EE120A Logic Design
Department of Electrical Engineering
University of California – Riverside



Instructor: Roman Chomko

# LABORATORY # 4
## L A B  M A N U A L

# Sequential Logic Design

---

PART 1 **Flight Attendant Call System**

PART 2 **Rising-edge Detector**

PART 3 **LED Display Time-Multiplexing Circuit***

* Note that in this part of the lab it is required to design a logic circuit and verify its performance ONLY. **DO NOT SYNTHESIZE and realize it on the Basys board**. This will be part of Lab 5 assignment.

# Objectives

**Lab 4** contains 3 parts: **Part 1** – implementation of a sequential circuit discussed in class; **Part 2** – design and implementation of a state machine; **Part 3** – design of time multiplexing circuits for four-LED display. Its purposes are to get familiar with:

1. Clock synchronous state machine design, synthesis and implementation;
2. Usage of function generator for external "clock" input for Basys FPGA boards via PMOD input/output connectors;
3. Creating and using symbol libraries within Xilinx ISE;
4. Using buses in schematic capture;
5. Vector entries in configuration files, control of external clocks

## Equipment

- PC or compatible
- Function Generator (Agilent 33120A)
- Digilent's Basys Spartan-3E FPGA Evaluation Board

## Software

- Xilinx ISE Design Software Suite 10.1
- ModelSim XE III modeling software
- Digilent's Adept ExPort Software

## Parts

- Connecting wires

## PART 1. Flight Attendant Call System

In this FPGA application development experiment, we will implement and test the "flight attendant call system" discussed in class using a function generator's SYNC output as an external driving clock. Use JA-1 PMOD pin for SYNC (clock) input.

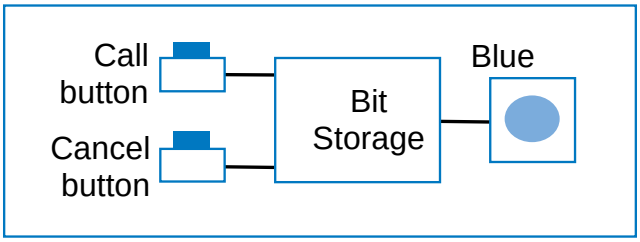
### Specification

The Flight Attendant System functions according to the following rules:

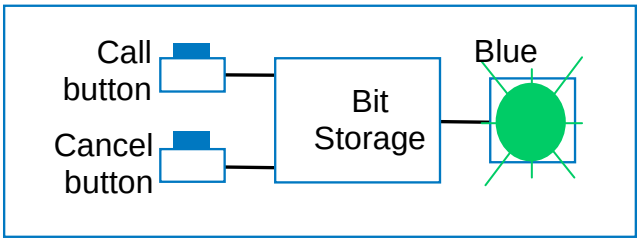
**Flight attendant call button**

- *Press* **CALL**: light turns on
  - ***Stays on*** after button released
- *Press* **CANCEL**: light turns off

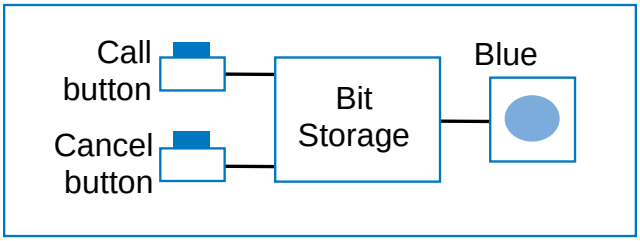
And is shown diagrammatically in **Figure 1**.




*1. Call button pressed – light turns **ON***




*2. Call button released – light **STAYS ON***




*3. Cancel button pressed – light turns **OFF***

**Figure 1.** Flight Attendant System State Machine Description

## System Analysis and Implementation

As discussed in Lecture 7 from the problem description we can obtain the following state **output/transition table**.

this is a current state

this will be latched with the rising clock edge

| Call | Cancel | Q | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Table 1.** System State Output/Transition Table

Derive **Excitation equation** which leads to the following implementation schematic:

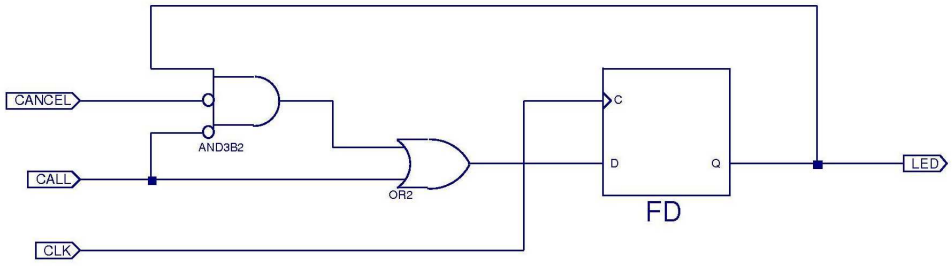



**Figure 2.** Flight Attendant System Schematic

Conduct the Behavioral Simulation.

