FPGA-Programming Course Report

SS-2021

Prof. Dr. Martin Schramm

Group P

Rashed Al-Lahaseh – 00821573

Farouk Turki – 00803941

Huthaifa Jadallah – 00816417

Ravi Yadav - 12102064

Table of Contents

Overview

The main idea of the project is to build an ordering laundry machine for the laundry room in students hall, that helps managing the process of cleaning/washing student's clothes.

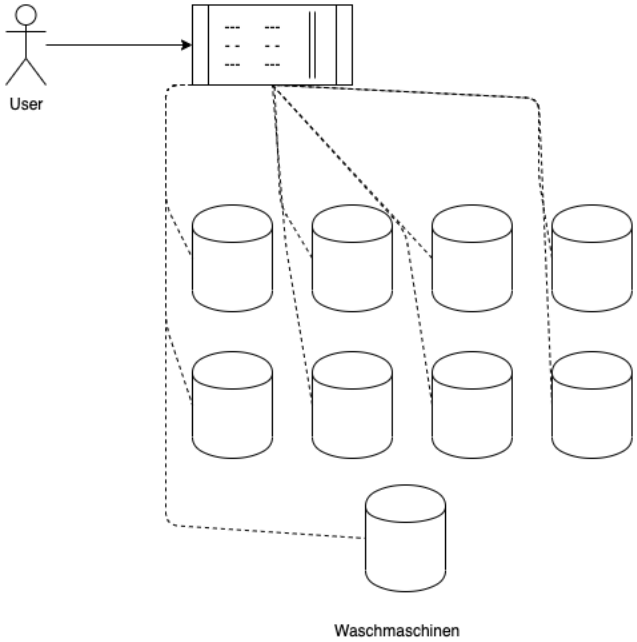

Figure 1: UML Use Case Diagram

Implementation Strategy

Main Steps

The room consist of 9 washing machines and will involve several numbers of sensors for the input also, a payment system will be incorporated.

A 7-segment-display will be used to guide the user during the process (ex: Start, Full, Empty, Machine Number)

Each washing machine will be locked by a password the user enters at the beginning.

Now to check which washing machines are available we are going to use the memory, where the system will have the ability to count the number of used washing machines.

States

- no_action_state

- user_detect_state
  This state detects if a user near the controller via sensor.

- verification_state
  This state comes after user detected state where the user has been detected and the user will need to enter a password. Then verify the password will be handled here and if the password is correct the used washing machine number will be displayed on the 7-segement-display.

- availability_state
  A memory check will be proceeded here to check if the entered washing machine number to be used is available or not.

- payment_state
  Now the user can pay for the wash using 50, 20, 10 and 5 cents.

- process_state
  Checking if the washing machine finished the work.

- finished_processing_state
  After the process has been finished will be moving to the start state.

- hold_state
  Where the process still in progress this state going to be the current state.

- start_state
  The user ready to use the washing machine.

- full_state
  Will display full sign on the 7-segment-display screen.

Testing Functionality

So, after we implemented the code for our system, we had to do the self-checking testbench which is used to fully functional verification step for the correct behavior of the circuit.

```
  begin
    -- procedure calls here


    P_sync_app(1);
    P_nearby_person('1'); -- sensor detect a person -> change state to 1, PASS displayed
    P_sync_app(3);
    P_set_password("0110"); -- entering wrong password
    P_sync_app(3);
    P_password_button; -- wrong password -> state does not change
    P_sync_app(3);
    P_set_password("0000"); -- valid password
    P_sync_app(3);
    P_password_button; -- valid password -> change state to 2, Addr displayed
    P_sync_app(3);
    P_adress_button; --adress button enabled -> change state to 3, PAy displayed. If the machine is available -> change state to
    P_sync_app(3);
    P_pay("1000"); -- pay 1 euro, the total payment is displayed 100
    P_sync_app(3);
    P_pay("0100"); -- pay 0.5 euro, the total ayment is displayed 150
    P_sync_app(3);
    P_pay("1000"); -- pay 1 euro, the total ayment is displayed 250, payment is done and coins_out set to 0100 (0.5 euro)
              --  change state to 5, Procs displayed
    P_sync_app(3);
    P_nearby_person('0'); -- no persen detected in front of the system sensor, the person moved to the selected washing machine
    P_sync_app(3); -- processing is not done -> change state to 7, Hold displayed
    P_processing_is_done; -- processing is done -> change state to 6, display start
    P_sync_app(3);
    P_start_washing; --washing started, -> change to state 8, increment the count of working washing machines
    P_sync_app(10); --return to state 0

    -- If we run the same simulation code (copying above code multiple times) multiple times, in the 9th time the state will be
    -- 9 represents the full_state, since our system only contains 9 washing machines.

    assert false report "--- END OF SIMULATION ---"
      severity failure;

  end process p_stimulus;
```

