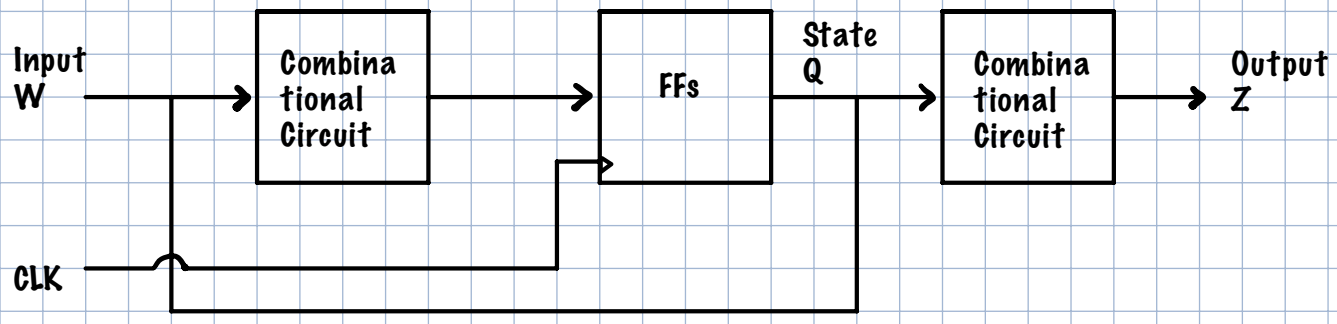
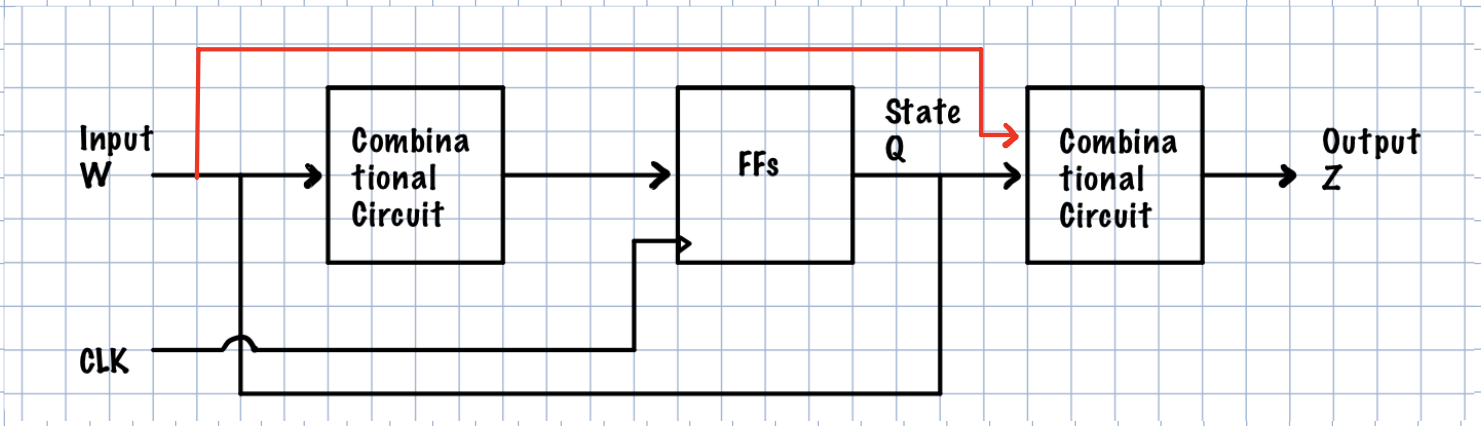
Chapter 8 - Sequential Circuit Design

Finite State Machines (FSMs)

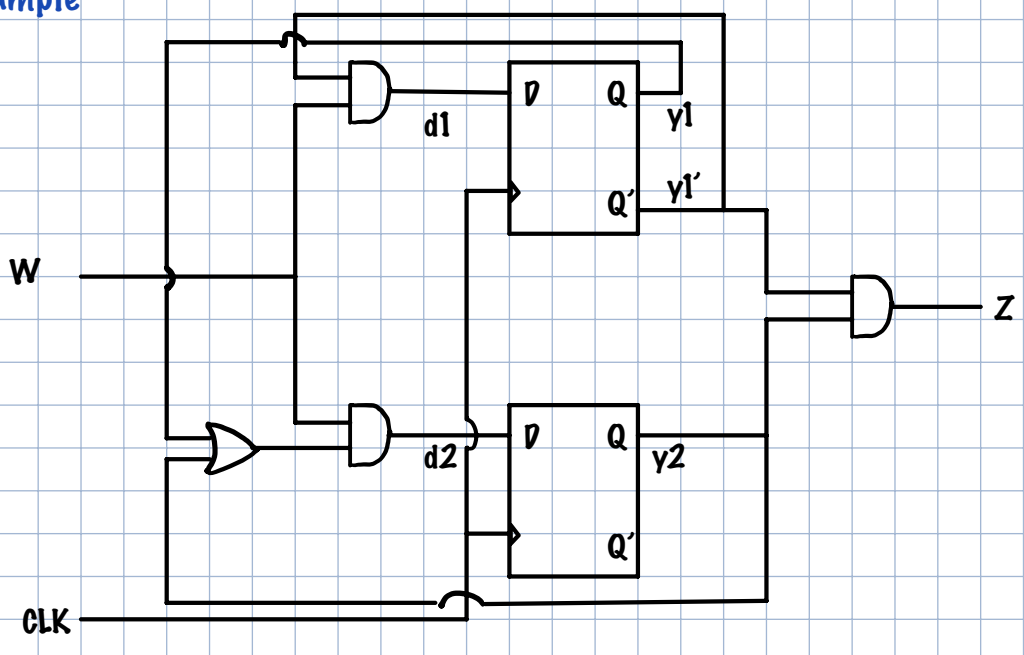
1. Moore type FSM (Edward Moore)



2. Mealy type FSM (George Mealy)

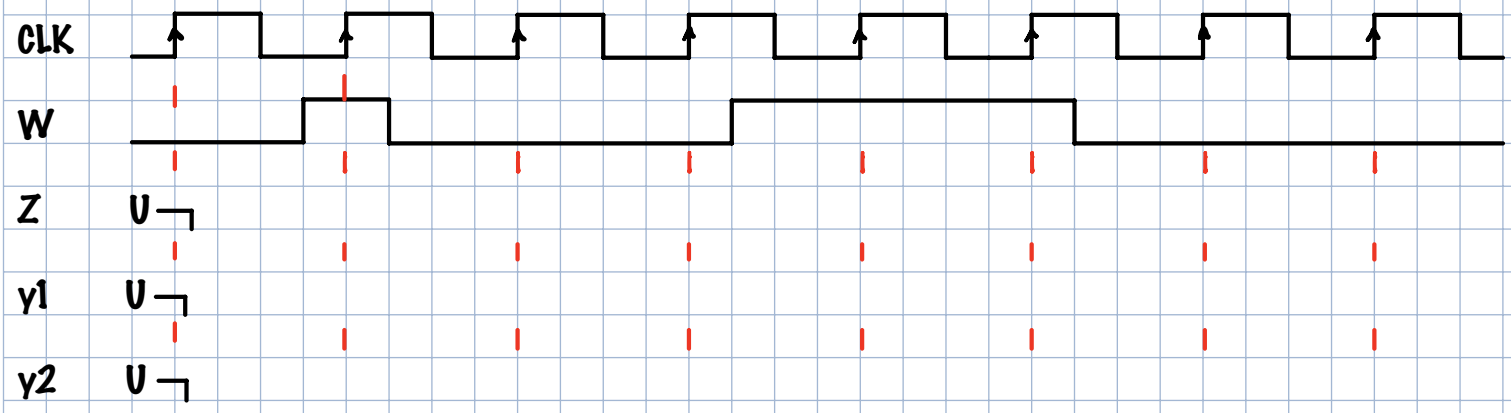


Example

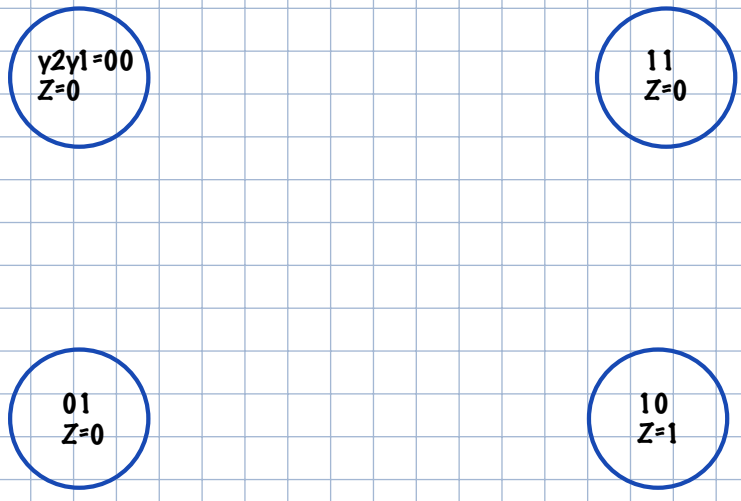


Moore or Mealy?
Output equation?
FF input Equations?

Timing diagram



State Diagram



State Table

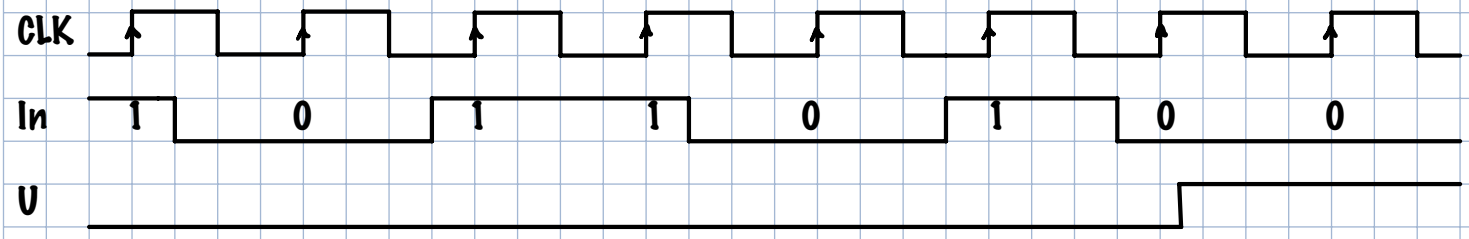
Present State y2y1	Next State w=0 y2+y1+	Next State w=1 y2+y1+	Output z
00	00	01	0
01	00	10	0
10	00	11	1
11	00	10	0

MT : Average = 70 , 23.10.2023 @ 8.30pm : midterm explanation session.

