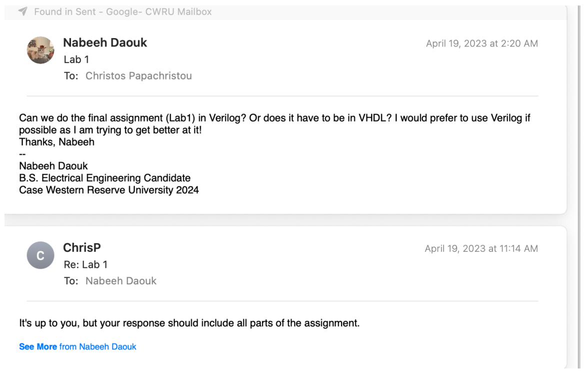
Lab 1 Report

Nabeeh Daouk and Gabriel Foss



Found in Sent – Google- CWRU Mailbox

Nabeeh Daouk April 19, 2023 at 2:20 AM
Lab 1
To: Christos Papachristou

Can we do the final assignment (Lab1) in Verilog? Or does it have to be in VHDL? I would prefer to use Verilog if possible as I am trying to get better at it!
Thanks, Nabeeh
--
Nabeeh Daouk
B.S. Electrical Engineering Candidate
Case Western Reserve University 2024

ChrisP April 19, 2023 at 11:14 AM
Re: Lab 1
To: Nabeeh Daouk

It's up to you, but your response should include all parts of the assignment.

See More from Nabeeh Daouk

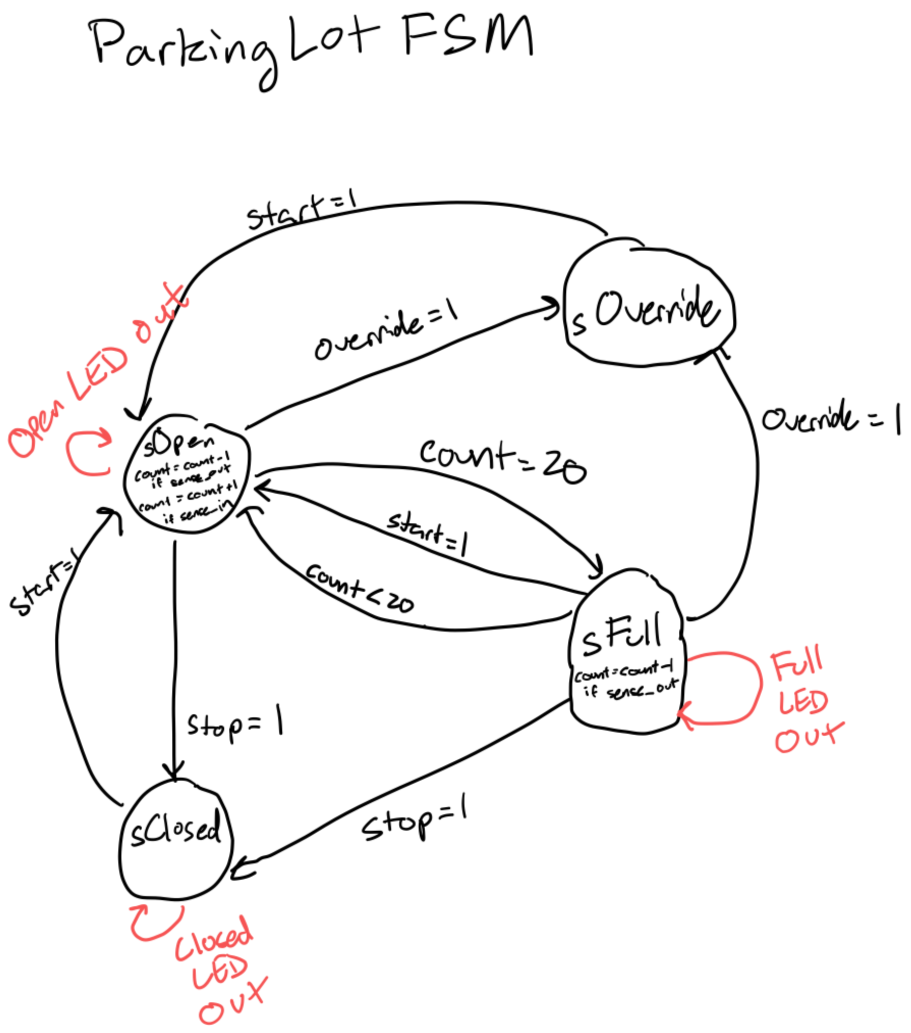
We received permission from Chris Papachristou to do the lab in Verilog. We wrote a version in verilog, and a version in VHDL.

Reports in **Vivado Reports** Folder
Verilog Source Files included in **VerilogSourceCode**
VHDL Source Files included in **VHDLSourceCode**
Synopsys output files and log are included in **SynopsysOutput**
Terminal transcripts are included
Demo on 04/27/2023

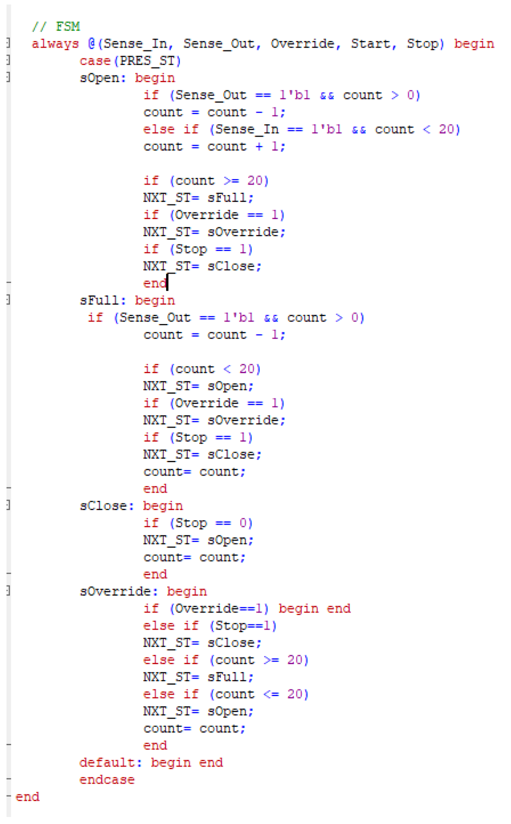
Source Files and Reports Included in Lab1_Files.zip folder

Demo was shown on 4/27/2023 to William Brosie.

