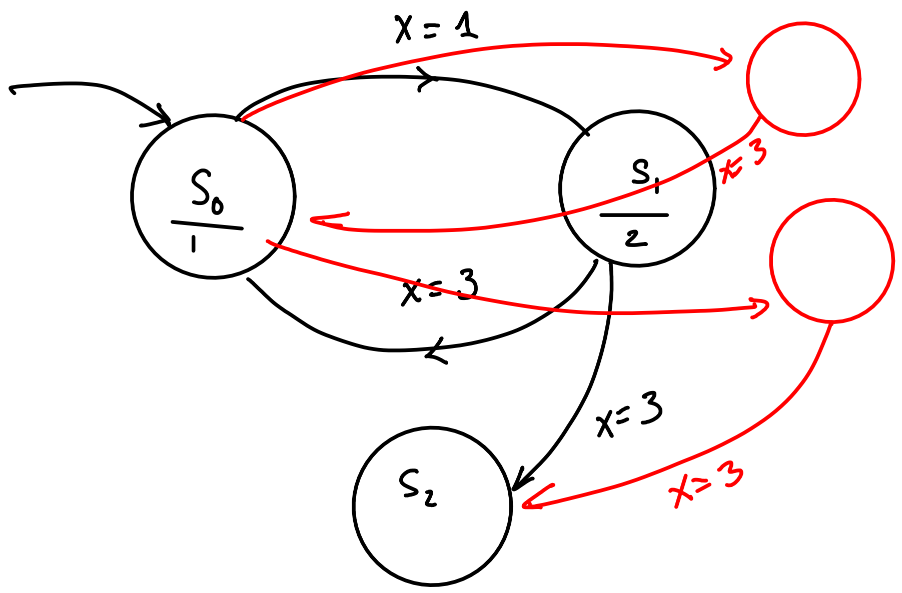
State Machine Design

Exercise 1: Which signal in the above diagrams indicates the current state?

Exercise 2: The link below shows a game. List the top-level game states. Decompose each of these into multiple states. Repeat.



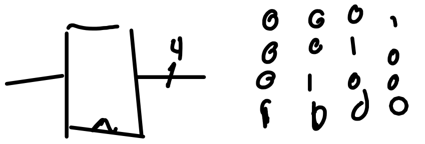
Exercise 3: If we used 8-bits of state information, how many states could be represented? What if we used 8 bits of state but used a "one-hot encoding"?

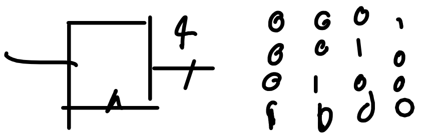
binary: $2^8 = 256$

one-hot 8

enum x = { S0, S1, P, X }

{ s0=8'b1, s1=8'b10, ... }





always_comb $\longrightarrow$ state_next = state ←

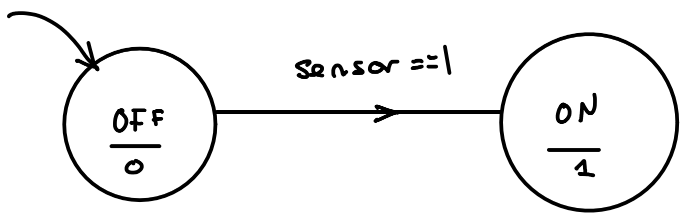
case (state)

S0: if x==3 state_next = S1
else

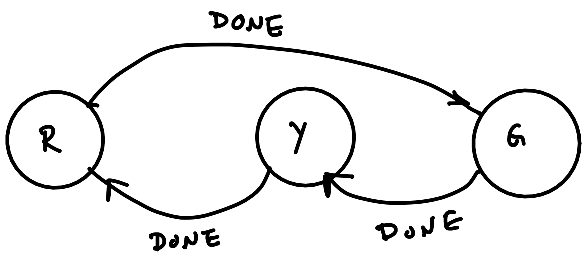
s1:

s2:

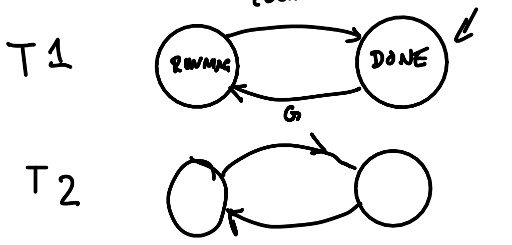
Exercise 4: Draw the state transition diagram.



Exercise 5: Draw the state transition diagram. Make a table showing the possible output values. What type of state encoding would be most appropriate?



R	Y	G
1	0	0
0	1	0
0	0	1



Exercise 6: Write the output encodings for a 2-bit Gray-coded counter.

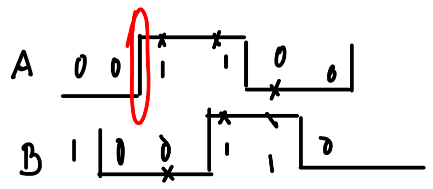
0 0

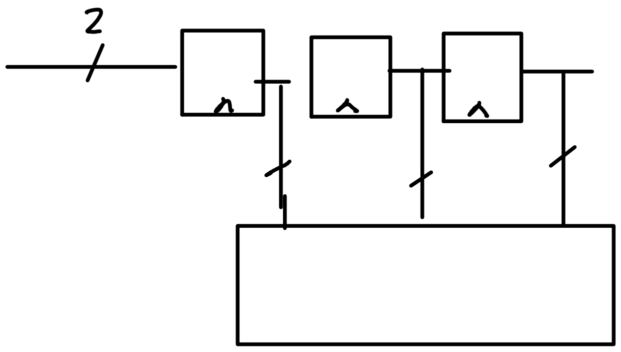
1 0

1 1

0 1

Exercise 7: A state machine has two inputs (A and B) with overlapping pulses. Design a state machine that detects when the rising edge of A happens before the rising edge of B.





Exercise 8: Which of the above state machines could you use in your project?

Exercise 9: Write the tabular description of a resetable 2-bit counter with demultiplexed outputs (only one of the four outputs is true at any time). You can assume the counter will always be reset before being used. How does this counter compare to the previous one in terms of number of flip-flops and the complexity of the combinational logic?

Exercise 10: Draw the state transition diagram.