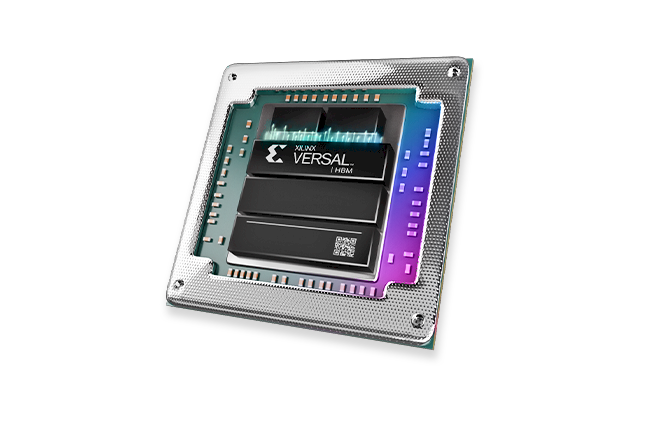
**EC204**

**Digital System Design Lab**

**Lab – 6**



**Utkarsh R Mahajan**

**201EC164**

1] Design a synchronous modulo-5 counter using D FF

**->** To design a synchronous modulo-5 counter, we will need 3 D Flip flops. Since it’s a synchronous counter, one clock will be connected to clock input of all D flip flops.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qan | Qbn | Qcn | Qan+1 | Qbn+1 | Qcn+1 | Da | Db | Dc |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Using the above table for Qn+1 and D.

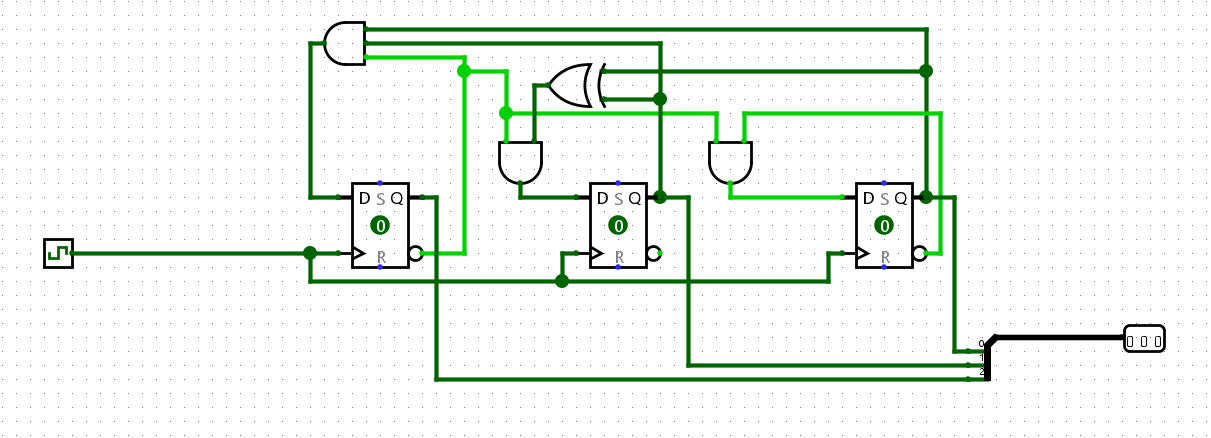
We get the following state equations,

Da= Qa’QbQc

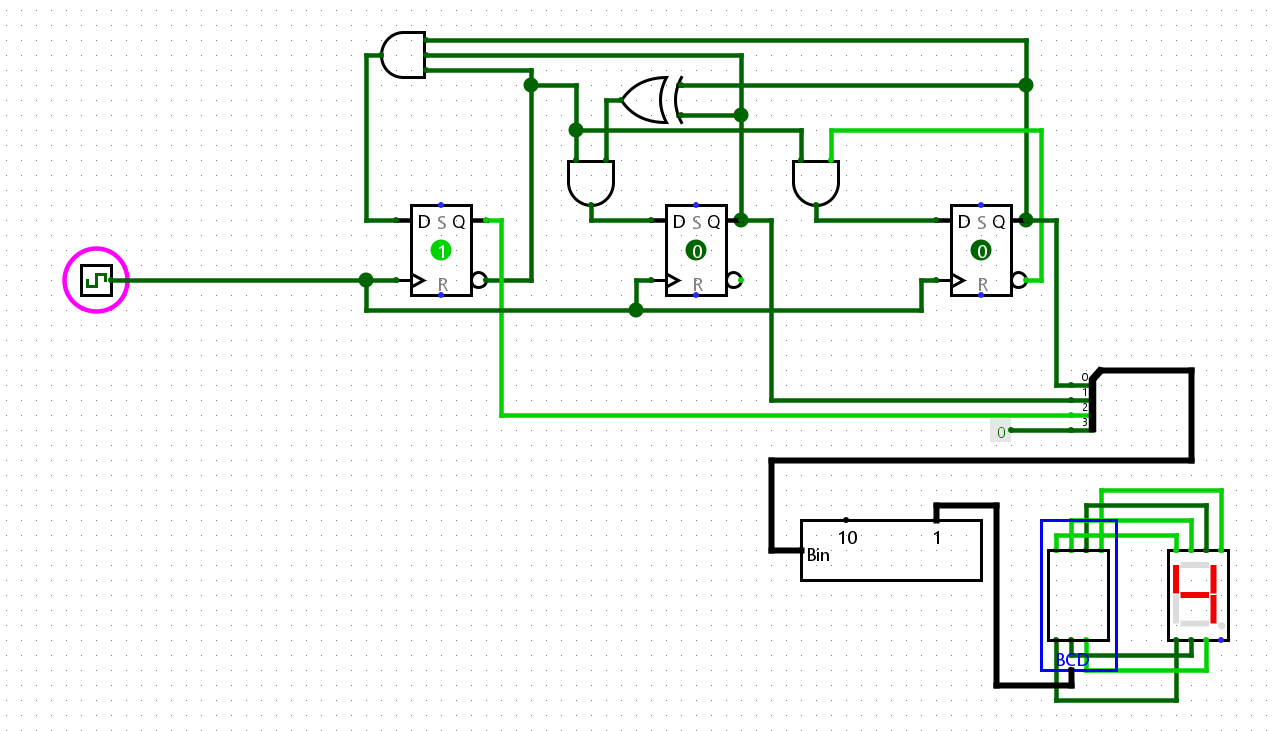
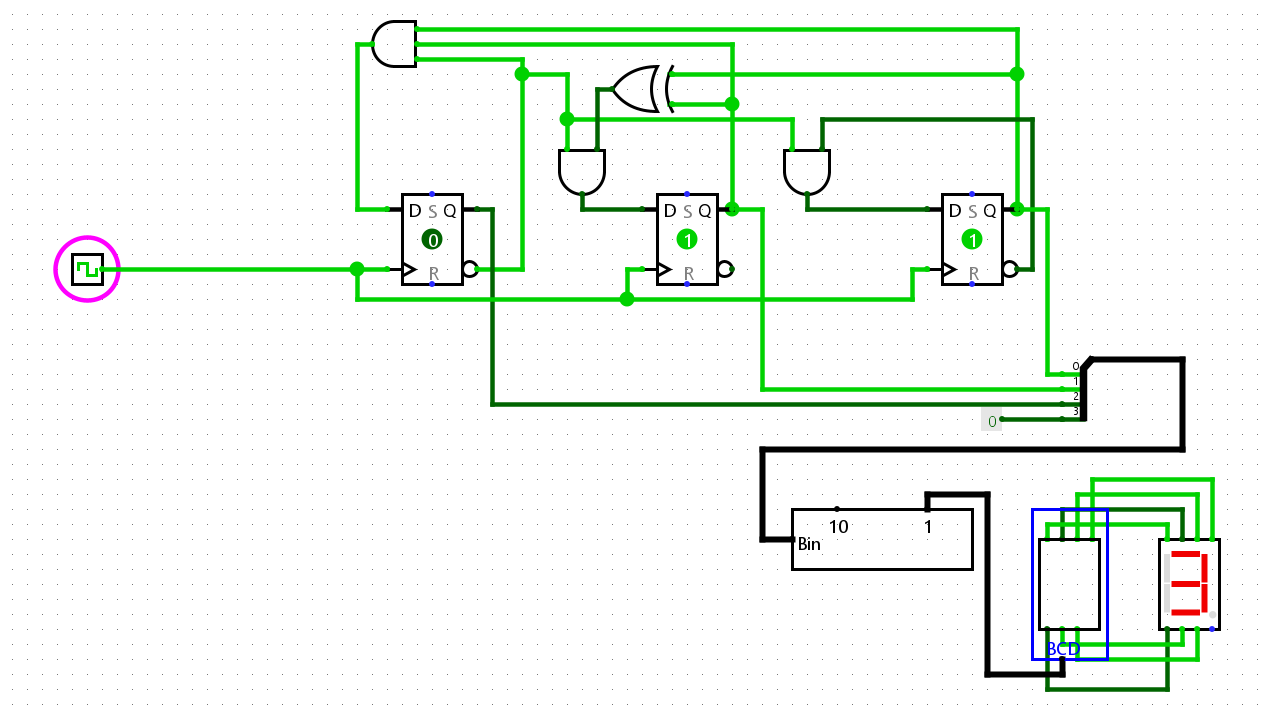
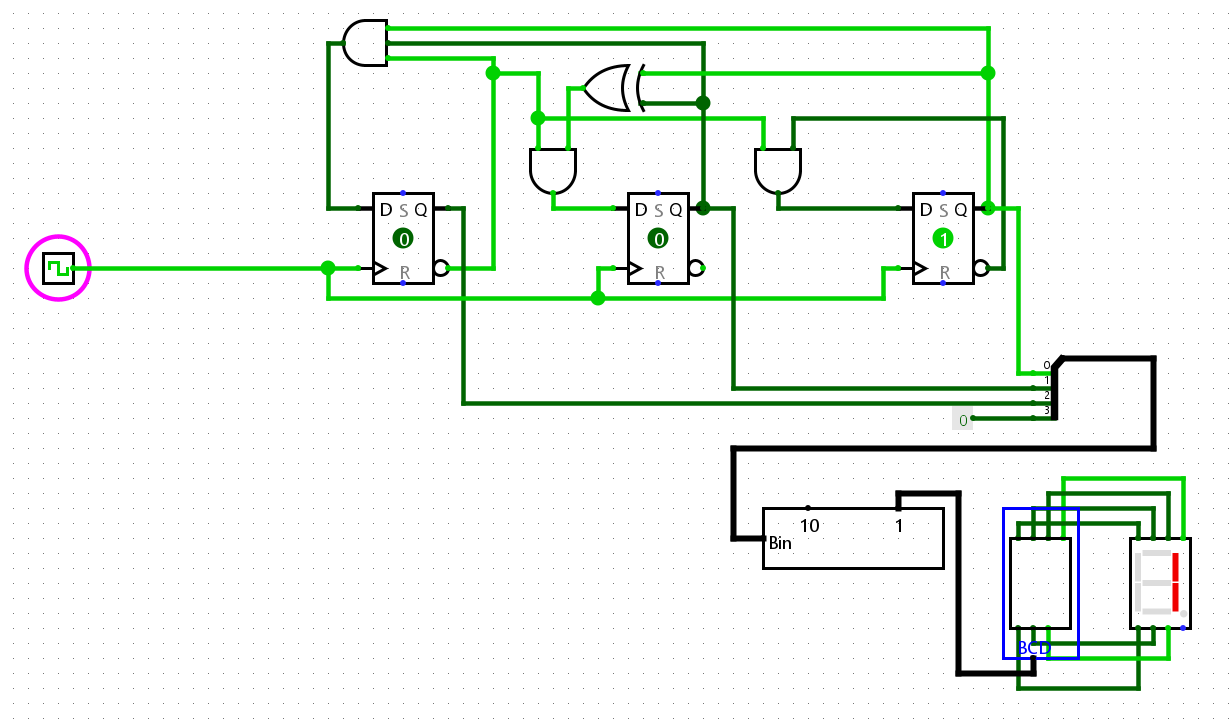
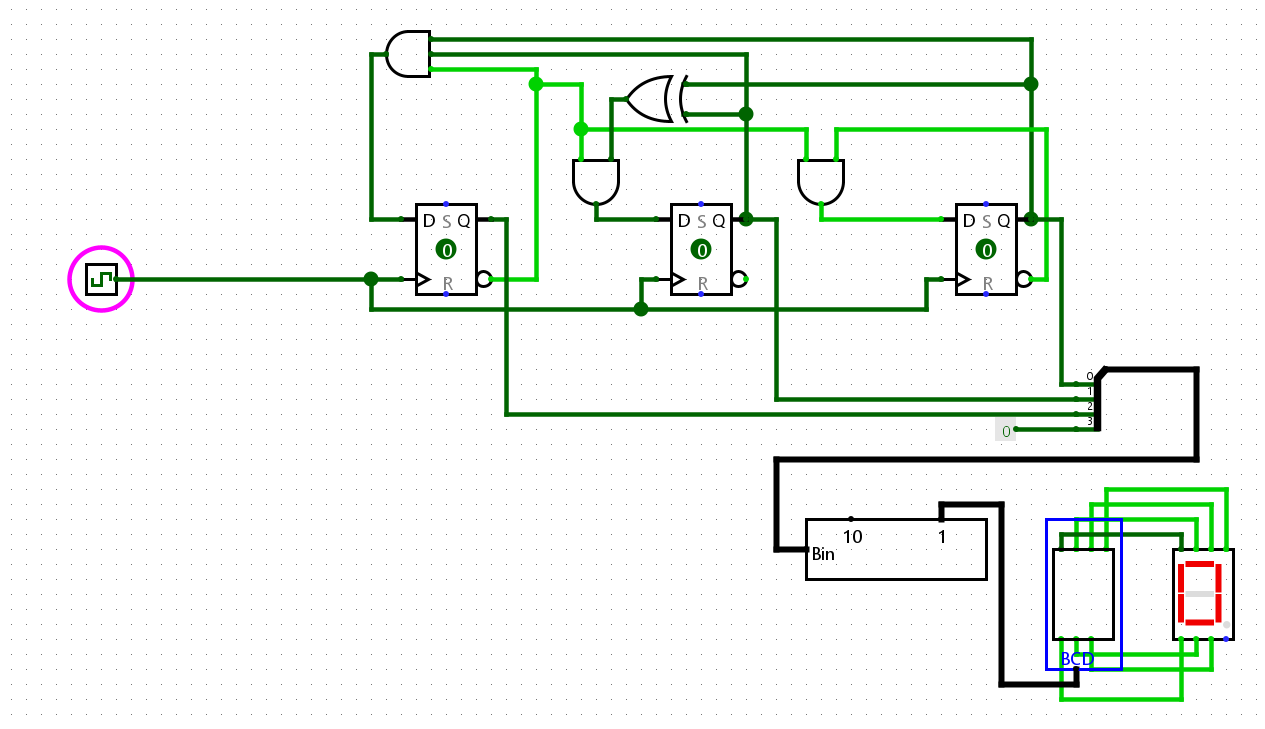
Db= Qa’Qb’Qc+Qa’QbQc’

Dc= Qa’Qc’

Connection following conditions to respective D input of flipflops we finally get the respective output bits at Q of each flip flop.

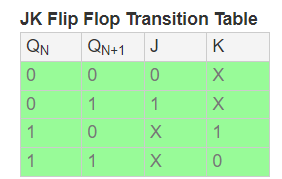
****

Connecting a binary to BCD decoder and BCD to seven segment decoder and seven segment display, we can see the following outputs.



2] Design a synchronous 4 bit up-down counter using JK FF

-> To design a synchronous modulo-7 up down counter, we will need 4 JK Flip flops. Since it’s a synchronous counter, one clock will be connected to clock input of all JK flip flops.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | Qan | Qbn | Qcn | Qdn | Qan+1 | Qbn+1 | Qcn+1 | Qdn+1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Using the above State table and JK flip flop transition table.

We get the following state equations,

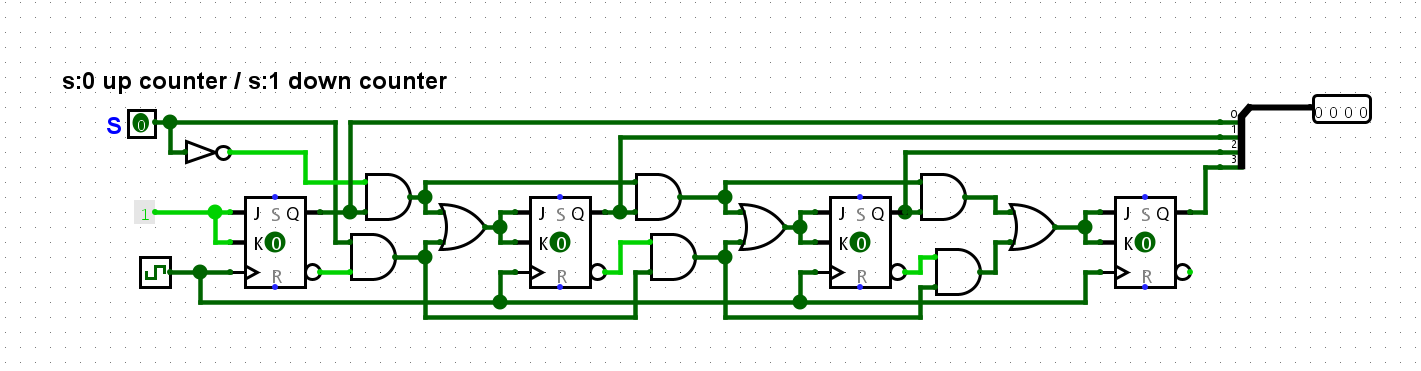
Ja=Ka=1

Jb=Kc=S’Qa+SQa’

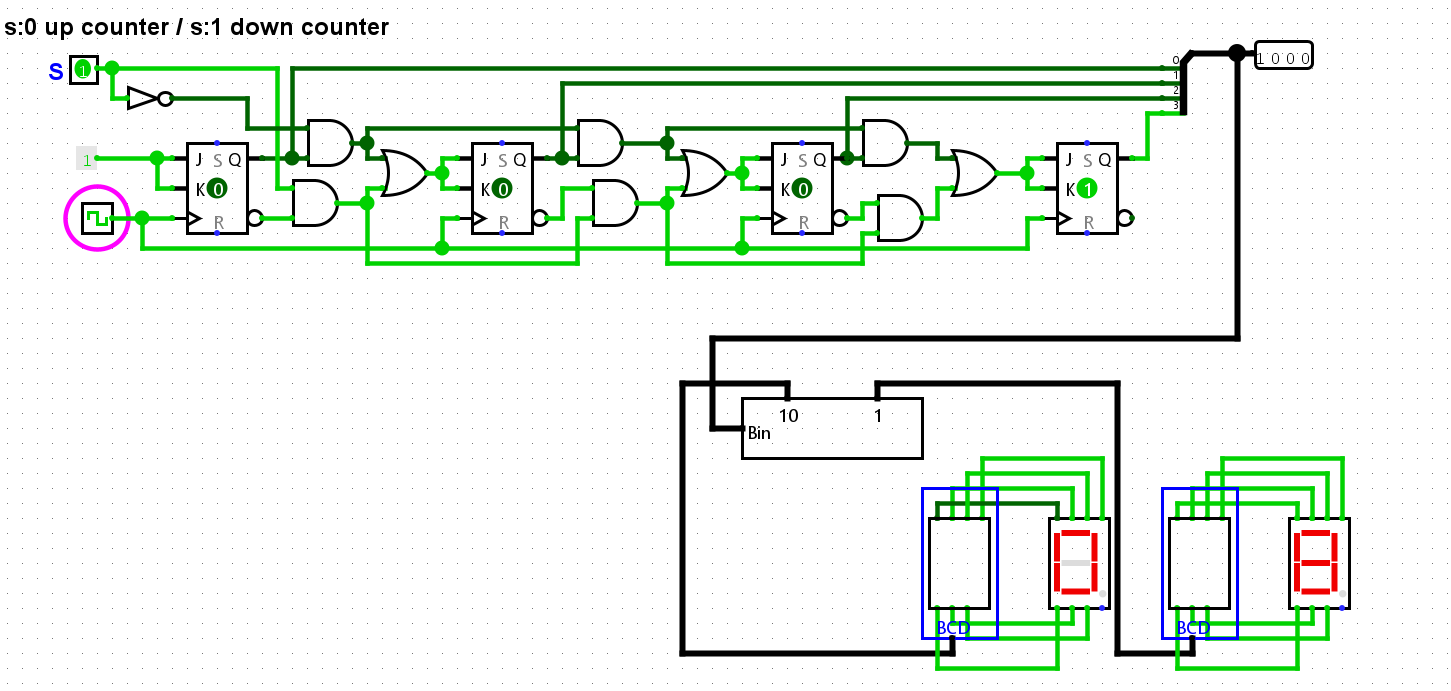
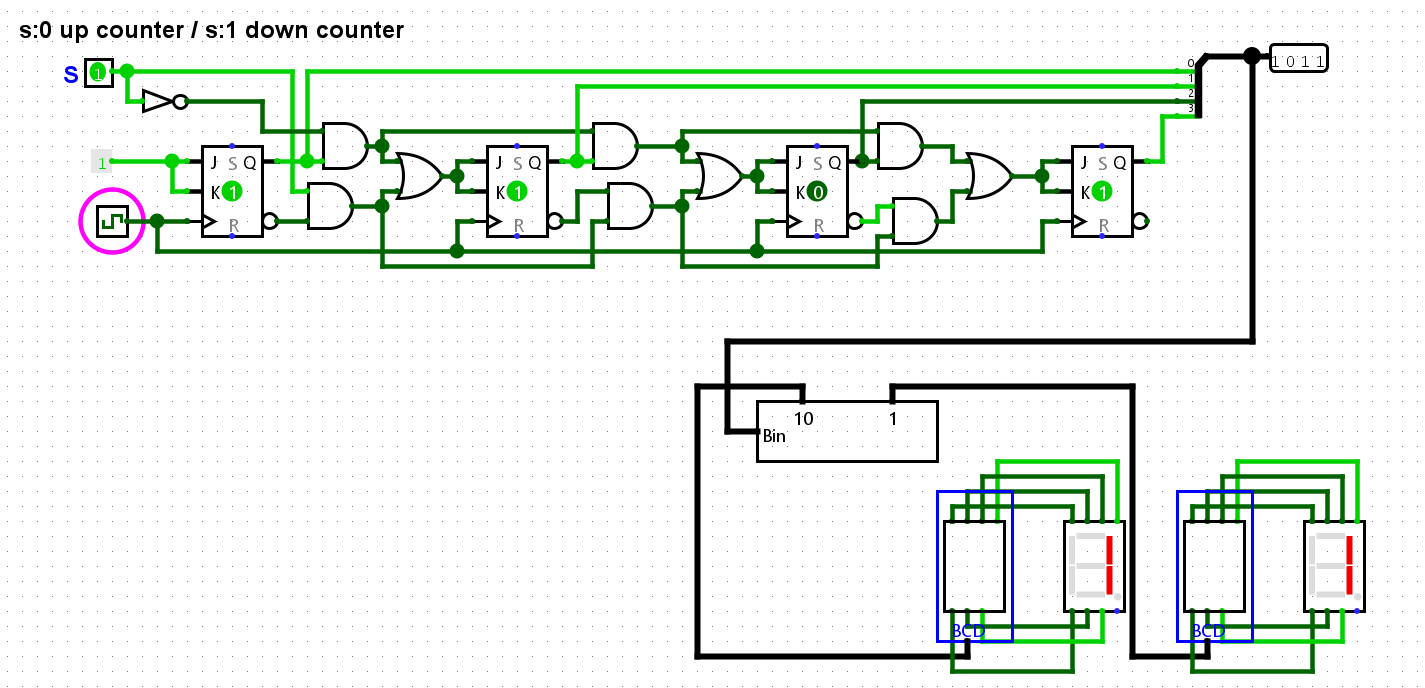
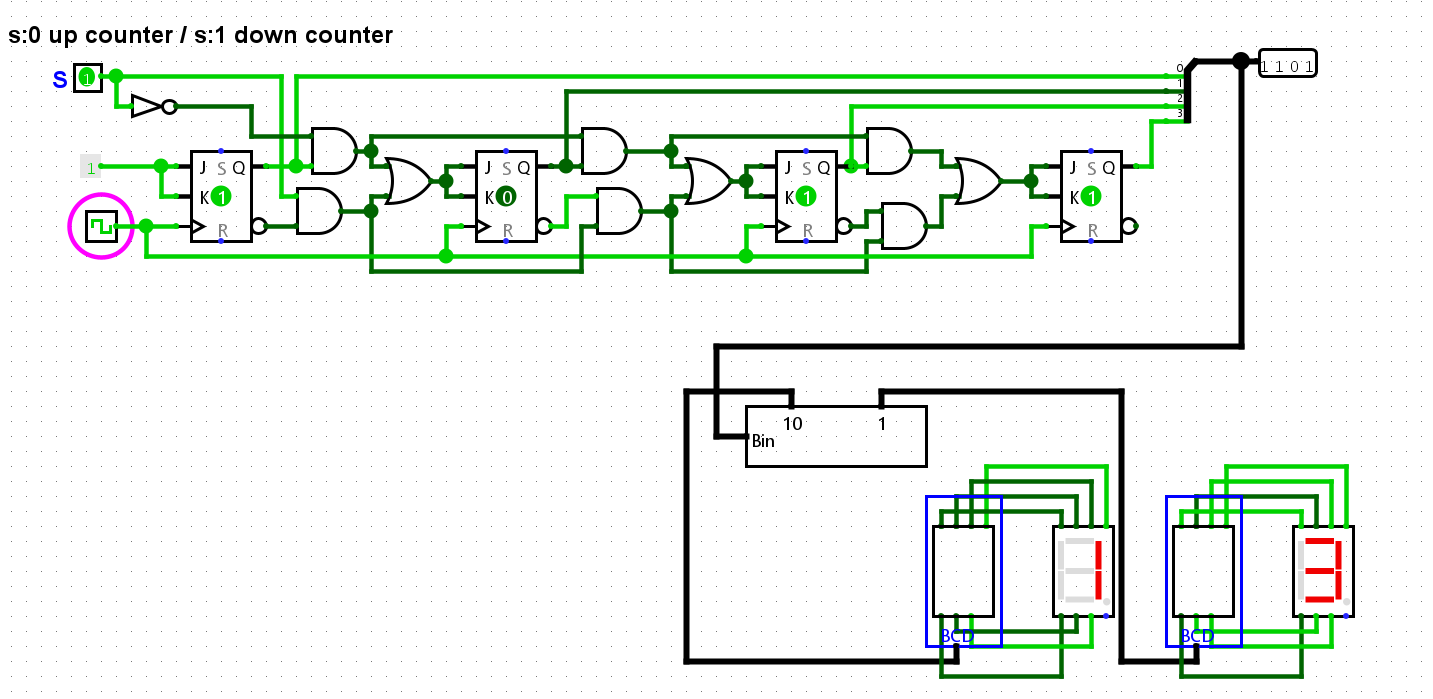
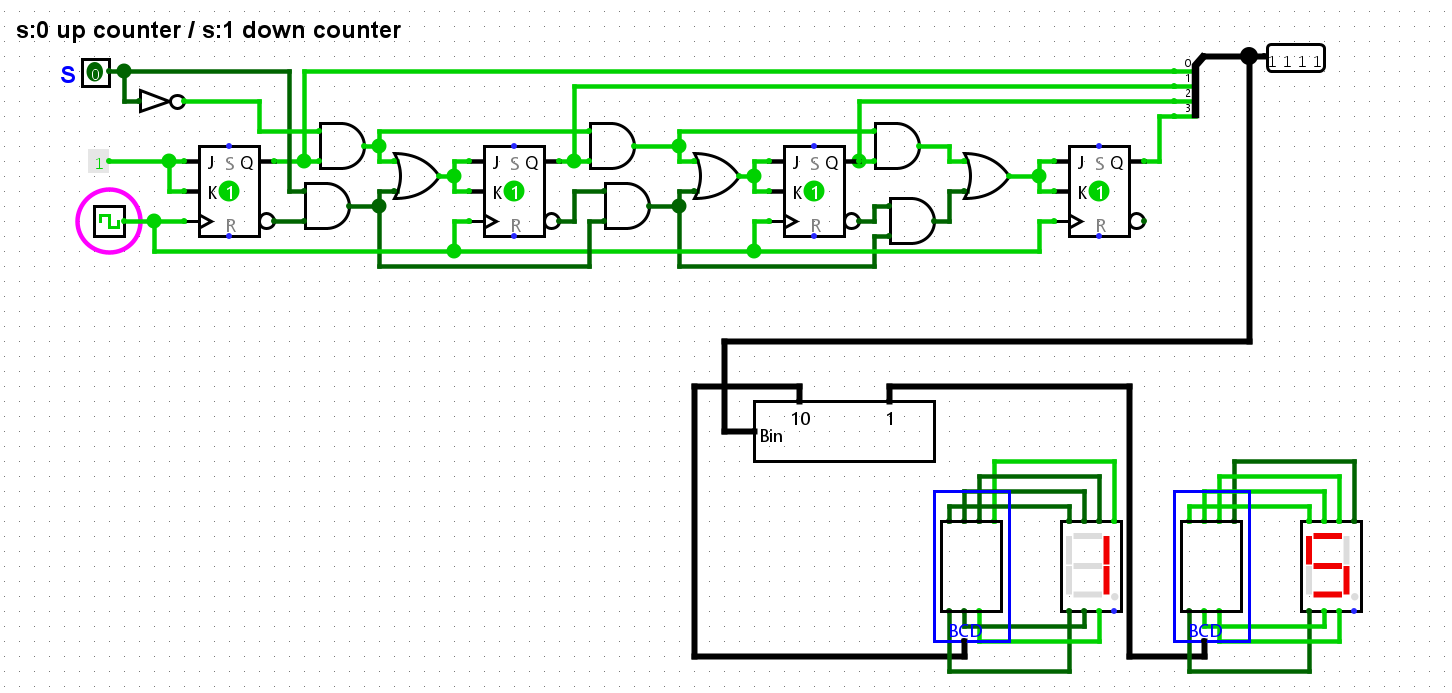
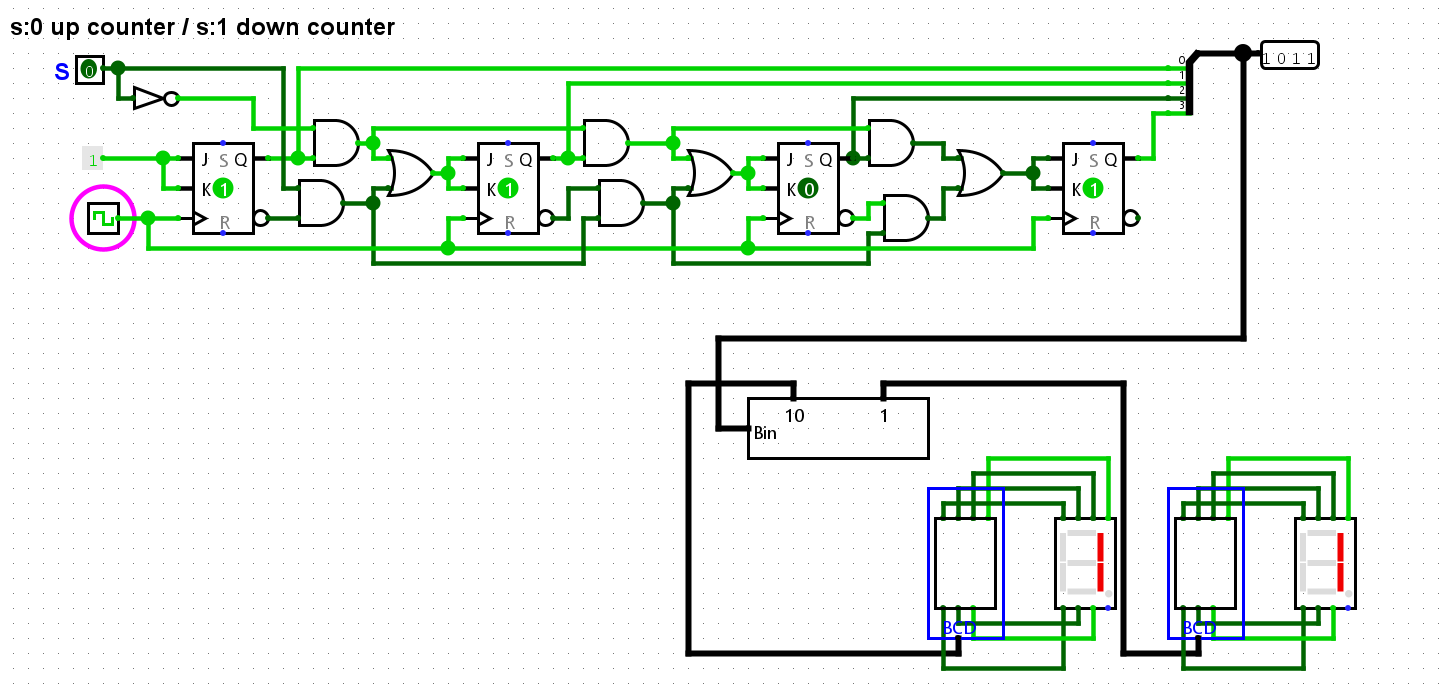
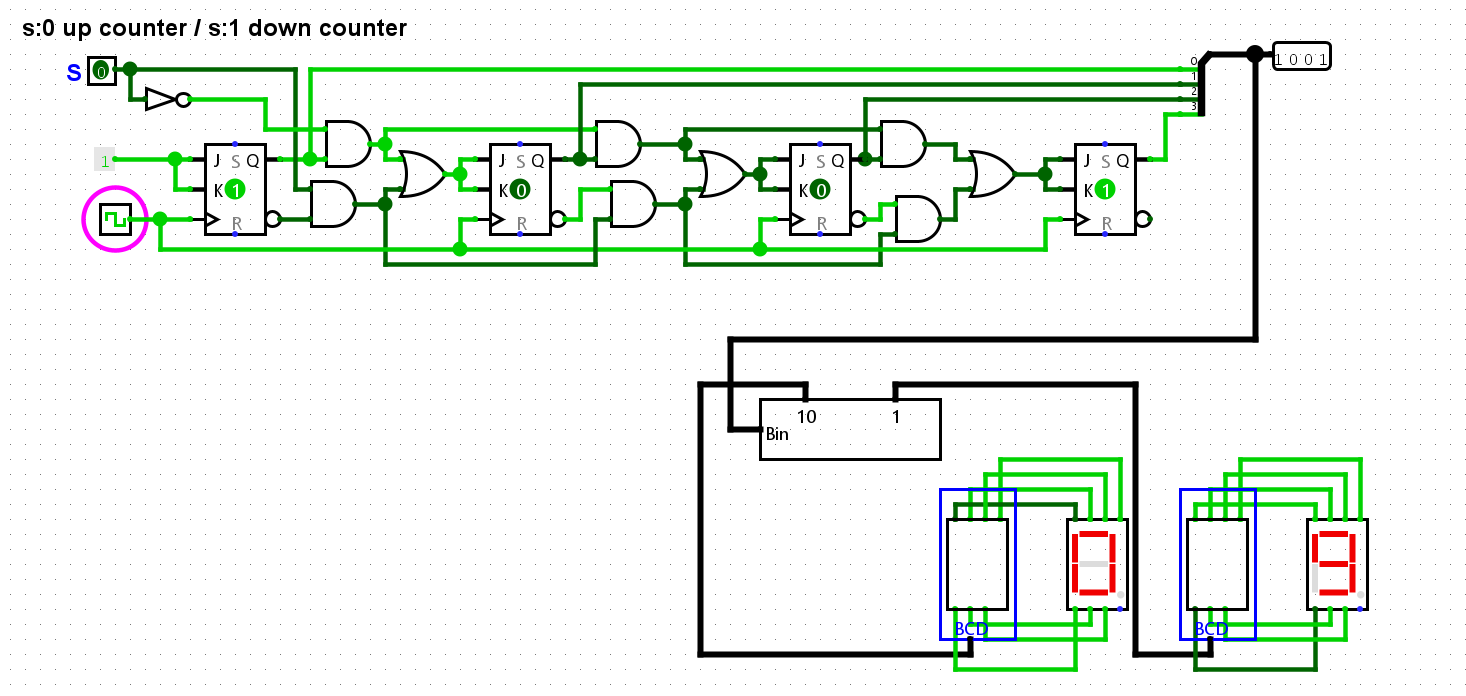
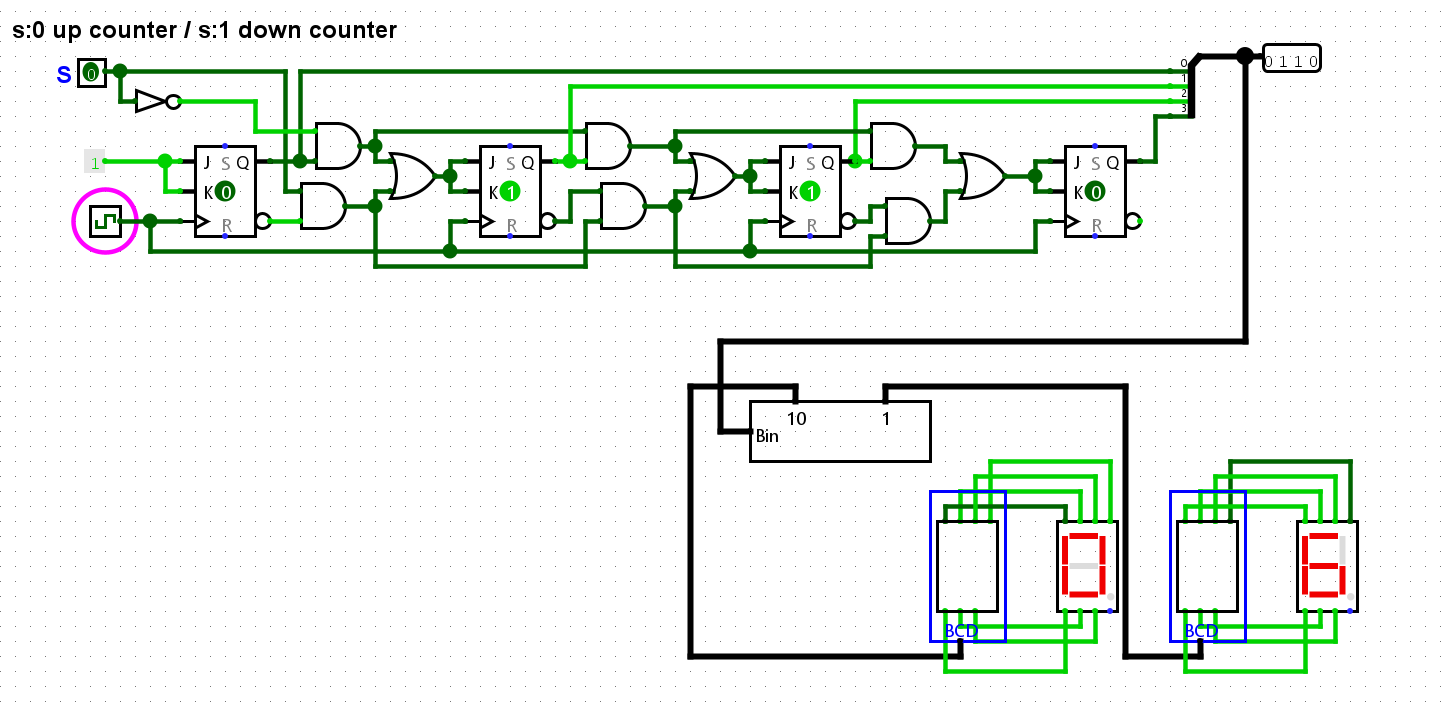
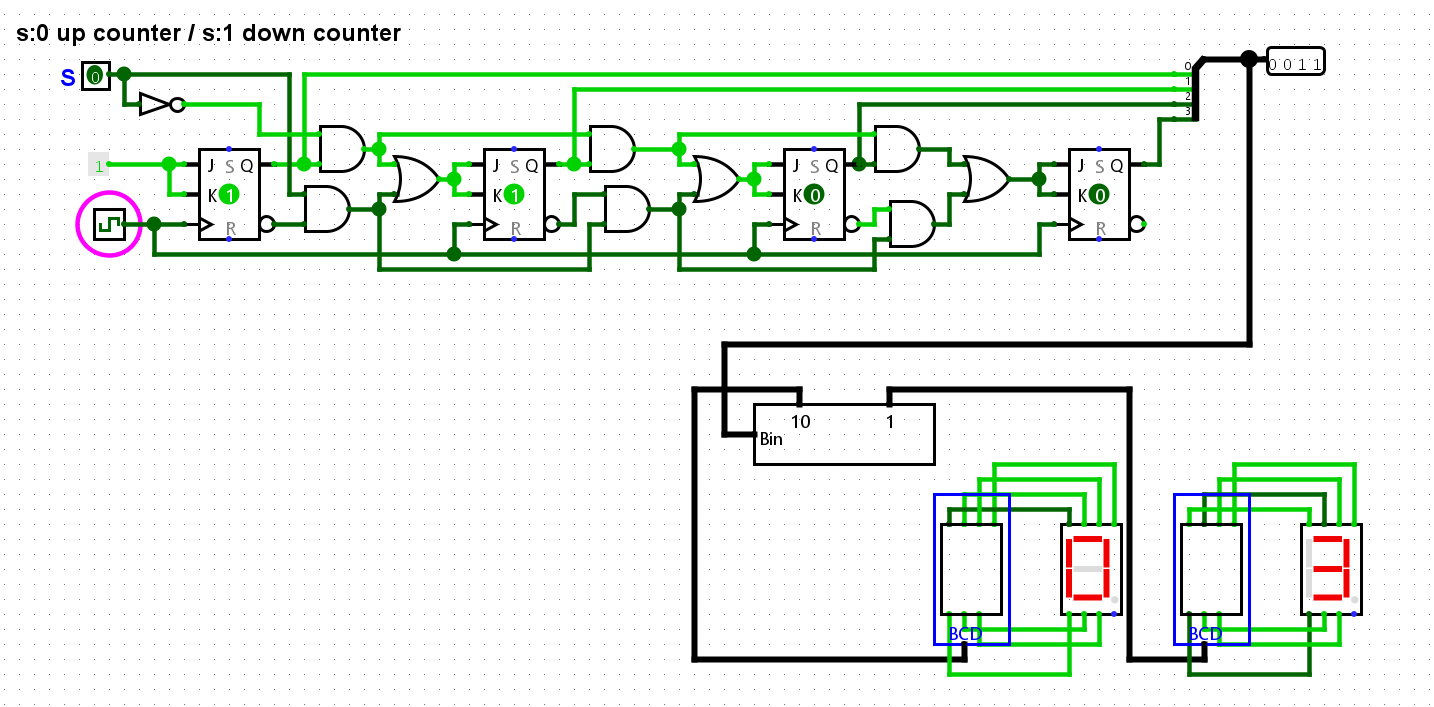
Jc=Kc=S’QbQa+SQb’Qa’

Jd=Kd= S’QcQbQa+SQc’Qb’Qa’

Connection of following conditions to respective J and K input of flipflops we finally get the respective output bits at Q of each flip flop.

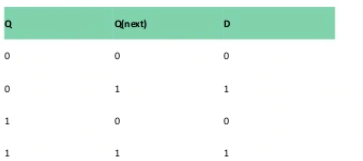


Connecting a binary to BCD decoder and BCD to seven segment decoder and seven segment display, we can see the following outputs.



3] Design a circuit that generates the sequence 0-8-12-6-13-11-7-3-1-0 using DFF.

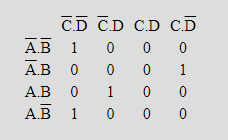
-> to design a circuit to generate that sequence, we will need 4 D flip flops. State table for D flip flop:



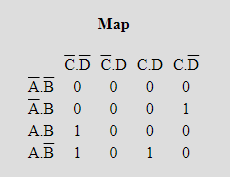
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Qan | Qbn | Qcn | Qdn | Qan+1 | Qbn+1 | Qcn+1 | Qdn+1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Using the above two tables for Qn and D.

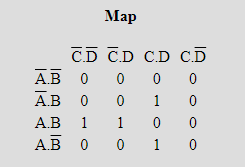
We get the following equations,



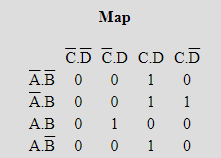
Da= Qb'Qc'Qd' + Qa'QbQcQd' + QaQbQc'Qd



Db= QaQc'Qd' + Qa'QbQcQd' + QaQb'QcQd

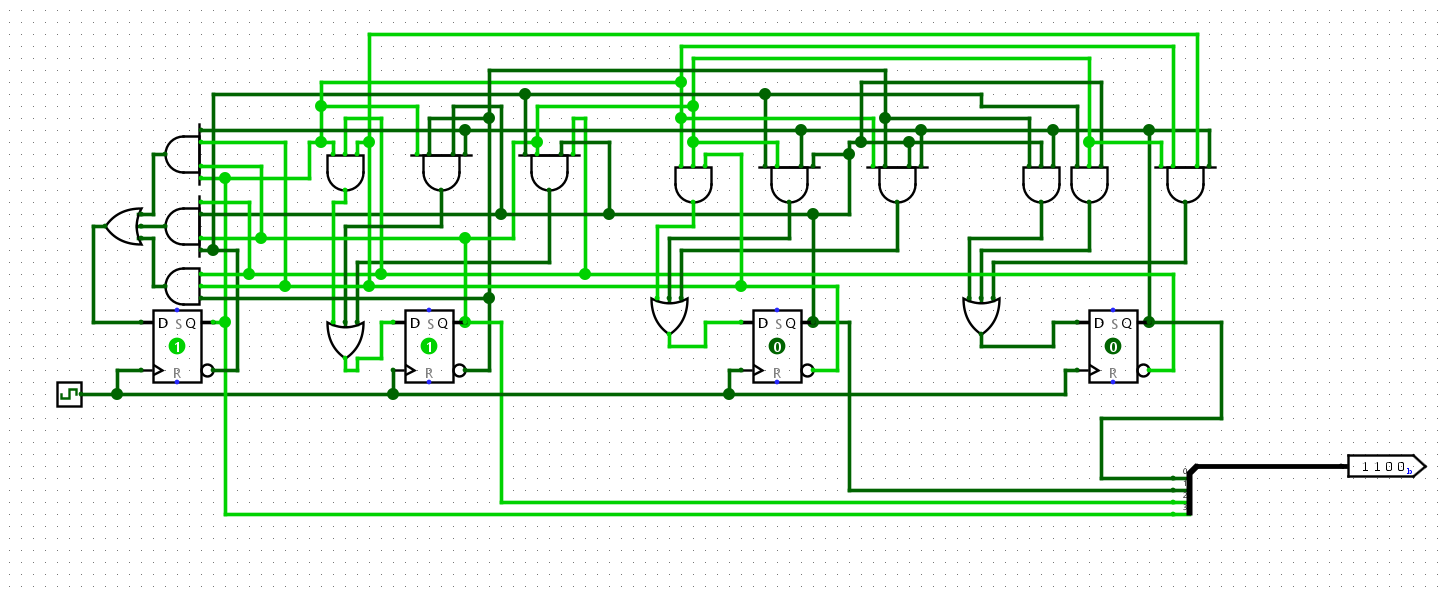


Dc= QaQbQc' + Qa'QbQcQd + QaQb'QcQd

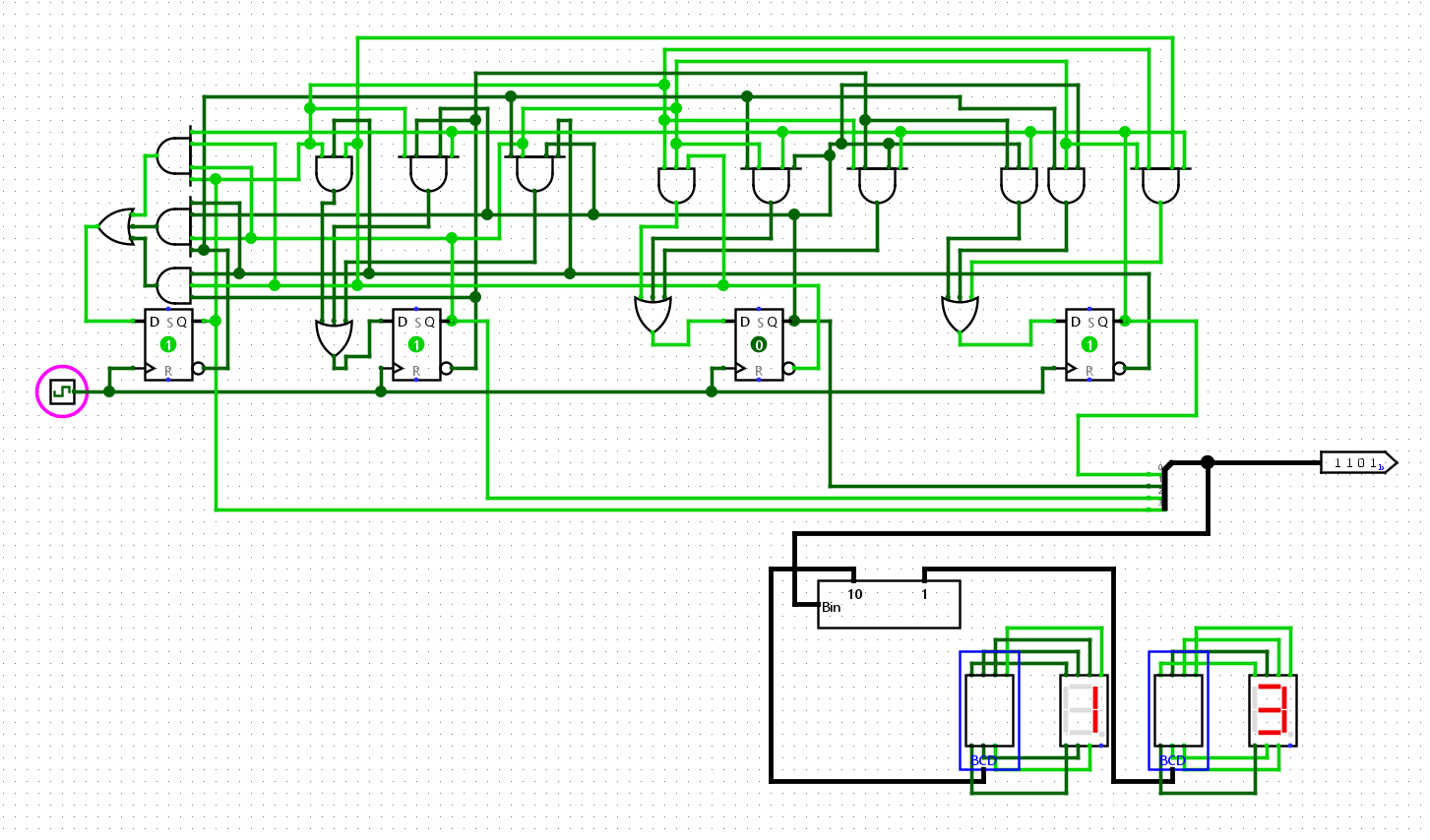
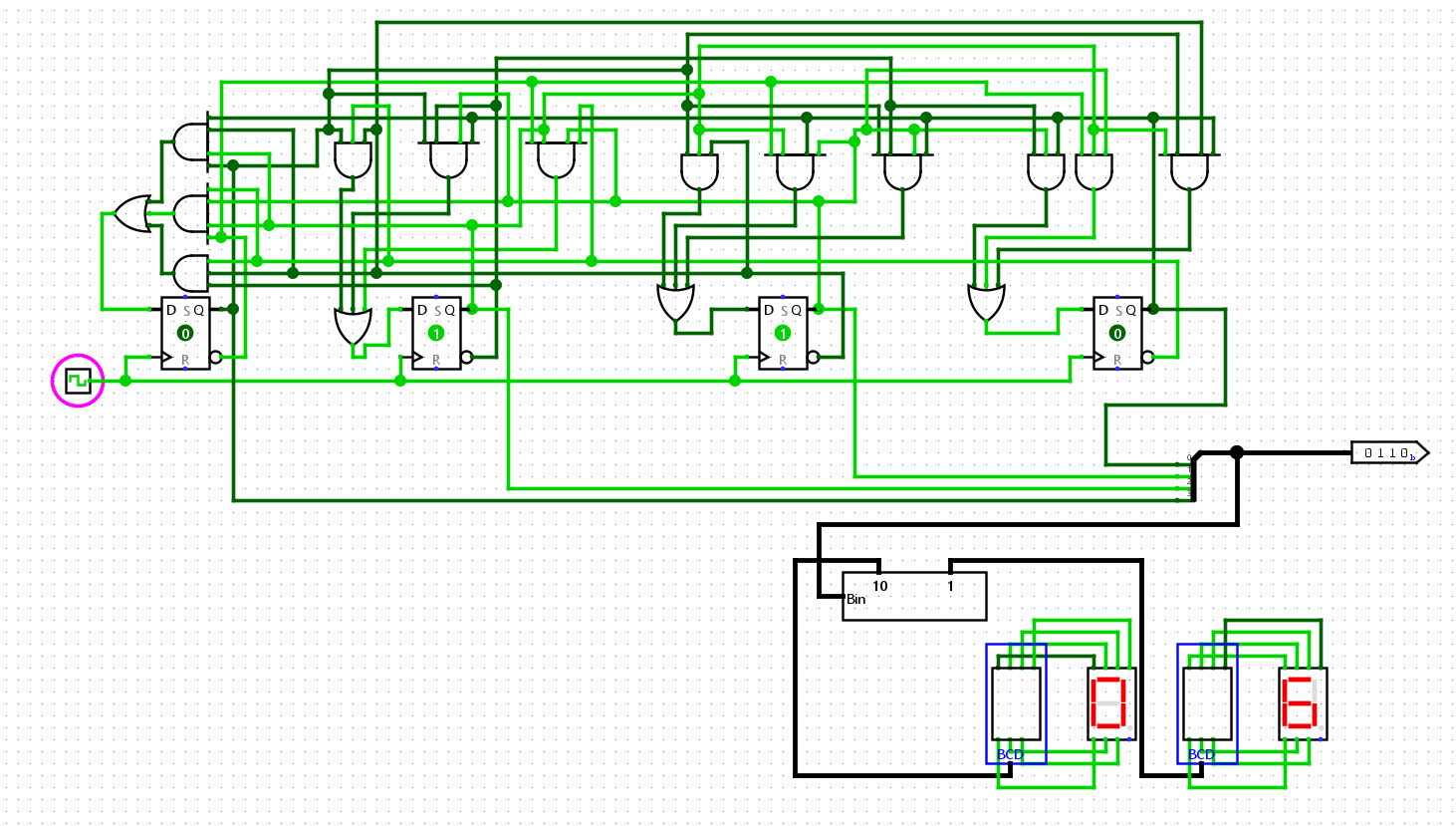
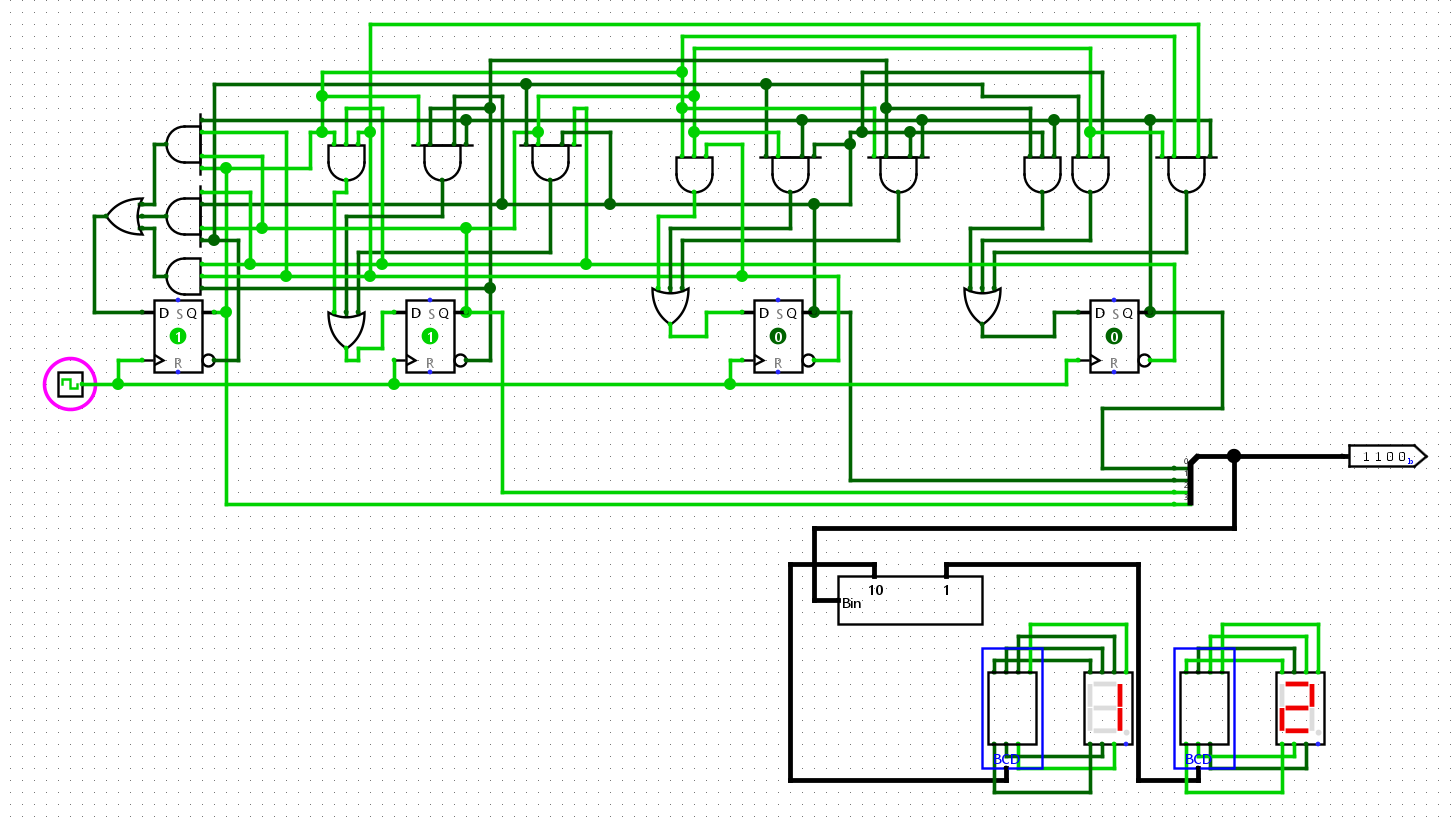
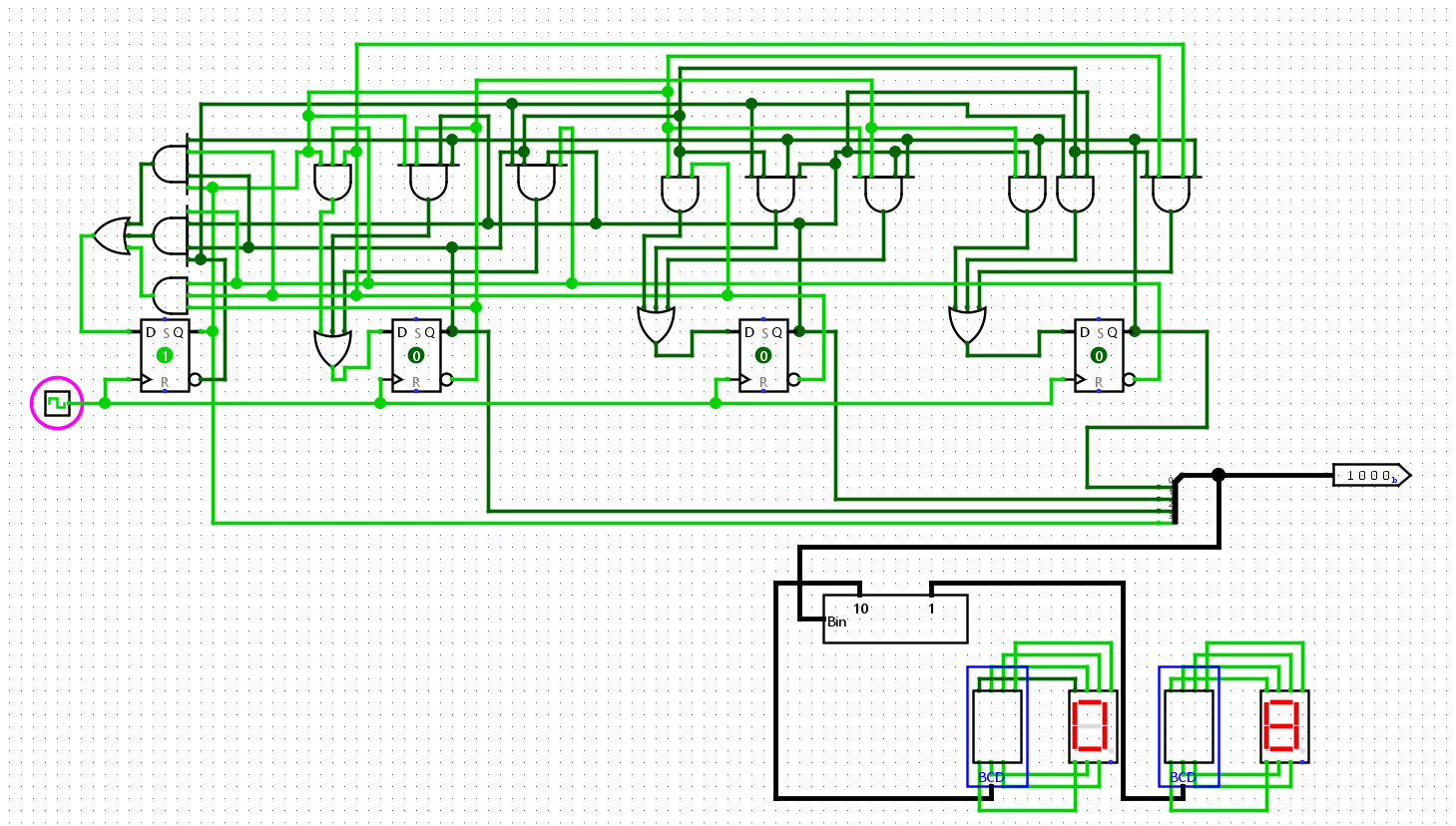
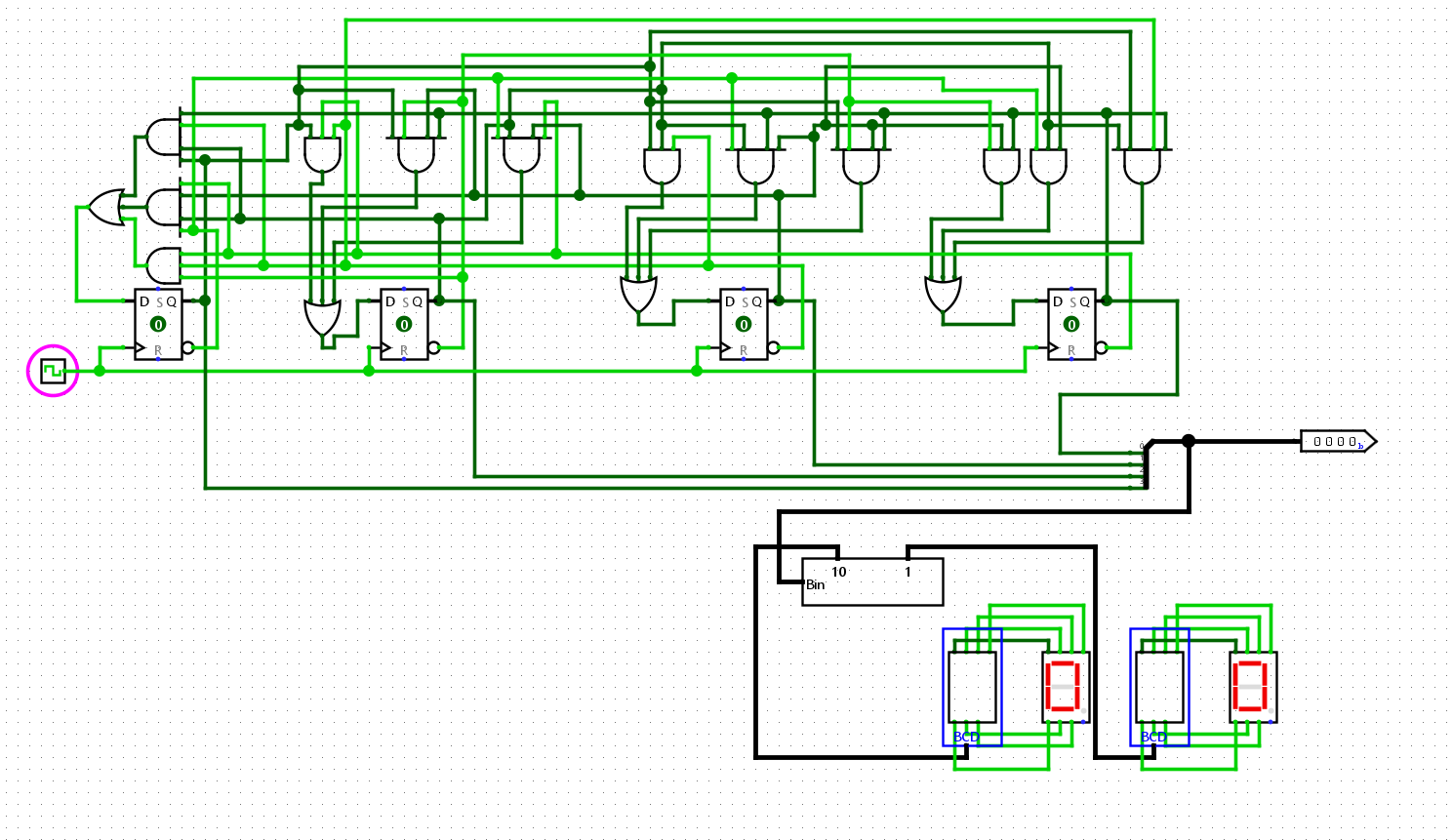


