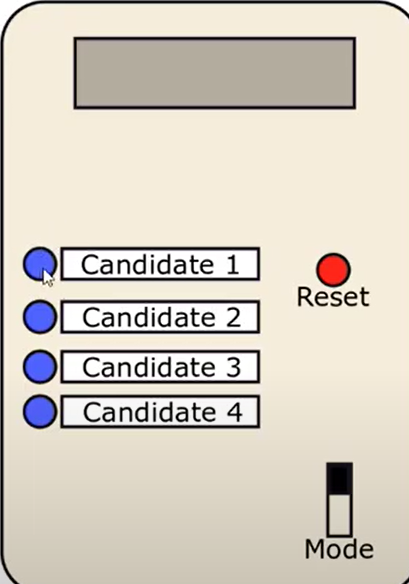
Designing a Simple Voting Machine using FPGAs with Verilog HDL and Vivaado

Work- simple voting machine which works in two modes

1. Voting mode:- user can cast their vote only
2. Result mode:- get the count of votes for a particular candidate



Sample machine view

Board- zedboard

Hierarchy design approach:- part by part design and combine all later

At a time 4 candidate can be on board because of 4 buttons available

And a slide switch as mode

And a button as reset

And 8 leds for output

A long press can cast only one vote unless it will pressed less than 1s.

STEPS:=

1. **Module buttonControl:**

. circuit use for casting vote.

Define the time period so that min 1s it will taken as a real press

We use 4 buttons for casting vote

A reset

A clock

Create a new source file named buttonControl.v

CODE”=>

`timescale 1ns / 1ps

module buttonControl(

input clock,

input reset,

input button,

output reg valid\_vote

);

//counter to count for 1 s to check whether button pressed more than or equal to 1 s or not

reg [30:0] counter;

always @(posedge clock)

begin

if(reset)

counter<=0;

else

begin

if(button & counter<100000001)

counter<=counter+1;

else if(!button)

counter<=0;

end

end

//get a valid\_vote high

always @(posedge clock)

begin

if(reset)

valid\_vote<=1'b0;

else

begin

if(counter==100000000)

valid\_vote<=1'b1;

else

valid\_vote<=1'b0;

end

end

endmodule

for counting vote

add new source named as voteLogger.v

it will count the vote casted to each candidate

CODE:=>

**2. Module voteLogger.v:**

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 WILL GIVE 8 BIT VOTE RECEIVED THAT IS MAX UPTO 255 VOTES .

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

);

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

// mode==0 is necessary condition ,, if it is not given here whether a button pressed for counting it will increment a vote also while it is in result mode

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

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

else if(cand2\_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

now also create a new file source named as mode control for mode control whther it will be in result mode or voting mode

**3.Module modeControl.v**

|  |
| --- |
| 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 |
|  | ); |
|  |  |
|  | reg [30:0] counter;  //need a counter how long led will glow |
|  |  |
|  | always @(posedge clock) |
|  | begin |
|  | if(reset) |
|  | counter <= 0; //Whenever reset is pressed, counter started from 0 |
|  | else if(valid\_vote\_casted) //If a valid vote is casted, counter becomes 1 |
|  | counter <= counter + 1; |
|  | else if(counter !=0 & counter < 100000000)//If counter is not 0, increment it till 100000000 |
|  | counter <= counter + 1; |
|  | else //Once counter becomes 100000000, reset it to zero |
|  | counter <= 0; |
|  | end |
|  |  |
|  | always @(posedge clock) |
|  | begin |
|  | if(reset) |
|  | leds <= 0; |
|  | else |
|  | begin |
|  | if(mode == 0 & counter > 0 ) //mode0 -> voting mode, mode 1 -> result mode |
|  | leds <= 8'hFF; |
|  | else if(mode == 0) |
|  | leds <= 8'h00; |
|  | else if(mode == 1) //result mode |
|  | 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 |

**4. final main module voteMachine.v**

**Make this as set top module**

**=1> how to combine diffetent sources in another source.v**

Using dot maping and set formal port

CODE=>

//inputs:=>

module voteMachine(

input clock,

input reset,

input mode,

input button1,

input button2,

input button3,

input button4,

output [7:0] led

);

//**combining buttonControl.v module**

buttonControl bc1(

Dot mapping

.clock(clock),

Formal port(in voteMachine with which port it will be connected

.reset(reset),

.button(button1),

.valid\_vote(valid\_vote\_1)

);

//Similiarly for all 4 candidte

//**combining voteLogger**

voteLogger VL(

.clock(clock),

.reset(reset),

.mode(mode),

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

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

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

.cand4\_vote\_valid(valid\_vote\_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)

);

**//combining modeControl**

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(valid\_vote\_1),

.candidate2\_button\_press(valid\_vote\_2),

.candidate3\_button\_press(valid\_vote\_3),

.candidate4\_button\_press(valid\_vote\_4),

.leds(led)

);