## Digilent Basys Board Implementation Prototype

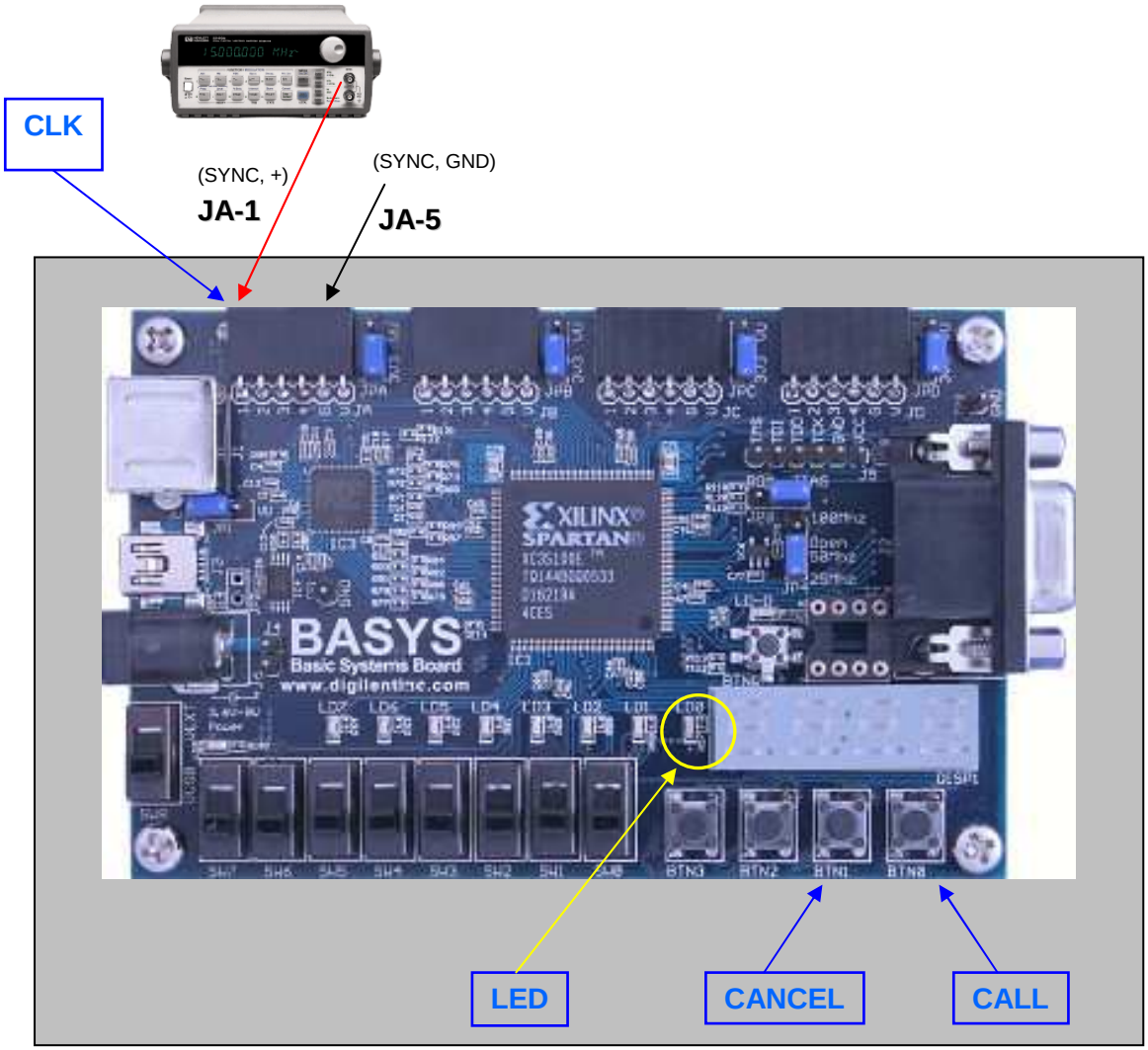



**Figure 3.** Flight Attendant System Basys Board Set-Up

## Synthesis, Mapping and Routing Procedure

Configure the Xilinx ISE project for XC3S100E-TQ144 FPGA on the Digilent's Basys board.

