Special Counter Design based on questions in exams in 2014 and 2015

1.1 Mr. Trojan says that the 3-bit counter has 9 states instead of 8 states and repeats one of the states a second time. Analyze the schematic M#1 and the RTL state diagram M#2 and find the sequence of the 9 states. Note that even though the 9-state is same in both, the Flag Flip-Flop behavior is slightly different.

Sequence of M#1: 0, 1, 2, $3$, $4$, $4$, $5$, $6$, $7$, $0$. Flip-Flop F is set for 2 clocks.

Sequence of M#2: 0, 1, 2, $3$, $4$, $4$, $5$, $6$, $7$, $0$. Flip-Flop F is set for 1 clocks.

2.1.1 Based on the above design, build a special 3-bit counter which repeats 3, 4 as shown in the sequence (0, 1, 2, 3, 4, 3, 4, 5, 6, 7). Complete the following designs. First complete the state diagram. Then implement the NSL and OFL logic for this 2-state state machine using the encoded state assignment method of EE101.

1.2 Complete the state diagram below for a similar 3-bit special down counter with no enable, but it repeats 5, 4, 3, two more times as shown in (7, 6, 5, 4, 3, 5, 4, 3, 2, 1, 0). Reset asynchronously initializes to 7 and $FIFO$ to 00.
3.2 A special 3-bit counter Q is built, by using two other 2-bit counters T and S as per the state diagram on the side. Instead of drawing waveforms, fill-up the count sequence table recording the values of the three counters, Q, T, and S.

Notice that the 4 repeated Q times, 1-time, 2 times, and then three times.

38 clock pattern repeats

| Clock | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Q     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
| T     | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| S     | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |

2.2 You are aware of the Verilog code #1 for a counter with clear, load, and enable controls where CLR (clear) has the highest (highest/lowest) priority and the EN (enable) has the lowest (highest/lowest) priority.

Here the CLR control is synchronous (synchronous/asynchronous).
The codes #2 and #3 are similar to code #1 but these two codes use 3 separate if statements.
Only #3 (#2/#3) behaves like code #1.
It is A (A/B/C) to uncomment the last two commented lines in code #1.
It is C (A/B/C)) to uncomment the last two commented lines in the code #2 or #3 chosen by you.
A. OK though not needed.
B. These lines must be uncommented.
C. These lines should never be uncommented.

always @(posedge CLK) begin : COUNTER
begin : COUNTER
begin : COUNTER
if (CLR)
    Count <= 3'b000;
else if (LOAD)
    Count <= Load_value;
else if (EN)
    Count <= Count +1;
//else
// Count <= Count;
end
always @(posedge CLK)
begin : COUNTER
begin : COUNTER
begin : COUNTER
if (CLR)
    Count <= 3'b000;
if (LOAD)
    Count <= Load_value;
if (EN)
    Count <= Count +1;
//else
// Count <= Count;
end
always @(posedge CLK)
begin : COUNTER
begin : COUNTER
begin : COUNTER
if (EN)
    Count <= Count +1;
if (LOAD)
    Count <= Load_value;
if (CLR)
    Count <= 3'b000;
//else
// Count <= Count;
end