EGR 400 A
Advanced Digital
System Design Using FPGAs

Lab 4: Alternative Debouncing Circuit

Prepared for: Dr. Foist

Christopher Parisi

College of Engineering
California Baptist University

11/02/12

Introduction

The objective for this lab was to design, synthesize, and verify a debouncing circuit. In the textbook, Dr. Chu presents a debouncing circuit which uses a state machine. This lab instructs students on how to create an alternative to Dr. Chu's debouncing design. This design uses a state machine and a counter in order to solve the debouncing problem.

Procedure

Design and Simulation

Step 1: State Diagram and Block Diagram

A partially-completed state diagram and a block diagram were given and I was tasked with completing the state diagram. Using my understanding of how a debouncing circuit works, I filled in the missing parts of the state diagram. My state diagram and block diagram are shown below.

Figure 1.1: State Diagram

Figure 1.2: Block Diagram

Part 2: Determining N Value for the Counter

This debounce design utilizes a 30-40msec counter. In order to implement this counter, I had to determine the number of bits needed to simulate this amount of time using a counter. Given that the S3E board provides a 50 MHz clock, I performed the following calculations to determine the number of bits I would require for my counter.

$$f = 50MHz$$

$$T = \frac{1}{f} = \frac{1}{50MHz} = 20n\sec$$

$$Tx = 30m\sec$$

$$x = \frac{30m\sec}{20n\sec} = 1.5M$$

$$2^N = 30m\sec$$

In order to reach a count of at least 1.5 million, the value of N must be 21. This value actually allows for about 2 million so the counter is somewhere above 30msec.

Part 3: Designing and Testing the Finite State Machine

Using the state diagram in Figure 1.1 as a guide, I wrote the Verilog code for the state machine. This state machine has four states which represents the status of the push button. State “wait_1” waits for the first rising edge of the bush button signal. When it encounters the rising edge, the debounced signal is set high, the circuit moves to the second state “filt_1” and the timer is started. Once the timer finishes counting (roughly 30msec have passed), the circuit moves to the “wait_0” sate and waits for the push button signal to go low (off). Once the push button signal goes low, the circuit moves to the fourth state in which the timer is again enabled and counts for about 30msec. Once the counter is finished, the debounced signal is set low, and the circuit returns to the initial state. The code for this state machine can be found in Appendix A.

After writing the state machine, I tested my design using a Verilog Test Fixture. The simulation corresponded correctly with the state diagram and confirmed my design. The simulation waveform clearly shows the circuit move from each state with the correct input values. The code for this test bench as well as the simulation results can be in Appendix A.

Part 4: Designing the Counter

I then proceeded to design my counter using the previously calculated N value. I designed an up counter which outputs a signal when the highest value is reached. The code for this counter can be found in Appendix B.

Part 5: Designing and Testing the Top Design

After designing the necessary modules, I wrote a top level module named “dbounce_fast.v” and connected the modules together according to the block diagram in Figure 1.2. This code for this top design can be found in Appendix C.

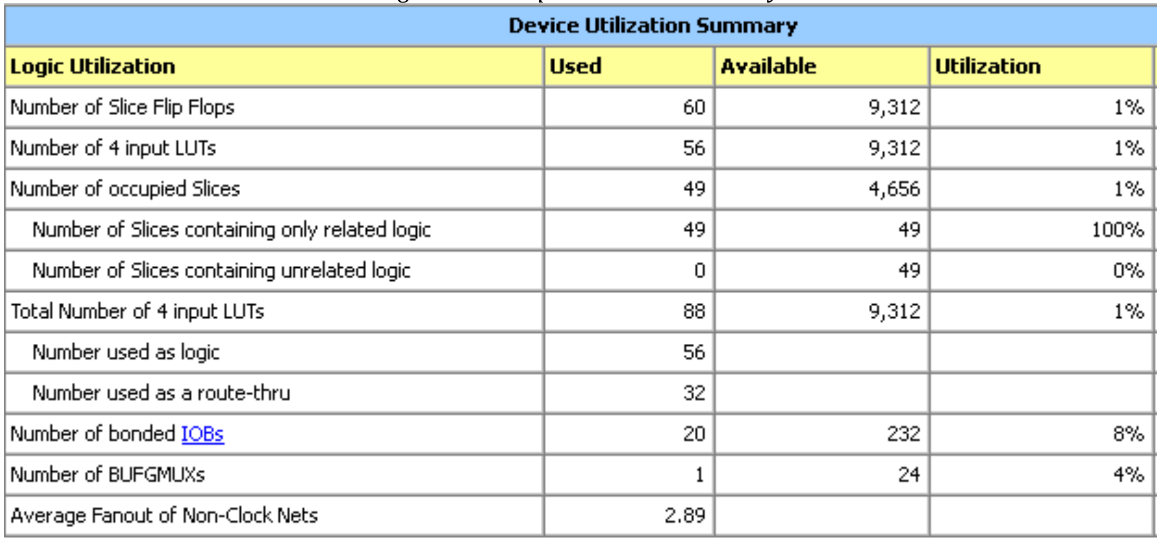
I then tested my design with a Verilog Test Fixture and the simulation results confirmed my design. I simulated a switch debounce by making the switch signal switch between high and low for a period of time. The simulation waveform shows how the debounce was ignored because of the time delay created by the counter. The results of my simulation and module code can be found in Appendix C.

