Lab 1: Decimal Push Button Counter

Cory Brynds

co224015@ucf.edu

EEL5722C: FPGA Design

Section 0012

Due: 20 September 2023

Submitted: 20 September 2023

1.0 Objectives

Using the 7-segment LED displays on the BASYS 3 FPGA development board, implement a two-digit decimal counter. The tens and ones places on the counter are incremented by two push buttons on the board.

2.0 Equipment

- Xilinx Vivado Design Suite
- BASYS 3 FPGA Development Board

3.0 Experimental Explanation

To implement a decimal counter on the BASYS 3, the two rightmost digits of the seven-segment display were used as the tens and ones place. The two digits are incremented by two push buttons (PB1 and PB2) on the board. PB1 will increment the tens place of the display by one. PB2 will increment the ones place on the display by one. The maximum number able to be displayed on the counter is 99. Below is a breakdown of the different modules used to implement this project:

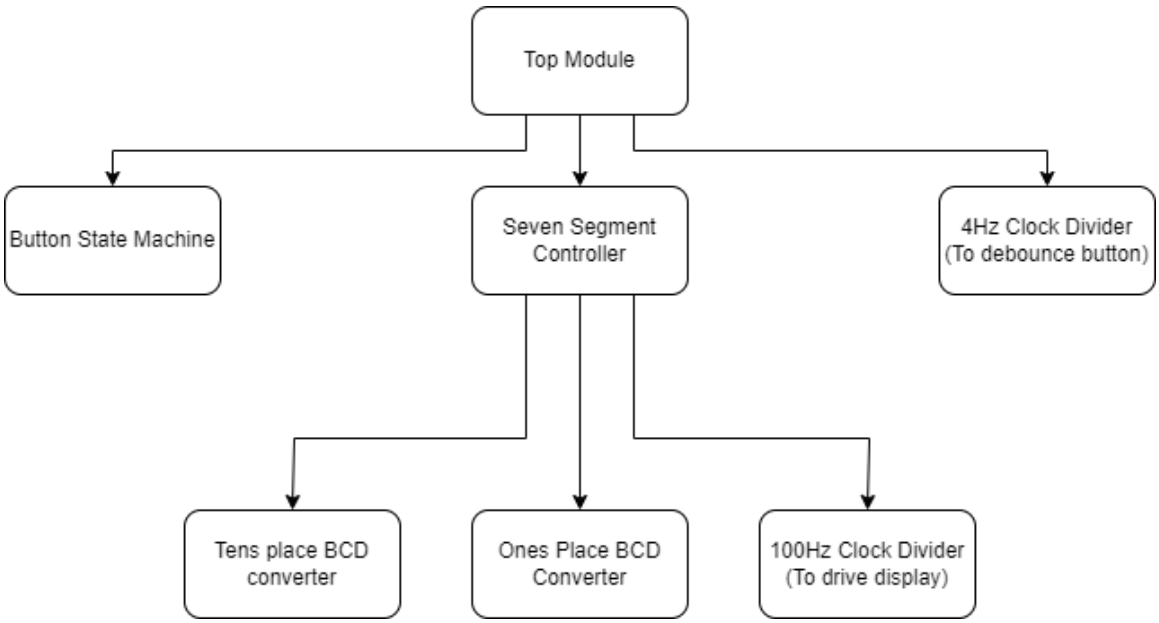

Figure 1: Block diagram of Verilog modules

- **Top Module**: instantiates the seven-segment controller, button state machine, and clock divider modules.
    - **Seven-segment controller**: instantiates the module to convert the tens and ones place from BCD to the common anode configuration for the display and the 100 Hz clock divider to drive the display. It also selects between the two multiplexed digits of the display.
        - **Seven-segment converter module**: takes the BCD value for the digit as an input and outputs the corresponding binary value to drive the common-anode seven-segment display.

