# ASIC Design of ATM Controller

Presentation | 16th April, 2025

## Presenters:

Sumit Vishwakarma - 202201320

Tirh Modi - 202201513

Mentor: Dr. Rutu Parekh

## Abstract

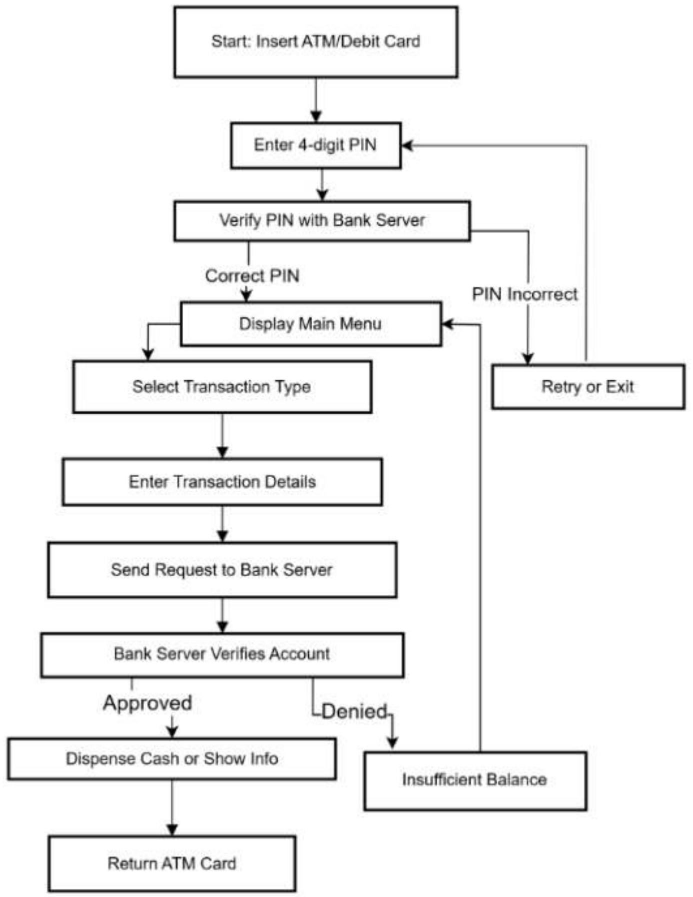
Our report demonstrates the working of the controller of ATM machine with its core logic explained through a Finite State Machine Diagram.

Initially, we understood the flow of operations taking place in an ATM Machine,, with the help of a flowchart.

Secondly, we tried to model an FSM diagram after understanding the key states and transitions.

Then, we wrote a Verilog code to test and verify its functionality .

At last, we synthesized the Verilog code with the help of Q-flow digital synthesis tool.


Start: Insert ATM/Debit Card
Enter 4-digit PIN
Verify PIN with Bank Server
Correct PIN
PIN Incorrect
Display Main Menu
Retry or Exit
Select Transaction Type
Enter Transaction Details
Send Request to Bank Server
Bank Server Verifies Account
Approved
Denied
Dispense Cash or Show Info
Insufficient Balance
Return ATM Card

