**Prelab for Lab5**

Xiaoyu Zhou, 1004081147

**Part I**

* 1. The schematic

图片包含 文字, 白板

描述已自动生成

3.Write the Verilog

module mux(SW, KEY, HEX0, HEX1);

wire [7:0] Q;

input [1:0] SW;

input [2:0] KEY;

output [6:0] HEX0;

output [6:0] HEX1;

counter a(

.enable(SW[1]),

.clock(KEY[0]),

.clearb(SW[0]),

.Q(Q[7:0])

);

hex u0(

.in2(Q[3]),

.in3(Q[2]),

.in4(Q[1]),

.in5(Q[0]),

.o2(HEX0[0]),

.o3(HEX0[1]),

.o4(HEX0[2]),

.o5(HEX0[3]),

.o6(HEX0[4]),

.o7(HEX0[5]),

.o8(HEX0[6])

);

hex u1(

.in2(Q[7]),

.in3(Q[6]),

.in4(Q[5]),

.in5(Q[4]),

.o2(HEX1[0]),

.o3(HEX1[1]),

.o4(HEX1[2]),

.o5(HEX1[3]),

.o6(HEX1[4]),

.o7(HEX1[5]),

.o8(HEX1[6])

);

endmodule

module counter(enable, clock, clearb, Q);

input enable;

input clock;

input clearb;

output [7:0] Q;

wire in1, in2, in3, in4, in5, in6, in7, in8;

T\_flipflop a(

.T(enable),

.clock(clock),

.c(clearb),

.Q(Q[7])

);

assign in1 = enable & Q[7];

T\_flipflop b(

.T(in1),

.clock(clock),

.c(clearb),

.Q(Q[6])

);

assign in2 = in1 & Q[6];

T\_flipflop c(

.T(in2),

.clock(clock),

.c(clearb),

.Q(Q[5])

);

assign in3 = in2 & Q[5];

T\_flipflop d(

.T(in3),

.clock(clock),

.c(clearb),

.Q(Q[4])

);

assign in4 = in3 & Q[4];

T\_flipflop e(

.T(in4),

.clock(clock),

.c(clearb),

.Q(Q[3])

);

assign in5 = in4 & Q[3];

T\_flipflop f(

.T(in5),

.clock(clock),

.c(clearb),

.Q(Q[2])

);

assign in6 = in5 & Q[2];

T\_flipflop g(

.T(in6),

.clock(clock),

.c(clearb),

.Q(Q[1])

);

assign in7 = in6 & Q[1];

T\_flipflop h(

.T(in7),

.clock(clock),

.c(clearb),

.Q(Q[0])

);

endmodule

module T\_flipflop(T, clock, c, Q);

parameter N = 1;

input T;

input [N-1:0] clock;

input [N-1:0] c;

output Q;

wire connect;

assign connect = T ^ Q;

D\_flipflop dd(

.d(connect),

.clock(clock),

.reset\_n(c),

.q(Q)

);

endmodule

module D\_flipflop(d, clock, reset\_n, q);

parameter N = 1;

input d;

input [N-1:0] clock;

input [N-1:0] reset\_n;

output q;

reg q;

always @(posedge clock, negedge reset\_n)

begin

if (reset\_n == 1'b0)

q <= 0;

else

q <= d;

end

endmodule

module hex(in2, in3, in4, in5, o2, o3, o4, o5, o6, o7, o8);

input in2;

input in3;

input in4;

input in5;

output o2;

output o3;

output o4;

output o5;

output o6;

output o7;

output o8;

hex0 u1(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out1(o2)

);

hex1 u2(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out2(o3)

);

hex2 u3(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out3(o4)

);

hex3 u4(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out4(o5)

);

hex4 u5(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out5(o6)

);

hex5 u6(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out6(o7)

);

hex6 u7(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out7(o8)

);

endmodule

module hex0(c3, c2, c1, c0, out1);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out1; //output

assign out1 = ~c3 & c2 & ~c1 & ~c0 | c3 & c2 & ~c1 & c0 | c3 & ~c2 & c1 & c0 | ~c3 & ~c2 & ~c1 & c0;

endmodule

module hex1(c3, c2, c1, c0, out2);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out2; //output

assign out2 = c2 & c1 & ~c0 | c3 & c1 & c0 | ~c3 & c2 & ~c1 & c0 | c3 & c2 & ~c0;

endmodule

module hex2(c3, c2, c1, c0, out3);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out3; //output

assign out3 = ~c3 & ~c2 & c1 & ~c0 | c3 & c2 & c1 | c3 & c2 & ~c0;

