

# INTEL UNNATI INDUSTRIAL TRAINING-SUMMER 2023

## Design and Implementation of Any Time Electricity Bill Payment (ATP) Machine Controller

### Team Spark

*Name 1:* Surepally Vishnnu Vardhini

*Name 2:* Aditya Kumar Sharma

*Institution:* Nitte Meenakshi Institute of Technology, Bangalore.

#### *Institution Mentor:*

Dr. Shashidhara K S, Professor, Dept of ECE.

#### *Industry Mentor:*

Mr. Abhishek Nandy, Intel Unnati.

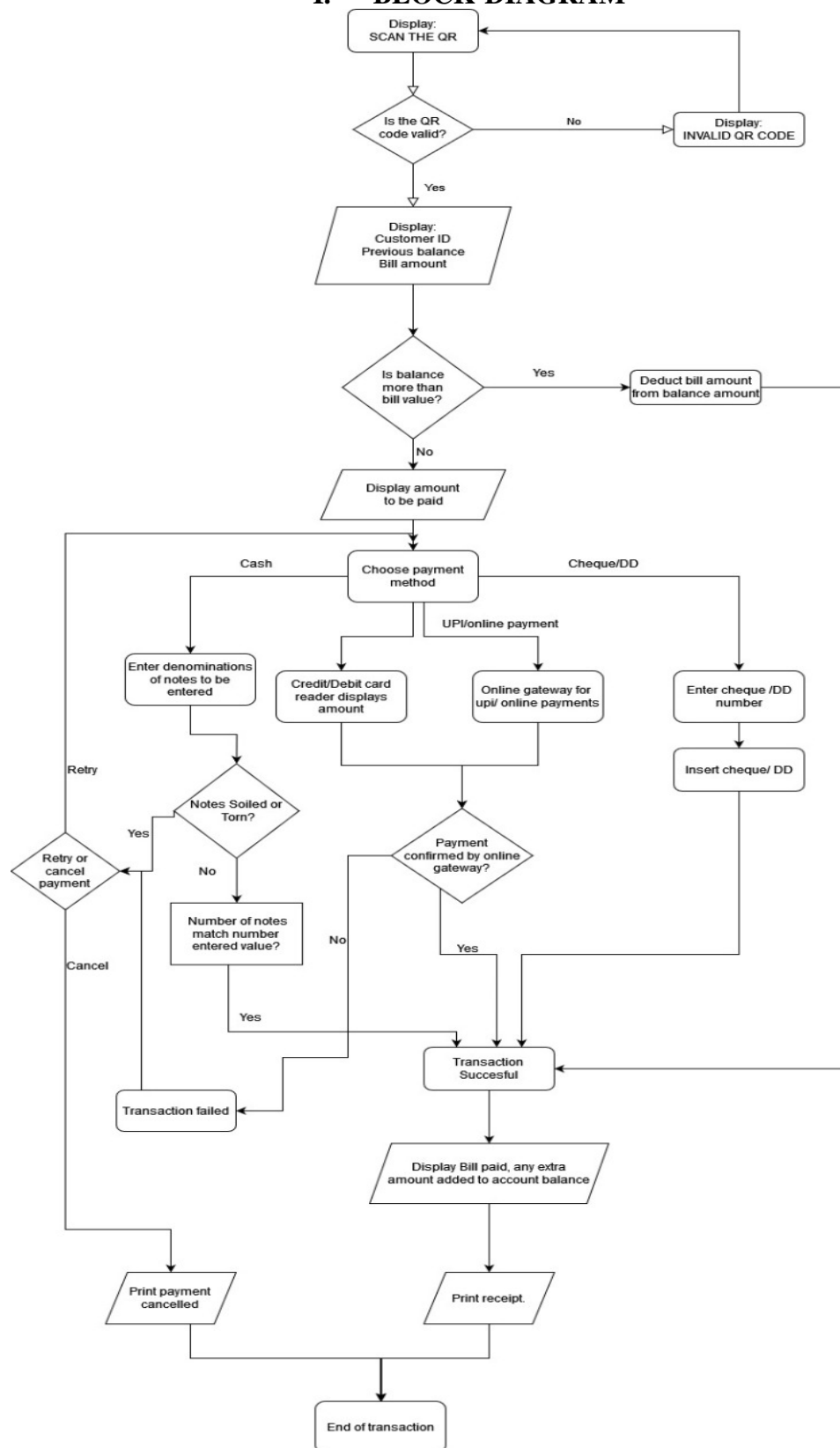
