Counter Design

Design a 3-bit counter that counts in binary or in Gray code, depending on the values of a “mode control input”. This synchronous 3-bit counter has a mode control input m. As long as m = 0, the counter steps through the binary sequence 000, 001, 010, 011, 100, 101, 110, 111 and repeats this sequence. As long as m = 1, the counter advances through the Gray code sequence 000, 001, 011, 010, 110, 111, 101, 100 and repeats this sequence. However, it is possible that the mode input changes at any time in any of these sequences.

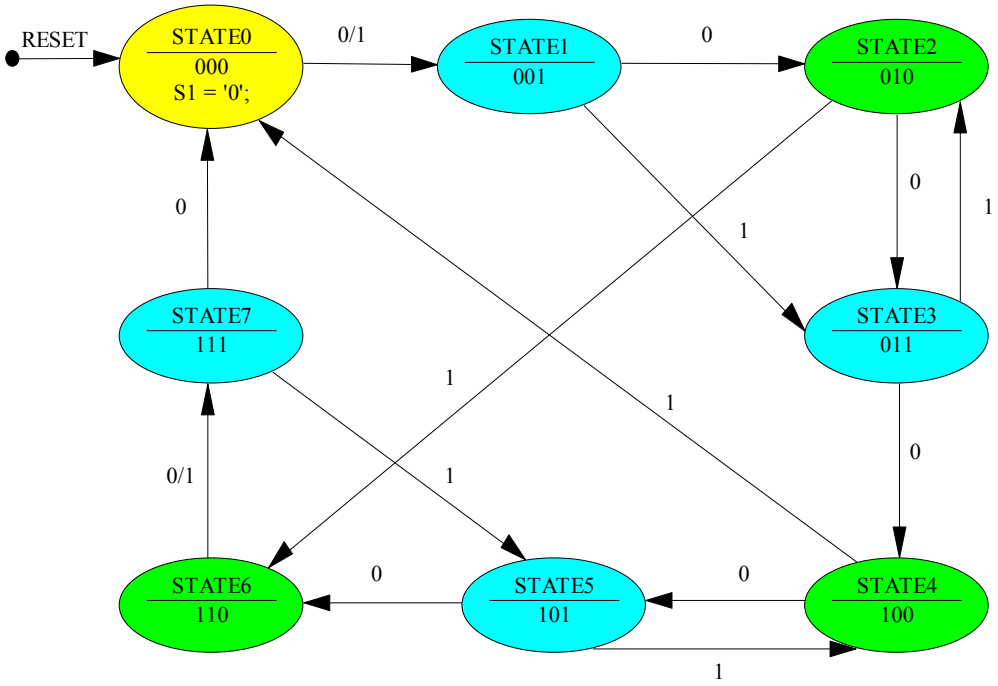
For example, assume that the mode input is 0 for the first two clock cycles but changes to 1 in the third clock cycle and stays at 1 in the fourth clock cycle. (That is, m goes through the sequence 0, 0, 1, 1.) Then, the output will be 000, 001, 010, 110, 111. In this example, 000 is the initial state. The first two state transitions (000->001->010) occurs in the binary counting mode. Then, because m = 1 from the third clock cycle onwards, the state goes from 010->110->->111, as indicated in the Gray code sequence.

In addition to the mode control input, there is a reset input. This synchronous counter should go to the 000 state if asserted.

a, State the inputs and outputs:

```
input m, reset, clock;
output [2:0] count;
reg [2:0] state;
```

b, Implement the counter as a Moore machine and draw the state diagram



c, Implement this counter as a Verilog module.

```
module GrayCounter(Clock, m, reset, count);

  input Clock, m, reset;
  output [2:0] count;
  reg [2:0] state;

  parameter S0 = 3'b000;
  parameter S1 = 3'b001;
  parameter S2 = 3'b010;
  parameter S3 = 3'b011;
  parameter S4 = 3'b100;
  parameter S5 = 3'b101;
  parameter S6 = 3'b110;
  parameter S7 = 3'b111;

  assign count = state;

  always @ (posedge Clock)
    if (reset)
      state<=S0;
    else
      begin
        case (state)
          S0 : state <= S1;
          S1 : state <= m? S3:S2;
          S2 : state <= m? S6:S3;
          S3 : state <= m? S2:S4;
          S4 : state <= m? S0:S5;
          S5 : state <= m? S4:S6;
          S6 : state <= S7;
          S7 : state <= m? S5:S0;
        endcase
      end
endmodule
```

State Reduction

Reduce the following state table using an implication chart.

PS	NS, S	
	x=0	x=1
A	F, 0	B, 1
B	G, 0	A, 1
C	B, 0	C, 1
D	C, 0	B, 1
E	D, 0	A, 1
F	E, 1	F, 1
G	E, 1	G, 1

Pass I:

B	F-G					
C	B-F C-B	B-G A-C				
D	C-F	C-G A-B	B-C			
E	D-F A-B	D-G	B-D A-C	C-D A-B		
F	X	X	X	X	X	
G	X	X	X	X	X	√
	A	B	C	D	E	F

Elimination of differing outputs: AF, AG, BF, BG, CF, CG, DF, DG, EF, EG.
Equivalent: FG

Pass II:

B	F-G					
C	X	X				
D	X	X	X			
E	X	X	X	X		
F	X	X	X	X	X	
G	X	X	X	X	X	√
	A	B	C	D	E	F

Elimination: AC, AD, AE, BC, BD, BE, CD, CE.

AB are equivalent.

Reduced to 5 states: AB, C, D, E, FG.

Write the Verilog code for a 4-bit up counter.

```
module fourbit_upcount(Reset, Clock, E, Q);
  input Reset, Clock, E;
  output [3:0] Q;
  reg [3:0] Q;

  always @ (posedge Clock)
    if (!Reset)
      Q <= 0;
    else if (E)
      Q <= Q+1;

endmodule
```