FPGA Implementation

After completing the steps for designing a debounce circuit, I downloaded Dr. Chu’s debounce circuit project and tested it on the S3E board. When pressing the button, the debounced 7-seg-display counted regularly whereas the non-debounced 7-seg-display counted sporadically. This shows that his deboucning design was correct.

I then replaced his debouncing circuit with mine and tested it on the S3E board. My design functioned the same and the debounce error was fixed. The code for the top level FPGA implementation can be found in Appendix D. The User Constraints File used for this FPGA implementation can be found in Appendix E. The resources used in this design are shown in Figure 6.1.

Figure 6.1: Chip Resources Summary



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	60	9,312	1%
Number of 4 input LUTs	56	9,312	1%
Number of occupied Slices	49	4,656	1%
Number of Slices containing only related logic	49	49	100%
Number of Slices containing unrelated logic	0	49	0%
Total Number of 4 input LUTs	88	9,312	1%
Number used as logic	56		
Number used as a route-thru	32		
Number of bonded IOBs	20	232	8%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	2.89		

Conclusion

This lab went through the steps of designing, simulation, and implementing a debouncing circuit. A finite state machine and a counter were used to create this debouncing circuit. The state machine corresponded to the sections of the push button signal, and the counter was used to simulate a delay of about 30msec.

Everything in this lab worked correctly and a debounced counter was clearly observed in the FPGA implementation. I learned how to design and test an alternate debouncing circuit. I learned more about creating counters to simulate a time delay and how to use this module in a top level design.

Appendix A

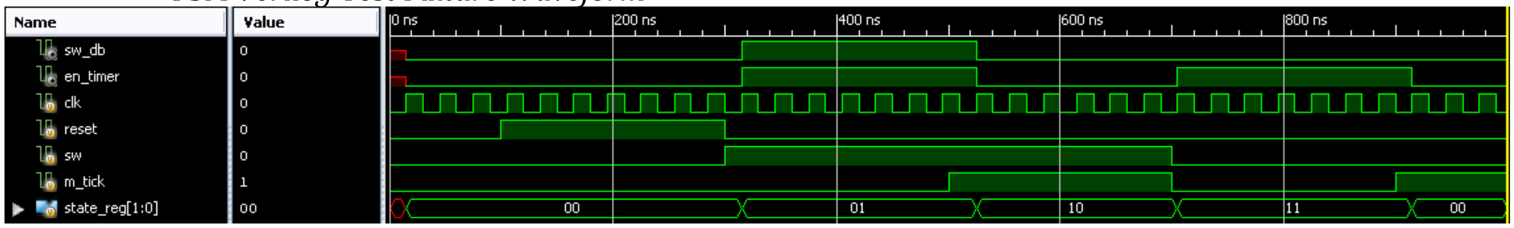
FSM Verilog Code

```
module fsm
        (
                input clk, reset, sw, m_tick,
                output sw_db, en_timer
        );

        //symbolic state declerations
        localparam [1:0] wait_1 = 2'b00,
                        filt_1 = 2'b01,
                        wait_0 = 2'b10,
                        filt_0 = 2'b11;

        //signal declerations
        reg [1:0] state_reg, state_next;

        //state register
        always @(posedge clk)
                if (reset)
                        state_reg <= wait_1;
                else
                        state_reg <= state_next;

        //next-state logic
        always @*
                case (state_reg)
                        wait_1: if(sw)
                                        state_next = filt_1;
                                else
                                        state_next = state_reg;
                        filt_1: if(m_tick)
                                        state_next = wait_0;
                                else
                                        state_next = state_reg;
                        wait_0: if(!sw)
                                        state_next = filt_0;
                                else
                                        state_next = state_reg;
                        filt_0: if(m_tick)
                                        state_next = wait_1;
                                else
                                        state_next = state_reg;
                        default: state_next = wait_1;
                endcase

        //Output
        assign sw_db = (state_reg == filt_1 || state_reg == wait_0);
        assign en_timer = (state_reg == filt_1 || state_reg == filt_0);

endmodule
```

