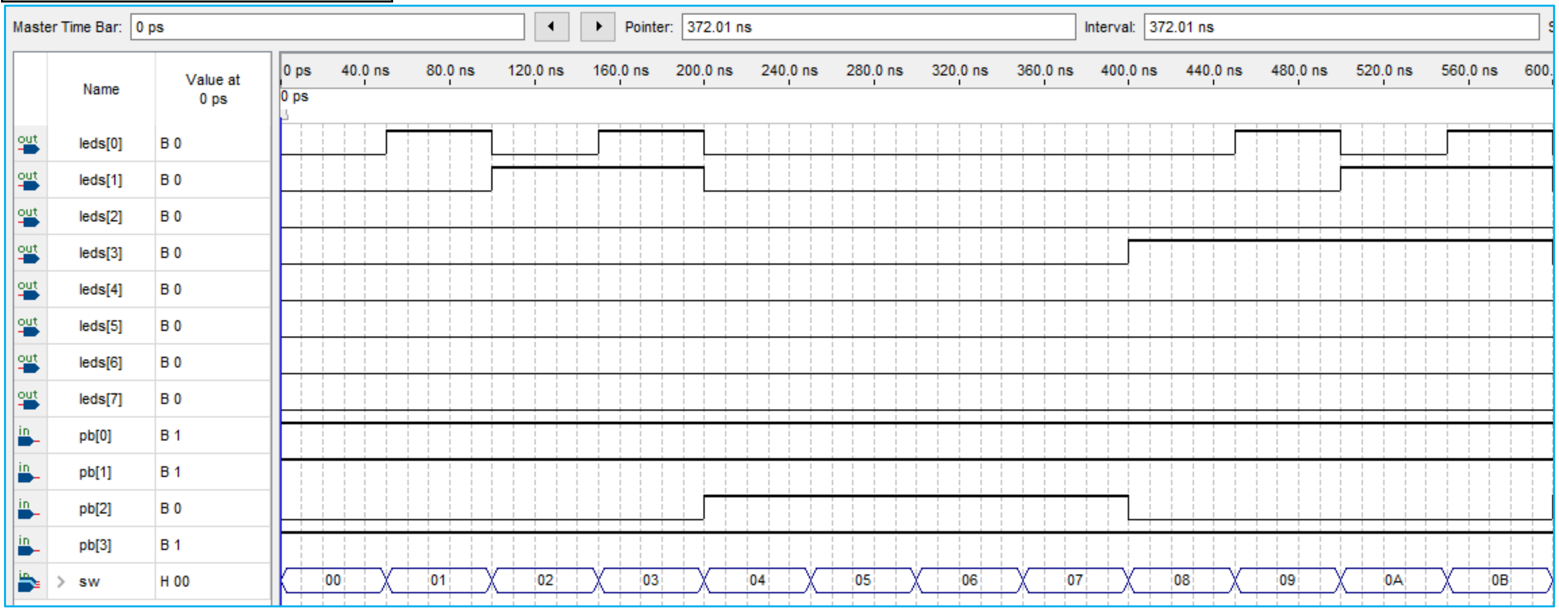
2. OR Logic Gate Simulation


Master Time Bar: 0 ps Pointer: 151.61 ns Interval: 151.61 ns

Name	Value at 0 ps
leds[0]	B 0
leds[1]	B 0
leds[2]	B 0
leds[3]	B 0
leds[4]	B 0
leds[5]	B 0
leds[6]	B 0
leds[7]	B 0
pb[0]	B 1
pb[1]	B 0
pb[2]	B 1
pb[3]	B 1
sw	H 00

