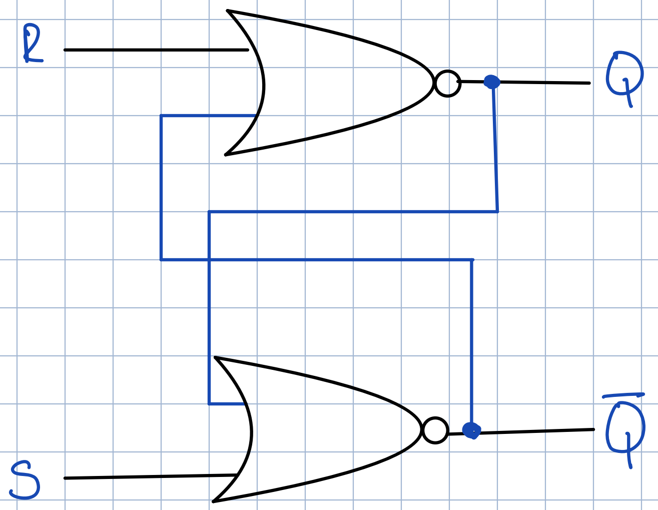
Lab-07:

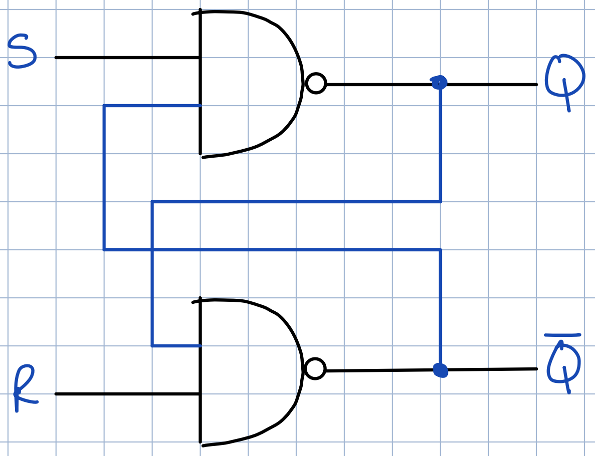
Latches:

S	R	
0	0	Invalid
0	1	SET
1	0	RESET
1	1	Memory

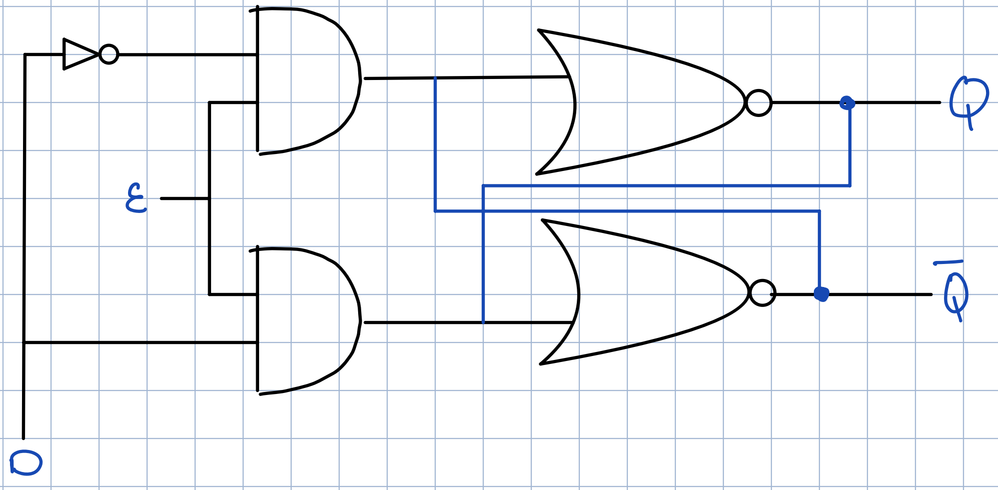
1. S-R Latch:



OR

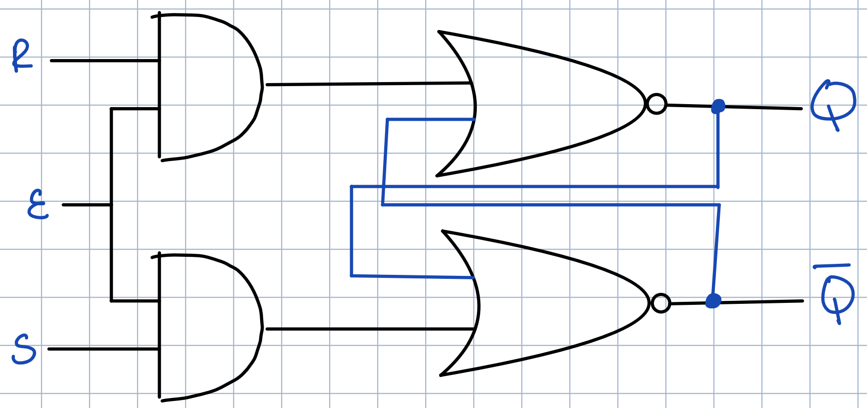


3. D-Latch



S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	Invalid mode	

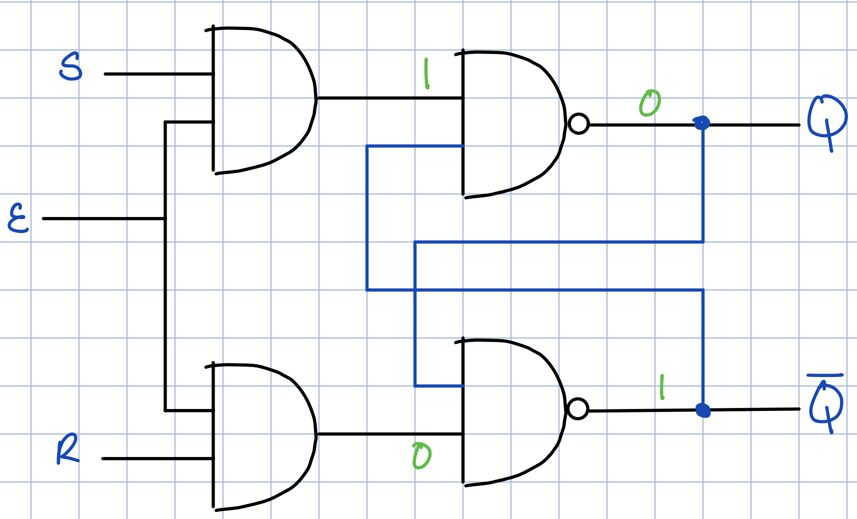
2. Gated-SR Latch:



D	Q	$\overline{Q}$
0	0	1
1	1	0

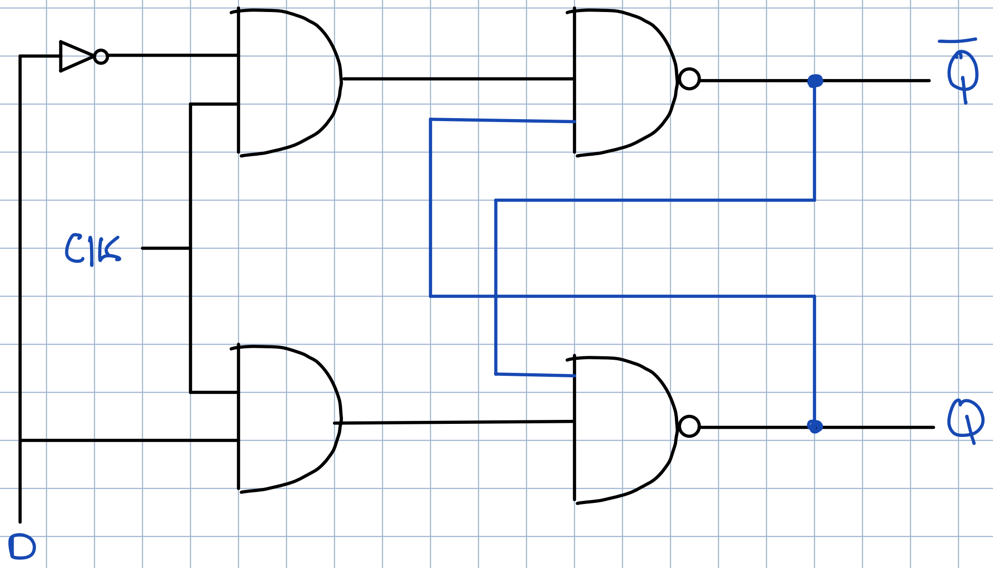
Flip flop:

1. S-R flip flop:



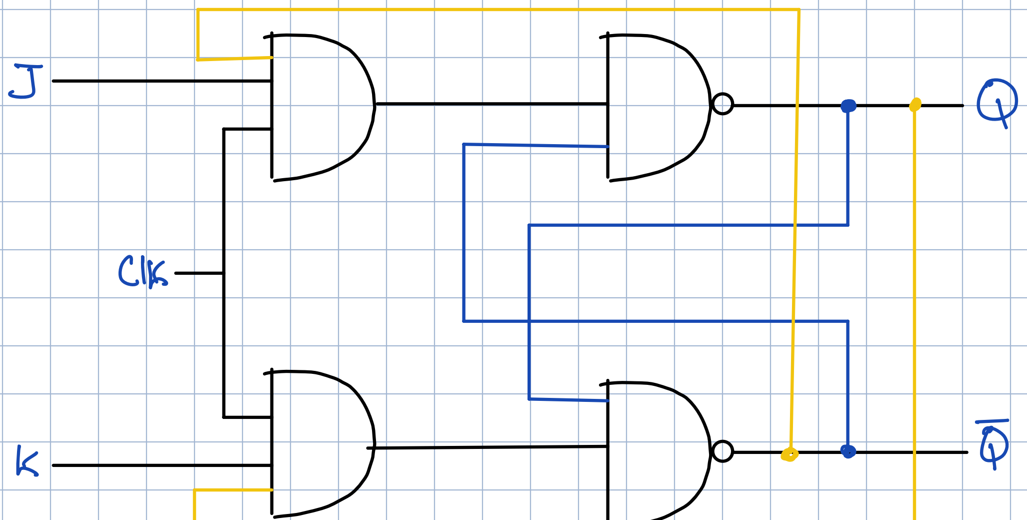
S	R	Q	$\overline{Q}$
0	0	Invalid	
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$