Dd= Qb'QcQd + Qa'QbQc + QaQbQc'Qd

Connection following conditions to respective D input of flipflops we finally get the respective output bits at Q of each flip flop.

****

Connecting a binary to BCD decoder and BCD to seven segment decoder and seven segment display, we can see the following outputs.



1] Design a Real time clock HRS:MIN:SEC using the counter available in the logisim library. Display the results using seven segment display. There should be provision to set the time (using load). Optional – Add provision for setting an alarm and displaying it

- > We will need 3 counters for the actual time counter, we will set all the counters to falling edge activation, the first hours counter to maximum of output 24 in binary while setting 60 in binary as maximum for seconds and minutes counter. Then connect the carry(1 when maximum) to clear of their respective counters so that it goes to 0 instead of the maximum number and also finishes proper time.

We will connect the clock to second’s counter and carry output of seconds counter to minute’s clock and minute’s carry output to hour counters clock input. The output will give us time in binary.

Now for setting up Load/Set Time counter. We will build a similar circuit with difference that the clock input of each counters will be button. And the outputs also connected to the respective Data loads of each actual time counters. A load button will be used to activate all the Load input and turn off count input while activating clock(so that values change).

This will let us change the time according to our needs using buttons.

Now for Alarm, we will use the Load/set time counters for setting an alarm timer. A toggle for alarm. If the time in time counters matches the one in the load/set time counters then they will start the alarm if the alarm toggle is 1. So we will XNOR respective bits of both actual time counters and load/set time counters and use And gate to And those all XNOR outputs so if all are equal they will result in 1 which further used with AND gate for the switch we get an Alarm, the alarm is shown using LED.

Circuit:

