# Lab Member

Table Number : <Insert your table number here>

| Name | Student Number |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

# 

# Objectives

1. Introducing the workflow of the FPGA development.
2. Understand the combinational logic on Verilog.

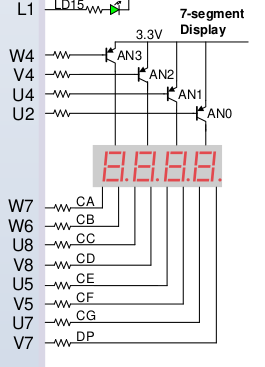
# 7-Segments Display on Basys3 Board

The Basys 3 board features a four-digit common anode seven-segment LED display. Each digit consists of 8 LEDs, corresponding to segments A, B, C, D, E, F, G, and DP, as illustrated in Figure 1. All LEDs within the same digit share a common anode. Additionally, identical segments across all digits share the same cathode; for instance, the "A" segments in all digits are connected to the same cathode pin, labeled CA. This configuration minimizes the number of pins required to control the 7-segment display.

# 

**Figure 1 : 7-Segment Display Configuration**

To illuminate each segment, the corresponding anode (AN) must be set to 0. As shown in Figure 2, each anode is connected to a transistor, and setting the corresponding cathode (CX) to 0 allows current to flow from the anode to the cathode, thereby turning on the segment.

****

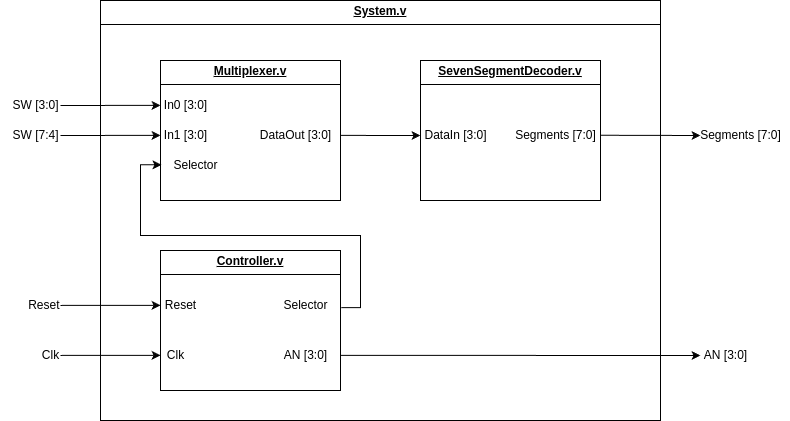
**Figure 2 : 7-Segment Display Pin**

To illuminate multiple digits simultaneously, a technique called multiplexing is used. This involves rapidly turning each digit on and off at a rate imperceptible to the human eye. However, this functionality is not the focus of this lab exercise and the module handling this functionality will be provided.

# 

# Lab Exercise

In this lab exercise, you are tasked with developing a system that takes input from 8 onboard switches (SW0 to SW7) to represent two hexadecimal values. SW0 to SW3 form the first hexadecimal value, with SW0 as the least significant bit (LSB), while SW4 to SW7 form the second hexadecimal value, with SW4 as the LSB. The system should display the first hexadecimal value on the first digit of the 7-segment display and the second hexadecimal value on the second digit. The third and fourth digits of the 7-segment display will not be used.



**Figure 3: System Overview.**

Figure 3 illustrates the System Overview to be implemented in this lab exercise. The system comprises four modules: **System**, **Multiplexer**, **SevenSegmentDecoder**, and **Controller**. The roles of each module are as follows:

1. **System Module**: Serves as the top-level module, integrating and connecting the other submodules.
2. **Multiplexer Module**: Selects which hexadecimal value is sent to the SevenSegmentDecoder module for translation into control signals for the 7-segment display.
3. **SevenSegmentDecoder Module**: Converts a 4-bit binary input into control signals for the segments of the 7-segment display.
4. **Controller Module**: Controls the on/off cycling of each digit on the 7-segment display to ensure all digits appear visible to the observer. It does this by sending a selector signal to the Multiplexer to determine the active switch data and manipulating the AN signal to turn each digit on or off. This module is pre-implemented for you.

## Part 1 : Multiplexer

In this part of the lab, you are tasked with completing the **Multiplexer** module. The module has 3 input ports: **In0[3:0]**, **In1[3:0]**, and **Selector**, and 1 output port: **DataOut[3:0]**. The module must be implemented as combinational logic according to the truth table provided below.

| Selector | DataOut [3:0] |
| --- | --- |
| 0 | In0 |
| 1 | In1 |

**Table 1: Multiplexer Truth table**

### Instruction

1. Create a new project and import all the necessary files from the following source: <https://github.com/2110363-HW-SYN-LAB/lab/tree/main/Lab1> or MCV. Follow the instructions provided in the Lab1 Guide file for setup.