## Contents

### Design and Implementation of Any Time Electricity Bill Payment (ATP)

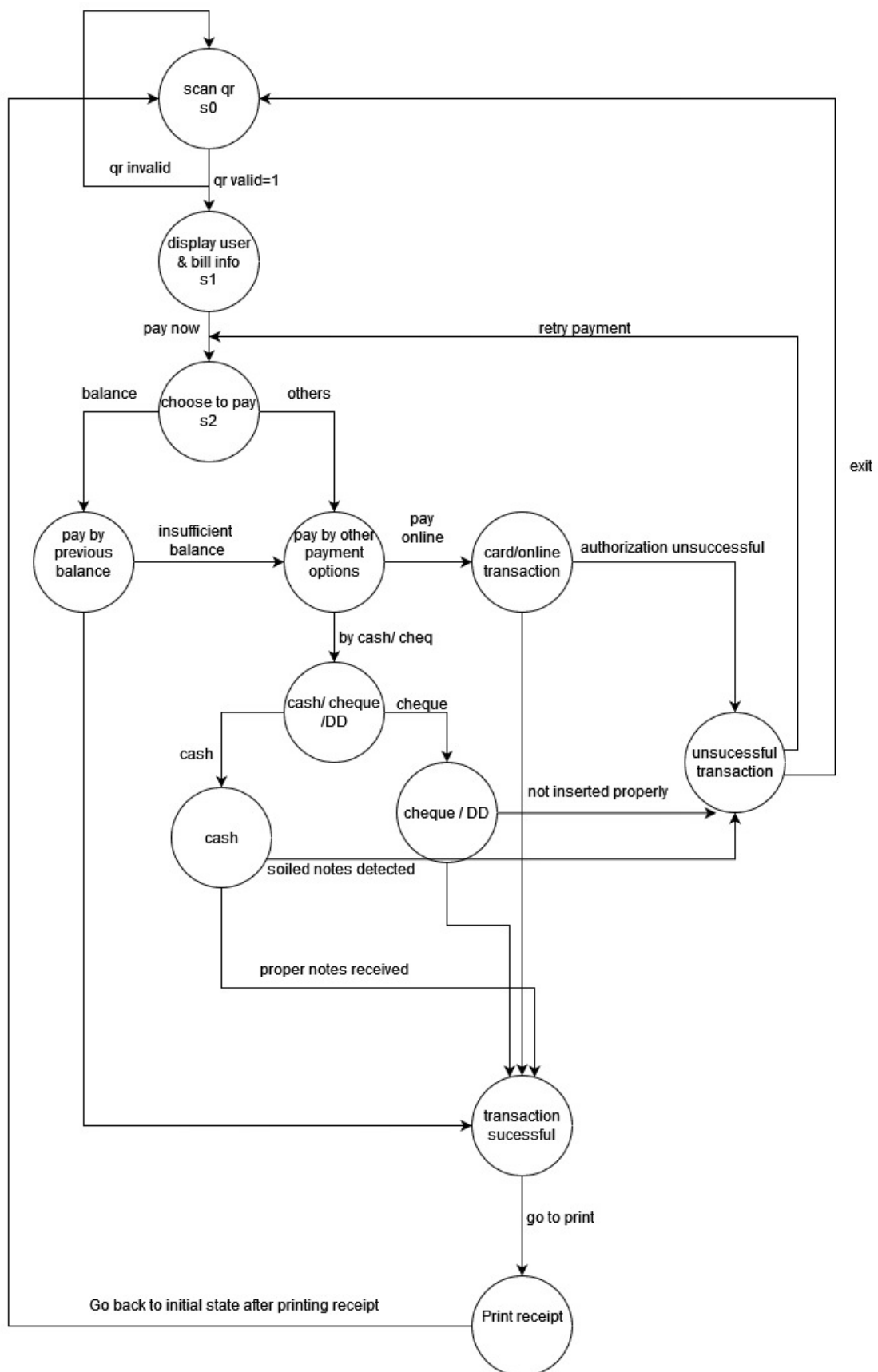
Machine Controller .....	1
I. BLOCK DIAGRAM .....	3
II. FSM .....	4
III. APPROACH.....	5
IV. METHODOLOGY/FLOW.....	5
RTL View: .....	13
Error: .....	13
V. GITHUB FILE SCREENSHOT .....	14
VI. RESULTS AND SUMMARIZATIONS .....	14
VII. REFERENCES .....	14

In this project, we have tried to design and implement an Any Time Electricity Bill Payment (ATP) Machine Controller using FPGA.

