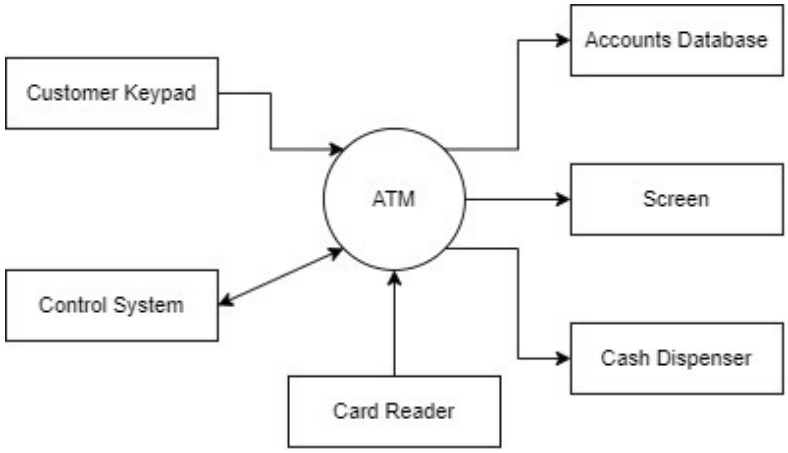
Introduction

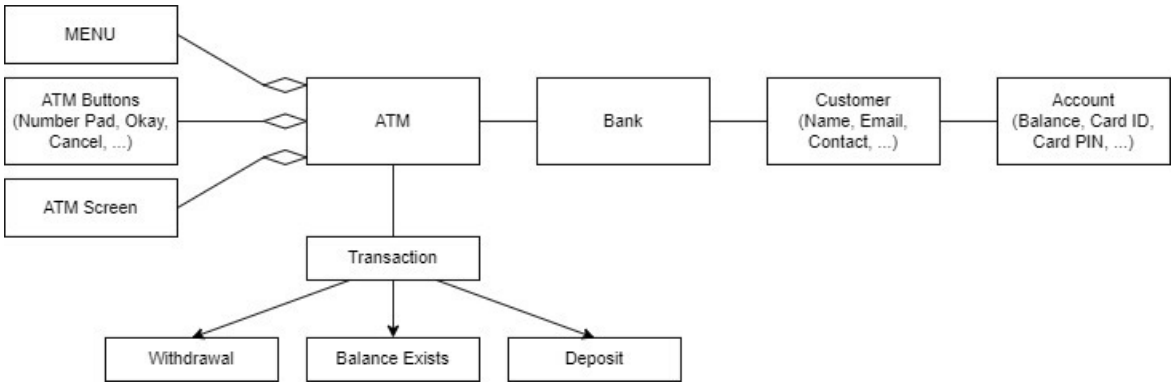
The project is implementation core of ATM system using Verilog FSM (finite state machine) code considering ATM functionalities as follows:

- Card handling
- Card password
- Operations (withdraw – deposit – balance check)
- Amount for transactions
- Another services