endmodule

module hex3(c3, c2, c1, c0, out4);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out4; //output

assign out4 = ~c3 & c2 & ~c1 & ~c0 | ~c2 & ~c1 & c0 | c2 & c1 & c0 | c3 & ~c2 & c1 & ~c0;

endmodule

module hex4(c3, c2, c1, c0, out5);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out5; //output

assign out5 = ~c3 & c2 & ~c1 | ~c2 & ~c1 & c0 | ~c3 & c0;

endmodule

module hex5(c3, c2, c1, c0, out6);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out6; //output

assign out6 = ~c3 & ~c2 & c0 | ~c3 & ~c2 & c1 | ~c3 & c1 & c0 | c3 & c2 & ~c1 & c0;

endmodule

module hex6(c3, c2, c1, c0, out7);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

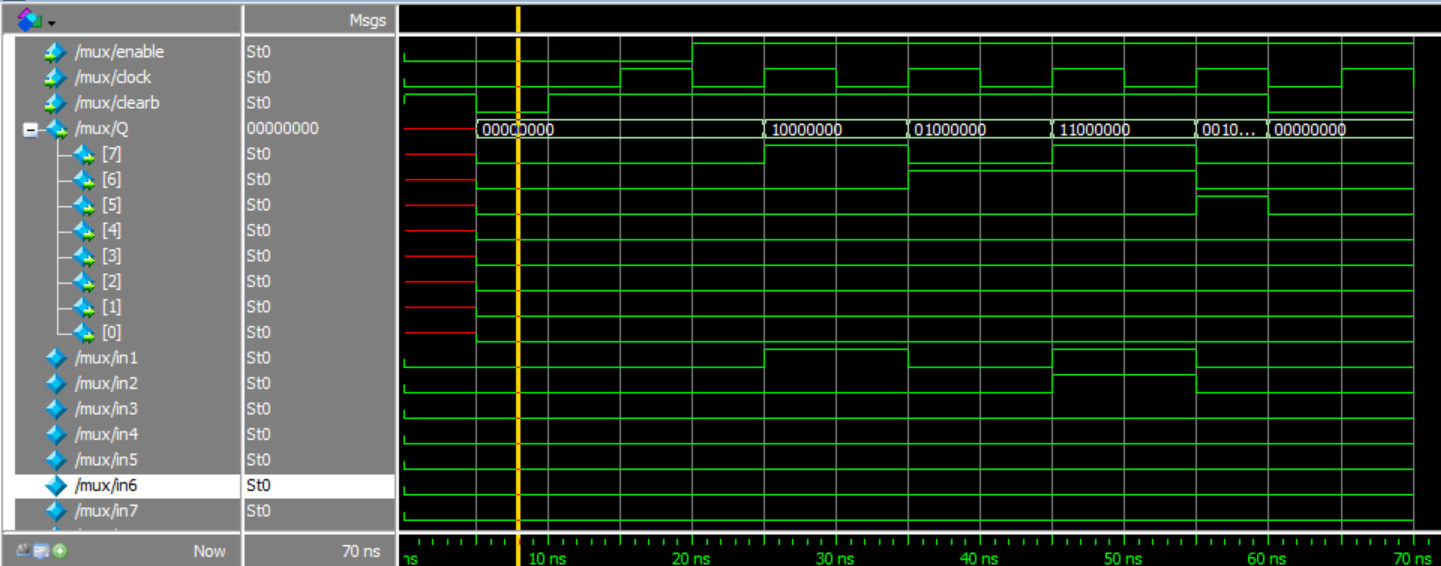
input c0;

output out7; //output

assign out7 = ~c3 & ~c2 & ~c1 | c3 & c2 & ~c1 & ~c0 | ~c3 & c2 & c1 & c0;

endmodule

4. Simulate first.



5. Simulate second time, with connect to the switches.

图片包含 监视器, 墙壁

描述已自动生成

6.(1) The percentage of FPGA logic resources are used is less than 1%.

图片包含 屏幕截图

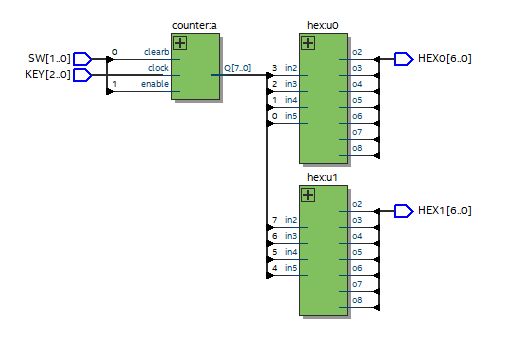
描述已自动生成

(2) The maximum clock frequency, Fmax is 621.89 MHz..

