**IE3-DI Digital Circuits LAB REPORT**

|  |  |
| --- | --- |
| Lab session (please tick)  1 2 3 4 | Topic: |
| Lab group (please tick)  01 02 03  Team name / number | Team members Responsible Author   (please tick) |
| 1  |
| 2  |
| 3  |

|  |  |
| --- | --- |
| correctness of **lab preparation** (hard copy) | / 5 pts. |
| **lab tasks** successfully completed (proven by protocol) | / 5 pts. |
| **lab report**  purpose and set-up of the lab tasks are properly introduced and supported by sketches, figures, flowcharts, etc. results are summarized at the end  correctness of information given in report  results are proven by e.g. measurements, simulations, calculations, source code etc. and have been discussed  correctness of formal aspects, e.g. figures with numbers and captions, citation of foreign sources, structure, spelling and style | / 3 pts.  / 2 pts.  / 2 pts.  / 3 pts. |
| **sum** |  |
|  | |

**Introduction**

The 3rd lab session was about counters (FSM, Finite State Machines) and developing a modulo-16 up/down counter with loadable preset value in VHDL based on a given specification and display the count value on a 7-segment display. Thus the objective was to gain experience in VHDL modeling, implementing, testing of sequential circuits and checking their outputs using Moore and Mealy models.

The laboratory was set up using the following tools and devices:

* Function generator ( PM 5138 )
* Digital mixed domain oscilloscope ( Tektronix MDO 3034 )
* LCD board with 7-segment display for displaying the current count value
* IOM extension board
* ModSys board ( XC2C128 – TQG144 - 7 )

**Lab Task 2.1**

The first task was to synthesize the VHDL model of mod-16 counter with load input, implement it on the CPLD and verify the correct operation of the CPLD with test cases.

Here we have a VHDL code of modulo-16 counter, which has two processes: P\_SYNC and P\_OUT. First one consists the counting with the next state logic with ARESETN and CLK signals. The second one has an output CNT which depends on three signals called S\_REG, DIR and TCEN. On the next page we can see the contents of an UCF file.

entity MOD16\_UD\_CNT\_WLOAD is

port ( TCEN, CNTEN, DIR, LOAD, ARESETN, CLK : in bit;

PRE : in unsigned (3 downto 0);

CNT : out bit\_vector (3 downto 0);

TC : out bit);

end entity MOD16\_UD\_CNT\_WLOAD;

architecture BEHAVIORAL of MOD16\_UD\_CNT\_WLOAD is

signal S\_REG : unsigned(3 downto 0);

begin

P\_SYNC : process (ARESETN, CLK)

begin

if ARESETN = '0' then S\_REG <= "0000" after 5 ns;

else

if CLK'event and CLK='1' then

if LOAD='1' then S\_REG <= PRE after 5 ns;

else if CNTEN='0' then

if DIR = '0' then S\_REG <= S\_REG - 1 after 5 ns;

else S\_REG <= S\_REG + 1 after 5 ns;

end if;

end if;

end if;

end if;

end if;

end process P\_SYNC;

P\_OUT : process (S\_REG, DIR, TCEN)

begin

TC <= '0' after 5 ns;

if TCEN='0' then

if (DIR = '0' and S\_REG = 0) or (DIR = '1' and S\_REG = 15) then TC <= '1' after 5 ns;

end if;

end if;

CNT <= to\_bitvector(std\_logic\_vector(S\_REG));

end process P\_OUT;

end;

The UCF file (shown below) allows seeing how are the CLPD board’s LEDs and signals of the VHDL code connected.