Analysis



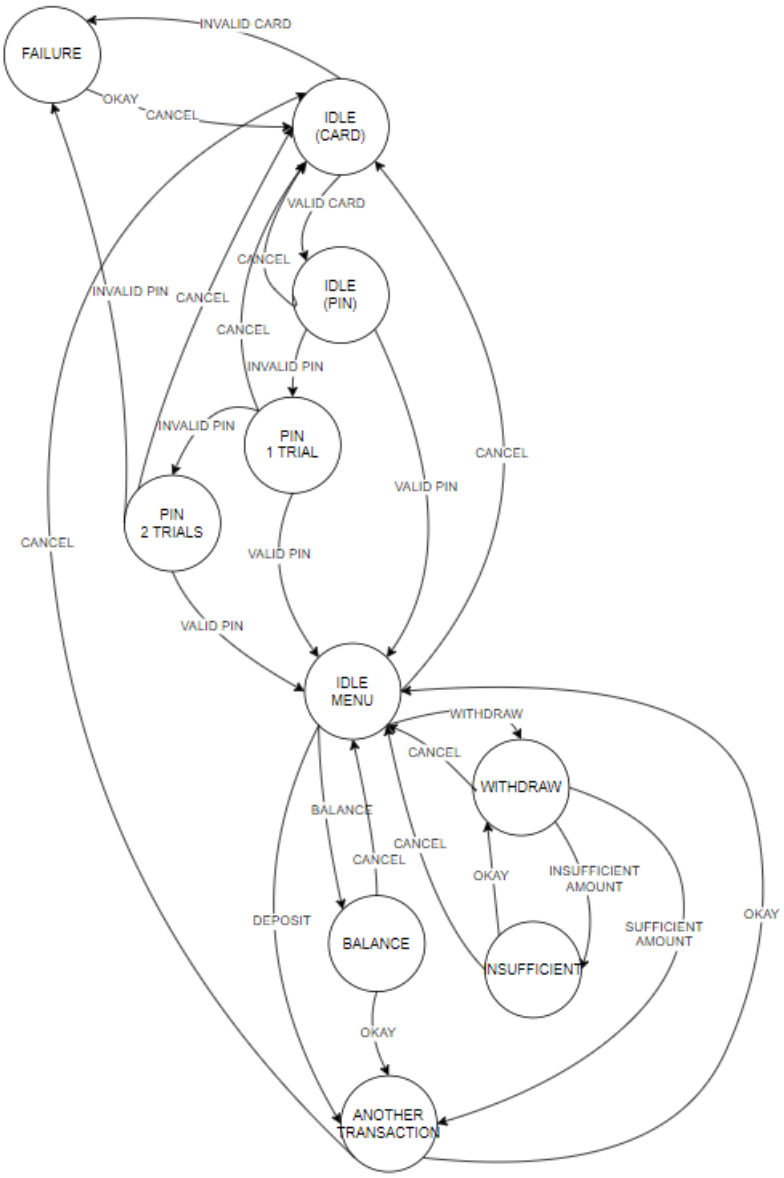
System Architecture



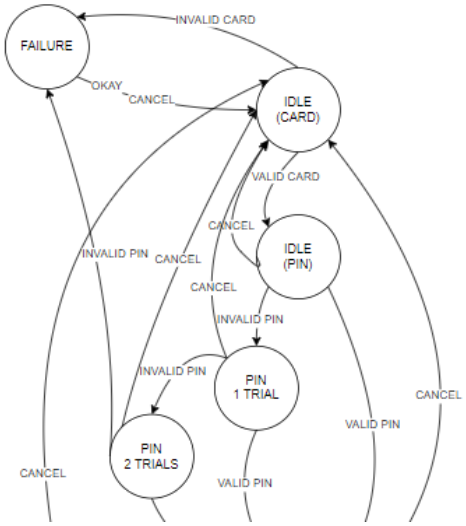
High-level model

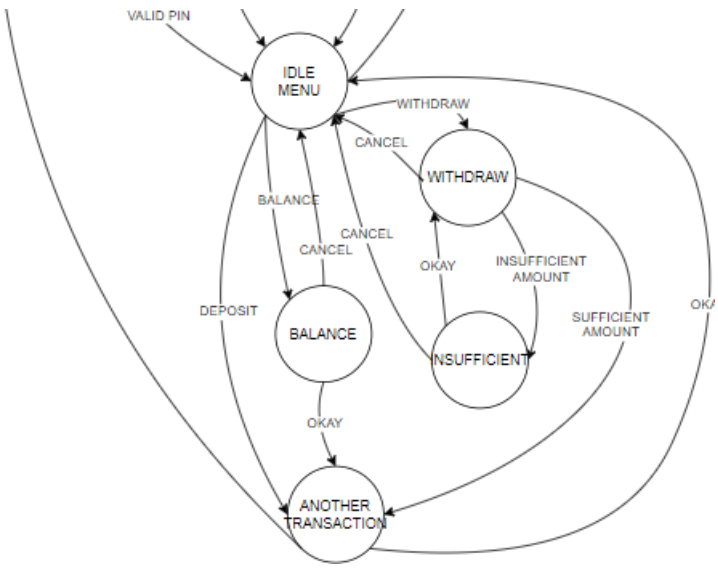
Design and state diagram

To implement ATM functionalities, we used 10 states with 33 transitions each state has it's own condition to proceed. States and transitions are illustrated in the following state diagram:



The state diagram consists of 2 phases. The first one is the validation phase while the second one is the transaction phase.





