UCSC Extension

Digital Design with FPGA

**Homework #6**

Jae-Yang Park ([jaeyangp@gmail.com](mailto:jaeyangp@gmail.com))

Problem 4.5.2 Collatz Conjecture

**Code:** collatz.v and collatz\_top.v (other modules are from previous homework)

`timescale 1ns / 1ps

**module collatz(clk, arst, en, out);**

parameter N = 6; // number to calc for collatz conjucture

parameter C = 8; // size for count, 8 bit: ~ 255

parameter O = 14; // output size

input clk, arst, en;

output [O-1:0] out;

reg [O-1:0] number;

assign out = number;

always @(posedge clk or posedge arst) begin

if (arst == 1'b1) begin

number <= N;

end

else if (en) begin

if (number[0] == 1'b1) begin // odd number

number <= 3 \* number + 1;

end

else begin // even number

number <= number >> 1; // number/2

end

if (number == 1)

number <= number;

end

end

endmodule

**module collatz\_top(CLK1, arst, seg, an, Led);**

parameter N = 7;

parameter W = 4;

parameter S = 8;

parameter H = 14;

**parameter CL = 6; // number to test**

input CLK1, arst;

output [0:N-1] seg;

output [W-1:0] an;

output [S-1:0] Led;

wire [15:0] text;

wire one\_sec;

wire [S-1:0] num\_sel;

wire [H-1:0] bin\_text;

assign Led = (bin\_text == 14'b1) ? Led : num\_sel;

one\_second ONE (.CLK1(CLK1), .arst(arst), .one\_sec\_clock(one\_sec));

counter #(S) CNT (.clk(CLK1), .arst(arst), .en(one\_sec), .q(num\_sel));

collatz #(.N(CL)) CALC (.clk(CLK1), .arst(arst), .en(one\_sec), .out(bin\_text));

bin2bcd #(H, W) BCD (.in(bin\_text), .ones(text[3:0]), .tens(text[7:4]), .hundreds(text[11:8]), .thousands(text[15:12]));

display T (.text(text), .clk(CLK1), .arst(arst), .seg(seg), .an(an));

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 one\_second(CLK1, arst, one\_sec\_clock);**

parameter C = 26; // counter,

parameter CRYSTAL = 50; // 50MHz

parameter NUM\_SEC = 1;

parameter [C-1:0] STOPAT = (CRYSTAL \* 1\_000\_000 \* NUM\_SEC) - 1;

input CLK1, arst;

output one\_sec\_clock;

wire [C-1:0] clock;

mod\_counter #(C, STOPAT) ONE\_MC (.clk(CLK1), .arst(arst), .q(clock), .done(one\_sec\_clock));

endmodule

**module add3(in, out);**

parameter N = 4;

input [N-1:0] in;

output [N-1:0] out;

reg [N-1:0] out;

always @ (in) begin

case (in)

4'b0000: out <= 4'b0000;

4'b0001: out <= 4'b0001;

4'b0010: out <= 4'b0010;

4'b0011: out <= 4'b0011;

4'b0100: out <= 4'b0100;

4'b0101: out <= 4'b1000;

4'b0110: out <= 4'b1001;

4'b0111: out <= 4'b1010;

4'b1000: out <= 4'b1011;

4'b1001: out <= 4'b1100;

default: out <= 4'b0000;

endcase

end

endmodule

**module bin2bcd(in, ones, tens, hundreds, thousands);**

parameter N = 14; // input size, decimal 9999 is hex 270f (14 bits)

parameter C = 4; // chunk size

input [N-1:0] in;

output [C-1:0] ones, tens, hundreds, thousands;

wire [C-1:0] d1, d2, d3, d4, d5, d6, d7, d8;

wire [C-1:0] d9, d10, d11, d12, d13, d14, d15, d16;

wire [C-1:0] d17, d18, d19, d20, d21, d22, d23, d24, d25, d26;

wire [C-1:0] c1, c2, c3, c4, c5, c6, c7, c8;

wire [C-1:0] c9, c10, c11, c12, c13, c14, c15, c16;

wire [C-1:0] c17, c18, c19, c20, c21, c22, c23, c24, c25, c26;

assign d1 = {1'b0, in[13:11]};

add3 #(C) m1(d1, c1);