Example

Unlock key when "1011010" comes in sequence

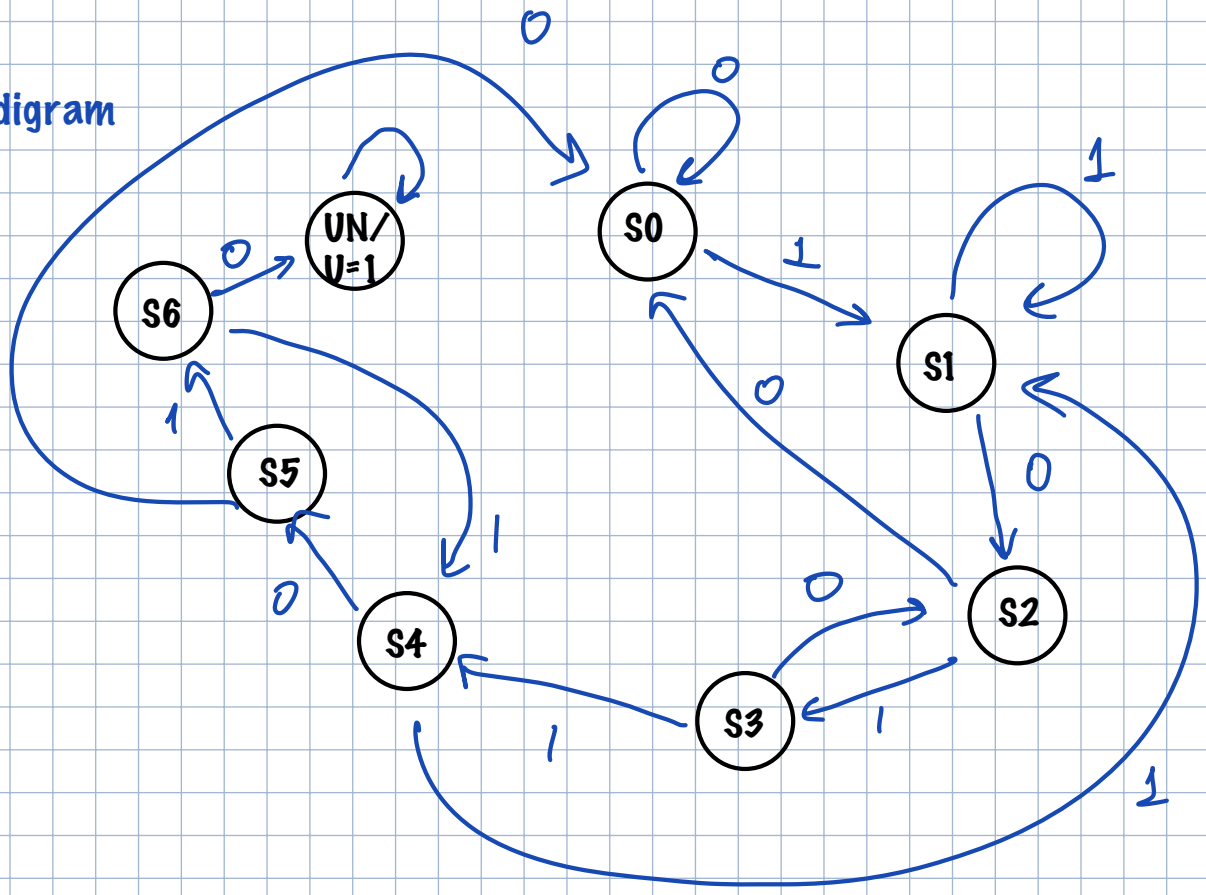
Timing diagram



What should be the states?

S0: Initial state
S1: '1'
S2: '10'
S3: '101'
S4: '1011'
S5: '10110'
S6: '101101'
UN: '1011010'

State digram



→ ceiling fn.

S: state space

How many flip flops do we need?

$\lceil \log_2 8 \rceil = \lceil 3 \rceil = 3$ FFs

$\lceil \log_2 |S| \rceil$ ↳ cardinality of S

at minimum (more is also possible)

State assignment

↳ how many possibilities = 8!

S0	S1	S2	S3	S4	S5	S6	UN
000	001	010	011	100	101	110	111

⇒ intuitive assignment

# of bits correctly entered so far

State table

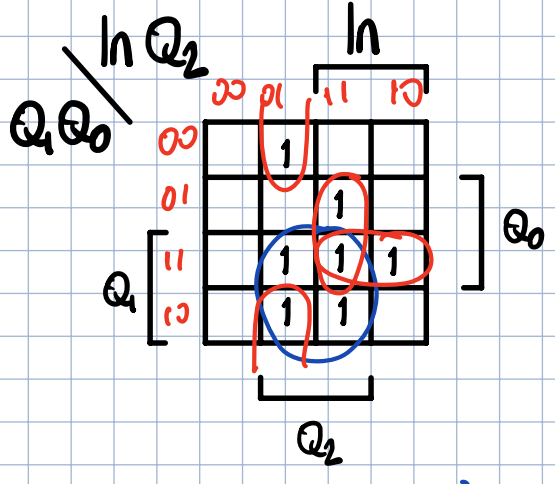
Present State Q2Q1Q0	Next State In=0 (Q2Q1Q0)+	Next State In=1 (Q2Q1Q0)+	Output U
000 (S0)	000 (S0)	001 (S1)	0
001 (S1)	010 (S2)	001 (S1)	0
010 (S2)	000 (S0)	011 (S3)	0
011 (S3)	010 (S2)	100 (S4)	0
100 (S4)	101 (S5)	001 (S1)	0
101 (S5)	000 (S0)	110 (S6)	0
110 (S6)	111 (UN)	100 (S4)	0
111 (UN)	111 (UN)	111 (UN)	1

→ single column since from machine

$U = Q_2 Q_1 Q_0$

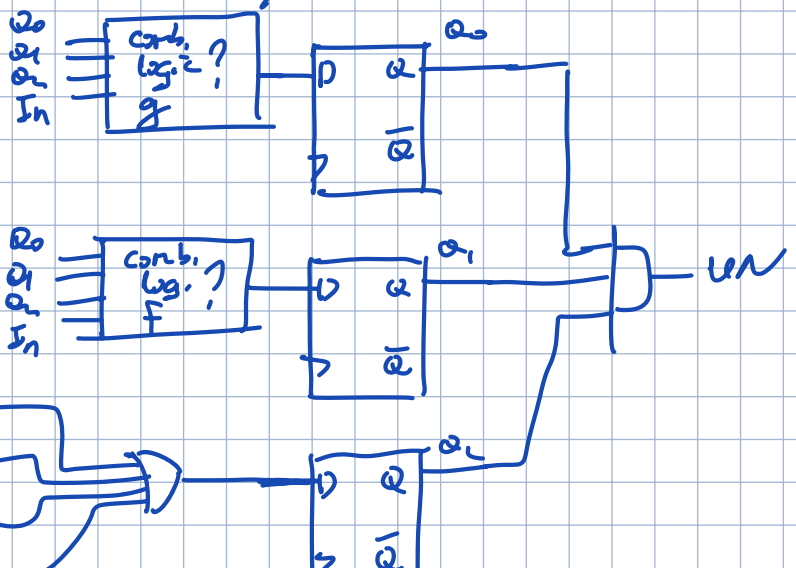
Let's use D-FFs to implement the circuit. We need to find FF input equations D2D1D0 since we now that Q(t+1) = D

KMAP for D2

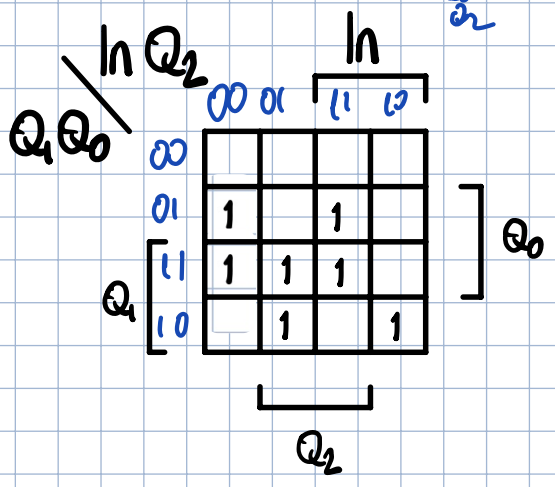


$D_2 = Q_1 Q_2 + In\, Q_0 Q_2 + In\, Q_1 Q_0 + \overline{In}\, \overline{Q_1} Q_2$

→ exercise: find these.

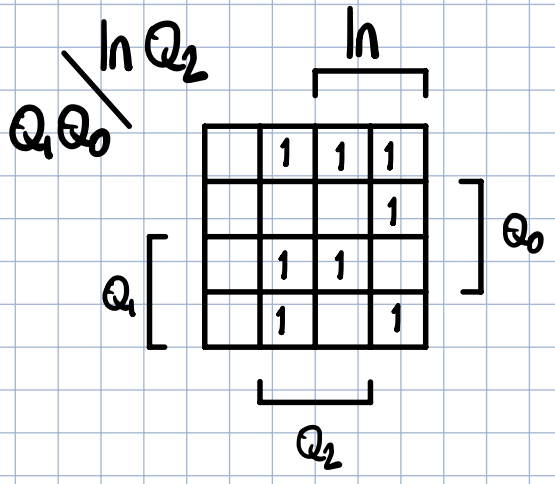


KMAP for D1



$\Rightarrow D_1 = f(Q_0, Q_1, Q_2, I_n)$

KMAP for D0



$\Rightarrow D_0 = g(Q_0, Q_1, Q_2, I_n)$

Let's use T-FFs to implement the circuit. We need to find FF input equations T2 T1 T0

Characteristic table

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

Excitation table

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Present State Q2Q1Q0	Next State In=0 (Q2Q1Q0)+	T2T1T0	In=1 (Q2Q1Q0)+	T2T1T0	Output U
000 (S0)	000 (S0)	000	001 (S1)	001	0
001 (S1)	010 (S2)	011	001 (S1)	000	0
010 (S2)	000 (S0)	010	011 (S3)	001	0
011 (S3)	010 (S2)	001	100 (S4)	111	0
100 (S4)	101 (S5)	001	001 (S1)	101	0
101 (S5)	000 (S0)	101	110 (S6)	011	0
110 (S6)	111 (UN)	001	100 (S4)	010	0
111 (UN)	111 (UN)	000	111 (UN)	000	1

InQ2/ Q1Q0 — T0

	00	01	11	10
00	0	1	1	1
01	1	1	1	0
11	1	0	0	1
10	0	1	0	1

InQ2/ Q1Q0 — T1

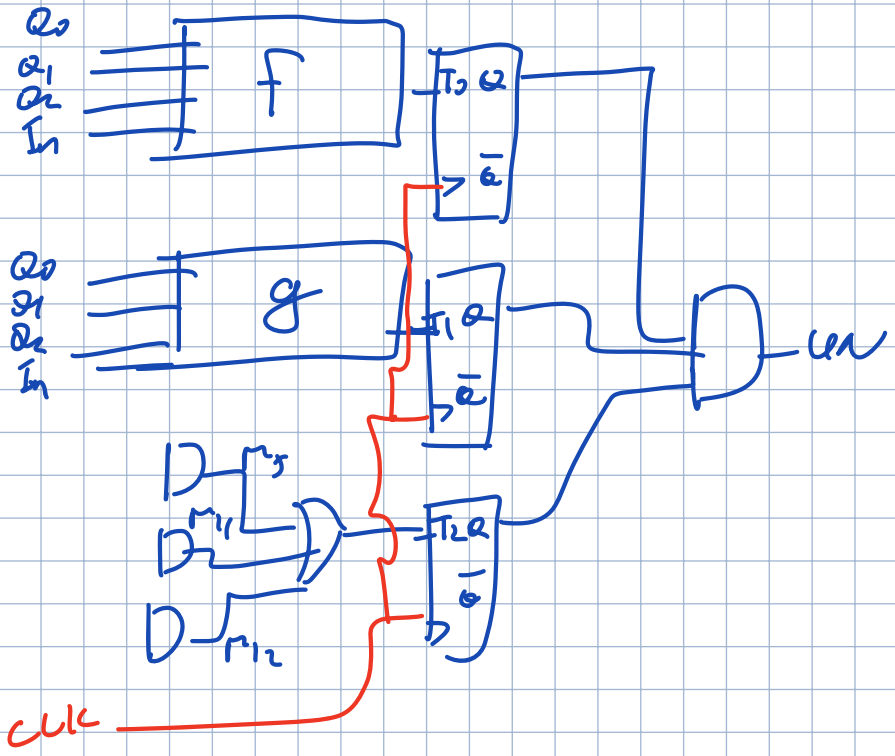
	00	01	11	10
00	0	0	0	1
01	1	0	1	0
11	0	0	0	1
10	1	0	1	0

InQ2/ Q1Q0 — T2

1100

	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	0	0	0	1
10	0	0	0	0

$T_0 = f(Q_0, Q_1, Q_2, In)$ $T_1 = g(Q_0, Q_1, Q_2, In)$ $T_2 = m_5 + m_{11} + m_{12}$