Figure 2: e_my_automated_laundry_system.vht (Simulations Scenario)

```
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_7seg_display.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_7seg_bcd_decoder.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_laundry_fsm.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_memory.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_payment_fsm.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/e_my_automated_laundry_system.vhd
vcom -reportprogress 300 -work work C:/intelFPGA_lite/18.1/FPGA_Excercises/project/simulation/modelsim/e_my_automated_laundry_sy
vsim work.e_my_automated_laundry_system_vhd_tst
do wave.do
run -all
```

Figure 3: run.do

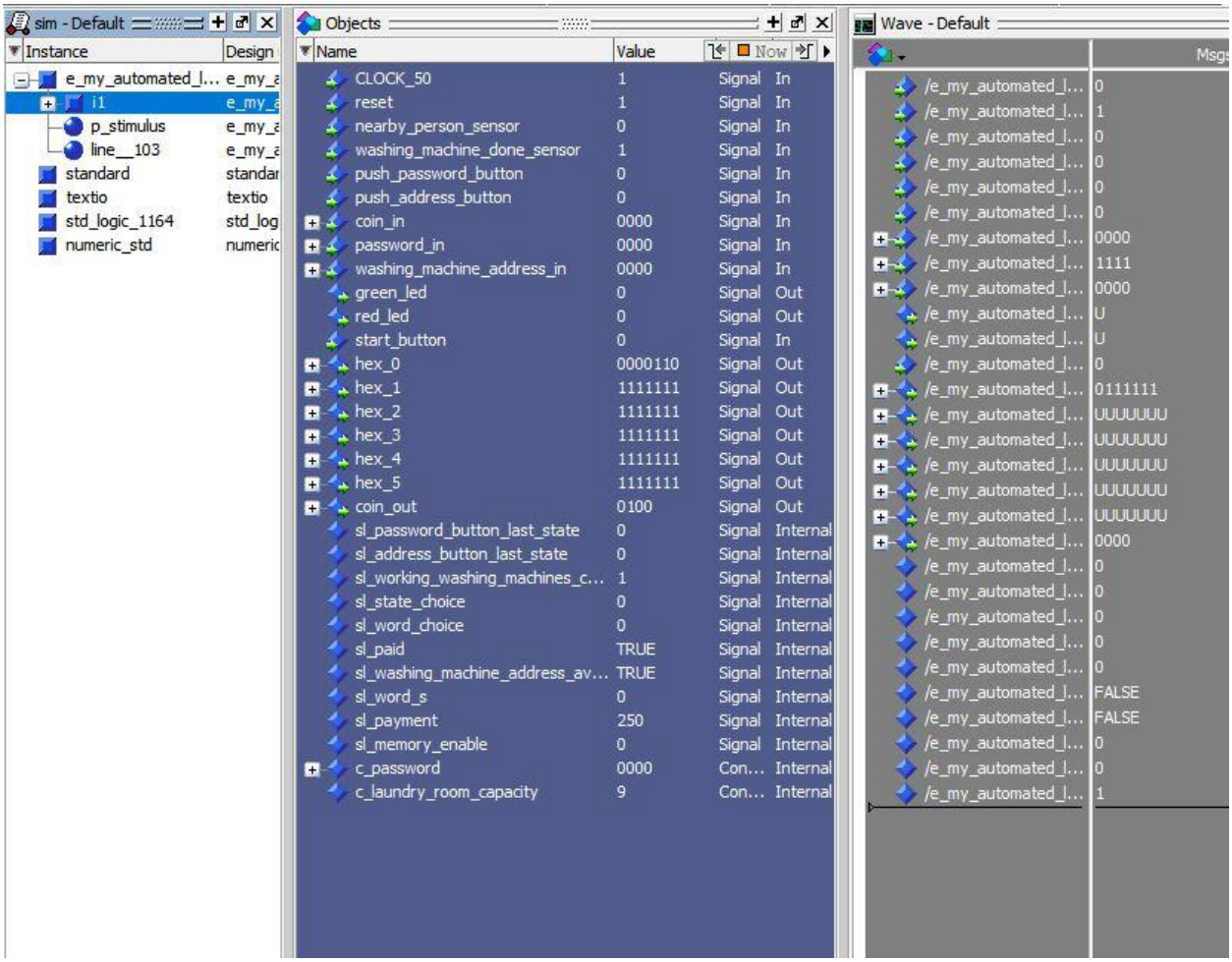

Figure 4: RTL Simulation

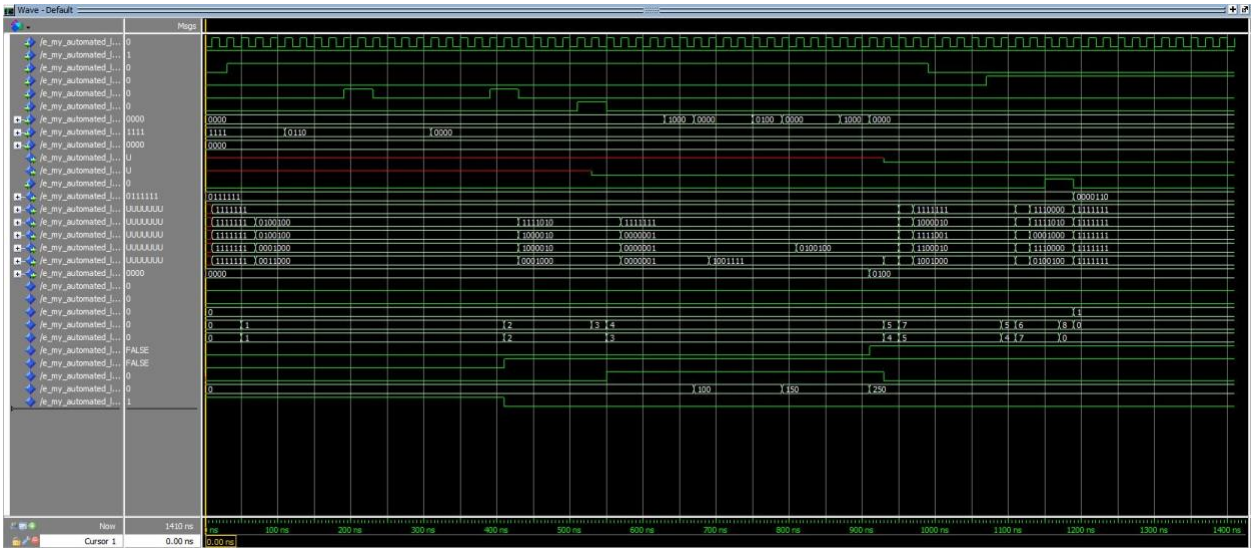

Figure 5: RTL Simulation Results

Time Analyzer

By running the report timing for the default clocks, we will get the following results