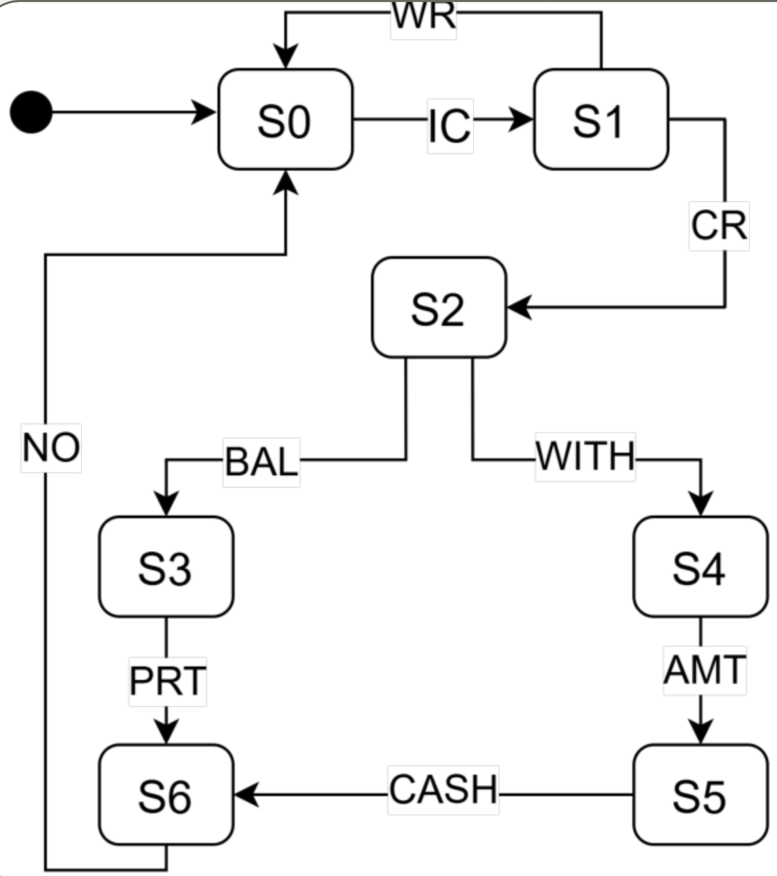
# Flowchart

To model an FSM diagram, firstly, we understood the active states, transitions and flow of operations taking place in an ATM machine. Based on that, we further modelled an FSM diagram.

We identified some key transitions happening across common states.

We then arranged the states and their respective transitions based on their order of occurrence.

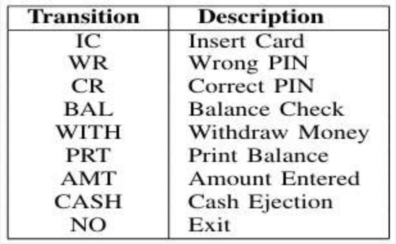
This was a crucial step for modelling our FSM



# FSM Diagram

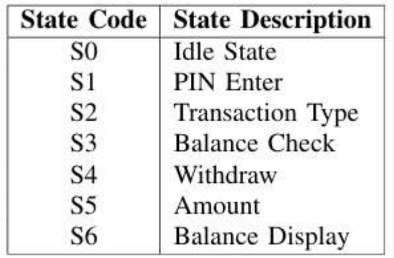
The following tables define the states present in the FSM diagram along with the transitions occurring between multiple states.