## I. BLOCK DIAGRAM



## II. FSM



### III. APPROACH

1. In order to solve this problem, we drew a flowchart first which roughly summarizes our agenda and gives us a brief on methodology and how to work on it.
2. Users can pay through existing Balance, Cash, Card, Online Transactions, DD and Cheque.
3. The machine controller uses QR Scanner in order to initiate the process.
4. After the QR gets scanned, the machine controller displays Customer ID, Existing Balance, Amount to be paid and Pay option.
5. If Balance is greater than the Bill amount then the amount gets deducted automatically else the machine controller moves further and displays other payment options.
6. The machine controller displays QR for UPI payments, redirects to Online Gateway for Online payments, displays Insert DD or Cheque option for DD or Cheque and displays Insert Cash option for Cash mode of payment.
7. The machine controller does not accept soiled or torn notes.
8. If the transaction fails then the machine controller redirects to Retry Payment option.
9. If the transaction is successful then the machine controller displays Bill paid successfully and Extra amount to be added option.
10. If the user wants to add balance, then the machine controller redirects the user to choose payment mode option else the receipt is printed and transaction ends.

### IV. METHODOLOGY/FLOW

1. Intel Quartus Prime Lite Software has been used to obtain results.
2. A Verilog HDL code and the test bench has been compiled successfully without errors to obtain the State Diagram of the given problem statement using block diagram as reference as shown in figure below.

#### *Verilog code:*

```
module atp (output reg [3:0] data_out, input in0, in1, in2, in3, in4, in5, in6, in7, in8, in9, in10, in11, input clk, rst);
```

```
parameter s0= 4'b0000; //scanqr
```

```
parameter s1= 4'b0001; //data display
```

```
parameter s2= 4'b0010; //balance or oth
```

```

parameter s3= 4'b0011; // pay by bal
parameter s4= 4'b0100; // pay by oths
parameter s5= 4'b0101; //pay caash or dd or cheq
parameter s6= 4'b0110; // card or upi
parameter s7= 4'b0111; //cash
parameter s8= 4'b1000; //dd/chq
parameter s9= 4'b1001; //FAIL
parameter s10=4'b1010; // success
parameter s11=4'b1011; // print receipt

```

```

reg [1:0] c_state,n_state;

```

```

always @(posedge clk)

```

```

begin

```

```

if (rst==1)

```

```

begin

```

```

    c_state=0;

```

```

        n_state=0;

```

```

        end

```

```

else

```

```

    c_state=n_state;

```

```

case(c_state)

```

```

s0: begin

```

```

if (in0==0) n_state = s0;

```

```

else if ( in0==1) n_state = s1;

```

```

end

```

```

s1: begin

```

```
if (in1==0) n_state = s0;  
else if ( in1==1) n_state = s2;  
end
```

```
s2: begin  
if ( in2==0) n_state = s3;  
  
else if ( in2==1) n_state = s4;  
  
end
```

```
s3: begin  
if (in3==0) n_state = s4;  
else if ( in3==1) n_state = s10;  
end
```

```
s4: begin  
if (in4==0) n_state = s5;  
  
else if (in4==1)n_state = s6;  
  
end
```

```
s5: begin  
if (in5==0) n_state = s7;  
else if (in5==1)n_state = s8;  
end
```

```
s6: begin  
if (in6==0) n_state = s9;  
else if (in6==1)n_state = s10;
```

```

end
s7: begin
if (in7==0) n_state = s9;
else if (in7==1)n_state = s10;
end
s8: begin
if (in8==0) n_state = s9;
else if (in8==1)n_state = s10;
end
s9: begin
if (in9==0) n_state = s4;
else if (in9==1)n_state = s0;
end
s10: begin
if (in10==0) n_state = s11;
else if (in10==1)n_state = s11;
end
s11: begin
if (in11==0) n_state = s0;
else if (in11==1)n_state = s0;
end
endcase
end
endmodule