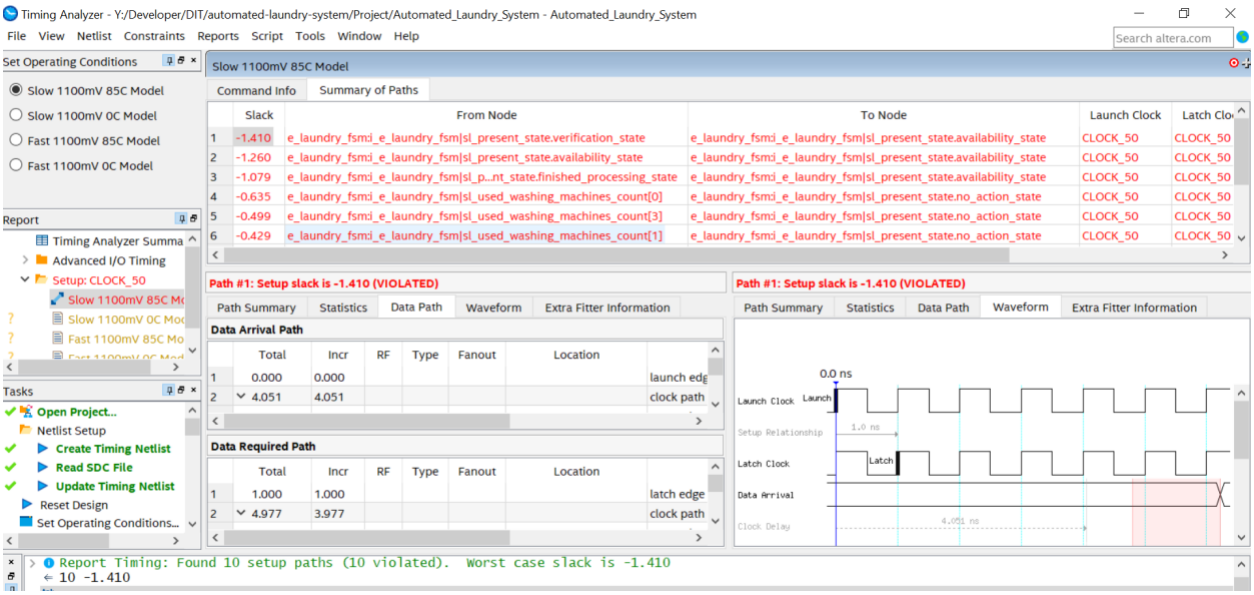

Figure 6: Quartus - Time Quest Analyzer

Then we did calculate the resulted values

Clock-skew, the delay between launch clock delay and latch clock delay

$$t_{skew} = 4.051\ ns - 4.004\ ns$$
$$= 0.047\ ns$$

Final clock skew, we add the clock pessimism

$$= 0.047\ ns + 0.204\ ns$$
$$= 0.251\ ns$$

Slack, difference between required arrival time and actual arrival time

$$t_{slack} = 1\ ns - 4.432\ ns + t_{skew}$$
$$= 1\ ns - 4.432\ ns + 0.251\ ns$$
$$= -3.181\ ns$$

Clock uncertainty

$$t_{slack} = -3.181\ ns + (-0.03\ ns)$$
$$= -3.211\ ns$$

uTsu

$$t_{slack} = -3.211\ ns - (-0.024\ ns)$$
$$= -3.187\ ns$$

We set the new values to constraint the input clock signal

Clocks Summary

	Clock Name	Type	Period	Frequency	Rise	Fall
1	CLOCK_50	Base	4.000	250.0 MHz	0.000	2.000