Demo Notes

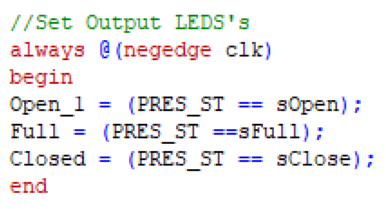

Parking Lot FSM
sOpen
count = count - 1 if sense_out
count = count + 1 if sense_in
Open LED out
s Override
sFull
count = count - 1 if sense_out
Full LED out
sClosed
Closed LED out
start = 1
override = 1
Override = 1
count = 20
start = 1
count < 20
start = 1
stop = 1
stop = 1

FSM Diagram



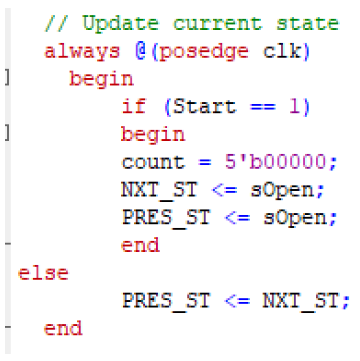
```
// FSM
always @(Sense_In, Sense_Out, Override, Start, Stop) begin
      case(PRES_ST)
      sOpen: begin
              if (Sense_Out == 1'b1 && count > 0)
              count = count - 1;
              else if (Sense_In == 1'b1 && count < 20)
              count = count + 1;

              if (count >= 20)
              NXT_ST= sFull;
              if (Override == 1)
              NXT_ST= sOverride;
              if (Stop == 1)
              NXT_ST= sClose;
              end
      sFull: begin
       if (Sense_Out == 1'b1 && count > 0)
              count = count - 1;

              if (count < 20)
              NXT_ST= sOpen;
              if (Override == 1)
              NXT_ST= sOverride;
              if (Stop == 1)
              NXT_ST= sClose;
              count= count;
              end
      sClose: begin
              if (Stop == 0)
              NXT_ST= sOpen;
              count= count;
              end
      sOverride: begin
              if (Override==1) begin end
              else if (Stop==1)
              NXT_ST= sClose;
              else if (count >= 20)
              NXT_ST= sFull;
              else if (count <= 20)
              NXT_ST= sOpen;
              count= count;
              end
      default: begin end
      endcase
end
```

FSM Code
Note that the FSM will check its current state and determine the next state whenever the inputs to the FSM change.



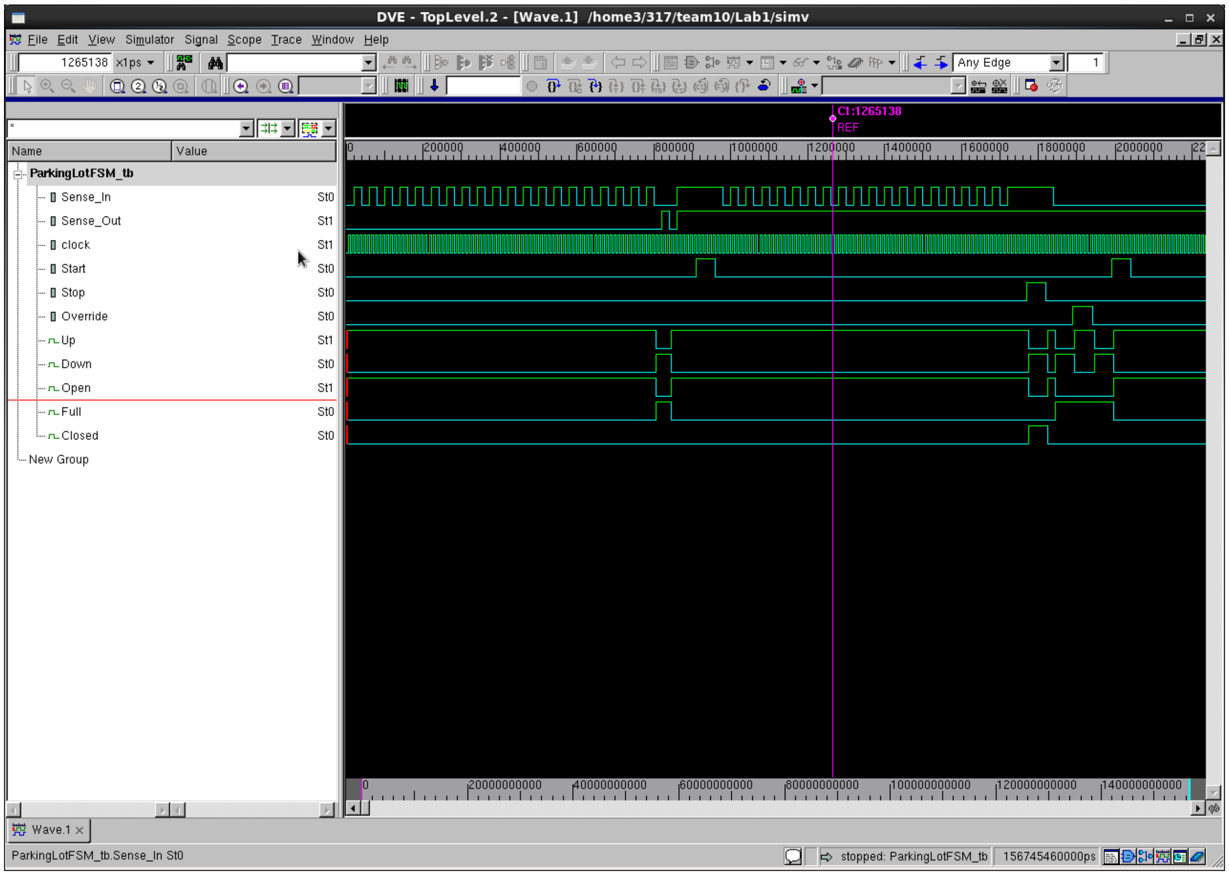
```
//Set Output LEDS's
always @(negedge clk)
begin
Open_1 = (PRES_ST == sOpen);
Full = (PRES_ST ==sFull);
Closed = (PRES_ST == sClose);
end
```

Outputs Code
Each output LED corresponds to a state of the FSM. Override will turn off the LED. Note that the LED is set at every clock edge as it corresponds to solely the state at the time.



```
  // Update current state
  always @(posedge clk)
    begin
        if (Start == 1)
        begin
        count = 5'b00000;
        NXT_ST <= sOpen;
        PRES_ST <= sOpen;
        end
else
        PRES_ST <= NXT_ST;
  end
```

Update State/ Reset Code
Every clock cycle, the next state is saved as the present state. Note that if start is pressed, the FSM will reset the count and set both next and present state to sOpen with a count of 0. This functions as a hard reset to the FSM.



