RMIT UNIVERSITY

SCHOOL OF ENGINEERING

EEET2162/2035 – ADVANCED DIGITAL DESIGN 1 / DESIGN WITH HARDWARE DESCRIPTION LANGUAGES

LABORATORY 2

SEVEN SEGMENT DECODER

1 AIMS

(i) To design, simulate, implement and test a digital circuit using the Quartus Prime toolchain.

(ii) To demonstrate the workflow when using the Verilog HDL to construct a design for a physical Field Programmable Gate Array (FPGA) target.

(iii) To utilise modern electronic test equipment to confirm the final circuit implementation.

2 INTRODUCTION

Students are expected to work in individually and submit a report which will be assessed by the laboratory demonstrator. This particular laboratory session will run for one week (week 3), however the submission will occur at the end of week 5 (soft copy to the subject Canvas website) .

In this laboratory you will be required to design, simulate, implement and test a Binary Coded Decimal (BCD) seven-segment decoder. The individual segments will be represented using LED6 through to LED0 (MSB to LSB) on DE-10 Nano Development Board. Furthermore, the slide-switches SW3 – SW0 (MSB to LSB) will be used to input the BCD value into the decoder.

Seven segment decoders are used convert a binary value into a digit that can be readily displayed on a LED display. The outputs from the decoder turn on LEDs which are used to form part of a larger display. Figure 1 depicts the standard structure of a seven segment display, where the letters 'a' - ' 'g' depict the individual LEDs.

On the DE-10 Development board the LEDs are active high and as such a logic 1 will illuminate the LED. To simulate the seven segment display itself we will be assuming a common cathode configuration where as the name suggests, the cathodes of the LED elements are tied together, typically directly to ground.

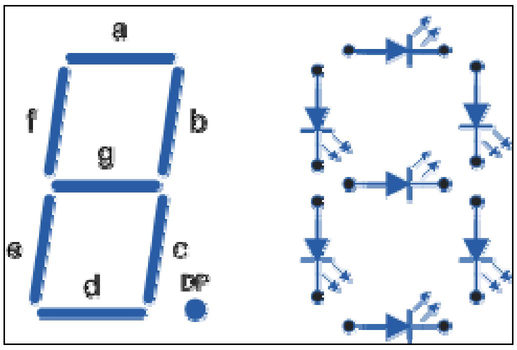

Figure 1 - Seven Segment Display [1]

By illuminating the individual segments on the display alternative digits can be rendered. Typically these displays are used to display numeric values from 0 - 9, however they can be used to display a subset of the alphabet. In this laboratory we are only interested in displaying numeric values. Note that the decimal point (DP) has not been considered, however is supported by some display units.

The truth-table to display the digits from 0 - 9 appears in Table 1. Give that four slide switches are used to form the input binary number, there are sixteen possible combinations (0 - 15). For display outputs that are not listed in Table 1, all LEDs should be disabled (off).

Segment Inputs							Display Output
a	b	c	d	e	f	g	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	1	1	9

Table 1 - Seven Segment Display Segment Mapping

As part of the Quartus project development you will need to specify a set of physical inputs and outputs that can be used to exercise the entire BCD inputs. The mapping for the various interfaces appears in Table 2.

When configuring a project in Quartus it is important to select the correct target FPGA. For the DE10-Nano Development board, the target device is a 5CSEBA6U23I7DK. If an incorrect device is selected, then the configuration for the Logic Elements (LEs) will be incorrect and hence the system will not function.

Function	Mapping	Physical I/O	I/O Standard
SW0	BCD_BIT0	Y24	3.3-V LVTTL
SW1	BCD_BIT1	W24	3.3-V LVTTL
SW2	BCD_BIT2	W21	3.3-V LVTTL
SW3	BCD_BIT3	W20	3.3-V LVTTL
LED0	a	W15	3.3-V LVTTL
LED1	b	AA24	3.3-V LVTTL
LED2	c	V16	3.3-V LVTTL
LED3	d	V15	3.3-V LVTTL
LED4	e	AF26	3.3-V LVTTL
LED5	f	AE26	3.3-V LVTTL
LED6	g	Y16	3.3-V LVTTL

Table 2 - DE10-Nano Pin Mapping

3 DESIGN AND SIMULATION

The first part of the laboratory involves developing a behavioural model in Verilog. Whilst the mapping between the segment output logic could be implemented using a structural modelling approach (discrete primitive gates), the benefits of using Verilog to describe the overall functionality will be demonstrated in this laboratory. The required tasks are as follows (show all working where necessary):

a) Create a functional block diagram depicting the overall inputs and outputs to the system. This will assist in the development of the module code in the top-level entity. The direction of the I/O should be clearly displayed.

b) Create a truth-table that links the various switch positions to the required outputs. Ensure that you include the decimal equivalents for the switch positions and the display as it would appear on the simulated seven-segment display. The developed table will be used for verification purposes in both simulation and once the design is deployed on the actual hardware.

c) The next step is to produce a Verilog project in Quartus. Launch Quartus, and create a new project via the 'File -> New Project Wizard' option. Ensure that the 'Working Directory' is set to a drive where you have write permissions (*c:\scratch\student_number*). The name of the project and top-level entity should be set to a useful descriptor. The remaining project entries, with the exception of the device selection ('Family, Device & Board Settings') can be left at their default values. Ensure that you set the actual device to 5CSEBA6U23I7DK by entering the value in the 'Name Filter'. Step through the dialogs and press 'Finish' when complete to build the project.