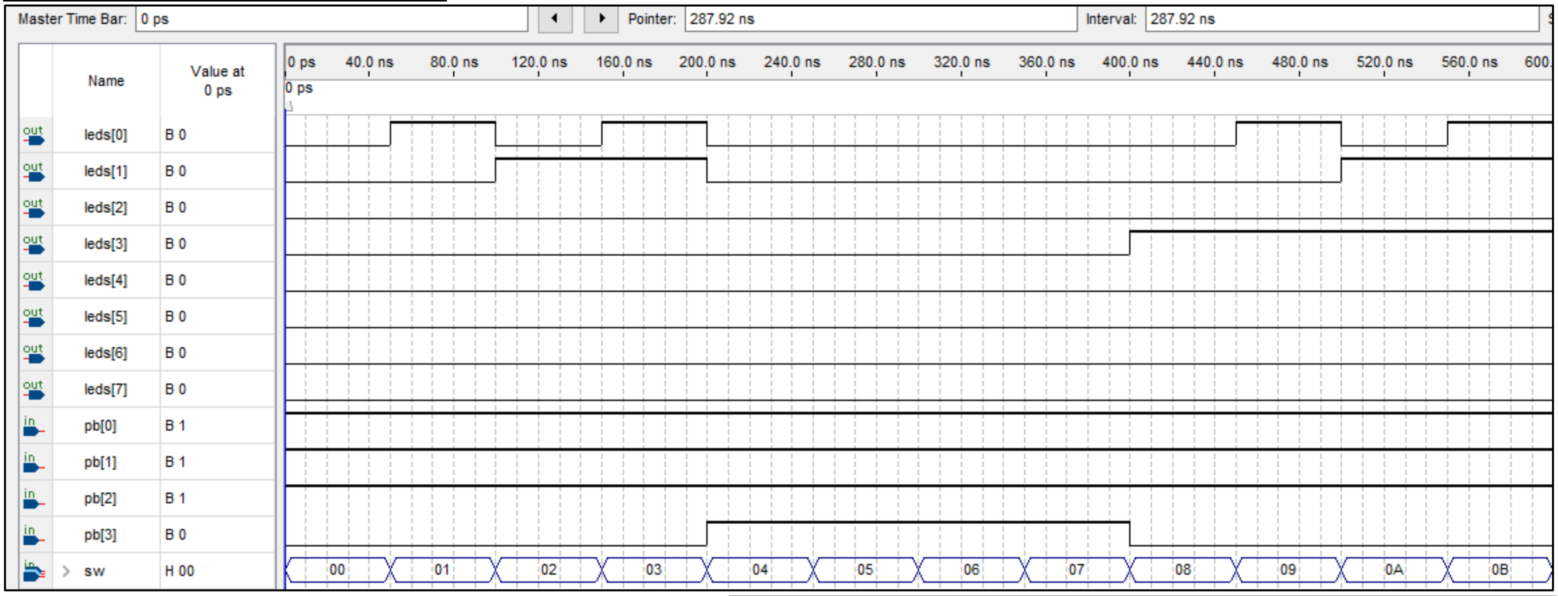
pb[1] is varying while the rest of the push buttons are set to ‘OFF’

3. XOR Logic Gate Simulation



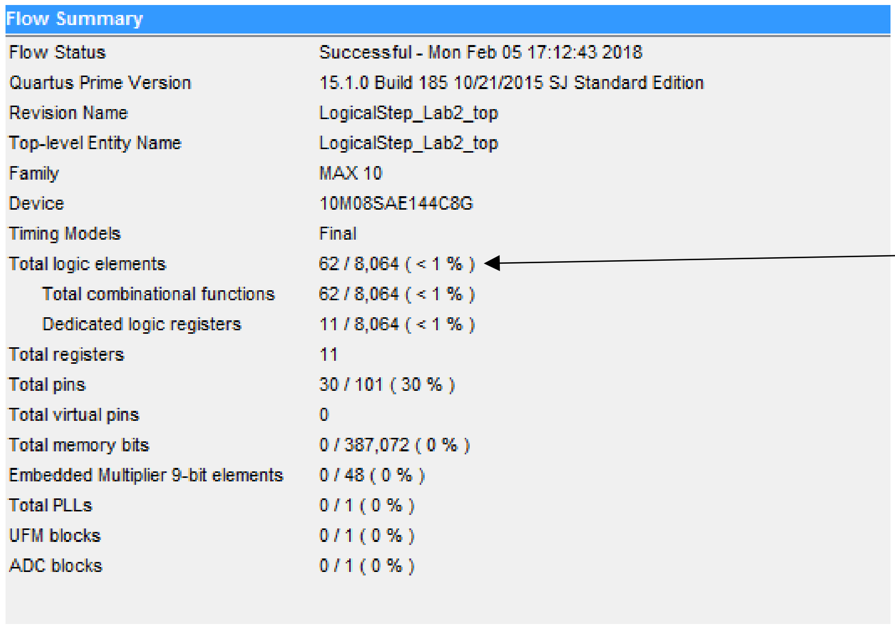
pb[2] is varying while the rest of the push buttons are set to 'OFF'