Figure 7: Quartus - Time Quest Analyzer - Frequency

By regenerating the results will be as following

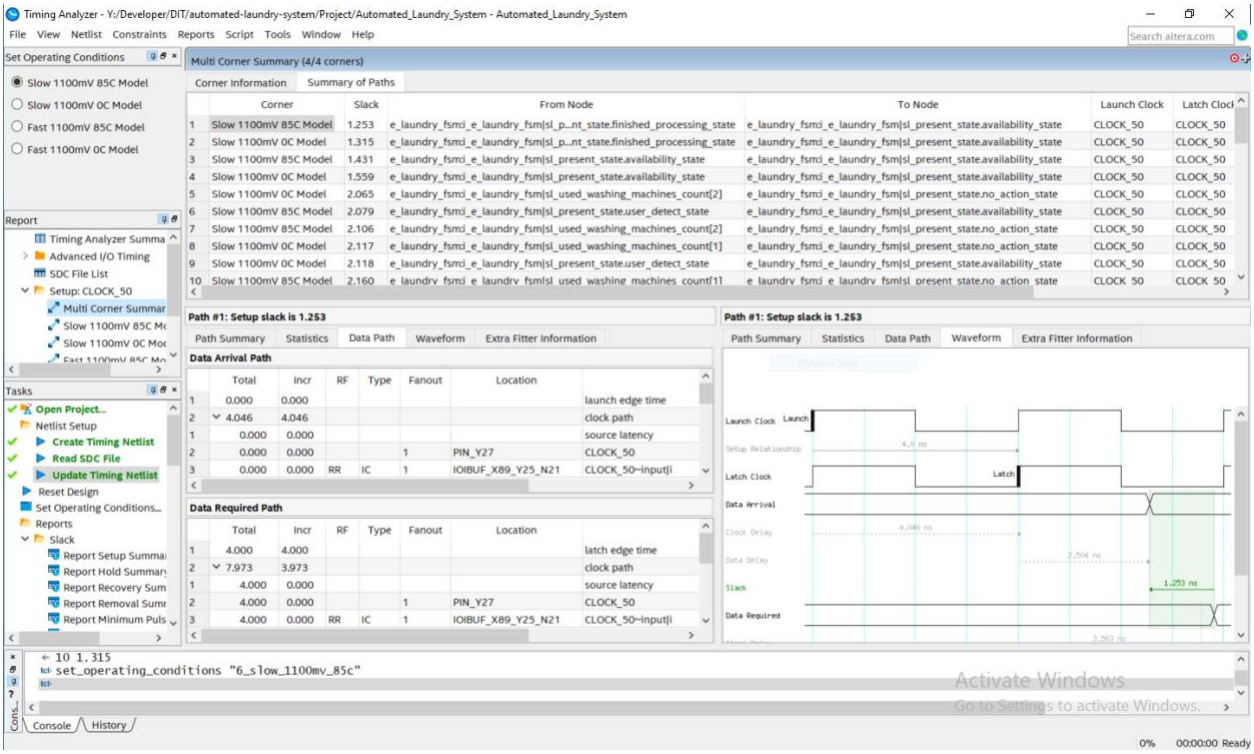

Figure 8: Quartus - Time Quest Analyzer

Algorithmic State Machine (ASM)