States where introduced using Grey State Encoding, because it take less switching power as State bits change only 1 bit between sequential states.

```
localparam [3:0]  IDLE      = 4'b0000,
                  FAILURE   = 4'b0001,
                  IDLEPIN   = 4'b0011,
                  PINTRIAL1 = 4'b0010,
                  PINTRIAL2 = 4'b0110,
                  IDLEMENU  = 4'b0111,
                  WITHDRAW  = 4'b0101,
                  INSUFFICIENT = 4'b0100,
                  BALANCE   = 4'b1100,
                  ANOTHER   = 4'b1101;
```

Input and outputs

Our design handled input and output as shown in Figure 1 , CARD_ID used to get the ID of the card for card handling, where PIN[3:0] used for entering password digit by digit, TRANSACTION used to determine which operation is required to procced to it's state, AMOUNT represent the amount of money needed for transactions.

Okay and cancel represent keys for proceeding in specific operation or not.

```
module ATM_MOORE (
  input wire unsigned [3:0] CARD_ID,
  input wire unsigned [3:0] PIN0,
  input wire unsigned [3:0] PIN1,
  input wire unsigned [3:0] PIN2,
  input wire unsigned [3:0] PIN3,
  input wire [1:0] TRANSACTION,
  input wire unsigned [31:0] AMOUNT,
  input wire OKAY,
  input wire CANCEL,

  input wire CLK,
  input wire RESET,

  output reg SUCCESS
);
```

Figure 1

Output SUCCESS is used to indicate whether the program has been proceeded to the next state required or not.

States and Transitions

There is 10 states:

1. IDLE (CARD) : card handling take place at this state. Where the card is handled by determining whether it's ID belong to our database or not. If it was found in our database the system will go to the next state which is IDLE(PIN). Otherwise, it will go to FAILURE state.

```
IDLE:
begin
  if(CARD_ID >= DBSIZE || CARD_ID < 0) NEXT = FAILURE;
  else if(CARD_ID < DBSIZE && CARD_ID >= 0) NEXT = IDLEPIN;
  else NEXT = CURRENT;
end
```

2. IDLE(PIN): at this state it is required from the user to enter the password. The PIN digits entered by the user is evaluated and validated against the one saved in PINS . Where PINS are regs used to hold the password for each card. If the user press CANCEL then the system will change it's state to IDLE. Otherwise, if the PINS are validated and it's true the system will proceed by going to IDLEMENU, else if it's not true it will go to PINTRIAL1.

```
IDLEPIN:
begin
  if(CANCEL)
    NEXT = IDLE;
  else if(OKAY && (PIN0 != PINS[CARD_ID][0] || PIN1 != PINS[CARD_ID][1] ||
PIN2 != PINS[CARD_ID][2] || PIN3 != PINS[CARD_ID][3]))
    NEXT = PINTRIAL1;
  else if(OKAY && PIN0 == PINS[CARD_ID][0] && PIN1 == PINS[CARD_ID][1] &&
PIN2 == PINS[CARD_ID][2] && PIN3 == PINS[CARD_ID][3])
    NEXT = IDLEMENU;
  else NEXT = CURRENT;
end
```

3. PINTRIALS: those states are trials to make the user enter the password again if they entered it incorrectly or get transitioned to the FAILURE state if they entered it incorrectly for 3 consecutive times. If the password was correct it will continue through IDLEMENU.

```
  PINTRIAL1:
  begin
    if(CANCEL)
      NEXT = IDLE;
    else if(OKAY && (PIN0 != PINS[CARD_ID][0] || PIN1 != PINS[CARD_ID][1] ||
 PIN2 != PINS[CARD_ID][2] || PIN3 != PINS[CARD_ID][3]))
      NEXT = PINTRIAL2;
    else if(OKAY && PIN0 == PINS[CARD_ID][0] && PIN1 == PINS[CARD_ID][1] &&
PIN2 == PINS[CARD_ID][2] && PIN3 == PINS[CARD_ID][3])
      NEXT = IDLEMENU;
    else NEXT = CURRENT;
  end
  PINTRIAL2:
  begin
    if(CANCEL)
      NEXT = IDLE;
    if(OKAY && (PIN0 != PINS[CARD_ID][0] || PIN1 != PINS[CARD_ID][1] || PIN2
 != PINS[CARD_ID][2] || PIN3 != PINS[CARD_ID][3]))
      NEXT = FAILURE;
    else if(OKAY && PIN0 == PINS[CARD_ID][0] && PIN1 == PINS[CARD_ID][1] &&
PIN2 == PINS[CARD_ID][2] && PIN3 == PINS[CARD_ID][3])
      NEXT = IDLEMENU;
    else NEXT = CURRENT;
  end
```

4. IDLEMENU: this state contain the main operations of the ATM (deposit – withdraw – balance check) first the system detect which operation is selected, in case of withdraw and balance checking operation were selected the system go to the state of each operation from them while in case of deposit the entered value of the amount will be added to the balance saved for a specific card.

```
IDLEMENU:
begin
  if(CANCEL)
    NEXT = IDLE;
  else if(TRANSACTION == WITHDRAWOP)
    NEXT = WITHDRAW;
  else if(TRANSACTION == INQUIREOP)
    NEXT = BALANCE;
  else if(TRANSACTION == DEPOSITOP)
  begin
    BALANCES[CARD_ID] = BALANCES[CARD_ID] + AMOUNT;
    NEXT = ANOTHER;
  end
  else NEXT = CURRENT;
end
```

5. WITHDRAW: this state check whether the entered amount is smaller than the balance or not. If the amount is smaller than the balance, the withdrawal will be successful processing to next state, which is ANOTHER, and the amount will be discarded from the balance. If not, it will go to state of INSUFFICENT.

```
WITHDRAW:
begin
  if(CANCEL)
    NEXT = IDLEMENU;
  else if(OKAY && (AMOUNT > BALANCES[CARD_ID]))
    NEXT = INSUFFICIENT;
  else if(OKAY && AMOUNT <= BALANCES[CARD_ID])
  begin
    BALANCES[CARD_ID] = BALANCES[CARD_ID] - AMOUNT;
    NEXT = ANOTHER;
  end
  else NEXT = CURRENT;
end
```

6. INSUFFICENT: at this state if the user wanted to proceed and select OKAY it will go back to the state of WITHDRAW to enter another amount, the other case that the user will select CANCEL and the system will get him back to IDLEMENU.

```
INSUFFICIENT:
begin
  if(CANCEL)
    NEXT = IDLEMENU;
  else if(OKAY)
    NEXT = WITHDRAW;
  else NEXT = CURRENT;
end
```

7. BALANCE: this state has 2 transitions where it goes to IDLEMENU if CANCEL is selected. If OKAY is selected, it goes to ANOTHER. It produces flag with value of 1 if the balance of the card is bigger than 0. If it is equivalent to 0, SUCCESS flag value will become 0.

```
BALANCE:
begin
  if(CANCEL)
    NEXT = IDLEMENU;
  else if(OKAY)
    NEXT = ANOTHER;
  else NEXT = CURRENT;
end
```

```
  BALANCE:
  begin
      if(BALANCES[CARD_ID] > 0) SUCCESS = 1'b1;
      else if(BALANCES[CARD_ID] == 0) SUCCESS = 1'b0;
      else SUCCESS = 1'bz;
  end
default: SUCCESS = 1'bz;
endcase
```

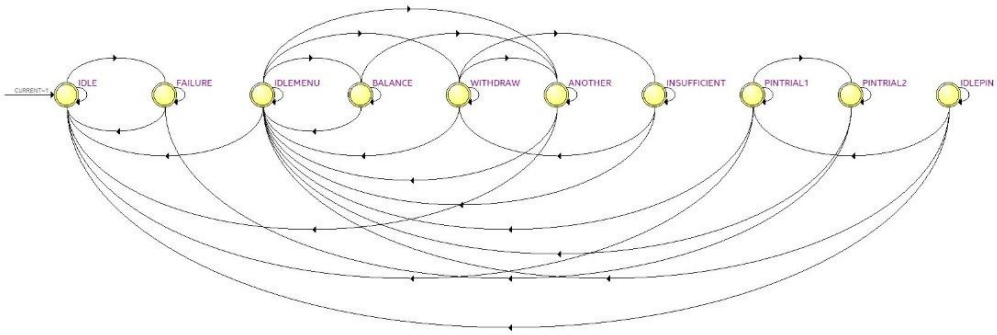
8. ANOTHER: system get respond from user by entering values for keys (OKAY and CANCEL) to know whether it will continue in another transaction so it go to IDLEMENU or it will go to IDLE(CARD) to exit.

```
ANOTHER:
begin
  if(CANCEL)
    NEXT = IDLE;
  else if(OKAY)
    NEXT = IDLEMENU;
  else NEXT = CURRENT;
end
```

9. FAILURE: this state indicates that password validation was failed and it will go back to IDLE(CARD) with any key entered from the user.

```
FAILURE:
begin
  if(CANCEL || OKAY)
    NEXT = IDLE;
  else
    NEXT = CURRENT;
end
```

QuestaSim Synthesis Report



Finite State Machine Diagram

State Table

	Name	ANOTHER	BALANCE	IDLEMENU	PINTRIAL2	WITHDRAW	INSUFFICIENT	IDLEPIN	PINTRIAL1	FAILURE	IDLE
1	IDLE	0	0	0	0	0	0	0	0	0	0
2	FAILURE	0	0	0	0	0	0	0	0	1	1
3	PINTRIAL1	0	0	0	0	0	0	0	1	0	1
4	IDLEPIN	0	0	0	0	0	0	1	0	0	1
5	INSUFFICIENT	0	0	0	0	0	1	0	0	0	1
6	WITHDRAW	0	0	0	0	1	0	0	0	0	1
7	PINTRIAL2	0	0	0	1	0	0	0	0	0	1
8	IDLEMENU	0	0	1	0	0	0	0	0	0	1
9	BALANCE	0	1	0	0	0	0	0	0	0	1
10	ANOTHER	1	0	0	0	0	0	0	0	0	1

States Encoding

State Table

	Source State	Destination State	Condition
1	ANOTHER	ANOTHER	(!OKAY),(!CANCEL)
2	ANOTHER	IDLE	(CANCEL)
3	ANOTHER	IDLEMENU	(OKAY),(!CANCEL)
4	BALANCE	ANOTHER	(OKAY),(!CANCEL)
5	BALANCE	BALANCE	(!OKAY),(!CANCEL)
6	BALANCE	IDLEMENU	(CANCEL)
7	FAILURE	FAILURE	(!OKAY),(!CANCEL)
8	FAILURE	IDLE	(!CANCEL),(OKAY) + (CANCEL)
9	IDLE	FAILURE	(!CARD_ID[0]),(!CARD_ID[2]) + (!CARD_ID[0]),(CARD_ID[2]),(!CARD_ID[3]) + (CARD_ID[0]),(!CARD_ID[1]),(!CARD_ID[2]) + (CARD_ID[0]),(!CARD_ID[1]),(CARD_ID[2]),(!CARD_ID[3]) + (CARD_ID[0]),(CARD_ID[1]),(!CARD_ID[3])
10	IDLE	IDLE	(!CARD_ID[0]),(CARD_ID[2]),(CARD_ID[3]) + (CARD_ID[0]),(!CARD_ID[1]),(CARD_ID[2]),(CARD_ID[3]) + (CARD_ID[0]),(CARD_ID[1]),(CARD_ID[3])
11	IDLEMENU	ANOTHER	(!CANCEL),(!TRANSACTION[0]),(TRANSACTION[1])
12	IDLEMENU	BALANCE	(!CANCEL),(TRANSACTION[0]),(!TRANSACTION[1])
13	IDLEMENU	IDLE	(CANCEL)
14	IDLEMENU	IDLEMENU	(!CANCEL),(TRANSACTION[0]),(TRANSACTION[1])
15	IDLEMENU	WITHDRAW	(!CANCEL),(!TRANSACTION[0]),(!TRANSACTION[1])
16	IDLEPIN	IDLEPIN	(!OKAY),(!CANCEL)
17	IDLEPIN	IDLE	(CANCEL)
18	IDLEPIN	IDLEMENU	(OKAY),(!CANCEL),(Equal4),(Equal3),(Equal2),(Equal1)
19	IDLEPIN	PINTRIAL1	(!Equal4),(OKAY),(!CANCEL) + (Equal4),(!Equal3),(OKAY),(!CANCEL) + (Equal4),(Equal3),(!Equal2),(OKAY),(!CANCEL) + (Equal4),(Equal3),(Equal2),(!Equal1),(OKAY),(!CANCEL)
20	INSUFFICIENT	IDLEMENU	(CANCEL)
21	INSUFFICIENT	INSUFFICIENT	(!OKAY),(!CANCEL)
22	INSUFFICIENT	WITHDRAW	(OKAY),(!CANCEL)
23	PINTRIAL1	IDLE	(CANCEL)
24	PINTRIAL1	IDLEMENU	(OKAY),(!CANCEL),(Equal4),(Equal3),(Equal2),(Equal1)
25	PINTRIAL1	PINTRIAL1	(!OKAY),(!CANCEL)
26	PINTRIAL1	PINTRIAL2	(!Equal4),(OKAY),(!CANCEL) + (Equal4),(!Equal3),(OKAY),(!CANCEL) + (Equal4),(Equal3),(!Equal2),(OKAY),(!CANCEL) + (Equal4),(Equal3),(Equal2),(!Equal1),(OKAY),(!CANCEL)
27	PINTRIAL2	FAILURE	(!Equal4),(OKAY) + (Equal4),(!Equal3),(OKAY) + (Equal4),(Equal3),(!Equal1),(OKAY) + (Equal4),(Equal3),(Equal1),(!Equal2),(OKAY)
28	PINTRIAL2	IDLEMENU	(OKAY),(Equal4),(Equal3),(Equal2),(Equal1)
29	PINTRIAL2	PINTRIAL2	(!OKAY)
30	WITHDRAW	ANOTHER	(OKAY),(!CANCEL),(LessThan6),(!LessThan5)
31	WITHDRAW	IDLEMENU	(CANCEL)
32	WITHDRAW	INSUFFICIENT	(OKAY),(!CANCEL),(LessThan5)
33	WITHDRAW	WITHDRAW	(!OKAY),(!CANCEL) + (OKAY),(!CANCEL),(!LessThan6),(!LessThan5)

States Transitions

Coverage Results:

The total coverage was affected due to the number of bits used for the balance registers and the number of bits used for the PINs digits, however, we covered all the possible scenarios mentioned in the FSM using directed testing which makes the verification plan entirely satisfactorily.

```
# QuestaSim-64 vcover 10.6c Coverage Utility 2017.07 Jul 26 2017
# Start time: 23:56:36 on Dec 18,2022
# vcover report ATM_MOORE_TB.ucdb
# Coverage Report Summary Data by file
#
# ================================================================================
# === File: ATM.v
# ================================================================================
#     Enabled Coverage            Active      Hits    Misses % Covered
#     ----------------            ------      ----    ------ ---------
#     Stmts                           59        49        10      83.0
#     Branches                        61        52         9      85.2
#     FEC Condition Terms             40         7        33      17.5
#     FEC Expression Terms             0         0         0     100.0
#     FSMs                                                        85.7
#         States                      10        10         0     100.0
#         Transitions                 28        20         8      71.4
#     Toggle Bins                    206       112        94      54.3
#
# ================================================================================
# === File: ATM_tb.v
# ================================================================================
#     Enabled Coverage            Active      Hits    Misses % Covered
#     ----------------            ------      ----    ------ ---------
#     Stmts                          227       227         0     100.0
#     Branches                         2         2         0     100.0
#     FEC Condition Terms              0         0         0     100.0
#     FEC Expression Terms             0         0         0     100.0
#     FSMs                                                        100.0
#         States                       0         0         0     100.0
#         Transitions                  0         0         0     100.0
#     Toggle Bins                    190       123        67      64.7
#
#
# TOTAL ASSERTION COVERAGE: 62.5%  ASSERTIONS: 8
#
# Total Coverage By File (code coverage only, filtered view): 68.9%
#
# End time: 23:56:36 on Dec 18,2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
```