**🡪 MAIN FSM Assertions**

1. If main\_state=’b000 and in\_valid\_reg=1 and most two significant bits of in\_data equals to ‘b00 , then after one clock cycle same\_bank\_reg should be equal to 1
2. If main\_state=’b100 and done =1 and in\_valid\_reg=0 , then after one clock cycle flag=1
3. If main\_state=’b100 and done=0 and flag=0 ,then in same clock cycle disp\_sig\_main=’b00011 and after one clock cycle main\_state=’b100

**Deposit FSM Assertions**

1. if dep\_state=’b000 and enable not equal to 001 then dep\_next\_state=000
2. if dep\_state=’b000 and enable=001 then after 1 clock cycle dep\_state=’b001
3. if dep\_state=’b001 and timer\_done=0 and in\_valid\_reg=0 then after 1clk cycle dep\_ state=000
4. if dep\_state=’b001 and timer\_done=0 and in\_valid\_reg=0 then after 1 clock cycle dep\_ state=’b000
5. if dep\_state=’b000 and enable=1 then after 1 clock cycle dep\_state=’b001 ,then after 9 clock cycles if timer\_done=1 and in\_valid\_reg=1 , then dep\_next state will be equal to ‘b010
6. if dep\_state equals to **010** then in **same clock cycle** disp\_sig\_deposit should be equal to **01001** , and data\_valid\_dep equals to **1** ,and out\_data\_dep equals to cash\_amount
7. if dep\_next\_state =’b011 then after nine clock cycle timer\_done change from 0 to 1 , and after one clock cycle dep\_state equals to 001
8. if dep\_state =’b100 then in same clock cycle out\_data\_dep equals to Balance
9. if dep\_state =’b101 then in same clock cycle done\_dep should change from 0 to 1 , and 1 clock cycle dep\_state should be equal to ‘b000

**Balance Inquiry FSM Assertions**

1. if inq\_state equals to ‘b01 , then in same clock cycle out\_data should be equal to Balance and data\_valid\_inq equals to 1.
2. If inq\_state equals to ‘b000 and enable not equal to ‘b010 then after in same clock cycle inq\_next\_state is equal to ‘b000
3. If inq\_next\_state=’b01 ,then after one clock cycle inq\_state should be equal to ‘b01
4. If inq\_state equals to ’b11 then after 8 clock cycles done\_inq changes from 0 to 1 and inq\_next\_state =’b000
5. If enable =010 ,then data\_valid=data\_valid\_inq

**Withdrawal FSM Assertions:**

1. If “withdraw\_current\_state” is “000” and “enable” becomes equal to “001” then “withdraw\_next\_state” is “001” otherwise “withdraw\_next\_state” is equal to “000” and in both cases, at the following rising edge “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
2. If “withdraw\_current\_state” is “001” then “disp\_sig\_with” becomes equal to “01110” and the system waits in this state until “in\_valid\_reg” is raised. Whenever it’s raised, “withdraw\_next\_state” becomes “010” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
3. If “withdraw\_current\_state” is “010” and if “withdrawal\_choice” is low then “disp\_sig\_with” becomes “01111” and the system waits in this state until “in\_valid\_reg” is raised. Whenever it’s raised, “withdraw\_next\_state” becomes “011” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
4. If “withdraw\_current\_state” is “010” and if “withdrawal\_choice” is high then “disp\_sig\_with” becomes “10000” and the system waits in this state until “in\_valid\_reg” is raised. Whenever it’s raised, “withdraw\_next\_state” becomes “011” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
5. If “withdraw\_current\_state” is “011” and if “same\_bank\_reg” is high then check if “withdraw\_amount” is greater than “Balance”,

if so then “disp\_sig\_with” becomes “10100” and “withdraw\_next\_state” becomes “010”

Otherwise check if “withdraw\_amount” is greater than “Bank\_withdraw\_Limit”, if so then “disp\_sig\_with” becomes “10101” and “withdraw\_next\_state” becomes “010”

Otherwise “out\_data\_with” is set to “withdraw\_amount” and

“disp\_sig\_with” is set to “01101” and “data\_valid\_with” is raised and “Balance” changes; becomes equal to “Balance” minus “withdraw\_amount” and “withdraw\_next\_state” becomes “100” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”

1. If “withdraw\_current\_state” is “011” and if “same\_bank\_reg” is low then check if “withdraw\_amount” plus 5 is greater than “Balance”, if so then “disp\_sig\_with” becomes “10100” and “withdraw\_next\_state” becomes “010”