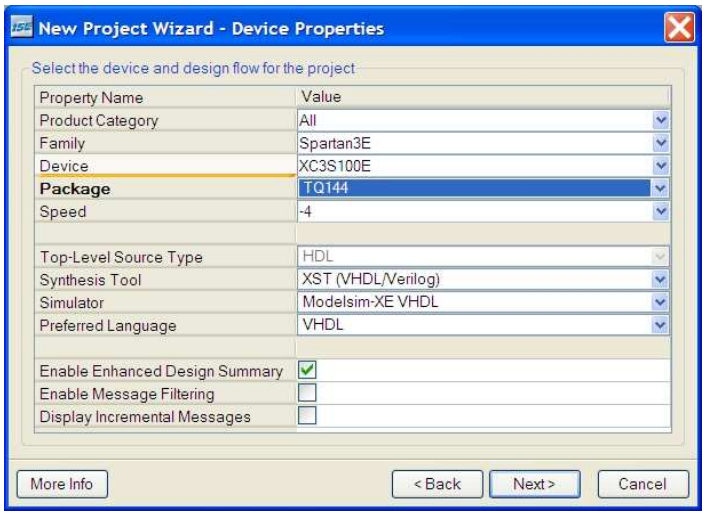



**Figure 4.** Device Properties Configuration

A function generator Agilent's 33120A SYNC signal is used as an external clock. This however makes Xilinx ISE confused as to how to map and route CLK signal to the D flip-flop in the circuit. The following actions must be taken in order to disable automatic CLK routing procedures:

1. Configuration file must contain the following CLK line which says to ignore any automatic clock routing procedures (no clock)[†]:

```
# PMOD connectors
NET "CLK"    LOC = "p81" | CLOCK_DEDICATED_ROUTE = FALSE; # JA1 = pin 1, JA5 = GND
```

2. In the Implementation/Map Options ignore user timing constraints:

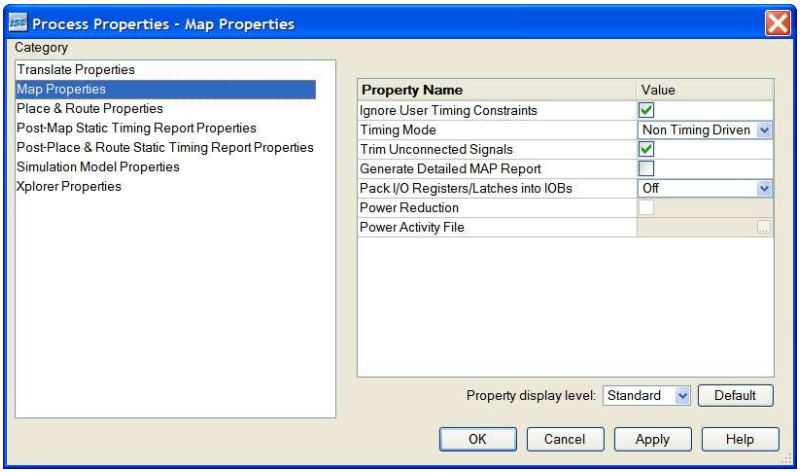



**Figure 5.** Implementation/Map Configuration

[†] There is a better way to deploy external clocks but it may be confusing for students at this point