2. Modify the **Multiplexer** module to implement the behavior described in the truth table above

Insert your modified Multiplexer module here

|  |
| --- |

3. Run the testbench to verify that your module functions correctly.

Insert your Xilinx testbench (MultiplexerTB) result here.

|  |
| --- |

Insert your Cocotb testbench result (from folder /cocotb/MultiplexerTB/) here.

|  |
| --- |

Insert your testbench result waveform here. (Either Cocotb or Xilinx Testbench)

|  |
| --- |

4. Call TA to inspect your work.

## 

## Part 2 : 7-Segment Decoder

In this part of the lab, you are tasked with completing the **SevenSegmentDecoder** module. This module has 1 input port: **DataIn[3:0]** and 1 output port: **Segments[7:0]**. The output segments are assigned as follows: **[CA, CB, CC, CD, CE, CF, CG, DP]**, where **CA** corresponds to **Segments[7]** and **DP** corresponds to **Segments[0]**.

This module must be implemented as combinational logic. The **DP** signal controls the decimal point (dot) on the 7-segment display. For this lab exercise, the decimal point should remain turned off.



**Figure 4 : 7-Segment Display of hexadecimal value**

### Instruction

1. Complete the truth table below, which converts 4 bit **DataIn** to the corresponding **Segments[7:0]**.

| DataIn [3:0] | Segments [7:0] |
| --- | --- |
| 0000 |  |
| 0001 |  |
| 0010 |  |
| 0011 |  |
| 0100 |  |
| 0101 |  |
| 0110 |  |
| 0111 |  |
| 1000 |  |
| 1001 |  |
| 1010 |  |
| 1011 |  |
| 1100 |  |
| 1101 |  |
| 1110 |  |
| 1111 |  |

3. Create a testbench for this module to verify that it works correctly. You will need to create both **Xilinx** and **Cocotb** testbenches:

Insert your Xilinx testbench file (SevenSegmentDecoderTB) here.

|  |
| --- |

Insert your Cocotb testbench file (from folder /cocotb/SevenSegmentDecoderTB/) here.

|  |
| --- |

4. Modify the **SevenSegmentDecoder** module to work according to the truth table.

Submit your modified 7SegmentDecoder module here.

|  |
| --- |

5. Run the testbench and report the result here.

Insert your Xilinx testbench (SevenSegmentDecoderTB) result here.

|  |
| --- |

Insert your Cocotb testbench (from folder /cocotb/SevenSegmentDecoderTB/) result here.

|  |
| --- |

Insert your testbench result waveform here. (Either Cocotb or Xilinx Testbench)

|  |
| --- |

6. Call TA to inspect your work.

## Part 3 : System Integration

Now that you have completed all the submodules, it's time to integrate them into the **System** module, which will serve as the top module.

### Instruction

1.Modify the System module to connect all the submodules according to figure 3.

Submit your Modify System Module here

|  |
| --- |

2. Run the testbench to verify your module (either Xilinx or Cocotb one).

Insert your testbench (SystemTB) result here.

|  |
| --- |

3. Call TA to inspect your work.

## Part 4 : Programming the Hardware

Now that the entire system is complete and functions correctly according to the testbench, it is time to generate the bitstream and program it onto the **Basys3** board. To do this, we first need to create a **constraint file** to map the system's inputs and outputs to the appropriate pins on the **Basys3** board. Afterward, we will run synthesis, implementation, and generate the bitstream.

### Instruction

1. Complete this mapping table below to map your system's inputs and outputs onto the **Basys3** pins. This system uses **BTNU** for resetting purposes, and **AN[x]** is used to enable the corresponding digit of the 7-segment display.

Hint, look up the Basys3 reference manual page 6 and 15.

| **System.v input/output** | **Basys3 Pins** |
| --- | --- |
| Reset |  |
| Clk |  |
| SW[0] |  |
| SW[1] |  |
| SW[2] |  |
| SW[3] |  |
| SW[4] |  |
| SW[5] |  |
| SW[6] |  |
| SW[7] |  |
| AN[0] |  |
| AN[1] |  |
| AN[2] |  |
| AN[3] |  |
| Segments[0] |  |
| Segments[1] |  |
| Segments[2] |  |
| Segments[3] |  |
| Segments[4] |  |
| Segments[5] |  |
| Segments[6] |  |
| Segments[7] |  |

1. Create the constraint file. Follow the instructions in the Lab1 Guide.

Submit your constraint file here.

|  |
| --- |

1. Run the synthesis and then open the Schematic.

Explain what you observed in the schematic and submit a picture of the schematic here.

|  |
| --- |

1. Run the implementation, generate the bitstream, and program the **Basys3** device.

Take a picture of your Board working after programming.

|  |
| --- |

1. Call TA to inspect your work.
2. Submit this sheet to the MCV.