## Transition Description

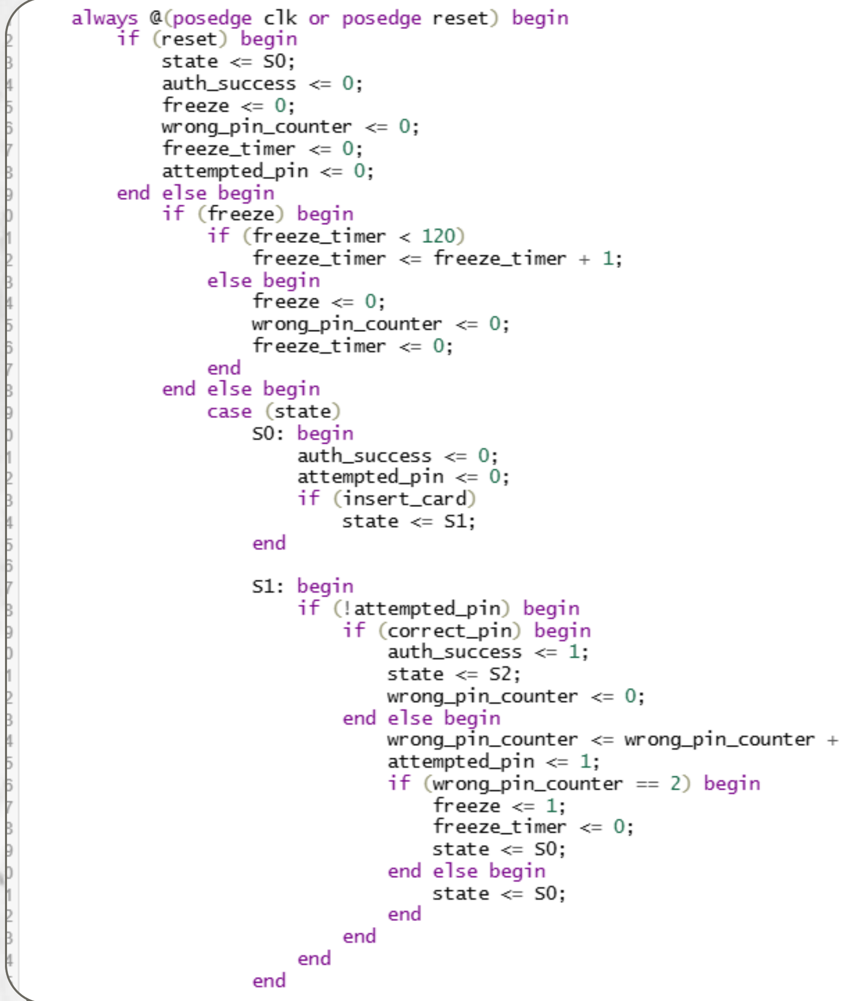


| Transition | Description |
|---|---|
| IC | Insert Card |
| WR | Wrong PIN |
| CR | Correct PIN |
| BAL | Balance Check |
| WITH | Withdraw Money |
| PRT | Print Balance |
| AMT | Amount Entered |
| CASH | Cash Ejection |
| NO | Exit |

## State Description:



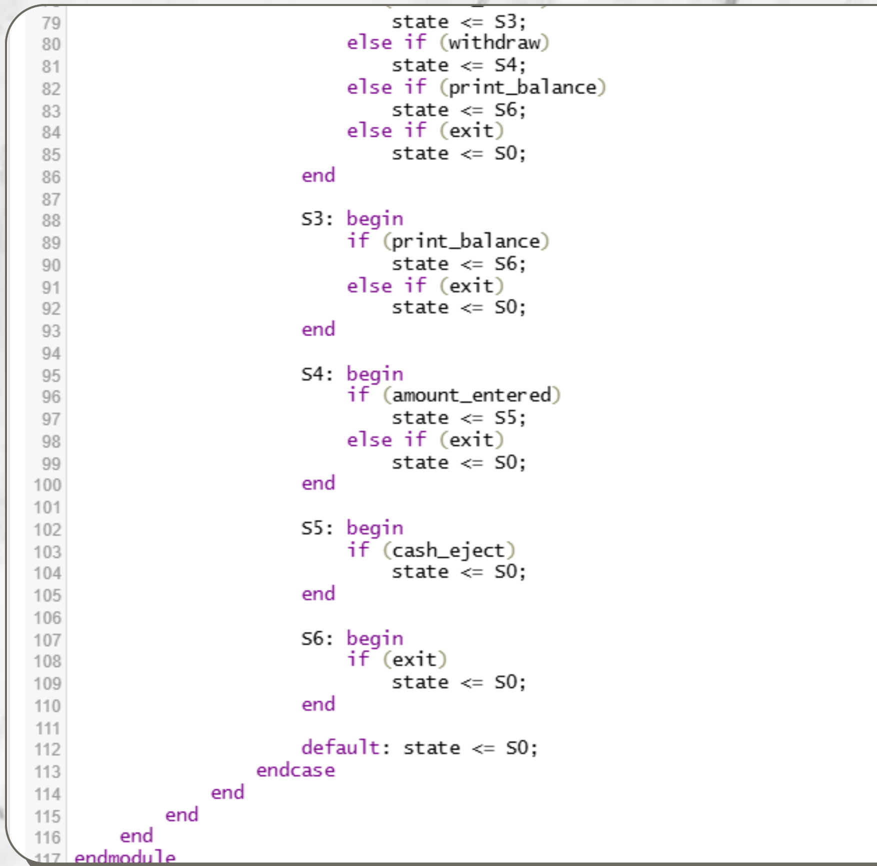
| State Code | State Description |
|---|---|
| S0 | Idle State |
| S1 | PIN Enter |
| S2 | Transaction Type |
| S3 | Balance Check |
| S4 | Withdraw |
| S5 | Amount |
| S6 | Balance Display |



```verilog
always @(posedge clk or posedge reset) begin
    if (reset) begin
        state <= S0;
        auth_success <= 0;
        freeze <= 0;
        wrong_pin_counter <= 0;
        freeze_timer <= 0;
        attempted_pin <= 0;
    end else begin
        if (freeze) begin
            if (freeze_timer < 120)
                freeze_timer <= freeze_timer + 1;
            else begin
                freeze <= 0;
                wrong_pin_counter <= 0;
                freeze_timer <= 0;
            end
        end else begin
            case (state)
                S0: begin
                    auth_success <= 0;
                    attempted_pin <= 0;
                    if (insert_card)
                        state <= S1;
                end

                S1: begin
                    if (!attempted_pin) begin
                        if (correct_pin) begin
                            auth_success <= 1;
                            state <= S2;
                            wrong_pin_counter <= 0;
                        end else begin
                            wrong_pin_counter <= wrong_pin_counter +
                            attempted_pin <= 1;
                            if (wrong_pin_counter == 2) begin
                                freeze <= 1;
                                freeze_timer <= 0;
                                state <= S0;
                            end else begin
                                state <= S0;
                            end
                        end
                    end
                end
```

