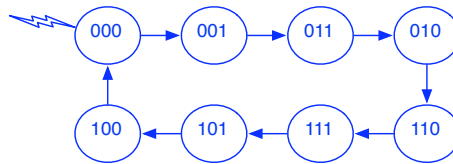
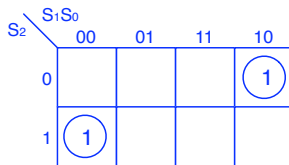


1. Gray codes have the useful property that consecutive numbers differ in only a single bit position. Design a 3-bit Gray code counter FSM with no inputs and three outputs. When reset, the output should be 000. On each clock edge, the output should advance to the next Gray code. After reaching 100, it should repeat with 000. Draw a schematic for this counter using T flip-flops.

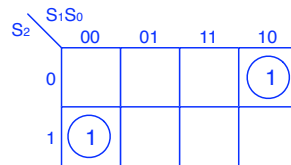
Number	Gray code
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0



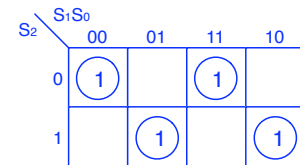
S_2	S_1	S_0	S_2^+	S_1^+	S_0^+	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0
1	0	1	1	0	0	0	0	1
1	0	0	0	0	0	1	0	0



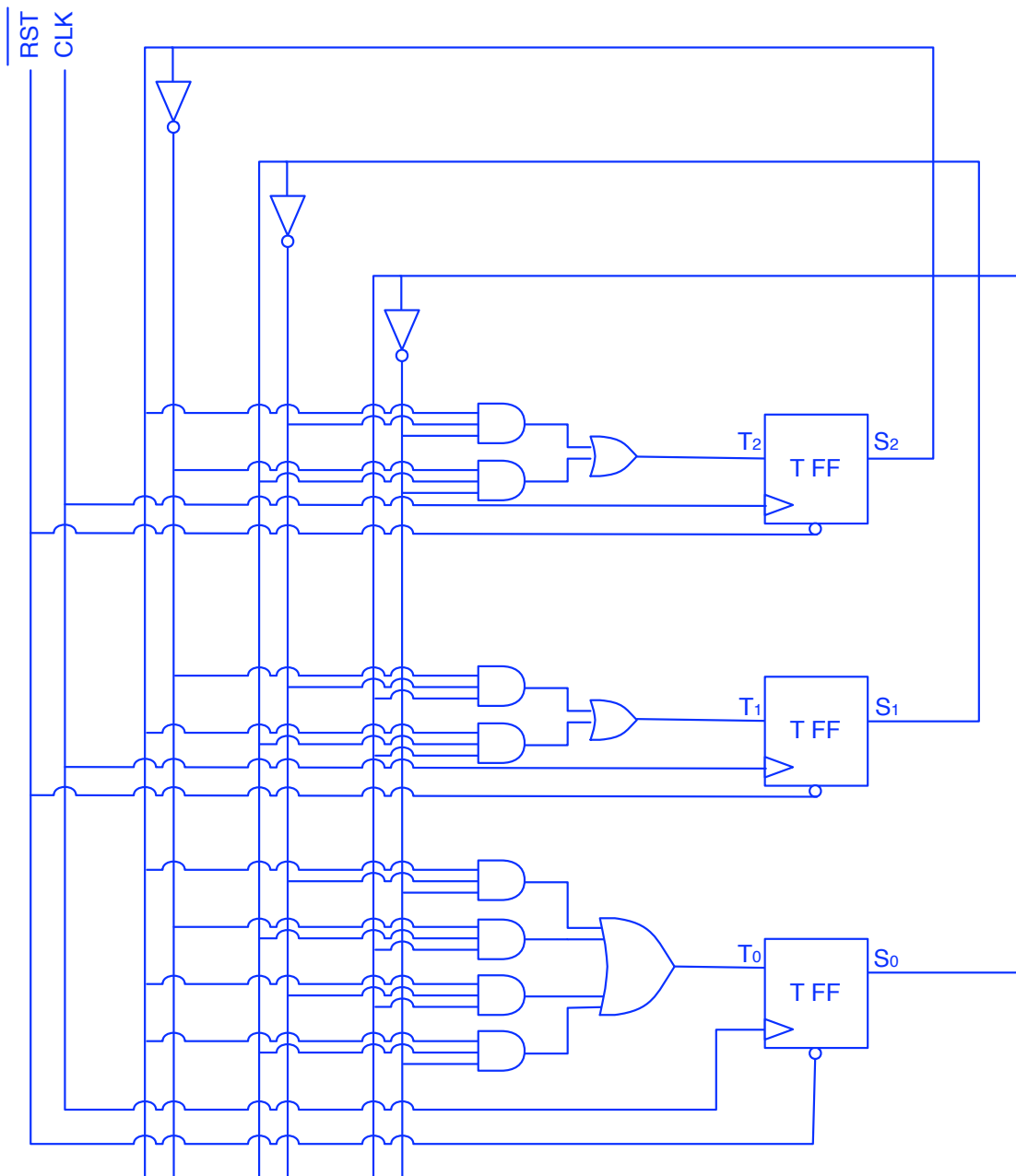
$$T_2 = S_2 \overline{S_1} \overline{S_0} + \overline{S_2} S_1 \overline{S_0}$$