FSM Verilog Test Fixture Code

```
`timescale 1ns / 1ps

module fsm_tb;

    // Inputs
    reg clk;
    reg reset;
    reg sw;
    reg m_tick;

    // Outputs
    wire sw_db;
    wire en_timer;

    // Instantiate the
    // Unit Under Test (UUT)
    fsm uut (
        .clk(clk),
        .reset(reset),
        .sw(sw),
        .m_tick(m_tick),
        .sw_db(sw_db),
        .en_timer(en_timer)
    );

    //40ns clock
    always #15 clk = !clk;

    initial begin
        // Initialize Inputs
        clk = 0;
        reset = 0;
        sw = 0;
        m_tick = 0;

        // Wait 100 ns
        #100;

        // Reset Circuit
        reset = 1;
        sw = 0;
        m_tick = 0;
        #200;

        // Activate State filt_1
        reset = 0;
        sw = 1;
        m_tick = 0;
        #200;

        // Activate State wait_0
        reset = 0;
        sw = 1;
        m_tick = 1;
        #200;

        //Activate State filt_0
        reset = 0;
        sw = 0;
        m_tick = 0;
        #200;

        //Loop back to State wait_1
        reset = 0;
        sw = 0;
        m_tick = 1;
        #200;
    end

endmodule
```

FSM Verilog Test Fixture Waveform



Appendix B

Counter Verilog Code

```
module counter
	#(parameter N=21)
	(
		input clk, reset, en,
		output m_tick
	);

	//Signal Decleration
	reg [N-1:0] count;

	//body
	always @(posedge clk, posedge reset)
		if (reset)
			count <= 0;
		else if (en)
			count <= count + 1;
		else
			count <= 0;

	assign m_tick = (count==2**N-1) ? 1'b1 : 1'b0;

endmodule
```

Appendix C

Top Design Verilog Code

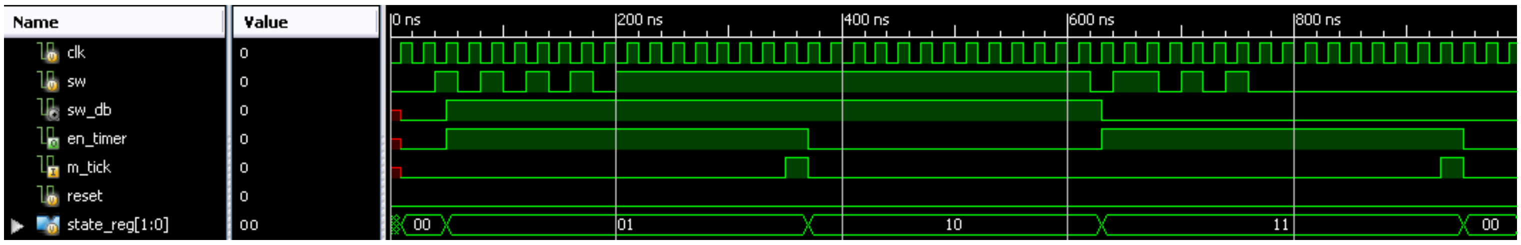
```
module dbounce_fast
		(
			input sw, clk, reset,
			output sw_db
  );


		wire en_timer, m_tick;

		//Instatiate Modules
		//Finite State Machine
		fsm u1
			(.sw(sw), .clk(clk), .reset(reset), .en_timer(en_timer), .m_tick(m_tick),
.sw_db(sw_db));

		//Counter
		counter u2
			(.en(en_timer), .clk(clk), .reset(reset), .m_tick(m_tick));

endmodule
```

Top Design Verilog Test Fixture Code

```
`timescale 1ns / 1ps

module dbounce_fast_tb;

		// Inputs
		reg sw;
		reg clk;
		reg reset;

		// Outputs
		wire sw_db;

		// Instantiate the Unit Under Test
		dbounce_fast uut (
			.sw(sw),
			.clk(clk),
			.reset(reset),
			.sw_db(sw_db)
		);

		//Clock
		always #10 clk = !clk;
```

```
initial begin
        // Initialize Inputs
        sw = 0;
        clk = 0;
        reset = 0;
        #40;

        //Bouncing Switch On
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;

        //Switch On
        sw = 1;
        reset = 0;
        #400;

        //Bouncing Switch Off
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #0;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;
        sw = 1;
        reset = 0;
        #20;
        sw = 0;
        reset = 0;
        #20;

        //Switch Off
        sw = 0;
        reset = 0;
        #500;
    end

endmodule
```