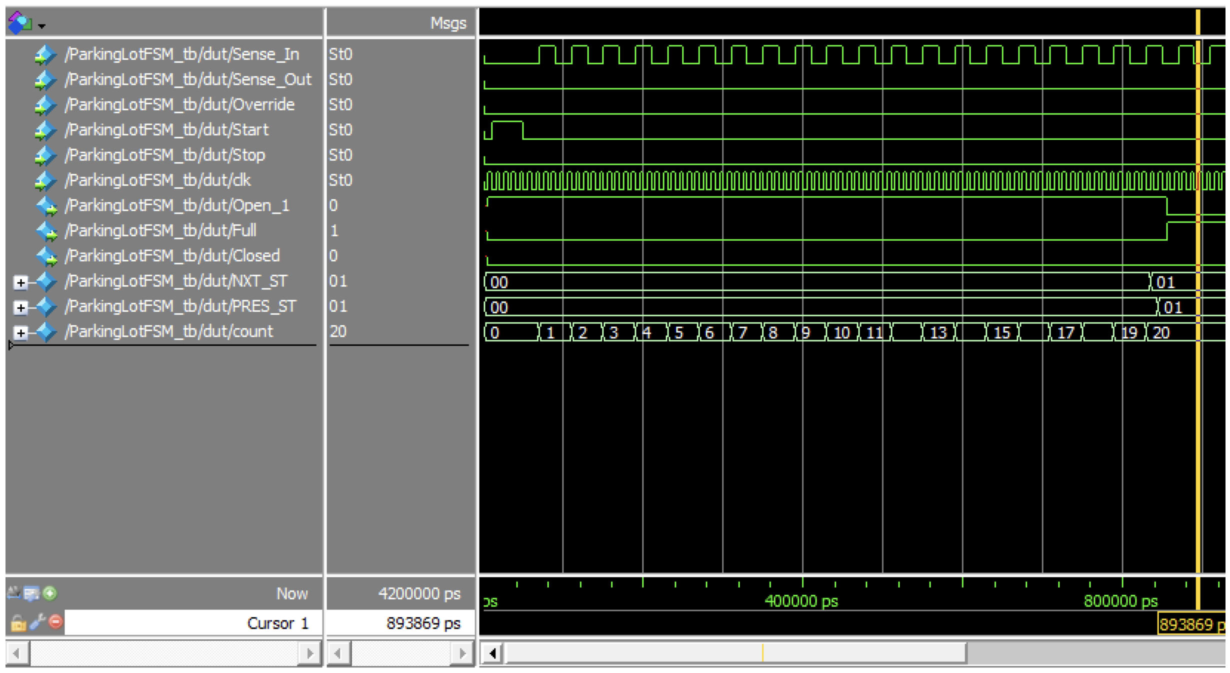
DVE - TopLevel.2 - [Wave.1] /home3/317/team10/Lab1/simv

Name	Value
ParkingLotFSM_tb	
Sense_In	St0
Sense_Out	St1
clock	St1
Start	St0
Stop	St0
Override	St0
Up	St1
Down	St0
Open	St1
Full	St0
Closed	St0
New Group	

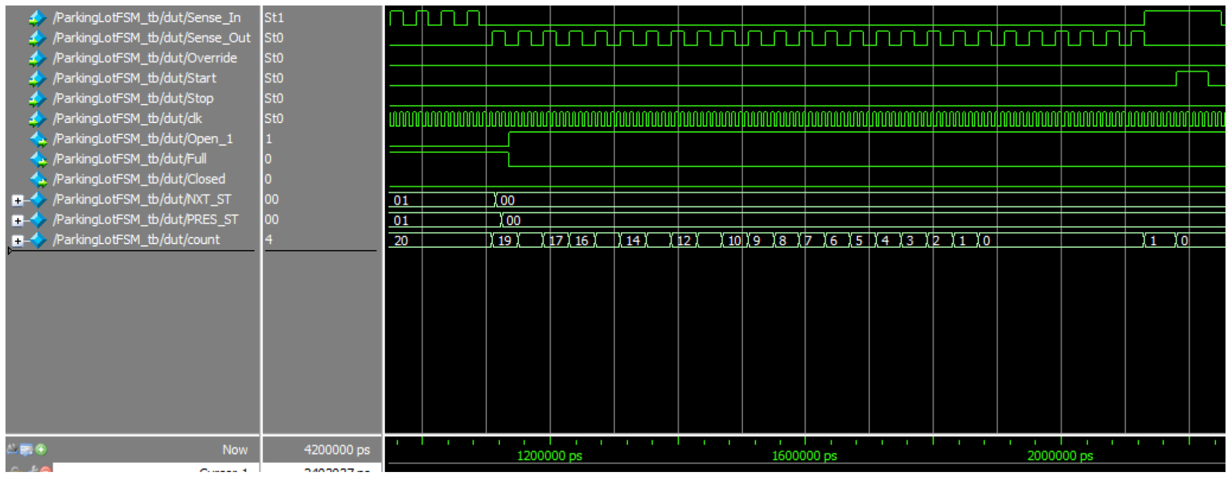
ParkingLotFSM_tb.Sense_In St0

stopped: ParkingLotFSM_tb 156745460000ps

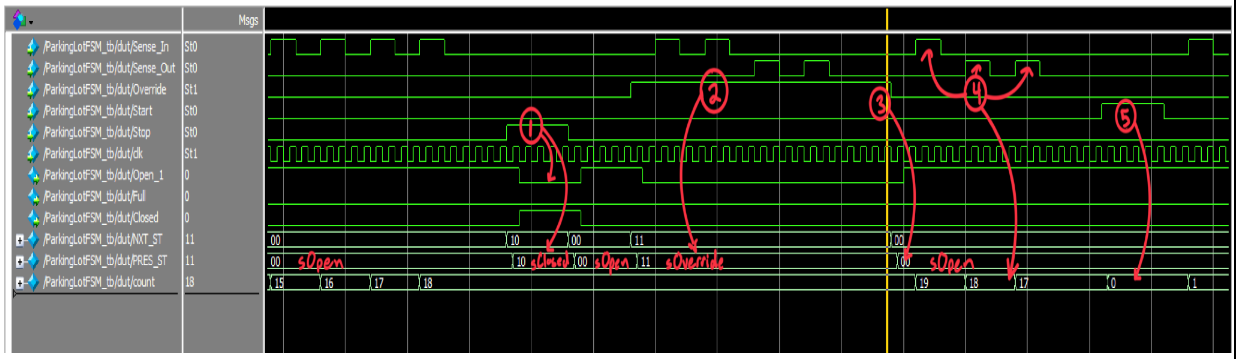
Testbench: This is the full waveform that shows all test situations. Each situation will be discussed below. Modelsim will be used as screen captures are more clear. But we initially simulated in Synopsys.


Msgs
/ParkingLotFSM_tb/dut/Sense_In St0
/ParkingLotFSM_tb/dut/Sense_Out St0
/ParkingLotFSM_tb/dut/Override St0
/ParkingLotFSM_tb/dut/Start St0
/ParkingLotFSM_tb/dut/Stop St0
/ParkingLotFSM_tb/dut/clk St0
/ParkingLotFSM_tb/dut/Open_1 0
/ParkingLotFSM_tb/dut/Full 1
/ParkingLotFSM_tb/dut/Closed 0
/ParkingLotFSM_tb/dut/NXT_ST 01
/ParkingLotFSM_tb/dut/PRES_ST 01
/ParkingLotFSM_tb/dut/count 20
00
01
00
01
0 1 2 3 4 5 6 7 8 9 10 11 13 15 17 19 20
Now 4200000 ps
Cursor 1 893869 ps
400000 ps
800000 ps
893869 p

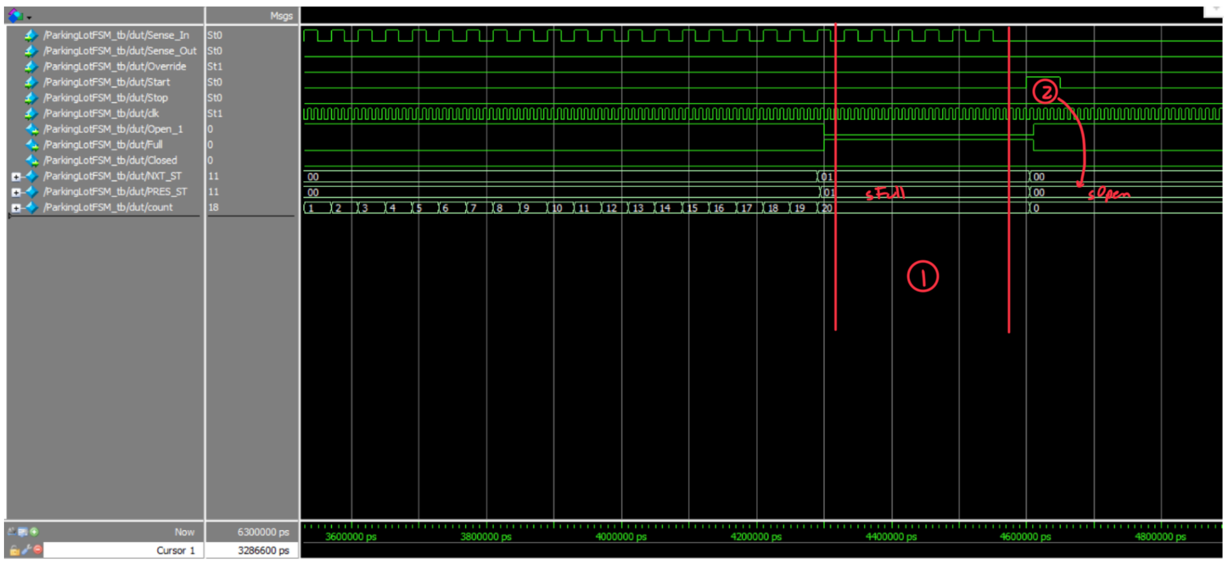
Here we see **start** asserted and then count increments up to a max of 20 as 20 cars **enter**. During this time the Open LED is asserted and the FSM is in the sOpen(00) state. When count reaches 20, the Full LED is asserted as the FSM enters PRES_ST=01(sFull). All Sense_In/ entering car input signals are rejected when count is 20.


/ParkingLotFSM_tb/dut/Sense_In St1
/ParkingLotFSM_tb/dut/Sense_Out St0
/ParkingLotFSM_tb/dut/Override St0
/ParkingLotFSM_tb/dut/Start St0
/ParkingLotFSM_tb/dut/Stop St0
/ParkingLotFSM_tb/dut/clk St0
/ParkingLotFSM_tb/dut/Open_1 1
/ParkingLotFSM_tb/dut/Full 0
/ParkingLotFSM_tb/dut/Closed 0
/ParkingLotFSM_tb/dut/NXT_ST 00
/ParkingLotFSM_tb/dut/PRES_ST 00
/ParkingLotFSM_tb/dut/count 4
01
00
01
00
20 19 17 16 14 12 10 9 8 7 6 5 4 3 2 1 0 1 0
Now 4200000 ps
1200000 ps
1600000 ps
2000000 ps