```



### ***Verilog testbench:***

```
module atp_tb;

    reg in0, in1, in2, in3, in4, in5, in6, in7,in8, in9, in10,in11;
    reg clk, rst;
    wire [3:0] data_out;

    atp (
        .data_out(data_out),
        .in0(in0),
        .in1(in1),
        .in2(in2),
        .in3(in3),
        .in4(in4),
        .in5(in5),
        .in6(in6),
        .in7(in7),
            .in8(in8),
            .in9(in9),
            .in10(in10),
            .in11(in11),
        .clk(clk),
        .rst(rst)
    );

    // Clock generation
    always #5 clk = ~clk;

    // Test stimulus
    initial begin
        // Initialize inputs
```

```
in0 = 0;
in1 = 0;
in2 = 0;
in3 = 0;
in4 = 0;
in5 = 0;
in6 = 0;
in7 = 0;
rst = 1;
clk = 0;
```

```
// Reset sequence
```

```
#10 rst = 0;
#20 rst = 1;
```

```
// Test case 1: Transition from S0 to S1
```

```
#10 in0 = 1;
#10 in1 = 1;
#10 in0 = 0;
#10 in1 = 0;
```

```
// Test case 2: Transition from S1 to S2
```

```
#10 in1 = 1;
#10 in2 = 1;
#10 in1 = 0;
#10 in2 = 0;
//3
#10 in2 = 1;
#10 in3 = 1;
#10 in2 = 0;
```

```
#10 in3 = 0;
//4
#10 in3 = 1;
#10 in4 = 1;
#10 in3 = 0;
#10 in4 = 0;
//5
#10 in4 = 1;
#10 in5 = 1;
#10 in4 = 0;
#10 in5 = 0;
//6
#10 in5 = 1;
#10 in6 = 1;
#10 in5 = 0;
#10 in6 = 0;
//7
#10 in6 = 1;
#10 in7 = 1;
#10 in6 = 0;
#10 in7 = 0;
//8
#10 in7 = 1;
#10 in8 = 1;
#10 in7 = 0;
#10 in8 = 0;
//9
#10 in8 = 1;
#10 in9 = 1;
#10 in8 = 0;
```

```

#10 in9 = 0;

//10

#10 in9 = 1;

#10 in10 = 1;

#10 in9= 0;

#10 in10 = 0;

//11

#10 in10= 1;

#10 in11= 1;

#10 in10 = 0;

#10 in11= 0;


// End simulation

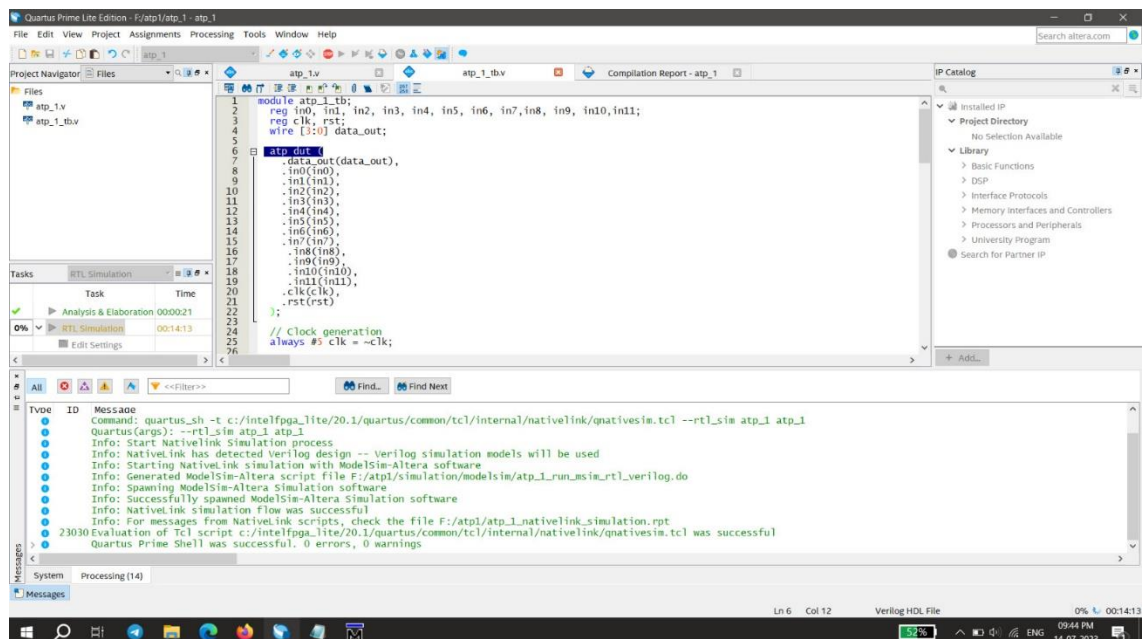
#10 $finish;

end

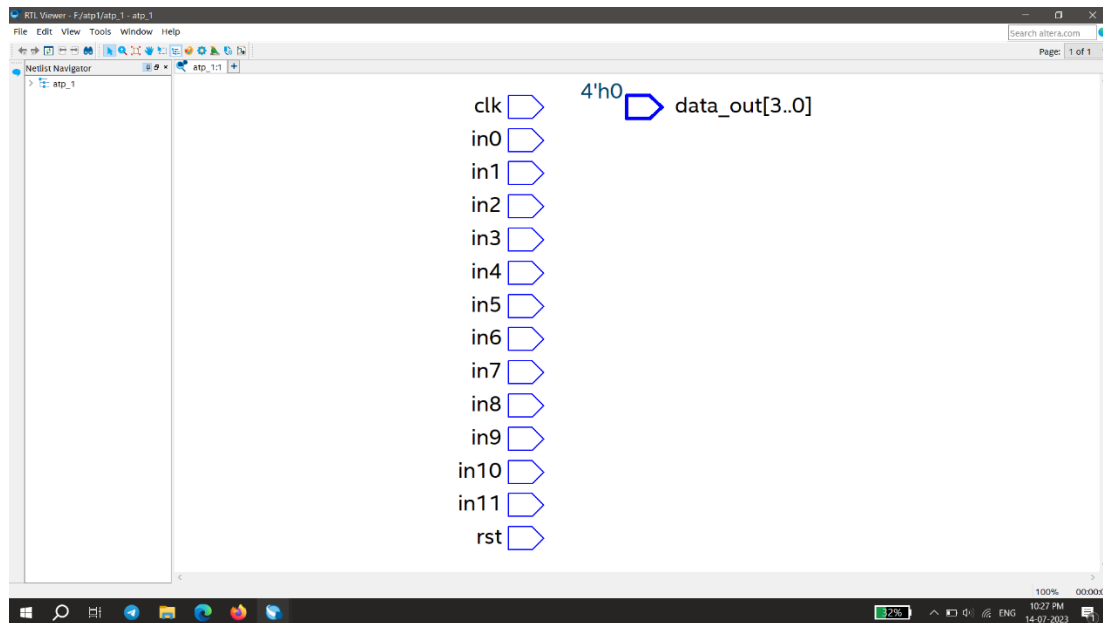
endmodule

