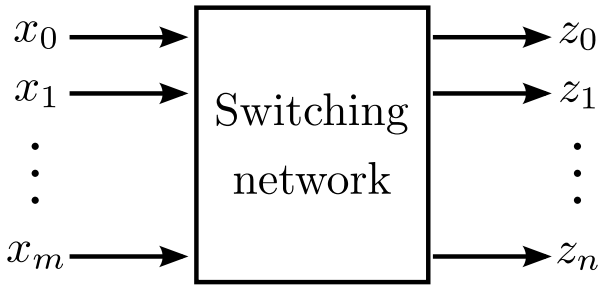
ECE 210
Introduction to Digital Logic Design

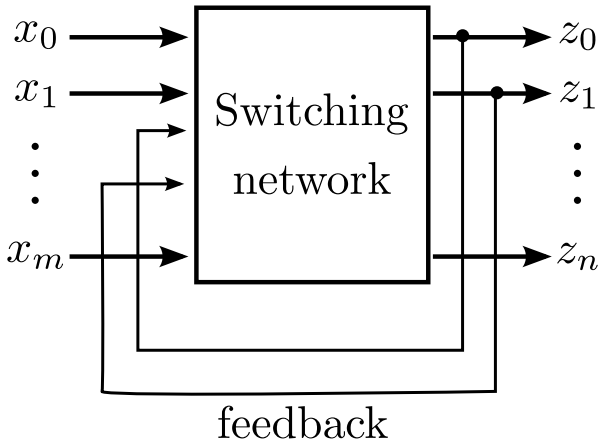
Lecture 22
Storage Elements: Latches

Categories of digital networks - Review (Lecture 1)

- Combinational: Output values depend only on *present* input values.



- Sequential: Output values depend on both *present and past* inputs.



Definitions

Latches and flip-flops are the simplest types of sequential circuits.

Latches:

$\rightarrow$ Latches are asynchronous,

$\rightarrow$ The output changes when the input changes.

Flip-flops:

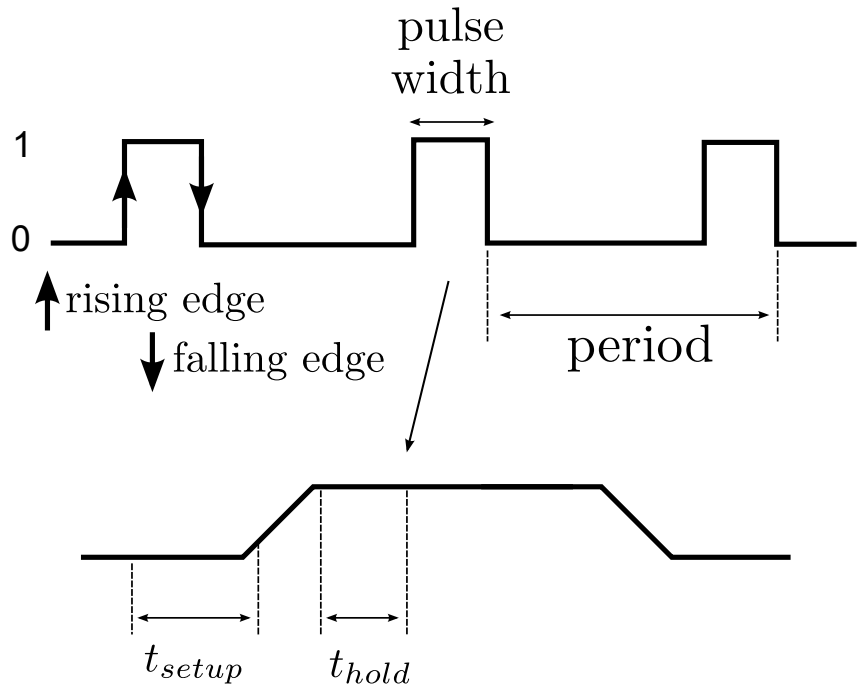
$\rightarrow$ A flip-flop is a synchronous version of the latch,

$\rightarrow$ The outputs of all the sequential circuits change **simultaneously** to the rhythm of a global **clock signal**.

Timing diagrams

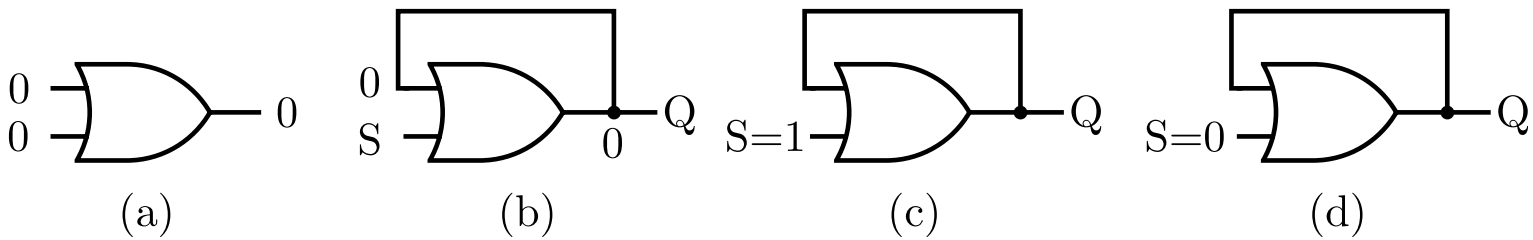
In sequential switching networks, the output depends not only on the present state, but also on the past *sequence* of inputs.

Timing diagram - assume ideal signals



Basic memory latch

Latch is a circuit's ability to **remain** at a particular logic level after having been driven to that state by an external signal.



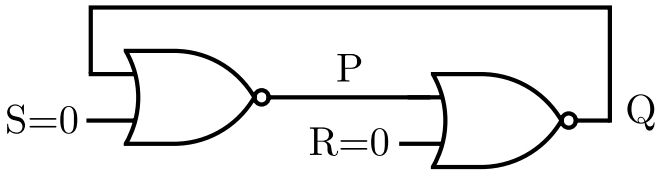
(a) Initial conditions

(b) Feedback: $S = 0$ feeds back to keep $Q =$

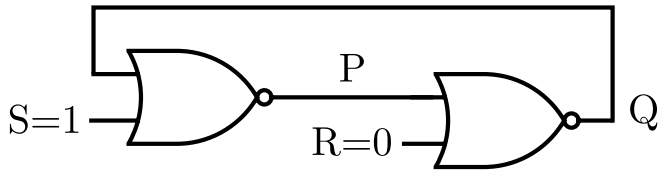
(c) $S = 1$, feeds back $\therefore Q =$

(d) S returns to 0, then

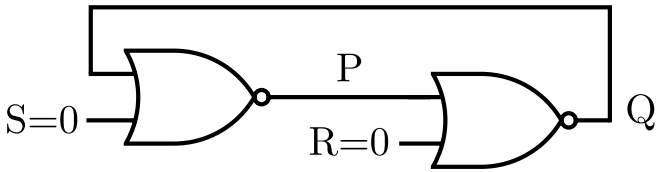
NOR gate latch - Set/Reset



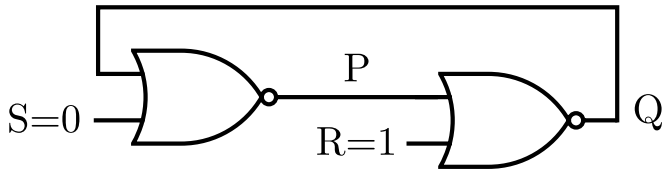
(a)



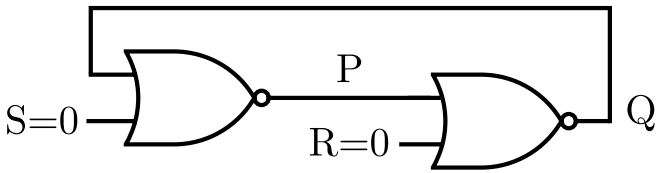
(b)



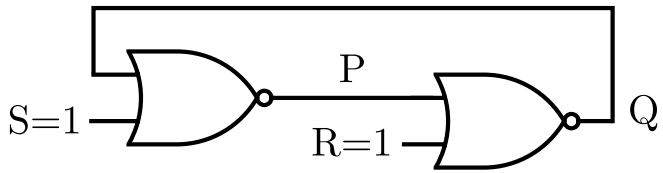
(c)



(d)



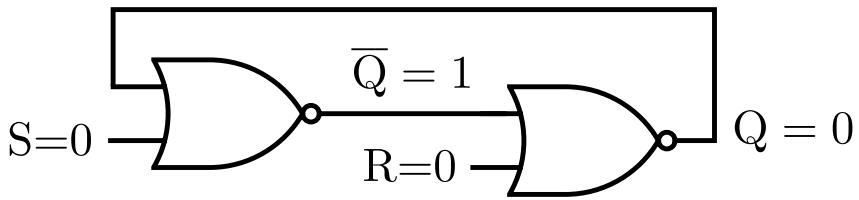
(e)

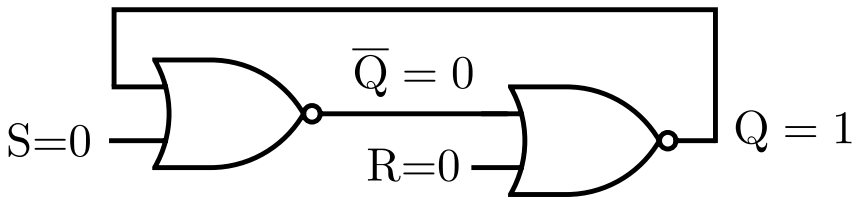


forbidden

NOR gate latch - Set/Reset

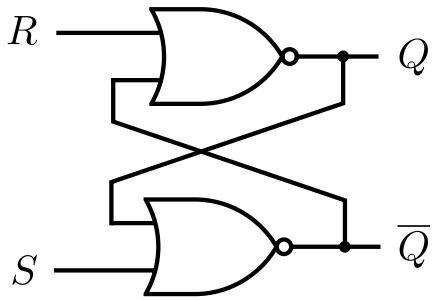
The circuit has 2 stable states for the same inputs, and they depend on the past sequence of inputs.

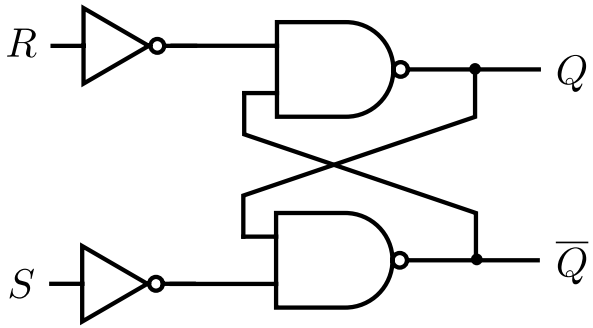


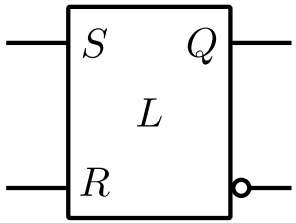


Notice that the output of the first gate is $\overline{Q}$.

Set-Reset Latch







Function table:

		Next states	
R	S	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1	$\times$	$\times$

Properties:

1. 2 inputs, 2 outputs
2. $S = 1 \rightarrow$
3. $R = 1 \rightarrow$
4. $S = R = 1 \rightarrow$ indeterminate state

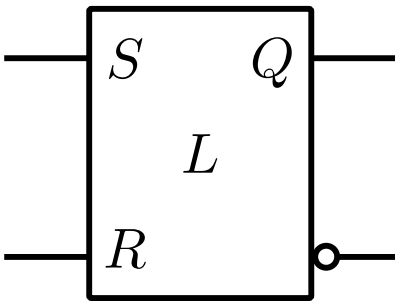
S-R Latch Operation Summary

$\rightarrow$ Both inputs 0 : no change

$\rightarrow$ Set input momentarily 1: Q goes to 1

$\rightarrow$ Reset input momentarily 1: Q goes to 0

$\rightarrow$ Reset and reset made 1: Indeterminate state



S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	$\times$
1	1	1	$\times$

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\times$

SR Latch characteristic equation

K-map for Q^+

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	×
1	1	1	×

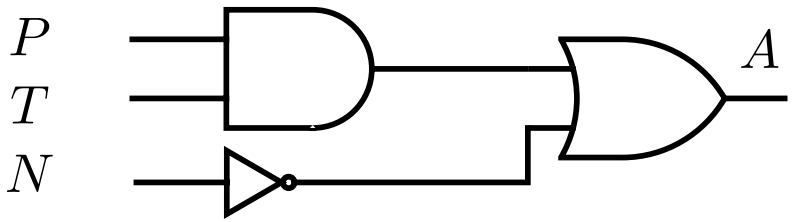
RQ \ S	0	1
00	0	1
01	1	1
11	0	×
10	0	×

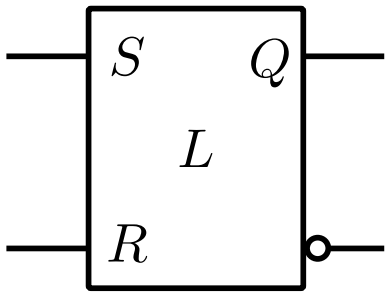
$$Q^+ = S + \overline{R}Q$$

$$Q(t+\epsilon) = S(t) + \overline{R(t)}Q(t)$$

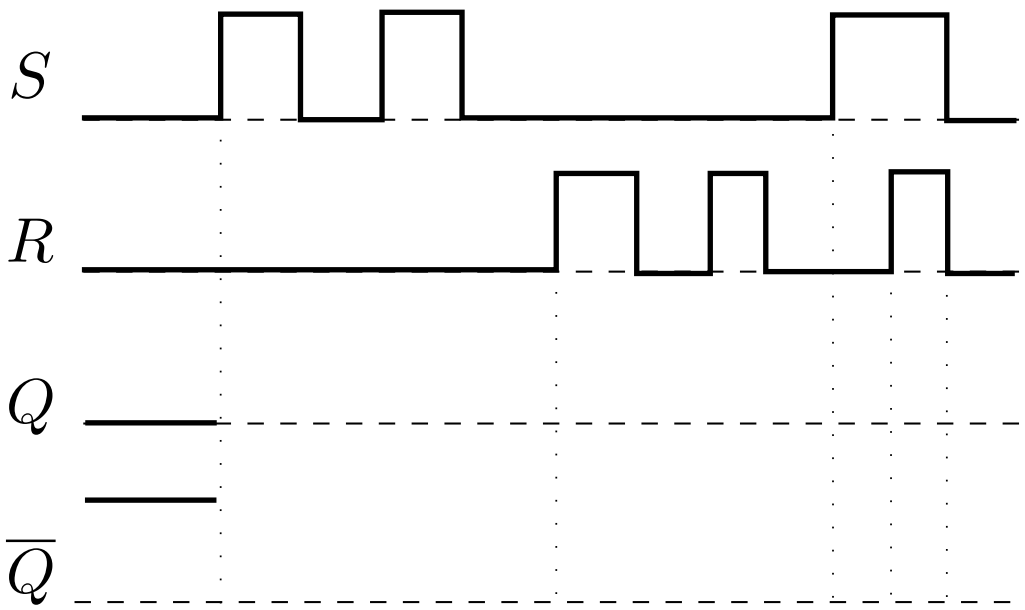
S and R not allowed to be 1 at the same time.

SR latch example

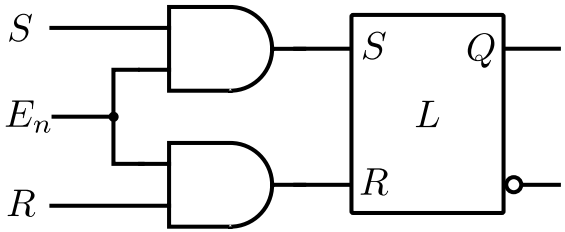


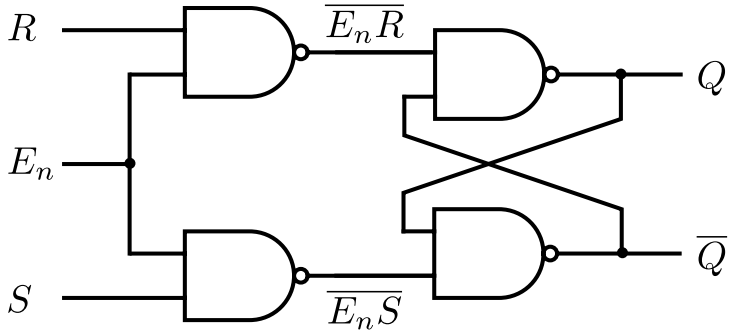


Timing diagram for S-R latch



SR Latch with control input



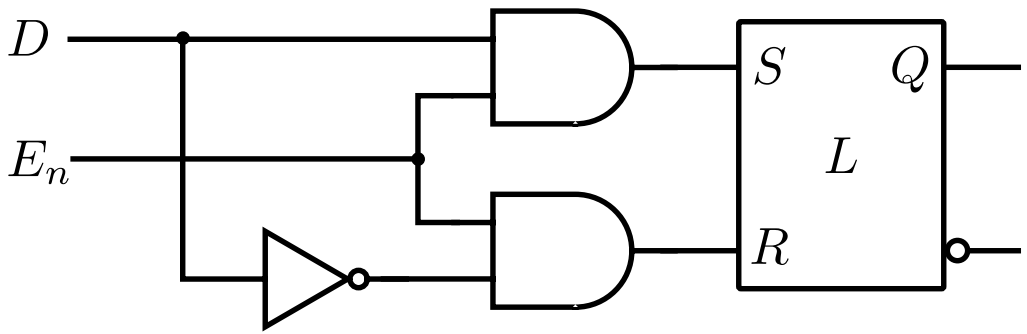


E_n	S	R	Q^+
0	×	×	
1	0	0	
1	0	1	
1	1	0	
1	1	1	forbidden

D latch - Transparent latch

Ensures that S and R are never equal 1 at the same time.

The D latch can hold ***D**ata* in its internal storage.