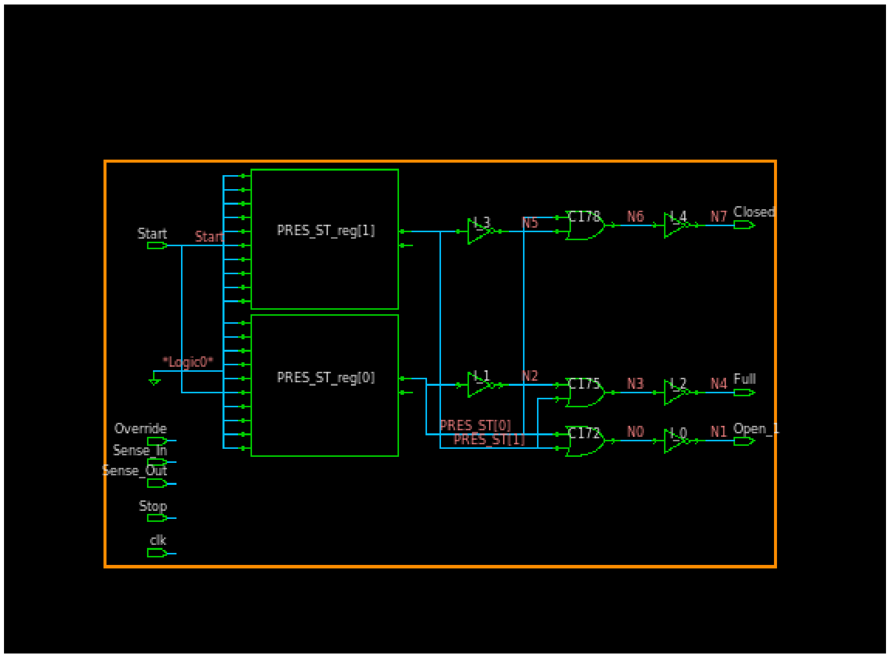
Then we see 20 cars **exit**. Once we see the count go from 20 to 19, the PRES_ST changes from 01(sFull) to 00(sOpen) and the Open LED is asserted again as there are under 20 cars in the lot, making it open to cars. Once the lot is empty, 1 car enters the lot. This sets count to count=1. Then **start** is asserted and the **count** is reset to 0. This shows that start resets the count and state properly.


Msgs
/ParkingLotFSM_tb/dut/Sense_In St0
/ParkingLotFSM_tb/dut/Sense_Out St0
/ParkingLotFSM_tb/dut/Override St1
/ParkingLotFSM_tb/dut/Start St0
/ParkingLotFSM_tb/dut/Stop St0
/ParkingLotFSM_tb/dut/clk St1
/ParkingLotFSM_tb/dut/Open_1 0
/ParkingLotFSM_tb/dut/Full 0
/ParkingLotFSM_tb/dut/Closed 0
/ParkingLotFSM_tb/dut/NXT_ST 11
/ParkingLotFSM_tb/dut/PRES_ST 11
/ParkingLotFSM_tb/dut/count 18
00 10 00 11 00
00 sOpen 10 sClosed 00 sOpen 11 sOverride 00 sOpen
15 16 17 18 19 18 17 0 1
1
2
3
4
5

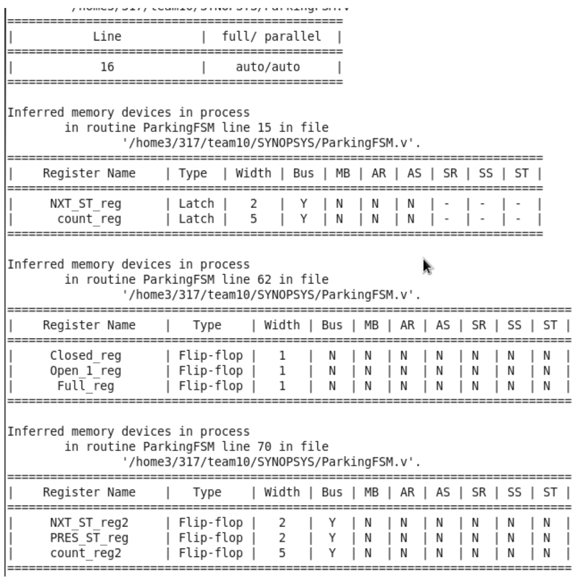
1. Here we see **Stop** asserted and the garage Closed led will light up as the FSM enters the sClosed state (PRES_ST=10). Then Stop returns to 0 causing the FSM goes back into sOpen and the Open LED is asserted again as the lot is reopened.
2. Then **Override** is asserted and we go to the sOverride state (11). While in sOverride, 2 pulses are detected on the Sense_In and Sense_Out respectively. We can see that the count DOES NOT change while override is asserted. Therefore the sOverride state is properly rejecting the sensor inputs while Override input is asserted.
3. After Override returns to zero, the FSM returns to the sOpen state and accepts and re-asserts the Open LED.
4. 1 car enters and 2 exit after override is released. We see that the count function has returned to normal
5. Finally, Start is asserted and count resets.


Msgs
/ParkingLotFSM_tb/dut/Sense_In St0
/ParkingLotFSM_tb/dut/Sense_Out St0
/ParkingLotFSM_tb/dut/Override St1
/ParkingLotFSM_tb/dut/Start St0
/ParkingLotFSM_tb/dut/Stop St0
/ParkingLotFSM_tb/dut/clk St1
/ParkingLotFSM_tb/dut/Open_1 0
/ParkingLotFSM_tb/dut/Full 0
/ParkingLotFSM_tb/dut/Closed 0
/ParkingLotFSM_tb/dut/NXT_ST 11
/ParkingLotFSM_tb/dut/PRES_ST 11
/ParkingLotFSM_tb/dut/count 18
00 01 00
00 01 sFull 00 sOpen
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 0
1
2
Now 6300000 ps
Cursor 1 3286600 ps
3600000 ps 3800000 ps 4000000 ps 4200000 ps 4400000 ps 4600000 ps 4800000 ps

1. Here we see that when the FSM is is sFull, it will reject entering cars. Also note that the Full LED is asserted.
2. Here we see that the Start input functions as a total reset of the FSM. When count is pressed, regardless of the state, the FSM will return to sOpen and set the count to 0.