$$T_1 = \overline{S_2} \overline{S_1} S_0 + S_2 S_1 S_0$$

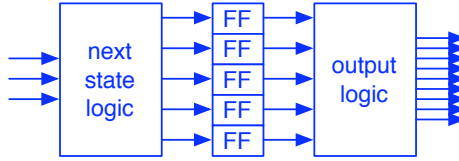


$$T_0 = \overline{S_2} \overline{S_1} \overline{S_0} + \overline{S_2} S_1 S_0 + S_2 \overline{S_1} S_0 + S_2 S_1 \overline{S_0}$$



2. Suppose you are told that a Moore machine has five flip-flops, three inputs, and nine outputs. Assume that the states are encoded as compactly as possible (e.g., no one-hot encoding) and that transitions are all explicitly labelled with an actual input value (e.g., no transitions with “x” labels). Answer the following questions:

The described Moore machine has the following general structure:



- (a) What are the minimum and maximum numbers of states in the state diagram?

Assuming a state encoding that makes efficient use of the flip-flops, the minimum number of states in the diagram is $2^{5-1} + 1$ (17) and the maximum is 2^5 (32).

- (b) What are the minimum and maximum numbers of transition arrows starting at a particular state?

A fully defined machine needs to know what action to take for all possible inputs, so, at a minimum each node will have 2^3 (8) transitions originating from it. Similarly, it is not possible to have more than one transition on a particular input. Thus, the maximum number of outgoing transitions is also 2^3 (8).

- (c) What are the minimum and maximum numbers of transition arrows that can end in a particular state?

Because unreachable states are unnecessary to consider and implement, we consider only reachable states. Under this assumption, a reachable state has a minimum of 1 incoming transition. The maximum number of incoming transitions would occur when

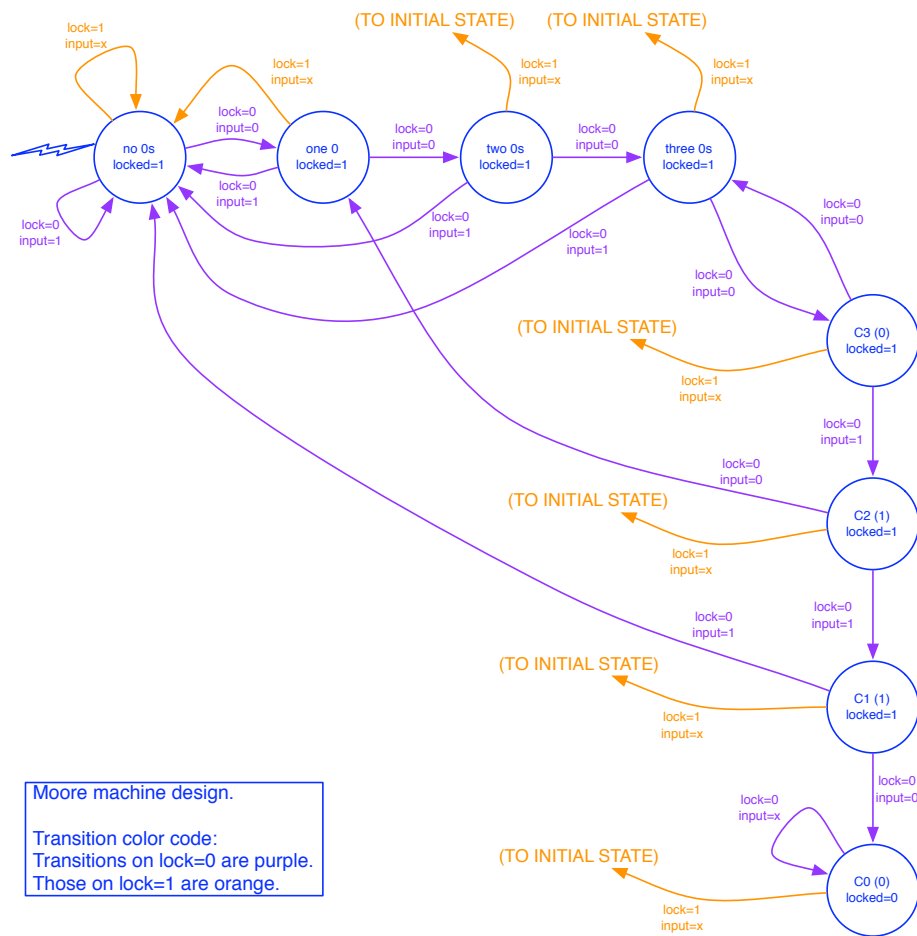
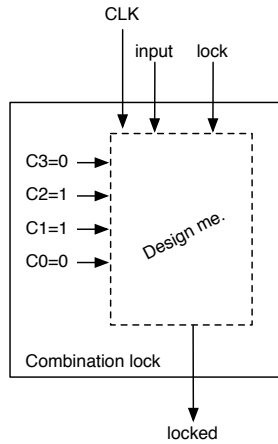
- all of the states (at most 2^5) are connected in a loop so that they are all reachable and valid, and
- all of the remaining edges (7: 2^3 per state less the 1 edge already part of the loop) are aimed at a single state

In this case, that single state would have 1 incoming edge from the loop plus $7 * 2^5$ incoming edges from the convergence, bringing the total to 225.