NET CLK LOC = P30 | IOSTANDARD=LVCMOS33; #GCLK0

#NET CLK LOC = P32 | IOSTANDARD=LVCMOS33; #GCLK1

#NET N\_RESET LOC = P118 | IOSTANDARD=LVCMOS33; #RESET LOW ACTIVE

NET ARESETN LOC = P35 | IOSTANDARD=LVCMOS33; # Systemreset

#IOM Board to xc2c128 Upper-Connector

############## INPUTS ##############################

#

#NET IOM\_IN\_CONN4<0> LOC = P120 | IOSTANDARD=LVCMOS33; #IN0

#NET IOM\_IN\_CONN4<1> LOC = P119 | IOSTANDARD=LVCMOS33; #IN1

#NET IOM\_IN\_CONN4<2> LOC = P118 | IOSTANDARD=LVCMOS33; #IN2

#NET IOM\_IN\_CONN4<3> LOC = P117 | IOSTANDARD=LVCMOS33; #IN3

#

#NET IOM\_IN\_CONN4<4> LOC = P116 | IOSTANDARD=LVCMOS33; #IN4

#NET IOM\_IN\_CONN4<5> LOC = P115 | IOSTANDARD=LVCMOS33; #IN5

#NET IOM\_IN\_CONN4<6> LOC = P113 | IOSTANDARD=LVCMOS33; #IN6

#NET IOM\_IN\_CONN4<7> LOC = P112 | IOSTANDARD=LVCMOS33; #IN7

#

#NET IOM\_IN\_CONN4<8> LOC = P104 | IOSTANDARD=LVCMOS33; #IN8

#NET IOM\_IN\_CONN4<9> LOC = P103 | IOSTANDARD=LVCMOS33; #IN9

#

############## OUTPUTS ################################

#NET IOM\_OUT\_CONN4<0> LOC = P129 | IOSTANDARD=LVCMOS33; #OUT0

#NET IOM\_OUT\_CONN4<1> LOC = P130 | IOSTANDARD=LVCMOS33; #OUT1

#NET IOM\_OUT\_CONN4<2> LOC = P131 | IOSTANDARD=LVCMOS33; #OUT2

#NET IOM\_OUT\_CONN4<3> LOC = P132 | IOSTANDARD=LVCMOS33; #OUT3

#NET IOM\_OUT\_CONN4<4> LOC = P133 | IOSTANDARD=LVCMOS33; #OUT4

#NET IOM\_OUT\_CONN4<5> LOC = P134 | IOSTANDARD=LVCMOS33; #OUT5

#NET IOM\_OUT\_CONN4<6> LOC = P136 | IOSTANDARD=LVCMOS33; #OUT6

#NET IOM\_OUT\_CONN4<7> LOC = P138 | IOSTANDARD=LVCMOS33; #OUT7

#NET IOM\_OUT\_CONN4<8> LOC = P10 | IOSTANDARD=LVCMOS33; #OUT8

#NET IOM\_OUT\_CONN4<9> LOC = P11 | IOSTANDARD=LVCMOS33; #OUT9

#IOM Board xc2c128 Lower-Connector

############## INPUTS ##############################

