EGR680 High Level Implementation on FPGA

Laboratory 02

Seven-Segment Display Applications Using PYNQ

Professor: Dr. C. Parikh

Student: Dimitri Häring

September 12, 2018

Contents

1 Introduction

The goal of the lab 2 is to familiarize the student with decoders and to introduce and teach top level hierarchy.

2 Design

In this section the design and decisions that where made to achieve it are discussed.

2.1 Part I - Behavioral Modeling of a Seven-Segment Display

Verilog is used to describe a decoder that shall control a seven-segment display with two switches and one button. The given task is as followed:

In this part, you will simulate a pattern decoder using the buttons, switches, and a single seven-segment display. Using switches, SW0 and SW1 as the pattern input, write a Verilog program that takes the binary input combinations of the switches and displays the decimal value of the binary switch combination on the seven-segment display. Table 1 shows the switch combinations with the expected output value on the seven-segment display.

Table 1: Input Combinations and Expected Output

SW1	SW0	BTN0	Display
0	0	0	OFF
0	0	1	0
0	1	1	1
1	0	1	2
1	1	1	3

As software package to implement the decoder in Verilog Vivado 2017.2 is used.

2.2 Part II - Hardware Implementation & Modular Design

For the second part a hierarchical design is achieved including the implementation and generation of a bit stream to use hardware or more specific the PYNQ development board. Therfore the folowing task is given:

In this part, you will modify your code from Part I. Instead of only using one button and one switch, you will design your Verilog code to display on four seven-segment displays by pressing the four buttons on the PYNQ board. Each seven-segment display will correspond to one button (i.e., BTN0 lights up the right-most seven-segment display and BTN1 will light up the adjacent seven-segment display and so on). Make sure that when a button is released, the corresponding seven-segment display turns OFF. Only one seven-segment display should be on at any given time.

To realize the constrained file the PYNQ reference manual was used to allocate the right pins for the switches and buttons, figure 1 shows the basic I/O pins assignment of the PYNQ development board.

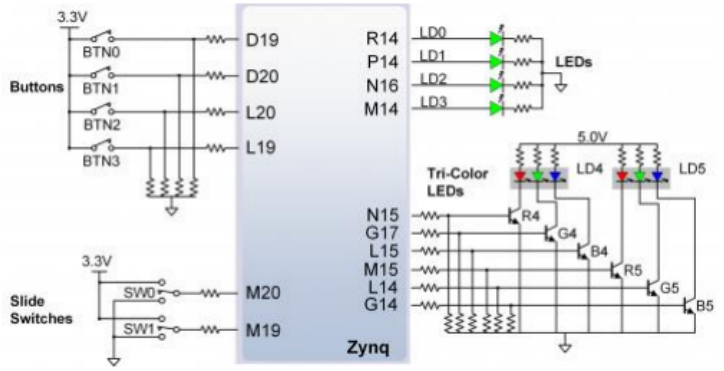

Figure 1: Schematic PYNQ Basic I/O. [1]

The one of solution to part II is two concatenated case statements the first to check for the buttons and the second to switch the right seven-segment display to show the number as defined by the two switches SW0 and SW1. The used code is listed in the following three listening 3, 4, and 5.

3 Simulation

Describes the result of the behavioral simulation based on the synthesized hardware description language.

3.1 Part I: behavioral simulation of decoder

After a successful simulation of the synthesized hardware description language that implements an decoder a test bench was written, see listening 2. The time variant simulation is shown in figure 2 which shows steps trough the different possible switch positions and shows the output on the decoder bus seg. In comparison to the simulation given in Lab 02 Part 1 it could be confirmed that there are mostly identical.