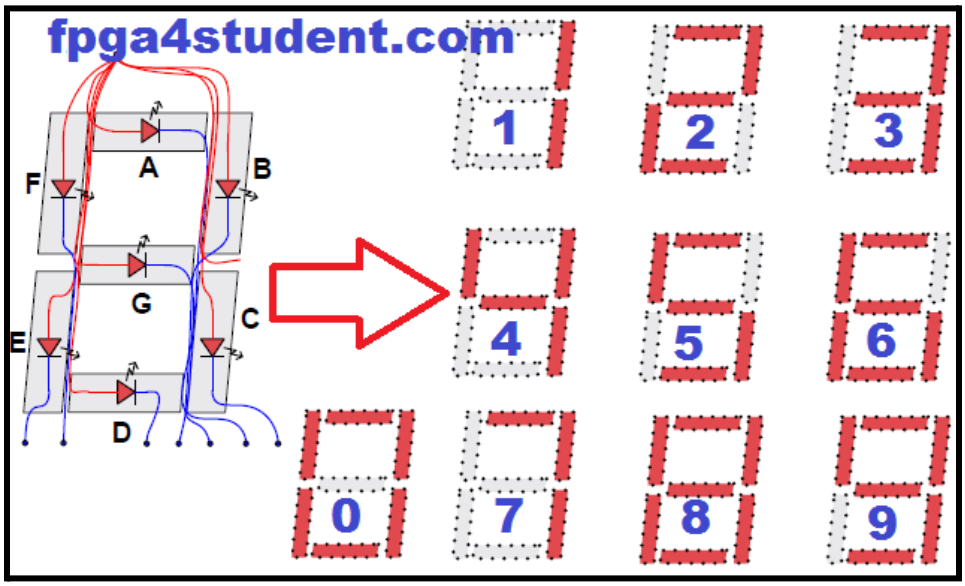

Figure 2: Seven-segment display diagram (source fpga4student.com)

        - **clk_100Hz module**: divides the 100 MHz clock signal into a 100 Hz clock signal.
    - **Button state machine module**: manages the logic of incrementing the display based on inputs from the buttons.
    - **clk_4Hz module**: divides the 100 MHz clock signal into a 4 Hz clock signal to debounce the push buttons.
    - **Button state machine**: handles the button input logic for the four separate cases of button presses.
        1. PB1 released, PB2 released - no increment

2. PB1 released, PB2 pressed - increment ones place
3. PB1 pressed, PB2 released - increment tens place
4. PB1 pressed, PB2 pressed - increment both ones and tens place

Additionally, as a bonus section of the experiment, the counter was modified to increment every second instead of at the press of a button. The modified button state machine and 1 Hz clock divider modules can be seen in the results section.

4.0 Results

4.1 Verilog code

```
module Top(
    input[1:0] Btn,
    input clk,
    output [6:0] seg,
    output [3:0] an
);
    wire [3:0] dec1, dec2;
    wire clk_4Hz;

    seven_seg_controller seg_ctrl (.clk(clk), .dec1(dec1), .dec2(dec2), .an(an), .seg(seg));
    BSM button_state_machine(.Btn(Btn), .clk_4Hz(clk_4Hz), .dec1(dec1), .dec2(dec2));
    clk_4Hz PBC_clk_divider(.clkIn(clk), .clkOut(clk_4Hz));


endmodule
```

Figure 3: Top module

```
module seven_seg_controller(
        input clk,
        input [3:0] dec1, dec2,
        output reg [6:0] seg,
        output [3:0] an
    );

    wire seg_select, clk_100Hz;
    wire [6:0] seg1, seg2;

    seven_seg_converter tens_converter(.dec(dec1), .seg(seg1));
    seven_seg_converter ones_converter(.dec(dec2), .seg(seg2));
    clk_100Hz seg_clk_divider(.clkIn(clk), .clkOut(clk_100Hz));
    seven_seg_display(.clk_100Hz(clk_100Hz), .an(an));

    assign seg_select = (an == 4'b1101) ? 1'b0: 1'b1;

    always @ (posedge clk_100Hz) begin
        case (seg_select)
            1'b0:
                seg <= seg2;
            1'b1:
                seg <= seg1;
        endcase
    end

endmodule
```

Figure 4: Seven-segment display controller

```
module seven_seg_converter(
    input [3:0] dec,
    output reg [6:0] seg
);

    always @ (*) begin
        case (dec)
        4'b0000: seg = 7'b1000000; // or 7'h3F
        4'b0001: seg = 7'b1111001; // or 7'h06
        4'b0010: seg = 7'b0100100; // or 7'h5B
        4'b0011: seg = 7'b0110000; // or 7'h4F
        4'b0100: seg = 7'b0011001; // or 7'h66
        4'b0101: seg = 7'b0010010; // or 7'h6D
        4'b0110: seg = 7'b0000010; // or 7'h7D
        4'b0111: seg = 7'b1111000; // or 7'h07
        4'b1000: seg = 7'b0000000; // or 7'h7F
        4'b1001: seg = 7'b0010000; // or 7'h4F
        default: seg = 7'b1111111;
        endcase
    end
endmodule
```

