Lab Report No. 03:

Logic Circuits

By:

Jiaqing Wan, Jiaxin Zheng

Instructor: Dr. Xi Zhou

Classnumber: ZM1501

NAU ID:

5358836, 5358840

October 6, 2017

I. INTRODUCTION

A. Objectives

At the completion of this lab, we will be able to:
Complete a VHDL design from a functional specification (write VHDL, simulate functional behavior using ModelSim, and demonstrate functional behavior on the Cyclone V demo board.

B. Background knowledge

we need know some VHDL laguage's syntax,such as vector.

C. Design strategy

a.Activity 1: VHDL Design
b.Activity 2: Using the Alphanumeric LED Displays.

D. Simulation strategy

TABLE I monitor set

input			output
push_in[1]	push_in[0]	switch_in	status
key1	key0	sw0	ledr0

TABLE II seg7 set

input			output	
blank test	upper4bit[7..4]	lower4bit[3..0]	sm_ db_1	sm_db_0
sw9 sw8	sw[7..4]	sw[3..0]	HEX1[6..0]	HEX0[6..0]
			HEX3[6..0]	HEX2[6..0]
			HEX5[6..0]	HEX4[6..0]

E. Demonstration strategy

Activity 1 monitor demonstration strategy:

- if only one of two pushbuttons is pressed,the ledr0 will lighting.
- other case the ledr0 will not lights.

Activity 2 seg7 demonstration strategy:

- LED display should have an input called blank. All LEDs are off when blank=1.
- LED display should have another input called test. All LEDs are on when test=1.
- If blank and test are 1 at the same time, no LED is on.
- outputs the correct signals for display of an input 8-bit binary standard logic vector in hex on two alphanumeric 7-segment LED digits when blank=0 and test=0.

II. LAB DETAILS

A. Instructor questions in the lab instruction

Activity 1 question

1. Generate the truth table for this circuit, remembering that the pushbuttons are logic 1 if not pressed.

TABLE III the logical of ledr0

Put_in[1]	Put_in[0]	Switch0	Ledr0
×	×	0	0
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	0

Activity 2 question

2. Fill out the truth table for a hexadecimal 7-seg display module.

TABLE IV the logical of seg7-BCD dispaly

Blank	Test	Num	In3	In2	In1	In0	Hex0[6]	HEX[5]	HEX[4]	HEX[3]	HEX[2]	HEX[1]	HEX[0]
1	×	×	×	×	×	×	1	1	1	1	1	1	1
0	1	×	×	×	×	×	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	1	1	1	1	1	0	0	1
0	0	2	0	0	1	0	0	1	0	0	1	0	0
0	0	3	0	0	1	1	0	1	1	0	0	0	0
0	0	4	0	1	0	0	0	0	1	1	0	0	1
0	0	5	0	1	0	1	0	0	1	0	0	1	0
0	0	6	0	1	1	0	0	0	0	0	0	1	0
0	0	7	0	1	1	1	1	1	1	1	0	0	0
0	0	8	1	0	0	0	0	0	0	0	0	0	0
0	0	9	1	0	0	1	0	0	1	0	0	0	0
0	0	A	1	0	1	0	0	0	0	1	0	0	0
0	0	b	1	0	1	1	0	0	0	0	0	1	1
0	0	C	1	1	0	0	1	0	0	0	1	1	0
0	0	d	1	1	0	1	0	1	0	0	0	0	1
0	0	E	1	1	1	0	0	0	0	0	1	1	0
0	0	F	1	1	1	1	0	0	0	1	1	1	0

3. How many possible input combinations are there for this circuit? Is that practical?

A: there are $2^{10} = 1024$ (the inputs are 10 switch) possible input combination for the circuit.It's not practical, because it's too much situation.

4. How many logic elements are used?

A: 14.

