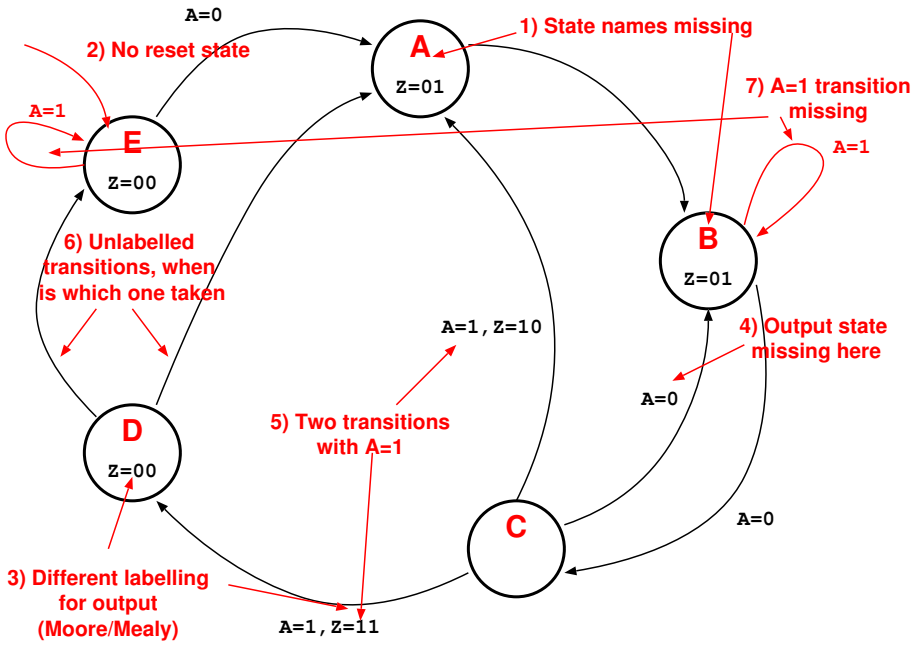
3. (a) (5 points) One of your colleagues is designing a finite state machine with a one-bit input (A) and a two-bit output (Z). He has started designing a state transitioning diagram as given below:



Even if you do not know the exact functionality of the FSM, there are some obvious problems with this diagram. Help your colleague by identifying the problems in this diagram. List *as many mistakes as you can find* in this diagram.

Solution: There are many problems with this diagram

1. The states do not have identifying names or labels
2. There is no reset state
3. Most states have a Moore labelling (output state in the bubble), one has a Mealy type labelling (output given with input transitions)
4. The output state is missing for the transition from the unlabelled state with $a = 0$.
5. There are two different transitions both with $a = 1$ from the same state
6. There are two different transitions from state $Z = 00$ that are not labelled
7. For two states only $a = 0$ makes a transition. What will happen with $a = 1$ is not shown, presumably will remain in the same state.

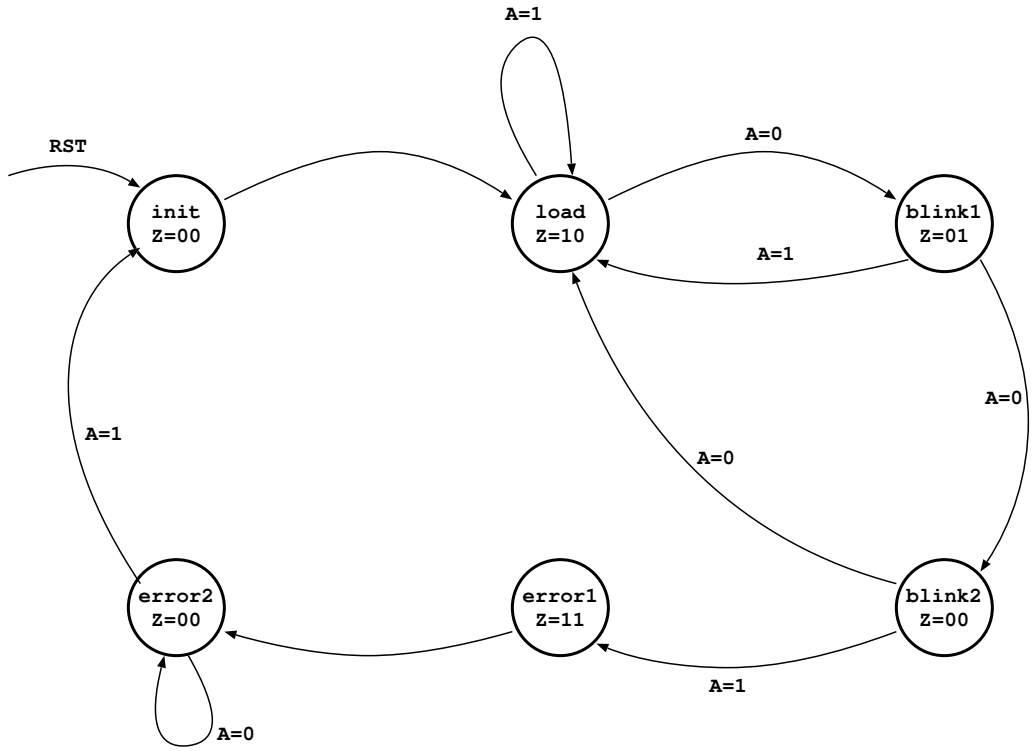
For every correctly spotted mistake you will get points, you will get full points if you have spotted 5 of the above.

(b) (4 points) After learning from his mistakes, your colleague has proceeded to write the following Verilog code for a much better (and different) FSM. The code has been verified for syntax errors and found to be OK.

```
1  module fsm (  input CLK, RST, A,
2                output [1:0] Z);
3
4    reg [2:0] next, present;
5
6    parameter init   = 3'b000;
7    parameter blink1 = 3'b010;
8    parameter blink2 = 3'b011;
9    parameter load   = 3'b100;
10   parameter error1 = 3'b110;
11   parameter error2 = 3'b111;
12
13   assign Z = (present == error1) ? 2'b11 :
14              (present == blink1) ? 2'b01 :
15              (present == load)   ? 2'b10 : 2'b00;
16
17   always @ (present, A)
18     case (present)
19       init   : next <= load;
20       load   : if (~A) next <= blink1;
21       blink1 : if (A) next <= load;
22                else   next <= blink2;
23       blink2 : if (A) next <= error1;
24                else   next <= load;
25       error1 : next <= error2;
26       error2 : if (A) next <=init;
27       default: next <= present;
28     endcase
29
30   always @ (posedge CLK, posedge RST)
31      if (RST)  present <= init;
32      else      present <= next;
33
34 endmodule
```

Draw a proper state transition diagram that corresponds to the FSM described in this Verilog code.

Solution:



(c) (1 point) Is the FSM described by the previous Verilog code a Moore or a Mealy FSM? Why?

Solution: Moore, the output Z only depends on the state (*present*) and not on the input (A).