#

NET PRE<0> LOC = P56 | IOSTANDARD=LVCMOS33; #IN0

NET PRE<1> LOC = P54 | IOSTANDARD=LVCMOS33; #IN1

NET PRE<2> LOC = P53 | IOSTANDARD=LVCMOS33; #IN2

NET PRE<3> LOC = P52 | IOSTANDARD=LVCMOS33; #IN3

#

NET TCEN LOC = P51 | IOSTANDARD=LVCMOS33; #IN4

NET CNTEN LOC = P50 | IOSTANDARD=LVCMOS33; #IN5

NET DIR LOC = P49 | IOSTANDARD=LVCMOS33; #IN6

NET LOAD LOC = P45 | IOSTANDARD=LVCMOS33; #IN7

#

#NET IOM\_IN\_CONN3<8> LOC = P28 | IOSTANDARD=LVCMOS33; #IN8

#NET IOM\_IN\_CONN3<9> LOC = P26 | IOSTANDARD=LVCMOS33; #IN9

#

############## OUTPUTS ################################

NET TC LOC = P58 | IOSTANDARD=LVCMOS33; #OUT0

NET CNT<0> LOC = P59 | IOSTANDARD=LVCMOS33; #OUT1

NET CNT<1> LOC = P60 | IOSTANDARD=LVCMOS33; #OUT2

NET CNT<2> LOC = P61 | IOSTANDARD=LVCMOS33; #OUT3

NET CNT<3> LOC = P68 | IOSTANDARD=LVCMOS33; #OUT4

#NET IOM\_OUT\_CONN3<5> LOC = P69 | IOSTANDARD=LVCMOS33; #OUT5

#NET IOM\_OUT\_CONN3<6> LOC = P70 | IOSTANDARD=LVCMOS33; #OUT6

#NET IOM\_OUT\_CONN3<7> LOC = P71 | IOSTANDARD=LVCMOS33; #OUT7

#NET IOM\_OUT\_CONN3<8> LOC = P79 | IOSTANDARD=LVCMOS33; #OUT8

#NET IOM\_OUT\_CONN3<9> LOC = P80 | IOSTANDARD=LVCMOS33; #OUT9

#

Now from the following photos we can see some result of using test cases with the CPLD board.

First we can to start from 0 (0000) then check if it we can make the counter reach 6 (0110).



Picture 1.1 Test case 0 (0000)

As visible in picture 1.2, counting up to 6 was successful. Besides, we can as well look at other examples:



Picture 1.2 Test case 6 (0110)

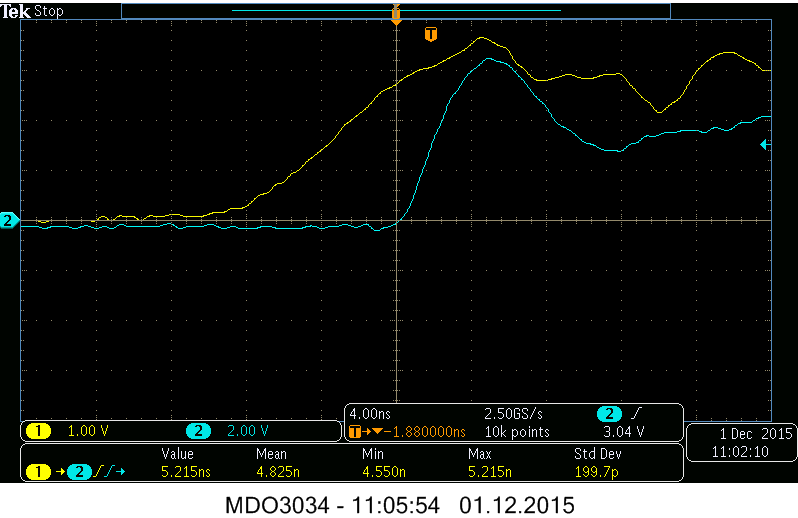
Picture 1.3 Test case 3 (0011) Picture 1.4 Test case 8 (1000)

**** ****

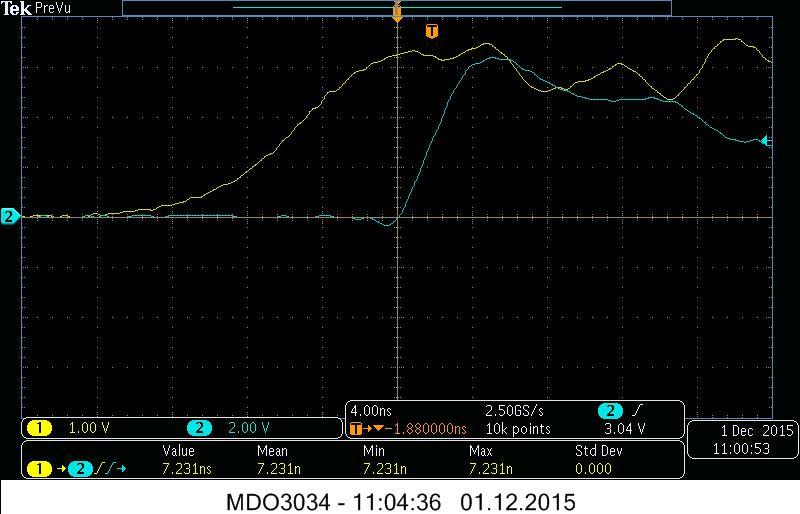
Picture 1.5 Test case 14 (1110)Picture 1.6 Test case 15 (1111)