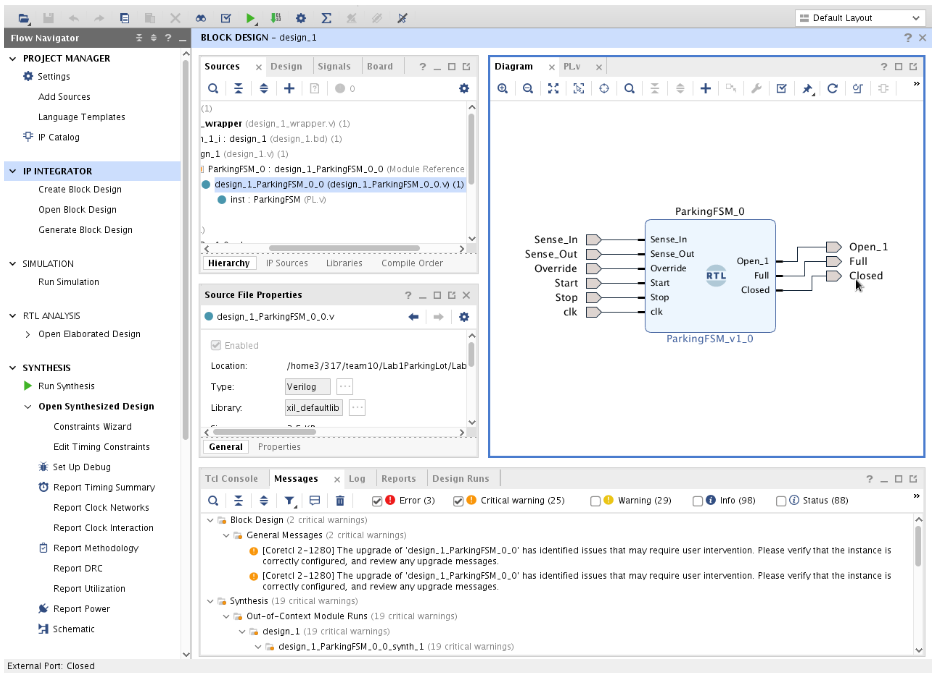


Synthesis System Level Schematic



```
================================================
|           Line            |  full/ parallel  |
================================================
|            16             |    auto/auto     |
================================================

Inferred memory devices in process
        in routine ParkingFSM line 15 in file
                '/home3/317/team10/SYNOPSYS/ParkingFSM.v'.
===========================================================================
|    Register Name     | Type  | Width | Bus | MB | AR | AS | SR | SS | ST |
===========================================================================
|     NXT_ST_reg       | Latch |   2   |  Y  | N  | N  | N  | -  | -  | -  |
|      count_reg       | Latch |   5   |  Y  | N  | N  | N  | -  | -  | -  |
===========================================================================

Inferred memory devices in process
        in routine ParkingFSM line 62 in file
                '/home3/317/team10/SYNOPSYS/ParkingFSM.v'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     Closed_reg       | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|     Open_1_reg       | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
|      Full_reg        | Flip-flop |   1   |  N  | N  | N  | N  | N  | N  | N  |
===============================================================================

Inferred memory devices in process
        in routine ParkingFSM line 70 in file
                '/home3/317/team10/SYNOPSYS/ParkingFSM.v'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     NXT_ST_reg2      | Flip-flop |   2   |  Y  | N  | N  | N  | N  | N  | N  |
|     PRES_ST_reg      | Flip-flop |   2   |  Y  | N  | N  | N  | N  | N  | N  |
|     count_reg2       | Flip-flop |   5   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
```

Synthesis Report Cells
Note: All synthesis reports are stored in the assignSynthesisLog in the zip file



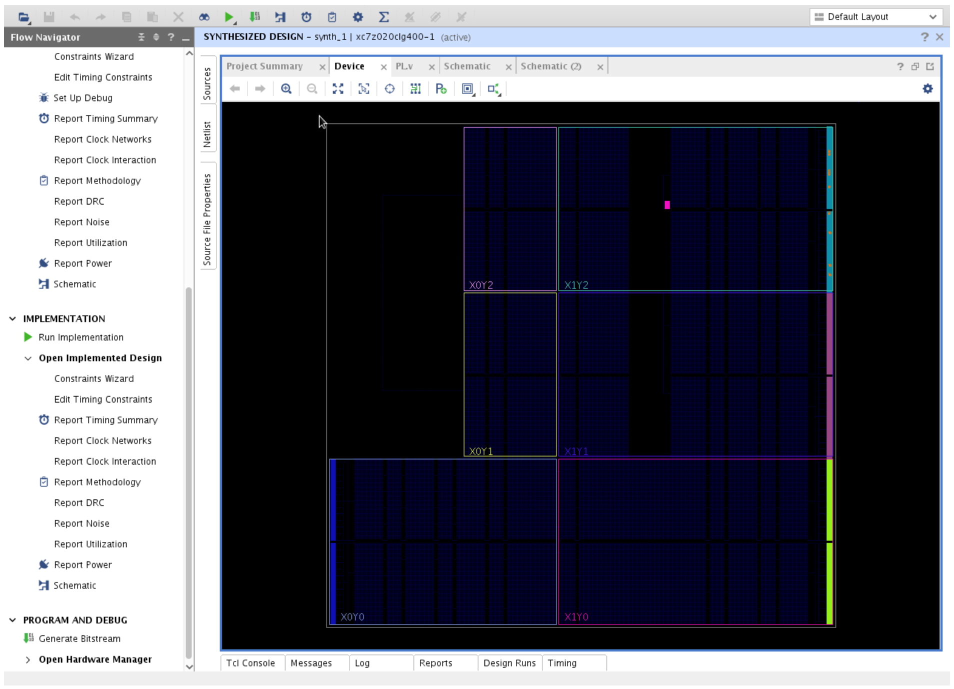
```
_wrapper (design_1_wrapper.v) (1)
1_1_i : design_1 (design_1.bd) (1)
gn_1 (design_1.v) (1)
ParkingFSM_0 : design_1_ParkingFSM_0_0 (Module Reference
design_1_ParkingFSM_0_0 (design_1_ParkingFSM_0_0.v) (1)
inst : ParkingFSM (PL.v)
```

