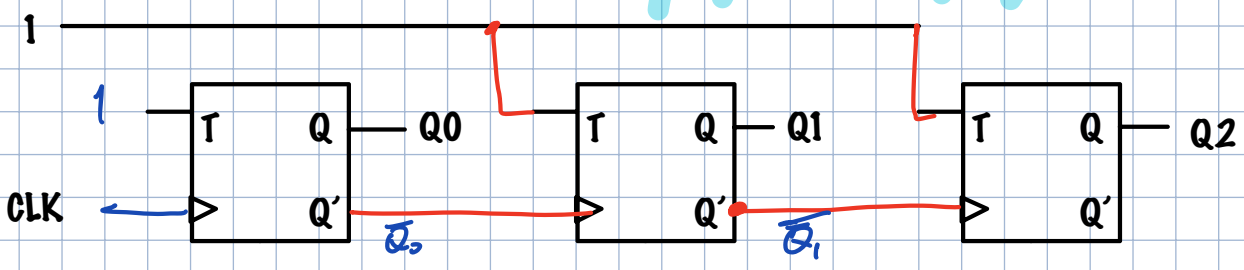
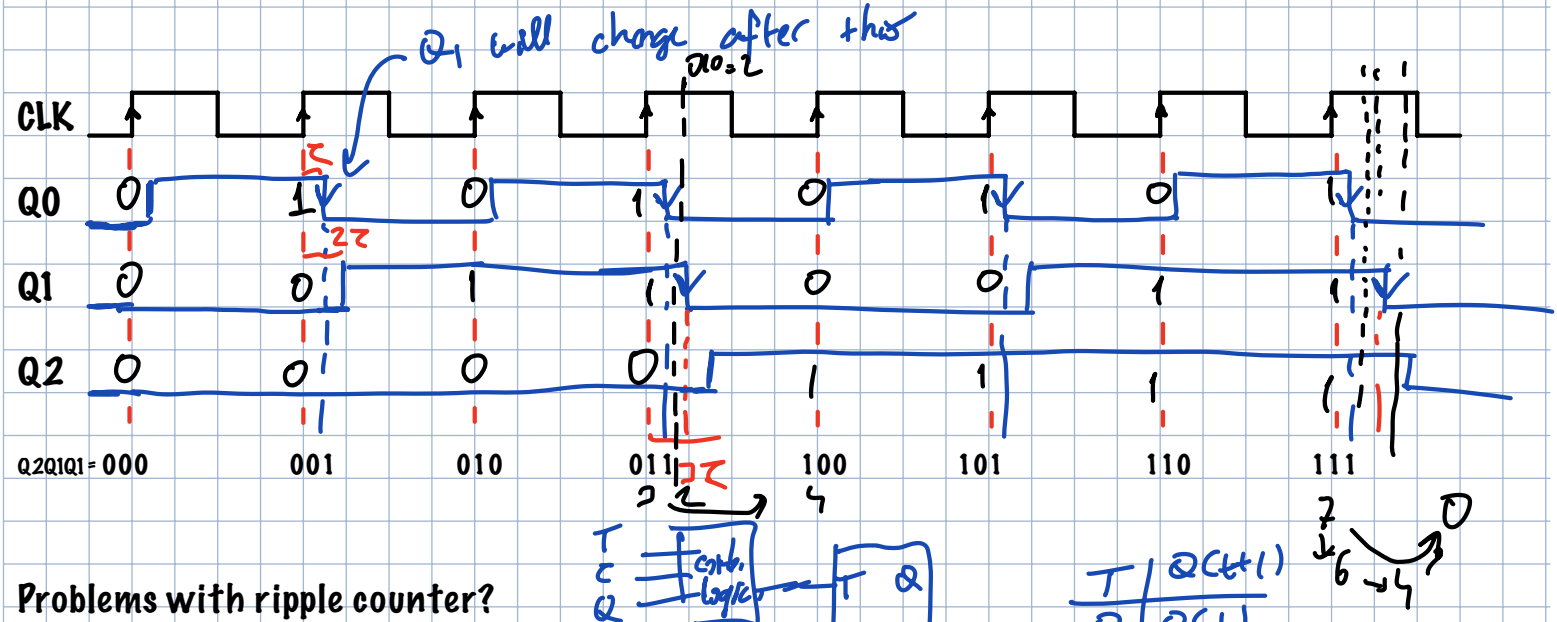
Chapter 7 - Sequential Circuits Part 3

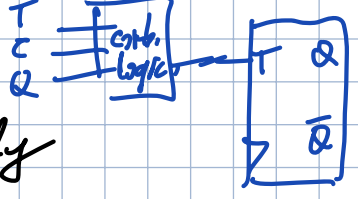
Ripple (up) counter

3-bit up counter

Want to count: 000-001-010-011-100-101-110-111-000-...







T	Q(t+1)
0	Q(t)
1	$\overline{Q}$(t)

Problems with ripple counter?

- delays on addition (especially problematic for counter with large # bits & high CLK frequency)
- asynchronous
- undesired transient values in count.