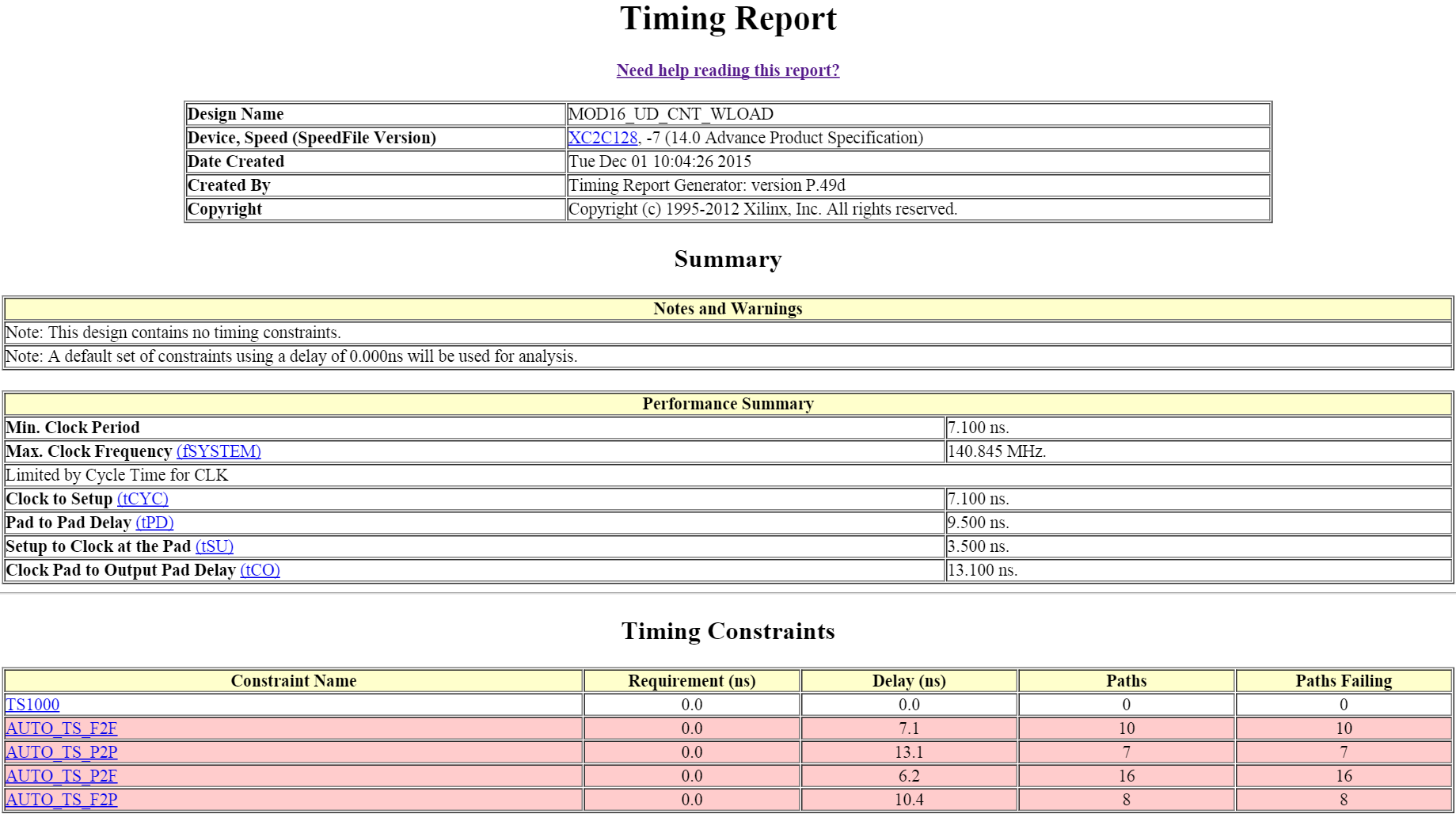
**Lab Task 2.2**

The task 2.2 was about the Propagation Delay Measurement. Firstly it was required to measure a delay between a rising clock edge and Moore output CNT. Then the propagation delay between a change of an input and the asynchronous change of the output TC had to be measured. Finally, both results had to be compared in Timing Reports.



