UCSC Extension – Digital Design using FPGA

HW #4

Jae-Yang Park (jaeyangp@gmail.com)

Problem 3.7.6 One minute counter

Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: UCSC Extendion - Digital Design using FPGA
// Engineer: Jae-Yang Park
//
// Create Date:    18:45:09 02/09/2014
// Design Name:  Problem 3.7.6
// Module Name:    oneminute
//////////////////////////////////////////////////////////////////////////////////
module counter(clk, arst, en, q);
	parameter N = 7;

	input clk, arst, en;
	output [N-1:0] q;
	reg [N-1:0] q;

	always @(posedge clk or posedge arst)
		if (arst == 1'b1)
			q <= 0;
		else if (en)
			q <= q + 1;
endmodule

module mod_counter(clk, arst, q, done);
	parameter N = 7;
	parameter MAX = 127;

	input clk, arst;
	output [N-1:0] q;
	output done;
	reg [N-1:0] q;
	reg done;

	always @(posedge clk or posedge arst) begin
		if (arst == 1'b1) begin
			q <= 0;
			done <= 0;
		end
		else if (q == MAX) begin
			q <= 0;
			done <= 1;
		end
		else begin
			q <= q + 1;
```

```
                    done <= 0;
                end
            end
endmodule

module oneminute(CLK1, arst, seg, an, Led);
            parameter C = 33;           // counter,
            parameter N = 7;            // seven segment
            parameter W = 4;            // size of binary number, # of Led
            parameter CRYSTAL = 50;  // 50MHz
            parameter NUM_SEC = 60;
            parameter [C-1:0] STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC) - 1;

            input CLK1, arst;
            output [0:N-1] seg;
            output [3:0] an;
            output [W-1:0] Led;
            wire [C-1:0] clock;
            wire [W-1:0] zero_to_f_counter;
            wire one_min_clock;

            assign an = 4'b0000;

            mod_counter #(C, STOPAT) U1 (.clk(CLK1), .arst(arst), .q(clock), .done(one_min_clock));
            counter #(W) S (.clk(CLK1), .arst(arst), .en(one_min_clock), .q(zero_to_f_counter));
            hex2_7seg_lut D (zero_to_f_counter, seg);

            assign Led = zero_to_f_counter;
endmodule
```

In order to make one minute counter, follow parameters in the module oneminute are changed:

parameter C = 33

Counting value for one minute is 50 * 10^6 * 60 – 1 is 2999999999, and 33 bits counter is needed.

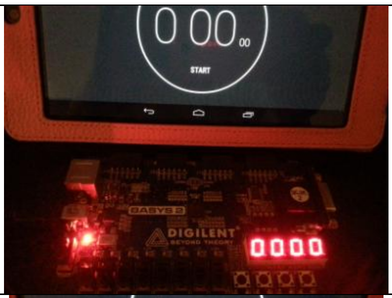
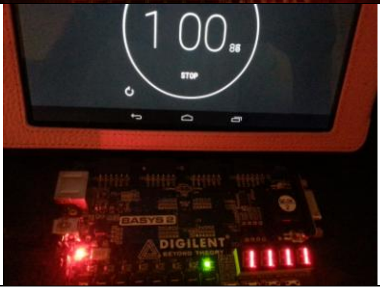
parameter NUM_SEC = 60

One minute is 60 seconds

parameter [C-1:0] STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC) – 1

For 60 sec counter, this STOPAT value is 2999999999, and it needs 33 bits.
Default integer value needs 32 bits. If the width is not mentioned, Xilinx tools can't place and route ths value, and display warning messages:

```
Command Line: bitgen -intstyle ise -f oneminute.ut oneminute.ncd
WARNING:PhysDesignRules:367 - The signal <CLK1_IBUF> is incomplete. The signal
   does not drive any load pins in the design.
```

Start: 0 Sec	
After 1 minute	

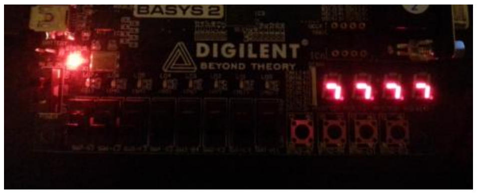
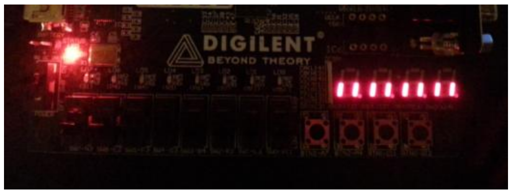
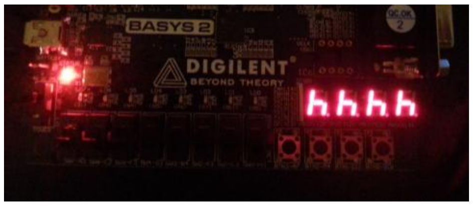
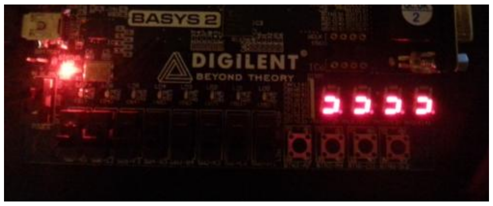
Problem 3.7.7 Display all possible patterns on the seven segment

Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: UCSC Extension - Digital Design using FPGA
// Engineer: Jae-Yang Park
//
// Create Date:    19:59:51 02/16/2014
// Design Name: Problem 3.7.7
// Module Name:    disp_all_pattern
//////////////////////////////////////////////////////////////////////////////////

module mod_counter(clk, arst, q, done);
	parameter N = 7;
	parameter MAX = 127;

	input clk, arst;
	output [N-1:0] q;
	output done;
	reg [N-1:0] q;
	reg done;

	always @(posedge clk or posedge arst) begin
		if (arst == 1'b1) begin
			q <= 0;
			done <= 0;
		end
		else if (q == MAX) begin
			q <= 0;
			done <= 1;
		end
		else begin
			q <= q + 1;
			done <= 0;
		end
	end
endmodule

module disp_all_pattern(CLK1, arst, seg, an);
	parameter N = 7; // seven segment
	parameter C = 27;		// counter,
	parameter CRYSTAL = 50;	// 50MHz
	parameter NUM_SEC = 1;
	parameter STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC) - 1;

	input CLK1, arst;
	output [0:N-1] seg;
	output [3:0] an;
	reg [0:N-1] seg;
	reg [3:0] an;
```

```
    mod_counter #(C, STOPAT) U1 (.clk(CLK1), .arst(arst), .done(one_sec_clock));

    always @ (posedge one_sec_clock or posedge arst) begin
        if (arst == 1'b1) begin
            seg <= 7'b111_1111;
            an <= 4'b1111;
        end
        else begin
            seg <= seg - 1;
            an <= 4'b0000;
        end
    end

endmodule
```

The number of all possible patterns = $2^7 - 1$ = **127** patterns (because when 7’b0000000, all LEDs on the segment are OFF)
Total time to display all possible patterns = **127 seconds** (every one second different pattern is displayed)

Problem 3.7.8 Heart beat pattern

Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: UCSC Extension-Digital Design using FPGA
// Engineer: Jae-Yang Park
//
// Create Date:    12:21:12 02/17/2014
// Design Name: Problem 3.7.8
// Module Name:    heart_beat
// Project Name:
//////////////////////////////////////////////////////////////////////////////////

`define USQUARE 7'b0011100
`define DSQUARE 7'b1100010

module mod_counter(clk, arst, q, done);
	parameter N = 7;
	parameter MAX = 127;

	input clk, arst;
	output [N-1:0] q;
	output done;
	reg [N-1:0] q;
	reg done;

	always @(posedge clk or posedge arst) begin
		if (arst == 1'b1) begin
			q <= 0;
			done <= 0;
		end
		else if (q == MAX) begin
			q <= 0;
			done <= 1;
		end
		else begin
			q <= q + 1;
			done <= 0;
		end
	end
endmodule

module heart_beat(CLK1, arst, seg, an);
	parameter N = 7; // seven segment
	parameter C = 27;		// counter,
	parameter CRYSTAL = 50;  // 50MHz
	parameter NUM_SEC = 1;
	parameter STOPAT = (CRYSTAL * 1_000_000 * NUM_SEC) - 1;

	input CLK1, arst;
	output [0:N-1] seg;
	output [3:0] an;
	reg [0:N-1] seg;
	reg [3:0] an, up_an, down_an;
	reg [2:0] shift_count;
	reg up_down;	// 0: up, 1: down square

	// one second counter
	mod_counter #(C, STOPAT) U1 (.clk(CLK1), .arst(arst), .done(one_sec_clock));

	always @ (posedge one_sec_clock or posedge arst) begin
		if (arst == 1'b1) begin
			seg <= `USQUARE;
			an <= 4'b1111;
			up_an <= 4'b0111;
			down_an <= 4'b1110;
			shift_count <= 3'b000;
			up_down <= 1'b0;		// 0: upper square, 1: down
		end
		else begin
			if (up_down == 1'b0) begin
				seg <= `USQUARE;
				an <= up_an;
				up_an <= {up_an[0], up_an[3:1]};	// rotate 1 bit right

				if (shift_count == 3'b011) begin
					up_down <= 1'b1;
					up_an <= 4'b0111;
				end
			end
			else if (up_down == 1'b1) begin
				seg <= `DSQUARE;
				an <= down_an;
				down_an <= {down_an[2:0], down_an[3]};	// rotate 1 bit left

				if (shift_count == 3'b111) begin
					up_down <= 1'b0;
					down_an <= 4'b1110;
				end
			end

			shift_count <= shift_count + 1;
		end
	end
endmodule
```