E_n	D	Next state
0	$\times$	$Q^+ = Q$ (no change)
1	0	$Q^+ = 0 = D$
1	1	$Q^+ = 1 = D$

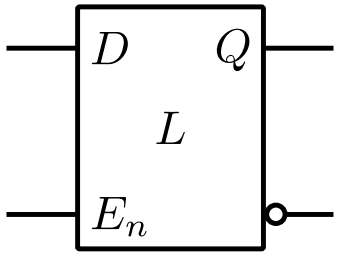
D latch - Transparent latch

From the truth table the characteristic equation for the D-latch is :

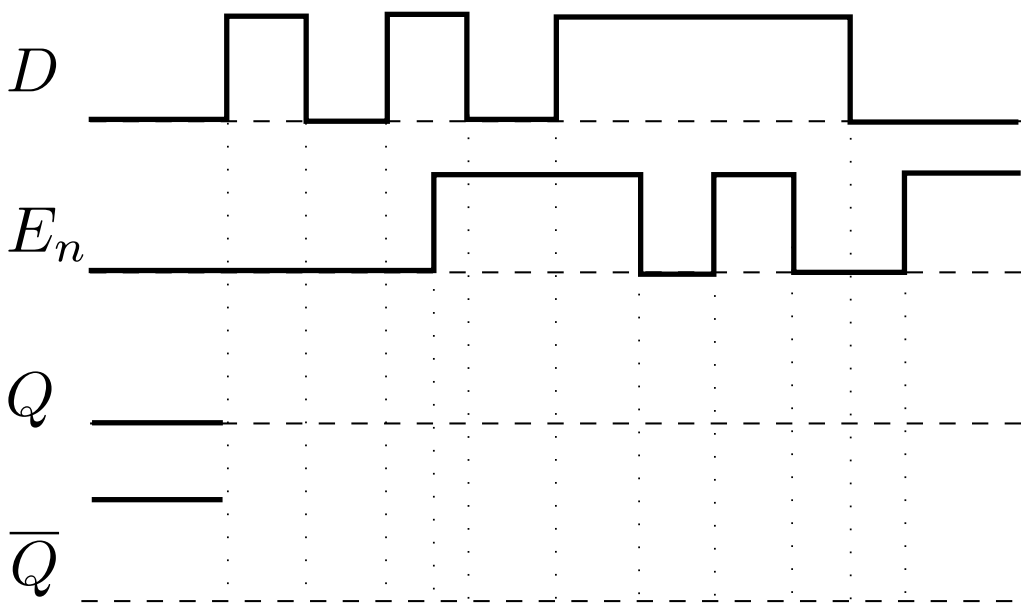
$$Q^+ = \overline{E_n} Q + E_n D$$

E_n	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

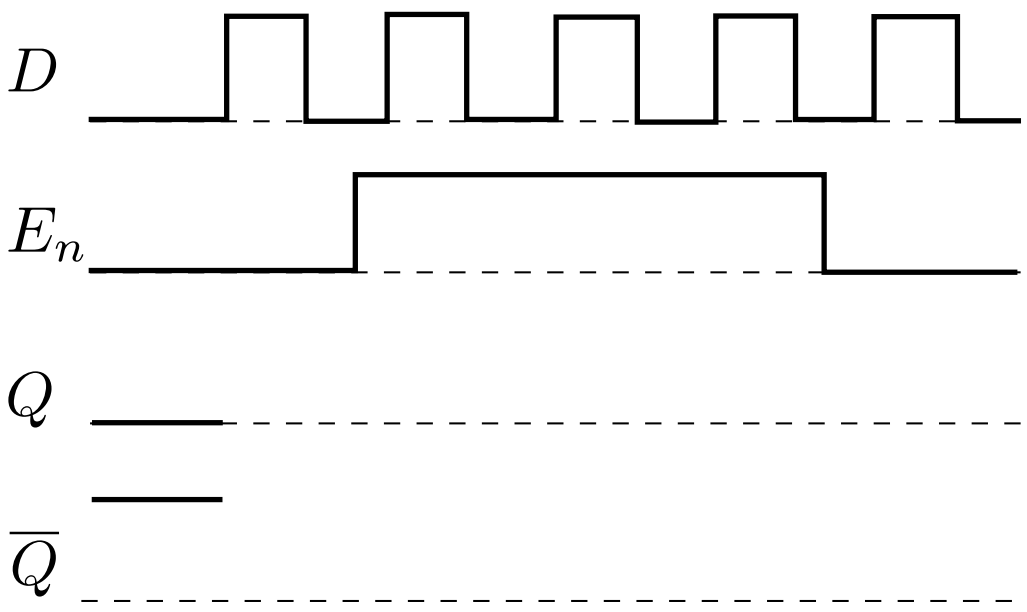
E_nD \ Q	0	1
00	0	1
01	0	1
11	1	1
10	0	0



Timing diagram for D latch



Timing diagram for D latch



Next class...

- Flip-flops
- Please read Lecture 23

ECE 210
Introduction to Digital Logic Design

Lecture 23
Storage Elements: Flip-Flops

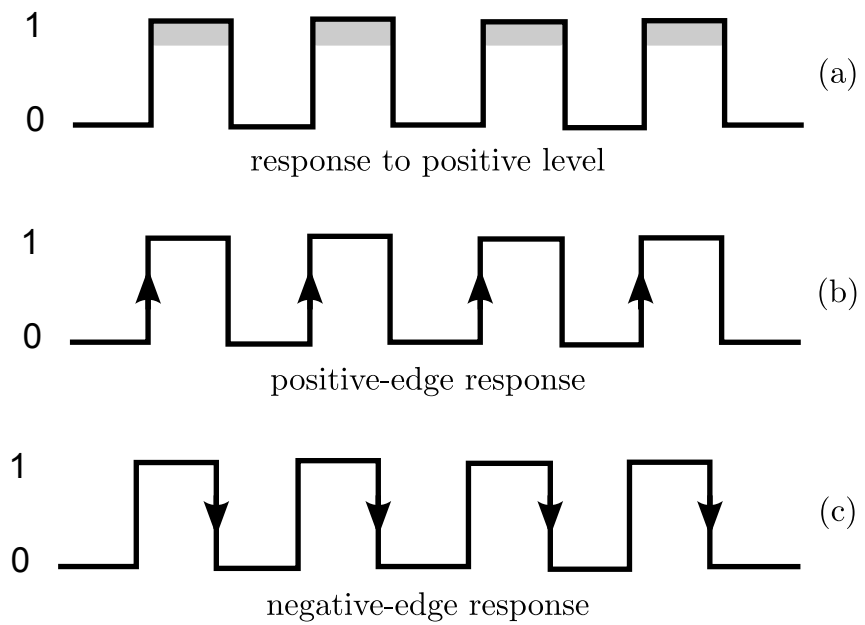
Flip-flops *vs* Latches

→ A latch responds to a change in the *level* of a signal (a),

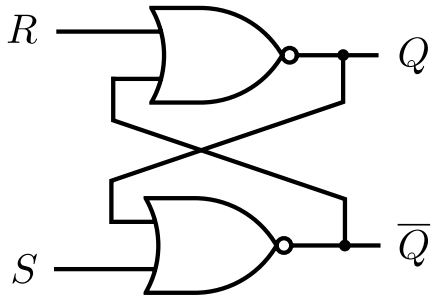
→ A flip-flop is a synchronous version of the latch,

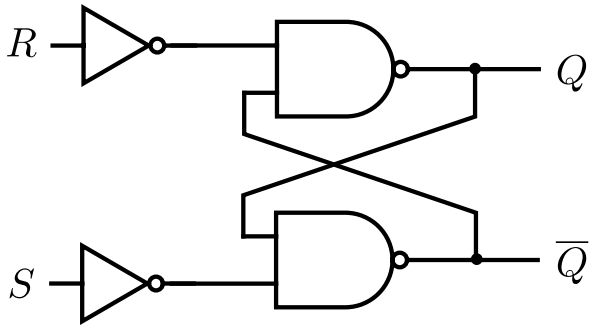
→ A flip-flop triggers only during a signal *transition* (b-c).

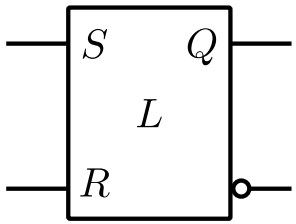
→ The outputs change to the rhythm of a global clock signal.



Set-Reset Latch (review)







Function table:

R	S	Q^+	$\overline{Q^+}$
0	0	Q	$\overline{Q^+}$
0	1	1	0
1	0	0	1
1	1	$\times$	$\times$

Properties:

1. $S = 1 \rightarrow$
2. $R = 1 \rightarrow$
3. $S = R = 1 \rightarrow$ indeterminate state

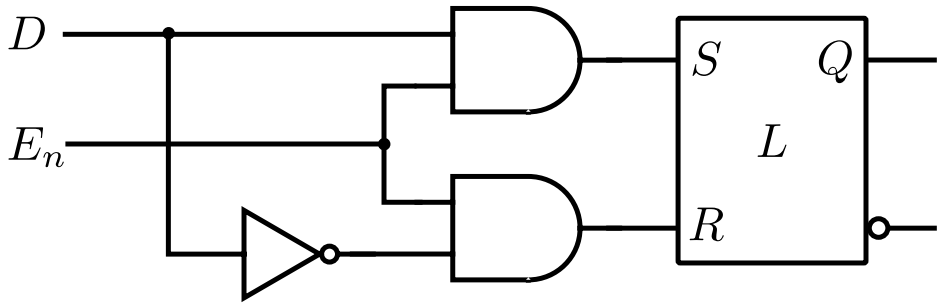
$$Q^+ = S + \overline{R}Q$$

$R = S = 1 \rightarrow$ forbidden input

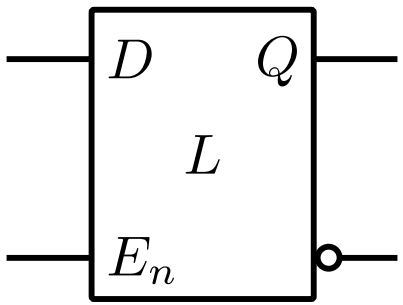
D latch - Transparent latch (review)

Ensures that S and R are never equal 1 at the same time.

The D latch can hold ***D**ata* in its internal storage.



E_n	D	Next state
0	$\times$	$Q^+ = Q$ (no change)
1	0	$Q^+ = 0 = D$
1	1	$Q^+ = 1 = D$



Characteristic equation:

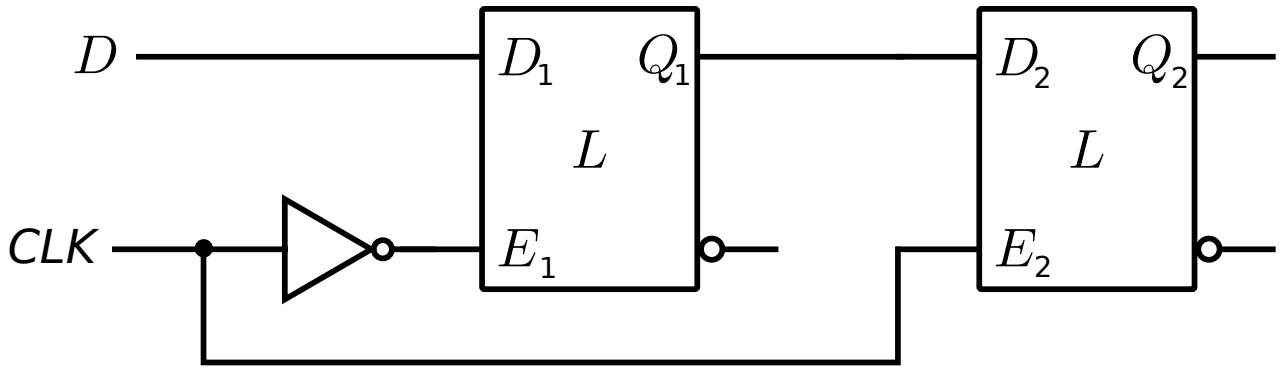
$$Q^+ = \overline{E_n} Q + E_n D$$

Edge-Triggered D Flip-flop

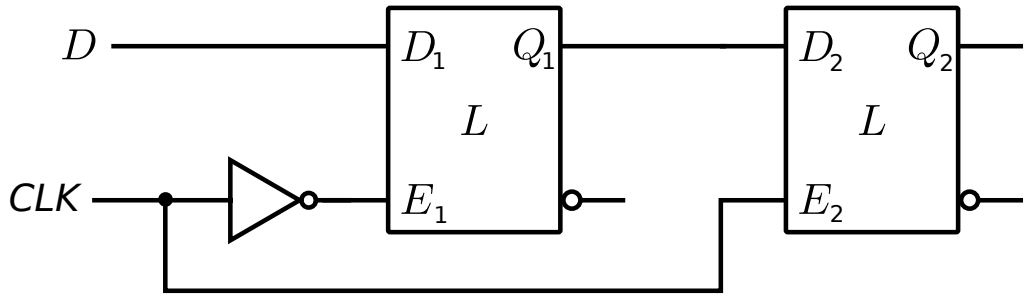
Unlike the D-latch, the D-flip flop output changes in response to a 0 to 1 **transition** on the **clock signal**.

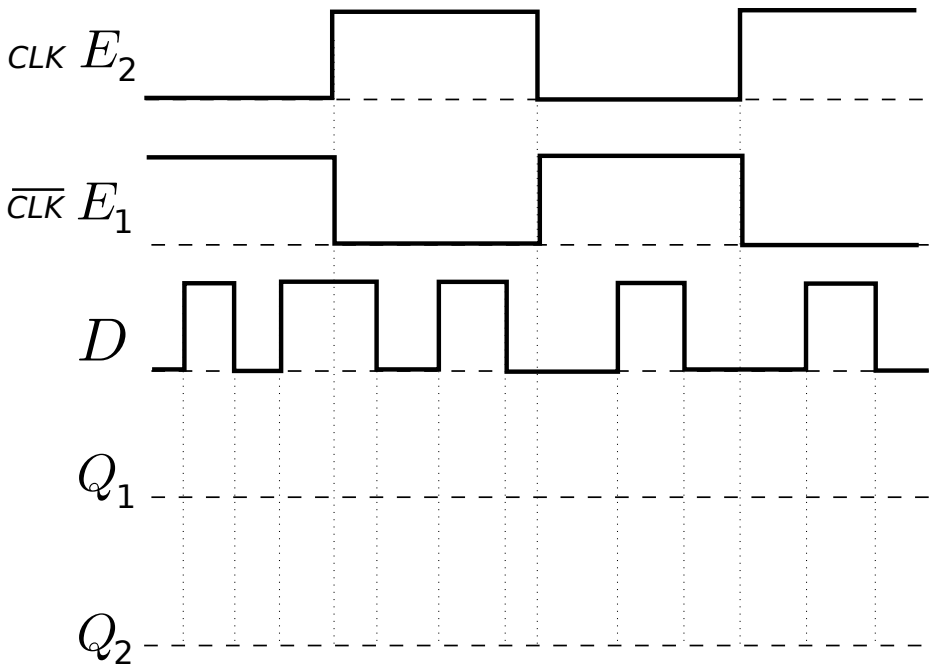
We say that the flip-flop is triggered on the *rising edge* (or positive edge) of the clock signal.

D flip-flops can be constructed from two D-latches

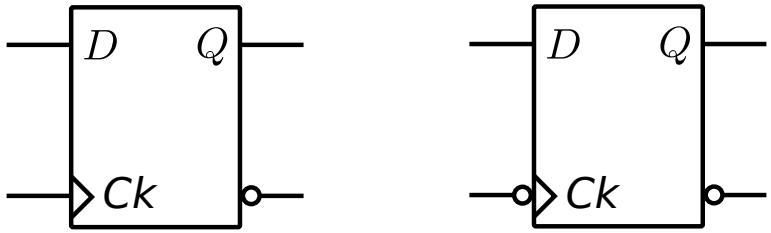


Edge-Triggered D Flip-flop





Edge-Triggered D Flip-flop

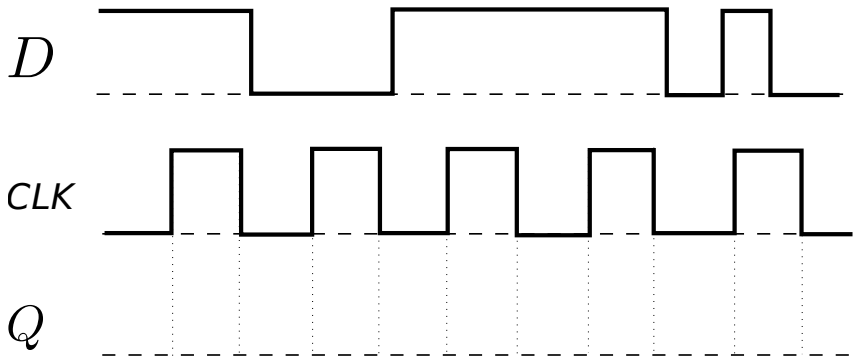


The state of a D flip-flop **after** the active clock edge is equal to the input D **before** the active edge.

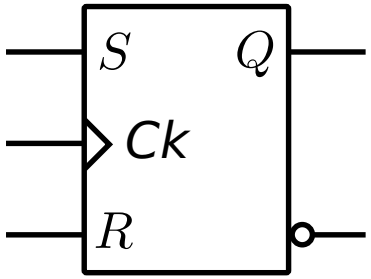
The output changes are delayed until after the active edge of the clock pulse.

The characteristic equation is $Q^+ = D$.