synchronous clear

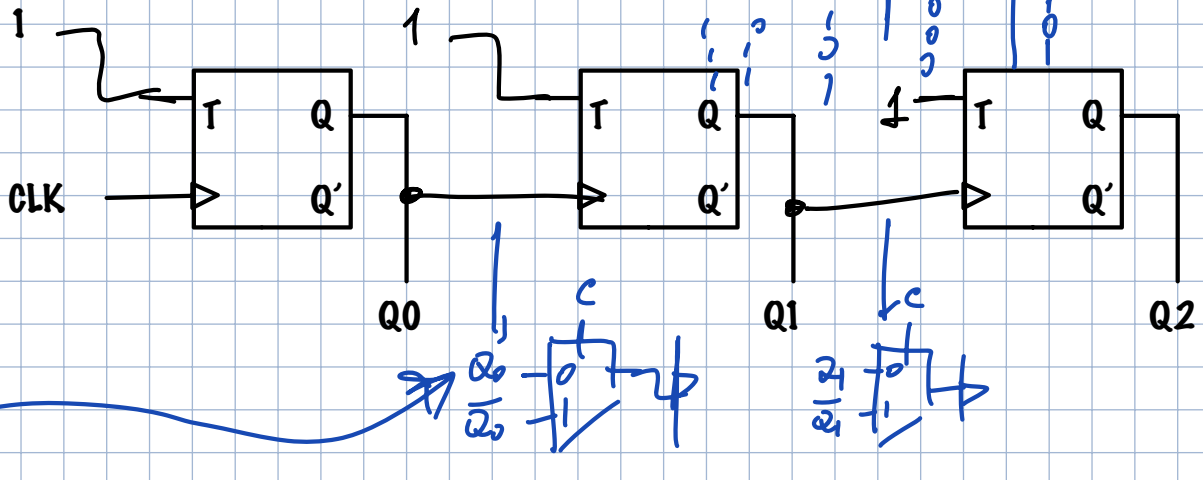
C = 1 → Q(t+1) = 0

C = 0 → Q(t+1) depends on T

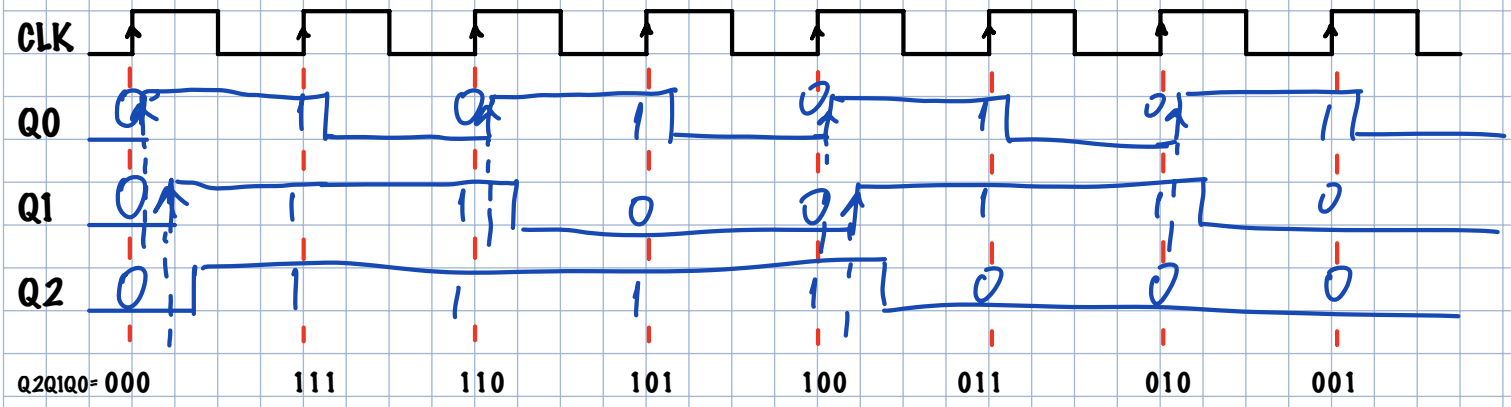
C	T	Q(t)	Q(t+1)	T'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

$T' = f(C, T, Q)$

Ripple (down) counter



111 - 110 - 101 - 100 - 011 - 010 - 001 - 000 - 111



ripple up-down counter (C=1 count up, C=0 count down)

Synchronous counters

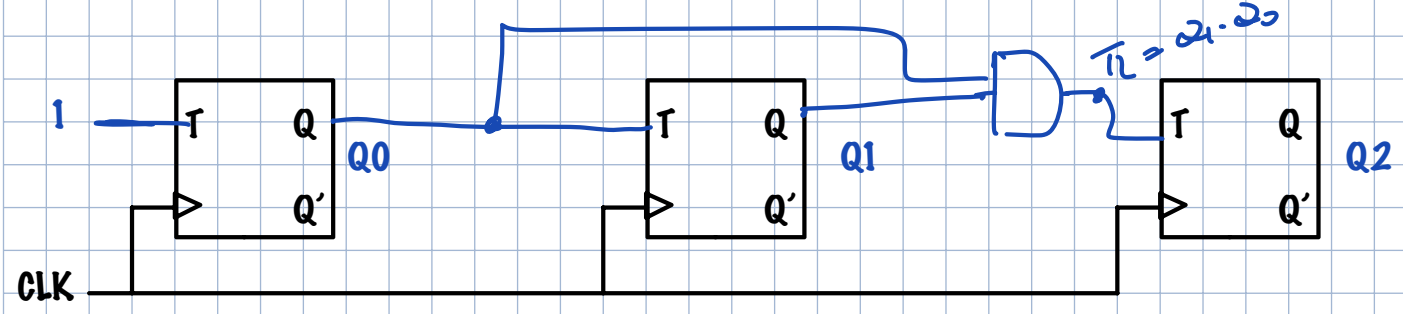
3-bit up counter

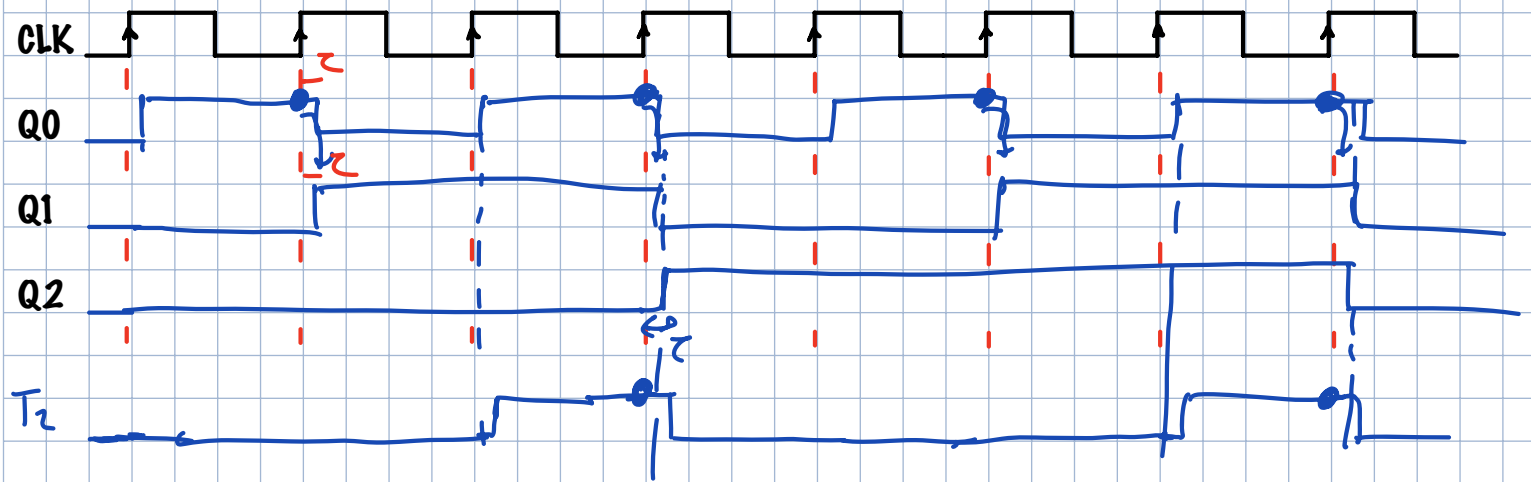
All flip flops will have the same CLK as input

clock cycle	Q2	Q1	Q0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Q1 changes

Q2 changes





VHDL code for synchronous up counter

```
architecture upcount_arch of upcount is
    signal Count: std_logic_vector(2 downto 0);
begin
    process(CLK) begin
        if rising_edge(CLK) then
            Count <= Count + 1; -> increment
        end if;
    end process;
    Qout <= Count;
end upcount_arch;
```

for arithmetic operations use: std_logic_unsigned package

How to design an n-bit up counter?