图片包含 屏幕截图

描述已自动生成

7. The schematic of RTL viewer looks much more complex. And it puts all T-flipflops together instead of putting them in order.

图片包含 文字, 地图

描述已自动生成

**Part II**

Q1: In this implementation, q is declared as a 4-bit value, which makes this a 4-bit counter. The check for the maximum value is not necessary in the example above. Why?

A: Since the range of this implementation is from 0 to F, which is also the range for the 4-bits.

Q2: If you wanted this 4-bit counter to count from 0-9, what would you change?

A: Change the if statement ‘q == 4’b1111’ into ‘q == 4’b1001’.

Q3: Full speed means that the display flashes at the rate of the 50 MHz clock provided on the DE1-SoC board. At this speed, what do you expect to see on the display?

A: The period will be 20ns. So, the flashes will be the same as the clock changes.

Q4: How large a counter is required to count 50 million clock cycles? A: 50 million.

Q5: How many bits would you need to represent such a value? A: 25 bits

1. Draw a schematic.

图片包含 文字, 白板

描述已自动生成

1. Write a Verilog module.

module mux(SW, KEY, CLOCK\_50, HEX0);

input [1:0] SW;

input [2:0] KEY;

input CLOCK\_50;

output [6:0] HEX0;

wire connect;

wire [3:0] Q;

RateDivider a(

.m(SW[1]),

.n(SW[0]),

.clock(CLOCK\_50),

.out(connect)

);

DisplayCounter b(

.enable(connect),

.reset\_n(KEY[0]),

.clock2(CLOCK\_50),

.q(Q[3:0])

);

hex u0(

.in2(Q[3]),

.in3(Q[2]),

.in4(Q[1]),

.in5(Q[0]),

.o2(HEX0[0]),

.o3(HEX0[1]),

.o4(HEX0[2]),

.o5(HEX0[3]),

.o6(HEX0[4]),

.o7(HEX0[5]),

.o8(HEX0[6])

);

endmodule

module RateDivider(m, n, clock, out);

parameter N = 28;

input m, n;

input clock;

output out;

reg [N-1:0] count;

reg out;

always @(posedge clock)

begin

if (m == 0)

begin

if (n == 0)

assign count = clock;

else if (n == 1)

begin

if (count == 28'b0010111110101111000001111111)

count <= 0;

else

count <= count + 1'b1;

end

end

else if (m == 1)

begin

if (n == 0)

begin

if (count == 28'b0101111101011110000011111111)

count <= 0;

else

count <= count + 1'b1;

end

else if (n == 1)

begin

if (count == 28'b1011111010111100000111111111)

count <= 0;

else

count <= count + 1'b1;

end

end

if (count == 0)

out <= 1;

else

out <= 0;

end

endmodule

module DisplayCounter(enable, reset\_n, clock2, q);

parameter k = 4;

input enable, reset\_n, clock2;

output [k-1:0] q;

reg [k-1:0] q;

always @(posedge clock2)

begin

if (reset\_n == 1'b0)

q <= 0;

else if (enable == 1'b1)

begin

if (q == 4'b1111)

q <= 0;

else

q <= q + 1'b1;

end

end

endmodule

module hex(in2, in3, in4, in5, o2, o3, o4, o5, o6, o7, o8);

input in2;

input in3;

input in4;

input in5;

output o2;

output o3;

output o4;

output o5;

output o6;

output o7;

output o8;

hex0 u1(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out1(o2)

);

hex1 u2(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out2(o3)

);

hex2 u3(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out3(o4)

);

hex3 u4(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out4(o5)

);

hex4 u5(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out5(o6)

);

hex5 u6(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out6(o7)

);

hex6 u7(

.c3(in2),

.c2(in3),

.c1(in4),

.c0(in5),

.out7(o8)