2. D-flip flop:



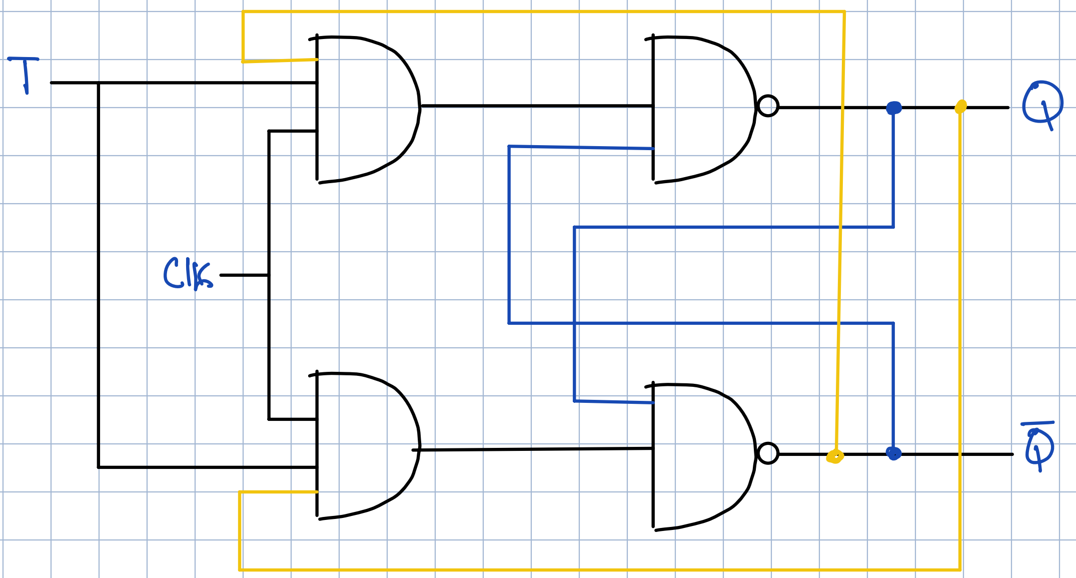
D	Q	$\overline{Q}$
0	0	1
1	1	0

3. J-K flip flop:



J	K	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	Toggle	

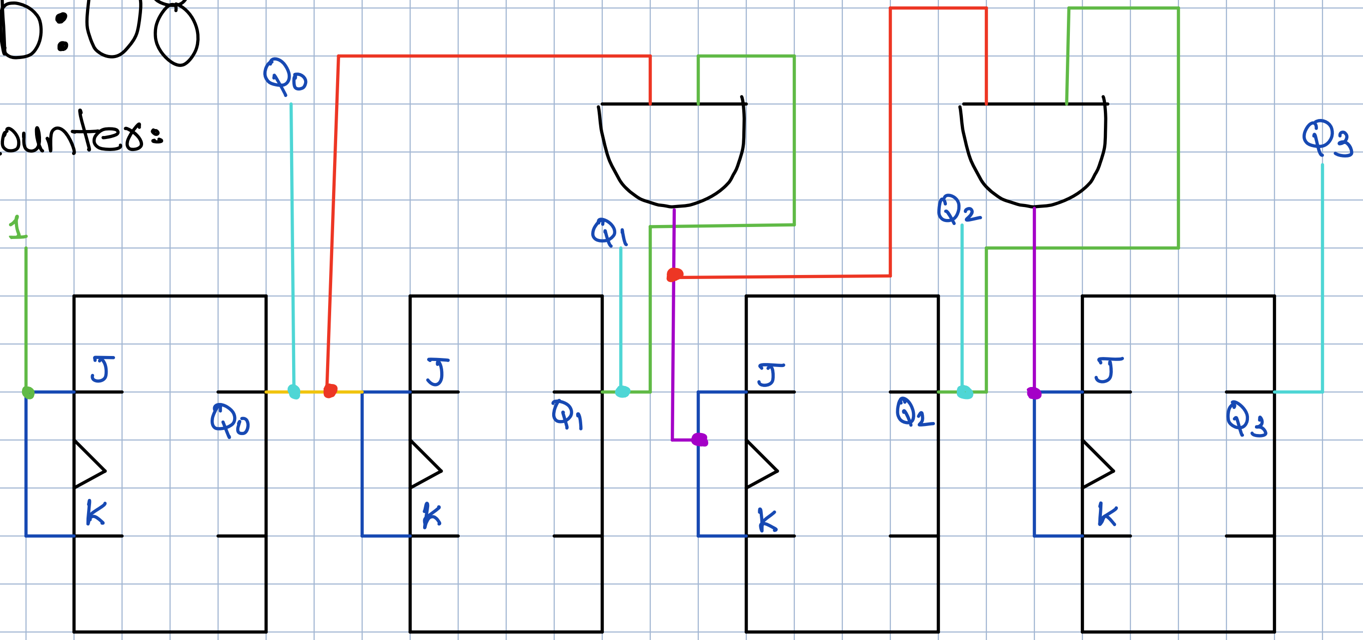
4. T-flip flop:



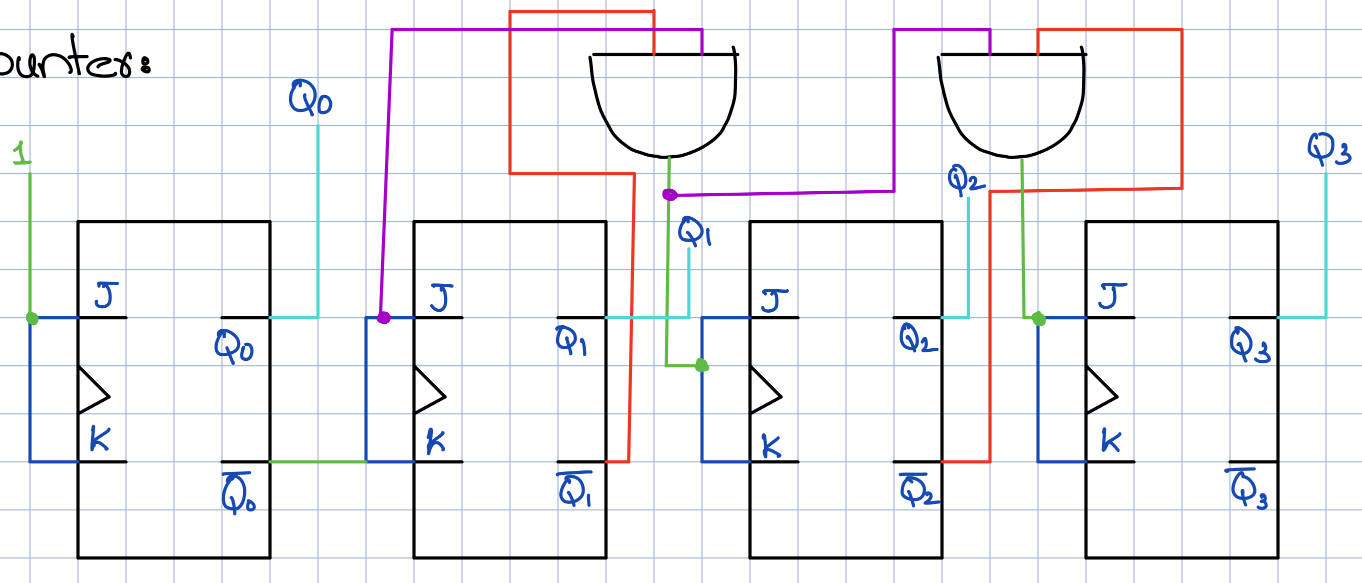
T	Q
0	Q
1	$\overline{Q}$

Lab:08

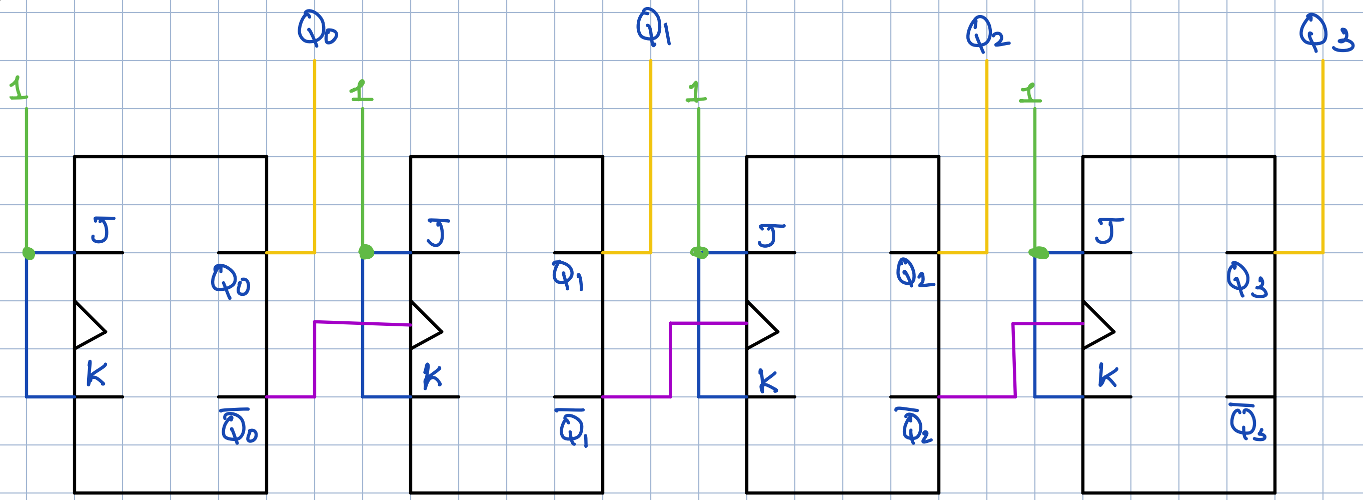
Up Counter:



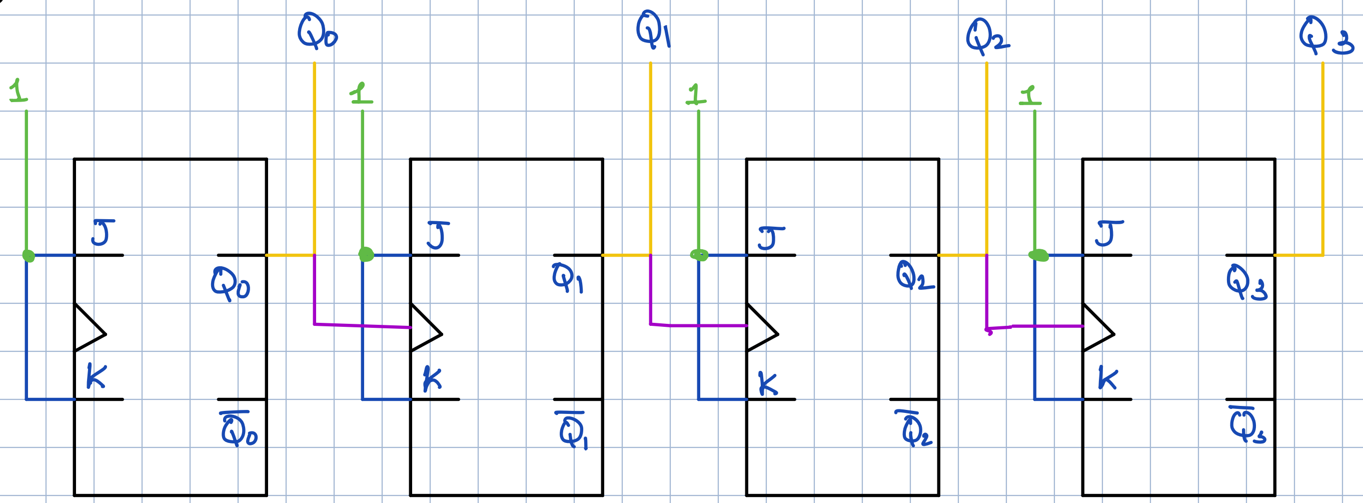
Down counter:



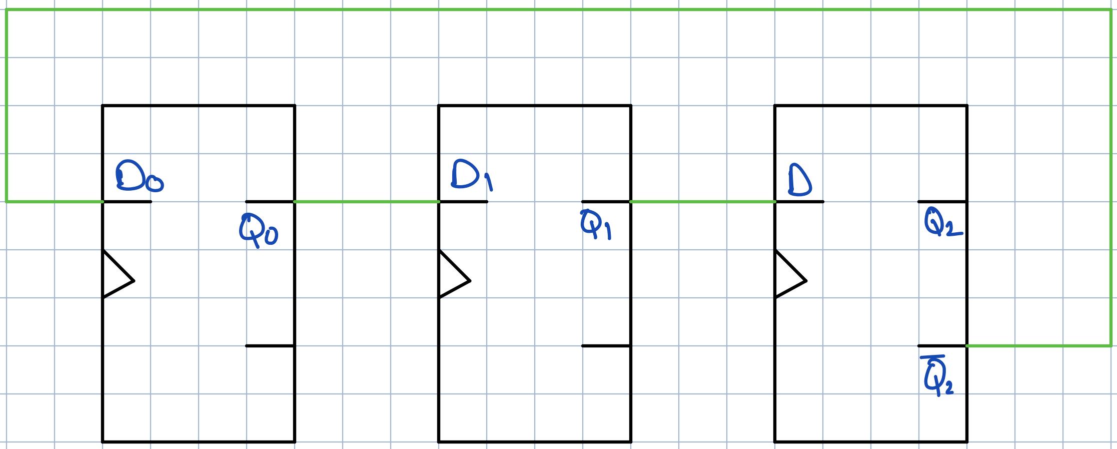
Asyn Up counter:



Asyn Up counter:

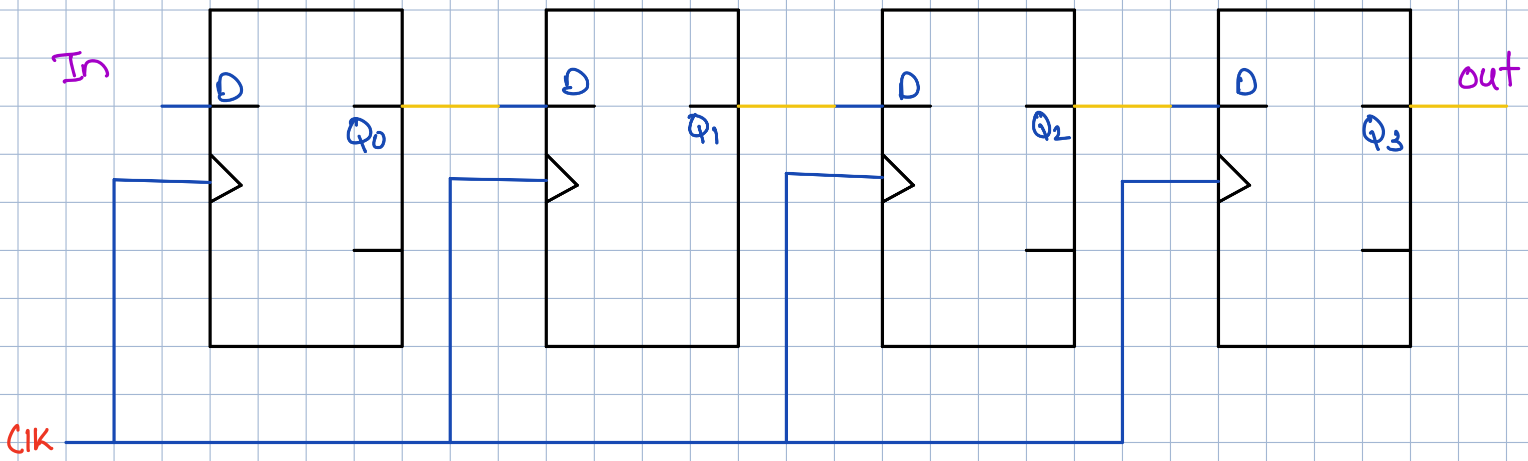


Johnson Counter:

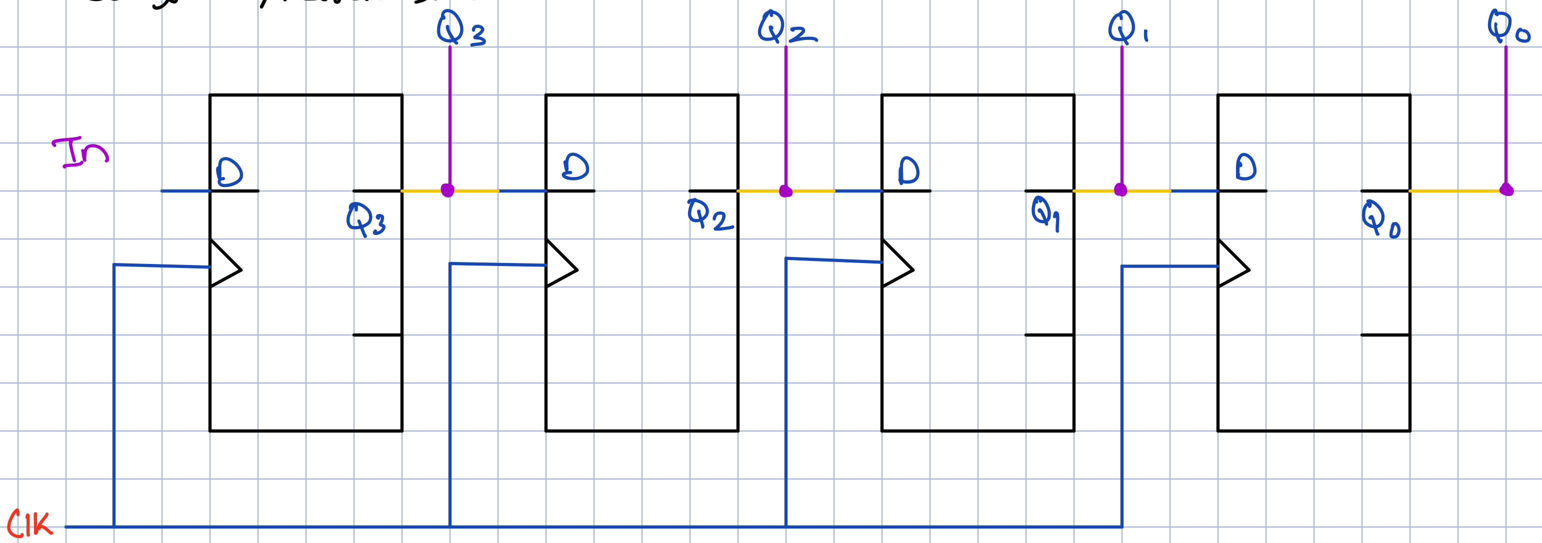


Lab: 09 Shift Register:

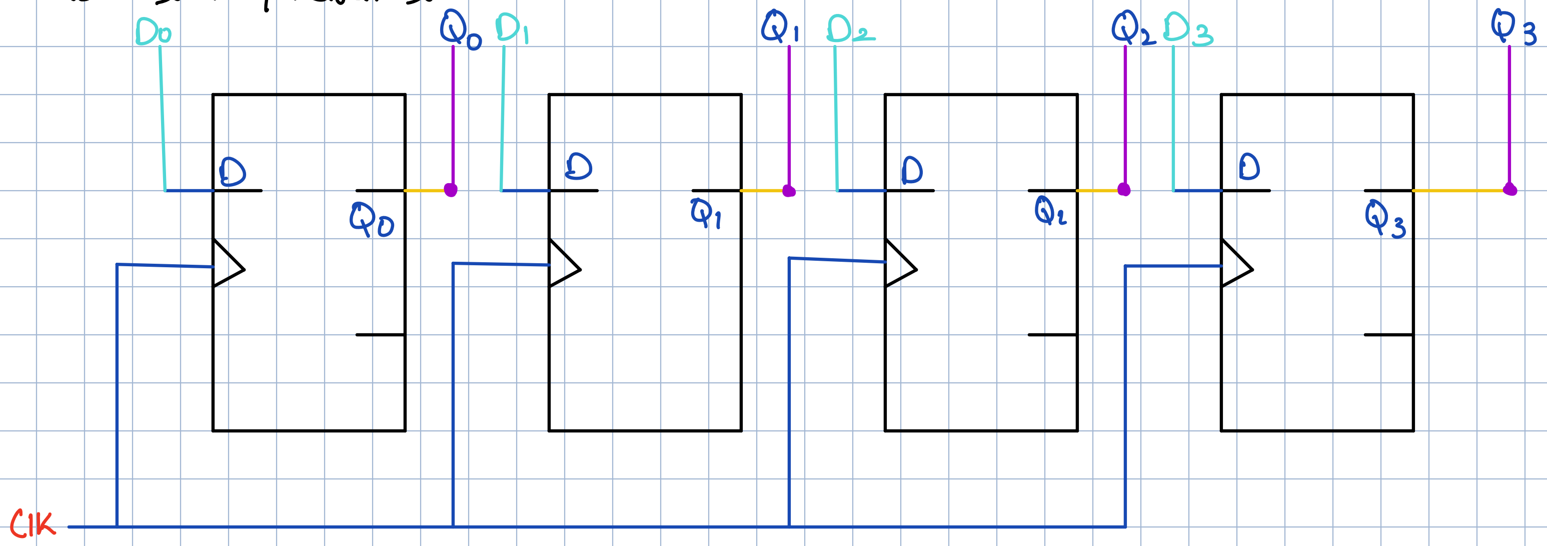
1. Serial In/ Serial out:



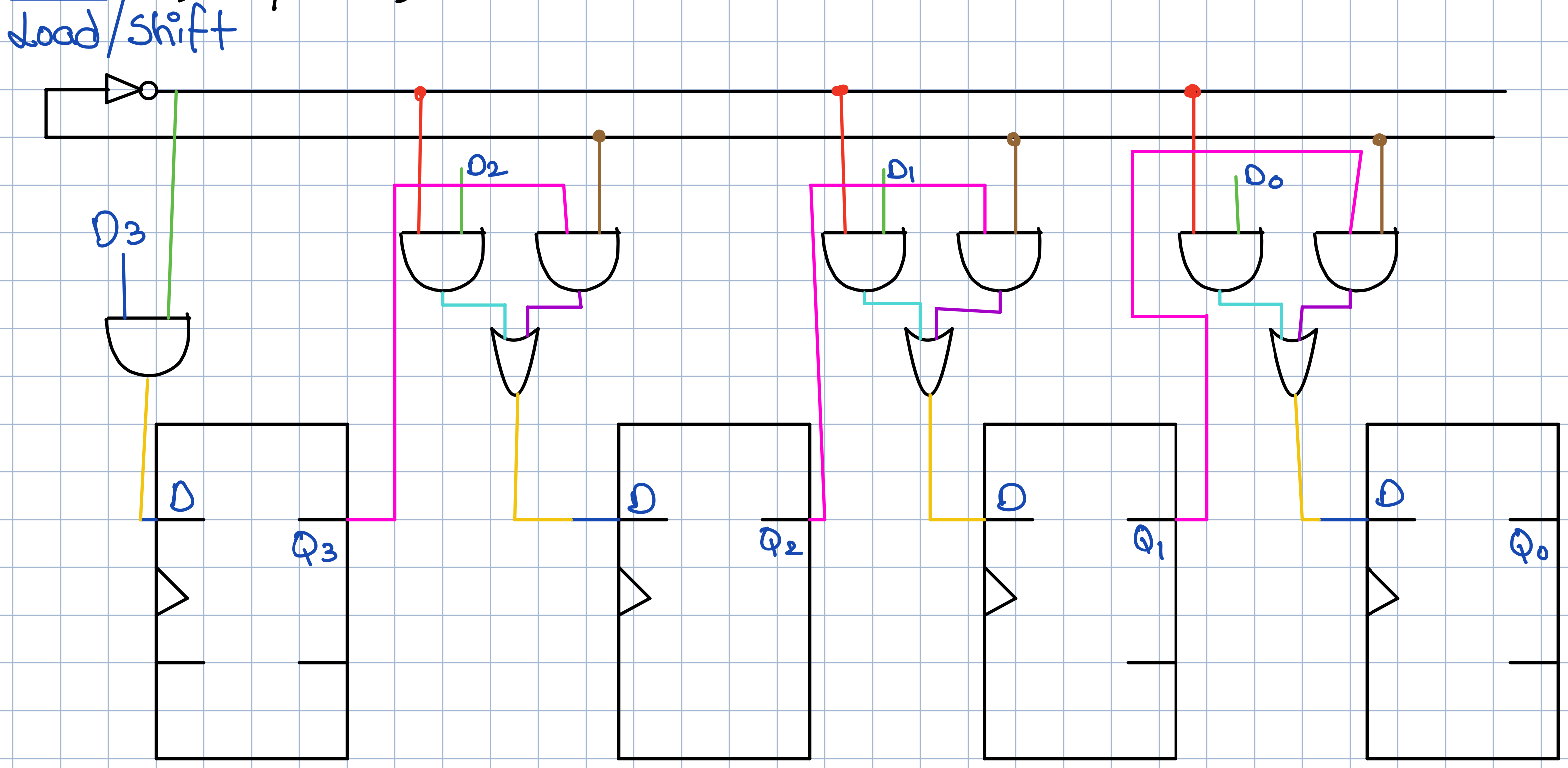
2. Serial In/Parallel out



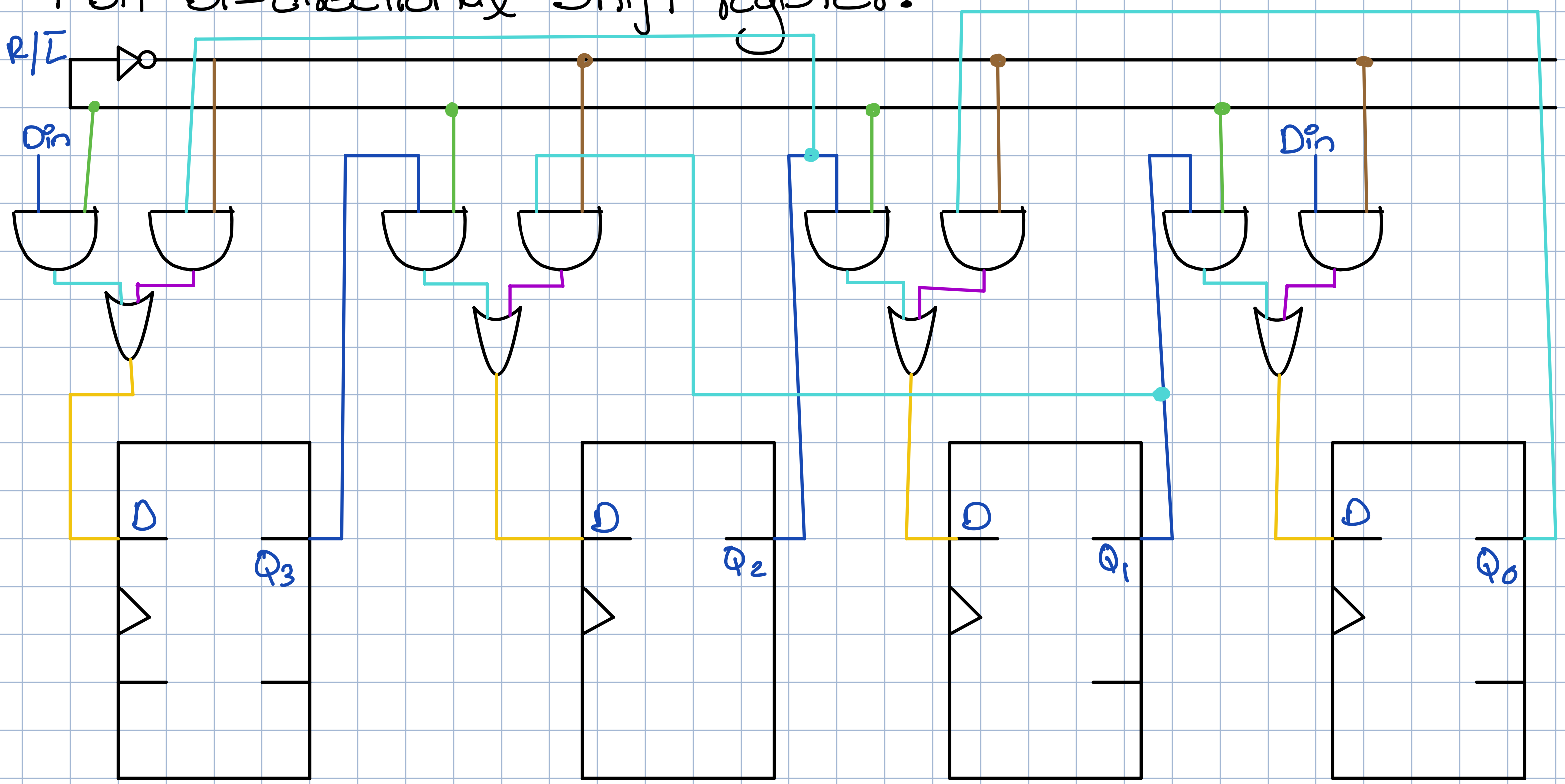
3. Parallel In/ Parallel out:



4. Parallel In / Serial out



4 bit bi-directional Shift register:



Lab:09 Vershilog

Module:

```