d) To invoke the Verilog editing tool, select 'File-> New' and then a 'Verilog HDL File'. A blank editor window will open with the filename Verilog1.v. Within this window the behavioural seven segment decoder will be created. As this is a relatively simple design the solution can be placed in the one file. Start your design by creating the module definition. Ensure that you fully comment the code as it will assist in debugging your solution. The lecture notes provide a sample on the syntax of the module definition. Once the module definition is complete, save the Verilog design. Hint: Create the input from the switches as a vector and set the individual outputs for the segments a single bit values.

e) As the name suggests, when creating a behavioural simulation the underlying logic details are synthesised by Quartus. In a behavioural simulation we can utilise standard program control statements such as 'if' and 'switch / case'. In this style of model an 'always' block is used to signify when certain events should occur. In advanced models, the 'always' block would contain a sensitivity list which would cause the block to be 'executed' under certain conditions. In this example, the 'always' block should have a sensitivity list of *. The following code segment can be used to create the block:

```
// Module declaration occurs here.

always @(*)           // Sensitivity list of all events
     begin
                      // Your code goes here

     end

// End of module declaration occurs here.
```

f) As previously mentioned, within the 'always' block standard program control statements can be utilised. In this particular laboratory the 'if' statement is useful. To utilise an 'if' statement, the following code segment can be utilised:

```
if (conditionToCheck == conditionValue)
     // Condition if correct.
else
     // Condition if false.
```

Based on the above 'if' statement, create an 'if-else-if' ladder for the seven segment truth table. Note that a 'reg' variable should be set within the 'always' block rather than the main segnebt outputs.

g) Once the behavioural aspects are complete, the output to the actual seven segments need to be configured. This can occur via the 'concatenation' operator. The basic form of this operator is as follows (this should be outside of the 'always' block):

```
assign {output0, output1, ....} = reg_variable;
```

h) Before a simulation of the design can commence the pins need to be mapped to the physical FPGA. From the Quartus main toolbar, select ‘Start Analysis and Synthesis’. This process will evaluate the schematic that has been created and determine the required external I/Os. Errors in this process need to be addressed before proceeding. If the synthesis is successful, then the ‘Pin Planner’ can be launched by the ‘Assignments Menu’. As its name suggests the ‘Pin Planner’ displays an overview of the FPGA pins that can be connected to the Schematic / HDL models created. Assign the pins as demonstrated in Table 2. Once completed, close the ‘Pin Planner’.

i) The next step in the process is to perform a simulation of the logic developed. Within Quartus there are two different options for simulation: ModelSim and the Simluation Wave Editor. As this is a relatively simple design, the Wave Editor will be used. To create the waveforms to verify the design, from the Quartus menu select ‘New->University Program VWF’. The individual nodes (I/Os) need to be added into the simulation. From the menu select ‘Edit -> Insert Node or Bus’ and then click on ‘Node Finder’. Click the ‘List’ button, select all of the nodes and press the ‘>>’ symbol to add all nodes into the simulation. Close all dialogs by selecting ‘OK’.

j) Once the process is successful all inputs and outputs will be displayed as waveforms (by default inputs will be ‘0’ and outputs ‘X’ (undefined). To exercise all of the input combinations (16 in total) the waveform needs to be manually created. At time increments of 100ns, the input states should change. By selecting the individual ranges (per input) the logic level can be changed. Note that only the ‘Forcing low’ and ‘Forcing high’ buttons should be used to set the logic levels. Furthermore, to assist with entering of waveforms the grid should be changed to 100ns. To perform the actual simulation select ‘Simulation->Run Functional Simulation’ from the waveform editor window. Confirm that you actual solution matches the truth table in Table 1 based on the input switch values.

4 IMPLEMENTATION

The final section of this laboratory is to implement the actual design on the DE10-Nano Hardware Development platform. If the simulation is successful then it is relatively simple process. To begin the compilation process only the main Quartus Window should be open.

a) Although the simulation has been developed and verified, it is necessary to create the configuration that will be downloaded to the FPGA. This is achieved by selecting the ‘Process ->Start Compilation’ option from the Quartus main window. Note that depending on the complexity the compilation can take between 5 – 10 minutes. Once complete, a binary configuration file is ready to be uploaded to the development hardware.

b) Connect the DE10-Nano to the PC using the USB-Blaster port (located near the DC power port). From the ‘Tools’ menu select ‘Programmer’. The development hardware will be detected automatically. As discussed in the lectures the platform that we are using is a System-on-Chip (SoC) and hence both the FPGA and Hard Processor System (HPS) will appear in the JTAG programming chain. To ensure successful deployment both the FPGA and HPS need to be entered into the device programming tool. This can be active by selecting the ‘Autodetect’ button. The additional device to add is a 5CSEBA6.

c) Once the new device has been added the programming file will need to be updated. Select the 5CSEBA6 device in the upper window and then press the ‘Change File’ button. Navigate to the ‘output_files’ directory and select the *project_name.sof* file. Select the device again in the upper window and press the ‘Program / Configure’ checkbox. The ‘Start’ button should now become active which you can press to send the configuration to the development board.

d) Now that the device has been programmed, exercise the switches and the extended truth-table developed in Section 3b) and confirm that is operates as expected.

5 REPORT

a) <u>Record all results and observations. All calculations, schematics, waveforms, RTL diagrams and the corresponding discussions should be present in the report.</u>

6 IMAGE REFERENCES

[1] https://www.electronics-tutorials.ws/wp-content/uploads/2013/08/comb15.gif