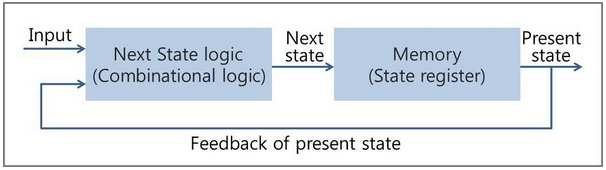
EE312 Assignment 1 Report

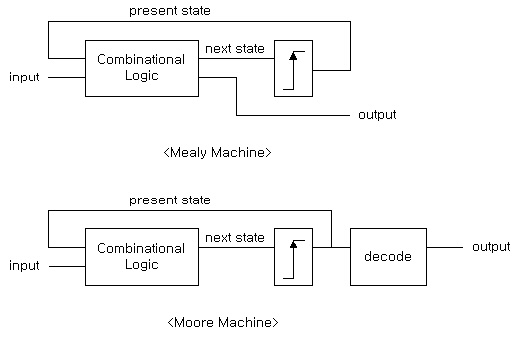
20110343 김진엽

1. Finite state machine

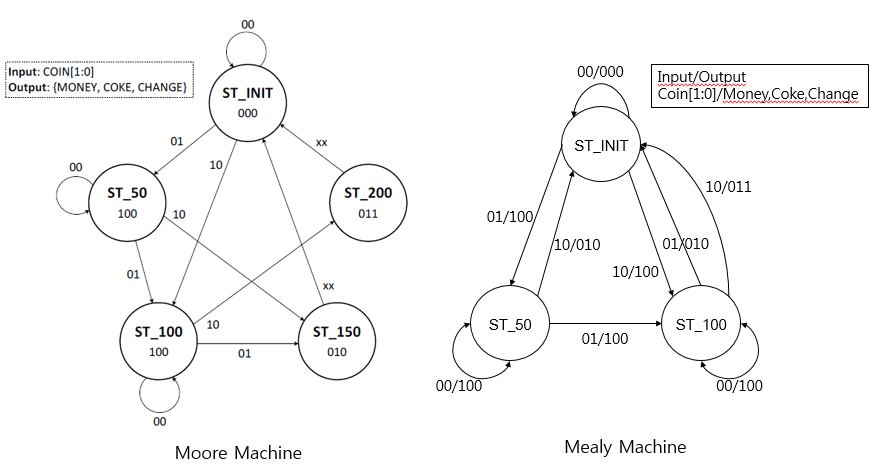


FSM is a machine that transits finite states with sequential logic. It has memory to save state. Input and past result decide next state with combinational logic. Then the state is stored in memory. Number of the memory is finite because of physical limits.

1. Mealy and Moore machine



Mealy and Moore machine are FSM. Output values of Mealy machine are determined both by its current state and the current inputs. In contrast, output values of Moore machine are determined solely by its current state.



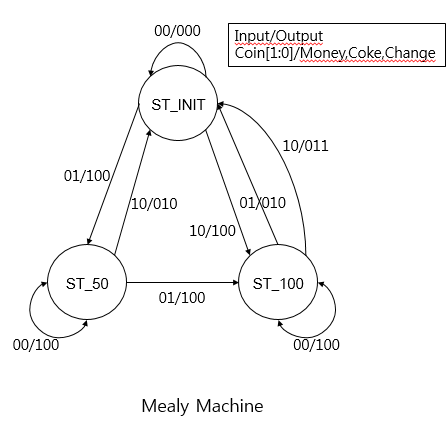
For example, the vending machine can be expressed with both machines.

In Moore machine, past state and input decide next state. Then the output is decoded state (ex: ST\_50 -> 100). However, Mealy machine directly decide output using past state and input.

Number of state in Mealy machine is usually smaller than Moore machine because of the difference. Moore machine has higher readability.

1. Vending machine implementation

I choose mealy machine to make vending machine. It need less number of states.



* 1. Module variables

module Vending\_Machine (

input wire CLK,

input wire RST,

input wire [1:0] COIN,

output wire MONEY,

output wire COKE,

output wire CHANGE,

output reg r\_money, r\_coke, r\_change,

output reg [0:2] STATE

);

Registers for MONEY, COKE, CHANGE are added. Also, a register that stores current state is added.

* 1. Initialization

Just set STATE as ST\_INIT.

<Code>

initial begin

STATE = ST\_INIT;

End

* 1. LOOP

To implement “Insertion of a coin is recognized at clock’s positive edge”, ‘Always @ (posedge CLK)’ is used.

If RST is true, then state become ST\_INIT.

If not, the FSM transits state.

<Code>

always @ (posedge CLK) begin

if(RST == 1'b1) begin

STATE = ST\_INIT;

end

else begin

/\* FSM transits state \*/

end

end

* 1. FSM

First, combinational logic with past state and input decide output.

r\_money = (~STATE[0]&~STATE[1]&~STATE[2]&COIN[1]&~COIN[0])+(~STATE[0]&~STATE[1]&STATE[2]&~COIN[1]&~COIN[0])+(~STATE[0]&~STATE[1]&~COIN[1]&COIN[0])+(~STATE[0]&STATE[1]&~STATE[2]&~COIN[1]&~COIN[0]);

r\_coke = (~STATE[0]&~STATE[1]&STATE[2]&COIN[1]&~COIN[0])+(~STATE[0]&STATE[1]&~STATE[2]&~COIN[1]&COIN[0])+(~STATE[0]&STATE[1]&~STATE[2]&COIN[1]&~COIN[0]);

r\_change = (~STATE[0]&STATE[1]&~STATE[2]&COIN[1]&~COIN[0]);

The logic is made using Karnaugh map.

The registers are assigned with MONEY, COKE, CHANGE wire, so the output is confirmed.

Then, store the next state.

1. Vending machine simulation

When 150 coin is inserted, a coke comes out. When 200 coin is inserted, a coke and change come out. When reset is true, the vending machine ignores all coins and state goes ST\_INIT(000).