00 -- 00
00 -- 01
00 -- 10
:

00011111 → 00100000

output : $Q_{n-1}\ Q_{n-2}$ -- $Q_1\ Q_0$

input : $T_{n-1}\ T_{n-2}$ -- $T_1\ T_0$

$T_0 = 1$

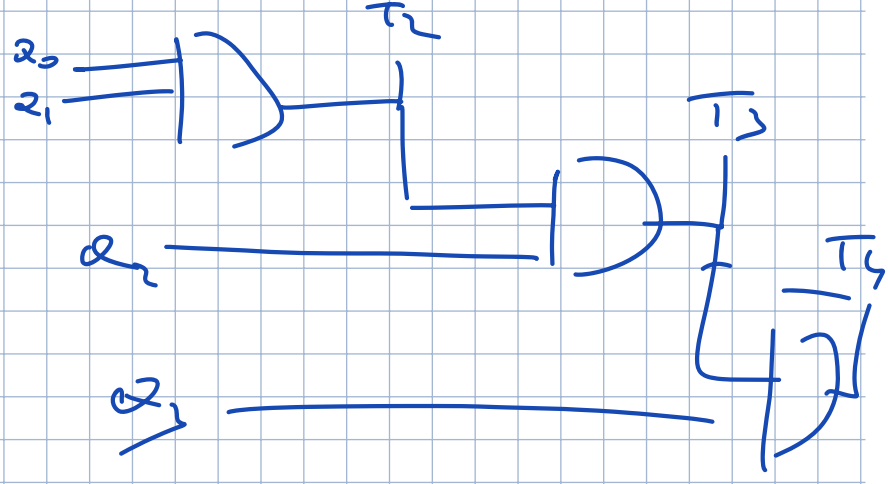
$T_1 = Q_0$

$T_2 = Q_0 \cdot Q_1$

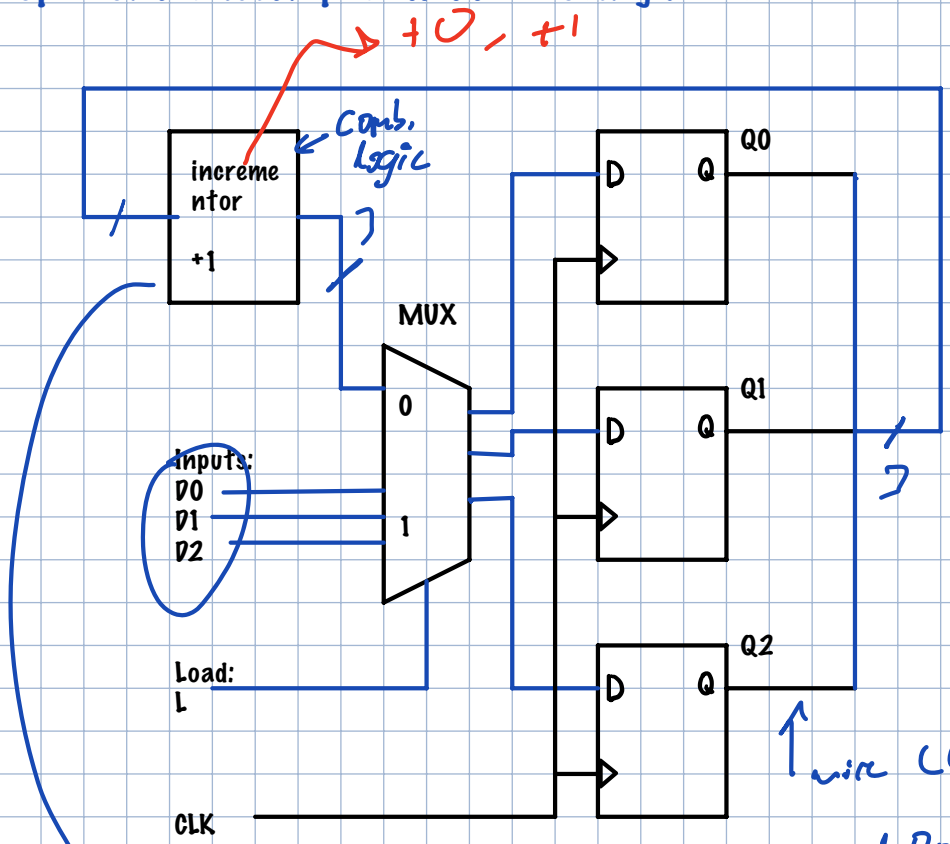
:

$T_{n-1} = (Q_0 \cdot Q_1) \cdots Q_{n-2}$

$T_3 = Q_0 \cdot Q_1 \cdot Q_2$

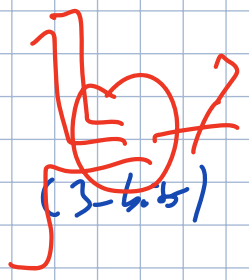


3-bit up counter with parallel load using D-FFs

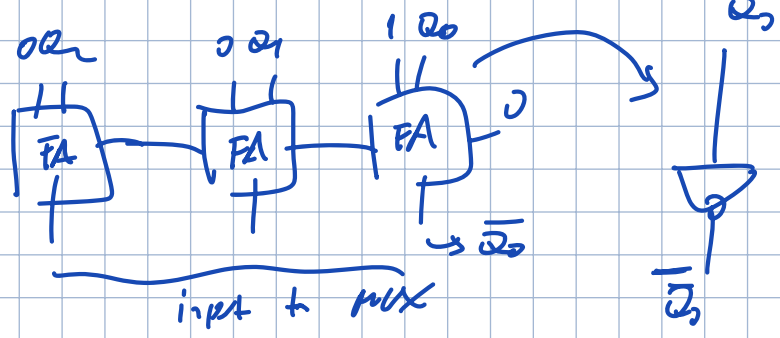


L=1 ⇒ Q=D

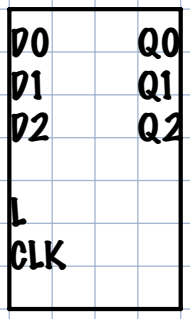
L=0 ⇒ Q ← Q+1



min (1-bit)