assign d2 = {c1[2:0], in[10]};

add3 #(C) m2(d2, c2);

assign d3 = {c2[2:0], in[9]};

add3 #(C) m3(d3, c3);

assign d4 = {c3[2:0], in[8]};

add3 #(C) m4(d4, c4);

assign d5 = {c4[2:0], in[7]};

add3 #(C) m5(d5, c5);

assign d6 = {c5[2:0], in[6]};

add3 #(C) m6(d6, c6);

assign d7 = {c6[2:0], in[5]};

add3 #(C) m7(d7, c7);

assign d8 = {c7[2:0], in[4]};

add3 #(C) m8(d8, c8);

assign d9 = {c8[2:0], in[3]};

add3 #(C) m9(d9, c9);

assign d10 = {c9[2:0], in[2]};

add3 #(C) m10(d10, c10);

assign d11 = {c10[2:0], in[1]};

add3 #(C) m11(d11, c11);

assign d12 = {1'b0, c1[3], c2[3], c3[3]};

add3 #(C) m12(d12, c12);

assign d13 = {c12[2:0], c4[3]};

add3 #(C) m13(d13, c13);

assign d14 = {c13[2:0], c5[3]};

add3 #(C) m14(d14, c14);

assign d15 = {c14[2:0], c6[3]};

add3 #(C) m15(d15, c15);

assign d16 = {c15[2:0], c7[3]};

add3 #(C) m16(d16, c16);

assign d17 = {c16[2:0], c8[3]};

add3 #(C) m17(d17, c17);

assign d18 = {c17[2:0], c9[3]};

add3 #(C) m18(d18, c18);

assign d19 = {c18[2:0], c10[3]};

add3 #(C) m19(d19, c19);

assign d20 = {1'b0, c12[3], c13[3], c14[3]};

add3 #(C) m20(d20, c20);

assign d21 = {c20[2:0], c15[3]};

add3 #(C) m21(d21, c21);

assign d22 = {c21[2:0], c16[3]};

add3 #(C) m22(d22, c22);

assign d23 = {c22[2:0], c17[3]};

add3 #(C) m23(d23, c23);

assign d24 = {c23[2:0], c18[3]};

add3 #(C) m24(d24, c24);

assign d25 = {1'b0, c20[3], c21[3], c22[3]};

add3 #(C) m25(d25, c25);

assign d26 = {c25[2:0], c23[3]};

add3 #(C) m26(d26, c26);

assign ones = {c11[2:0], in[0]};

assign tens = {c19[2:0], c11[3]};

assign hundreds = {c24[2:0], c19[3]};

assign thousands = {c26[2:0], c24[3]};

endmodule

**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 hex2\_7seg\_lut(in, out);**

input [3:0] in;

output [6:0] out;

LUT4 #(16'h2812) CA (out[6], in[0], in[1], in[2], in[3]); // a

LUT4 #(16'hd860) CB (out[5], in[0], in[1], in[2], in[3]); // b

LUT4 #(16'hd004) CC (out[4], in[0], in[1], in[2], in[3]); // c

LUT4 #(16'h8492) CD (out[3], in[0], in[1], in[2], in[3]); // d

LUT4 #(16'h02ba) CE (out[2], in[0], in[1], in[2], in[3]); // e

LUT4 #(16'h208e) CF (out[1], in[0], in[1], in[2], in[3]); // f

LUT4 #(16'h1083) CG (out[0], in[0], in[1], in[2], in[3]); // g

endmodule

**module decoder(text, s, y, val) ;**

input [15:0] text;

input [1:0] s ;

output reg [3:0] y ;

output reg [3:0] val ;

always @(\*) begin

case (s)

0: begin

y <= 4'b1110 ;

val <= text[3:0] ;

end

1:begin

y <= 4'b1101 ;

val <= text[7:4] ;

end

2:begin

y <= 4'b1011 ;

val <= text[11:8] ;

end

3: begin

y <= 4'b0111 ;

val <= text[15:12] ;

end

default:begin

y <= 4'bx ;

val <= 4'bx ;

end

endcase

end

endmodule

**module display(text, clk, arst, seg, an);**

parameter C = 26; // counter,

parameter N = 7; // seven segment

parameter W = 4;

parameter S = 2;

parameter ANODE\_FREQ = 19;

input [15:0] text;

input clk, arst;

output [0:N-1] seg;

output [W-1:0] an;

wire [C-1:0] q;

wire [S-1:0] sel;

wire [W-1:0] zero\_to\_f\_counter;

assign sel[1] = q[ANODE\_FREQ];

assign sel[0] = q[ANODE\_FREQ - 1];