**AFTER COMBINIG ALL THE MODULE voteMacine will be**

module voteMachine(

input clock,

input reset,

input mode,

input button1,

input button2,

input button3,

input button4,

output [7:0] led

);

wire valid\_vote\_1;

wire valid\_vote\_2;

wire valid\_vote\_3;

wire valid\_vote\_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 = valid\_vote\_1|valid\_vote\_2|valid\_vote\_3|valid\_vote\_4;

buttonControl bc1(

.clock(clock),

.reset(reset),

.button(button1),

.valid\_vote(valid\_vote\_1)

);

buttonControl bc2(

.clock(clock),

.reset(reset),

.button(button2),

.valid\_vote(valid\_vote\_2)

);

buttonControl bc3(

.clock(clock),

.reset(reset),

.button(button3),//

.valid\_vote(valid\_vote\_3)

);

buttonControl bc4(

.clock(clock),

.reset(reset),

.button(button4),

.valid\_vote(valid\_vote\_4)

);

voteLogger VL(

.clock(clock),

.reset(reset),

.mode(mode),

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

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

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

.cand4\_vote\_valid(valid\_vote\_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(valid\_vote\_1),

.candidate2\_button\_press(valid\_vote\_2),

.candidate3\_button\_press(valid\_vote\_3),

.candidate4\_button\_press(valid\_vote\_4),

.leds(led)

);

=>run synthesis

=> edit time constraint =>create a new clock(we can check whether it is created or not in constraint section as .xdc file )

And layout changed to I/O mapping

Map the different ports

Led0 – T22

Led1 -T21

Led2 – U22

Led3 – U21

Led4 – V22

Led5 – W22

Led6 – U19

Led7 – U14

Button1 – T18

Button2 – R18

Button3 – R16

Button4 – N15

Clock – Y9

Reset – P16

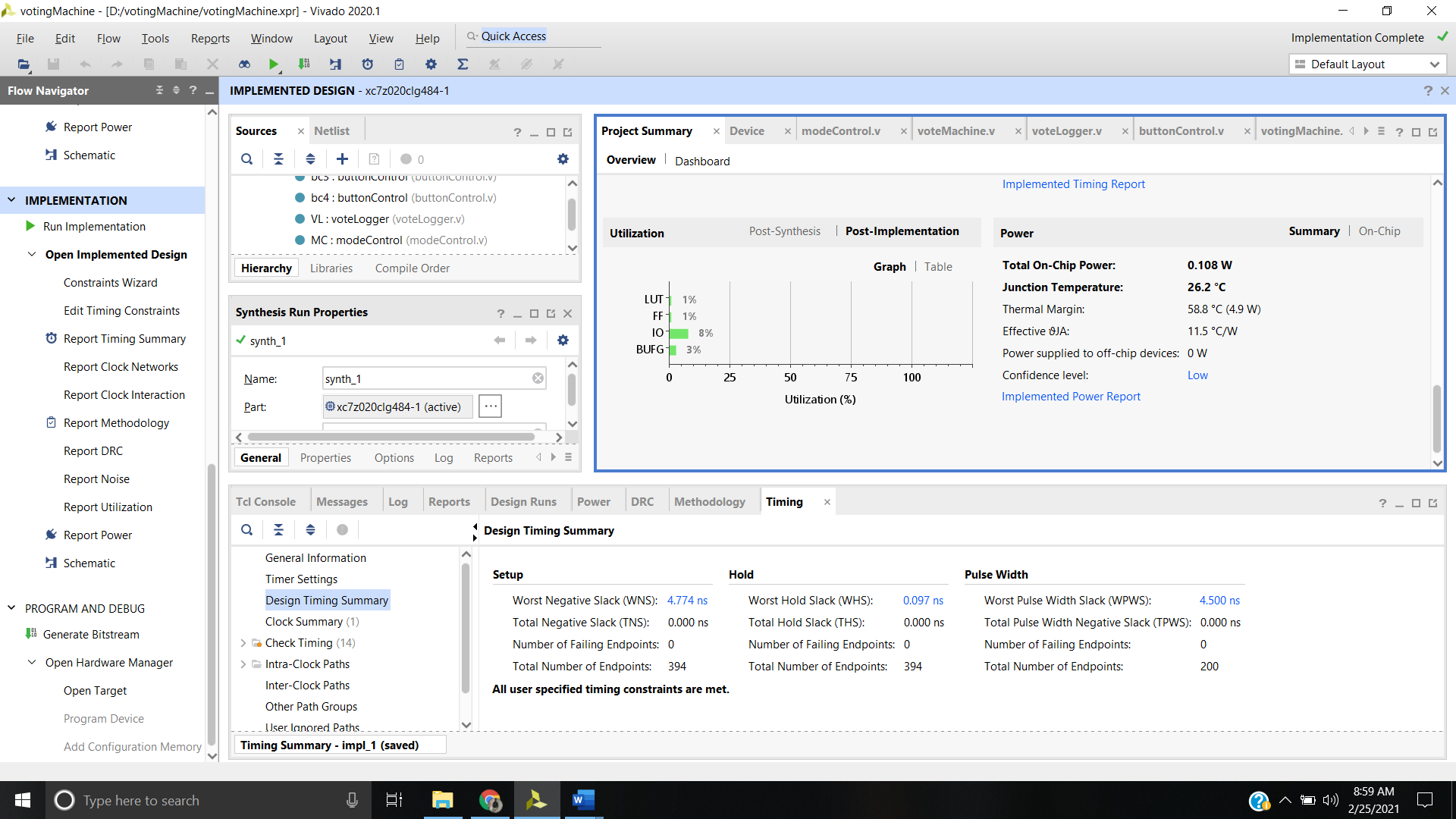
Mode – F22 (SLIDE SWITCH)

