Solutions to CSE3201 Assignment 4

1. 5.21 (From the textbook, as with the remainder of the questions)

Ans. in textbook

2. **5.22** (Just do the code and only show what's in the always block, absolutely no structural code, use only non-blocking assignments.)

```
module johnson8 (Resetn, Clock, Q);
    input Resetn, Clock;
    output reg [7:0] Q;
    reg [7:0] Q;

    always @(negedge Resetn, posedge Clock)
        if (!Resetn)
            Q <= 0;
        else
            Q <= {{Q[6:0]}, {~Q[7]}};

endmodule
```

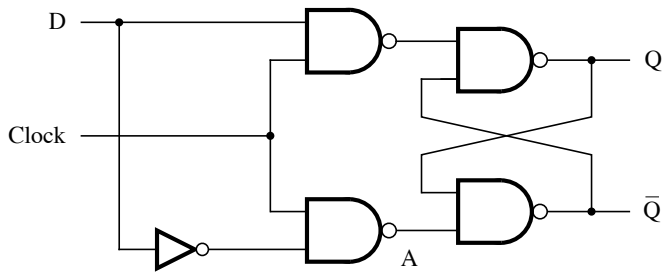
3. 5.23

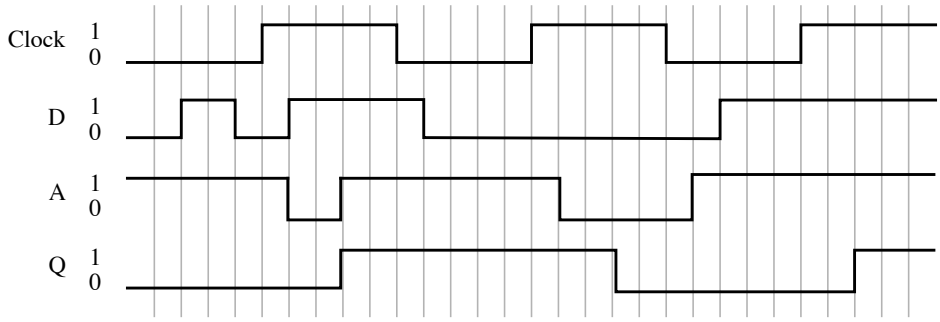
```
// Ring counter with synchronous reset
module ripplen (Resetn, Clock, Q);
    parameter n = 8;
    input Resetn, Clock;
    output reg [n-1:0] Q;

    always @(posedge Clock)
        if (!Resetn)
        begin
            Q[7:1] <= 0;
            Q[0] <= 1;
        end
        else
            Q <= {{Q[6:0]}, {Q[7]}};

endmodule
```

4. 5.25

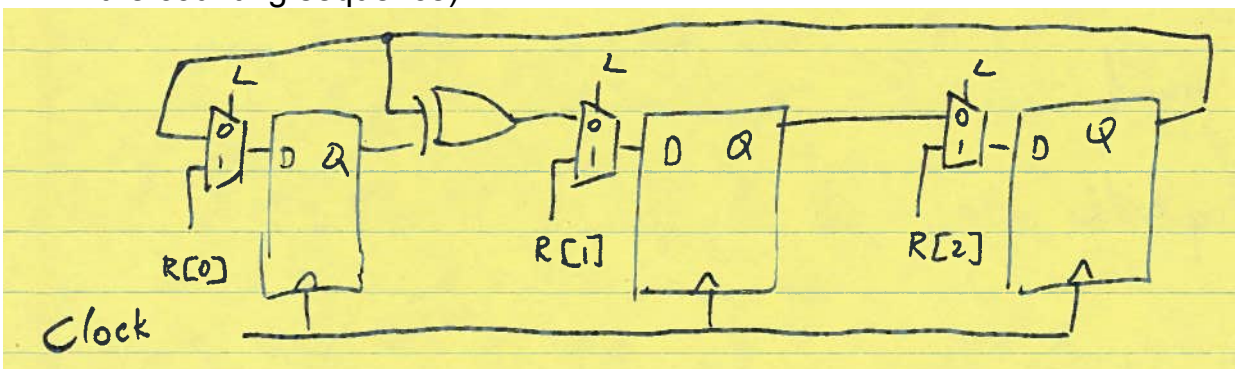




5. 5.26

Ans. in textbook

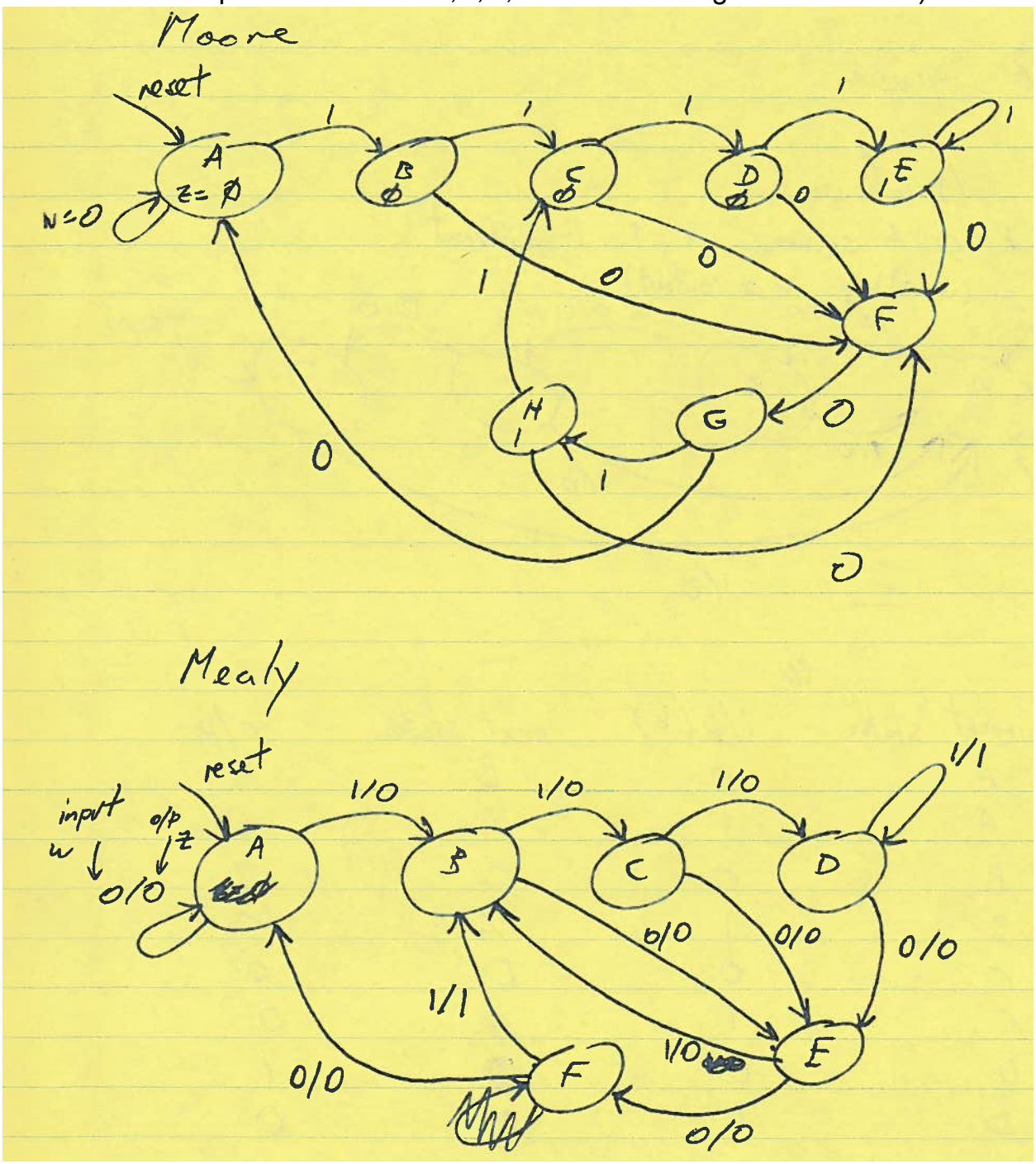
6. **5.28** (Just draw the diagram after carefully looking at the code and show the counting sequence).



7. 6.1 (Note: the book writes its state assignment table a little differently than we have, it denotes the inputs with the variable w and places different settings for it in different columns, our notation made it look just like a traditional truth table).

Ans. in textbook

8. **6.3** (Just draw the state transition diagram – no tables please –, label your states in alphabetical order A,B,C, etc. with A being the reset state).



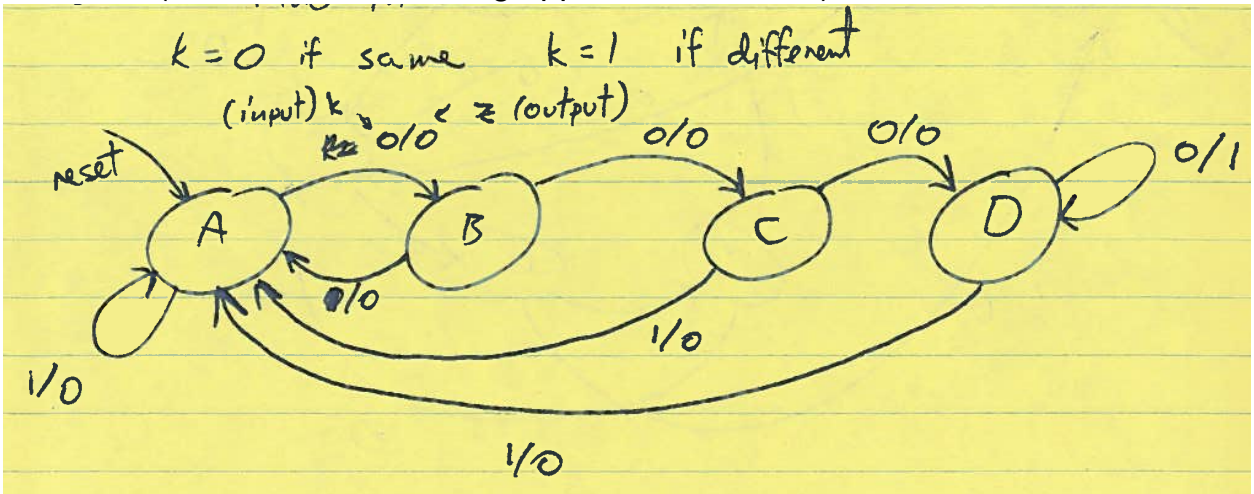
9. 6.5

Ans. in textbook

10. 6.6

Ans. in textbook

11. 6.9 (Use the same labeling approach as for 6.3)

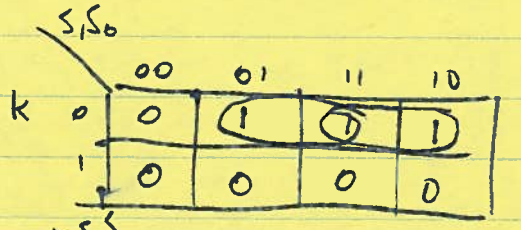


current state	i/p (k)	next state	o/p
A	0	B	0
A	1	A	0
B	0	C	0
B	1	A	0
C	0	D	0
C	1	A	0
D	0	D	1
D	1	A	0

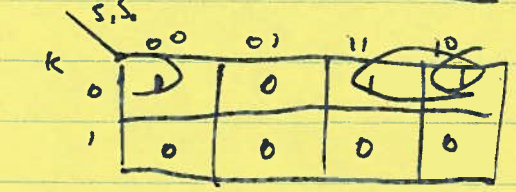
	S_1 S_0	
A	0 0	mapping states to bits
B	0 1	
C	1 0	
D	1 1	

Mealy State transition & O/P table

S_1 S_0 current state	k	S_1' S_0' next state	$Y = z$ o/p
0 0	0	0 1	0
0 0	1	0 0	0
0 1	0	1 0	0
0 1	1	0 0	0
1 0	0	1 1	0
1 0	1	0 0	0
1 1	0	1 1	1
1 1	1	0 0	0



$$S_1' = S_0\bar{k} + S_1\bar{k}$$



$$S_0' = \bar{S}_0\bar{k} + S_1\bar{k}$$

$$Y = S_1 S_0 \bar{k}$$

12. 6.10

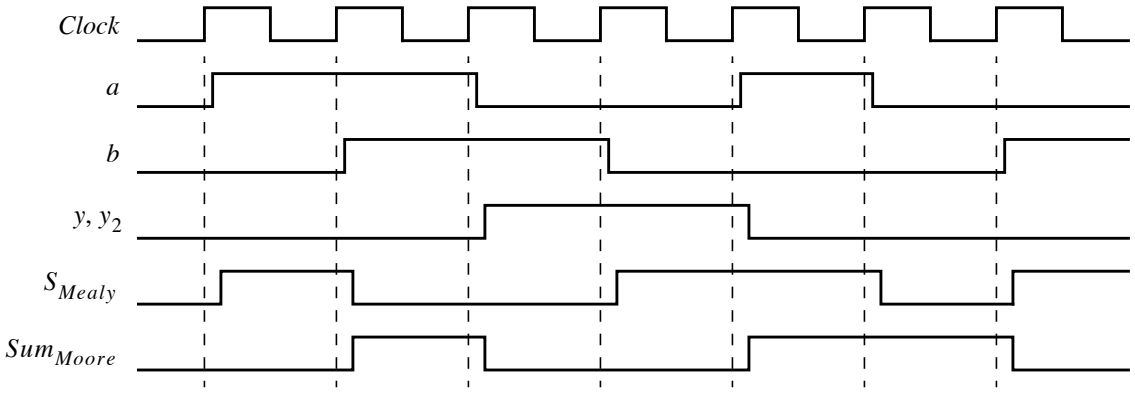
```
module  prob8_10 (Clock, Resetn, w1, w2, z);
    input  Clock, Resetn, w1, w2;
    output  reg  z;
    reg  [2:1] y, Y;
    wire  k;
    parameter  [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

    // Define the next state and output combinational circuits
    assign k = w1 ^ w2;
    always @(k, y)
        case (y)
            A:  if (k)   begin
                             Y = A;   z = 0;
                         end
                else     begin
                             Y = B;   z = 0;
                         end
            B:  if (k)   begin
                             Y = A;   z = 0;
                         end
                else     begin
                             Y = C;   z = 0;
                         end
            C:  if (k)   begin
                             Y = A;   z = 0;
                         end
                else     begin
                             Y = D;   z = 0;
                         end
            D:  if (k)   begin
                             Y = A;   z = 0;
                         end
                else     begin
                             Y = D;   z = 1;
                         end
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if  (Resetn == 0)   y <= A;
        else   y <= Y;

endmodule
```

13. 6.14



14. 6.15

Ans. in textbook

15. 6.20

present counter state | next counter state | output

present counter state	next counter state	output
0	1	0
1	2	2
2	3	1
3	0	3

input w can be the clock

map to bits

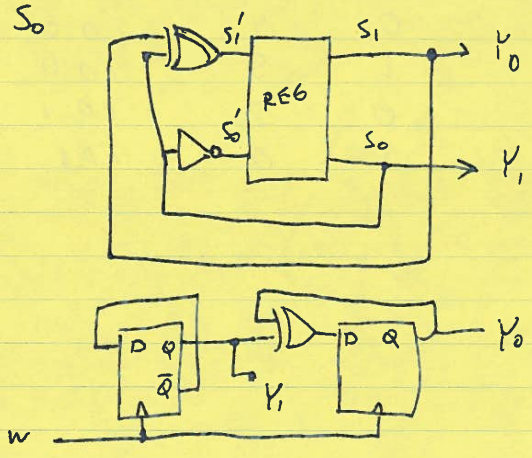
S_1 S_0	S_1' S_0'	Y_1 Y_0
0 0	0 1	0 0
0 1	1 0	1 0
1 0	1 1	0 1
1 1	0 0	1 1