Figure 5: Converter from BCD to common anode

```
module clk_100Hz(
    input clkIn,
    output reg clkOut
    );

    reg[19:0] N; // Value to divide clock, calculated by fout = fin/(2*N)
    always @ (posedge clkIn) begin
        if (N == 500000) begin
            clkOut <= ~clkOut;
            N <= 20'b0;
        end
        else
            N= N + 1'b1;
    end
endmodule
```

Figure 6: 100 Hz clock divider

```
module BSM(
    input [1:0] Btn,
    input clk_4Hz,
    output reg [3:0] dec1, dec2
);

always @ (posedge clk_4Hz) begin
    case(Btn)
        2'b00: begin
            dec1 <= dec1;
            dec2 <= dec2;
        end
        2'b01: begin
            dec1 <= (dec1 + 1'b1) % 10;
            dec2 <= dec2;
        end
        2'b10: begin
            dec1 <= dec1;
            dec2 <= (dec2 + 1'b1) % 10;
        end
        2'b11: begin
            dec1 <= (dec1 + 1'b1) % 10;
            dec2 <= (dec2 + 1'b1) % 10;
        end
        default: begin
            dec1 <= dec1;
            dec2 <= dec2;
        end
    endcase
end

endmodule
```

Figure 7: Button state machine

```
module clk_4Hz(
    input clkIn,
    output reg clkOut
    );

    reg[24:0] N; // Value to divide clock, calculated by fout = fin/(2*N)
    always @ (posedge clkIn) begin
        if (N == 12500000) begin
            clkOut <= ~clkOut;
            N <= 25'b0;
        end
        else
            N= N + 1'b1;
    end
endmodule
```

Figure 8: 4 Hz clock divider

4.2 Verilog Code for Bonus Section

```
module one_second_counter(
    input clk_1Hz,
    output reg [3:0] dec1, dec2
);

always @ (posedge clk_1Hz) begin
    if (dec2 == 9)
        dec1 <= (dec1 + 1'b1) % 10;
    dec2 <= (dec2 + 1'b1) % 10;
end

endmodule
```

Figure 9: Counter to increment display every one second

```
module clk_1Hz(
    input clkIn,
    output reg clkOut
    );

    reg[26:0] N; // Value to divide clock, calculated by fout = fin/(2*N)
    always @ (posedge clkIn) begin
        if (N == 50000000) begin
            clkOut <= ~clkOut;
            N <= 26'b0;
        end
        else
            N= N + 1'b1;
    end
endmodule
```