);

endmodule

module hex0(c3, c2, c1, c0, out1);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out1; //output

assign out1 = ~c3 & c2 & ~c1 & ~c0 | c3 & c2 & ~c1 & c0 | c3 & ~c2 & c1 & c0 | ~c3 & ~c2 & ~c1 & c0;

endmodule

module hex1(c3, c2, c1, c0, out2);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out2; //output

assign out2 = c2 & c1 & ~c0 | c3 & c1 & c0 | ~c3 & c2 & ~c1 & c0 | c3 & c2 & ~c0;

endmodule

module hex2(c3, c2, c1, c0, out3);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out3; //output

assign out3 = ~c3 & ~c2 & c1 & ~c0 | c3 & c2 & c1 | c3 & c2 & ~c0;

endmodule

module hex3(c3, c2, c1, c0, out4);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out4; //output

assign out4 = ~c3 & c2 & ~c1 & ~c0 | ~c2 & ~c1 & c0 | c2 & c1 & c0 | c3 & ~c2 & c1 & ~c0;

endmodule

module hex4(c3, c2, c1, c0, out5);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out5; //output

assign out5 = ~c3 & c2 & ~c1 | ~c2 & ~c1 & c0 | ~c3 & c0;

endmodule

module hex5(c3, c2, c1, c0, out6);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out6; //output

assign out6 = ~c3 & ~c2 & c0 | ~c3 & ~c2 & c1 | ~c3 & c1 & c0 | c3 & c2 & ~c1 & c0;

endmodule

module hex6(c3, c2, c1, c0, out7);

input c3; //selected when s is 0

input c2; //selected when s is 1

input c1; //select signal

input c0;

output out7; //output

assign out7 = ~c3 & ~c2 & ~c1 | c3 & c2 & ~c1 & ~c0 | ~c3 & c2 & c1 & c0;

endmodule

1. Simulation.

图片包含 墙壁

描述已自动生成

**Part III**

1. Use Table 1 to determine your code and bit-width.

|  |  |
| --- | --- |
| Letter | Pattern Representation (pattern length is 11 bits) |
| A | 10111000000 |
| B | 11101010100 |
| C | 11101011101 |
| D | 11101010000 |
| E | 10000000000 |
| F | 10101110100 |
| G | 11101110100 |
| H | 10101010000 |

1. Draw a schematic.

图片包含 文字, 白板

描述已自动生成

1. Write a Verilog module.

module mux(SW, KEY, CLOCK\_50, LEDR);

input [2:0] SW;

input [2:0] KEY;

input CLOCK\_50;

output [1:0] LEDR;

wire connect;

wire [11:0] Q;

RateDivider a(

.clock(CLOCK\_50),

.out(connect)

);

DisplayCounter b(

.enable(connect),

.reset\_n(KEY[0]),

.in(q[11:0]),

.start(KEY[1]0,

.clock2(CLOCK\_50),

.L(LEDR[0])

);

LUT c(

.m(SW[2]),

.n(SW[1]),

.o(SW[0]),

.outt(Q[11:0])

);

endmodule

module RateDivider(clock, out);

parameter N = 25;

input clock;

output out;

reg [N-1:0] count;

reg out;

always @(posedge clock)

begin

if (count == 25'b1011111010111100001000000)

count <= 0;

else

count <= count + 1'b1;

if (count == 0)

out <= 1;

else

out <= 0;

end

endmodule

module ShiftRegister(enable, reset\_n, in, start, clock2, L);

parameter k = 11;

parameter s = 3;

input enable, reset\_n, start, clock2;

input [k-1:0] in;

output L;

wire [s-1:0] q;

always @(posedge clock2, posedge reset\_n)

begin

if (reset\_n == 1'b0)

q <= 0;

else if (start == 1'b1)

q <= 0;

else if (enable == 1'b1)

begin

if (q == 4'b1111)

q <= 0;

else

q <= q + 1'b1;

end

end

always @(posedge clock2, posedge reset\_n)

begin

case(q)

4'b0000: L = 0;

4'b0001: L = in[10];

4'b0010: L = in[9];

4'b0011: L = in[8];

4'b0100: L = in[7];

4'b0101: L = in[6];

4'b0110: L = in[5];

4'b0111: L = in[4];

4'b1000: L = in[3];

4'b1001: L = in[2];

4'b1010: L = in[1];

4'b1011: L = in[0];

default L = 0

endcase

end

endmodule

module LUT(m, n, o, outt);

parameter a = 11;

input m, n, o;

output [a-1:0] outt;

always @(\*)

begin

case(in1)

3'b000: outt = 11'b10111000000;

3'b001: outt = 11'b11101010100;

3'b010: outt = 11'b11101011101

3'b011: outt = 11'b11101010000;

3'b100: outt = 11'b10000000000;

3'b101: outt = 11'b10101110100;

3'b110: outt = 11'b11101110100;

3'b111: outt = 11'b10101010000;

default: out1 = 11'b0000\_0000\_000;

endcase

end

endmodule