B. VHDL code

monitor.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity monitor is
port (
push_in : in std_logic_vector(1 downto 0);
switch_in : in std_logic;
status : out std_logic
);
end monitor;
architecture behav of monitor is
begin
process(switch_in, push_in)
begin
-- use the if elsif statement!
if switch_in = '1' AND (Push_in = B"01" OR Push_in = B"10") then
     status <= '1';
    ELSE
       status <= '0';
   END IF;
end process;
end behav;
```

seg7.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity seg7 is
   port(
      test: in STD_LOGIC;
      blank: in STD_LOGIC;
      upper4bit: in STD_LOGIC_VECTOR (7 downto 4);
      lower4bit: in STD_LOGIC_VECTOR (3 downto 0);
      sm_db_0: out STD_LOGIC_VECTOR (6 downto 0);
      sm_db_1: out STD_LOGIC_VECTOR (6 downto 0)
   );
end entity seg7;

architecture seg_display of seg7 is
   constant seg0: STD_LOGIC_VECTOR (6 downto 0) := "1000000";
   constant seg1: STD_LOGIC_VECTOR (6 downto 0) := "1111001";
   constant seg2: STD_LOGIC_VECTOR (6 downto 0) := "0100100";
   constant seg3: STD_LOGIC_VECTOR (6 downto 0) := "0110000";
   constant seg4: STD_LOGIC_VECTOR (6 downto 0) := "0011001";
   constant seg5: STD_LOGIC_VECTOR (6 downto 0) := "0010010";
   constant seg6: STD_LOGIC_VECTOR (6 downto 0) := "0000010";
   constant seg7: STD_LOGIC_VECTOR (6 downto 0) := "1111000";
   constant seg8: STD_LOGIC_VECTOR (6 downto 0) := "0000000";
   constant seg9: STD_LOGIC_VECTOR (6 downto 0) := "0010000";
   constant sega: STD_LOGIC_VECTOR (6 downto 0) := "0001000";
   constant segb: STD_LOGIC_VECTOR (6 downto 0) := "0000011";
   constant segc: STD_LOGIC_VECTOR (6 downto 0) := "1000110";
```

```
   constant segd: STD_LOGIC_VECTOR (6 downto 0) := "0100001";
   constant sege: STD_LOGIC_VECTOR (6 downto 0) := "0000110";
   constant segf: STD_LOGIC_VECTOR (6 downto 0) := "0001110";
begin
-----------------------
   process(test,blank,upper4bit,lower4bit)
      begin
         if (blank = '0' AND test = '1') then
            sm_db_0 <= "0000000";
            sm_db_1 <= "0000000";
         elsif (blank = '1') then
            sm_db_0 <= "1111111";
            sm_db_1 <= "1111111";
         else
            case lower4bit is
               when x"0" => sm_db_0 <= seg0;--1000000
               when x"1" => sm_db_0 <= seg1;--1111001
               when x"2" => sm_db_0 <= seg2;--0100100
               when x"3" => sm_db_0 <= seg3;--0110000
               when x"4" => sm_db_0 <= seg4;--0011001
               when x"5" => sm_db_0 <= seg5;--0010010
               when x"6" => sm_db_0 <= seg6;--0000010
               when x"7" => sm_db_0 <= seg7;--1111000
               when x"8" => sm_db_0 <= seg8;--0000000
               when x"9" => sm_db_0 <= seg9;--0010000
               when x"a" => sm_db_0 <= sega;--0001000
               when x"b" => sm_db_0 <= segb;--0000011
               when x"c" => sm_db_0 <= segc;--1000110
               when x"d" => sm_db_0 <= segd;--0100001
               when x"e" => sm_db_0 <= sege;--0000110
               when x"f" => sm_db_0 <= segf;--0001110
               when others => null;
            end case;
            case upper4bit is
         when x"0" => sm_db_1 <= seg0;--1000000
               when x"1" => sm_db_1 <= seg1;--1111001
               when x"2" => sm_db_1 <= seg2;--0100100
               when x"3" => sm_db_1 <= seg3;--0110000
               when x"4" => sm_db_1 <= seg4;--0011001
               when x"5" => sm_db_1 <= seg5;--0010010
               when x"6" => sm_db_1 <= seg6;--0000010
               when x"7" => sm_db_1 <= seg7;--1111000
               when x"8" => sm_db_1 <= seg8;--0000000
               when x"9" => sm_db_1 <= seg9;--0010000
               when x"a" => sm_db_1 <= sega;--0001000
               when x"b" => sm_db_1 <= segb;--0000011
               when x"c" => sm_db_1 <= segc;--1000110
               when x"d" => sm_db_1 <= segd;--0100001
               when x"e" => sm_db_1 <= sege;--0000110
               when x"f" => sm_db_1 <= segf;--0001110
               when others => null;

            end case;
         end if;
   end process;
-----------------------
end architecture seg_display;
```