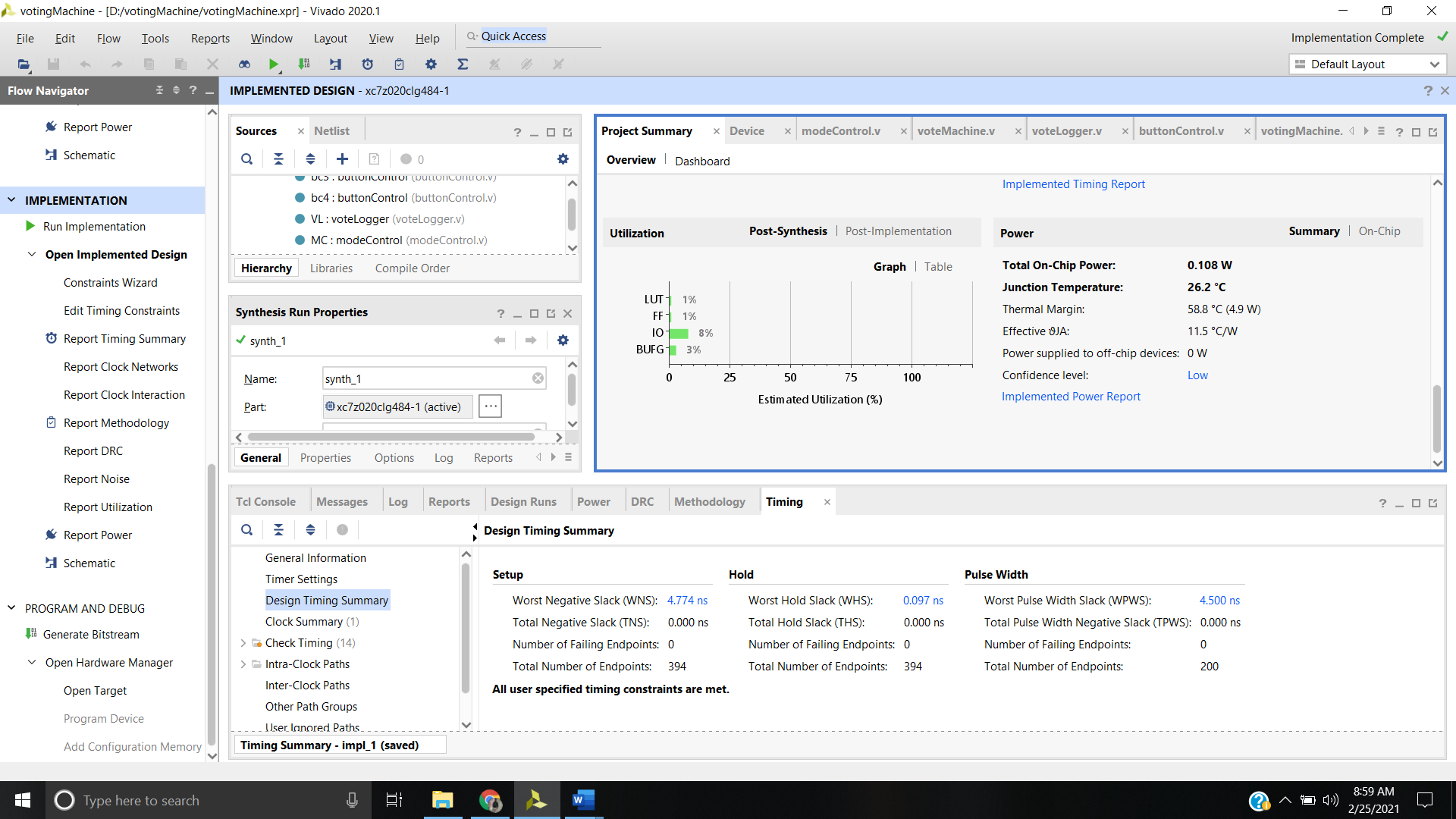
MAKE i/o STD FROM DEFAULT TO LVMCMOS18

=>SAVE IT=>MAKE synthesis design force-up- to- date

=>SAVE IT => GENERATE BITSTREAM =>PROGRAM (IF DEVICE IS CONNECTED)

Project summary:=>





=>generate bitstream=>program in board device