3. In the Implementation/Place & Route Options ignore user timing constraints too:

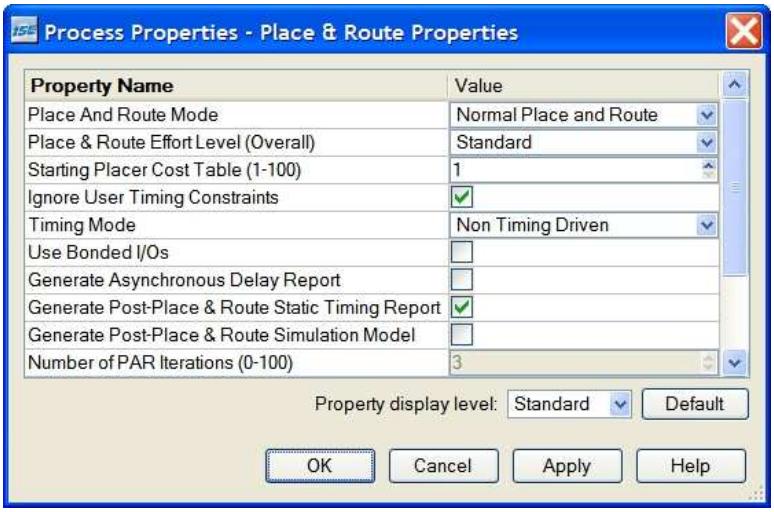



**Figure 6.** Implementation/Place & Route Configuration

4. ***Ignore*** the resulting ISE WARNINGs. The internal clock will not be used.

5. Make sure that **FPGA startup clock** is **JTAG Clock** (not the default CCLK) in Generate Programming File.

## Demonstration

Demostrate that the application performs according to specs.

## Questions

1. What will happen if the "clock" signal is of very low frequecy (1 Hz)? Experiment.
2. Design a testbench and verify the logic performance.

# PART 2. Rising-edge Detector

## Objective

In this assignment it is required to construct a Finite State Machine (FSM) state/output diagram, derive excitation equations and implement it on the Basys Board.

## Specification

The rising edge detector is a circuit that generates a short, one-clock-cycle pulse (called a ***tick***) when the input signal changes from '0' to '1'. It is usually used to indicate the onset of a slow time-varying input signal.

**NOTE:**

1. Use the signal generator's SYNC output for external clocking;
2. Use one of the switches to emulate the input signal;
3. Set clock signal to a sufficiently low frequency to clearly see the LED one-clock-cycle flash

## Demonstration

1. Derive a state diagram from the spec's description;
2. Show the output/transition table
3. Derive the excitation equations
4. Design a sequential logic circuit that implements the excitation equation
5. Verify the circuit performance by testbench simulation
6. Implement the application using on-board component of your choice. For clocking procedure use information from PART 1 of the lab above.

# PART 3. LED Display Time-Multiplexing Circuit ‡

## Specification

The Digilent Basys Board contains four seven segment LED displays with decimal points. To reduce the number of used of FPGA's I/O pins it is required to use a time-multiplexing sharing scheme. That is, the four displays have heir enable signals but share eight common signals to light the segments. All signals are active-low (i.e., enabled when a signal is '0'). The schematic of displaying '3' on the right-most LED is shown in **Figure L4P3-1**.