4. Arithmetic Function Simulation



pb[3] is varying while the rest of the push buttons are set to 'OFF'

Compilation Report

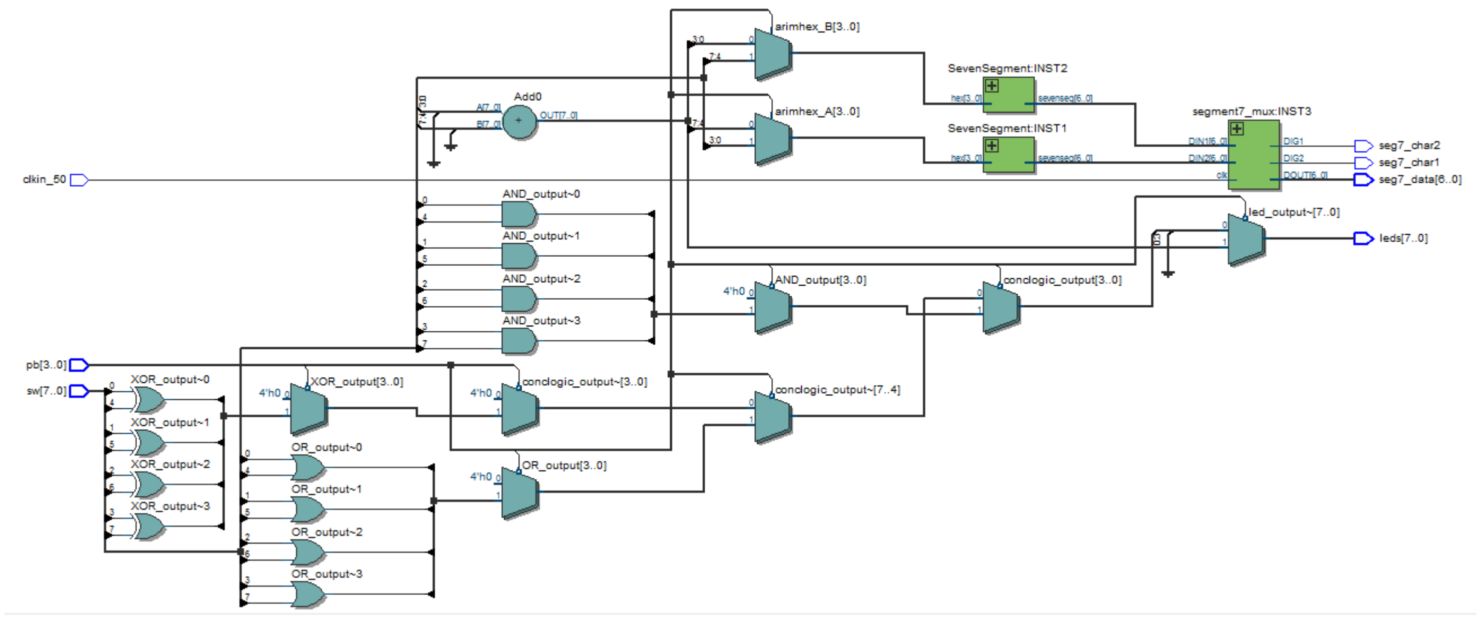


Flow Summary

Flow Status	Successful - Mon Feb 05 17:12:43 2018
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	62 / 8,064 (< 1 %)
Total combinational functions	62 / 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 %)

Total Design Logic Elements Used = 62

RTL Diagram



arimhex_B[3..0]
arimhex_A[3..0]
SevenSegment:INST2
SevenSegment:INST1
segment7_mux:INST3
Add0
clkin_50
pb[3..0]
sw[7..0]
AND_output~0
AND_output~1
AND_output~2
AND_output~3
XOR_output~0
XOR_output~1
XOR_output~2
XOR_output~3
XOR_output[3..0]
OR_output~0
OR_output~1
OR_output~2
OR_output~3
OR_output[3..0]
conclogic_output~[3..0]
conclogic_output~[7..4]
AND_output[3..0]
conclogic_output[3..0]
led_output~[7..0]
seg7_char2
seg7_char1
seg7_data[6..0]
leds[7..0]
4'h0
DIG1
DIG2
DOUT[6..0]
DIN1[6..0]
DIN2[6..0]
clk