- (d) What are the minimum and maximum numbers of different binary patterns that can be displayed on the outputs?

Because this is a Moore machine, the output is a function of *only* the 5 state bits. At most, the machine can display as many outputs as there are states, or 2^5 (32). The minimum number of output values depends on certain assumptions about the output logic. However, assuming any output logic that is a function of the 5 state bits is valid, logic that assigns a constant value to the output no matter the state would produce the minimal number of output values (1).

3. Design a combination lock as shown in the drawing below. Assume the internally known combination $C_3C_2C_1C_0$ as shown. The state of the lock is indicated by the output *locked*. Once unlocked (*locked* = 0) the lock should remain unlocked until the *lock* input is true. Once locked, the lock will remain locked until it has seen the initiation sequence “000” followed by the correct combination $C_3 = 0, C_2 = 1, C_1 = 1, C_0 = 0$ on the input. Design the internal logic of this lock (using D flip-flops) and draw the circuit that implements it.



The circuit can use the lock input as a reset signal or have a traditional reset signal and treat lock as a standard input. We show the latter here as it is slightly more complicated.

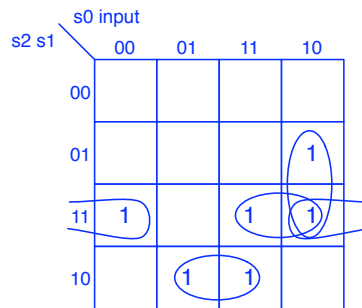
First, we encode the states as follows:

State	Encoding		
	s2	s1	s0
“no 0s”	0	0	0
“one 0”	0	0	1
“two 0s”	0	1	0
“three 0s”	0	1	1
“C3 (0)”	1	0	0
“C2 (1)”	1	0	1
“C1 (1)”	1	1	0
“C0 (0)”	1	1	1

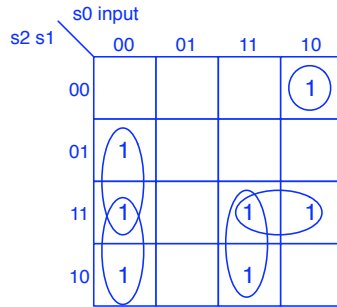
We next design two sets of next state logic, one for when $lock = 1$ and one when $lock = 0$.

- When $lock = 1$, the next state is always the initial state ($s2^+ = 0, s1^+ = 0, s0^+ = 0$).
- When $lock = 0$ we can design the next state logic as a function of the current state and the inputs according to the purple transitions above.

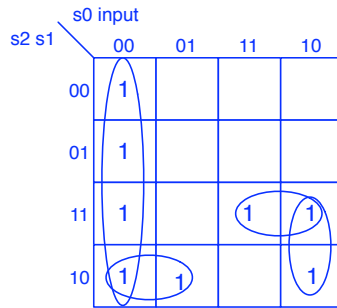
s2	s1	s0	input	s2 ⁺	s1 ⁺	s0 ⁺
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	0	0
0	1	1	0	1	0	0
0	1	1	1	0	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	0	1	1	1	1	0
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1



$$s2^+ = in \cdot s2 \cdot \overline{s0} + s0 \cdot s2 \cdot s1 + \overline{input} \cdot s2 \cdot s1 + s1 \cdot s0 \cdot \overline{input}$$



$$s1^+ = s1 \cdot \overline{s0} \cdot \overline{input} + s2 \cdot \overline{s0} \cdot \overline{input} + s2 \cdot s0 \cdot input + s2 \cdot s1 \cdot s0 + \overline{s2} \cdot \overline{s1} \cdot s0 \cdot \overline{input}$$



$$s0^+ = \overline{s0} \cdot \overline{input} + \overline{s0} \cdot s2 \cdot \overline{s1} + s0 \cdot s2 \cdot s1 + s2 \cdot s1 \cdot \overline{in}$$

Because we are designing with D flip-flops, the flip flop input is simply the next value to be stored there.

We then design the output logic. The output, *locked*, is 0 only in the last state which is encoded $s2 = 1, s1 = 1, s0 = 1$, thus $locked = \overline{s2s1s0}$

In the interest of distributing these solutions before the midterm we skip the gate-level schematic drawing and show instead the block-level schematic which indicates how the two blocks of next state logic interact.