module   m_name (Input x, Output y);
endmodule
```

~ = NOT

| = OR

& = AND

^ = XOR

~& = NAND

assign y = ~x

Steps:

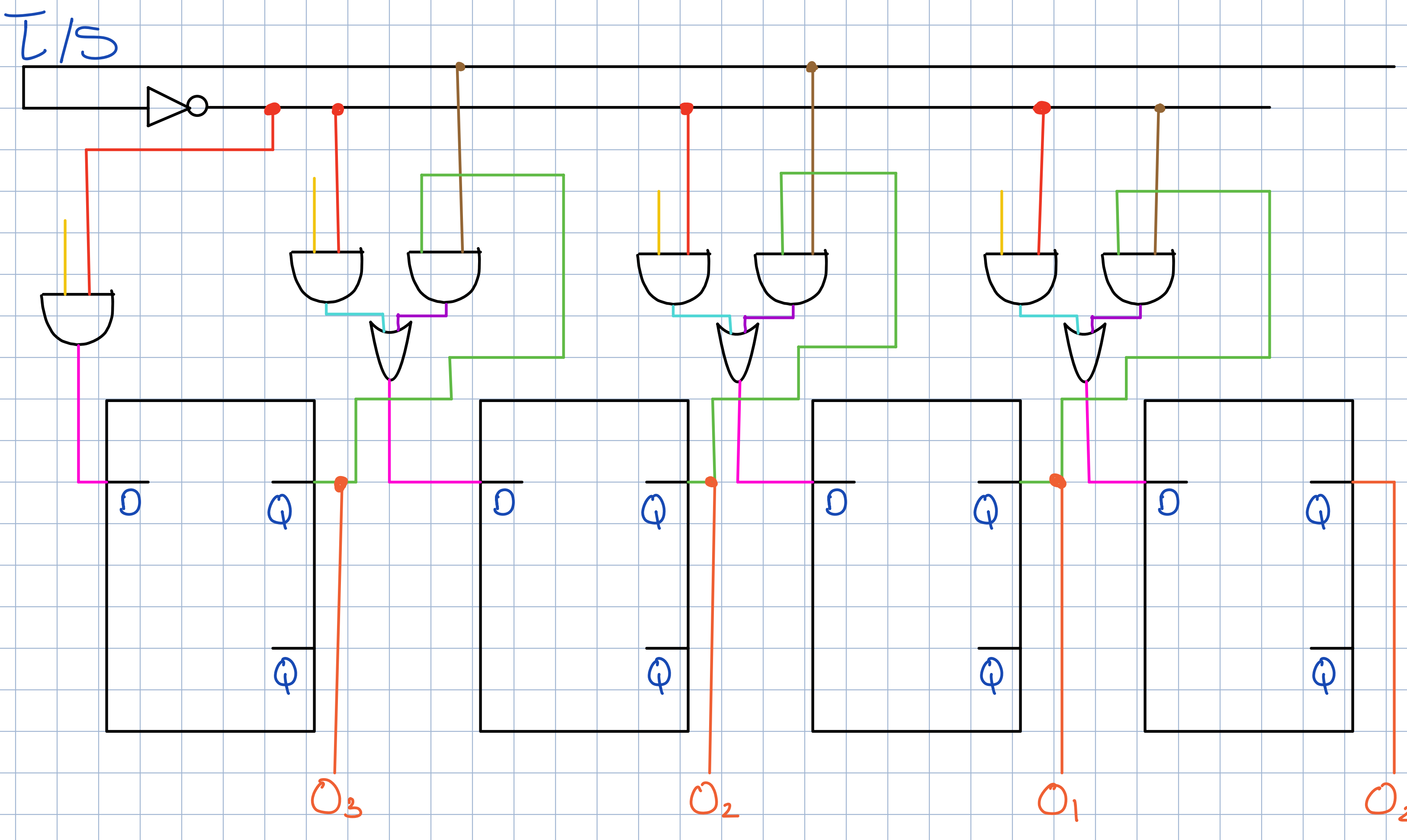
1. Open Vershilog
2. Create new project
3. Add new source → VHD module
4. Add new source
   ↳ VHD Test fixture