Source File Properties: design_1_ParkingFSM_0_0.v

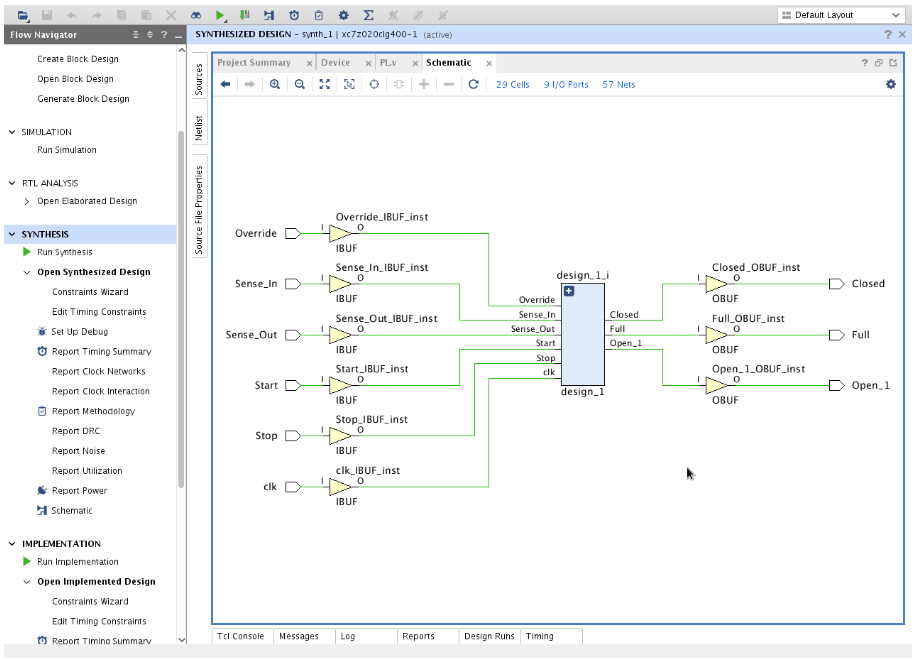
Location:	/home3/317/team10/Lab1ParkingLot/Lab
Type:	Verilog
Library:	xil_defaultlib

```
Error (3)   Critical warning (25)   Warning (29)   Info (98)   Status (88)
Block Design (2 critical warnings)
  General Messages (2 critical warnings)
    [Coretcl 2-1280] The upgrade of 'design_1_ParkingFSM_0_0' has identified issues that may require user intervention. Please verify that the instance is correctly configured, and review any upgrade messages.
    [Coretcl 2-1280] The upgrade of 'design_1_ParkingFSM_0_0' has identified issues that may require user intervention. Please verify that the instance is correctly configured, and review any upgrade messages.
Synthesis (19 critical warnings)
  Out-of-Context Module Runs (19 critical warnings)
    design_1 (19 critical warnings)
      design_1_ParkingFSM_0_0_synth_1 (19 critical warnings)
```

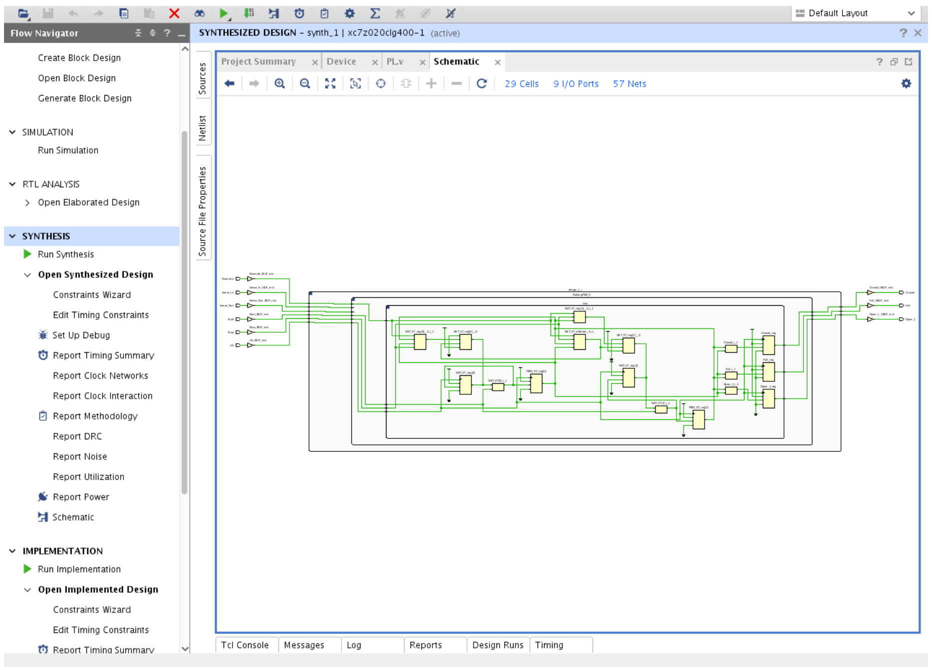
The block diagram from Vivado. 6 inputs and 3 outputs.