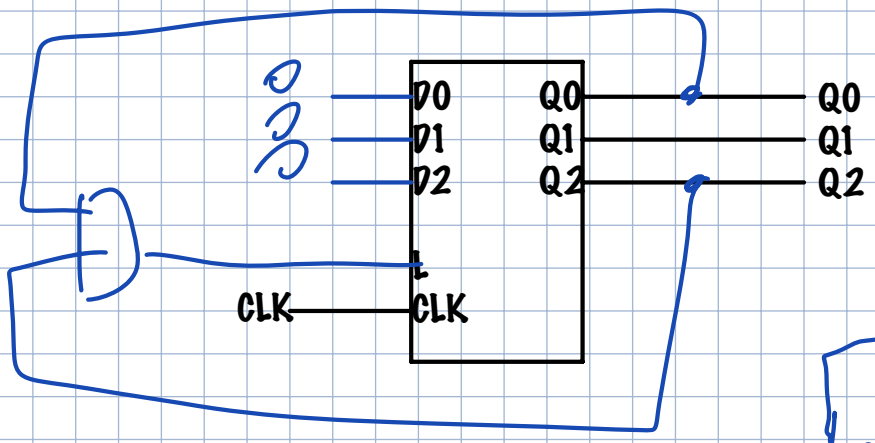
symbol



How can we make a mod 6 counter using the counter above?

Mod 6 counter: 000-001-010-011-100-101-000-001-...

0 1 2 3 4 5 0



$L = Q_2\overline{Q_1}Q_0$

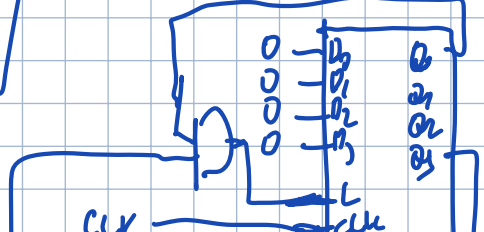
simplified to

$L = Q_2Q_0$

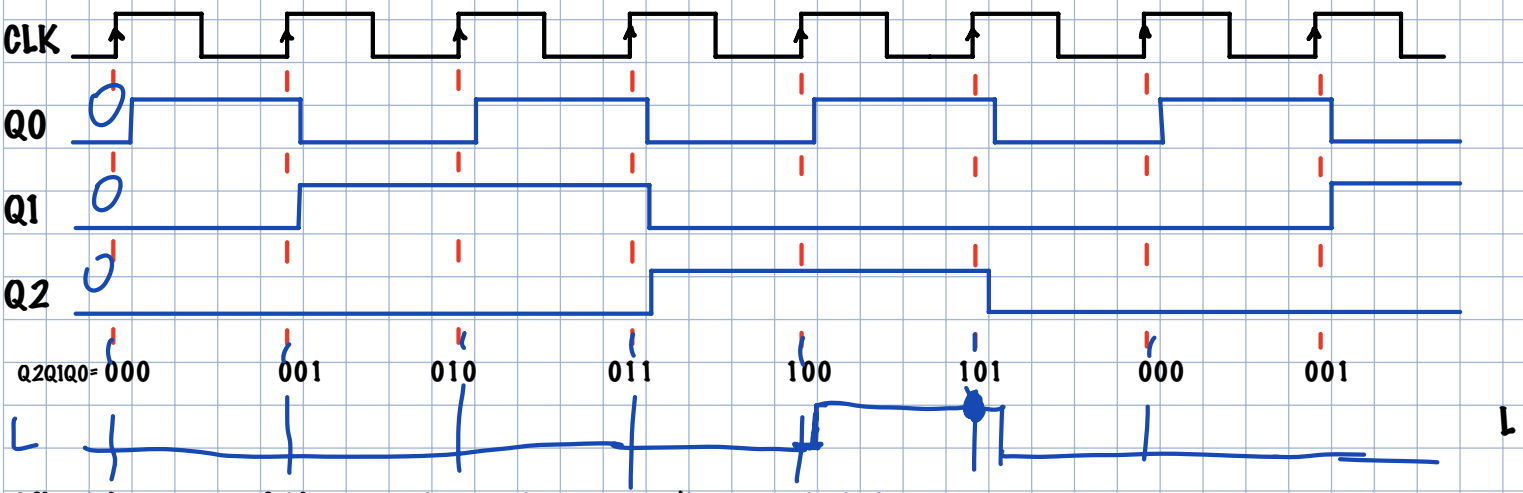
1001

BCD counter

0-1-2-3-4-5-6-7-8-9-0



timing diagram

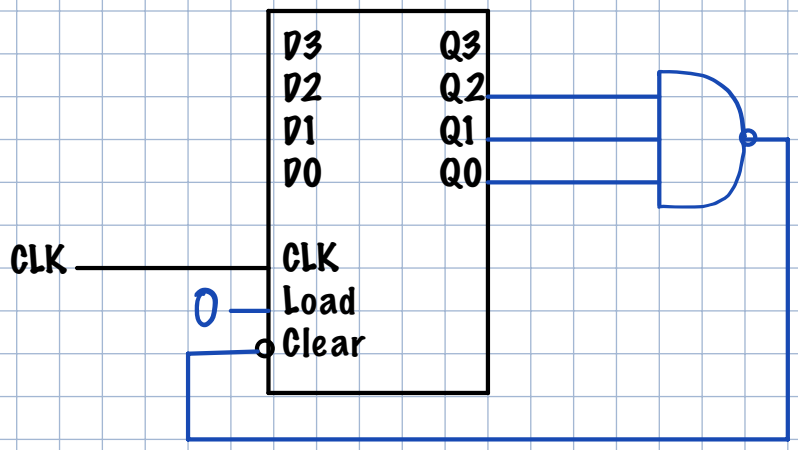


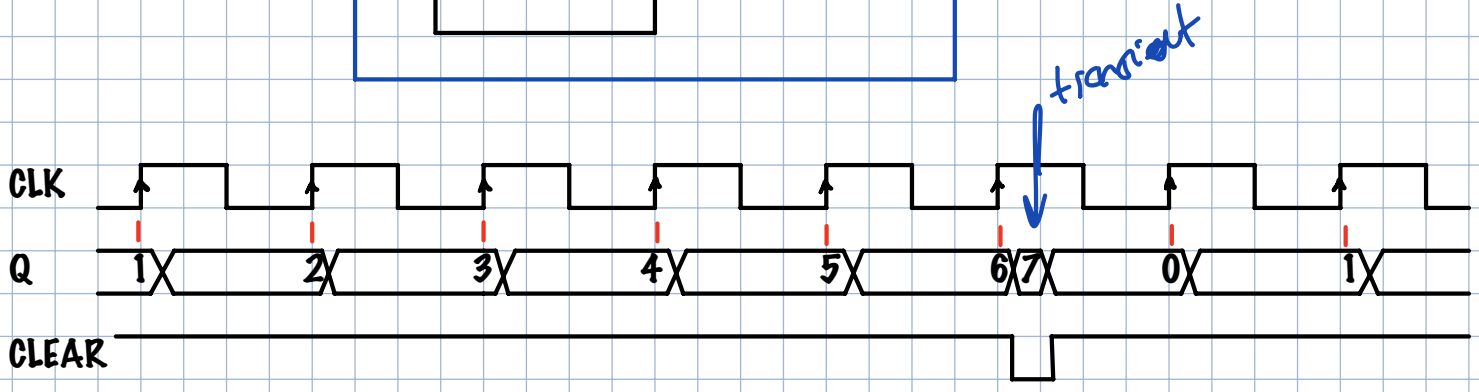
What happens if the counter enters a don't care state?