```

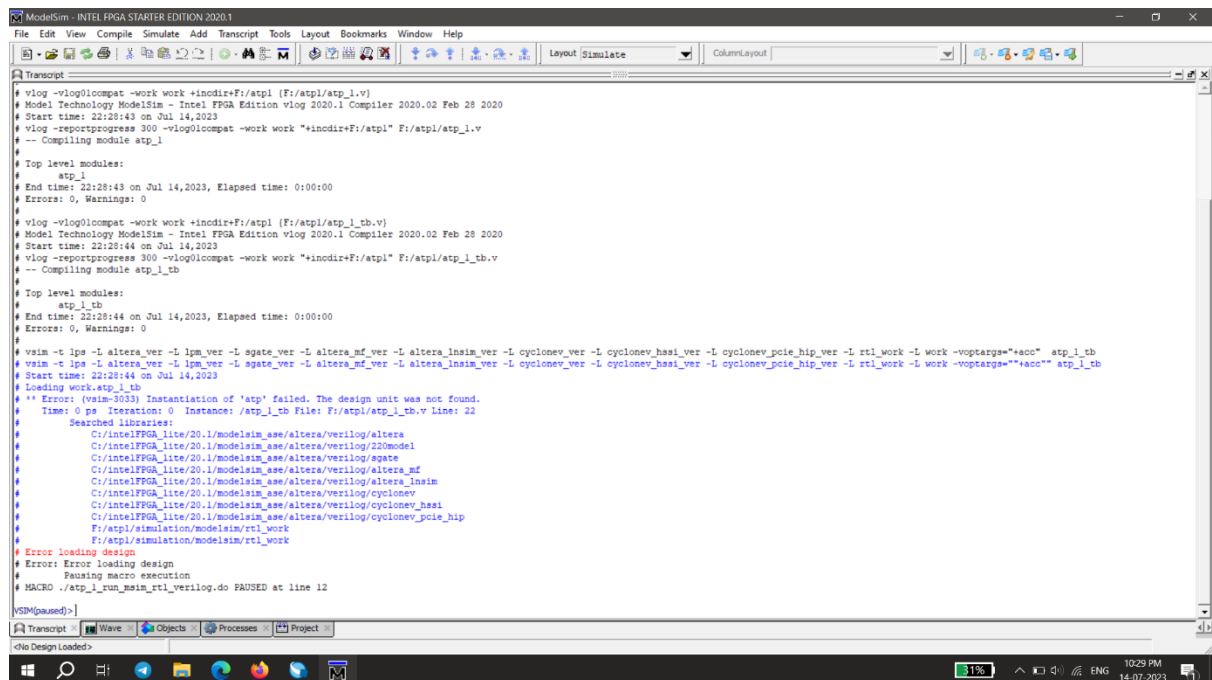
## Compilation output



## RTL View:



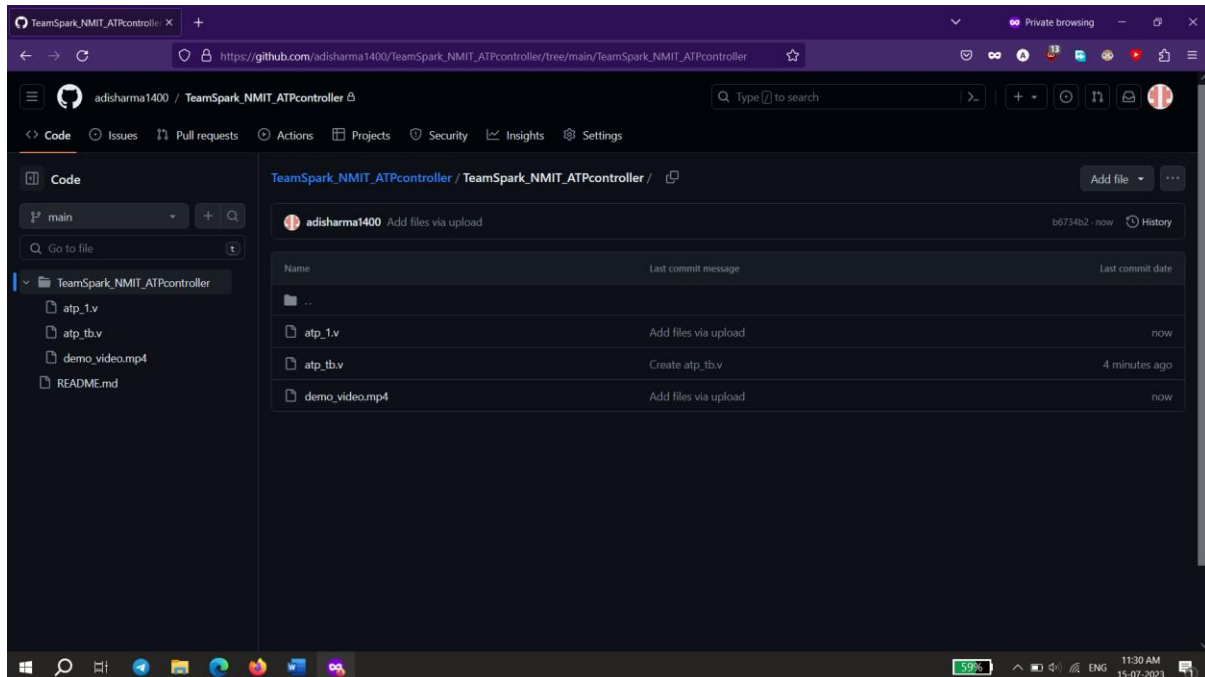
## Error in Modelsim:



## V. GITHUB FILE SCREENSHOT

*GitHub repo Link:*

[https://github.com/adisharma1400/TeamSpark\\_NMIT\\_ATPcontroller/tree/main/TeamSpark\\_NMIT\\_ATPcontroller](https://github.com/adisharma1400/TeamSpark_NMIT_ATPcontroller/tree/main/TeamSpark_NMIT_ATPcontroller)



## VI. RESULTS AND SUMMARIZATIONS

The main file and the test bench have been compiled without any errors. As observed due to some technical issue the RTL simulations could not be performed. Trying to troubleshoot this issue could not be done and there was also no updated documentation for the shown error.

## VII. REFERENCES

1. State Machine Coding Style for Synthesis.....Clifford E. Cummings (1998)
2. System Machine Design Techniques for Verilog HDL and VHDL.....Steve Golson.