# RTL Implementation

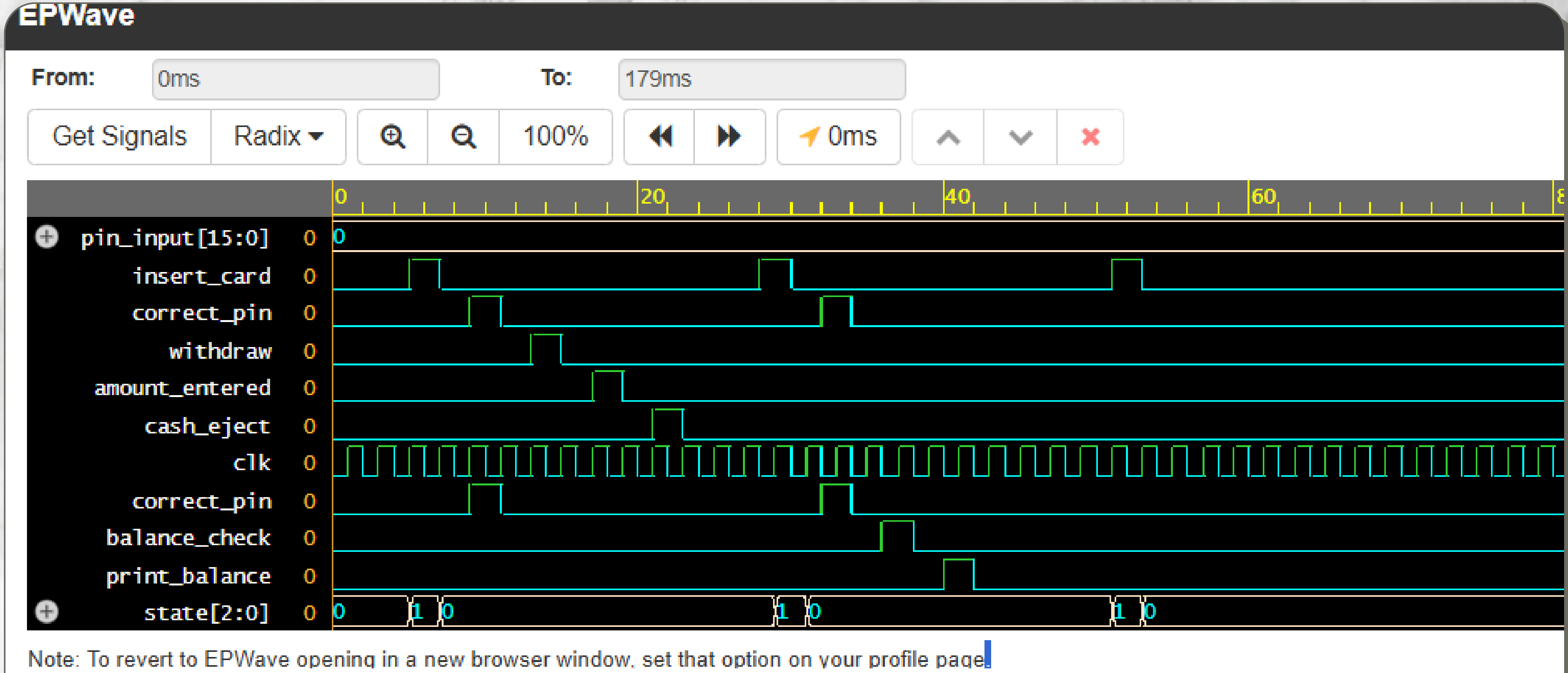
# Verilog Code



```verilog
                            state <= S3;
                        else if (withdraw)
                            state <= S4;
                        else if (print_balance)
                            state <= S6;
                        else if (exit)
                            state <= S0;
                    end

                    S3: begin
                        if (print_balance)
                            state <= S6;
                        else if (exit)
                            state <= S0;
                    end

                    S4: begin
                        if (amount_entered)
                            state <= S5;
                        else if (exit)
                            state <= S0;
                    end

                    S5: begin
                        if (cash_eject)
                            state <= S0;
                    end

                    S6: begin
                        if (exit)
                            state <= S0;
                    end

                    default: state <= S0;
                endcase
            end
        end
    end
endmodule
```