Falling edge trigger:



SR Flip-flop

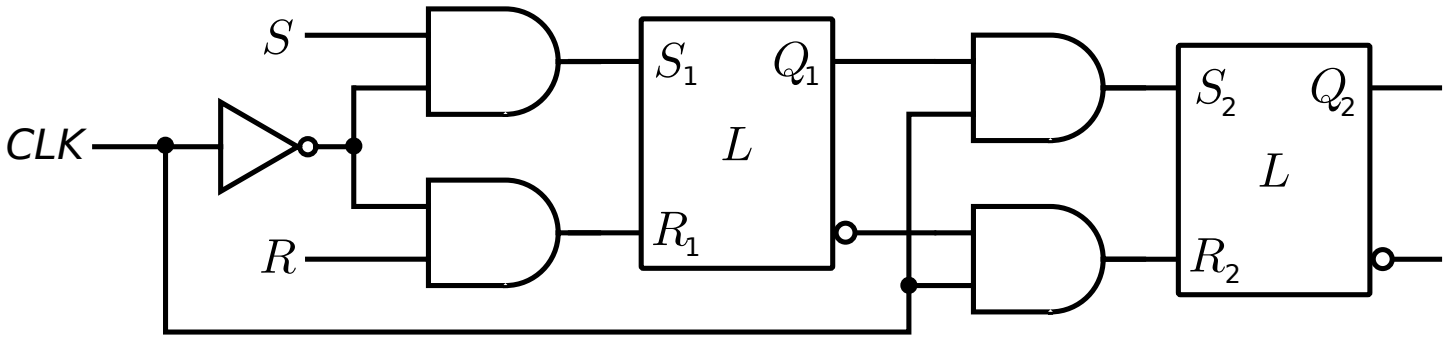


An SR flip-flop is similar to an SR latch:

$\rightarrow S = 1$ sets the output to 1

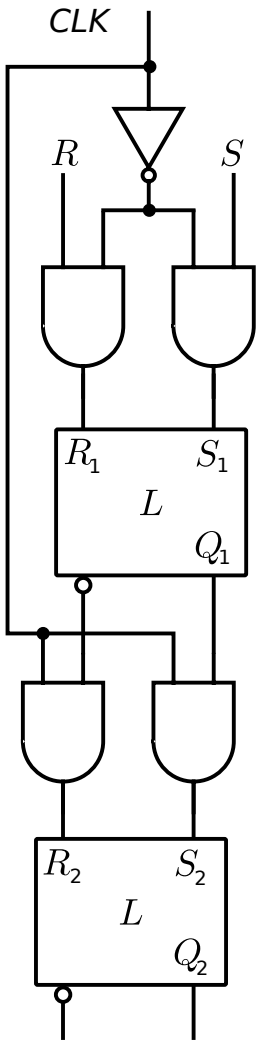
$\rightarrow R = 1$ resets the output to 0

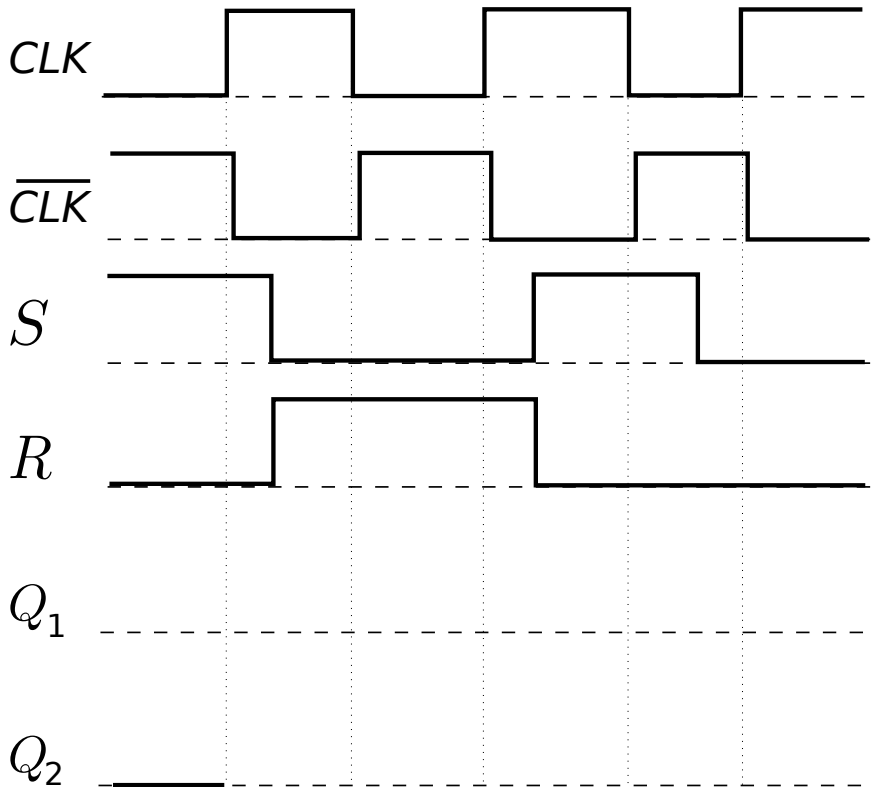
But: The flip-flop has a clock-input: The output changes only after an active clock edge.



$S = R = 1 \rightarrow$ not allowed.

SR Flip-flop





SR Flip-flop

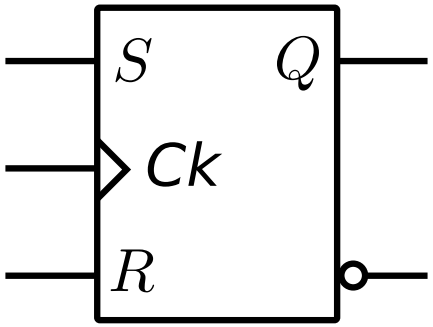
Operation summary:

$\rightarrow S = R = 0$ no state change

$\rightarrow S = 1,\ R = 0$ sets Q to 1 after active Ck edge

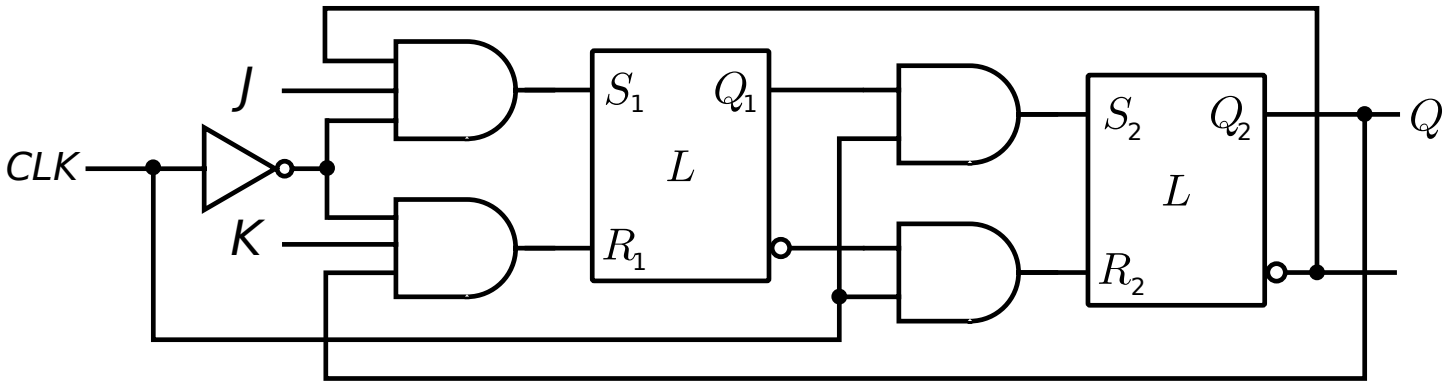
$\rightarrow S = 0,\ R = 1$ resets the output to 0 after active Ck edge

$\rightarrow S = R = 1$ not allowed



How do we ensure that S and R are never 1 at the same time ?

JK Flip-flop



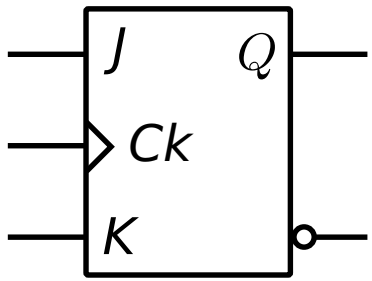
Same circuit as for the SR flip-flop, except that

$\rightarrow$ S and R have been replaced with J and K

$\rightarrow$ Q and $\overline{Q}$ are feeding back into the input

$\rightarrow$ Only one of S and R inputs can be 1 at the same time

$\rightarrow$ If $J = K = 1$, $Q^+ =$

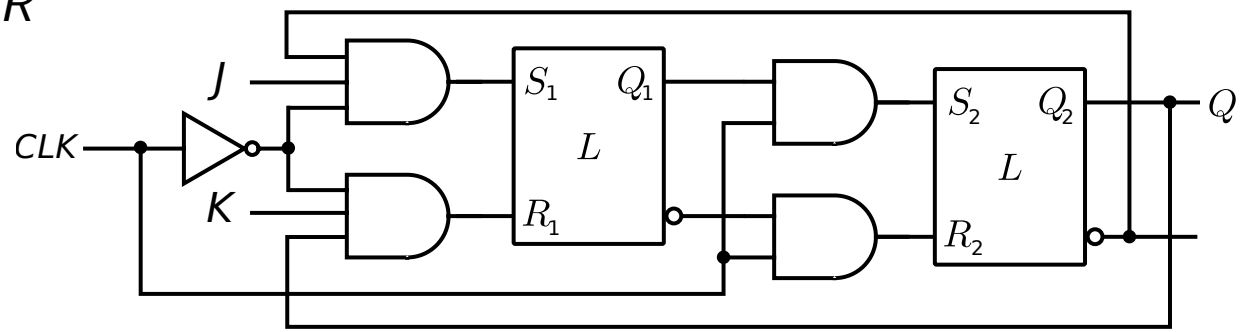


JK Flip-flop

The JK flip-flop is an extended version of SR flip-flop:

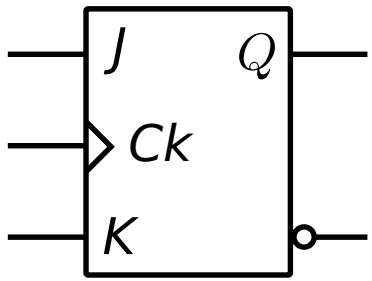
$\rightarrow$ J corresponds to S

$\rightarrow$ K corresponds to R

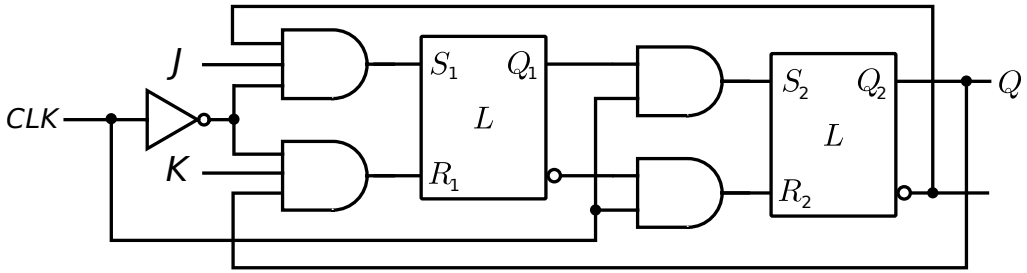


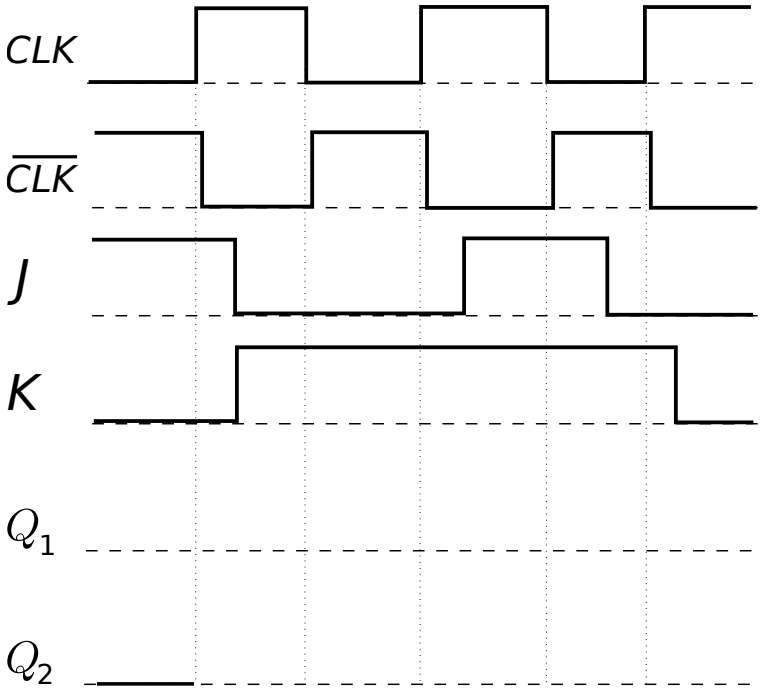
J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	**1**	0	1
1	**1**	1	0

$$Q^+ = J\overline{Q} + \overline{K}Q$$



JK Flip-flop





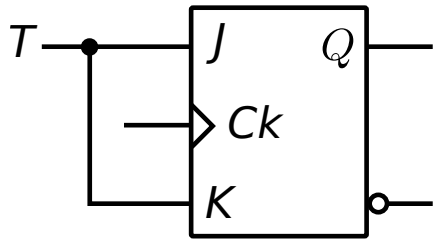
T Flip-flop

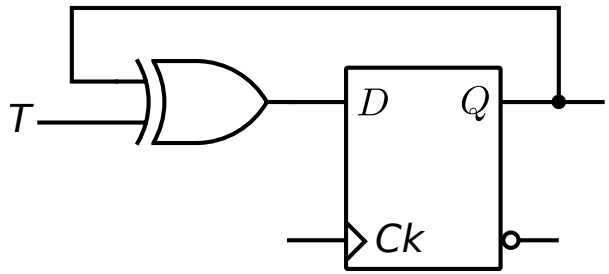
The T flip-flop or toggle flip-flop is often used in building counters.

When $T = 1$ the flip-flop changes state after the active edge of the clock.

When $T = 0$ no change occurs.

J	K	Q	Q^+	
0	**0**	0	0	Q
0	**0**	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	**1**	0	1	$\overline{Q}$
1	**1**	1	0	



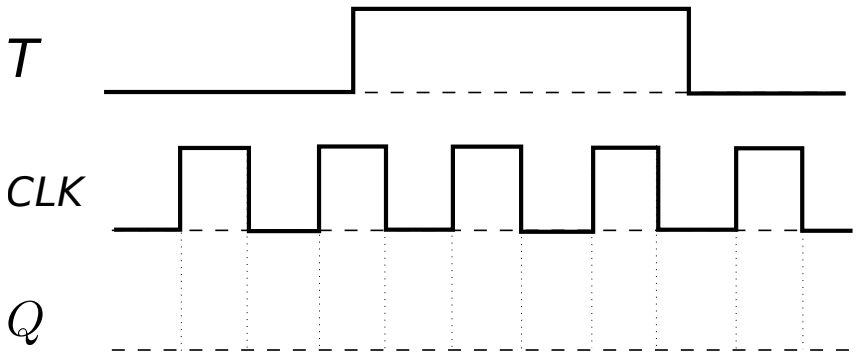


$$Q^+ = T\overline{Q} + \overline{T}Q = T \oplus Q$$

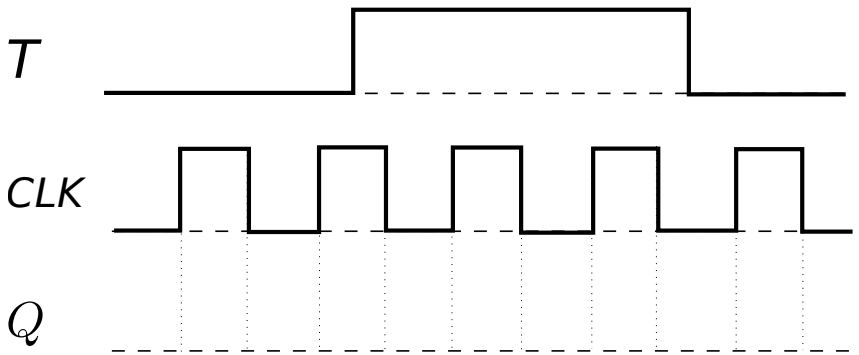
T flip-flop

Timing diagram for T flip-flop.

Rising edge trigger

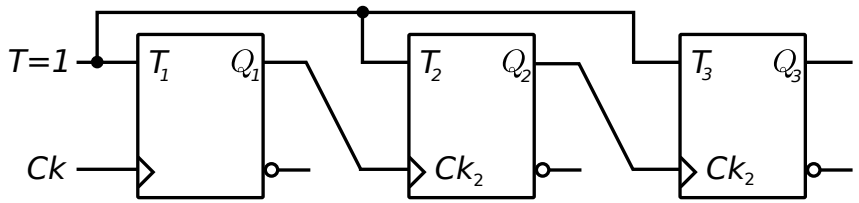


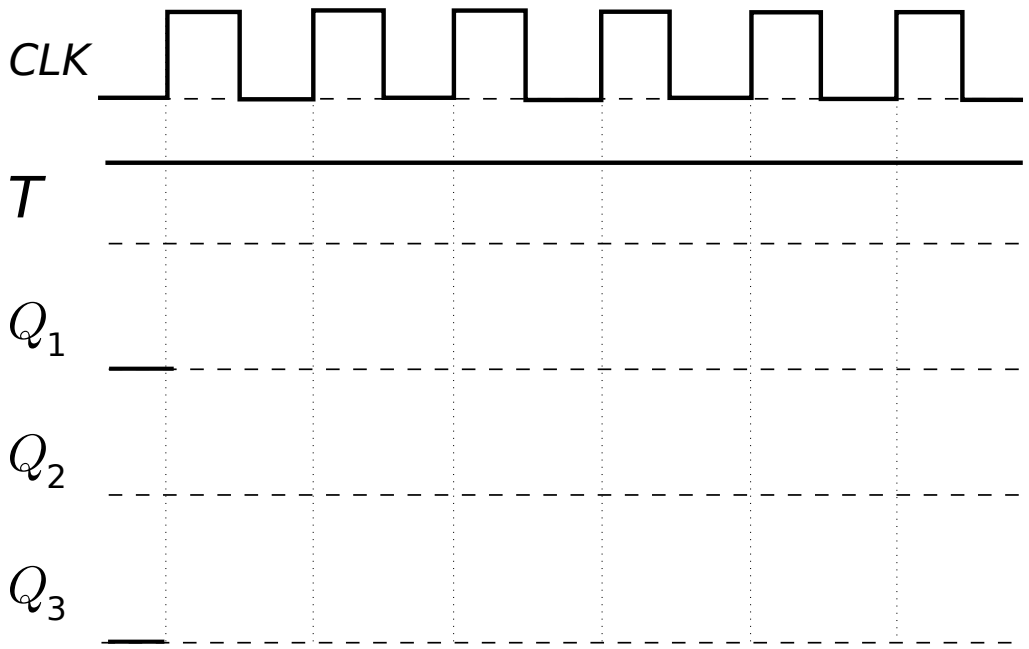
Falling edge trigger



T flip-flop

T flip-flops connected in a cascade mode.





Summary

Device	Characteristic equation
SR latch and flip-flop	$Q^+ = S + \overline{R}Q$
D-latch	$Q^+ = E_n D + \overline{E_n}Q$
D flip-flop	$Q^+ = D$
JK flip-flop	$J\overline{Q} + \overline{K}Q$
T flip-flop	$Q^+ = T \oplus Q = T\overline{Q} + \overline{T}Q$

Next class...