6 110?
7 111?

110 (L=0) → 111 (L=1) → 000 → 001 → 010 →
↳ corrected itself

Making a mod 7 counter from a 4-bit counter with synchronous load and asynchronous clear.
Want: 0->1->2->3->4->5->6->0->1->...





Not permitted during usual operation

asynchronous operation
AVOID this! "suicide counter"

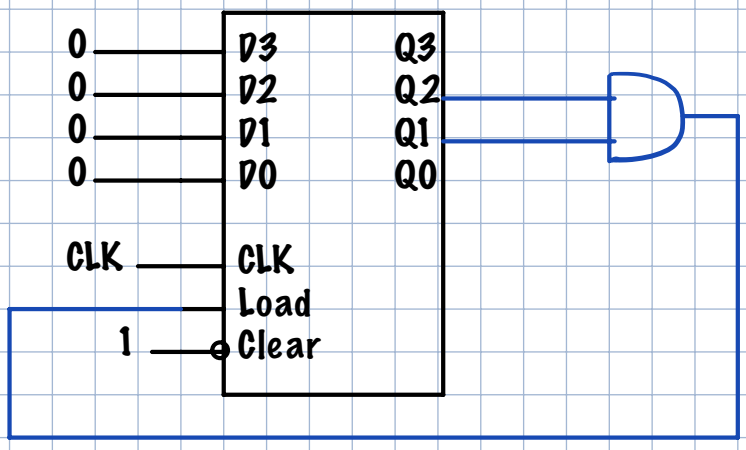
such things when you power up

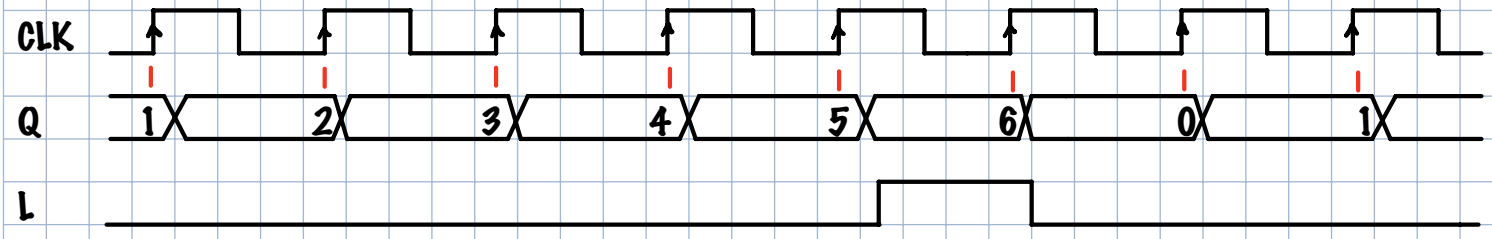
Better Alternative:

for the first time.

6
11
0
1
1
0

Synchronous load on terminal count of 6



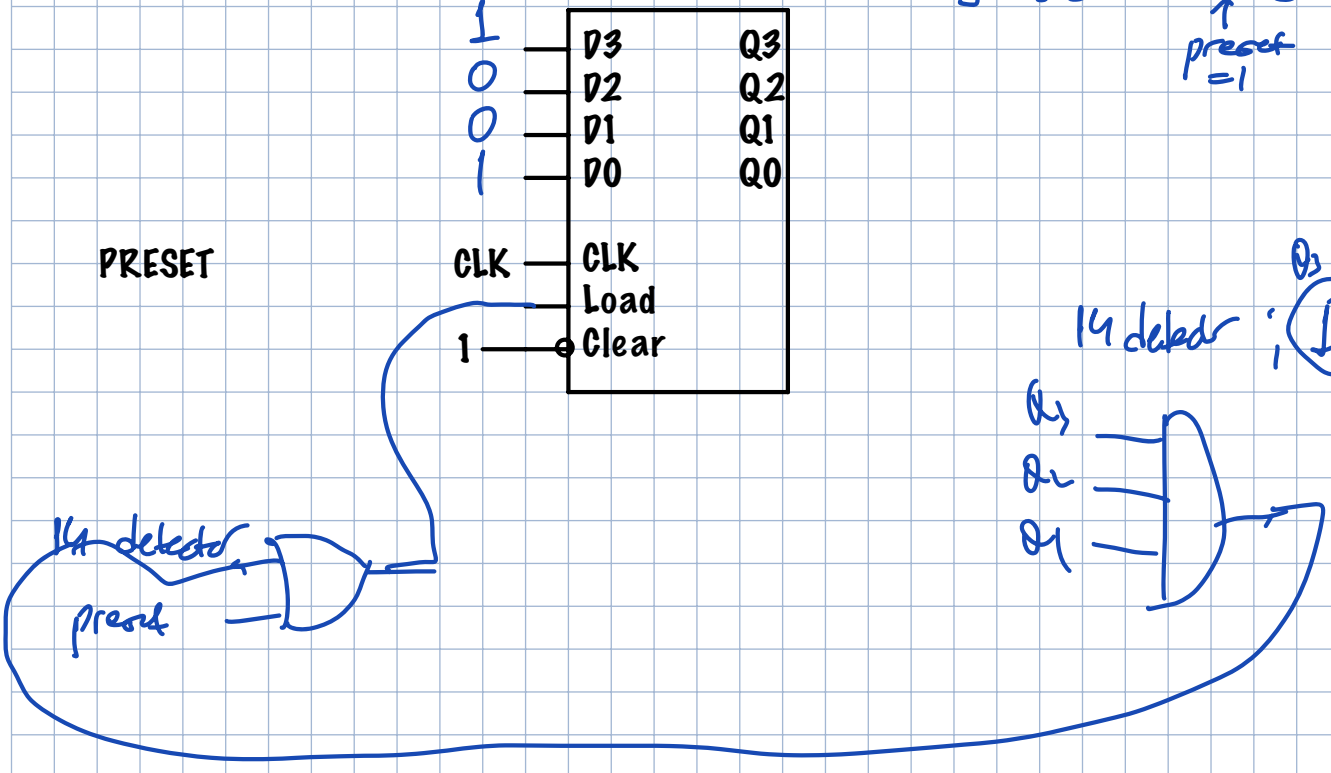


Modulo 6 counter with synchronous preset to 9 and synchronous load to 9 on terminal count 14:

9 - 10 - 11 - 12 - 13 - 14 - 9 - -

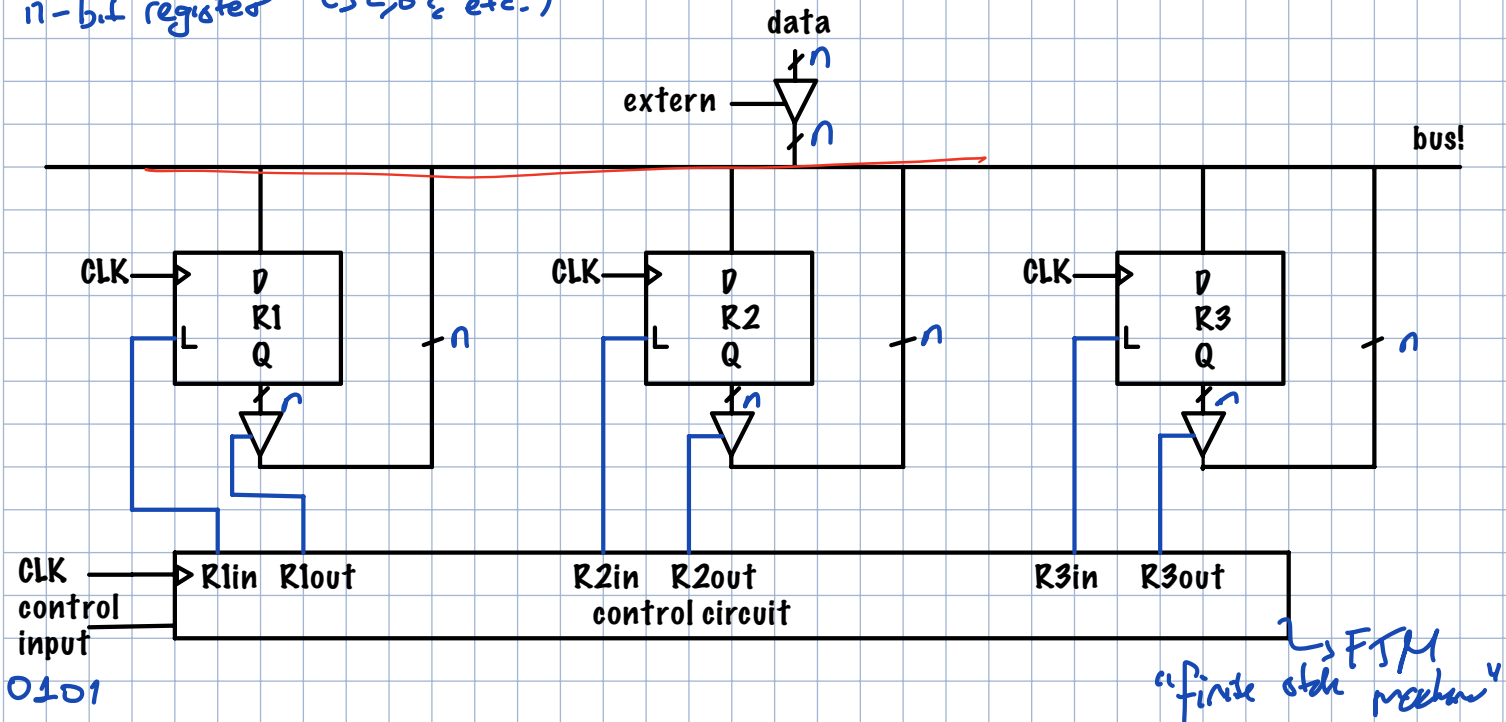
9 - 10 - 11 — 9 - 10 - 11 - 12 - 13 - 14 - 9 - -

preset = 1



Bus Architecture for Register Transfers

n-bit register (32, 64, etc.)



Control circuit must ensure that only 1 of R1out, R2out, R3out and extern is 1 at a given time.

100010 ↔ 010101

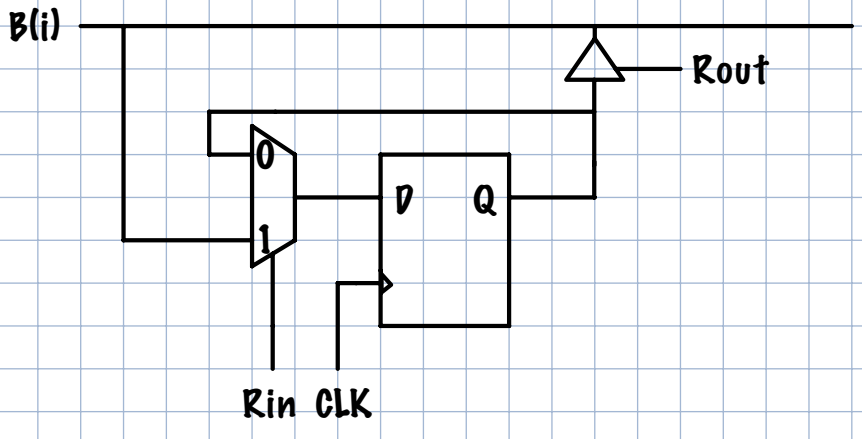
How can we swap the contents of Register 1 and Register 2: R1<->R2?
In how many CLK cycles?

1) $R_3 \leftarrow R_1$ ($R_{1out}=1$, $R_{3in}=1$, rest 0)

2) $R_1 \leftarrow R_2$ ($R_{2out}=1$, $R_{1in}=1$, rest 0)

3) $R_2 \leftarrow R_3$ ($R_{2in}=1$, $R_{3out}=1$, rest 0)

Each register in the circuit above consists of the following 1 bit cells:



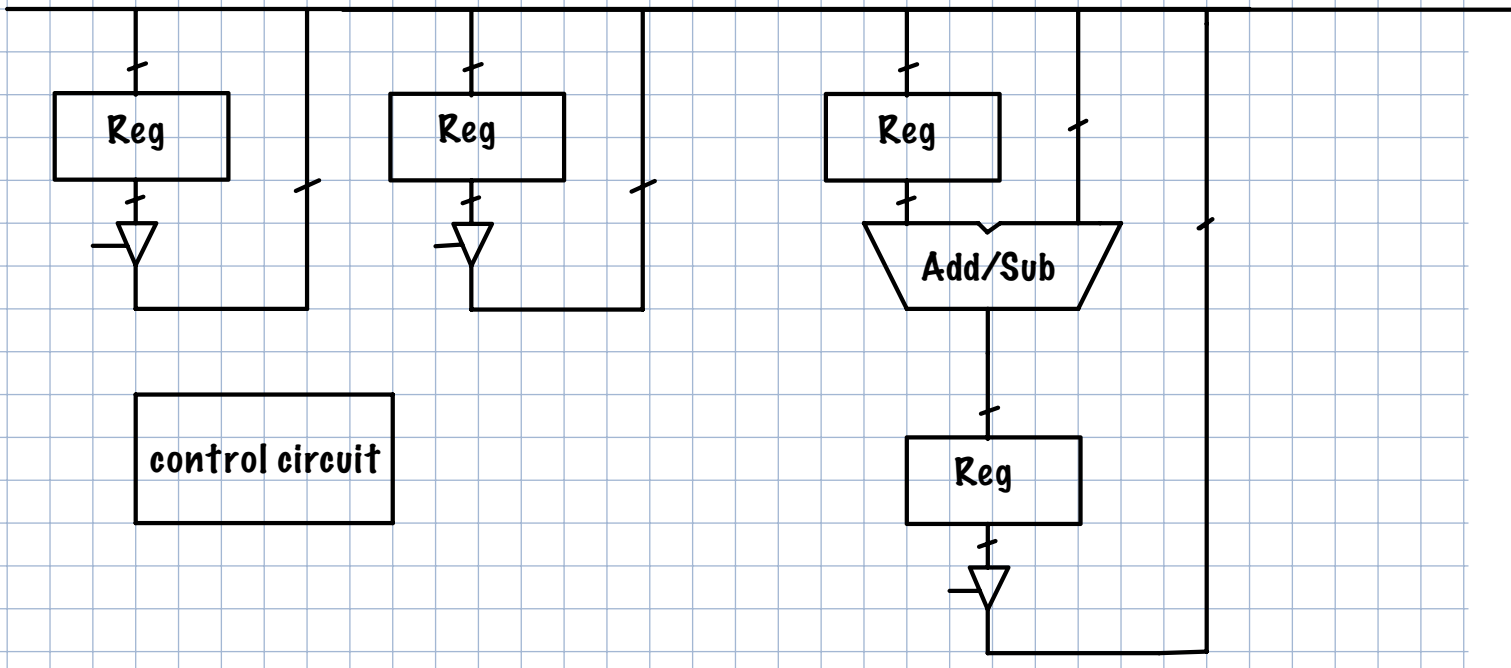
VHDL code for the cell

outputs of control circuit

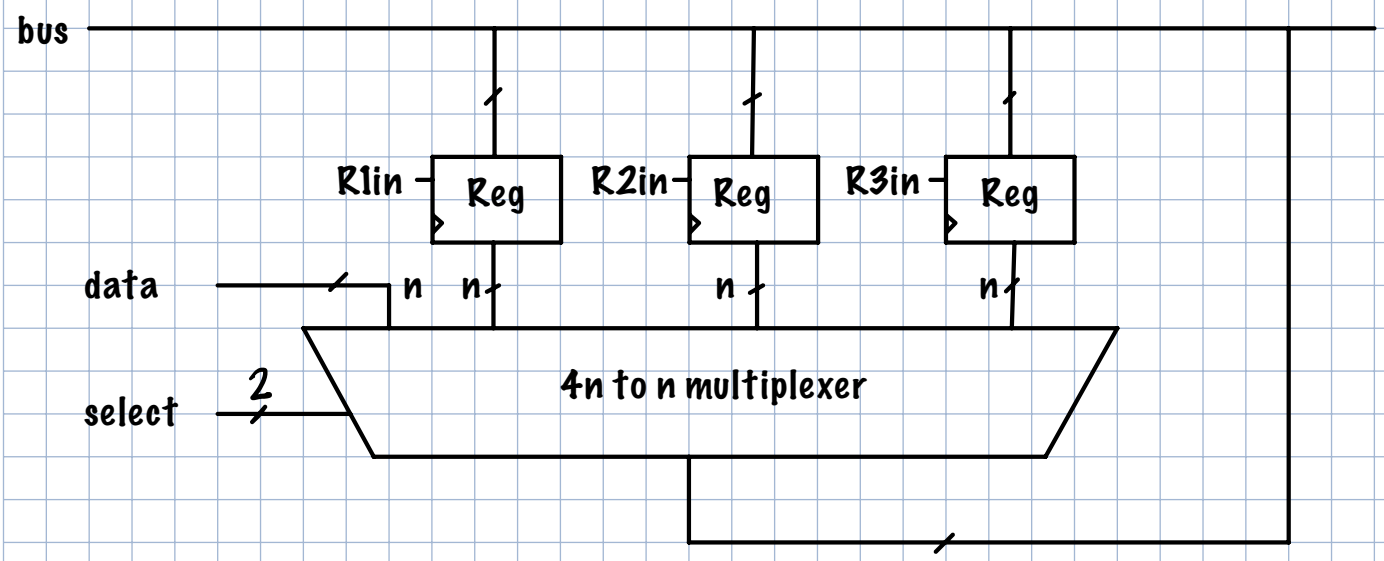
```
entity cell is
    port(Rout, Rin, CLK: in std_logic;
         B: inout std_logic);
end cell;

architecture cell_arch of cell is
    signal Q: std_logic;
begin
    B <='Z' when Rout='0' else Q;
    process(CLK) begin
        if rising_edge(CLK) then
            if Rin='1' then
                Q<=B;
            end if;
    end process;
end cell_arch;
```

