***LAB 03: Counters***

**Task 1: 4-Bit Counter using T-Flops**

[... converted to 16-bit]

File: lab04part01.v, B\_memoryFlops.v, B\_decoders.v, B\_counters.v

FPGA: DONE.

Code:

MAIN

module lab04part01(

output [15:0]countVal,

output [6:0]HEX3, HEX2, HEX1, HEX0,

output [2:0]LEDG,

input [2:0]SW

);

wire noUse;

counter16x c16(countVal[15:0], noUse, SW[0], SW[1], SW[2]);

hexDecoder7Seg d3(HEX3[6:0], countVal[15:12]);

hexDecoder7Seg d2(HEX2[6:0], countVal[11:8]);

hexDecoder7Seg d1(HEX1[6:0], countVal[7:4]);

hexDecoder7Seg d0(HEX0[6:0], countVal[3:0]);

assign LEDG[2:0] = SW[2:0];

endmodule

TESTBENCH

module tbl4p1;

reg [2:0]sw;

wire [15:0]lr;

wire [6:0]h3, h2, h1, h0;

integer i;

lab04part02 testl4p2(lr, h3, h2, h1, h0, sw);

initial begin

sw[1]=1'b0;

sw[0]=1'b1;

for(i=0; i<100; i=i+1) begin

#100;

if(i==0) //clearing the value

sw[2] = 0;

if(i==2)

sw[2] = 1;

sw[1] = ~sw[1]; //ticking the clock

end //for

end

endmodule

HEX DECODER 7 SEGMENT

module hexDecoder7Seg(

output [6:0]SevenSeg,

input [3:0]X

);

assign SevenSeg[6]=((~X[3]&~X[2]&~X[1])|(~X[3]&X[2]&X[1]&X[0])|(X[3]&X[2]&~X[1]&~X[0]));

assign SevenSeg[5]=((~X[3]&~X[2]&X[0])|(~X[3]&~X[2]&X[1])|(~X[3]&X[1]&X[0])|(X[3]&X[2]&~X[1]&X[0]));

assign SevenSeg[4]=((~X[3]&X[0])|(~X[2]&~X[1]&X[0])|(~X[3]&X[2]&~X[1]));

assign SevenSeg[3]=((X[3]&~X[2]&X[1]&~X[0])|(~X[3]&X[2]&~X[1]&~X[0])|(X[2]&X[1]&X[0])|(~X[3]&~X[2]&~X[1]&X[0]));

assign SevenSeg[2]=((X[3]&X[2]&~X[0])|(X[3]&X[2]&X[1])|(~X[3]&~X[2]&X[1]&~X[0]));

assign SevenSeg[1]=((X[2]&X[1]&~X[0])|(X[3]&X[1]&X[0])|(X[3]&X[2]&~X[0])|(~X[3]&X[2]&~X[1]&X[0]));

assign SevenSeg[0]=((~X[3]&~X[2]&~X[1]&X[0])|(~X[3]&X[2]&~X[1]&~X[0])|(X[3]&~X[2]&X[1]&X[0])|(X[3]&X[2]&~X[1]&X[0]));

endmodule

16-BIT COUNTER

/\* 16-BIT COUNTER USING T-FLIP FLOPS \*/

module counter16x(

output [15:0]countVal, ANDr,

input ANDl, CLK, RESN

);

wire [3:0]carryOn;

counter4x countA(countVal[ 3: 0], carryOn[3], ANDl, CLK, RESN);

counter4x countB(countVal[ 7: 4], carryOn[2], carryOn[3], CLK, RESN);

counter4x countC(countVal[11: 8], carryOn[1], carryOn[2], CLK, RESN);

counter4x countD(countVal[15:12], carryOn[0], carryOn[1], CLK, RESN);

assign ANDr = carryOn[0];

endmodule

4-BIT COUNTER

/\* 4-BIT COUNTER USING T-FLIP FLOPS \*/

module counter4x(

output [3:0]countVal, ANDr,

input ANDl, CLK, RESN

);

/\*genvar i;

generate

//for(i=1; i<17; i=i+1) begin: m

for(i=1; i<5; i=i+1) begin: m

lab04part01\_TFlipAnd (cBout[i], cBout[i-1], SW[1], SW[2]);

end //for

endgenerate\*/

wire [3:0]andOut;

lab04part01\_TFlipAnd a1(andOut[0], countVal[0], ANDl, CLK, RESN);

lab04part01\_TFlipAnd a2(andOut[1], countVal[1], andOut[0], CLK, RESN);

lab04part01\_TFlipAnd a3(andOut[2], countVal[2], andOut[1], CLK, RESN);

lab04part01\_TFlipAnd a4(andOut[3], countVal[3], andOut[2], CLK, RESN);

assign ANDr = andOut[3];

endmodule

T FLIP FLOP

module mem\_Tflippos(

output reg Q,

input T, clk, reset

);

wire d;

xor(d, Q, T);

always@(posedge clk)

begin

if(~reset)

Q <= 1'b0;

else

Q <= d;

end

endmodule

T FLIP FLOP W/ AND OUTPUT

// T FLIP FLOP & AND GATE

//[out+and][out][X][CLK][RESN]

module lab04part01\_TFlipAnd(

output Y, ya,

input X, CLK, RESN

);

mem\_Tflippos calll4p1(ya, X, CLK, RESN);

assign Y = (ya & X);

endmodule

**Task 2: 16-Bit Counter using +1 (INBUILT)**

File: lab04part02.v, B\_decoders.v, B\_counters.v

FPGA: X.

