```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    08:40:49 10/07/2019
6  -- Design Name:
7  -- Module Name:    Lab5 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21 -- Uncomment the following library declaration if using
22 -- arithmetic functions with Signed or Unsigned values
23 --use IEEE.NUMERIC_STD.ALL;
24
25 -- Uncomment the following library declaration if instantiating
26 -- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 library IEEE;
31 use IEEE.STD_LOGIC_1164.ALL;
32 USE ieee.std_logic_arith.all;
33 USE ieee.std_logic_unsigned.all;
34
35 -- Component used for selection of the 7 segment display .
36 --Selects one of the 4 7-segment display based on the 2 bit sel input
37 entity entaff is
38 Port ( Sel : in STD_LOGIC_VECTOR (1 downto 0);  -- 2 bit select input
39        AffOP : out STD_LOGIC_VECTOR (3 downto 0));  --4 bit output to select the digit
40 end entaff;
41
42 architecture aff of entaff is
43 signal Output : std_logic_vector(3 downto 0) := "0000";  --Store the result in a
   temporary signal as it is not recommended to directly write to the output line
44 begin
45 AffOP <= Output;  --Update the output to the output line
46 with Sel select
47 Output<=     "1110" when "00",    --Select the digit based on the 2-bit input.
48               "1101" when "01",
49                   "1011" when "10",
50                   "0111" when "11",
51                   "1111" when others;  --Dont select any display for invalid input (Does
                     not happen ideally)
52 end aff;
53
54
55 library IEEE;
56 use IEEE.STD_LOGIC_1164.ALL;
57 USE ieee.std_logic_arith.all;
58 USE ieee.std_logic_unsigned.all;
59 --Component used for generating the 7 bit value which will display the required digit
   on the 7 segment display.
60 -- Value 0 at a bit position means that the segment is turned on, value 1 at a bit
   position means that the segment is turned off.
61 entity sevenseg is
62 Port (DigitSelected: in STD_LOGIC_VECTOR (3 downto 0);  --Input which mentions the
   digit that has to be displayed. Digits from 0-9 are valid inputs
63       DigitOutput: out STD_LOGIC_VECTOR (6 downto 0));  --7 bit value which turns on
         the proper segments to display the digit.
```

```
64  end sevenseg;
65
66  architecture sevensegArch of sevenseg is
67  signal Output : std_logic_vector(6 downto 0) := "0000000";   --Store the result in a
    temporary signal
68  begin
69  DigitOutput <=Output;
70  with DigitSelected select
71  Output <=        "0000001" when "0000",     --Based on the input digit, select the 7
    bit output.
72                   "1001111" when "0001",
73                        "0010010" when "0010",
74                        "0000110" when "0011",
75                        "1001100" when "0100",
76                        "0100100" when "0101",
77                        "0100000" when "0110",
78                        "0001111" when "0111",
79                        "0000000" when "1000",
80                        "0000100" when "1001",
81                        "1111111" when others;  --For invalid digit, out of the range
                          0-9, turn off all segments
82  end sevensegArch;
83
84  library IEEE;
85  use IEEE.STD_LOGIC_1164.ALL;
86  USE ieee.std_logic_arith.all;
87  USE ieee.std_logic_unsigned.all;
88
89
90  -- This component handles clock division and also generates the select signal for the
    Aff component.
91  --Takes the on board clock as the input and gives the Aff signal as the output. Aff
    signal is updated once every 100 clock cycles, hence the frequency is divided.
92  entity Mod4 is
93  Port (Clk: in STD_LOGIC;
94        Aff: out STD_LOGIC_VECTOR (1 downto 0));
95  end Mod4;
96
97  architecture mod4Arch of Mod4 is
98  signal FreqDiv : integer;           --Signal to keep count of pulses
99  signal counter : STD_LOGIC_VECTOR (1 downto 0);   --Signal which stores the final Aff
    value
100 begin
101 Aff<=counter;  --Update the final output
102 process (clk)  --Clock is the input for this function. All activities are dependent on
    clock transition (Rising edge).
103 begin
104 if(rising_edge(clk)) then   --Check for rising edge of the clock
105 if(FreqDiv = 99) then    --If 100 pulses are reached, update the counter value and
    reset the FreqDiv variable to 0 to begin counting from 0 again.
106 counter<=counter+1;
107 FreqDiv<=0;
108 else
109 FreqDiv<=FreqDiv+1;    --If 100 pulses are not reached, keep incrementing the FreqDiv
    variable
110 end if;
111 end if;
112 end process;
113 end mod4Arch;
114
115 library IEEE;
116 use IEEE.STD_LOGIC_1164.ALL;
117 USE ieee.std_logic_arith.all;
118 USE ieee.std_logic_unsigned.all;
119
120
121 -- This component controls the push buttons.
