UCSC Extension – Digital Design using FPGA

**HW #4**

Jae-Yang Park ([jaeyangp@gmail.com](mailto: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 \* 106 \* 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 | 20140217_190411.jpg |
| After 1 minute | 20140217_190528.jpg |

**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** = 27 – 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)

|  |  |
| --- | --- |
| 20140217_190616.jpg | 20140217_190618.jpg |
|  |  |
| 20140217_190622.jpg | 20140217_190623.jpg |
|  |  |

**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

|  |  |  |  |
| --- | --- | --- | --- |
| 20140217_190723.jpg | 20140217_190724.jpg | 20140217_190725.jpg | 20140217_190726.jpg |
|  |  |  |  |
| 20140217_190731.jpg | 20140217_190729.jpg | 20140217_190728.jpg | 20140217_190727.jpg |