Code:

MAIN

module lab04part02(

output [15:0]count,

output [6:0]HEX3, HEX2, HEX1, HEX0,

input [2:0]SW

);

counter16x\_add altcounter(count[15:0], SW[0], SW[1], SW[2]);

hexDecoder7Seg h3(HEX3[6:0], count[15:12]);

hexDecoder7Seg h2(HEX2[6:0], count[11:8]);

hexDecoder7Seg h1(HEX1[6:0], count[7:4]);

hexDecoder7Seg h0(HEX0[6:0], count[3:0]);

endmodule

TESTBENCH

module tbl4p2;

reg [2:0]sw;

wire [15:0]ct;

wire [6:0]h3, h2, h1, h0;

integer i;

lab04part02 testl4p2(ct, h3, h2, h1, h0, sw);

initial begin

sw[1]=1'b0;

sw[0]=1'b1;

for(i=0; i<100; i=i+1) begin

#100;

if(i==0) //clearing the value

sw[2] = 0;

if(i==2)

sw[2] = 1;

sw[1] = ~sw[1]; //ticking the clock

end //for

end

endmodule

ALT\_COUNTER

/\* 16-BIT COUNTER USING +1 \*/

module counter16x\_add(

output reg [15:0]countVal,

input EN, CLK, RESN

);

always@(posedge CLK) begin

if(EN)

countVal <= countVal + 1;

if(~RESN)

countVal <= 16'h0000;

else if(~EN & RESN)

countVal <= countVal;

end //always

endmodule

**Task 3: 16-Bit Counter using LPM (INBUILT LIBRARY)**

[... converted to 16-bit]

File: lab04part03.v

FPGA: X.

Code:

MAIN

module lab04part03(

output [15:0]LEDR,

output [6:0]HEX3, HEX2, HEX1, HEX0,

input [1:0]SW,

input [1:1]KEY

);

//CNT\_EN = SW[0], S\_CLR = SW[1]

count1 lpmcount(KEY[1], SW[0], SW[1], LEDR[15:0]);

hexDecoder7Seg h3(HEX3[6:0], LEDR[15:12]);

hexDecoder7Seg h2(HEX2[6:0], LEDR[11:8]);

hexDecoder7Seg h1(HEX1[6:0], LEDR[7:4]);

hexDecoder7Seg h0(HEX0[6:0], LEDR[3:0]);

endmodule

TESTBENCH

module tbl4p3;

reg [2:0]sw;

wire [15:0]lr;

wire [6:0]h3, h2, h1, h0;

integer i;

lab04part03 testl4p3(lr, h3, h2, h1, h0, {sw[2], sw[0]}, sw[1]);

initial begin

sw[1]=1'b0;

sw[0]=1'b1;

for(i=0; i<100; i=i+1) begin

#100;

if(i==0) //clearing the value

sw[2] = 1;

if(i==2)

sw[2] = 0;

sw[1] = ~sw[1]; //ticking the clock

end //for

end

endmodule

COUNT1

module count1 (

clock,

cnt\_en,

sclr,

q);

input clock;

input cnt\_en;

input sclr;

output [15:0] q;

endmodule

**Task 4: Flashing after 1s Delay**

[... added LEDs to display 1s-delay-counter value]

File: lab04part04.v, B\_timers.v

FPGA: DONE.

Code:

MAIN

module lab04part04(

output [9:0]LEDR,

output [7:0]LEDG,

output [6:0]HEX0,

input CLOCK\_50,

input [0:0]SW

);

wire noUse;

wire [3:0]numDisp;

wire ifInc;

wire [25:0]countVal;

clock\_1s getDelay(countVal[25:0], ifInc, CLOCK\_50);

counter4x c4b(numDisp, noUse, 1'b1, ifInc, 1'b1);

hexDecoder7Seg h0(HEX0, numDisp);

assign LEDR[9:0] = countVal[25:16];

assign LEDG[7:0] = countVal[15:8];

endmodule

1s DELAY

// generate a tick after 1s using clock of 50MHz

module clock\_1s(

output reg [25:0]countVal,

output reg flagEnable,

input CLKcount

);

always@(posedge CLKcount) begin

if(countVal == 26'd50000000) begin

flagEnable = 1'b1;

countVal <= 26'd0000000;

end //if

else if(countVal < 26'd50000000) begin

flagEnable = 1'b0;

countVal <= countVal + 1'b1;

end //else if

else begin

countVal <= 26'd5000000;

end

end //always

endmodule