- Binary counters
- Please read Lecture 24

ECE 210
Introduction to Digital Logic Design

Lecture 24
Design of Binary Counters

Binary counter

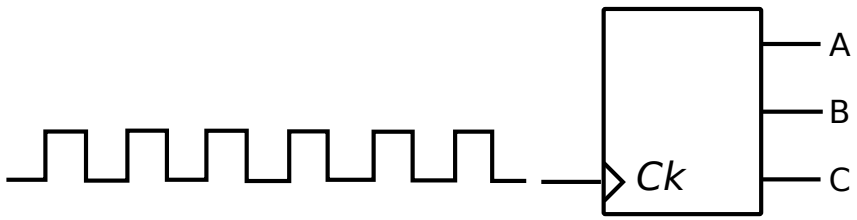
Counts pulses and displays count in binary form

$\rightarrow$ Implemented with flip-flops (FF),

$\rightarrow$ Pulses (input P) go to clock input of the FF,

$\rightarrow$ The counters discusses here are synchronous (all FF change state simultaneously),

$\rightarrow$ Ripple counter: A FF triggers the next FF (not discussed).



$$000 \rightarrow 001 \rightarrow 010 \rightarrow 011 \rightarrow 100 \rightarrow \ldots \rightarrow 111 \rightarrow 000$$

Binary counter

Next state table

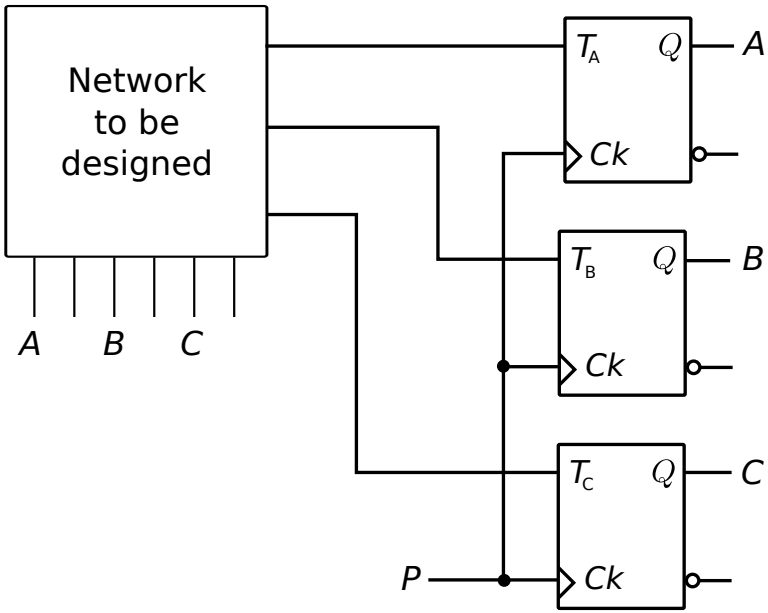
Current state			Next state		
A	B	C	A^+	B^+	C^+
0	0	0	0	0	1
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Synchronous counter

Using T flop-flops

$\rightarrow$ Three T flip-flops are used,

$\rightarrow$ Objective: Determine the inputs to each FF (i.e., T_A, T_B, and T_C),



T-FF transition tables

Q	T	Q^+
0	0	
0	1	
1	0	
1	1	

Q	Q^+	T
0	0	
0	1	
1	0	
1	1	

Binary counter

Next state table using T flip-flops

Current state			Next state			FF inputs		
A	B	C	A^+	B^+	C^+	T_A	T_B	T_C
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0	0	1	1			
0	1	1	1	0	0			
1	0	0	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			

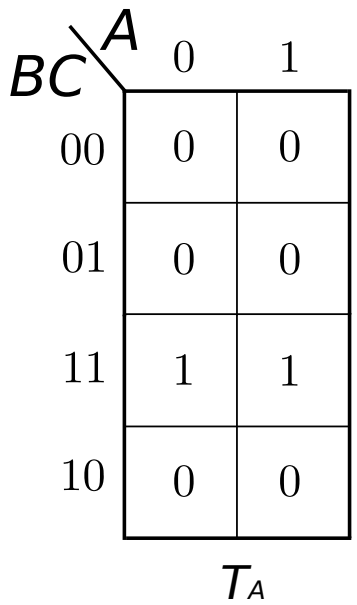
$\rightarrow$ $T = 0$ no change

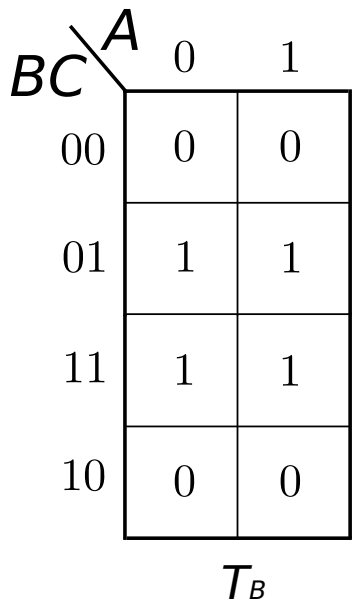
$\rightarrow$ $T = 1$ toggles output

Binary counter

K-maps for T_A and T_B

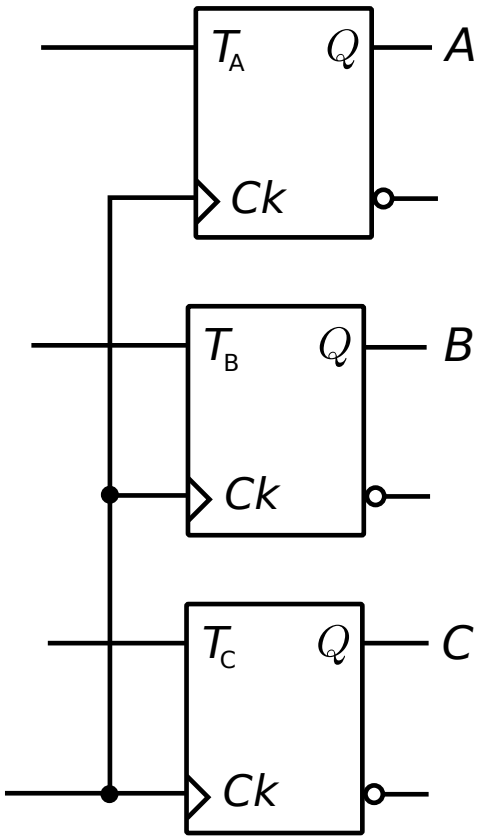
Current state			FF inputs	
A	B	C	T_A	T_B
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1



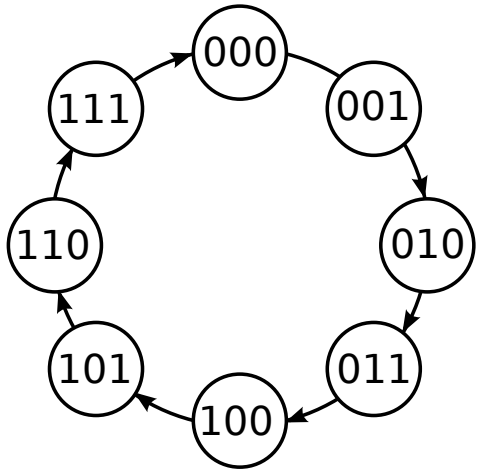


Binary counter

Logic diagram



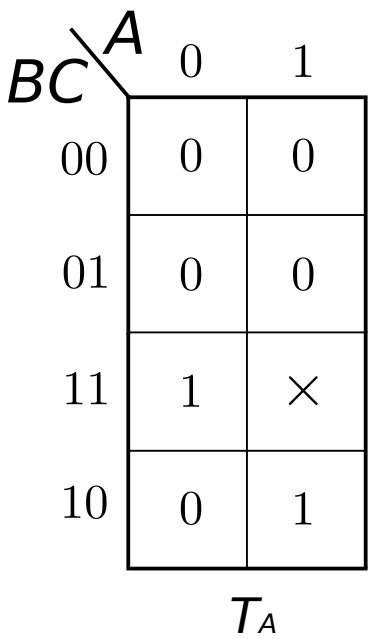
State diagram



Binary counter - Incomplete sequence

Assume that the state $ABC = 111$ is missing.

Current state			Next state			FF inputs		
A	B	C	A^+	B^+	C^+	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0						
1	1	1						

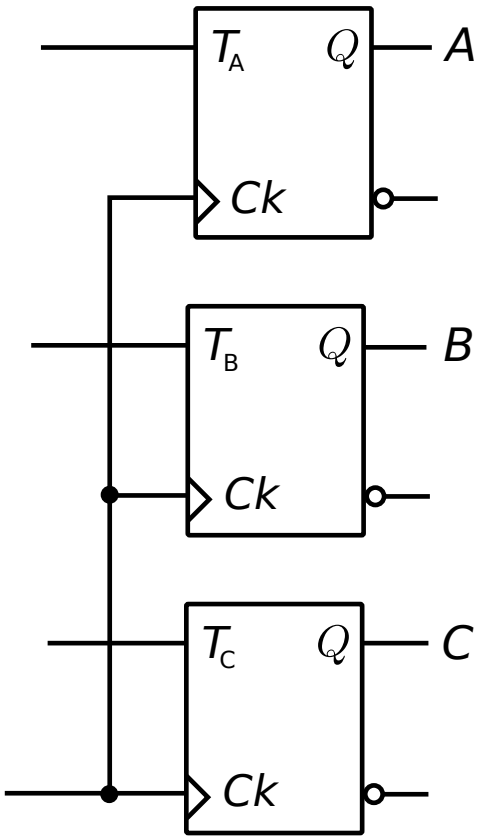


Binary counter - Incomplete sequence

Solve for T_B and T_C.

BC \ A	0	1
00		
01		
11		
10		

BC \ A	0	1
00		
01		
11		
10		

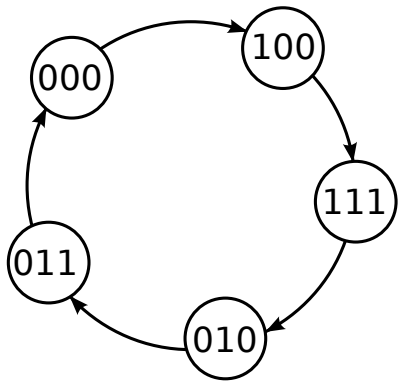


Counters for arbitrary sequences

The sequence of states of a counter is not always in straight binary order.

Design a counter for the sequence given in the state graph using T flip-flops

Current state			Next state		
A	B	C	A^+	B^+	C^+
0	0	0	1	0	0
0	0	1			
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1			
1	1	0			
1	1	1	0	1	0



Counters for arbitrary sequences

→ Start from the next state K-maps. Derive the T inputs from them.

BC \ A	0	1
00	1	1
01	×	×
11	0	0
10	0	×

$A+$

BC \ A	0	1
00	0	1
01	×	×
11	0	1
10	1	×

$B+$

BC \ A	0	1
00	0	1
01	×	×
11	0	0
10	1	×

$C+$

BC \ A	0	1
00		
01		
11		
10		

T_A

BC \ A	0	1
00	0	1
01	×	×
11	1	0
10	0	×

T_B

BC \ A	0	1
00	0	1
01	×	×
11	1	1
10	1	×

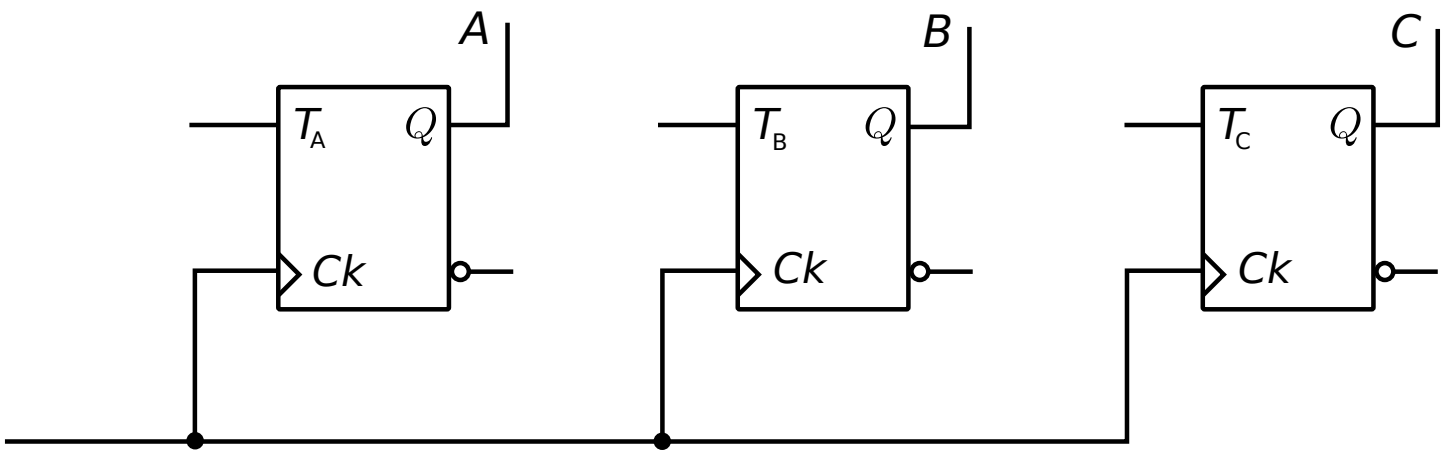
T_C

$T_A =$

$T_B =$

$T_C =$

Counters for arbitrary sequences



Counters design using T flip-flops

$\rightarrow$ Form a state table which gives the next FF states;

$\rightarrow$ Plot the next-state maps from the table;

$\rightarrow$ Plot a T input map for each FF. T is 1 whenever $Q^+ \neq Q$ and 0 otherwise;

$\rightarrow$ Find the T input equations from the maps and realize the circuit.

Note that although the original state table may not be completely specified, the actual design must specify all states using don't care conditions.

When the FF's are turned on, their initial states may be unpredictable.

All of the don't care states should be checked to make sure that they eventually lead into the main counting sequence.

Counter design using D flip-flops

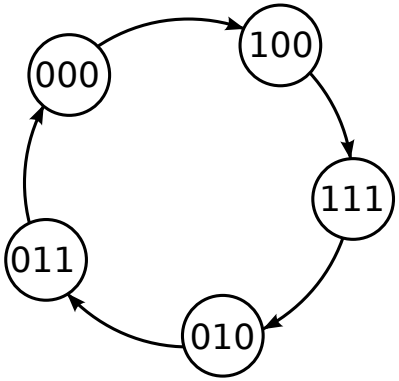
→ For a D flip-flop, $Q^+ = D$.

→ The D input map is identical to the next-state map.

→ The equation for D can be read from the Q^+ map.

Q	$Q+$	D
0	0	
0	1	
1	0	
1	1	

Counter design using D flip-flops



BC \ A	0	1
00	1	1
01	×	×
11	0	0
10	0	×

$A+$

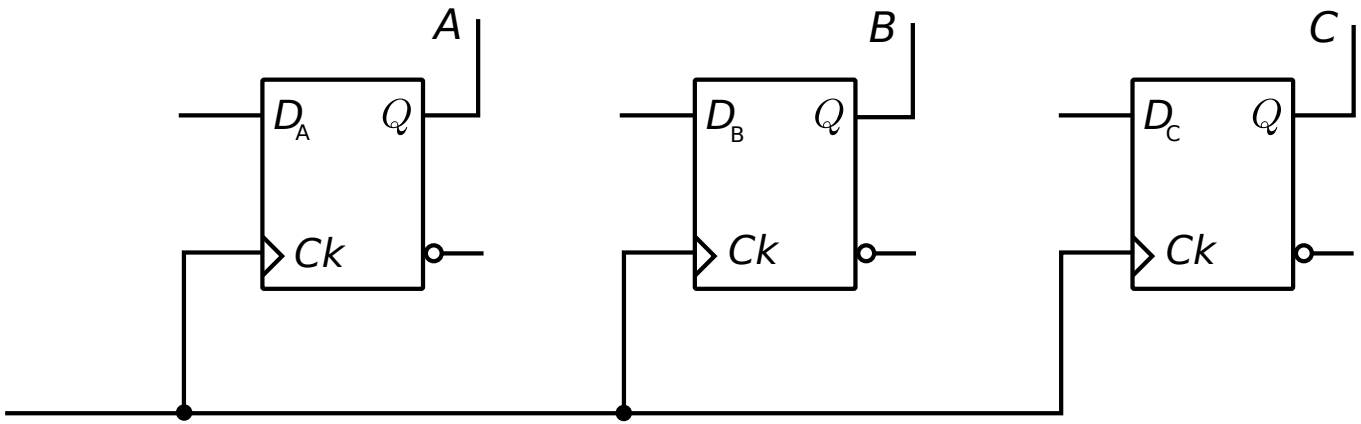
BC \ A	0	1
00	0	1
01	×	×
11	0	1
10	1	×

$B+$

BC \ A	0	1
00	0	1
01	×	×
11	0	0
10	1	×

$C+$

Counter design using D flip-flops



Next class...

- Registers
- Please read Lecture 25

ECE 210
Introduction to Digital Logic Design

Lecture 25
Registers

Registers

$\rightarrow$ A resister is a group of Flip-Flops that are used as a single unit to store a group of bits

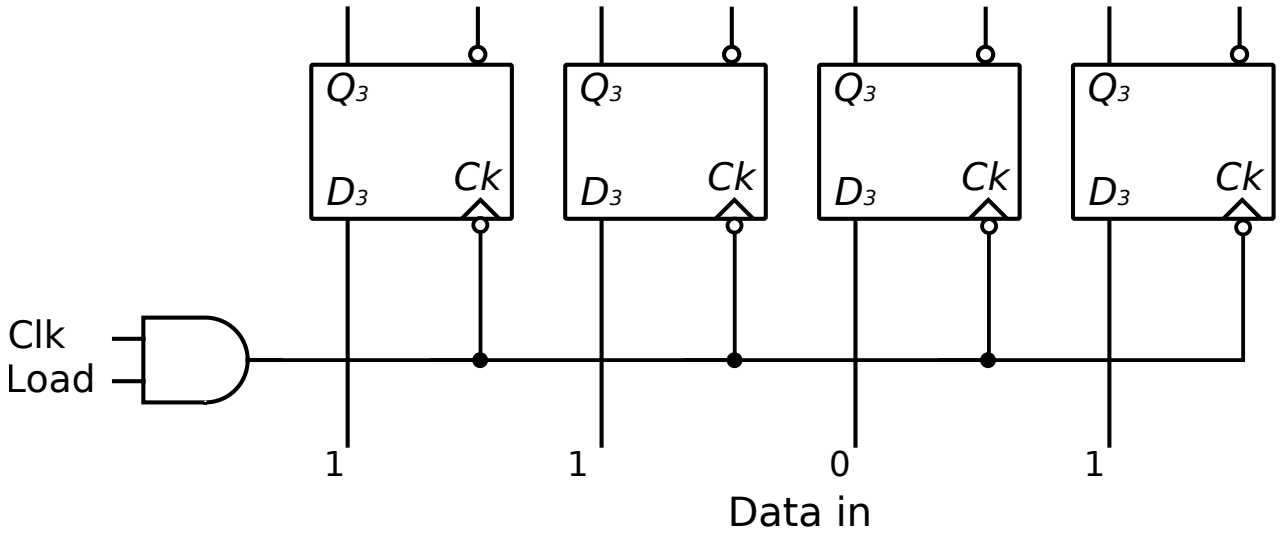
$\rightarrow$ Several D Flip-Flops may be grouped together with a common clock to form a single register

$\rightarrow$ Each FF can store one bit of information in the output Q_i

Characteristic equations of D-flip flops: $Q^+ =$

Registers

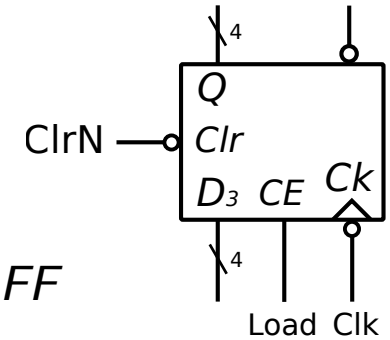
4-bit register with parallel load and mode control



Changes are triggered by the clock input on falling edge.