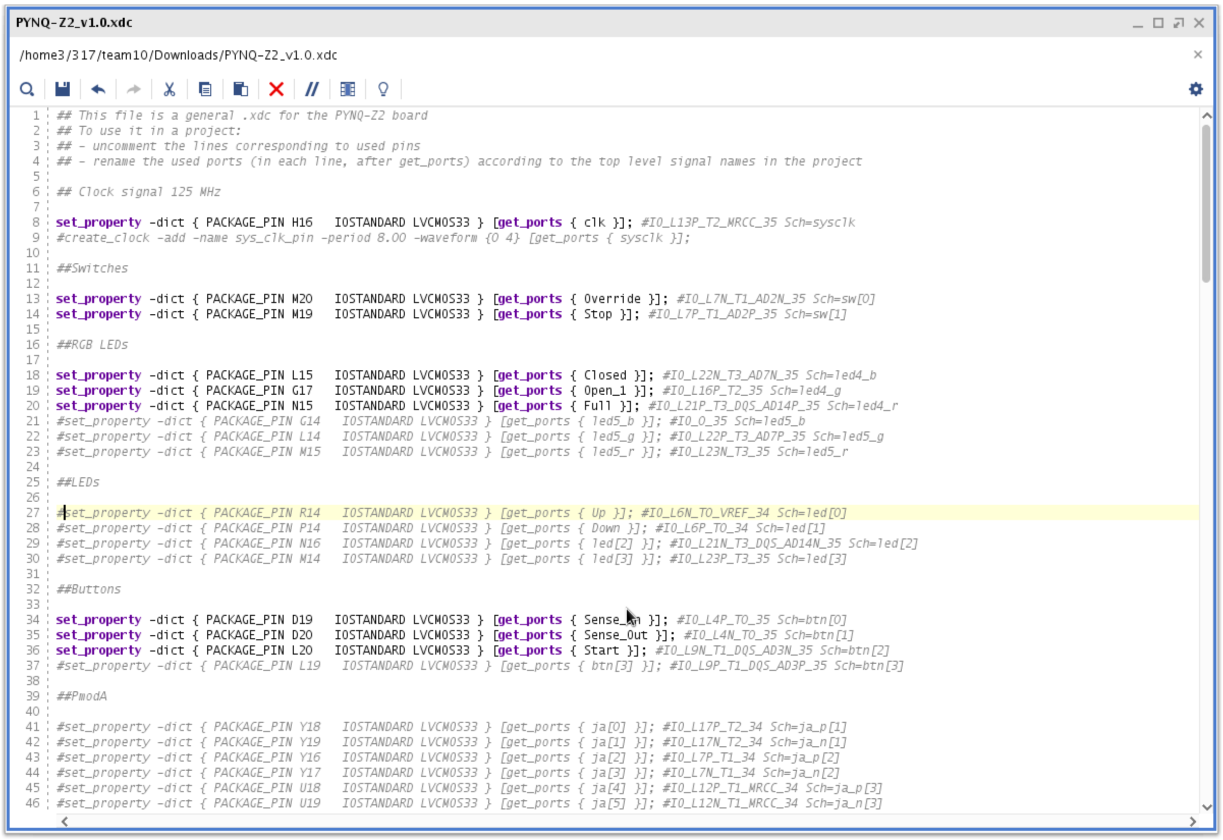
This is the design of the FPGA. Our program takes a very small amount of space, and is shown as the small pink region in block X1Y2.



The design after generating the wrapper.



A schematic of the logic. This is what is shown when opening the MainDesign_1 block above. This is a synthesized representation of our design.


PYNQ-Z2_v1.0.xdc

/home3/317/team10/Downloads/PYNQ-Z2_v1.0.xdc

```
## This file is a general .xdc for the PYNQ-Z2 board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal 125 MHz

set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L13P_T2_MRCC_35 Sch=sysclk
#create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { sysclk }];

##Switches

set_property -dict { PACKAGE_PIN M20   IOSTANDARD LVCMOS33 } [get_ports { Override }]; #IO_L7N_T1_AD2N_35 Sch=sw[0]
set_property -dict { PACKAGE_PIN M19   IOSTANDARD LVCMOS33 } [get_ports { Stop }]; #IO_L7P_T1_AD2P_35 Sch=sw[1]

##RGB LEDs

set_property -dict { PACKAGE_PIN L15   IOSTANDARD LVCMOS33 } [get_ports { Closed }]; #IO_L22N_T3_AD7N_35 Sch=led4_b
set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { Open_1 }]; #IO_L16P_T2_35 Sch=led4_g
set_property -dict { PACKAGE_PIN N15   IOSTANDARD LVCMOS33 } [get_ports { Full }]; #IO_L21P_T3_DQS_AD14P_35 Sch=led4_r
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { led5_b }]; #IO_0_35 Sch=led5_b
#set_property -dict { PACKAGE_PIN L14   IOSTANDARD LVCMOS33 } [get_ports { led5_g }]; #IO_L22P_T3_AD7P_35 Sch=led5_g
#set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led5_r }]; #IO_L23N_T3_35 Sch=led5_r

##LEDs

#set_property -dict { PACKAGE_PIN R14   IOSTANDARD LVCMOS33 } [get_ports { Up }]; #IO_L6N_T0_VREF_34 Sch=led[0]
#set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { Down }]; #IO_L6P_T0_34 Sch=led[1]
#set_property -dict { PACKAGE_PIN N16   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L21N_T3_DQS_AD14N_35 Sch=led[2]
#set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L23P_T3_35 Sch=led[3]

##Buttons

set_property -dict { PACKAGE_PIN D19   IOSTANDARD LVCMOS33 } [get_ports { Sense_In }]; #IO_L4P_T0_35 Sch=btn[0]
set_property -dict { PACKAGE_PIN D20   IOSTANDARD LVCMOS33 } [get_ports { Sense_Out }]; #IO_L4N_T0_35 Sch=btn[1]
set_property -dict { PACKAGE_PIN L20   IOSTANDARD LVCMOS33 } [get_ports { Start }]; #IO_L9N_T1_DQS_AD3N_35 Sch=btn[2]
#set_property -dict { PACKAGE_PIN L19   IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L9P_T1_DQS_AD3P_35 Sch=btn[3]

##PmodA

#set_property -dict { PACKAGE_PIN Y18   IOSTANDARD LVCMOS33 } [get_ports { ja[0] }]; #IO_L17P_T2_34 Sch=ja_p[1]
#set_property -dict { PACKAGE_PIN Y19   IOSTANDARD LVCMOS33 } [get_ports { ja[1] }]; #IO_L17N_T2_34 Sch=ja_n[1]
#set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { ja[2] }]; #IO_L7P_T1_34 Sch=ja_p[2]
#set_property -dict { PACKAGE_PIN Y17   IOSTANDARD LVCMOS33 } [get_ports { ja[3] }]; #IO_L7N_T1_34 Sch=ja_n[2]
#set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { ja[4] }]; #IO_L12P_T1_MRCC_34 Sch=ja_p[3]
#set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports { ja[5] }]; #IO_L12N_T1_MRCC_34 Sch=ja_n[3]
```

The config file mapping the ports of the board to the ports on the block diagram. This board-specific file was downloaded from the Canvas page and the important lines were uncommented. The variables were then renamed based on the design.