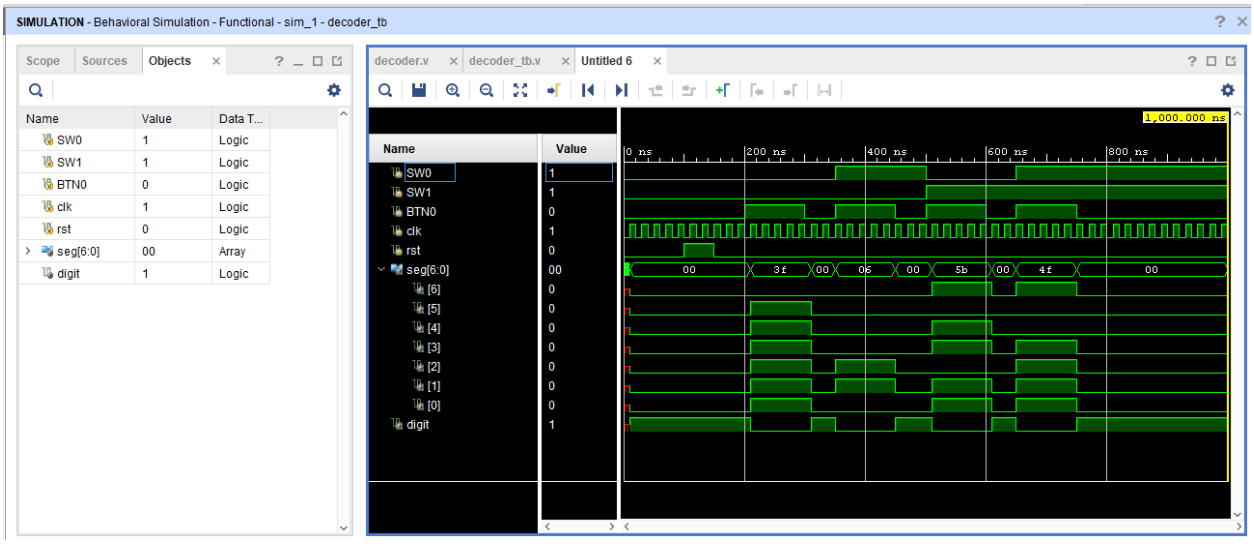

Figure 2: Vivado behavioral simulation of the decoder module.

4 Conclusion

5 Appendix

The appendix contains code listening and other large information parts that contain partial or complete relevance to the reports topic.

5.1 Lesson Learned

Figure 3 shows a handy tool build into Vivado software package that is called Language Templates and it seems to contain most of the verilog syntax.

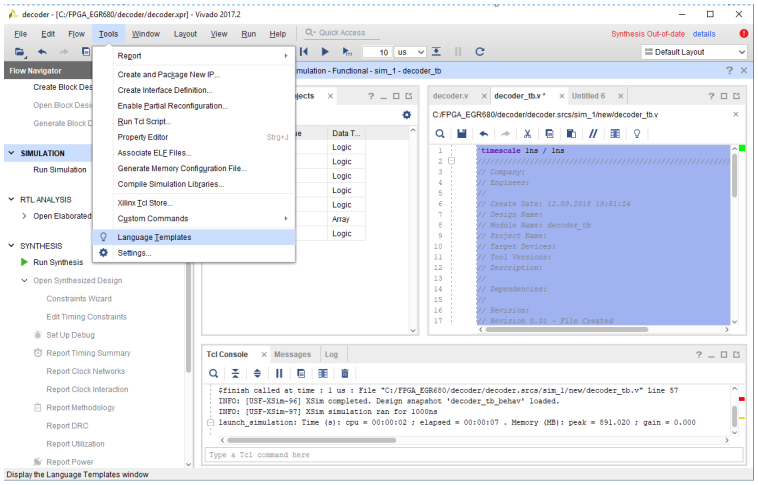

Figure 3: Vivado Language Templates.

5.2 Part I:Decoder module

Listing 1: Testbanche decoder part I.

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12.09.2018 19:31:54
// Design Name:
// Module Name: decoder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module decoder(
input SW0,
input SW1,
input BTN0,
input clk,
input rst,
output [6:0] seg,
output digit
);

reg [6:0] seg;
reg digit;

always @(posedge clk or posedge rst)
begin
if(rst)
begin
seg <= 7'b0000000;
digit <= 1'b1;  // Digit = 1 (7-segment OFF Digit= 0 (7-segment ON)
end
else
begin
// my code or from webpage http://verilogcodes.blogspot.com/2015/10/verilog-code-for-b
digit = ~BTN0;
if (BTN0 == 1) begin
case ({SW1, SW0}) //case statement
2'b00 : seg =  7'b0111111; //~7'b0000001;
2'b01 : seg =  7'b0000110;
2'b10 : seg =  7'b1011011;
2'b11 : seg =  7'b1001111;
4 : seg = ~7'b1001100;
5 : seg = ~7'b0100100;
6 : seg = ~7'b0100000;
7 : seg = ~7'b0001111;
8 : seg = ~7'b0000000;
9 : seg = ~7'b0000100;
//switch off 7 segment character when the bcd digit is not a decimal number.
default : seg = 7'b0000000;
endcase
end
else begin
seg = 7'b0000000;
end
end
end
endmodule
```

5.3 Part I: Behavioral simulation of decoder code test bench

Listing 2: Testbanche decoder part I.

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company:
```

```
// Engineer:
//
// Create Date: 12.09.2018 19:51:24
// Design Name:
// Module Name: decoder_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module decoder_tb;

reg SW0, SW1, BTN0, clk, rst;
wire [6:0] seg;
wire digit;

decoder X1(SW0, SW1, BTN0, clk, rst, seg, digit);  // Initialistaion

initial
begin
SW0 = 0;
SW1 = 0;
BTN0 = 0;
clk = 0;
rst = 0;
end

always #10 clk = ~clk;

initial begin
#100;
rst = 1; #50;
rst = 0; #50;
SW0 = 0; SW1 = 0; BTN0 = 1; #100;
BTN0 = 0; #50;
rst = 0; SW0 = 1; SW1 = 0; BTN0 = 1; #100;
BTN0 = 0; #50;
rst = 0; SW0 = 0; SW1 = 1; BTN0 = 1; #100;
BTN0 = 0; #50;
rst = 0; SW0 = 1; SW1 = 1; BTN0 = 1; #100;
BTN0 = 0;

end
initial #1000 $finish;
endmodule
```

5.4 Part II: Decoder module

Listing 3: Decoder part II.

```
module decoder(
input SW0,
input SW1,
input BTN0,
input BTN1,
input BTN2,
input BTN3,
input clk,
//input rst,
output [6:0] seg0,
output [6:0] seg1,
output cat0,
output cat1
//output digit
);

reg [6:0] seg0;
reg [6:0] seg1;
reg cat0;
reg cat1;

always @(posedge clk )
begin
case({BTN3,BTN2,BTN1,BTN0})
4'b0000:
begin
seg0 = 7'b0000000;
seg1 = 7'b0000000;
end
/*-------------------------------------------------*/
4'b0001:
begin
cat0 <= 1'b0;
case ({SW1, SW0}) //case statement

2'b00 : seg0 =  7'b0111111;//0
2'b01 : seg0 =  7'b0000110;//1
2'b10 : seg0 =  7'b1011011;//2
2'b11 : seg0 =  7'b1001111;//3
endcase
end
4'b0010:
begin
cat0 <= 1'b1;
case ({SW1, SW0}) //case statement

2'b00 : seg0 =  7'b0111111;//0
2'b01 : seg0 =  7'b0000110;//1
2'b10 : seg0 =  7'b1011011;//2
2'b11 : seg0 =  7'b1001111;//3
endcase
```

```
end
4'b0100:
begin
cat1 <= 1'b0;
case ({SW1, SW0}) //case statement

2'b00 : seg1 =  7'b0111111;//0
2'b01 : seg1 =  7'b0000110;//1
2'b10 : seg1 =  7'b1011011;//2
2'b11 : seg1 =  7'b1001111;//3
endcase
end
4'b1000:
begin
cat1 <= 1'b1;
case ({SW1, SW0}) //case statement

2'b00 : seg1 =  7'b0111111;//0
2'b01 : seg1 =  7'b0000110;//1
2'b10 : seg1 =  7'b1011011;//2
2'b11 : seg1 =  7'b1001111;//3
endcase
end
default :
begin
seg0 = 7'b0000000;
seg1 = 7'b0000000;
end

endcase
end
endmodule
```