Load	Next state Q_3^+	Q_2^+	Q_1^+	Q_0^+	
0	Q_3	Q_2	Q_1	Q_0	No change
1	D_3	D_2	D_1	D_0	Parallel load

Registers with *CE* input

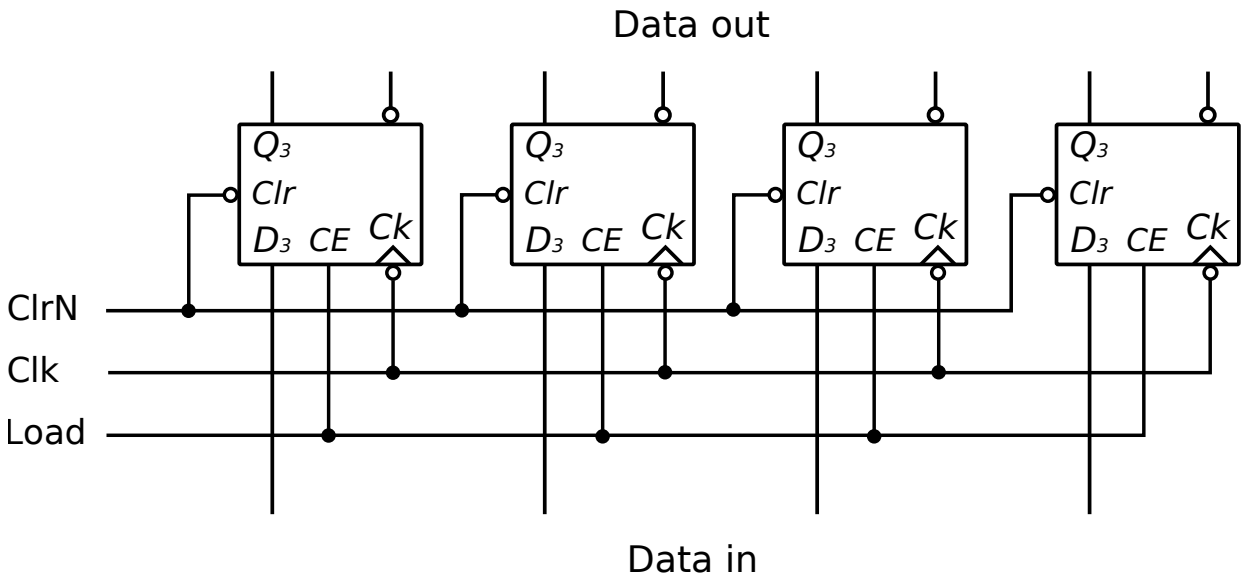


Gating the clock with *Load* can cause timing problems.

$\rightarrow$ *Load* $= 0$: the clock is disable and the data is locked in the *FF*

$\rightarrow$ *Load* $= 1$: data will be loaded into *FF* following the falling edge

FF may have a common clear signal: *ClrN* $= 0$ resets all *FF* outputs to 0.



Shift registers

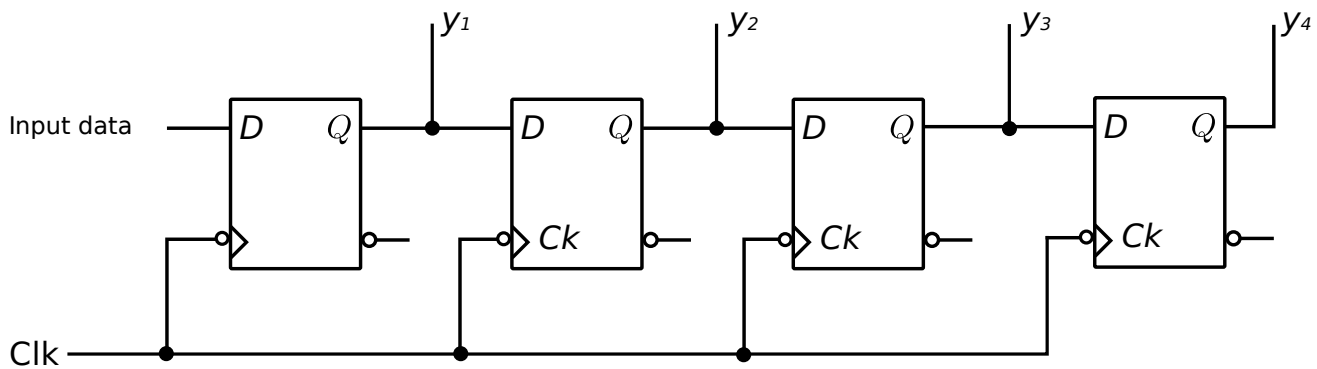
Right-shift register

$\rightarrow$ The stored data is shifted to the right on falling clock edge

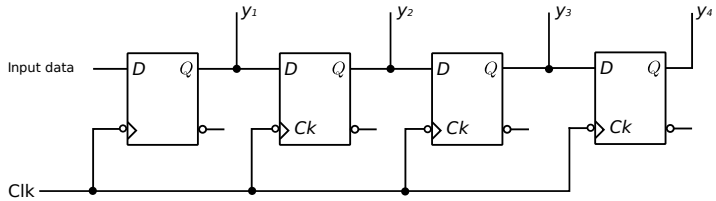
$\rightarrow$ The input line is transferred to the output of the 1^{st} FF;

$\rightarrow$ The output of 1^{st} FF is transferred to the 2^{st} FF;

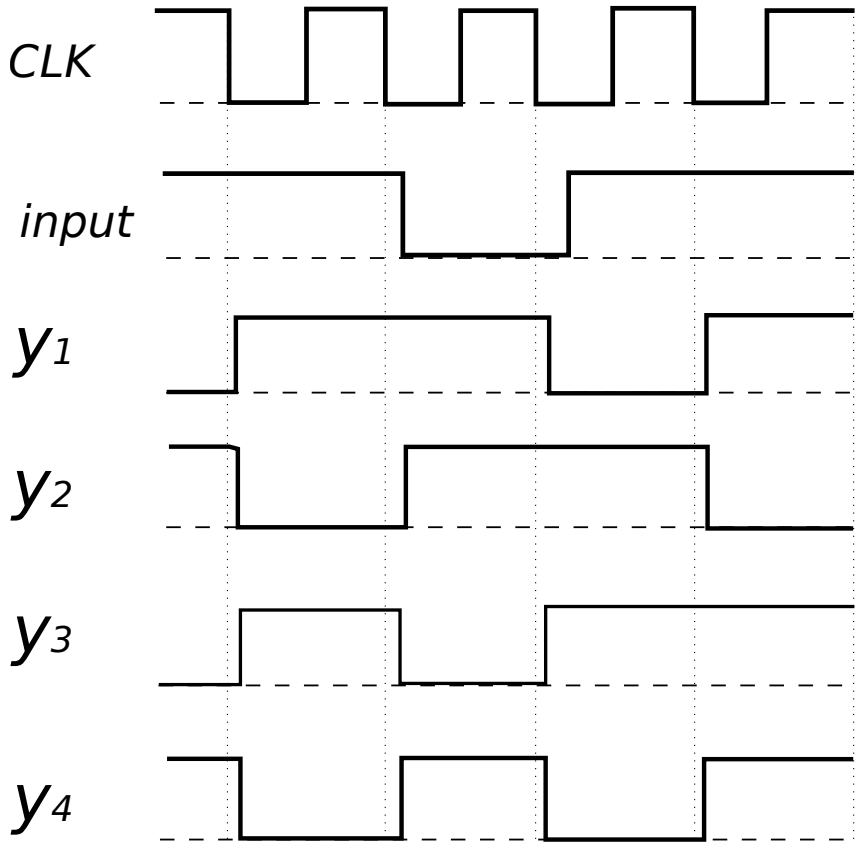
$\rightarrow$ The process carries on until the last FF.



Shift register - timing diagram



Initial state is 0101.



Shift registers

Serial *vs* parallel registers

$\rightarrow$ Serial registers take the data into the first *FF* one bit at a time.

$\rightarrow$ Serial registers provide output only in serial mode of the last *FF*. The output of the other *FF* are internal to the circuit.

$\rightarrow$ Parallel registers can load all bits at the same time;

$\rightarrow$ Parallel registers allow reading all bits at the same time;

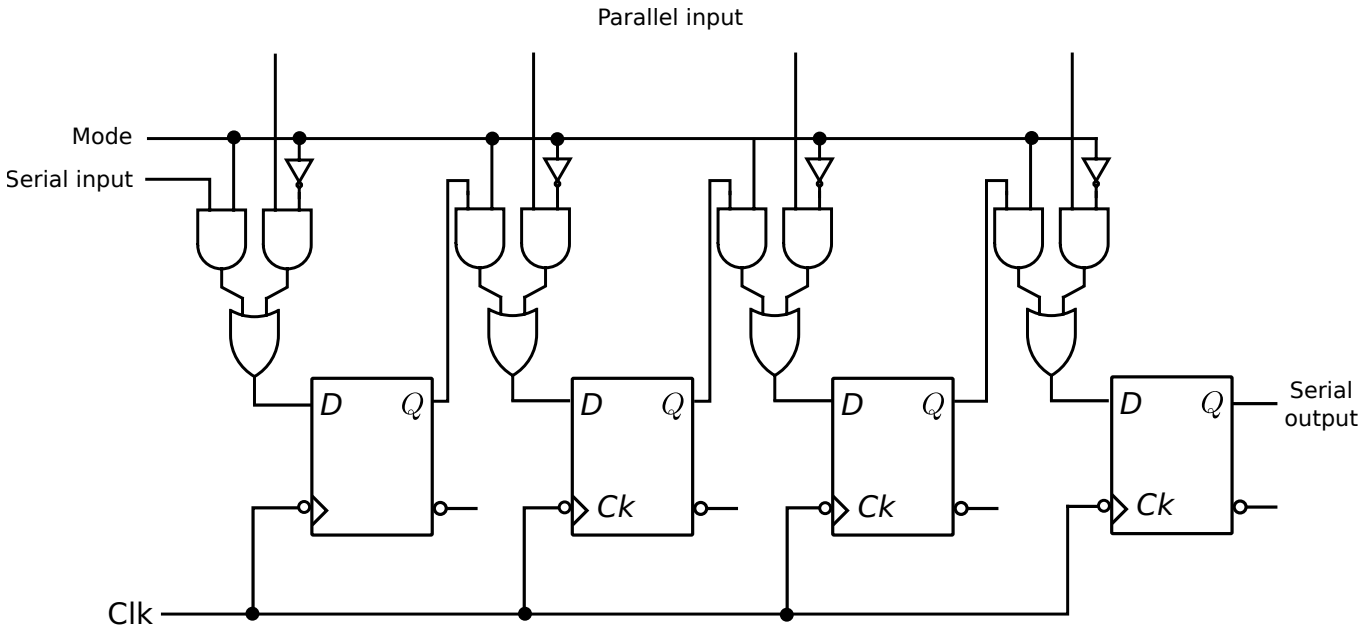
Parallel shift registers are used to convert parallel data to serial data.

Parallel and serial register

Parallel-load serial out shift register

$\rightarrow$ $Mode = 1$ The resister serially inputs the data

$\rightarrow$ $Mode = 0$ The resister inputs the data in parallel form



Universal shift register

Shift register can be used to convert serial to parallel data and vice versa.

→ Data enters and exits serially by shifting the register

→ Data entered in parallel can be taken out in serial

In a general shift register

→ A *clear* line forces the register to 0

→ A clock enable *CE* line leaves the information in the register unchanged

→ A parallel load control enables conversion from parallel to serial

Applications:

→ Transmission of data over a single line channel

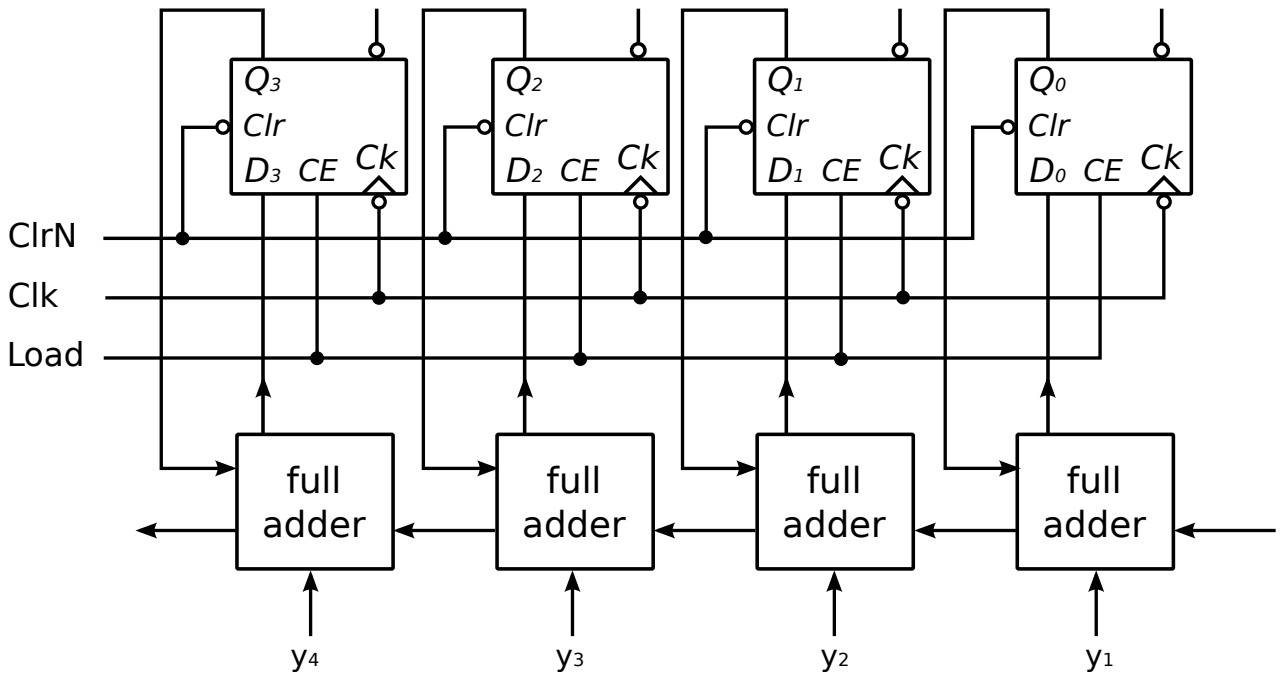
→ Adding binary numbers.

Parallel adder with accumulator

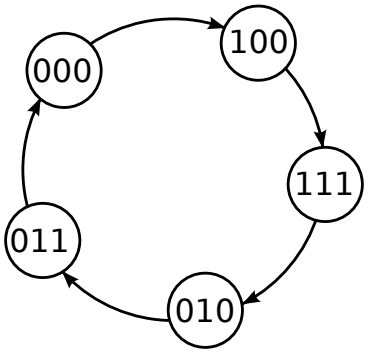
Stores on number in a register of FF ($X = Q_3 Q_2 Q_1 Q_0$)

Adds a second number to it ($Y = y_4 y_3 y_2 y_1$),

Leaves the result stored in the accumulator.



Counters using S-R Flip-Flops

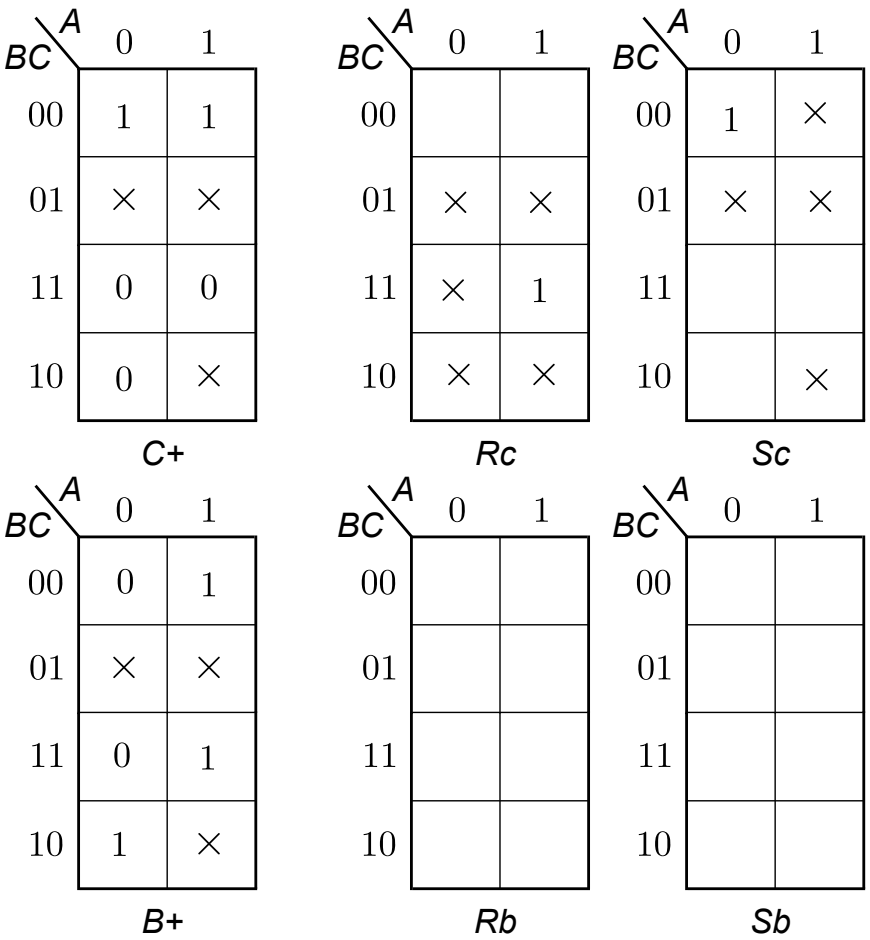


Next state table for a $S - R$ flip-flop.

