

**Switch LEDs and HEX displays with FPGA**

Embedded Systems

Laboratory work 2

Done by: EKSfmu-16 gr. st. Arūnas Butkus

Checked by: doc. dr. Andrius Ušinskas

**Work goal:** 1. Switch LEDs with switches. 2. Switch HEX display with switches.

Algorithms used in tasks.

Its FPGA: if something changes something else changes, no sequence to it, especially in these tasks. In case 1: swtich state changes – led state changes. Case 2: switch state changes – HEX display shows a different thing.

Results analysis.

More blinking LEDs. And also yet another in the series: here’s a task – figure out how to use all this software and hardware. With an added bonus of VHDL having significantly different syntax and logic when compared to any other programming language.

Another serious gripe I have with this LD – stupid Quartus. First, it installs in random places and projects in special locations - messing up my folder structure. Second, messed up my Windows services – enabled windows update – which then messed up my PC in general. As a bonus had to write .bat file that keeps disabling Windows update service repeatedly, because now for some reason it re-enables once in a blue moon and starts to make a mess again. And that’s not mentioning it doesn’t even work properly on Windows 10. Thirdly, all the extra crap I installed and search required to make it work as intended. All in all can’t wait to clean it out of my PC as thoroughly as possible.

So after all the troubleshooting and learning how to program with VHDL in the first place the code is simple and to the point. In first case you just push the state of switch to LED no need for FPGA intervention to begin with… In second case switch states are compared to a table in memory which then decides what should light up on HEX display (Fig. 1.).



**Fig. 1.** Example of HEX displays showing things [source: https://youtu.be/Wulkias8waE]

Conclusions.

Wading through all the troubleshooting, all the crap now littering my PC wasn’t worth it, but the thing works as asked. Only good that came of it is first intermediate exam signal generator – making that in the end was fun.

Source code.

Code for some project part of labworks I did that involves HEX dispalys.

01 library ieee;

02 use ieee.std\_logic\_1164.all;

03

04 entity counterTest is

05 Port (

06 CLOCK\_50 : IN STD\_LOGIC;

07 SW : IN STD\_LOGIC\_VECTOR(9 DOWNTO 0);

08 KEY : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

09 HEX0 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

10 HEX1 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

11 HEX2 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

12 HEX3 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

13 LEDG : OUT STD\_LOGIC\_VECTOR(9 downto 0)

14 );

15 end counterTest;

16

17 architecture Whatever of counterTest is

18

19 constant timer\_max : INTEGER := 50000000;

20

21 signal timer\_count : INTEGER := 0;

22 signal timer\_owerflow: STD\_LOGIC;

23 signal led\_state : STD\_LOGIC := '0';

24 signal counter : INTEGER := 0;

25

26 begin

27

28 WITH counter mod 10 SELECT HEX0 <=

29 "11000000" WHEN 0,

30 "11111001" WHEN 1,

31 "10100100" WHEN 2,

32 "10110000" WHEN 3,

33 "10011001" WHEN 4,

34 "10010010" WHEN 5,

35 "10000010" WHEN 6,

36 "11111000" WHEN 7,

37 "10000000" WHEN 8,

38 "10010000" WHEN 9,

39 "01111111" WHEN OTHERS;

40

41 WITH (counter / 10) mod 10 SELECT HEX1 <=

42 "11000000" WHEN 0,

43 "11111001" WHEN 1,

44 "10100100" WHEN 2,

45 "10110000" WHEN 3,

46 "10011001" WHEN 4,

47 "10010010" WHEN 5,

48 "10000010" WHEN 6,

49 "11111000" WHEN 7,

50 "10000000" WHEN 8,

51 "10010000" WHEN 9,

52 "01111111" WHEN OTHERS;

53

54 WITH (counter / 100) mod 10 SELECT HEX2 <=

55 "11000000" WHEN 0,

56 "11111001" WHEN 1,

57 "10100100" WHEN 2,

58 "10110000" WHEN 3,

59 "10011001" WHEN 4,

60 "10010010" WHEN 5,

61 "10000010" WHEN 6,

62 "11111000" WHEN 7,

63 "10000000" WHEN 8,

64 "10010000" WHEN 9,

65 "01111111" WHEN OTHERS;

66

67 WITH (counter / 1000) mod 10 SELECT HEX3 <=

68 "11000000" WHEN 0,

69 "11111001" WHEN 1,

70 "10100100" WHEN 2,

71 "10110000" WHEN 3,

72 "10011001" WHEN 4,

73 "10010010" WHEN 5,

74 "10000010" WHEN 6,

75 "11111000" WHEN 7,

76 "10000000" WHEN 8,

77 "10010000" WHEN 9,

78 "01111111" WHEN OTHERS;

79

80 process(CLOCK\_50)

81 begin

82 if rising\_edge(CLOCK\_50) then

83 timer\_count <= timer\_count + 1;

84 if (timer\_count = TIMER\_MAX) then

85 timer\_owerflow <= '1';

86 led\_state <= not led\_state;

87 counter <= counter + 1;

88 timer\_count <= 0;

89 else

90 timer\_owerflow <= '0';

91 end if;

92 end if;

93 end process;

94

95 LEDG(0) <= led\_state;

96

97 LEDG(1) <= KEY(1);

98

99 end Whatever;

Electrical schemes:



