module button\_control(input clock, input reset, input button, output reg valid\_vote);

reg [30:0] counter;

wire clk;

userclock(clock,clk);

always@(posedge clock)

begin

if(reset)

counter <= 0;

else

begin

if(button&counter<11)

counter<=0;

end

end

always@(posedge clock)

begin

if(reset)

valid\_vote<= 1'b0;

else

begin

if(counter == 10)

valid\_vote <= 1'b1;

else

valid\_vote <= 1'b0;

end

end

endmodule

module votelogger(

input clock,

input reset,

input mode,

input cand1\_vote\_valid,

input cand2\_vote\_valid,

input cand3\_vote\_valid,

input cand4\_vote\_valid,

output reg[7:0] cand1\_vote\_recvd,

output reg[7:0] cand2\_vote\_recvd,

output reg[7:0] cand3\_vote\_recvd,

output reg[7:0] cand4\_vote\_recvd

);

wire clk;

userclock(clock,clk);

always@(posedge clock)

begin

if (reset)

begin

cand1\_vote\_recvd <= 0;

cand2\_vote\_recvd <= 0;

cand3\_vote\_recvd <= 0;

cand4\_vote\_recvd <= 0;

end

else

begin

if(cand1\_vote\_valid & mode == 0)

cand1\_vote\_recvd <= cand1\_vote\_recvd+1;

else if(cand1\_vote\_valid& mode == 0)

cand2\_vote\_recvd <= cand2\_vote\_recvd+1;

else if(cand3\_vote\_valid& mode == 0)

cand3\_vote\_recvd <= cand3\_vote\_recvd+1;

else if(cand4\_vote\_valid & mode == 0)

cand4\_vote\_recvd <= cand4\_vote\_recvd+1;

end

end

endmodule

module modecontrol(

input clock,

input reset,

input mode,

input valid\_vote\_casted,

input [7:0] candidate1\_vote,

input [7:0] candidate2\_vote,

input [7:0] candidate3\_vote,

input [7:0] candidate4\_vote,

input candidate1\_button\_press,

input candidate2\_button\_press,

input candidate3\_button\_press,

input candidate4\_button\_press,

output reg[7:0] leds

);

wire clk;

userclock(clock,clk);

reg[30:0] counter;

always@ (posedge clock)

begin

if(reset)

counter<=0;

else if(valid\_vote\_casted)

counter <= counter+1;

else

counter <= 0;

end

always@(posedge clock)

begin

if(reset)

leds= 0;

else

begin

if(mode == 0 & counter >0)

leds <= 8'hFF;

else if(mode == 0)

leds <= 8'h00;

else if (mode == 1)

begin

if(candidate1\_button\_press)

leds <= candidate1\_vote;

else if (candidate2\_button\_press)

leds <= candidate2\_vote;

else if (candidate3\_button\_press)

leds <= candidate3\_vote;

else if (candidate4\_button\_press)

leds <= candidate4\_vote;

end

end

end

endmodule

module Voting\_machine(

input clock,

input reset,

input mode,

input button1,

input button2,

input button3,

input button4,

output [7:0]led

);

wire clk;

userclock(clock,clk);

wire validvote\_1;

wire validvote\_2;

wire validvote\_3;

wire validvote\_4;

wire [7:0] cand1\_vote\_recvd;

wire [7:0] cand2\_vote\_recvd;

wire [7:0] cand3\_vote\_recvd;

wire [7:0] cand4\_vote\_recvd;

wire anyvalidvote;

assign anyvalidvote = validvote\_1|validvote\_2|validvote\_3|validvote\_4;

button\_control bc1(

.clock(clock),

.reset(reset),

.button(button1),

.valid\_vote(validvote\_1)

);

button\_control bc2(

.clock(clock),

.reset(reset),

.button(button2),

.valid\_vote(validvote\_2)

);

button\_control bc3(

.clock(clock),

.reset(reset),

.button(button2),

.valid\_vote(validvote\_3)

);

button\_control bc4(

.clock(clock),

.reset(reset),

.button(button4),

.valid\_vote(validvote\_4)

);

votelogger VL(

.clock(clock),

.reset(reset),

.mode(mode),

.cand1\_vote\_valid(validvote\_1),

.cand2\_vote\_valid(validvote\_2),

.cand3\_vote\_valid(validvote\_3),

.cand4\_vote\_valid(validvote\_4),

.cand1\_vote\_recvd(cand1\_vote\_recvd),

.cand2\_vote\_recvd(cand2\_vote\_recvd),

.cand3\_vote\_recvd(cand3\_vote\_recvd),

.cand4\_vote\_recvd(cand4\_vote\_recvd)

);

modecontrol MC (

.clock(clock),

.reset(reset),

.mode(mode),

.valid\_vote\_casted(anyvalidvote),

.candidate1\_vote(cand1\_vote\_recvd),

.candidate2\_vote(cand2\_vote\_recvd),

.candidate3\_vote(cand3\_vote\_recvd),

.candidate4\_vote(cand4\_vote\_recvd),

.candidate1\_button\_press(validvote\_1),

.candidate2\_button\_press(validvote\_2),

.candidate3\_button\_press(validvote\_3),

.candidate4\_button\_press(validvote\_4),

.leds(led)

);

endmodule

module userclock(input clock,output clk);

reg clk\_out=0;

reg [25:0] count=0;

always @(posedge clock)

begin

count<=count+1;

if (count==9500000)

begin

count<=0;

clk\_out=~clk\_out;

end

end

assign clk=clk\_out;

endmodule