Q	Q^+	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

Current state			Next state			FF inputs	
A	B	C	A^+	B^+	C^+	S_C	R_C
0	0	0	1	0	0	1	0
0	0	1	×	×	×	×	×
0	1	0	0	1	1		
0	1	1	0	0	0		
1	0	0	1	1	1		
1	0	1	×	×	×	×	×
1	1	0	×	×	×	×	×
1	1	1	0	1	0	0	1

Counters using S-R Flip-Flops



Counters using S-R Flip-Flops

BC \ A	0	1
00	0	1
01	×	×
11	0	0
10	1	×

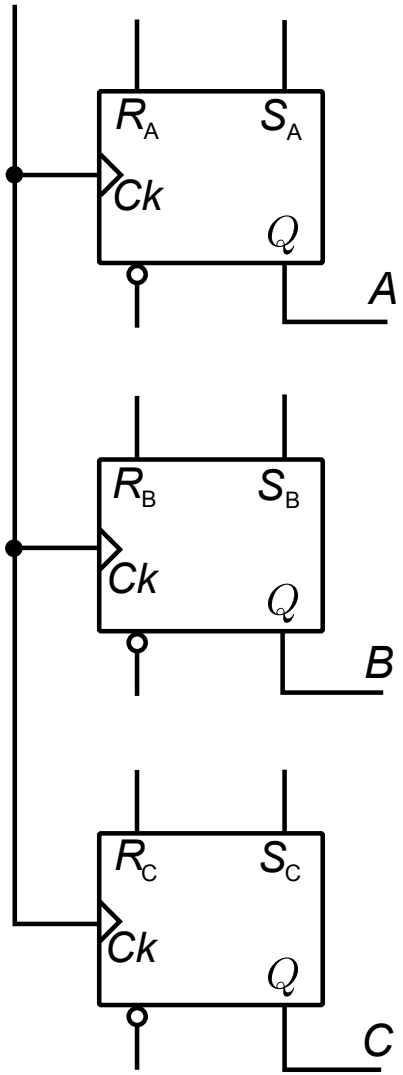
A+

BC \ A	0	1
00		
01		
11		
10		

Ra

BC \ A	0	1
00		
01		
11		
10		

Sa



Next class...

- Analysis of Clocked Sequential Circuits
- Please read Lecture 26

ECE 210
Introduction to Digital Logic Design

Lecture 26
Analysis of Clocked Sequential Circuits

Sequential circuits

In sequential circuits the sequences of outputs generally depends on the input sequence.

To analyse the circuit we can

$\rightarrow$ Trace 0 and 1 signals through the circuit (timing diagram)

$\rightarrow$ Create a state state to represent the behaviour of the circuit

Design requires studying timing relationship between inputs, clock, and output

Serial transmission

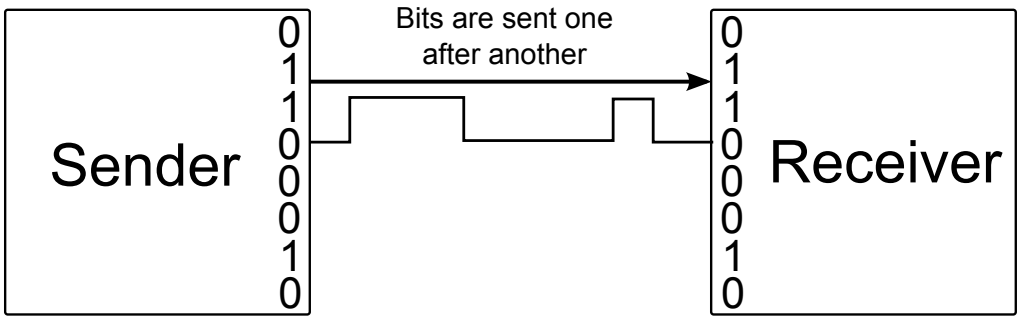
Transmission on a communications channel between two machines can occur in several different ways.

The transmission is characterised by:

→ The direction of the exchanges

→ The transmission mode: the number of bits sent simultaneously

→ Synchronization between the transmitter and receiver



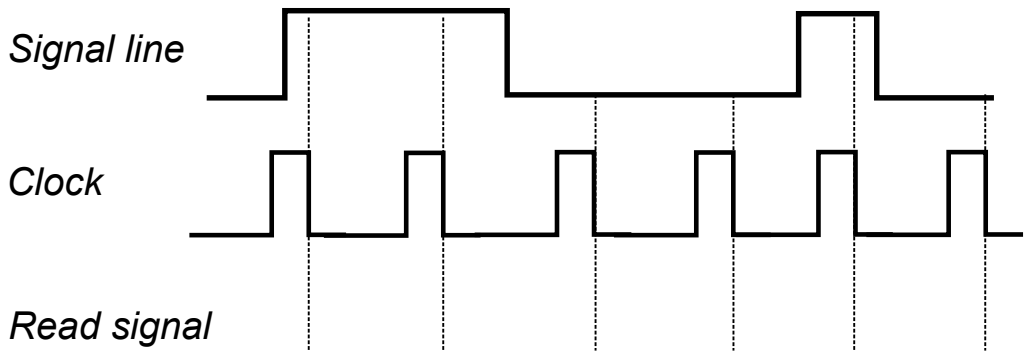
Serial transmission

Synchronous transmission

→ The transmitter and receiver are placed by the same clock.

→ The receiver continuously receives the information at the **same rate** the transmitter sends it.

→ Supplementary information is inserted to guarantee that there are no errors during transmission.



Sequential parity checker

Consider a 7-bit code for information exchange

In serial transmission, an 8^{th} bit is added for error detection

$\rightarrow$ If the 8^{th} bit makes total number of 1's odd : odd parity

$\rightarrow$ If the 8^{th} bit makes total number of 1's even : even parity

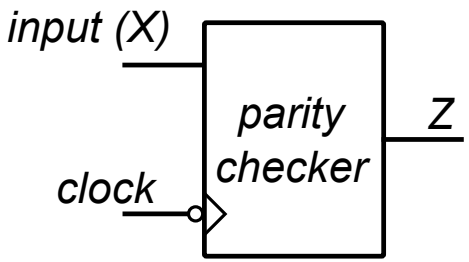
8-bit word	
0000000	1
0000001	0
1100000	1
0100101	0

The parity bit can be chosen such that the parity is always even:

$\rightarrow$ If any single bit changes during transmission, the parity is no longer checked.

$\rightarrow$ Transmission errors can be detected.

Parity checker for serial data



The output should be

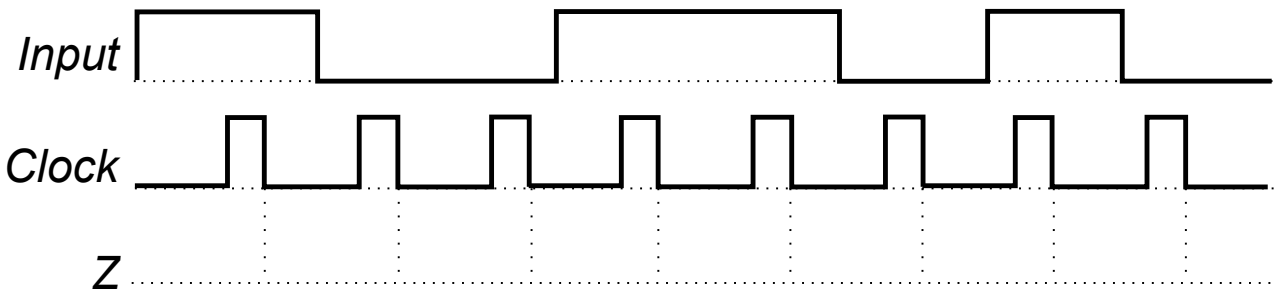
$\rightarrow$ $Z = 1$ if the total number of 1 is odd

$\rightarrow$ $Z = 0$ if the total number of 1 is even

Transmission error occurs if

$\rightarrow$ Data had odd parity and the final output is $Z = 0$

$\rightarrow$ Data had even parity and the final output is $Z = 1$



Parity checker for serial data

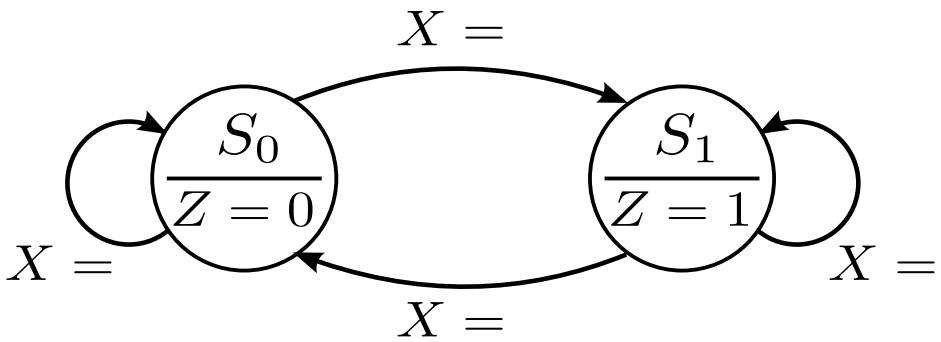
State graph

$\rightarrow$ Circuit must remember whether the number of 1 inputs is odd or even

$\rightarrow$ Two states are required: S_0 and S_1

$\rightarrow$ State S_0 means that the number of 1's is even: $Z =$

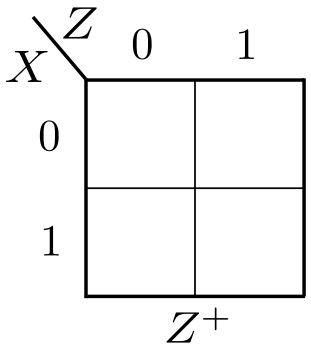
$\rightarrow$ State S_1 means that the number of 1 's is odd: $Z =$

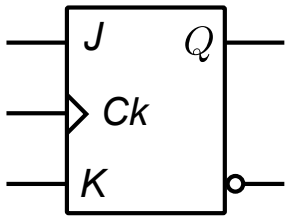


Parity checker for serial data

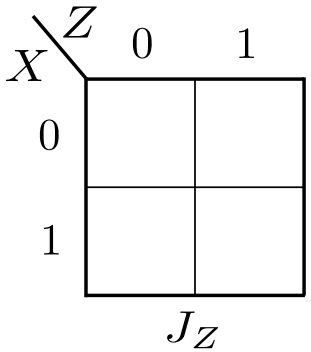
Implement the circuit using a JK flip-flop

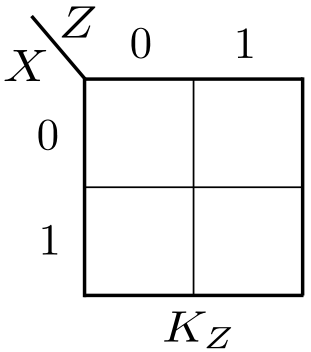
X	Z	Z^+
0	0	0
0	1	1
1	0	1
1	1	0



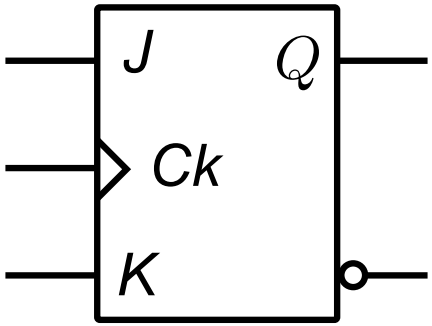


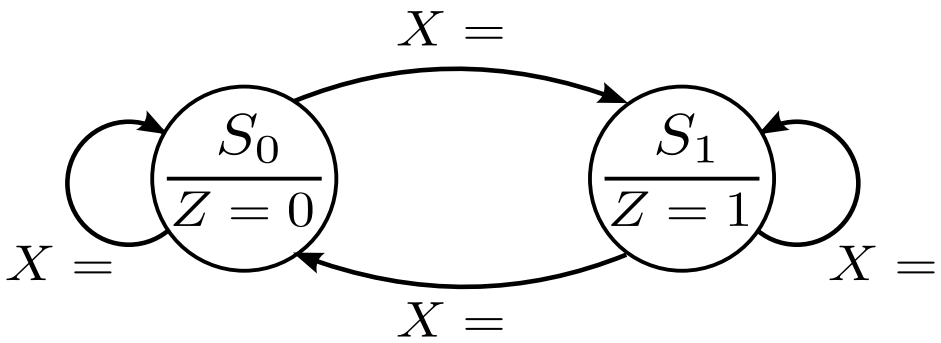
Q	Q^+	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0





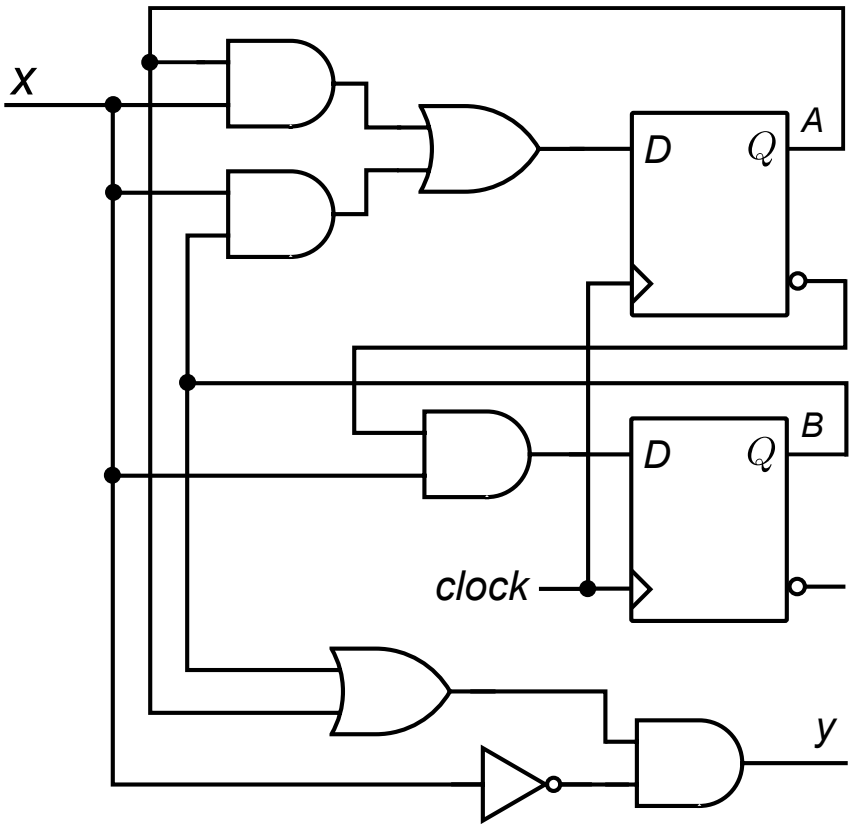
Parity checker for serial data





State equations

A state equation specifies the next state as a function of the present state and inputs.

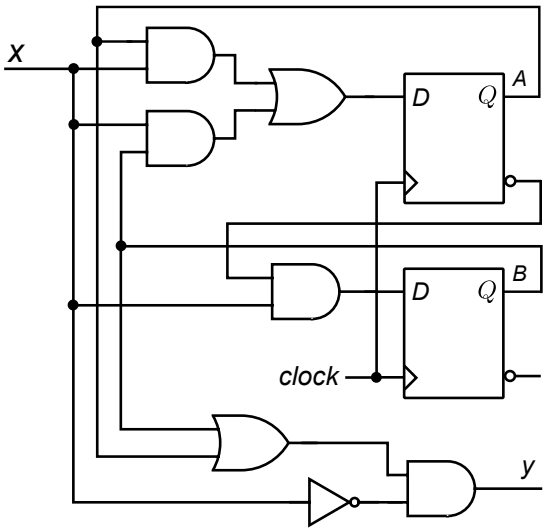


$A^+ =$

$B^+ =$

$y =$

State table

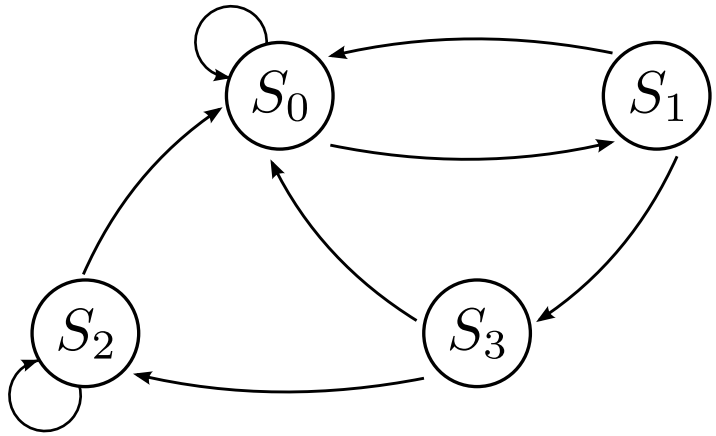


AB	A^+B^+	
	$X=0$	$X=1$
00	00	01
01	00	11
10	00	10
11	00	10

AB	A^+B^+	
	$X=0$	$X=1$
S_0		
S_1		
S_2		
S_3		

$$A^+ = X(A+B)$$

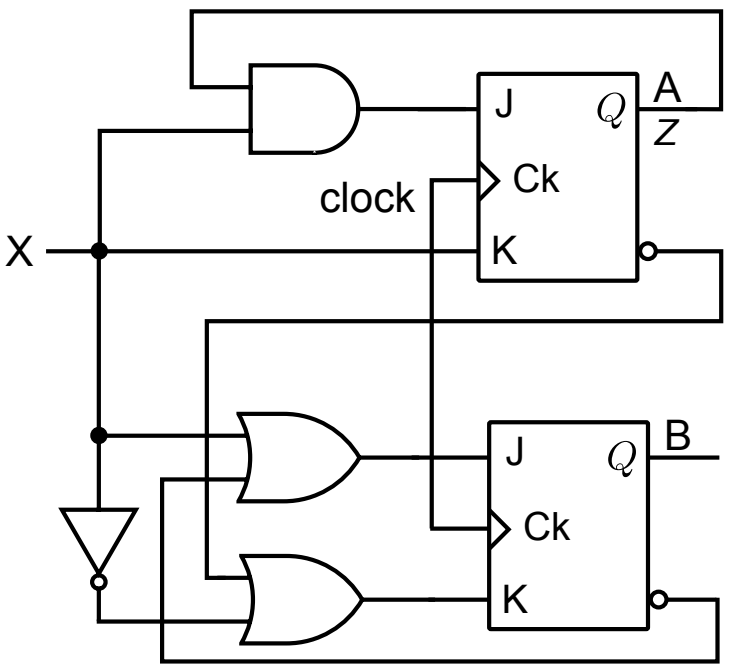
$$B^+ = \overline{A}X$$



State gragh (without output)

State equations

State equation for a JK-F: $Q^+ = J\overline{Q} + \overline{K}Q$.