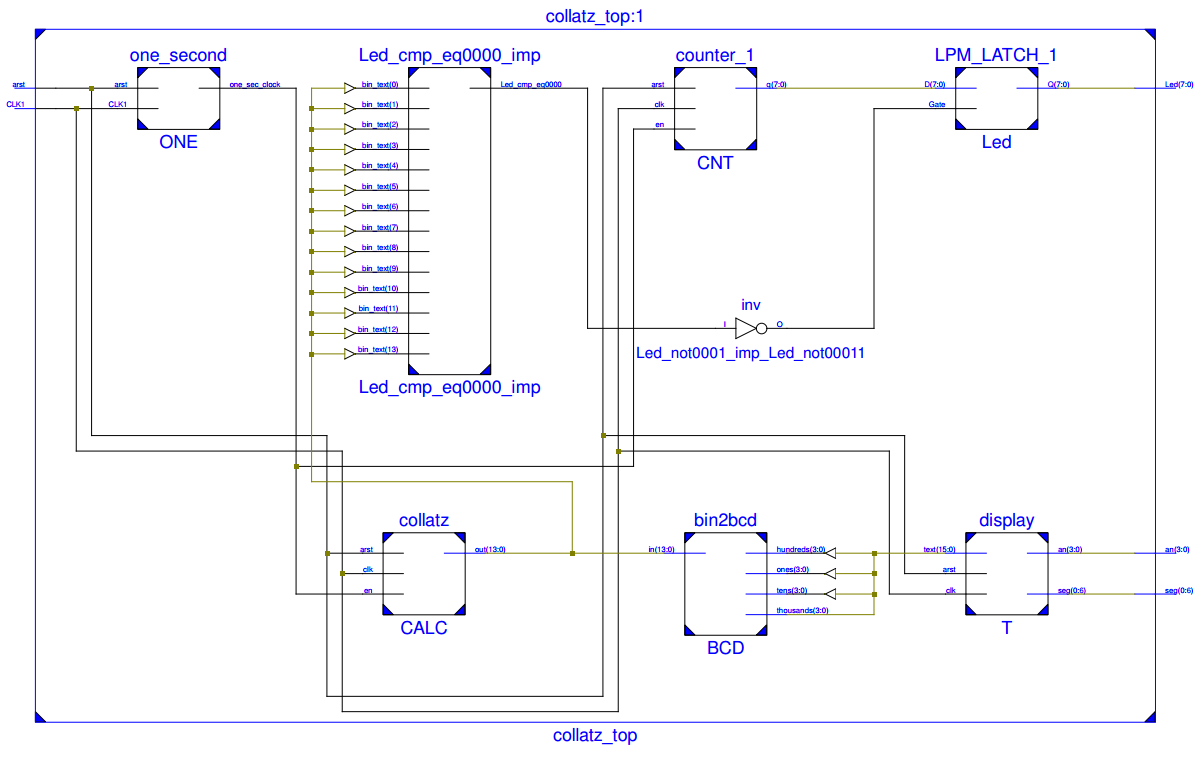
counter #C DISP\_C (.clk(clk), .arst(arst), .en(1), .q(q));

decoder DISP\_D (.text(text), .s(sel), .y(an), .val(zero\_to\_f\_counter));

hex2\_7seg\_lut DISP\_H (.in(zero\_to\_f\_counter), .out(seg));

endmodule

**Schematics**



|  |  |  |
| --- | --- | --- |
| At 0 sec, n = 6 | 1 sec | 2 sec |
| 20140303_215019.jpg | 20140303_215020.jpg | 20140303_215021.jpg |
|  |  |  |
| 3 sec | 4 sec | 5 sec |
| 20140303_215021(0).jpg | 20140303_215023.jpg | 20140303_215024.jpg |
|  |  |  |
| 6 sec | 7 sec | 8 sec |
| 20140303_215025.jpg | 20140303_215026.jpg | 20140303_215026(0).jpg |
|  |  |  |