5.5 Part II: Decoder Top Level

Listing 4: Decoder top level part II.

```
`timescale 1ns / 1ns
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12.09.2018 23:03:22
// Design Name:
// Module Name: decoder_top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
```

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module decoder_top(
input clk,
input SW0,
input SW1,
input BTN0,
input BTN1,
input BTN2,
input BTN3,
output cat0,
output cat1,
output [6:0] seg0,
output [6:0] seg1
);

/* Intaniate modules */
decoder dec1(SW0, SW1, BTN0, BTN1,BTN2,BTN3,clk, seg0,seg1, cat0,cat1 );
endmodule
```

5.6 Part II: Decoder Constrains

Listing 5: Decoder Constrains part II.

```
## Clock signal 125 MHz
set_property -dict { PACKAGE_PIN H16  IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];

## Pmod A
set_property -dict { PACKAGE_PIN Y18  IOSTANDARD LVCMOS33 } [get_ports { seg0[0] }];
set_property -dict { PACKAGE_PIN Y19  IOSTANDARD LVCMOS33 } [get_ports { seg0[1] }];
set_property -dict { PACKAGE_PIN Y16  IOSTANDARD LVCMOS33 } [get_ports { seg0[2] }];
set_property -dict { PACKAGE_PIN Y17  IOSTANDARD LVCMOS33 } [get_ports { seg0[3] }];
set_property -dict { PACKAGE_PIN U18  IOSTANDARD LVCMOS33 } [get_ports { seg0[4] }];
set_property -dict { PACKAGE_PIN U19  IOSTANDARD LVCMOS33 } [get_ports { seg0[5] }];
set_property -dict { PACKAGE_PIN W18  IOSTANDARD LVCMOS33 } [get_ports { seg0[6] }];
set_property -dict { PACKAGE_PIN W19  IOSTANDARD LVCMOS33 } [get_ports { cat0 }];

## Pmod B
set_property -dict { PACKAGE_PIN W14  IOSTANDARD LVCMOS33 } [get_ports { seg1[0] }];
set_property -dict { PACKAGE_PIN Y14  IOSTANDARD LVCMOS33 } [get_ports { seg1[1] }];
set_property -dict { PACKAGE_PIN T11  IOSTANDARD LVCMOS33 } [get_ports { seg1[2] }];
set_property -dict { PACKAGE_PIN T10  IOSTANDARD LVCMOS33 } [get_ports { seg1[3] }];
set_property -dict { PACKAGE_PIN V16  IOSTANDARD LVCMOS33 } [get_ports { seg1[4] }];
set_property -dict { PACKAGE_PIN W16  IOSTANDARD LVCMOS33 } [get_ports { seg1[5] }];
set_property -dict { PACKAGE_PIN V12  IOSTANDARD LVCMOS33 } [get_ports { seg1[6] }];
set_property -dict { PACKAGE_PIN W13  IOSTANDARD LVCMOS33 } [get_ports { cat1 }];

## Buttons
set_property -dict { PACKAGE_PIN D19  IOSTANDARD LVCMOS33 } [get_ports { BTN0 }];
set_property -dict { PACKAGE_PIN D20  IOSTANDARD LVCMOS33 } [get_ports { BTN1 }];
set_property -dict { PACKAGE_PIN L20  IOSTANDARD LVCMOS33 } [get_ports { BTN2 }];
```

```
set_property -dict { PACKAGE_PIN L19  IOSTANDARD LVCMOS33 } [get_ports { BTN3 }];

## Switches
set_property -dict { PACKAGE_PIN M20  IOSTANDARD LVCMOS33 } [get_ports { SW0 }];
set_property -dict { PACKAGE_PIN M19  IOSTANDARD LVCMOS33 } [get_ports { SW1 }];
```

References

[1] "PYNQ-Z1 Board Reference Manual", Tech. Rep., 2017. [Online]. Available: www.pynq.io..