# RTL Implementation

# Verilog Code



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

## Simulation Output :

This Slide shows the observed simulation result and the triggering of each states based on user actions.

## Q-Flow Completion

1 **Synthesis:**

Converts the written logic code into the gate level netlist.

2 **Placement:**

Assignment of gate level connections to a physical position.

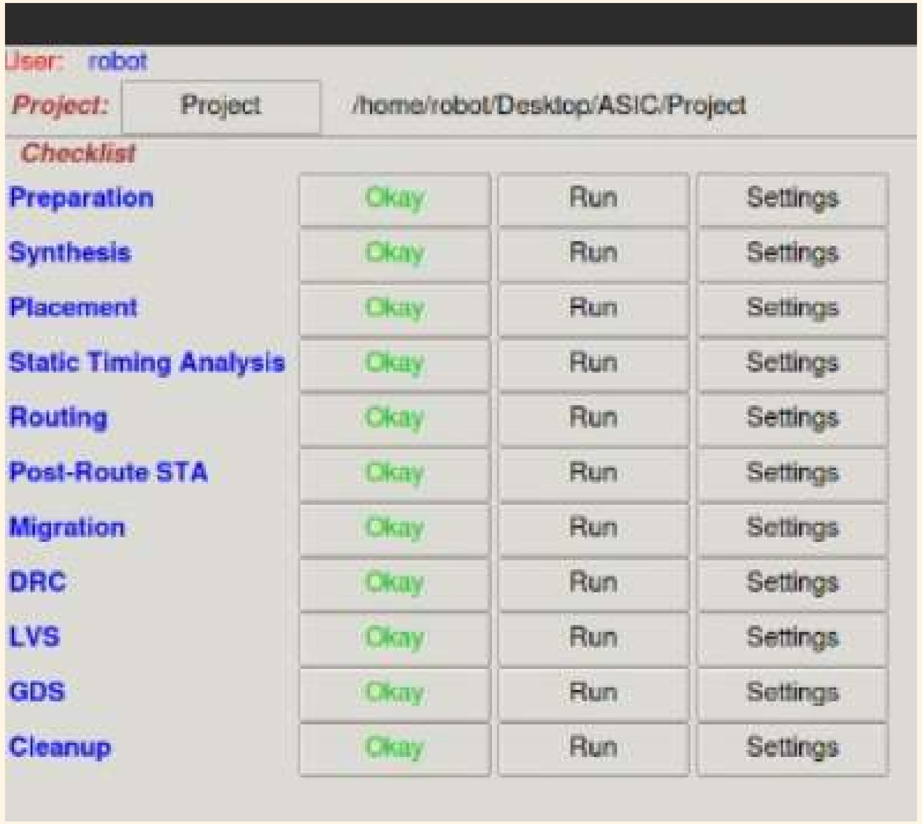
3 **STA:**

Verifies the timing constraints and correctness without running any kind of simulation.

4 **Routing**

Interconnect all these cells

using metal wires.