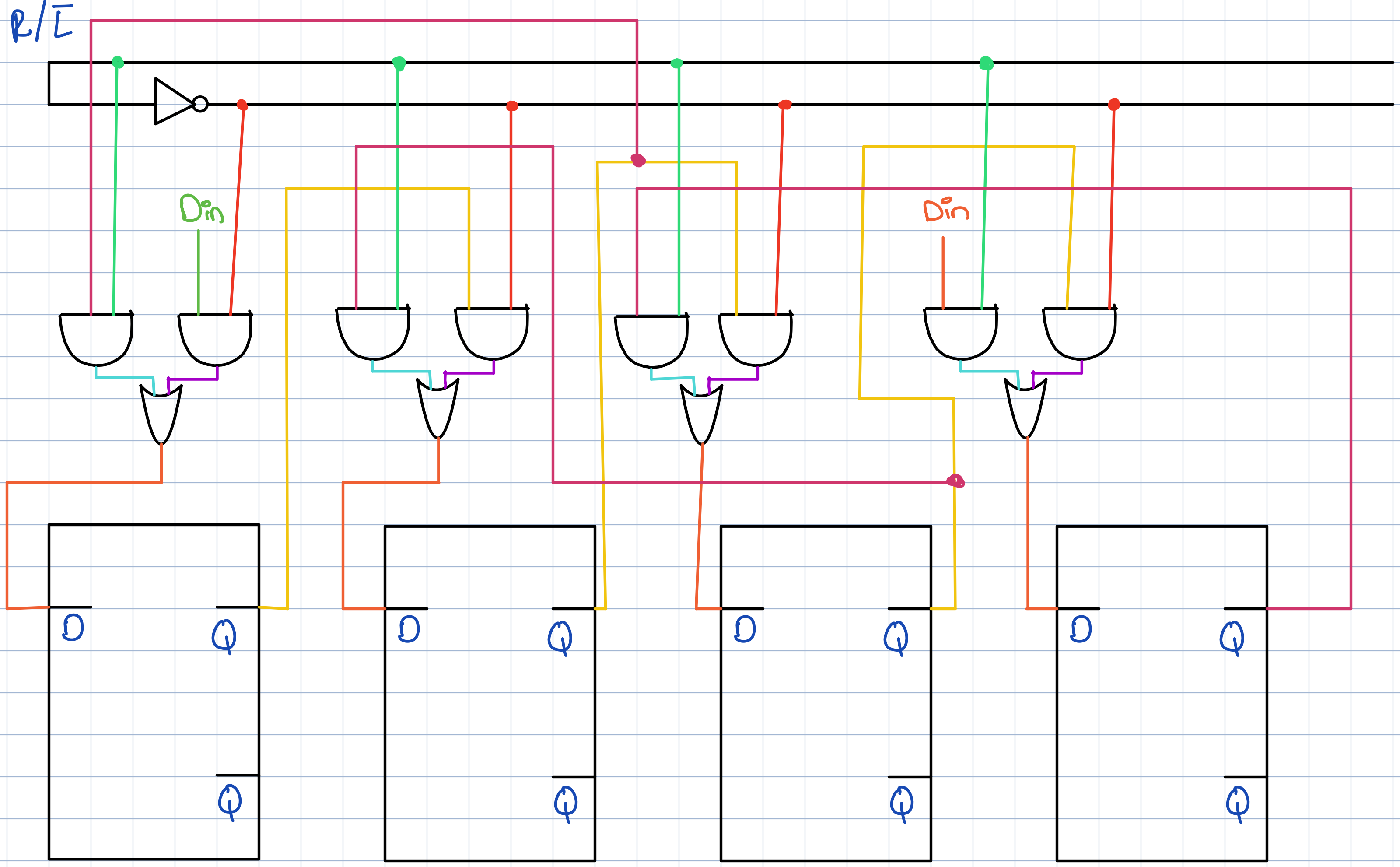
```
reg     x;
wire  y;
m_name uut (.x(x), .y(y));
initial begin
    Write truth table here.
    x=0;
    y=0; #10 → Delay
    $finish;
end
endmodule
```

Lab 10:

```
always @(posedge Clk)
  clock = ~clock;
$finish;
```

Practice #:





Verilog

↳ Create new proj

↳ New source
↳ VHD module

```
module AND(input X, input Y, output O);
    assign O = X & Y;

endmodule
```

↳ New Source
↳ VHD test fixture.

```
module tb;
  reg X, Y;
  wire O;
  AND uut (.X(X), .Y(Y), .O(O));
  initial begin
      X=0; Y=0;
      X=0; Y=1; #10
```

↳ Run.

```
X=1; Y=0; #10
X=1; Y=1; #10

$finish;
end
endmodule
```

J-k flip flop:

•vhd file:

hold value & change over time.

```
module J_k (input J, K, Clk, reg output Q);
  always @(posedge clk) begin
    if (J=0 && k=0) begin
       Q <= Q;
    end

    if (J=0 && k=1) begin
       Q <= 0;
    end

    if (J=1 && k=0) begin
       Q <= 1;
    end

    if (J=1 && k=1) begin
       Q <= ~Q;
    end

  end
endmodule
```

•v file:

```
module tb;

reg  J, K, Clk;
wire Q;

J_k (.J(J), .k(k), .Clk(Clk), .Q(Q));
```

```
always begin
  Clk = ~Clk; #5
end
initial begin
J=0; k=0; Clk=0;
 J=0; k=0;
J=0; k=1; #10
J=1; k=0; #10
J=1; k=0; #10
end
endmodule
```

Practice

Combinational	Sequential
1. Output depends on present inputs only	1. Output depends on both present inputs & states.
2. Feedback loop is NOT present	2. Feedback loop
3. Memory elements NOT required	3. Memory elements required
4. Clk signal NOT required	4. Clk signal required
5. Easy to design	5. Difficult to design.

Characteristic Equation:

J k Q_n Q_{n+1}

D Q_n Q_{n+1}

T Q_n Q_{n+1}

S R T Q_n Q_{n+1}