```
 1
 2   library ieee;
 3   use ieee.std_logic_1164.all;
 4   use ieee.numeric_std.all;
 5
 6
 7   entity LogicalStep_Lab2_top is port (
 8      clkin_50       : in  std_logic;
 9      pb             : in  std_logic_vector(3 downto 0);
10      sw             : in  std_logic_vector(7 downto 0); -- The switch inputs
11      leds           : out std_logic_vector(7 downto 0); -- for displaying the switch content
12      seg7_data      : out std_logic_vector(6 downto 0); -- 7-bit outputs to a 7-segment
13      seg7_char1     : out std_logic;                    -- seg7 digit1 selector
14      seg7_char2     : out std_logic                     -- seg7 digit2 selector
15
16   );
17   end LogicalStep_Lab2_top;
18
19   architecture SimpleCircuit of LogicalStep_Lab2_top is
20   --
21   -- Components Used ---
22   ------------------------------------------------------------------
23     component SevenSegment port (
24      hex         :  in  std_logic_vector(3 downto 0);   -- The 4 bit data to be displayed
25      sevenseg    :  out std_logic_vector(6 downto 0)    -- 7-bit outputs to a 7-segment
26      );
27      end component;
28
29      component segment7_mux port (
30         clk   : in std_logic := '0';
31         DIN2  : in std_logic_vector(6 downto 0);
32         DIN1  : in std_logic_vector(6 downto 0);
33         DOUT  : out std_logic_vector(6 downto 0);
34         DIG2  : out std_logic;
35         DIG1  : out std_logic
36      );
37      end component;
38
39      component concatenate_sum_mux port (
40      inp1        : in std_logic_vector(7 downto 0);   -- concatenate input
41      inp2        : in std_logic_vector(7 downto 0);   -- sum input
42      mux_select  : in std_logic;                      -- pb(3) as selector
43      output      : out std_logic_vector(7 downto 0)   -- 8 bit output
44      );
45      end component;
46
47      component logic_processor port (
48      inp1        : in std_logic_vector(3 downto 0);   -- hex_A input
49      inp2        : in std_logic_vector(3 downto 0);   -- hex_B input
50      pbs_inp     : in std_logic_vector(3 downto 0);   -- pbs(3..0) as input
51      logic_func  : out std_logic_vector(3 downto 0)   -- output, given by our inputs through
    push buttons
52      );
53      end component;
54
55
56   -- Defined logic vectors
57
58      signal seg7_A     : std_logic_vector(6 downto 0);
59      signal hex_A      : std_logic_vector(3 downto 0);
60      signal seg7_B     : std_logic_vector(6 downto 0);
61      signal hex_B      : std_logic_vector(7 downto 4);
62      signal add_inpA   : std_logic_vector(7 downto 0);
63      signal add_inpB   : std_logic_vector(7 downto 0);
64      signal outp       : std_logic_vector(7 downto 0);
65      signal outp2      : std_logic_vector(7 downto 0);
66      signal log        : std_logic_vector(3 downto 0);
67
68
69      signal arim_hex_A  : std_logic_vector(3 downto 0);
70      signal arim_hex_B  : std_logic_vector(3 downto 0);
71      signal concatenate : std_logic_vector(7 downto 0);
72      signal sum         : std_logic_vector(7 downto 0);
73      signal concatenate2: std_logic_vector(7 downto 0);
74
75   -- Here the circuit begins
```

```
76
77    begin
78
79       hex_A <= sw(3 downto 0); --takes input from switches(3-0)
80       hex_B <= sw(7 downto 4); --takes input from switches(7-4)
81
82       add_inpA <= std_logic_vector("0000" & hex_A); -- converts 4 bit hex_A signal to 8 bit by
    concatenating
83       add_inpB <= std_logic_vector("0000" & hex_B); -- converts 4 bit hex_B signal to 8 bit by
    concatenating
84
85       with pb select                                -- converting the 8 bit output from the
    multiplexer into two 4 bit signals
86       arim_hex_A <=  outp(3 downto 0) when "1110",
87                      outp(3 downto 0) when "1101",
88                      outp(3 downto 0) when "1011",
89                      outp(3 downto 0) when "0111",
90                      outp(3 downto 0) when "1111",
91                      "1000" when others;           -- when more than one pb is pressed, 8 is
    shown on digit 2
92
93       with pb select
94       arim_hex_B <=  outp(7 downto 4) when "1110",
95                      outp(7 downto 4) when "1101",
96                      outp(7 downto 4) when "1011",
97                      outp(7 downto 4) when "0111",
98                      outp(7 downto 4) when "1111",
99                      "1000" when others;           -- when more than one pb is pressed, 8 is
    shown on digit 1
100
101      concatenate <= std_logic_vector(hex_B & hex_A);               -- combines the two 4
    bit inputs to one 8 bit value
102
103      sum <= std_logic_vector(unsigned(add_inpB)+unsigned(add_inpA)); -- adding add_inpA and
    add_inpB and storing the value (4 bit adder)
104
105      concatenate2 <= std_logic_vector("0000" & log);               -- this signal stores
    the 4 bit output as a concatenated 8 bit value coming from the logic processor
106
107      with pb select                    -- LEDs output(which is the output from the multiplexer
    having inputs as sum and output of logic processor) when one or more pbs are pressed
108      leds <= outp2 when "1110",
109              outp2 when "1101",
110              outp2 when "1011",
111              outp2 when "0111",
112              outp2 when "1111",
113              "11111111" when others;  -- when more than one push button is pressed, all the
    LEDs will light up (error case)
114
115
116
117   -- COMPONENT HOOKUP
118
119   -- Instantiation of multiplexers, logic_processor and SevenSegment
120
121      INST1: SevenSegment port map(arim_hex_A, seg7_A);                        -- sending the 4
    bit output of the mulitplexer to display on the digits
122      INST2: SevenSegment port map(arim_hex_B, seg7_B);
123      INST3: segment7_mux port map(clkin_50, seg7_A, seg7_B, seg7_data, seg7_char2, seg7_char1);
124
125      INST4: concatenate_sum_mux port map(concatenate, sum, pb(3), outp);     -- depending on
    the input of pb(3) the output of the multiplexer will be concatenate or sum(of the hex
    signals)
126      INST5: concatenate_sum_mux port map(concatenate2, sum, pb(3), outp2);   -- depending on
    the input of pb(3) the output of the multiplexer will be 8 bit output of the logic
    processor("0000"&log) or sum
127      INST6: logic_processor port map(hex_A, hex_B, pb, log);                  -- depending on
    the input of pb(3-0) the output of the logic processor will be AND, OR or XOR of hex_A and
    hex_B
128
129   end SimpleCircuit;
130
131
```