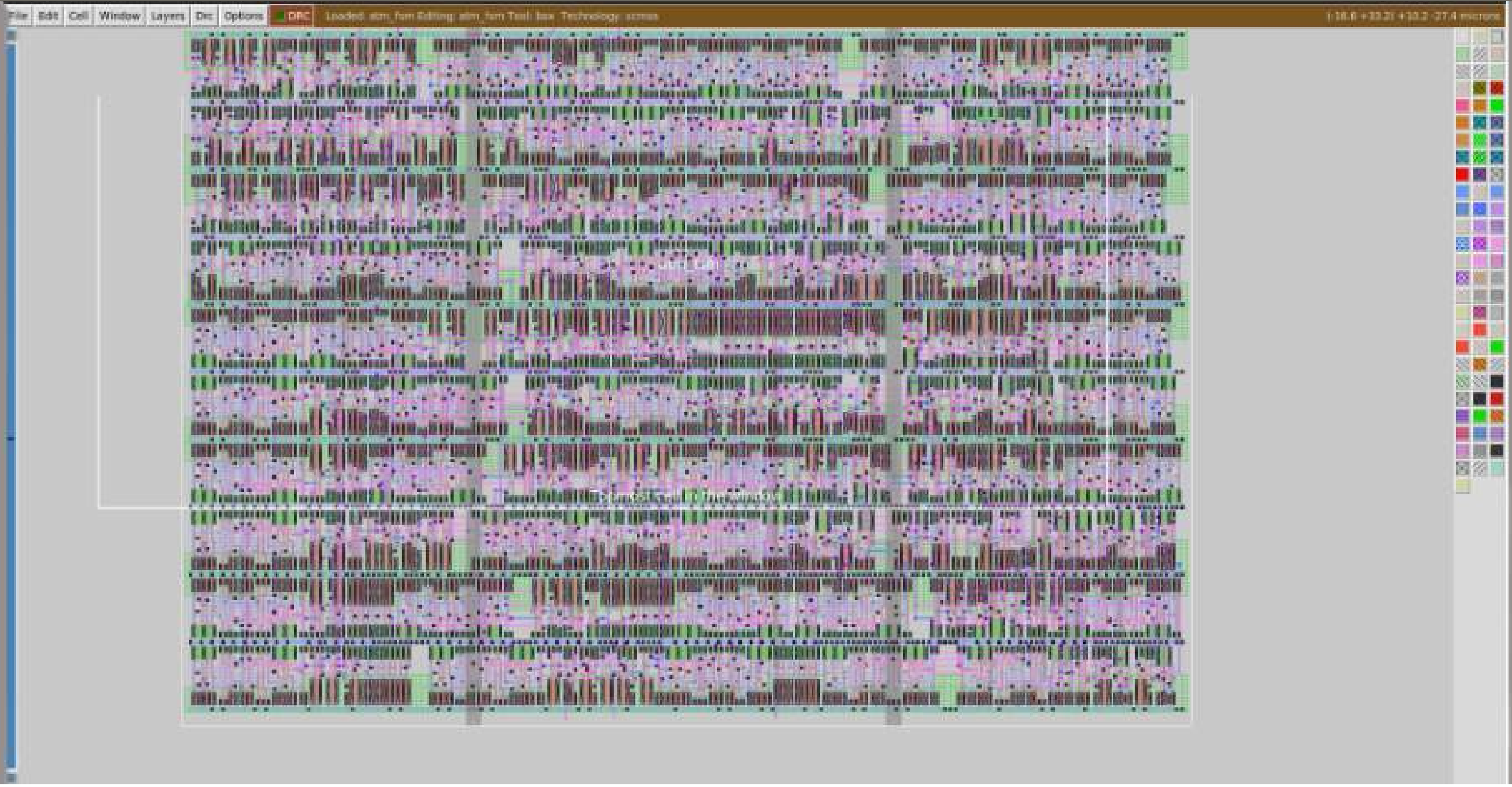


User: robot

Project: Project /home/robot/Desktop/ASIC/Project

Checklist

| | | | |
|---|---|---|---|
| Preparation | Okay | Run | Settings |
| Synthesis | Okay | Run | Settings |
| Placement | Okay | Run | Settings |
| Static Timing Analysis | Okay | Run | Settings |
| Routing | Okay | Run | Settings |
| Post-Route STA | Okay | Run | Settings |
| Migration | Okay | Run | Settings |
| DRC | Okay | Run | Settings |
| LVS | Okay | Run | Settings |
| GDS | Okay | Run | Settings |
| Cleanup | Okay | Run | Settings |


File
Edit
Cell
Window
Layers
Drc
Options
DRC
Loaded: alm_fsm Editing: alm_fsm Tool: box Technology: scmos
(-16.6 +33.2) +10.2 -27.4 microns

# Layout

# Conclusion

Through this Project, we tried to model a controller for an ATM machine.

1. RTL Code

   This manages the state transitions triggered by the user actions on the ATM Machine.

2. Simulation

   We tested thoroughly our design through simulations and after running multiple simulations, we ensured the correct working of our code.

3. Synthesis

   Performed using open source tool, Q-Flow and generated the layout.