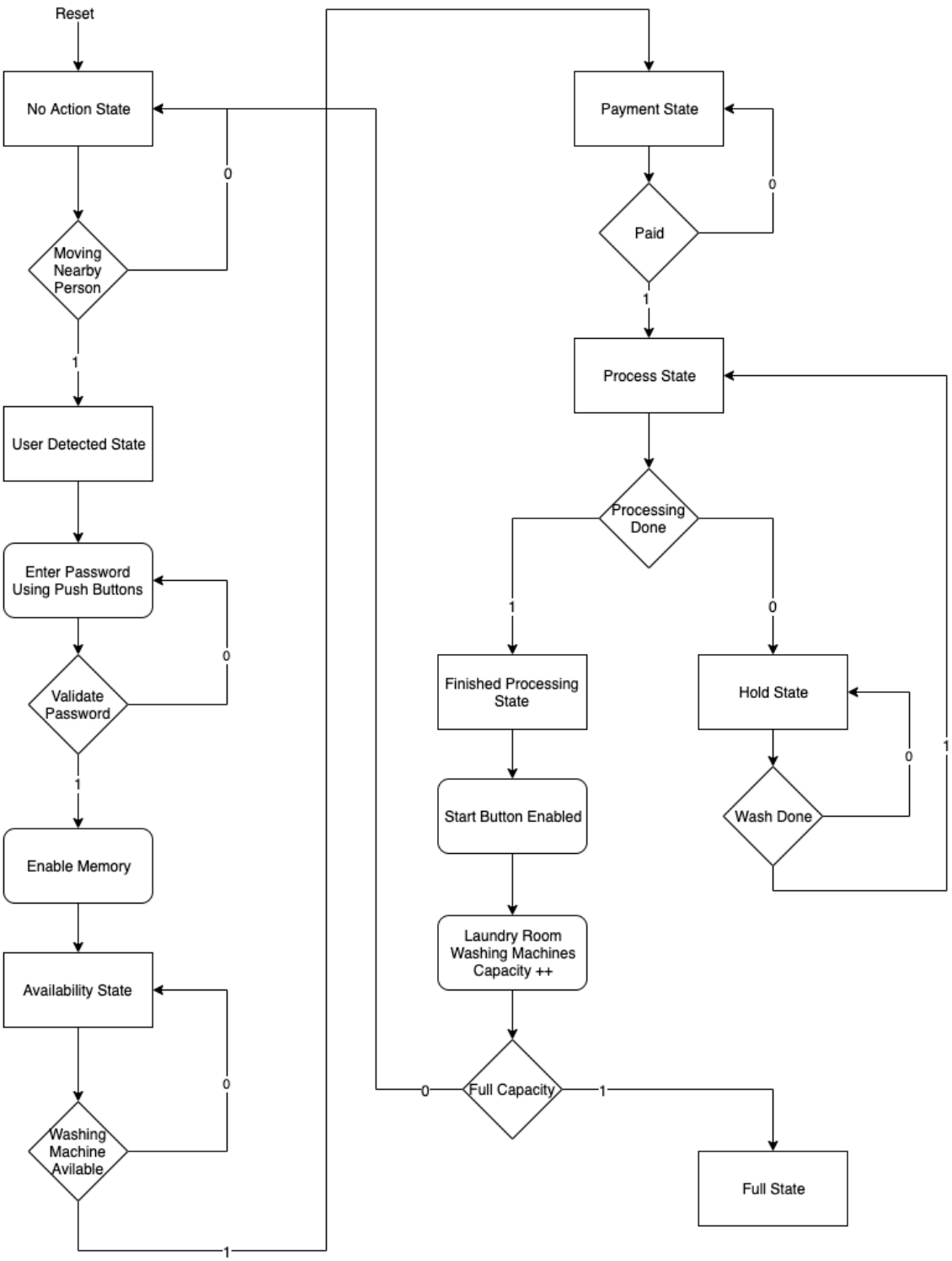

Figure 9: Algorithmic State Machine (ASM) Chart for the Circuit

Block Diagram