Top Design Verilog Test Fixture Waveform



Appendix D

FPGA Implementation Top Level Design Verilog Code

```
module debounce_fast_fpga
  (
           input wire clk, reset,
           input wire [1:0] btn,
           output wire an_lo,
           output wire an_hi,
           output wire [6:0] sseg_lo,
           output wire [6:0] sseg_hi
  );

  // signal declaration
  reg [7:0]  b_reg, d_reg;
  wire [7:0] b_next, d_next;
  reg  btn_reg, db_reg;
  wire db_level, db_tick, btn_tick, clr;

  // instantiate 7-seg LED display time-multiplexing module
  disp_hex_mux_s3e disp_unit_lo
           (.clk(clk), .reset(reset), .hex0(d_reg[7:4]), .hex1(d_reg[3:0]),
           .an(an_lo), .sseg(sseg_lo));

  disp_hex_mux_s3e disp_unit_hi
           (.clk(clk), .reset(reset), .hex0(b_reg[7:4]), .hex1(b_reg[3:0]),
            .an(an_hi), .sseg(sseg_hi));

  // instantiate debouncing circuit
           /*
           db_fsm db_unit //Dr. Chu's Debouncer
                    (.clk(clk), .reset(reset), .sw(btn[1]), .db(db_level));
           */
           dbounce_fast de_unit //My Debouncer
                    (.clk(clk), .reset(reset), .sw(btn[1]), .sw_db(db_level));

  // edge detection circuits
  always @(posedge clk)
           begin
                    btn_reg <= btn[1];
                    db_reg <= db_level;
           end
  assign btn_tick = ~btn_reg & btn[1];
  assign db_tick = ~db_reg & db_level;

  // two counters
  assign clr = btn[0];
  always @(posedge clk)
           begin
                    b_reg <= b_next;
                    d_reg <= d_next;
           end
  assign b_next = (clr)      ? 8'b0 :
```

```
                 (btn_tick) ? b_reg + 1 : b_reg;
   assign d_next = (clr)      ? 8'b0 :
                 (db_tick)  ? d_reg + 1 : d_reg;

endmodule
```

Appendix E

User Constraints File (UCF)

```
# s3e_debounce.ucf, 9.10.12, by Rod
# (adapted from my s3e_stopwatch.ucf)
#
#==========================================================
#    Pin assignment for Xilinx
#    Spartan-3E Starter board
#==========================================================

#==========================================================
# clock and reset
#==========================================================
# Period constraint for 50MHz operation, from Ken C.
#
NET "clk" PERIOD = 20.0ns HIGH 50%;
#
# soldered 50MHz Clock.
#
NET "clk" LOC = "C9" | IOSTANDARD = LVTTL;


#==========================================================
# buttons & switches
#==========================================================
# 4 pushbuttons
#NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "reset" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ; # "BTN_SOUTH"
NET "btn<0>" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ; # "BTN_NORTH"
NET "btn<1>" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ; # "BTN_WEST"
# 4 slide switches
#NET "sw<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP;
#NET "sw<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP;
#NET "sw<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP;
#NET "sw<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP;

# ------------------ Leds for debug -----------------
#NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "led<1>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; # led1,0 for debug

#NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "led<0>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#==========================================================
# 4-digit time-multiplexed 7-segment LED display
#==========================================================
NET "sseg_lo<6>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;  # segment a
NET "sseg_lo<5>" LOC = "C5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;  # segment b
NET "sseg_lo<4>" LOC = "B6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;  # segment c
NET "sseg_lo<3>" LOC = "F7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;  # segment d
NET "sseg_lo<2>" LOC = "C7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ; # segment e
NET "sseg_lo<1>" LOC = "E8" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ; # segment f
NET "sseg_lo<0>" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ; # segment g
NET "an_lo"      LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# 2-digits enable

NET "sseg_hi<6>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment a
NET "sseg_hi<5>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment b
NET "sseg_hi<4>" LOC = "B13" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment c
NET "sseg_hi<3>" LOC = "B14" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment d
NET "sseg_hi<2>" LOC = "D14" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment e
NET "sseg_hi<1>" LOC = "B16" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# segment f
NET "sseg_hi<0>" LOC = "C4"  | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ; # segment g
NET "an_hi"      LOC = "A11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;# 2-digits enable
```