In the first screenshot above we can see a 5.22ns propagation delay for the Moore model. For Mealy model the propagation delay result is 7.21ns as shown in picture below.





**Lab Task 2.3**

In this task we had to create a new project in Xilinx ISE Design Suite and add the VHDL model of LAB TASK 2.1 to it. Then it was required to modify the VHDL model and add the signals required for usage of the 7-seg display to the entity ports. Finally, it was asked to verify the correct operation of our implementation.

To make 7-segment display show a decimal number we had to modify the VHDL code shown in Lab Task 2.1. To be precise, we modified the entity part and output process by adding bit vectors DIG0 and DIG1.

**Modified entity part:**

entity MOD16\_UD\_CNT\_WLOAD is

port ( TCEN, CNTEN, DIR, LOAD, ARESETN, CLK : in bit;

PRE : in unsigned (3 downto 0);

DIG0 : out bit\_vector(3 downto 0);

DIG1 : out bit\_vector(3 downto 0);

CNT : out bit\_vector (3 downto 0);

TC : out bit);

end entity MOD16\_UD\_CNT\_WLOAD;

**Modified output process:**

P\_OUT : process (S\_REG, DIR, TCEN)

begin

TC <= '0' after 5 ns;

if TCEN='0' then

if (DIR = '0' and S\_REG = 0) or (DIR = '1' and S\_REG = 15) then TC <= '1' after 5 ns;

end if;

end if;

CNT <= to\_bitvector(std\_logic\_vector(S\_REG));

DIG0 <= to\_bitvector(std\_logic\_vector(S\_REG));

DIG1 <= "0000";

if S\_REG > 9 then

DIG0 <= to\_bitvector(std\_logic\_vector(S\_REG - "1010")); -- subtract 10, display the last digit

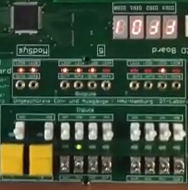
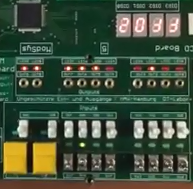
DIG1 <= "0001";

end if;

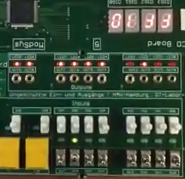
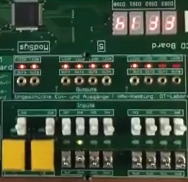
end process P\_OUT;

Since the board with 7-segment display didn’t have connection port on a side that would be comfortable for us, we just have to keep in mind that LEDs have to be “inverted” when looking at photos. The digits could be turned in the correct position by feeding "1" into the appropriate input PIN of the circuit (can be found in the Xilinx UCF file for the display), but we didn't do it for time reasons.

As seen in photos below we managed to realize the output of decimal numbers of the counter.

Picture 1.1 Test case 0 (0000) Pic 1.2 Test case 1 (0001) Pic 1.3 Test case 2 (0010)

Pic 1.4 Test case 10 (1010) Pic 1.5 Test case 14 (1110) Pic 1.6 Test case 15 (1111)

**Summary**

During lab session we learned more about counters (FSM, Finite State Machines) than we had during lectures. We also learned how to develop a modulo-16 up/down counter with loadable preset value in VHDL and got to see the differences between Moore and Mealy models with oscilloscope’s help. The most interesting experience was using Xilinx ISE Design Suite in which we could create a new project (more like fixing the existing one for our needs) and save it UCF file that allows seeing how the CLPD board’s LEDs and signals of the VHDL code are connected. With that we were able to implement the output of decimal numbers of the counter in 7-seg display.