Design using JK FFs

Characteristic table

JK	Q(t+1)	
00	Q(t)	no change
01	0	reset
10	1	set
11	Q'(t)	toggle

Excitation table

Q(t)	Q(t+1)	JK	
0	0	0X	00 or 01
0	1	1X	10 or 11
1	0	X1	01 or 11
1	1	X0	00 or 10

Present State Q2Q1Q0	In=0 (Q2Q1Q0)+	J2K2	J1K1	J0K0	In=1 (Q2Q1Q0)+	J2K2	J1K1	J0K0	Output U
000 (S0)	000 (S0)	0X	0X	0X	001 (S1)	0X	0X	1X	0
001 (S1)	010 (S2)	0X	1X	X1	001 (S1)	0X	0X	X0	0
010 (S2)	000 (S0)	0X	X1	0X	011 (S3)	0X	X0	1X	0
011 (S3)	010 (S2)	0X	X0	X1	100 (S4)	1X	X1	X1	0
100 (S4)	101 (S5)	X0	0X	1X	001 (S1)	X1	0X	1X	0
101 (S5)	000 (S0)	X1	0X	X1	110 (S6)	X0	1X	X1	0
110 (S6)	111 (UN)	X0	X0	X1	100 (S4)	X0	X1	0X	0
111 (UN)	111 (UN)	X0	X0	X0	111 (UN)	X0	X0	X0	1

InQ2/ Q1Q0 — J2

	00	01	11	10
00	0	X	X	0
01	0	X	X	0
11	0	X	X	1
10	0	X	X	0

In, Q2, Q1

$J_2 = I_n \cdot Q_1 \cdot Q_0$

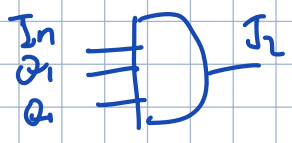
InQ2/ Q1Q0 — K2

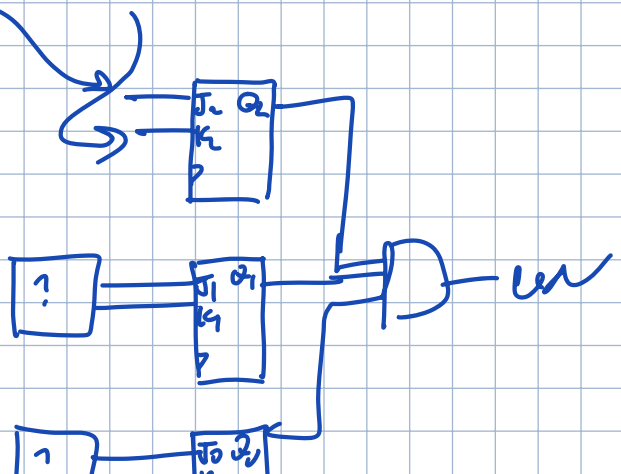
	00	01	11	10
00	X	0	1	X
01	X	1	0	X
11	X	0	0	X
10	X	0	0	X

$\bar{I}_n Q_0 \bar{Q}_1$

$I_n \bar{Q}_0 \bar{Q}_1$

$K_2 = \bar{I}_n Q_0 \bar{Q}_1 + I_n \bar{Q}_0 \bar{Q}_1$





VHDL code for the lock

'1011010'

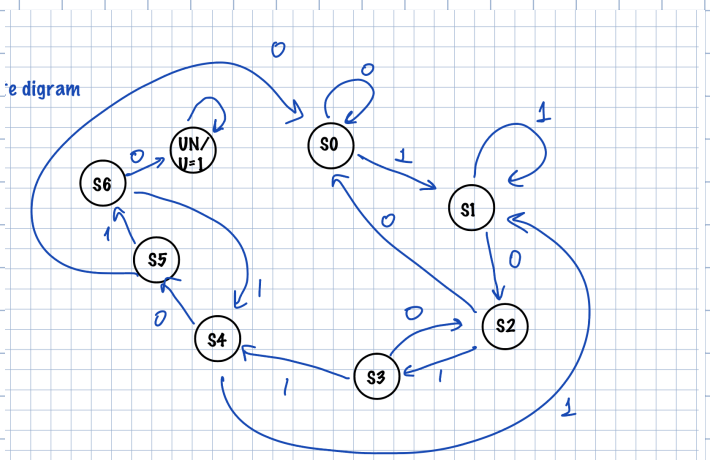
```
entity lock is
    port(I,CLK: in std_logic;
         U: out std_logic);
end lock;

architecture lock_arch of lock is
    type state_type is (S0,S1,S2,S3,S4,S5,S6,UN);
    signal next_state, state: state_type;
begin
    process(CLK) begin
        if rising_edge(CLK) then
            state <= next_state;
        end if;
    end process;
    U<='1' when state=UN else '0';
    process(state,I) begin
        case state is
            when S0=>
                if I='0' then next_state<=S0;
                else next_state<=S1; end if;
            when S1=>
                if I='0' then next_state<=S2;
                else next_state<=S1; end if;
            ...
            ...
            ...
            when others=>
                next_state<=S0;
    end process;
end lock_arch;
```

state transition logic

output logic

next-state logic

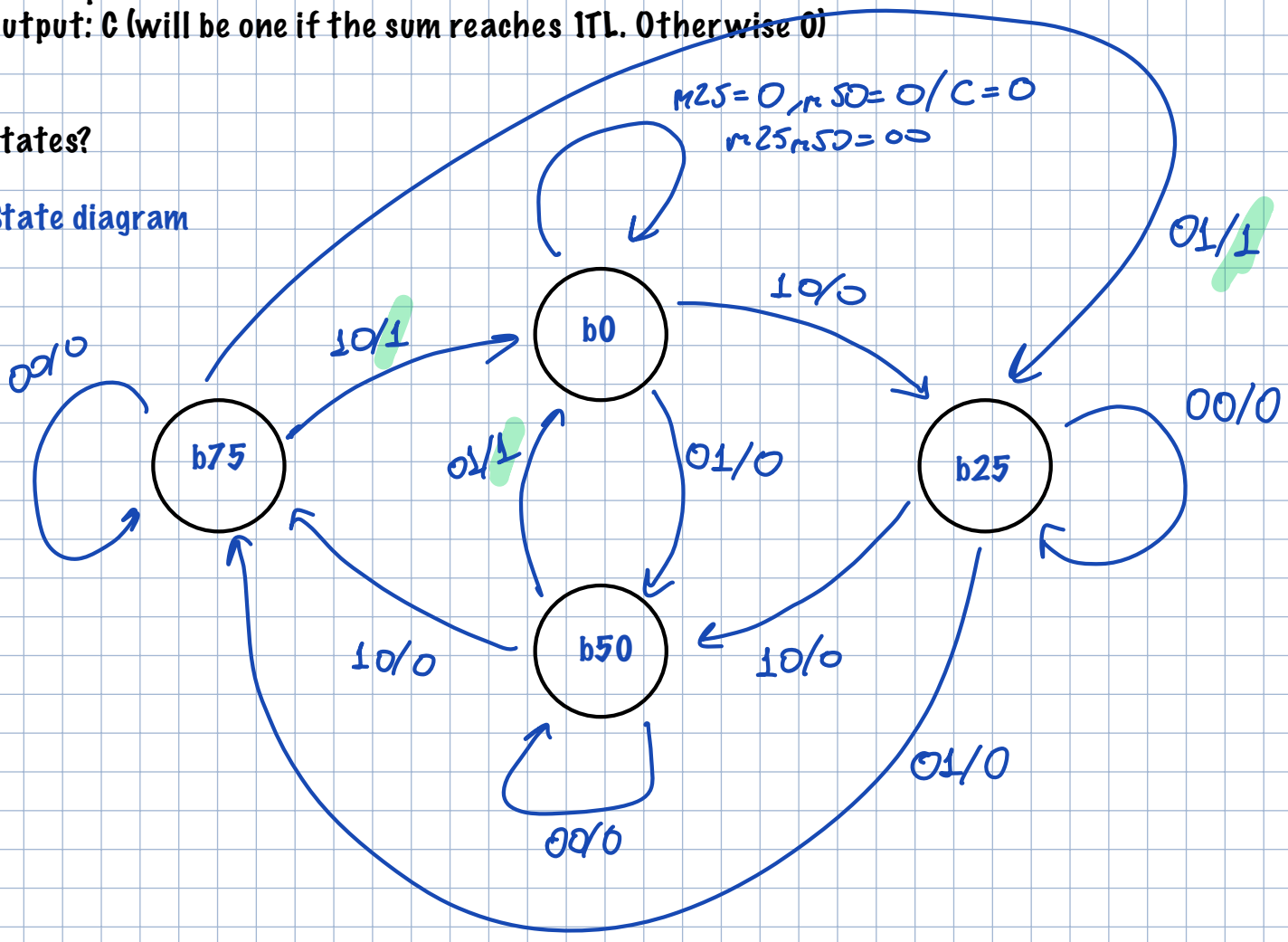


Mealy state machine example: Coffee wending machine

output = func(state,in)
inputs: 25kr (m25), 50kr (m50) [in a particular time slot, at most one of m25 and m50 is 1]
coffee price: 1TL
output: C (will be one if the sum reaches 1TL. Otherwise 0)

states?

State diagram



In Mealy machine, output is defined on branches. In Moore machine, output is defied on states.

Coffee wending machine design using D-FFs: => 2FFs Q_1Q_0

State assignment 4! possibilities

	Q1Q0
b0	00
b25	01
b50	10
b75	11

intuitive selection

State table

an input that never happens

Present State Q1Q0	Next State (Q1Q0)+ m50m25: 00	01	11	10	Output m50m25: 00	01	11	10
00 (b0)	00	01	XX	10	0	0	X	0
01 (b25)	01	10	XX	11	0	0	X	0
11 (b75)	11	00	XX	01	0	1	X	1
10 (b50)	10	11	XX	00	0	0	X	1

Q_1Q_0m25 Q_1m50

output logic $C = Q_1m50 + Q_1Q_0m25$

FF input equations and output equation ⇒ Since D-FFs Q^+ or $Q(t+1) = D(t)$

m50m25/ Q1Q0 — D1

Q1Q0 \ m50m25	00	01	11	10
00	0	0	X	1
01	0	1	X	1
11	1	0	X	0
10	1	1	X	0

$\bar{Q}_1m50$ $\bar{Q}_1Q_0\bar{m25}$ $Q_1\bar{m50}\bar{m25}$ $Q_1\bar{Q}_0\bar{m50}$

m50m25/ Q1Q0 — D0

Q1Q0 \ m50m25	00	01	11	10
00	0	1	X	0
01	1	0	X	1
11	1	0	X	1
10	0	1	X	0

$Q_0\bar{m25}$ $\bar{Q}_0m25$

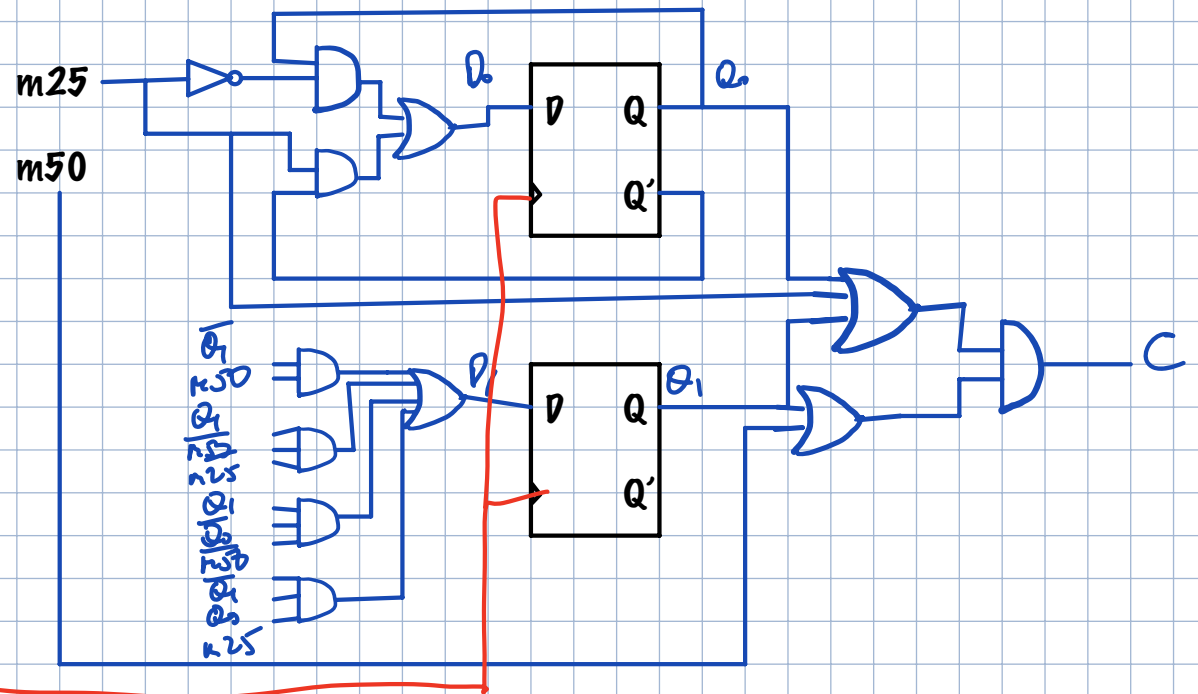
m50m25/ Q1Q0 — C

Q1Q0 \ m50m25	00	01	11	10
00	0	0	X	0
01	0	0	X	0
11	0	1	X	1
10	0	0	X	1

$D_1 = \bar{Q}_1m50 + Q_1\bar{m50}\bar{m25} + Q_1\bar{Q}_0\bar{m50} + \bar{Q}_1Q_0m25$

$D_0 = Q_0\bar{m25} + \bar{Q}_0m25 = Q_0 \oplus m25$

Circuit



VHDL code for the wending machine

```
entity wm is
   port(m25, m50, CLK: in std_logic;
        c: out std_logic);
end wm;

architecture wm_arch of wm is
   type state_type is (b0,b25,b50,b75);
   signal next_state, state: state_type;
begin
   process(CLK) begin
      if rising_edge(CLK) then
         case state is
            when b0=>
               if (m25='0' and m50='0') then
                  state <=b0;
               elsif (m25='1' and m50='0') then
                  state <= b25;
               ...
               ...
               end if;
            when b1=>
            ...
            ...
            ...
         end case;
      end if;
   end process;

   process(state,m25,m50) begin
      case state is
         when b0=>
            C<='0';
         ...
         ...
         when b75=>
            if (m25='1' or m50='1') then
               C<='1';
            else
               C<='0';
            end if;
      end case;
   end process;
end wm_arch;
```

state transition logic

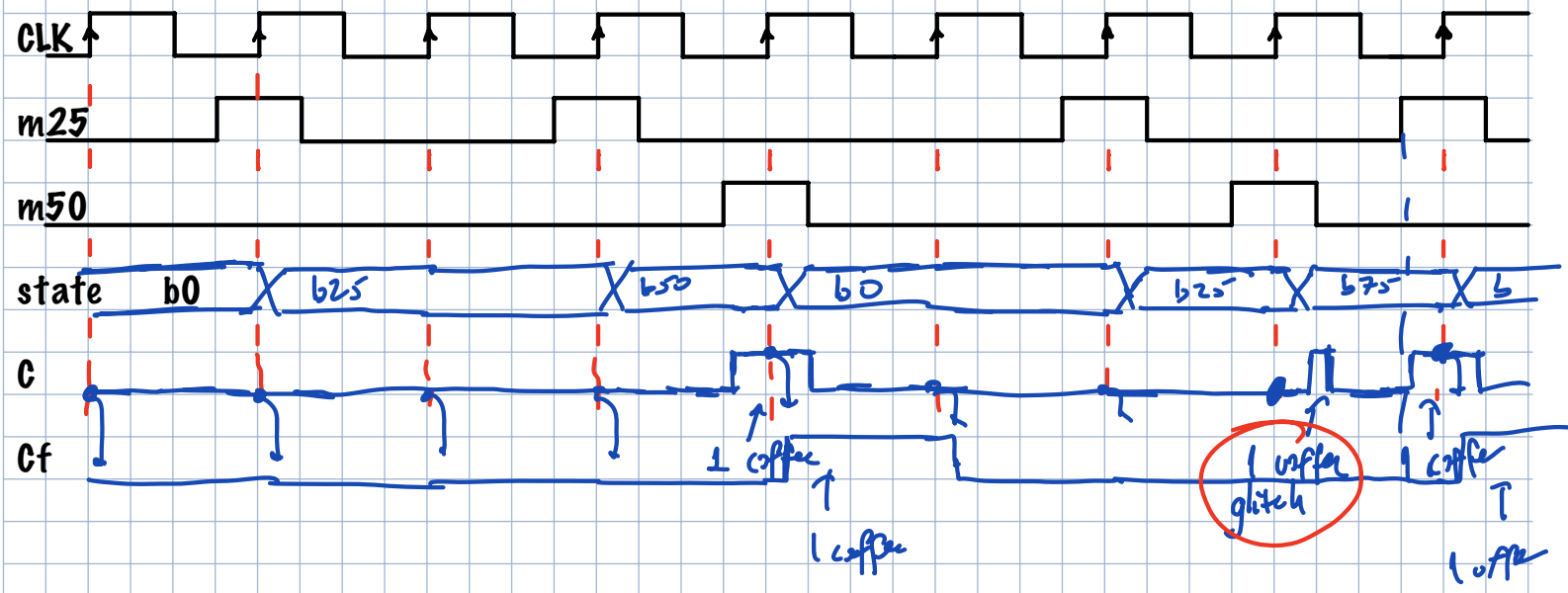
sequential logic

output logic

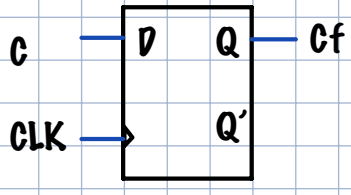
=> combinational log.

Timing diagram

$C = Q_1 m50 + Q_2 Q_3 m25$

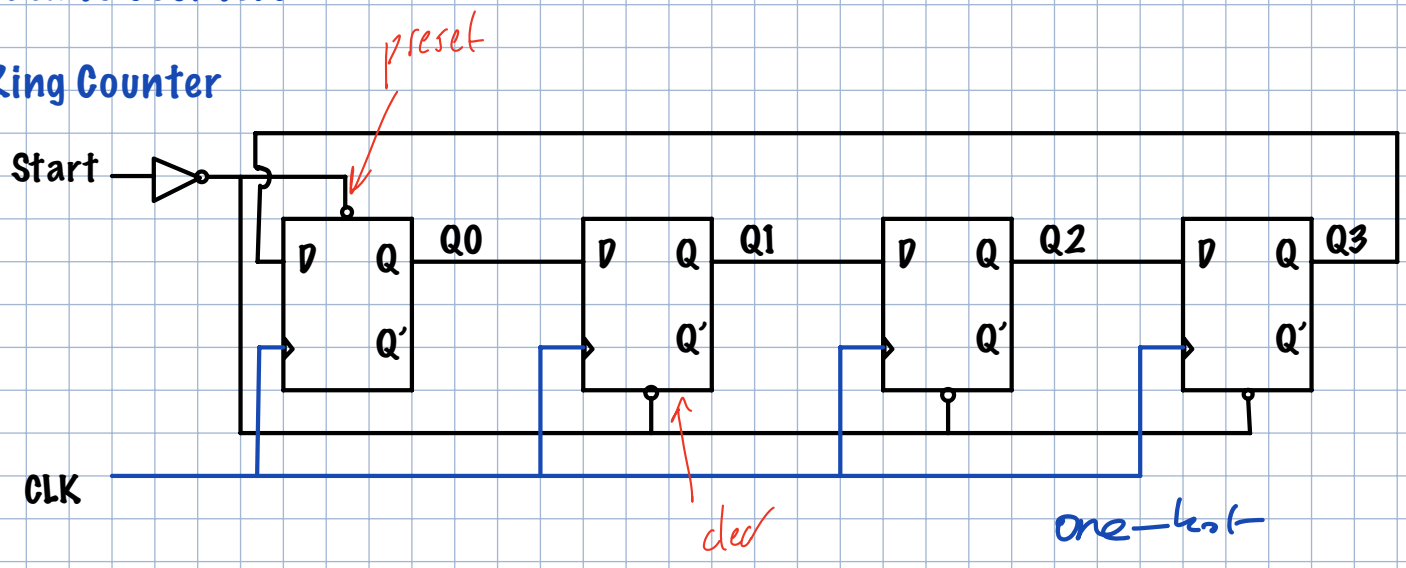


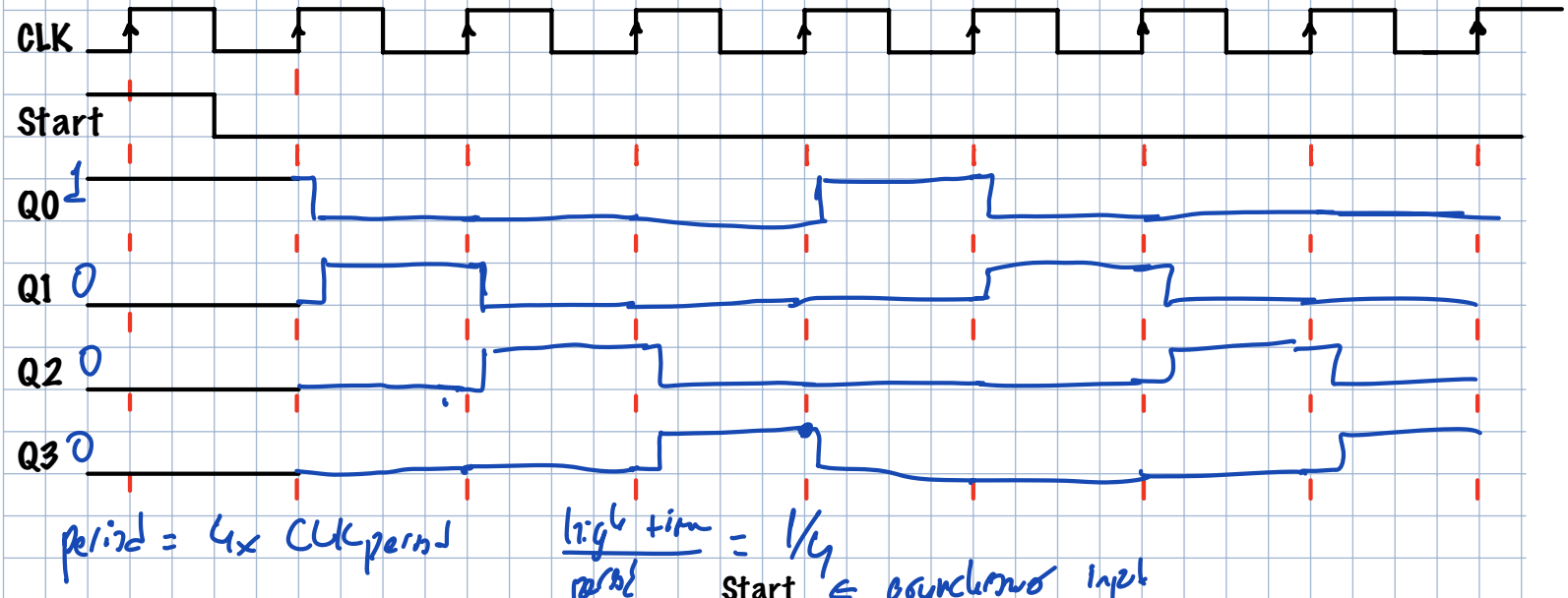
Glitch removal circuit



Back to Counters

Ring Counter





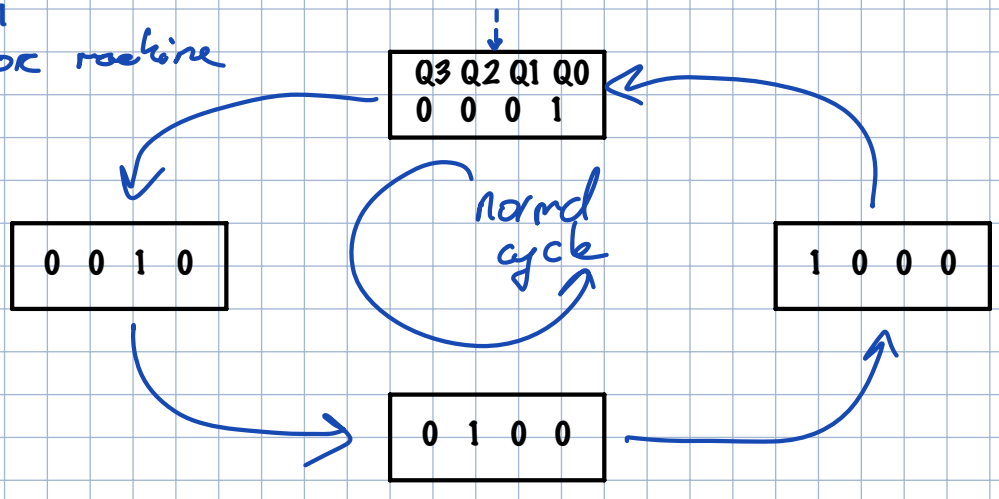
period = 4x CLK period

$\frac{\text{high time}}{\text{period}} = 1/4$

Start ← asynchronous input

State Diagram

* still a Moore machine



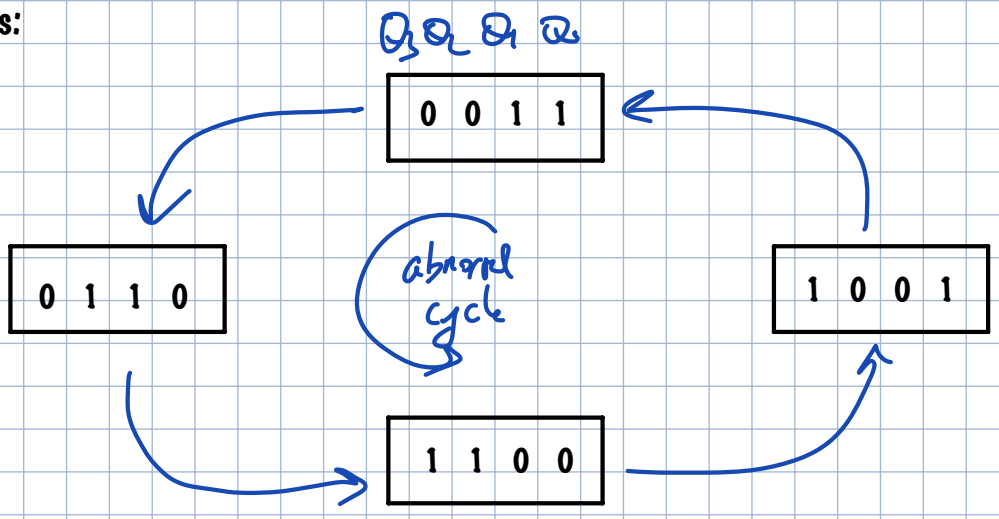
This is the normal mode of counting. In the normal mode, states are one-hot.

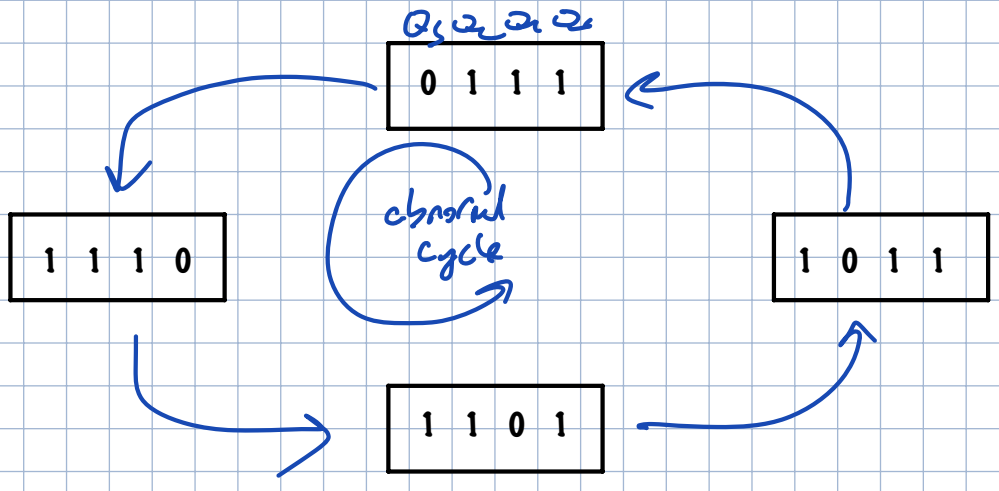
What is the total number of states?

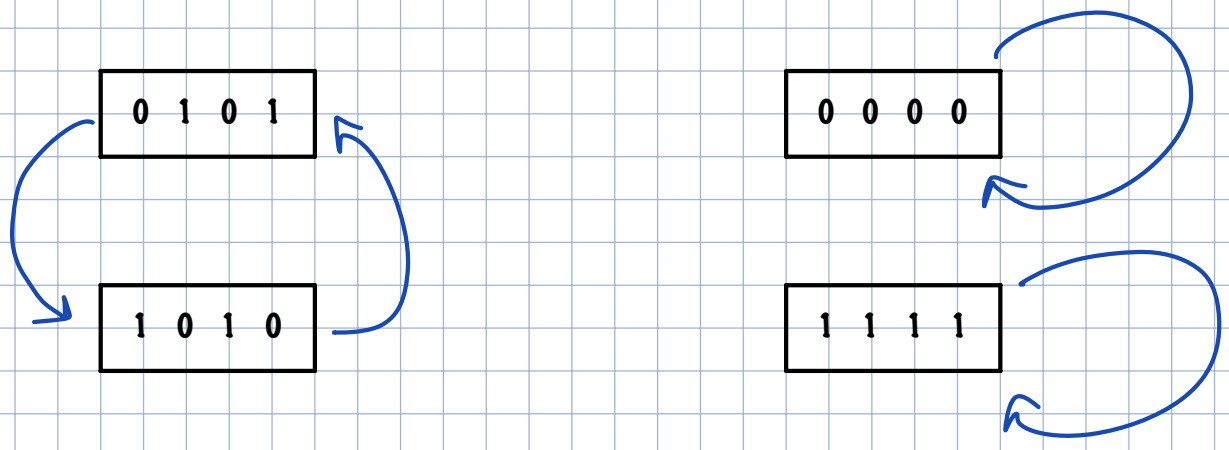
What will happen if we enter a state which is not given above?

an invalid state

Other modes:

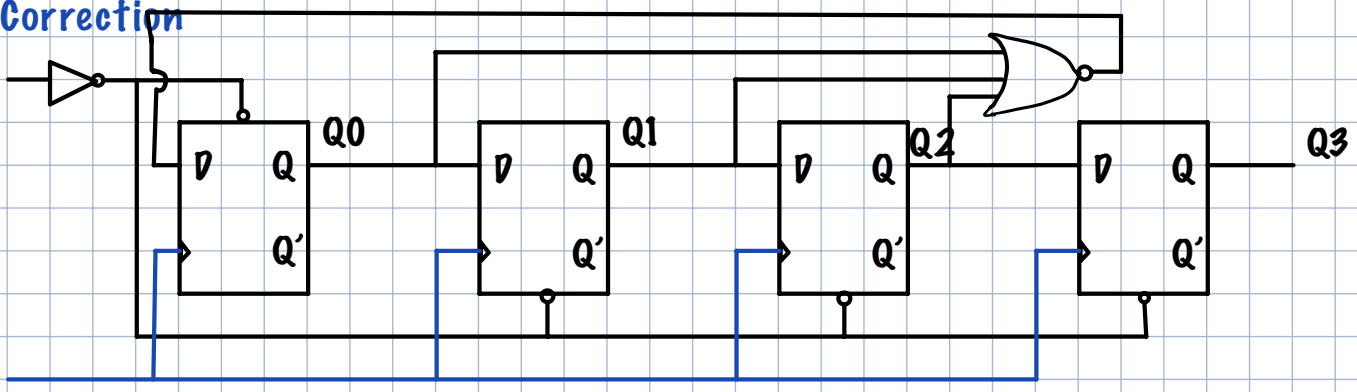






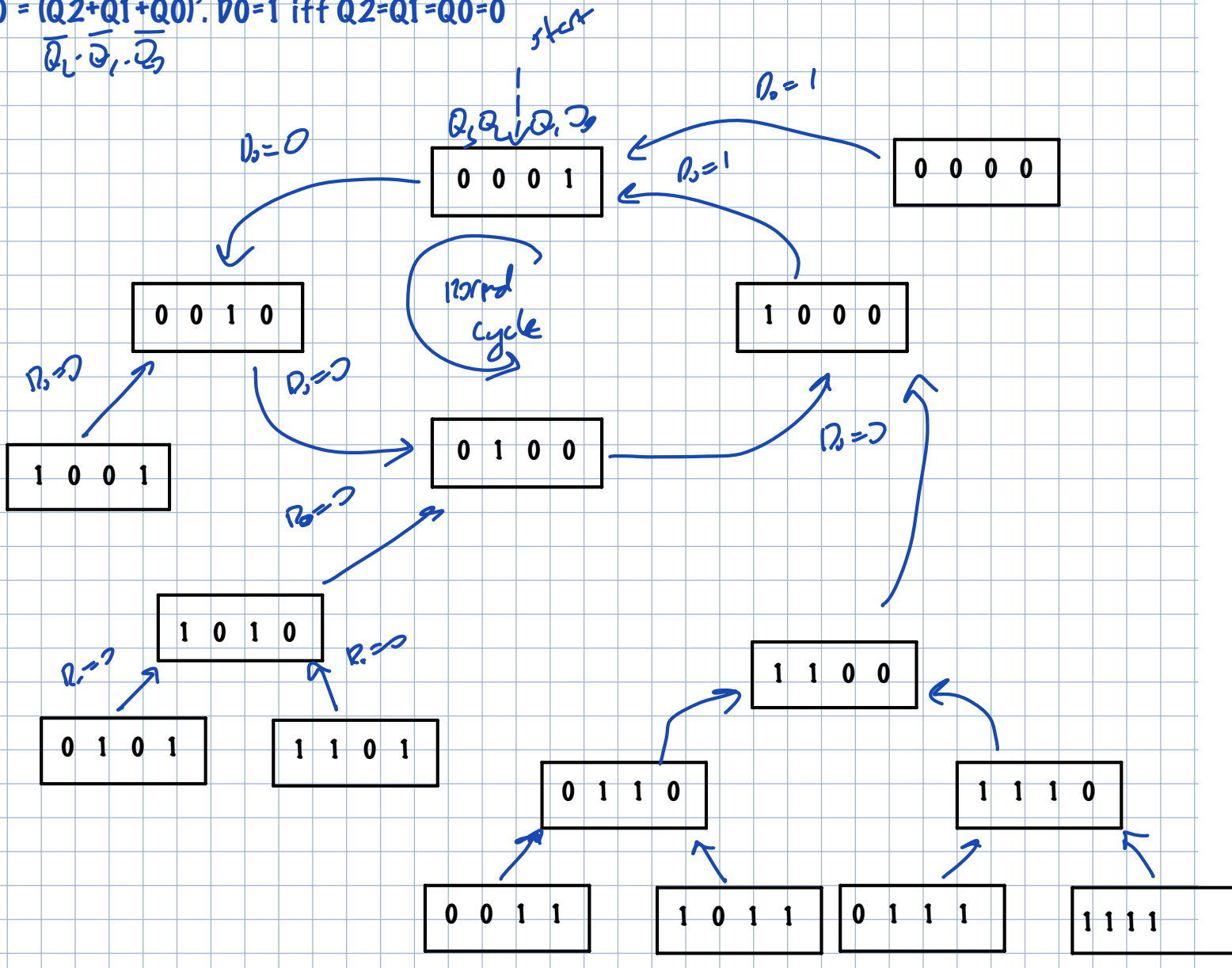
There are 6 modes of counting. 5 of them are abnormal cycles.

Self Correction



D0 = (Q2+Q1+Q0)'. D0=1 iff Q2=Q1=Q0=0

$\bar{Q_2} \cdot \bar{Q_1} \cdot \bar{Q_0}$



Reaches to 0001 from any state in at most 4 clock ticks. Corrects abnormal state in at most 3 clock ticks.

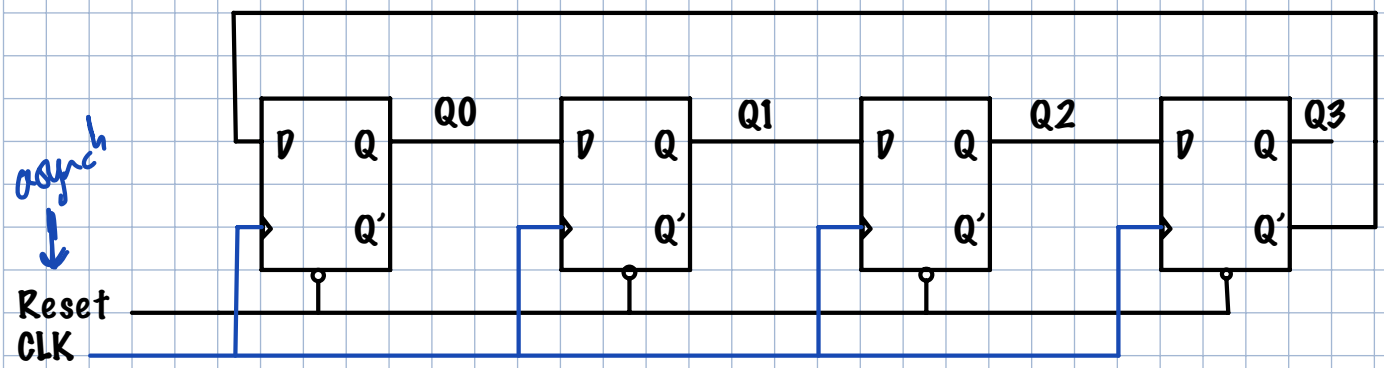
How can we construct n-bit self correcting ring counter?

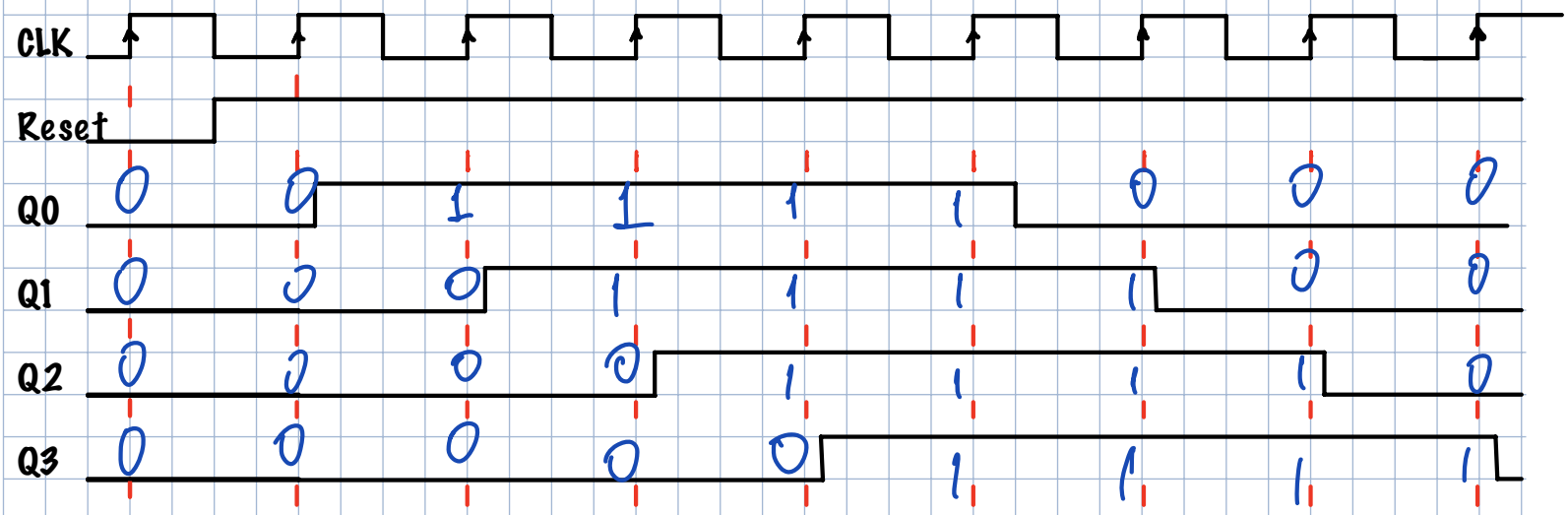
$Q_{n-1}\ Q_{n-2} - - Q_1 Q_0$

$D_0 = \overline{(Q_{n-2} + \dots + Q_1 + Q_0)}$ → n-1 input NOR

$D_1 = Q_0$, $D_2 = Q_1$, - - , $D_{n-1} = Q_{n-2}$

Johnson counter

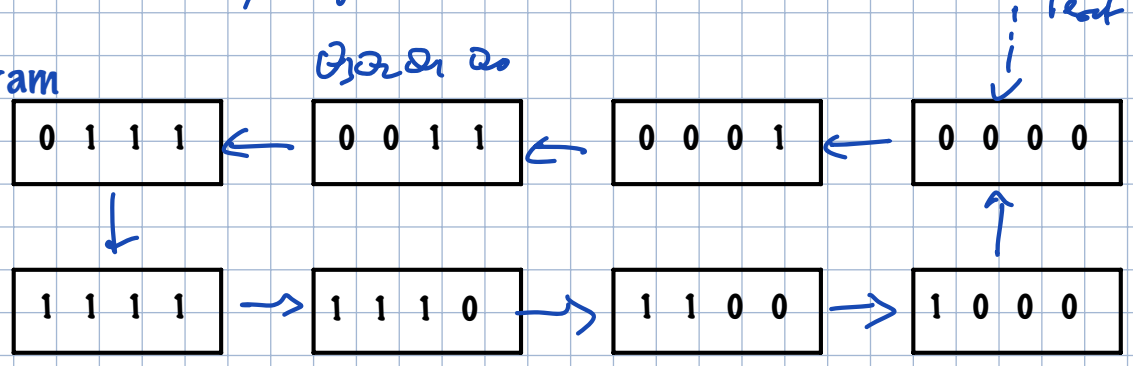




0000 → 0001 → 0011 → 0111 → 1111 → 1110 → 1100 → 1000 → 0000

period = 8 CLK , high time = 4 CLK

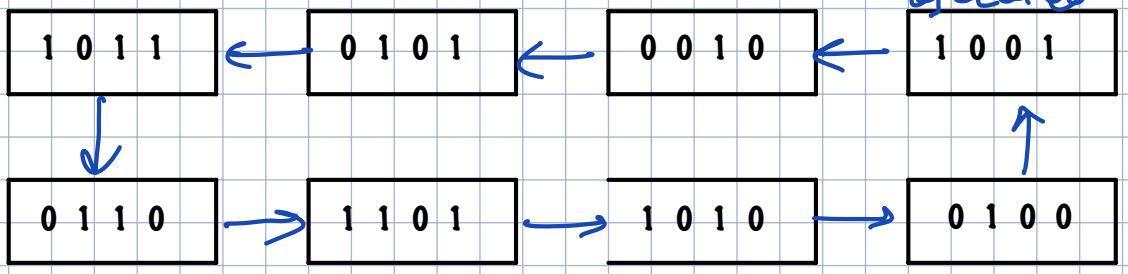
State Diagram



normal cycle

$D_2 = Q_3 Q_1 + Q_2 Q_1$

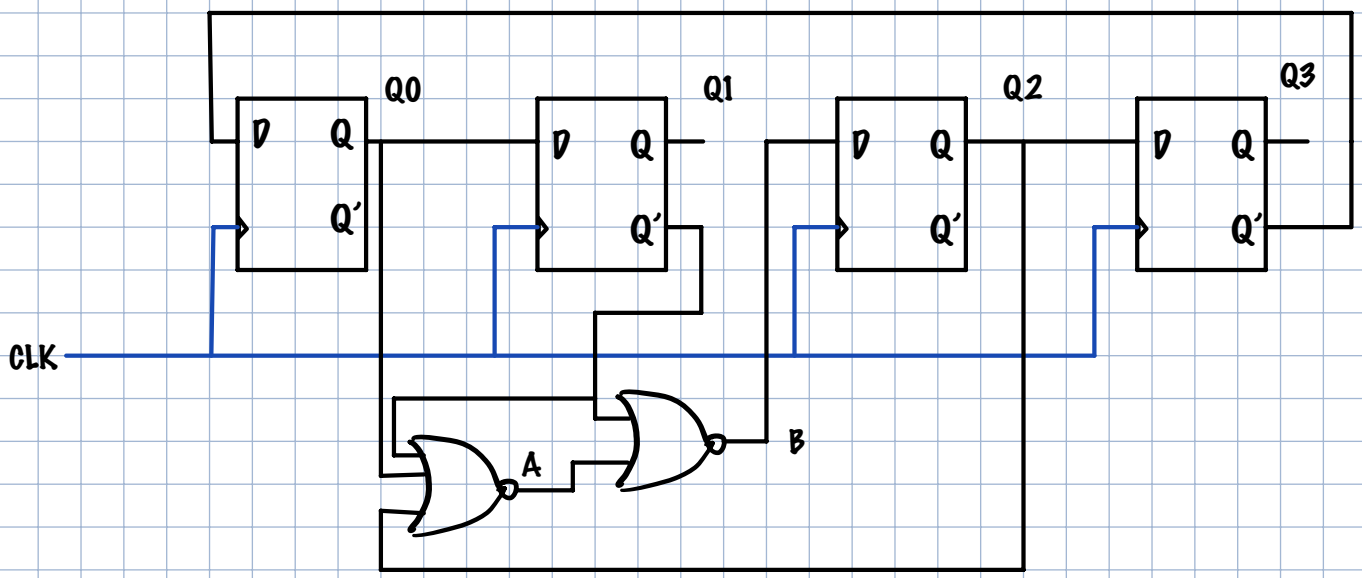
Note: Only a single bit changes its value during successive clock ticks. 8 states used in normal operation. 8 states unused.



changed cycle

n bit Johnson counter: Will go through a 2n state sequence during normal operation.

Self Correction

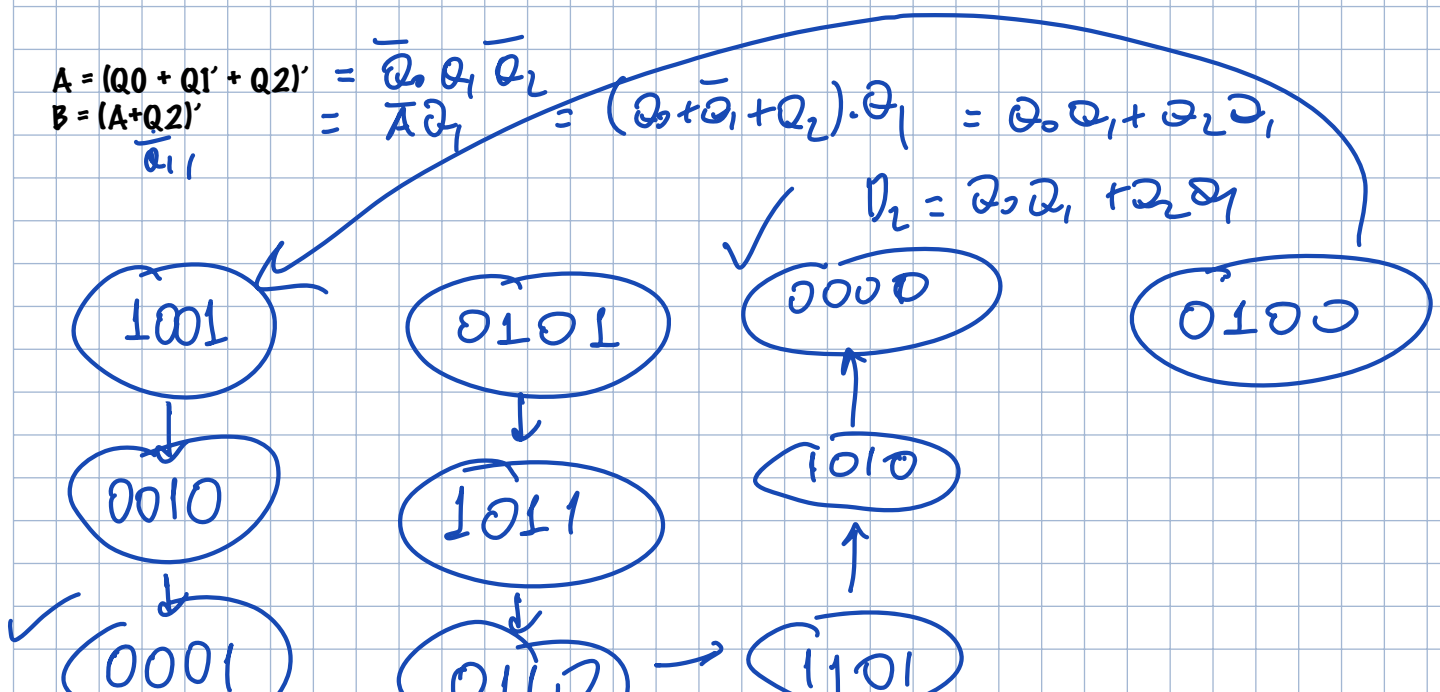


A = (Q0 + Q1' + Q2)' = $\bar{Q}_0 Q_1 \bar{Q}_2$

B = (A+Q2)' = $\bar{A} Q_2$ = $(Q_0 + \bar{Q}_1 + Q_2) \cdot Q_1$ = $Q_0 Q_1 + Q_2 Q_1$

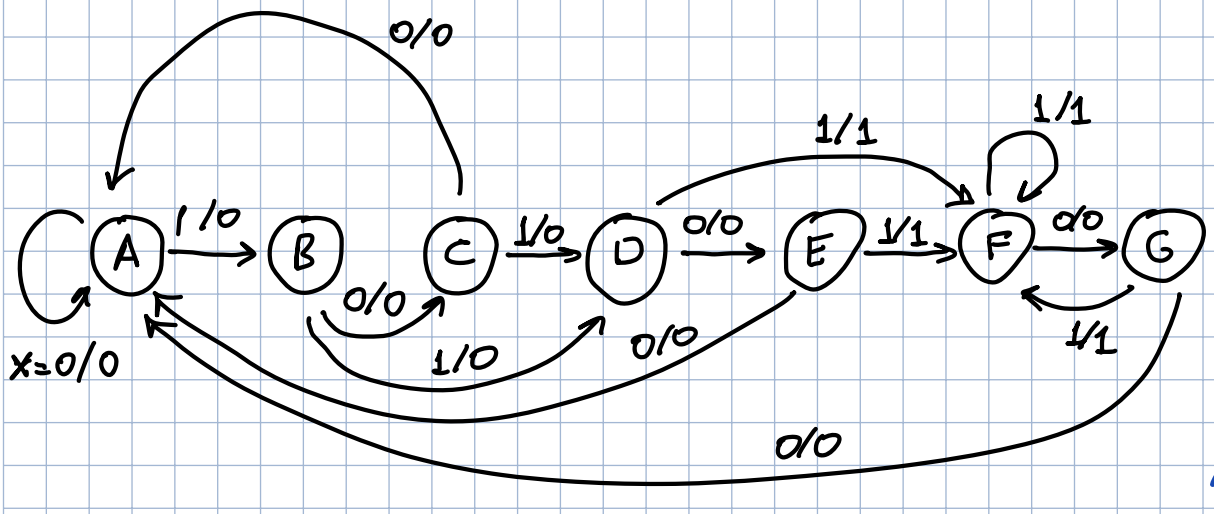
$\bar{Q}_1$ |

$D_2 = Q_0 Q_1 + Q_2 Q_1$



State Reduction

Is this state diagram minimal?



Mealy

Two or more states are equivalent if for every given input they exactly have the same output AND their next states are the same or equivalent

P.S.	N.S. x=0	N.S. x=1	out x=0	out x=1
A	A	B	0	0
B	C	D	0	0
C	A	D	0	0
D	E	F/D	0	1
E	A	F/D	0	1
F	~~G~~ E	F	0	1
~~G~~	A	F	0	1

E & G are equivalent

F & D " "

removing states

⇒ draw the state diagram.

Self-correcting ring counter

design using D-FFs.

→

⇒ Find FF input eqs.

Lots of gates.

P.S. $Q_3Q_2Q_1Q_0$	N.S. $Q_3^+Q_2^+Q_1^+Q_0^+$
0000	0001
0001	0010
0010	0100
0011	0001
0100	1000
0101	0001
0110	0001
0111	0001
1000	0001
1001	
1010	
1011	

Serial adder

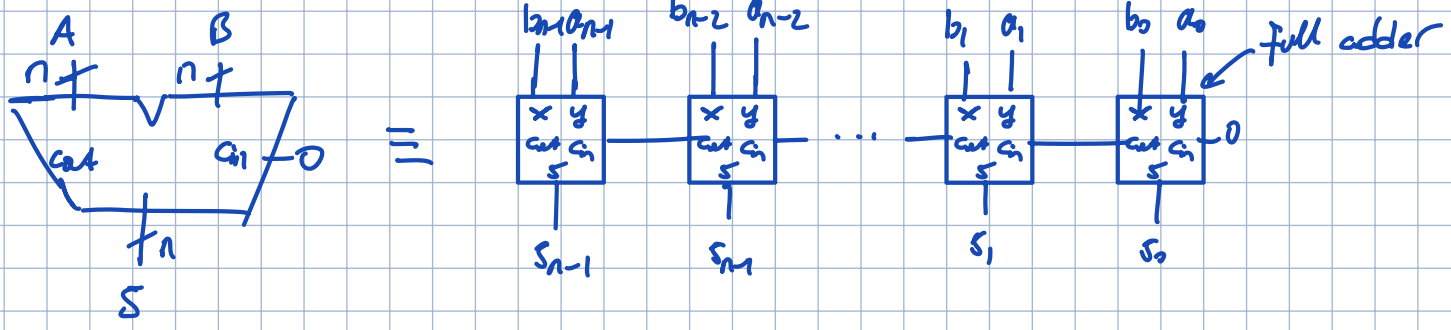
$A = a_{n-1} a_{n-2} \cdots a_0$

$B = b_{n-1} b_{n-2} \cdots b_0$

$S = s_{n-1} s_{n-2} \cdots s_0$

$S = A + B$

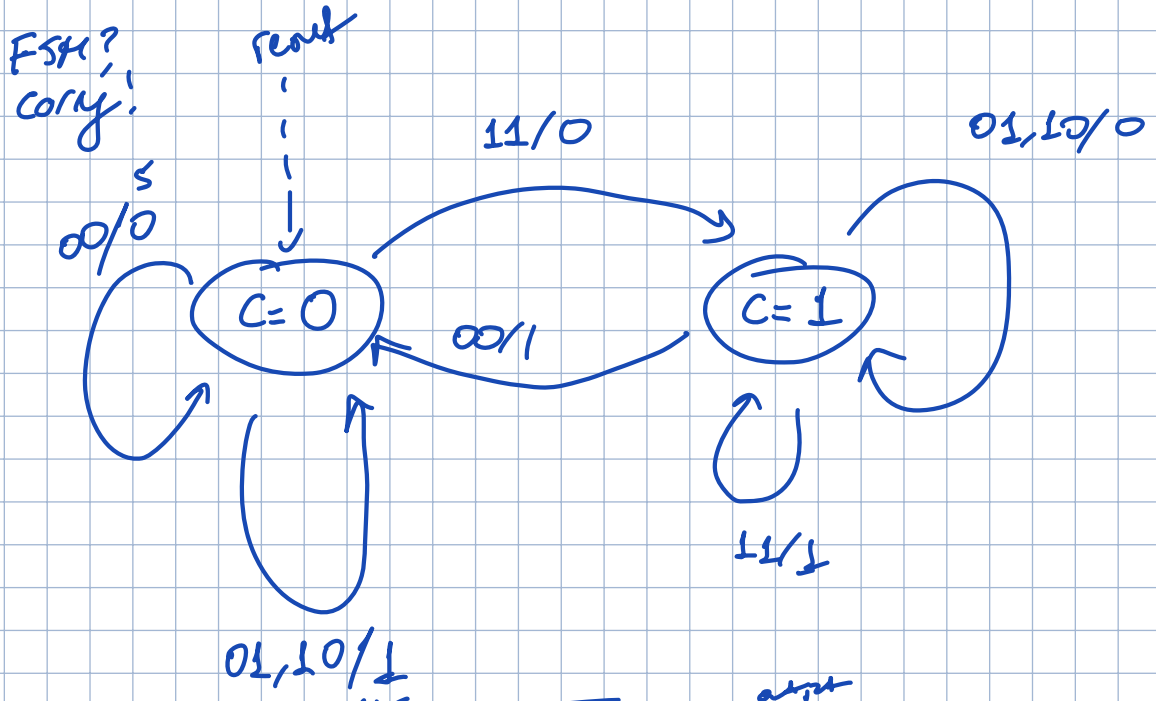
Recall n-bit combinational adder;



Serial addition $n = 4$

	1	1	1	0
x	0	1	0	1
+ y	0	0	1	1
	1	0	0	0

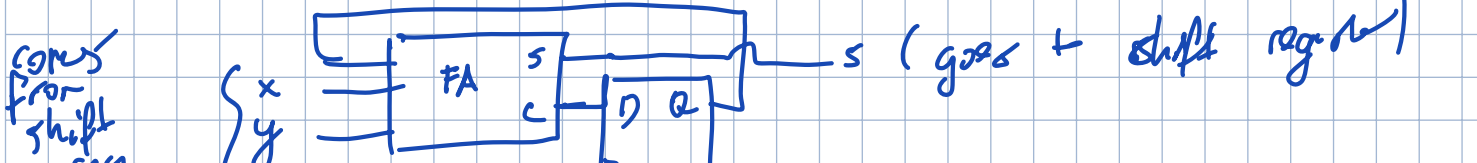
exercise!
Moore FSM



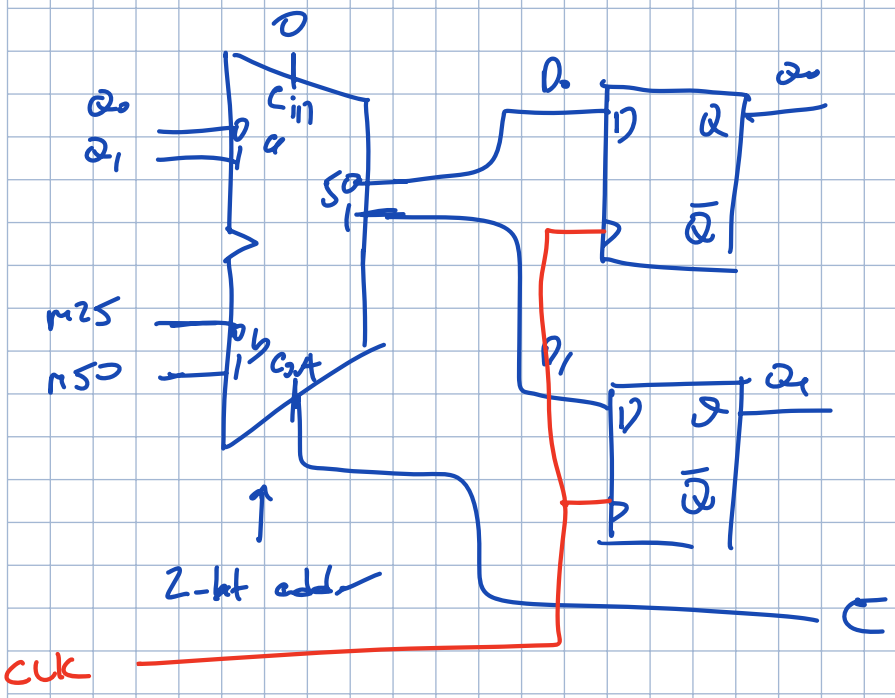
1-FF

P.S	y_i: 00	01	11	10	00	01	11	10
0	0	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1	0

$S = Q \oplus x \oplus y$



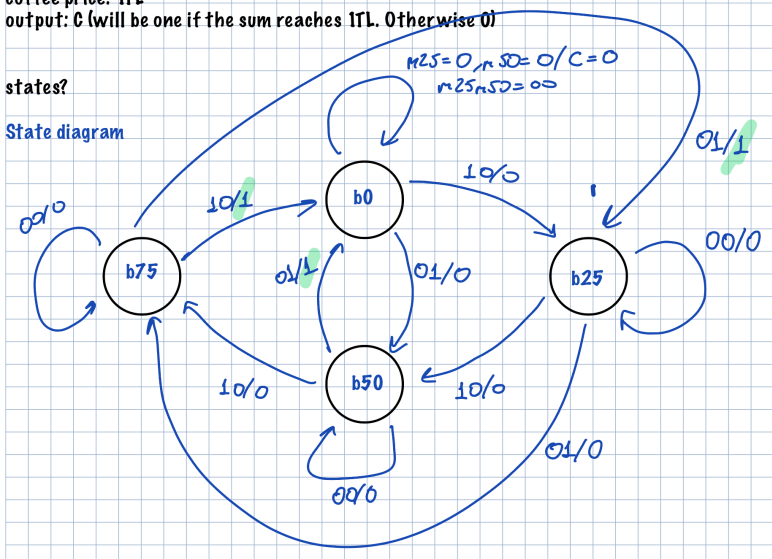
Coffee Vending Machine - Alternative Design



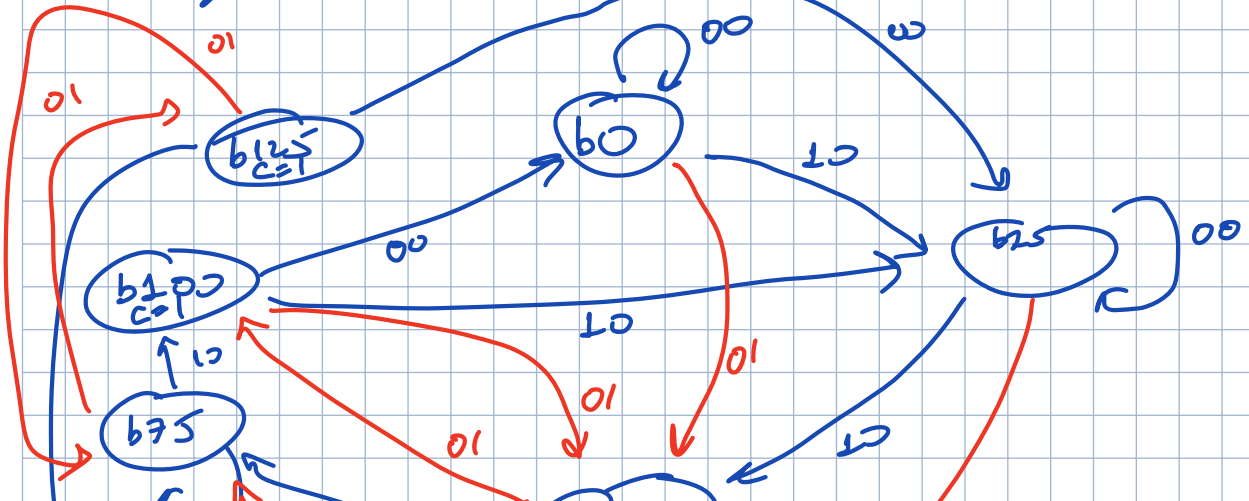
output = func(state,in)
inputs: 25kr (m25), 50kr (m50) [in a particular time slot, at most one of m25 and m50 is 1]
coffee price: 1TL
output: C (will be one if the sum reaches 1TL. Otherwise 0)

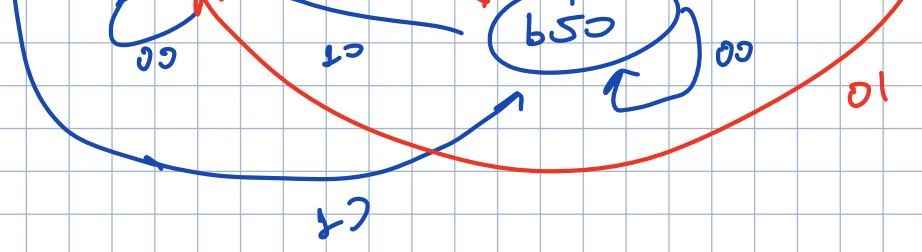
states?

State diagram



Design a Moore FSM for vending machine





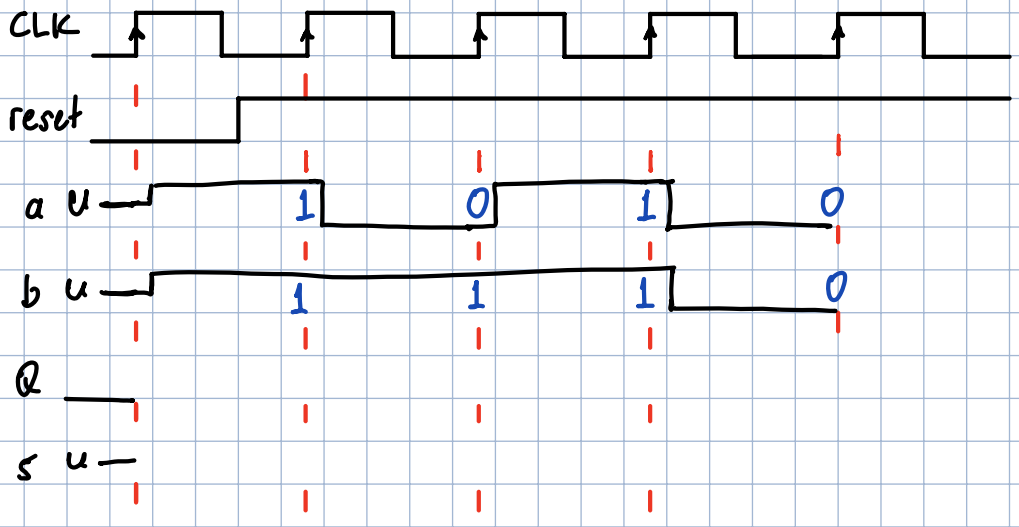
⇒ 2 more states for this example

⇒ 3 FFs

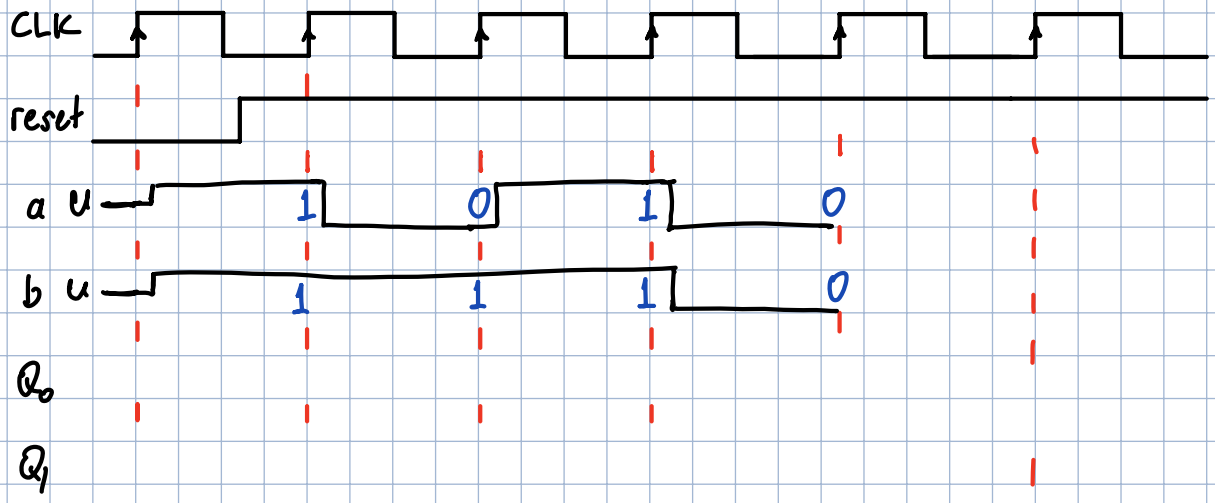
But more states does not always mean more FFs

Example: Vending machine with coffee price 1.5 TL instead of 1 TL,

Timing diagram (Mealy)

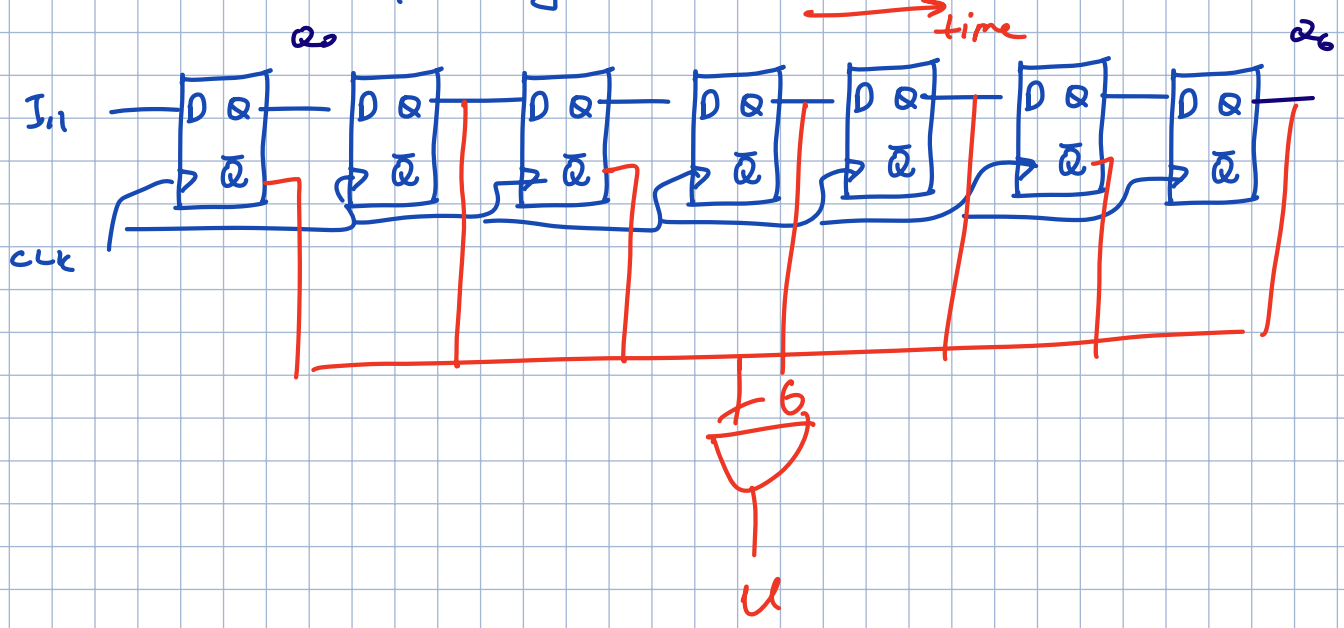


Timing diagram (Moore)



Unlock with shift register

'1011010'



Mealy version