C. Block diagram

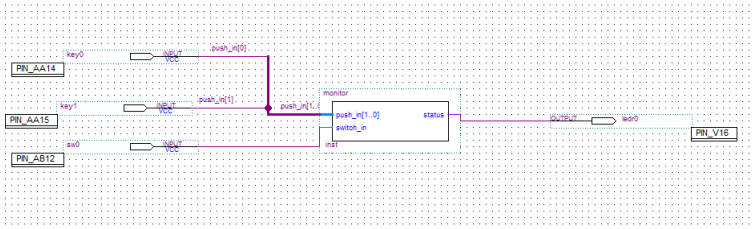

FIG. II.1 monitor_tb Block diagram

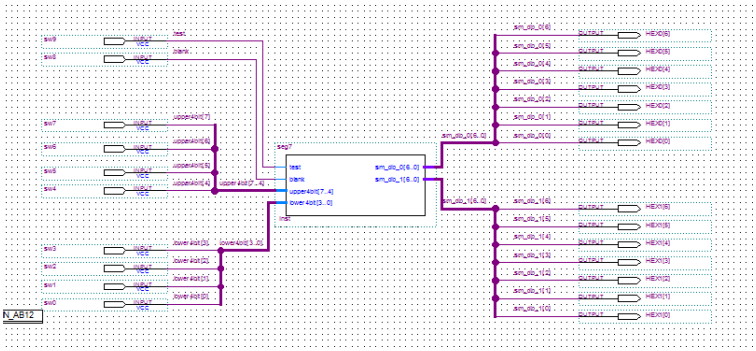

FIG. II.2 seg7_tb Block diagram sheet 1

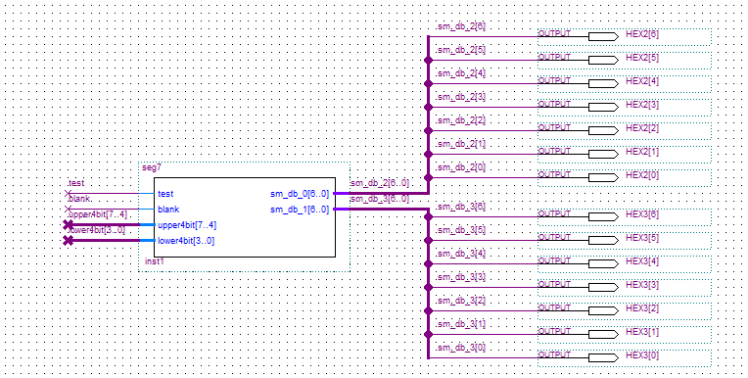

FIG. II.3 seg7_tb Block diagram sheet 2

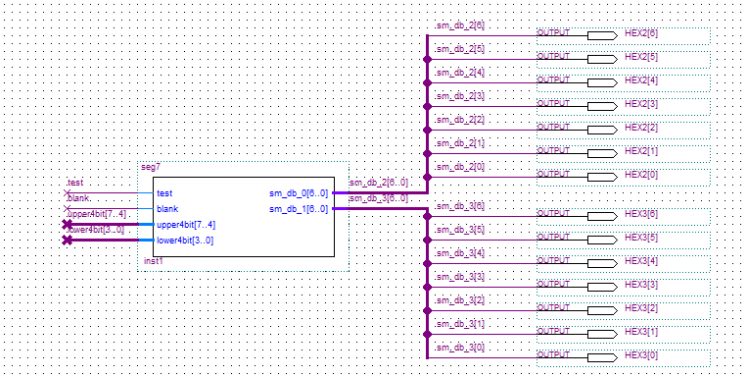

FIG. II.4 seg7_tb Block diagram sheet 3

D. ModelSim force file

monitor_tb.txt:

```
force switch_in  0
force push_in(0) 0
force push_in(1) 0
run 100ns


force switch_in  0
force push_in(0) 1
force push_in(1) 0
run 100ns

force switch_in  0
force push_in(0) 0
force push_in(1) 1
run 100ns

force switch_in  0
force push_in(0) 1
force push_in(1) 1
run 100ns

force switch_in  1
force push_in(0) 1
force push_in(1) 1
run 100ns

force switch_in  1
force push_in(0) 1
force push_in(1) 0
run 100ns
```

seg7_tb.txt:

```
force test  1
force blank 0
run 200ns;#all the light will be on,0000000,000000

force blank 1
run 200ns;#all the light will be off,1111111,1111111

force test  0
run 200ns;#all the light will be off,1111111,1111111
force test  0
force blank 0;# active switch control seg7-display function

force upper4bit 0000
force lower4bit 0000
run 100ns
force lower4bit 0001
run 100ns
force lower4bit 0010
run 100ns
force lower4bit 0011
run 100ns
force lower4bit 0100
run 100ns
force lower4bit 0101
run 100ns
force lower4bit 0110
run 100ns
force lower4bit 0111
run 100ns
force lower4bit 1000
run 100ns
force lower4bit 1001
run 100ns
force lower4bit 1010
run 100ns
force lower4bit 1011
run 100ns
force lower4bit 1100
run 100ns
force lower4bit 1101
run 100ns
force lower4bit 1110
run 100ns
force lower4bit 1111
run 100ns


force upper4bit 0000
run 100ns
force upper4bit 0001
run 100ns
force upper4bit 0011
run 100ns
force upper4bit 0100
run 100ns
force upper4bit 0101
run 100ns
force upper4bit 0110
run 100ns
```

```
force upper4bit 0111
run 100ns
force upper4bit 1000
run 100ns
force upper4bit 1001
run 100ns
force upper4bit 1011
run 100ns
force upper4bit 1100
run 100ns
force upper4bit 1101
run 100ns
force upper4bit 1110
run 100ns
force upper4bit 1111
run 100ns
```

E. imulation results and detailed explanation

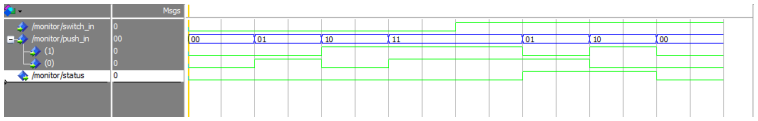

FIG. II.5 monitor_tb Wave

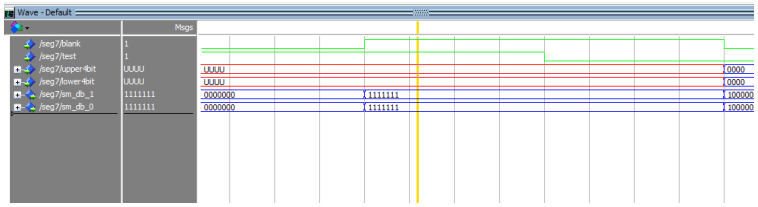

FIG. II.6 seg7_tb Wave sheet1

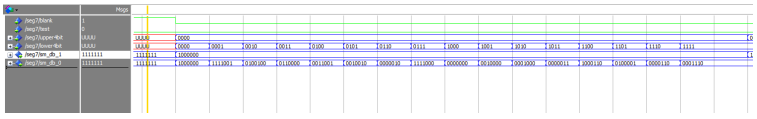

FIG. II.7 seg7_tb Wave sheet2

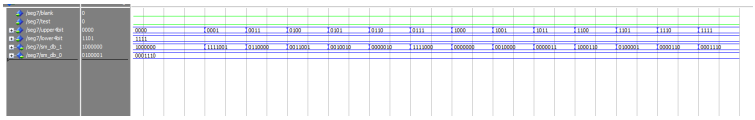

FIG. II.8 seg7_tb Wave sheet3

F. Board demonstration result

monitor_tb.sof:

The logical is the same as Table in the Board demonstration.

seg7_tb.sof:

The logical is the same as Table in the Board demonstration.

III. CONCLUSION

The lab is successful.

I realize all function we should achieve. The most difficulty things is the Instructor questions.

I refer to manual and ask classmates that I solved it.