122 -- It takes the push button as the input and increments the number if the button was
    pushed.
```

```
-- The output is the final number after the increment. Initially the number is 0
--The number increments upto 9 after which it rolls over to 0 and begins counting again.
entity Button is
Port (Button: in STD_LOGIC;  --Push button input
      Number: out STD_LOGIC_VECTOR (3 downto 0));   --Output Number
end Button;

architecture ButtonArch of Button is
Signal output : STD_LOGIC_VECTOR (3 downto 0);  --Intermediate signal to store the output
begin
Number<= output;  --Update the result
process(Button)     --This function depends on change in the button level
begin
if(rising_edge(Button)) then  --If change in button level observed, increment the number
if output = "1001" then  --Check if the number is already at the max value of 9 before
incrementing. If it is 9, roll over back to 0. Else increment by 1
output <= "0000";
else
output <= output+1;
end if;
end if;
end process;
end ButtonArch;


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;

--This component is used to convert the 7 bit binary digit into 2 4-bit BCD outputs.
--This is needed as we need the digit in BCD format to display it on the output.
--Double dabble architecture is used
entity BIN2BCD is
    Port ( BCDIN: in  STD_LOGIC_VECTOR (6 downto 0);  --7 -bit binary input
           OutputDigit1,OutputDigit2 : out  STD_LOGIC_VECTOR (3 downto 0));  -- 2 4-bit
           BCD outputs

end BIN2BCD;

architecture BIN2BCDArch of BIN2BCD is

begin

process(BCDIN)
variable temp : std_logic_vector(6 downto 0);   --Intermediate variables to hold the
value
variable BCDVar : std_logic_vector(7 downto 0);
begin
temp(6 downto 0):= BCDIN;  --Copy the input onto the temp variable
BCDVar(7 downto 0):= "00000000";  --Initialize the output variable to 0
for i in 0 to 6 loop  --Loop until all the 7 bits of input is processed
if(BCDVar (3 downto 0) > 4) then  --If a nibble is greater than 4, add 3 to the nibble

BCDVar (3 downto 0) := BCDVar (3 downto 0) + 3;
end if;

if(BCDVar (7 downto 4) > 4) then   --If a nibble is greater than 4, add 3 to the nibble
BCDVar (7 downto 4) := BCDVar (7 downto 4) + 3;
end if;
BCDVar(7 downto 0) := BCDVar(6 downto 0) & temp(6);  --Shift the MSB of the input onto
the LSB of the BCD
temp(6 downto 0) := temp (5 downto 0) & '0';  --Shift the input to the left by 1
end loop;
OutputDigit1 <= BCDVar(3 downto 0);  --Copy the converted BCD value of unit's digit to
the output digit 1
OutputDigit2 <= BCDVar(7 downto 4);  --Copy the converted BCD value of ten's digit to
the output digit 1
end process;
```

```
186  end BIN2BCDArch;
187
188
189  library IEEE;
190  use IEEE.STD_LOGIC_1164.ALL;
191  USE ieee.std_logic_arith.all;
192  USE ieee.std_logic_unsigned.all;
193  use IEEE.numeric_std.all;
194
195  -- This component is used to perform the arithmetic operation on the input digits and
     update the result to the output
196  -- The operation performed depends on the switch position.