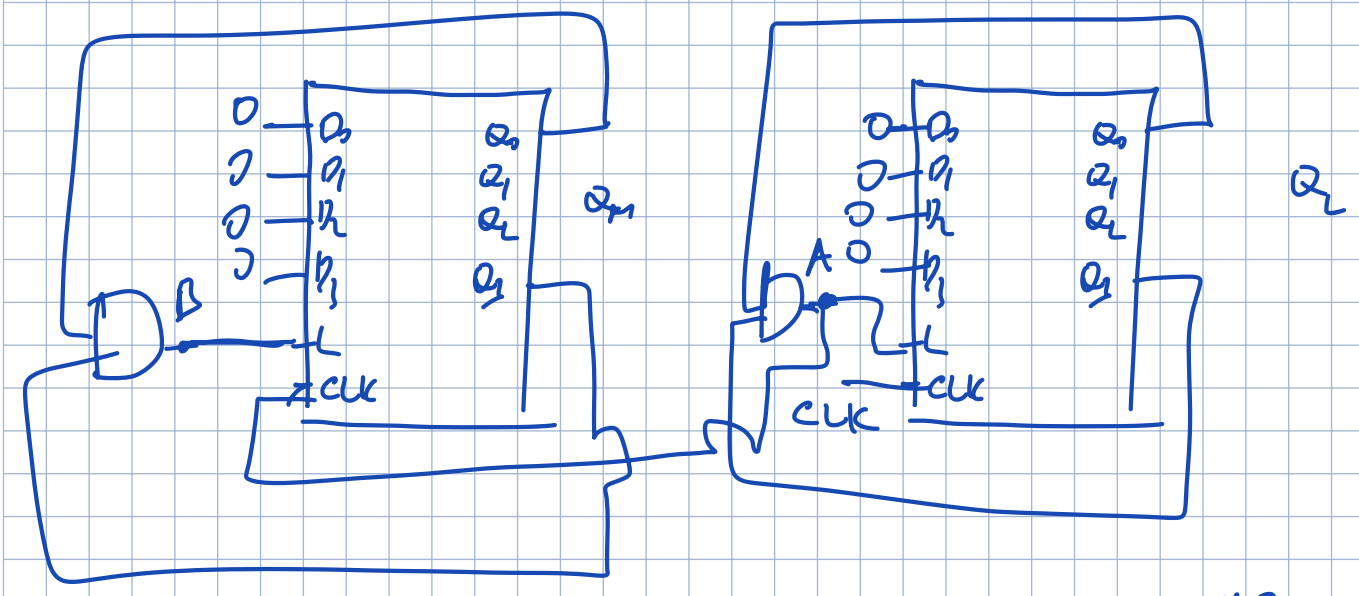
A simple microprocessor



Bus with multiplexer

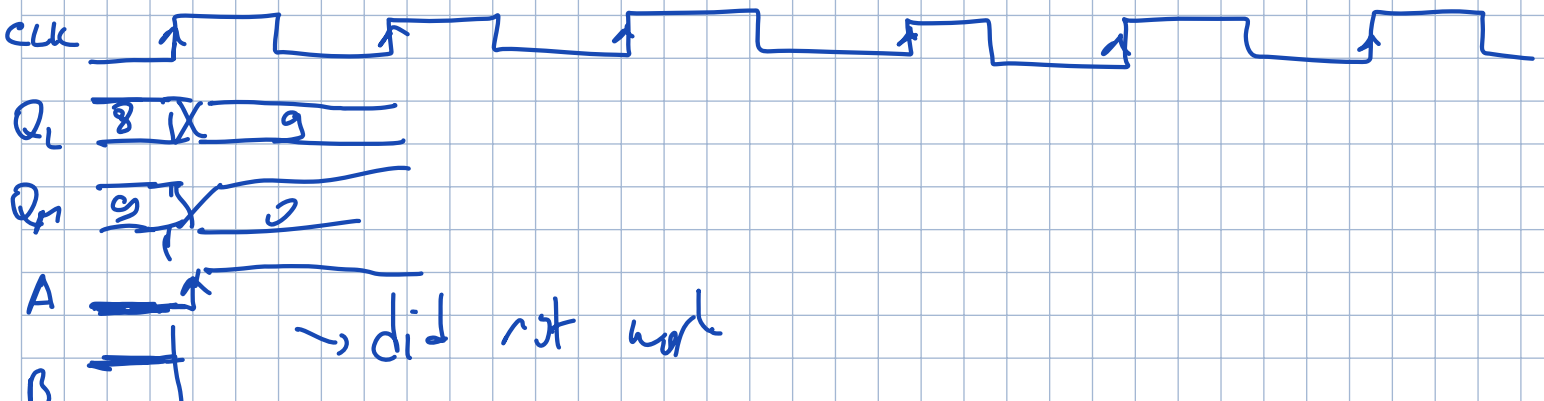


00-01--- 09-10-11-------19-20--- 99-00



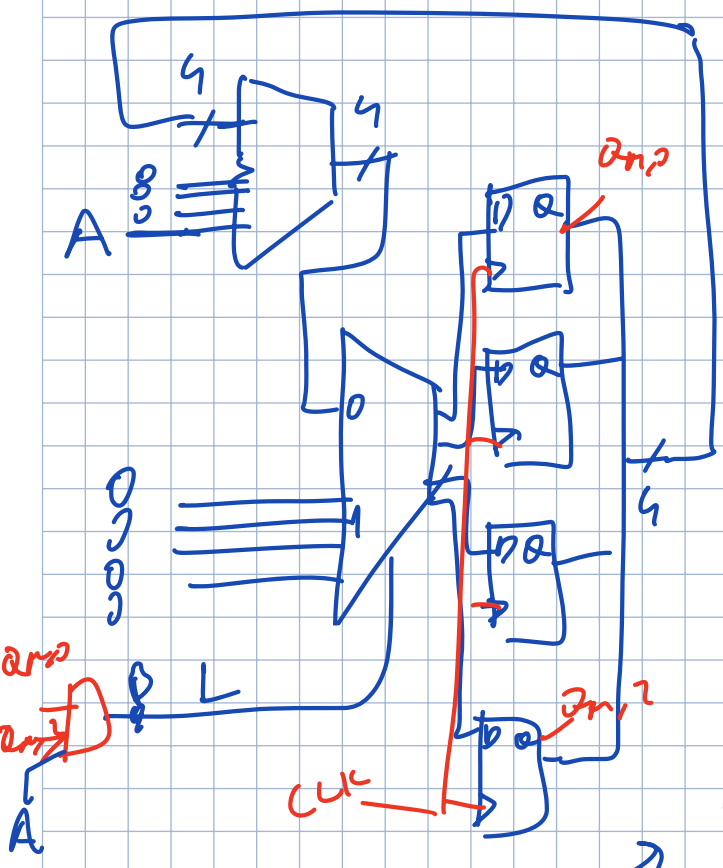
98-09

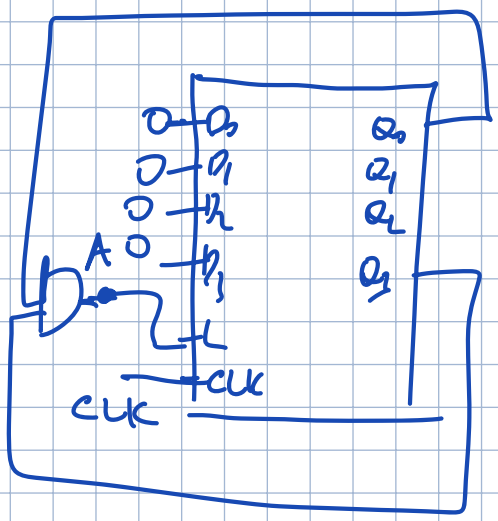
90-91-92-93--98-99



→ did not work

Working design.





09 to 10