Otherwise check if “withdraw\_amount” plus 5 is greater than “Bank\_withdraw\_Limit”, if so then “disp\_sig\_with” becomes “10101” and “withdraw\_next\_state” becomes “010”

Otherwise “out\_data\_with” is set to “withdraw\_amount” and

“disp\_sig\_with” is set to “01101” and “data\_valid\_with” is raised and “Balance” changes; becomes equal to “Balance” minus “withdraw\_amount” minus 5 and “withdraw\_next\_state” becomes “100” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”

1. If “withdraw\_current\_state” is “100” then “disp\_sig\_with” becomes “01101” and “timer\_en\_with” is raised and the system stays in this state for 8 clk cycles then timer\_done is raised then at the 9th clock cycle, “withdraw\_next\_state” becomes “101” and “timer\_en\_with” is lowered and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
2. If “withdraw\_current\_state” is “101” then “disp\_sig\_with” becomes “10001” then wait until “in\_valid\_reg” is raised. Whenever it’s raised, “print receipt” changes and “withdraw\_next\_state” becomes “110” and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
3. If “withdraw\_current\_state” is “110” and if “print\_receipt” is high then check if “same\_bank\_reg” is high if so then “disp\_sig\_with” becomes “10010” else “disp\_sig\_with” becomes “10011” and in both cases “out\_data\_with” becomes equal to “withdraw\_amount” and “data\_valid\_with” is raised and “timer\_en\_with” is raised then the system stays in this state for 8 clk cycles then timer\_done is raised then at the 9th clock cycle, “withdraw\_next\_state” becomes “111” and “timer\_en\_with” is lowered and at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
4. If “withdraw\_current\_state” is “110” and if “print\_receipt” is low then check if “same\_bank\_reg” is high if so then “disp\_sig\_with” becomes “10010” else “disp\_sig\_with” becomes “10011” and in both cases “out\_data\_with” becomes equal to “withdraw\_amount” and “data\_valid\_with” is raised then “withdraw\_next\_state” becomes “111” then at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
5. If “withdraw\_current\_state” is “111” then check if “same\_bank\_reg” is high if so then “disp\_sig\_with” becomes “10010” else “disp\_sig\_with” becomes “10011” and in both cases “out\_data\_with” becomes equal to “withdraw\_amount” and “data\_valid\_with” is raised and “done\_with” is raised then “withdraw\_next\_state” becomes “000” then at the following rising edge, “withdraw\_current\_state” becomes equal to “withdraw\_next\_state”
6. If “withdraw\_current\_state” is none of the above then “disp\_sig\_with” becomes “01110” and “withdraw\_next\_state” becomes “000”

**PIN Change FSM Assertions**

1. If pin\_state = 000 & enable = 100, pin\_next\_state = 001 in same clock cycle and in the next clock cycle pin\_state = 001
2. If pin\_state = 000 & enable != 100, pin\_next\_state = 000 in same clock cycle and in the same clock cycle pin\_state = 000
3. If pin\_state = 001 & in\_valid\_reg = 1, pin\_next\_state = 010 in same clock cycle and in the next clock cycle pin\_state = 010
4. If pin\_state = 001 & in\_valid\_reg = 0, pin\_next\_state = 001 in same clock cycle and in the next clock cycle pin\_state = 001
5. If pin\_state = 010 & in\_valid\_reg = 1, pin\_next\_state = 011 in same clock cycle and in the next clock cycle pin\_state = 011
6. If pin\_state = 010 & in\_valid\_reg = 0, pin\_next\_state = 010 in same clock cycle and in the next clock cycle pin\_state = 010
7. If pin\_state = 011 & RE\_PIN\_temp = PIN\_temp, timer\_en\_pin = 1 & pin\_next\_state = 011 in same clock cycle and in the next clock cycle pin\_state = 011 and after complete 7 clock cycles, timer\_done = 1 & timer\_en\_pin = 0 & pin\_next\_state = 100 and in the next clock cycle pin\_state = 100
8. If pin\_state = 011 & RE\_PIN\_temp != PIN\_temp, pin\_next\_state = 001 in same clock cycle and in the next clock cycle pin\_state = 001
9. If pin\_state = 100 & main\_state = 100, pin\_next\_state = 000 & done\_pin = 1 & done = 1 & flag = 1 & enable = 000 in same clock cycle and in the next clock cycle pin\_state = 000 & done\_pin = 0 & done = 0
10. If pin\_state != 000 & pin\_state != 001 & pin\_state != 010 & pin\_state != 011 & pin\_state != 100, pin\_next\_state = 000 in same clock cycle and in the next clock cycle pin\_state = 000

**Note:** any internal signal that is stated in some cases and not in the others is by default = 0