197  --Before updating the output, the result has to be converted from BIN to BCD which is
     done by the BIN2BCD component
198  entity Calc is
199      Port ( InputDigit1, InputDigit2 : in  STD_LOGIC_VECTOR (3 downto 0);
200             OutputDigit1,OutputDigit2 : out  STD_LOGIC_VECTOR (3 downto 0);
201                switch1, switch2 : in STD_LOGIC);
202  end Calc;
203
204  architecture CalcArch of Calc is
205  component BIN2BCD    --Define the BIN2BCD component
206  Port ( BCDIN: in  STD_LOGIC_VECTOR (6 downto 0);
207             OutputDigit1,OutputDigit2 : out  STD_LOGIC_VECTOR (3 downto 0));
208
209  end component;
210
211  signal resultstd : STD_LOGIC_VECTOR (6 downto 0);  --Intermediate variable to hold the
     result
212  signal IP1, IP2 : STD_LOGIC_VECTOR (6 downto 0);   --Variables which hold the 4 bit
     input digits in 7 bit format. We choose 7 bits as it can hold the maximum possible result
213  --I.e. 2^7 = 128 and the max possible result is 81 (9*9)
214  begin
215  IP1<= "000" & InputDigit1;  --Convert the 4 bit input to a 7 bit input
216  IP2<= "000" & InputDigit2;
217  process(InputDigit1,InputDigit2,switch1,switch2) --Update the result if the input or
     the switches change
218  begin
219  if(switch1 = '0' and switch2 = '0') then   --Perform addition when both switches are in
     the off position
220  resultstd <= IP1+ IP2;
221  end if;
222  if(switch1 = '1' and switch2 = '0') then  --Perform subtraction when switch 1 is on and
     switch 2 is off
223  if (IP1 < IP2) then  --Make sure to subtract the lower number from the higher number
224  resultstd <= IP2-IP1;
225  else
226  resultstd <= IP1-IP2;
227  end if;
228  end if;
229  if(switch2 = '1') then  --If switch 2 is on then perform multiplication
230  resultstd <= IP2*IP1;
231  end if;
232  end process;
233  Convert: BIN2BCD
234  port map (resultstd,OutputDigit1,OutputDigit2);  --Convert the result to BCD and update
     the output
235  end CalcArch;
236
237
238  library IEEE;
239  use IEEE.STD_LOGIC_1164.ALL;
240  USE ieee.std_logic_arith.all;
241  USE ieee.std_logic_unsigned.all;
242
243
244  --This is the main entity and architecture for the implementation of the Calculator
245
246  entity Lab5 is
```

```
247       Port ( clk : in  STD_LOGIC;   --The clock drives the display
248              Digit : out  STD_LOGIC_VECTOR (3 downto 0);  --Output to select the display
249              Segment : out  STD_LOGIC_VECTOR (6 downto 0);  --Output to display the digit.
250                 pb1, pb2, SW1, SW0 : in STD_LOGIC);  --Input push buttons and switches
251   end Lab5;
252
253   architecture Behavioral of Lab5 is   --Declare all the components which are needed
254   component entaff
255   port (Sel : in STD_LOGIC_VECTOR (1 downto 0);
256         AffOP : out STD_LOGIC_VECTOR (3 downto 0));
257   end component;
258
259   component sevenseg
260   Port (DigitSelected: in STD_LOGIC_VECTOR (3 downto 0);
261         DigitOutput: out STD_LOGIC_VECTOR (6 downto 0));
262   end component;
263
264   component mod4
265   Port (Clk: in STD_LOGIC;
266         Aff: out STD_LOGIC_VECTOR (1 downto 0));
267   end component;
268
269   component Button
270   Port (Button: in STD_LOGIC;
271         Number: out STD_LOGIC_VECTOR (3 downto 0));
272   end component;
273
274   component calc
275
276   Port ( InputDigit1, InputDigit2 : in  STD_LOGIC_VECTOR (3 downto 0);
277              OutputDigit1,OutputDigit2 : out  STD_LOGIC_VECTOR (3 downto 0);
278                 switch1, switch2 : in STD_LOGIC);
279   end component;
280
281   signal SegOut: std_logic_vector(6 downto 0) ;  --Signal to hold the 7 bit value for the
      display
282   signal DigitOut: std_logic_vector(3 downto 0) ;  --Signal to select the digit
283   signal DigitSel, Dig1, Dig2, Dig3, Dig4 : std_logic_vector(3 downto 0) ;  --Signal that
      holds the value of the digit
284   signal Aff_Sig : std_logic_vector (1 downto 0) ;  --Signal for the select line
285
286   begin
287   Digit<=DigitOut;  --Assign the outputs
288   Segment<=SegOut;
289   AffFunc : mod4  --Generate the AffSignal from Entaff function
290   port map(Clk,Aff_Sig);
291   SelectFunc : entaff   --Select the digit on which the number is to be displayed
292   port map(Aff_Sig,DigitOut);
293   DispFunc : sevenseg    --Get the 7 bit output for the Seven Segment display
294   port map(DigitSel,SegOut);
295   BP1: Button           -- Evaluate the input from button 1, increment digit if button
      was pressed
296   port map(pb1,Dig1);
297
298   BP2: Button            --Evaluate input from button 2, increment digit if button was
      pressed
299   port map(pb2,Dig2);
300
301   Calc1 : Calc    --Invoke the calculator function
302   port map(Dig1, Dig2,Dig3, Dig4, sw0, sw1);
303
304
305
306   process(Aff_Sig)
307   begin
308   case Aff_Sig is
309   when "00" => DigitSel <= Dig1;    --Update the right output for each digit. Ex: When
      the select line is "00", we display the first digit (Dig1)
310   when "01" => DigitSel <= Dig2;
```

```
311     when "10" => DigitSel <= Dig3;
312     when others => DigitSel <= Dig4;
313     end case;
314     end process;
315 
316     end Behavioral;
317 
318 
```