$S_1' = S_1 \oplus S_0$

$S_0' = \overline{S_0}$

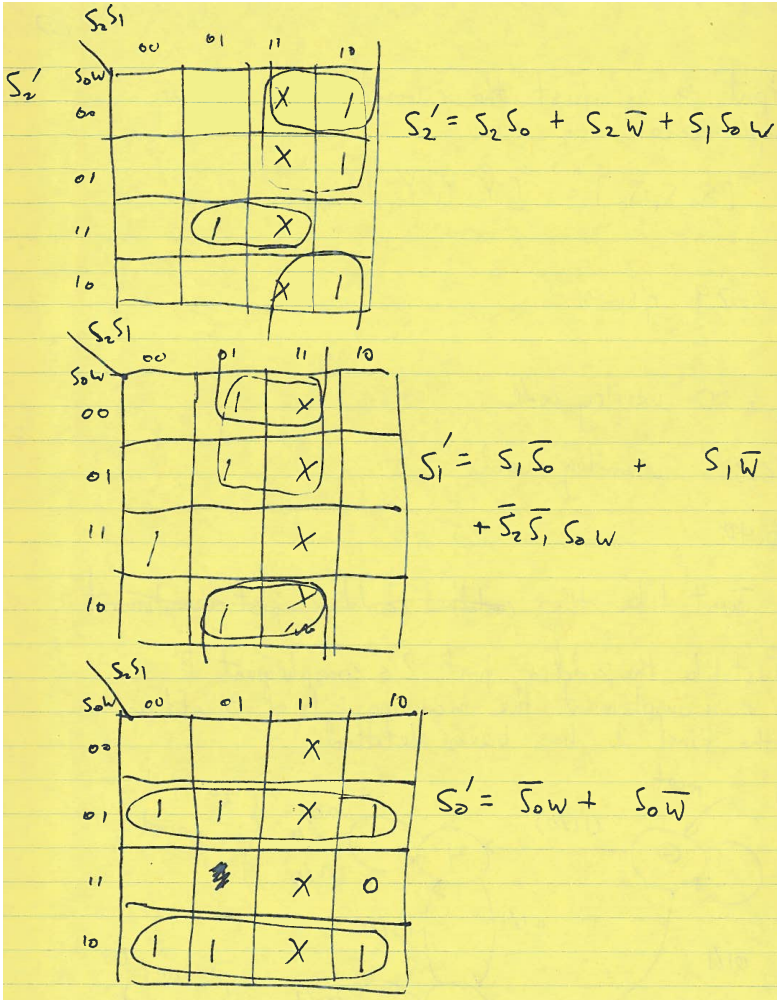
$Y_1 = S_0$

$Y_0 = S_1$



16. 6.23

present state	input	next state	$S_2 S_1 S_0$	w	$S_2' S_1' S_0'$	
0	0	0	000	0	000	0
0	1	1	000	1	001	1
1	0	1	001	0	001	2
1	1	2	001	1	010	3
2	0	2	010	0	010	4
2	1	3	010	1	011	5
3	0	3	011	0	011	6
3	1	4	011	1	100	7
4	0	4	100	0	100	8
4	1	5	100	1	101	9
5	0	5	101	0	101	10
5	1	0	101	1	000	11



$S_2' = S_2 S_0 + S_2 \bar{w} + S_1 S_0 w$

$S_1' = S_1 \bar{S_0} + S_1 \bar{w} + \bar{S_2} \bar{S_1} S_0 w$

$S_0' = \bar{S_0} w + S_0 \bar{w}$

output → is just the current state in this simple case

$[S_2 S_1 S_0] = [Y_2 Y_1 Y_0]$

17. 6.29

Ans. in textbook

18. 6.30

```
module  prob8_30 (Clock, Resetn, D, N, z);
    input  Clock, Resetn, D, N;
    output  z;
    reg  [3:1] y, Y;
    wire  [1:0] K;
    parameter  [3:1] S1 = 3'b000, S2 = 3'b001, S3 = 3'b010, S4 = 3'b011, S5 = 3'b100;

    // Define the next state combinational circuit
    assign K = {D, N};
    always @(K, y)
        case (y)
            S1: if (K == 2'b00)   Y = S1;
                else if (K == 2'b01)   Y = S3;
                else if (K == 2'b10)   Y = S2;
                else   Y = 3'bxxx;
            S2: if (K == 2'b00)   Y = S2;
                else if (K == 2'b01)   Y = S4;
                else if (K == 2'b10)   Y = S5;
                else   Y = 3'bxxx;
            S3: if (K == 2'b00)   Y = S3;
                else if (K == 2'b01)   Y = S2;
                else if (K == 2'b10)   Y = S4;
                else   Y = 3'bxxx;
            S4: if (K == 2'b00)   Y = S1;
                else   Y = 3'bxxx;
            S5: if (K == 2'b00)   Y = S3;
                else   Y = 3'bxxx;
            default:   Y = 3'bxxx;
        endcase

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if  (Resetn == 0)   y <= S1;
        else   y <= Y;

    // Define output
    assign  z = (y == S4) | (y == S5);

endmodule
```

19. 6.31

```
module prob8_31 (Clock, Resetn, D, N, z);
   input Clock, Resetn, D, N;
   output z;
   reg [3:1] y;
   wire [1:0] K;
   parameter [3:1] S1 = 3'b000, S2 = 3'b001, S3 = 3'b010, S4 = 3'b011, S5 = 3'b100;

   assign K = {D, N};
   // Define the sequential block
   always @(negedge Resetn, posedge Clock)
      if (Resetn == 0)  y <= S1;
      else
         case (y)
            S1: if (K == 2'b00)  y <= S1;
                else if (K == 2'b01)  y <= S3;
                else if (K == 2'b10)  y <= S2;
                else  y <= 3'bxxx;
            S2: if (K == 2'b00)  y <= S2;
                else if (K == 2'b01)  y <= S4;
                else if (K == 2'b10)  y <= S5;
                else  y <= 3'bxxx;
            S3: if (K == 2'b00)  y <= S3;
                else if (K == 2'b01)  y <= S2;
                else if (K == 2'b10)  y <= S4;
                else  y <= 3'bxxx;
            S4: if (K == 2'b00)  y <= S1;
                else  y <= 3'bxxx;
            S5: if (K == 2'b00)  y <= S3;
                else  y <= 3'bxxx;
            default:  y <= 3'bxxx;
         endcase

   // Define output
   assign z = (y == S4) | (y == S5);

endmodule
```

20. 6.40