Figure 10: 1 Hz clock divider

4.2 Pictures of Experimental Implementation

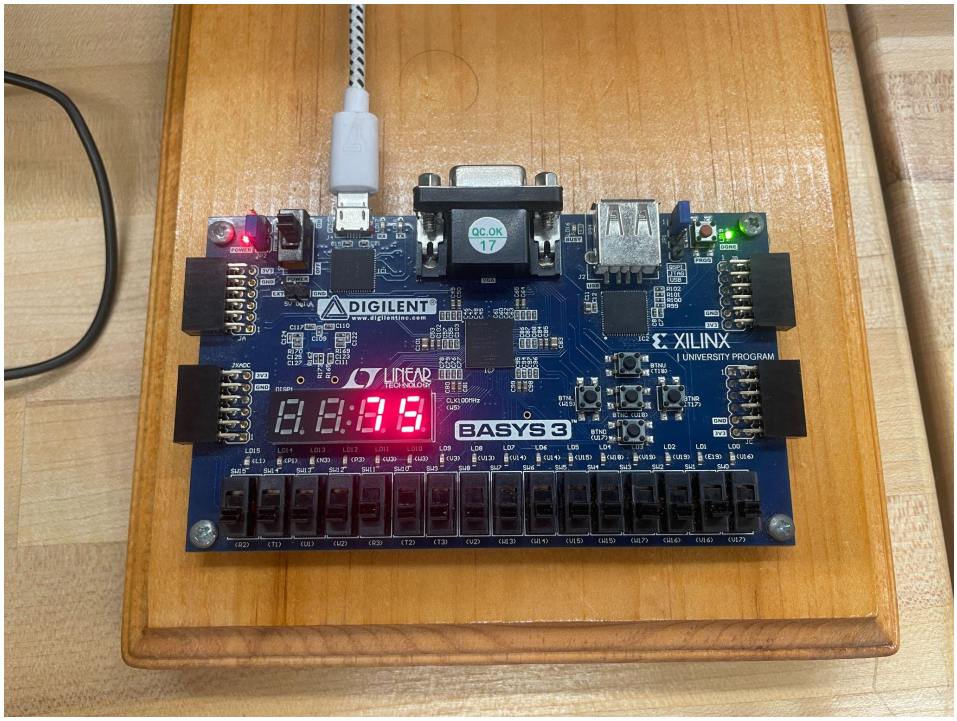

Figure 11: Counter implemented on BASYS 3

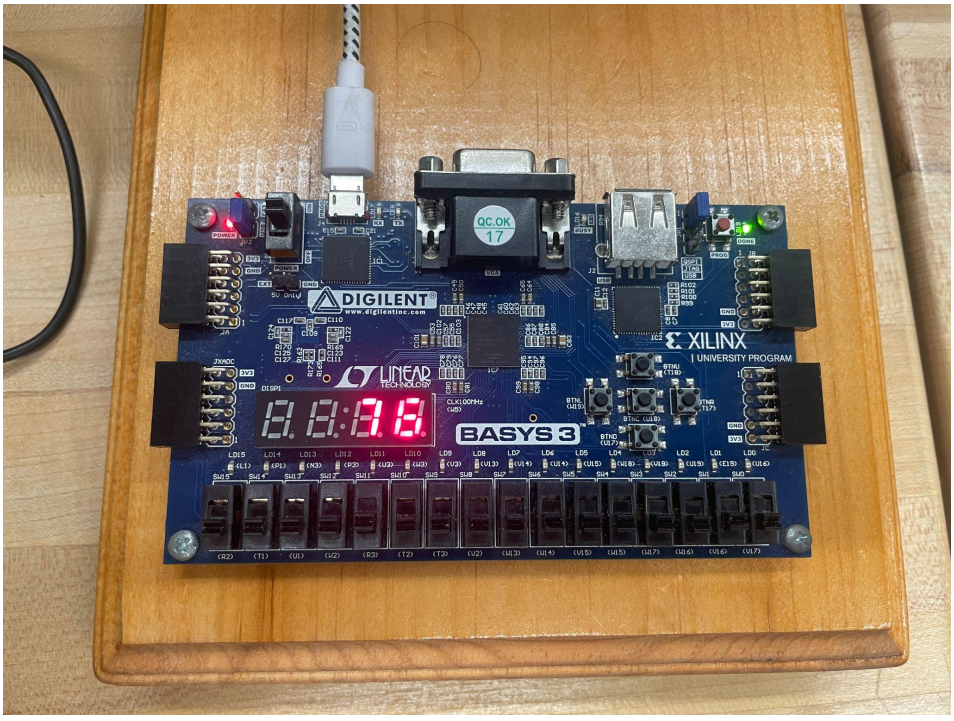

Figure 12: Counter implemented on BASYS 3

4.3 Explanation of Results

The implementation of the Verilog code for the decimal push button counter functioned as expected. Pressing PB2 incremented the ones place on the display, and pressing PB1 incremented the tens place of the display. When the digit “9” was reached on either of the two digits, the counter looped back around to zero.

Some difficulties encountered during this project included finding the proper package pin mappings for the seven-segment display. The documentation with the proper pin declarations was found online, solving this issue. This documentation will help to solve any future issues with pin declarations.

5.0 Conclusion

Overall, implementing a two-digit push button counter on the BASYS 3 served as a strong introduction to many fundamental Verilog concepts:

- Driving displays at a set frequency
- Dividing Verilog code into several submodules
- Using I/O devices to interface with combinational and sequential logic
- Reading documentation

It will serve as a strong basis for future experiments in this course, as well as the final project.