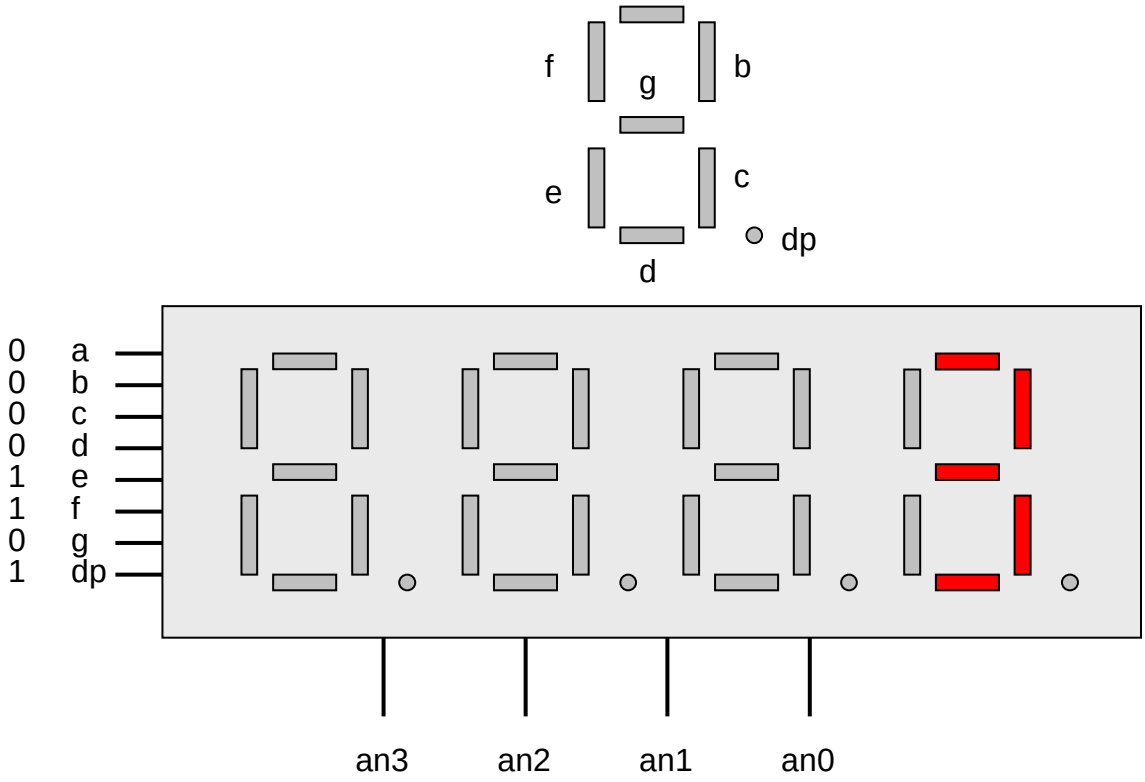



**Figure L4P3-1.** Displaying '3' on the LED display

Note that the enable signal (i.e., an) is '1110'. This configuration clearly can enable only one display at a time. We can time-multiplex the four LED patterns by enabling the four displays in turn, as shown in the simplified timing digram in **Figure P4P3-2**. If the refreshing rate of the enable signal is fast enough, the human eye cannot distinguish the on and off intervals of the LEDs and perceives

‡ Note, in this part of the lab it is required to design a logic circuit and verify its performance ONLY. DO NOT SYNTHESIZE and realize it on the board. This will be part of Lab 5 assignment.

that all four displays are lit simultaneously[§]. This scheme reduces the number of I/O pins from 32 to 12 (i.e., eight LED segments plus four enable signals) but requires a time multiplexing circuit.

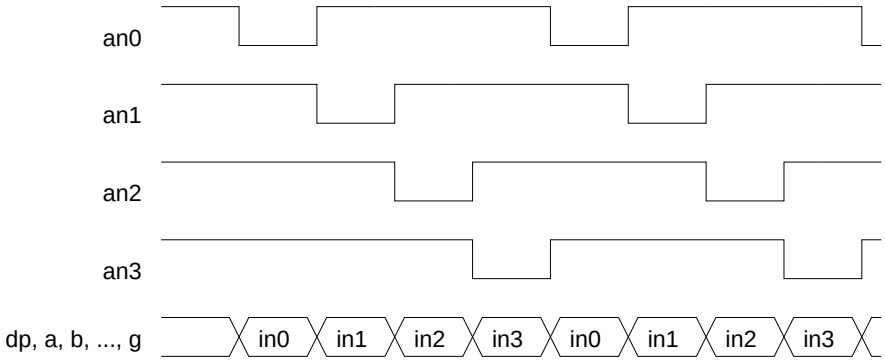



**Figure L4P3-2**. Timing diagram of time multiplexing circuit

One of possible realizations is shown in the block diagram of **Figure L4P3-3**. Use it as a guide to implement the circuit and verify (simulate ONLY) its performance.