$J_A =$

$K_A =$

$J_B =$

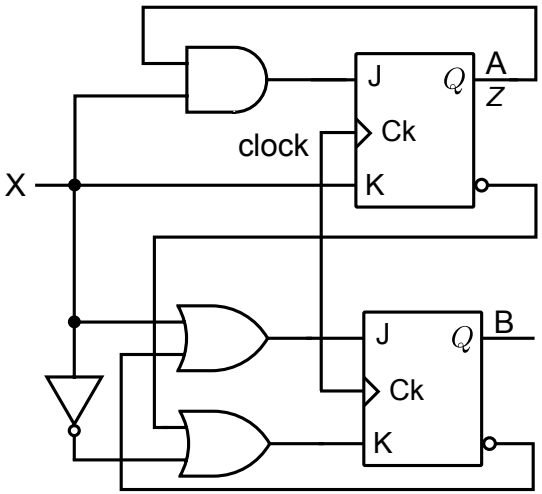
$K_B =$

$A^+ =$

$B^+ =$

$z = A$

State table

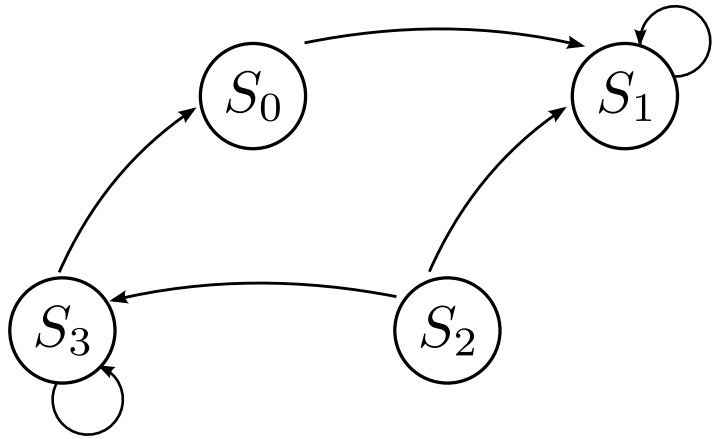


AB	A^+B^+	
	$X=0$	$X=1$
00	01	01
01	01	01
10	11	01
11	11	00

AB	A^+B^+	
	$X=0$	$X=1$
S_0		
S_1		
S_2		
S_3		

$$A^+ = \overline{X}A$$

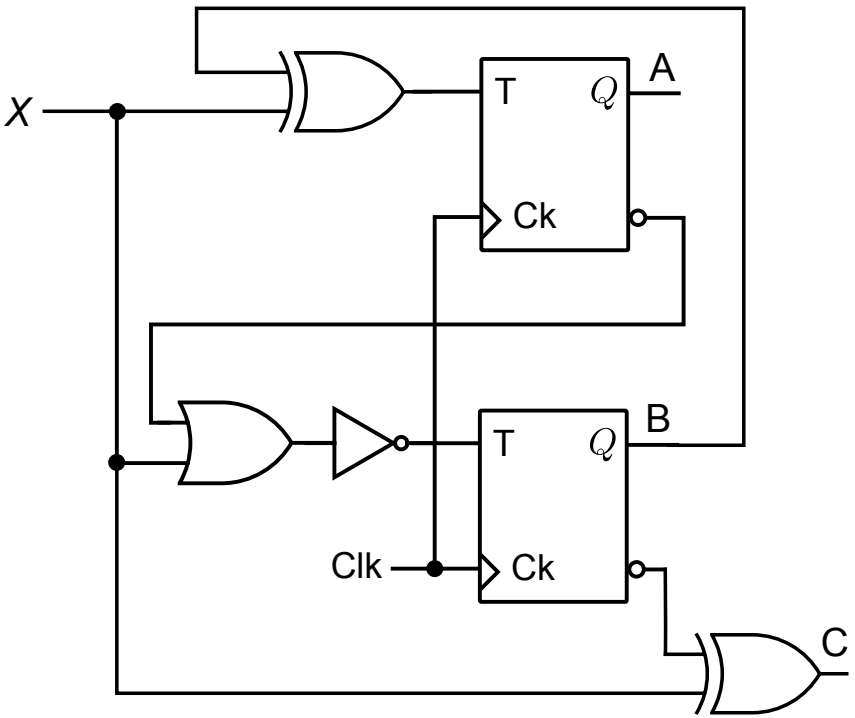
$$B^+ = \overline{A} + \overline{B} + \overline{X}$$



State gragh (without output)

State equations

State equation for a *TFF*: $Q^+ = T \oplus Q$.



$T_A =$

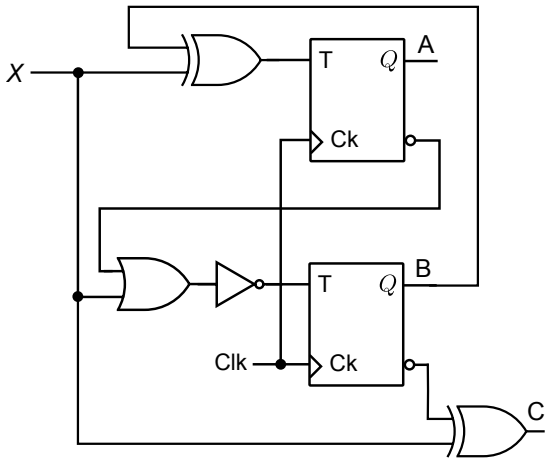
$T_B =$

$A^+ =$

$B^+ =$

$C = \overline{B} \oplus X$

State table



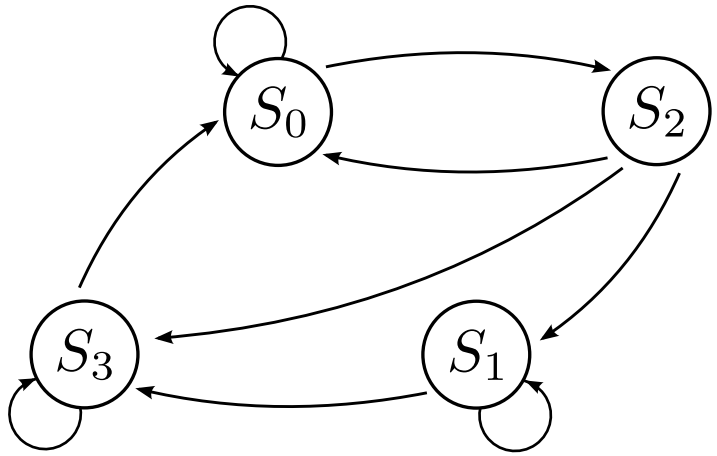
AB	A^+B^+	
	$X=0$	$X=1$
00	00	10
01	11	01
10	11	00
11	00	11

AB	A^+B^+	
	$X=0$	$X=1$
S_0		
S_1		
S_2		
S_3		

$$A^+ = A \oplus B \oplus X$$

$$B^+ = B \oplus (A\overline{X})$$

State gragh (without output)



Next class...

- Moore State Machines
- Please read Lecture 27

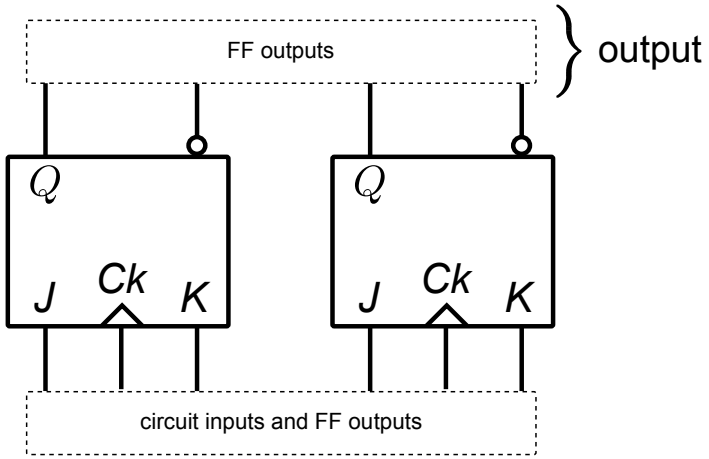
ECE 210
Introduction to Digital Logic Design

Lecture 27
Analysis of Moore State Machines

Categories of sequential circuits

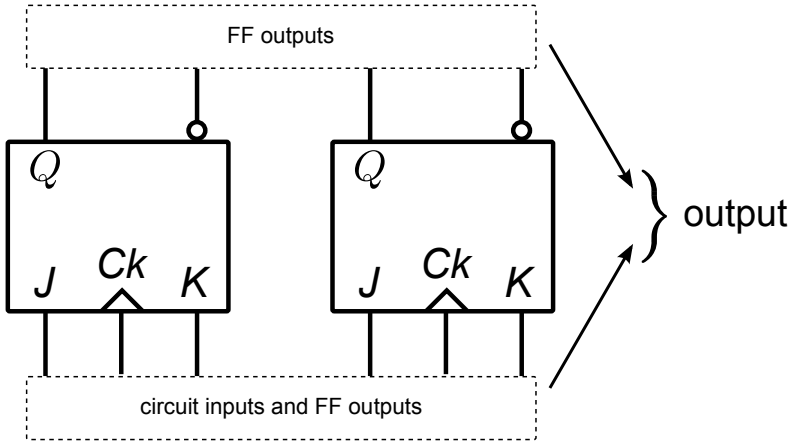
Moore machine

→ Output is a function of
- The present state of *FF*'s only



Mealy machine

→ Output is a function of
- The present state of *FF*'s
- And the input to the circuit



Analysis of sequential networks

Two methods for analysis of sequential circuits:

Using signal tracing and timing diagram

$\rightarrow$ This will work for small circuits

$\rightarrow$ Also used to design sequential circuits

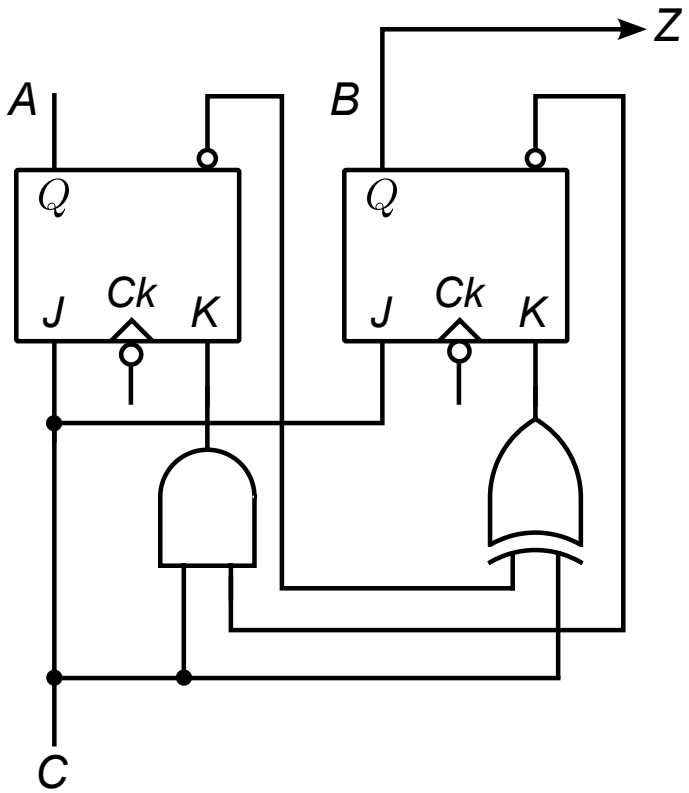
Using state tables and graphs

$\rightarrow$ More convenient for large circuits

Analysis of sequential networks

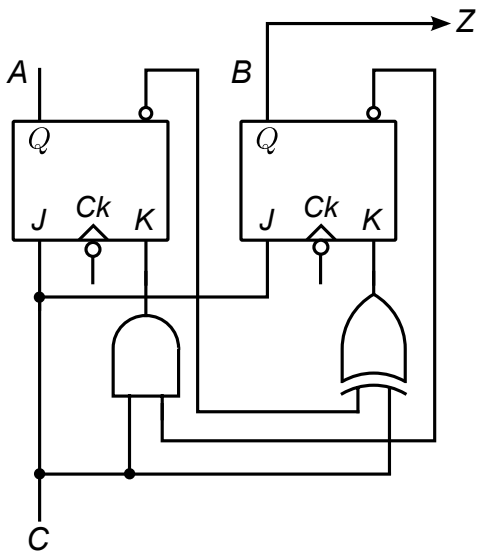
Moore machine

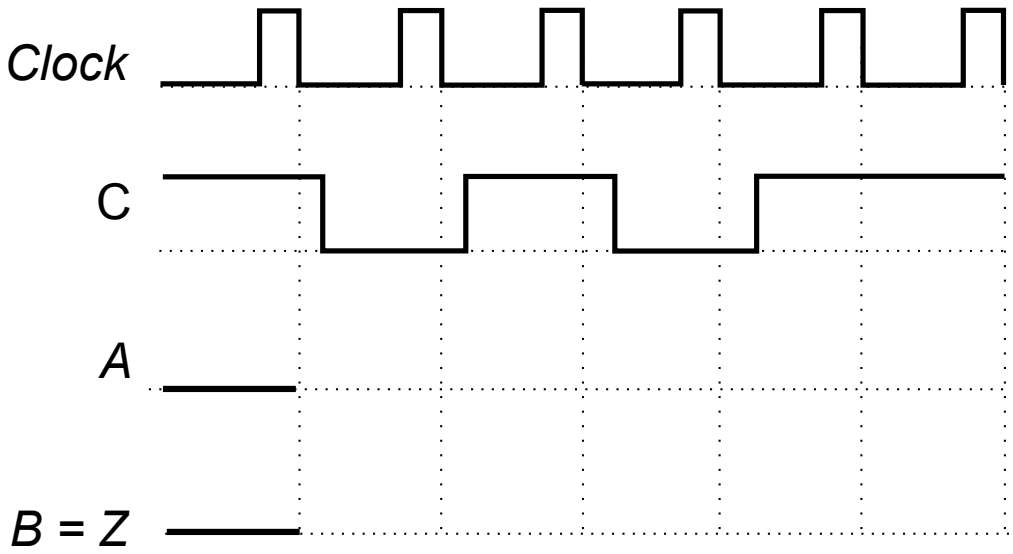
Example 1: The output is a function of the present state of the FF only



Analysis of sequential networks

Moore machine: Changes only occur after active clock edge





KQ \ J	0	1
00	0	1
01	1	1
11	0	0
10	0	1

Q+

Analysis of sequential networks

Using state tables and graphs

1. Determine the *FF* input and output equations
2. Derive the next state equations for each FF
3. Plot the next state K-map for each FF
4. Create a state table from step 2 or 3