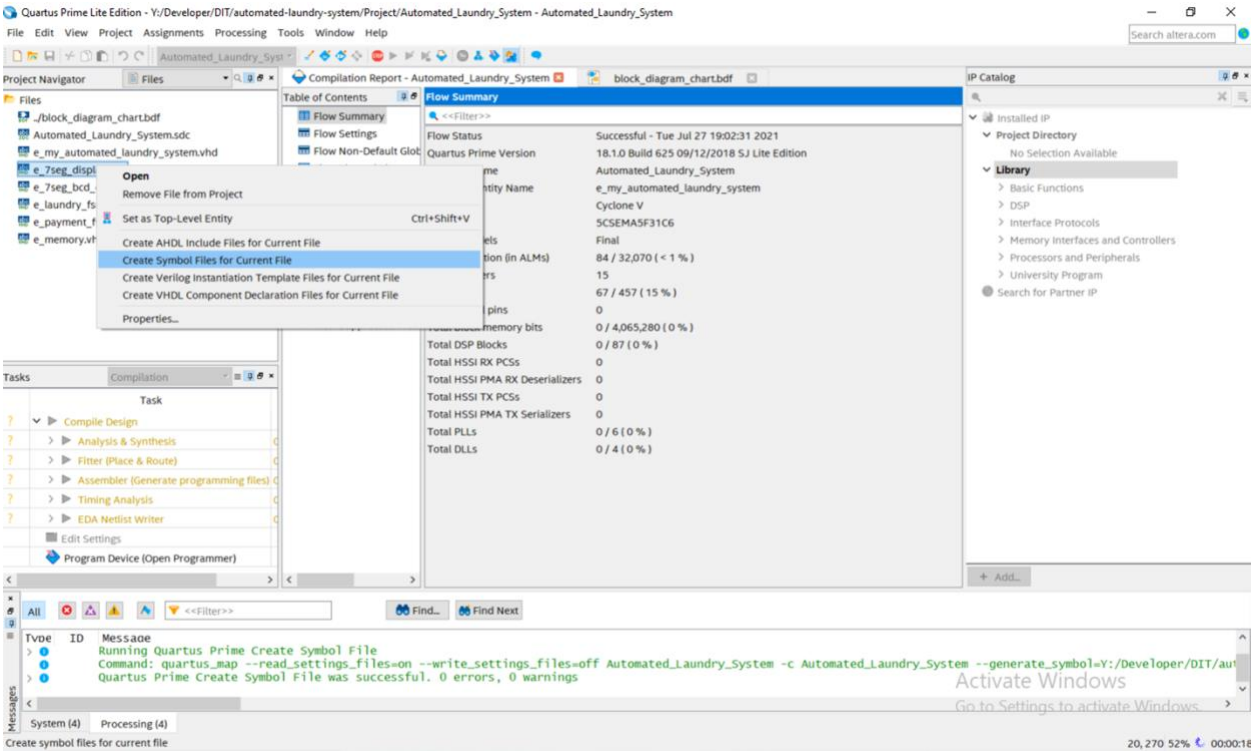

Figure 10: Creating Symbol Files

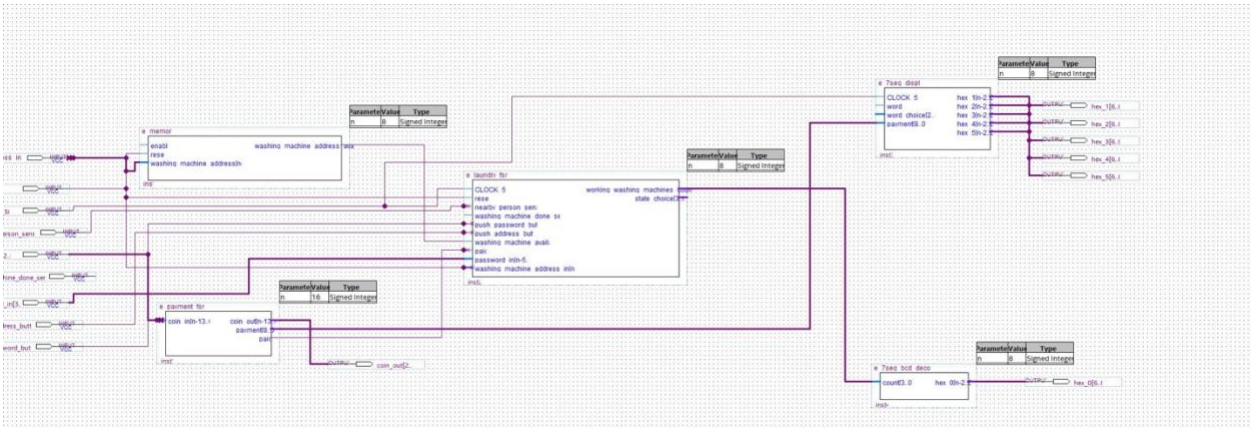

Figure 11: Block Diagram Chart