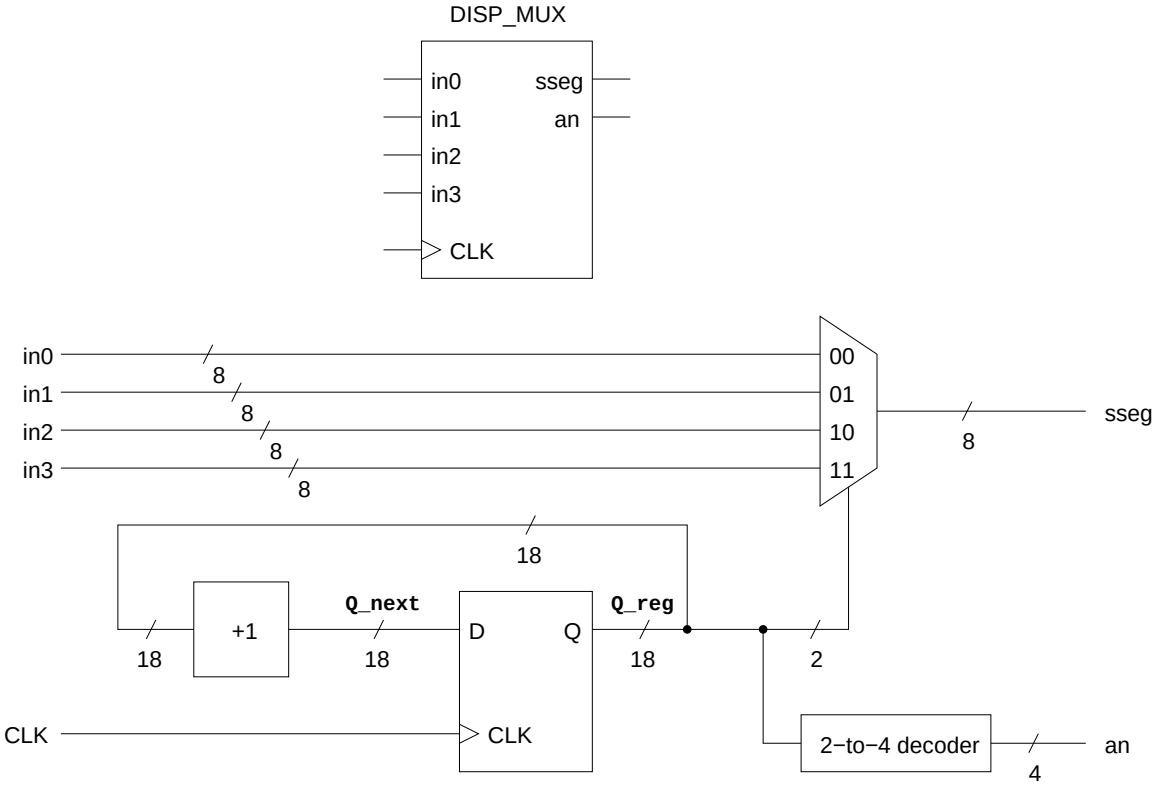



**Figure L4P3-3**. Symbol and Block diagram of the time-multiplexing circuit

[§] You may have observed damaged traffic lights where LEDs were flickering

## Schematic Utility

To expedite correct schematic capture** copy-paste the following VHDL code into a "New VHDL Source Code" and create a schematic symbol for it which can be used to further manipulate the circuit logic.

```
-- ----------------------------------------------------------
-- File:    hex2led.vhd
-- Created: R.Chomko
-- Date:    2009/01/25
-- Purpose: decodes Hex (+BCD, +dp) into 7 segment LED Display
--
-- ----------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity HEX_TO_LEDSEG is
   port(
      hex:    in  STD_LOGIC_VECTOR(3 downto 0);
      dp:     in  STD_LOGIC;
      ledseg: out STD_LOGIC_VECTOR(7 downto 0)
   );
end HEX_TO_LEDSEG;

architecture ARCH of HEX_TO_LEDSEG is
begin
   with hex select
      ledseg(6 downto 0) <=
         "0000001" when "0000", -- 0
         "1001111" when "0001", -- 1
         "0010010" when "0010", -- 2
         "0000110" when "0011", -- 3
         "1001100" when "0100", -- 4
         "0100100" when "0101", -- 5
         "0100000" when "0110", -- 6
         "0001111" when "0111", -- 7
         "0000000" when "1000", -- 8
         "0000100" when "1001", -- 9
         "0001000" when "1010", -- a
         "1100000" when "1011", -- b
         "0110001" when "1100", -- c
         "1000010" when "1101", -- d
         "0110000" when "1110", -- e
         "0111000" when others; -- f
      ledseg(7) <= dp;          -- dec.point
end ARCH;
```

**Listing 5.** HEX-TO-LEDSEG VHDL Code that can also encode DP and Hex Symbols

## Demonstration

Demonstrate testbench simulation results and all the supporting material used in designing the system.

** Note that material of Lab 3 can be used if it behaves correctly

## Procedures

1. Xilinx ISE Design and Synthesis environment;
2. Creation of Configuration files;
3. Usage of Adept ExPort download software;

## Presentation and Report

Must be presented according to the general EE120A lab guidelines posted in iLearn.

## Prelab

1. Familiarize yourself with ISE and ModelSim tutorials posted in iLearn;
2. Review Lectures 7-10;
3. Try to answer all the questions, prepare logic truth tables, do all necessary computations