Analysis of sequential networks

Using state tables and graphs

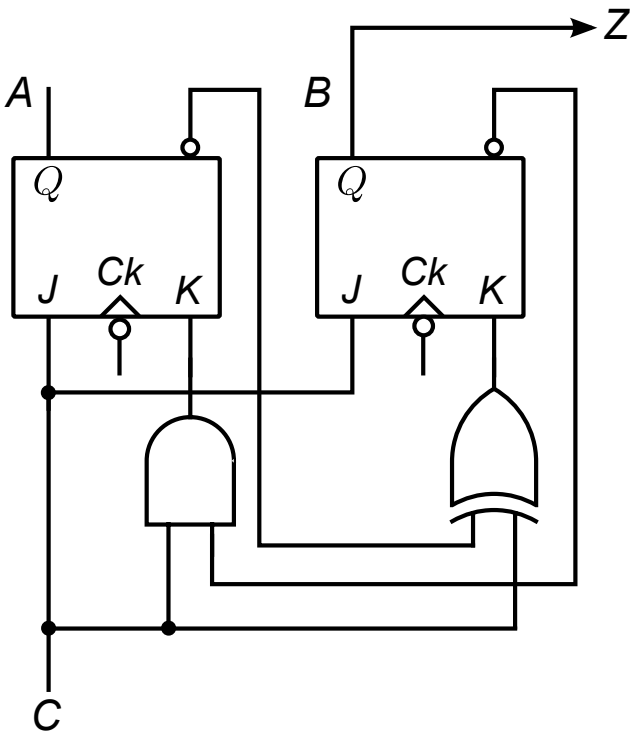
Example using Moore machine

Step 1 - FF input and output equations

$J_A =$ $K_A =$

$J_B =$ $K_B =$

$Z = B$



Analysis of sequential networks

Using state tables and graphs

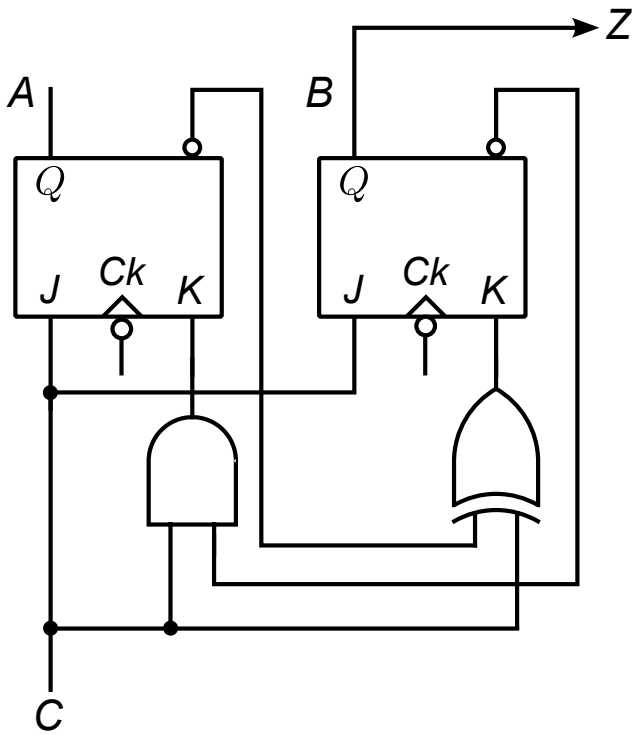
Example using Moore machine

Step 2 - FF next state equations

$A^+ = J_A\overline{A} + \overline{K}_A A$

$A^+ =$

$B^+ =$

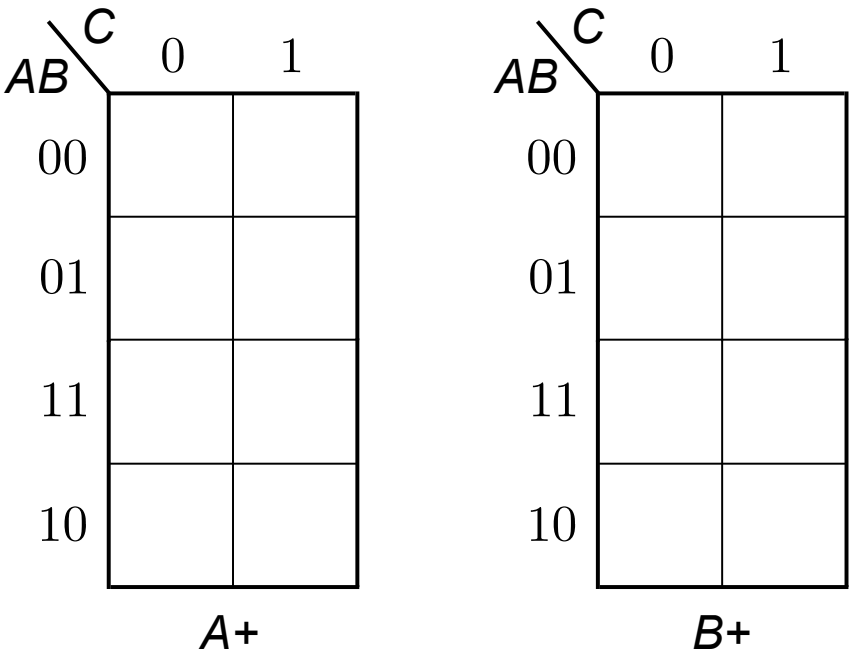


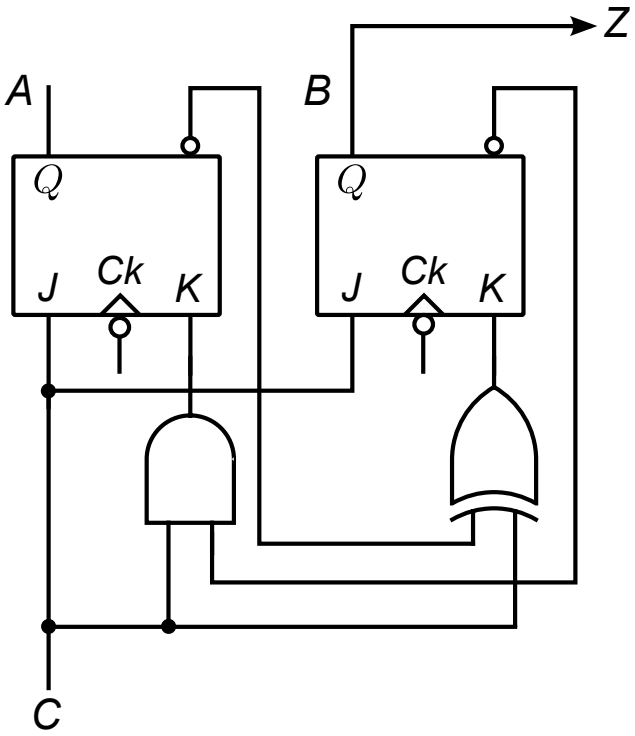
Analysis of sequential networks

Using state tables and graphs

Example using Moore machine

Step 3 - Next state K-map for each FF





$$A^+ = \overline{A}C + A\overline{C} + AB$$

$$B^+ = \overline{B}C + \overline{A}BC + AB\overline{C}$$

$$Z = B$$

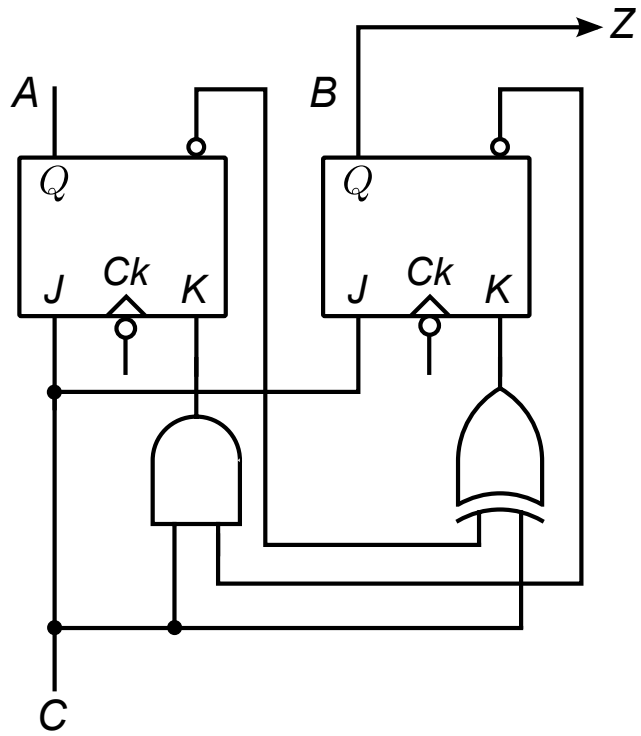
Analysis of sequential networks

Using state tables and graphs

Example using Moore machine

Step 4 - Transition table

AB	A^+B^+ $C=0$	A^+B^+ $C=1$	present output Z
00			
01			
11			
10			



AB \ C	0	1
00	0	1
01	0	1
11	1	1
10	1	0

A+

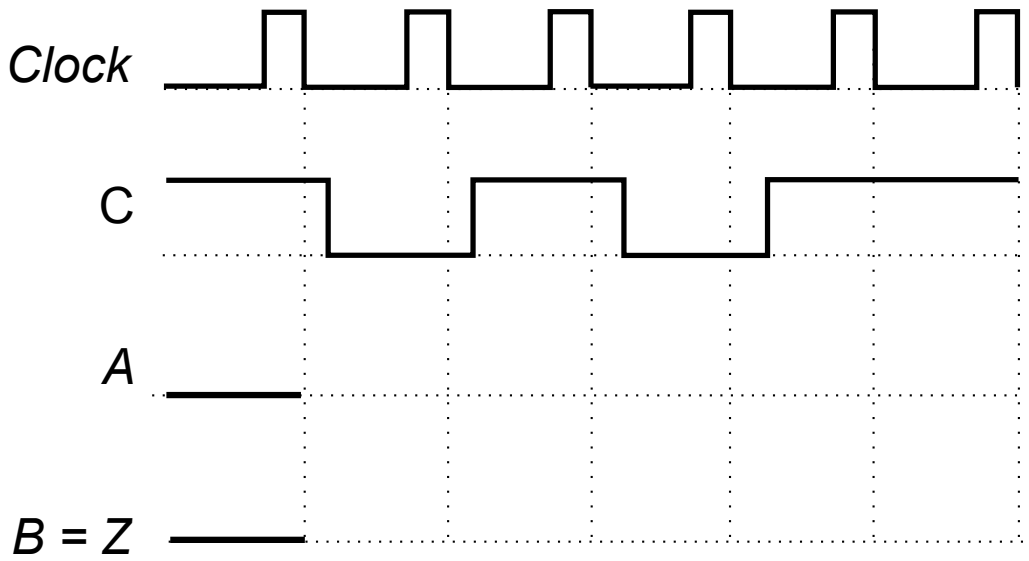
AB \ C	0	1
00	0	1
01	0	1
11	1	0
10	0	1

B+

Analysis of sequential networks

Timing chart from the transition table

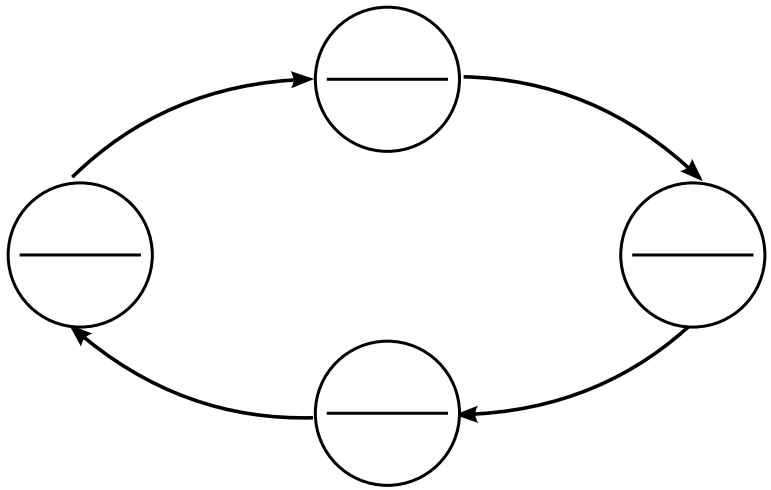
	A^+B^+		present
AB	$C = 0$	$C = 1$	output Z
00	00	11	0
01	00	11	1
10	10	01	0
11	11	10	1



Analysis of sequential networks

Moore state graph.

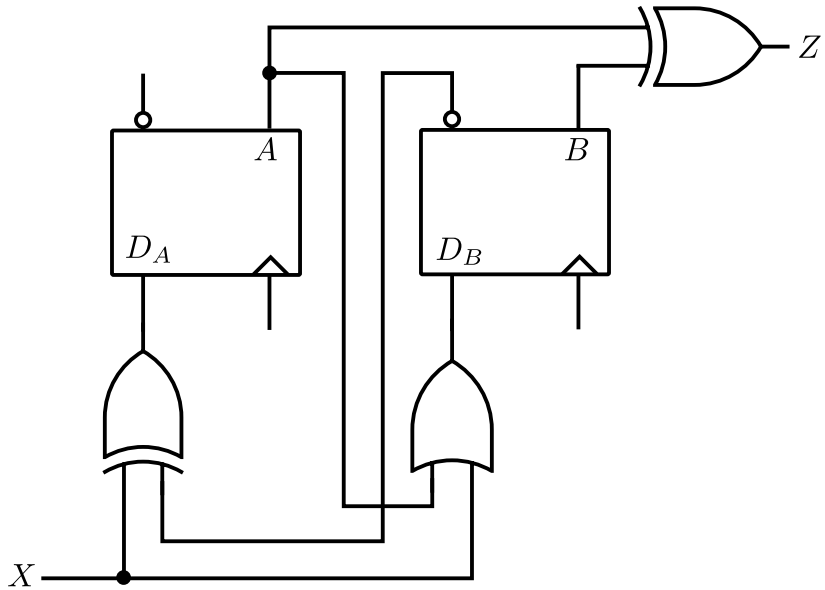
	A^+B^+			present	next state		
AB	$C=0$	$C=1$	Z	state	$C=0$	$C=1$	Z
00	00	11	0	S_0			
01	00	11	1				
11	11	10	1				
10	10	01	0				



Analysis of sequential networks

Using state tables and graphs

Example using Moore machine



FF input and output equations

$D_A =$

$D_B =$

$Z =$

$A^+ =$

$B^+ =$

Analysis of sequential networks

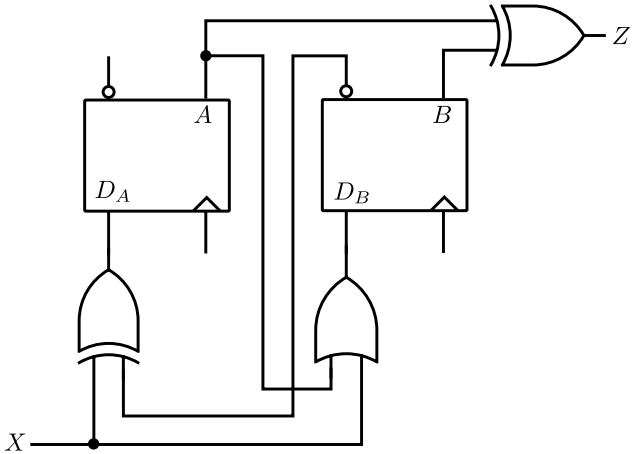
Next state K-map for each FF

AB \ X	0	1
00		
01		
11		
10		

A+

AB \ X	0	1
00		
01		
11		
10		

B+



$$A^+ = X \oplus \overline{B}$$

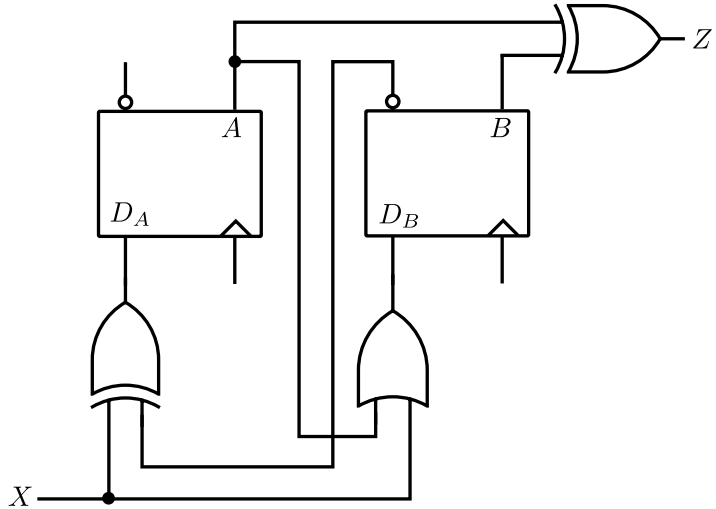
$$B^+ = X + A$$

$$Z = A \oplus B$$

Analysis of sequential networks

Transition table

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	present output Z
00			
01			
11			
10			



State table

present state	next state $X=0$	next state $X=1$	Z
S_0			
S_1			
S_2			
S_3			

AB \ X	0	1
00	1	0
01	0	1
11	0	1
10	1	0

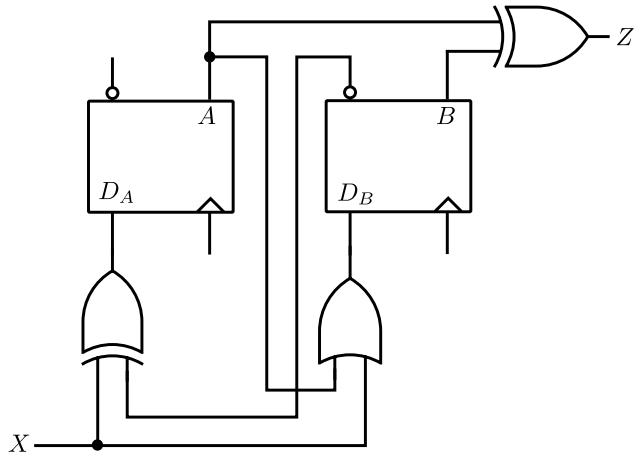
A+

AB \ X	0	1
00	0	1
01	0	1
11	1	1
10	1	1

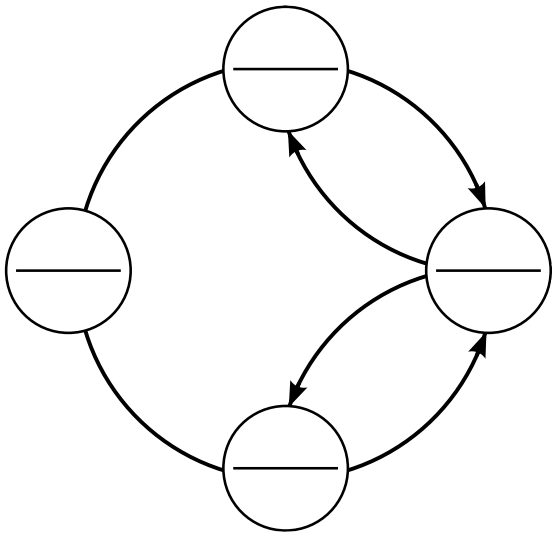
B+

Analysis of sequential networks

present state	next state $X=0$	next state $X=1$	Z
00	10	01	0
01	00	11	1
11	01	11	0
10	11	01	1



present state	next state $X=0$	next state $X=1$	Z
S_0	S_3	S_1	0
S_1	S_0	S_2	1
S_2	S_1	S_2	0
S_3	S_2	S_1	1



Next class...

- Mealy state machines
- Please read Lecture 28

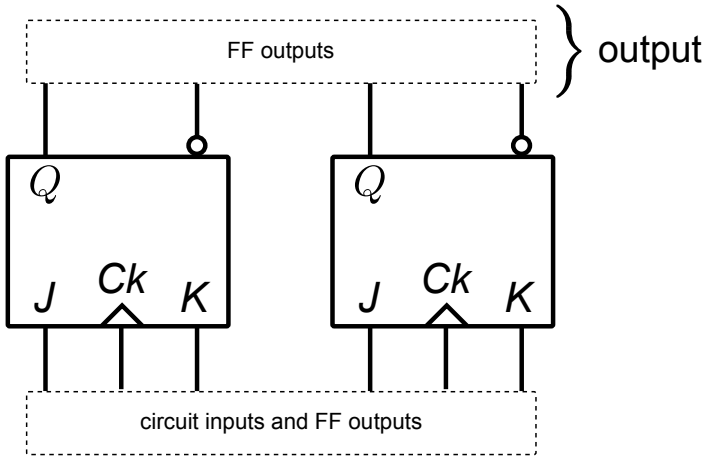
ECE 210
Introduction to Digital Logic Design

Lecture 28
Analysis of Mealy State Machines

Categories of sequential circuits

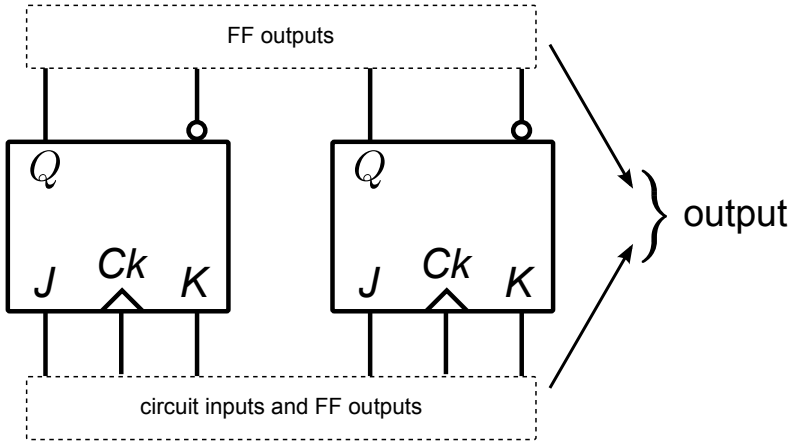
Moore machine

→ Output is a function of
- The present state of *FF*'s only



Mealy machine

→ Output is a function of
- The present state of *FF*'s
- And the input to the circuit



Analysis of sequential networks

Two methods for analysis of sequential circuits:

Using signal tracing and timing diagram

$\rightarrow$ This will work for small circuits

$\rightarrow$ Also used to design sequential circuits

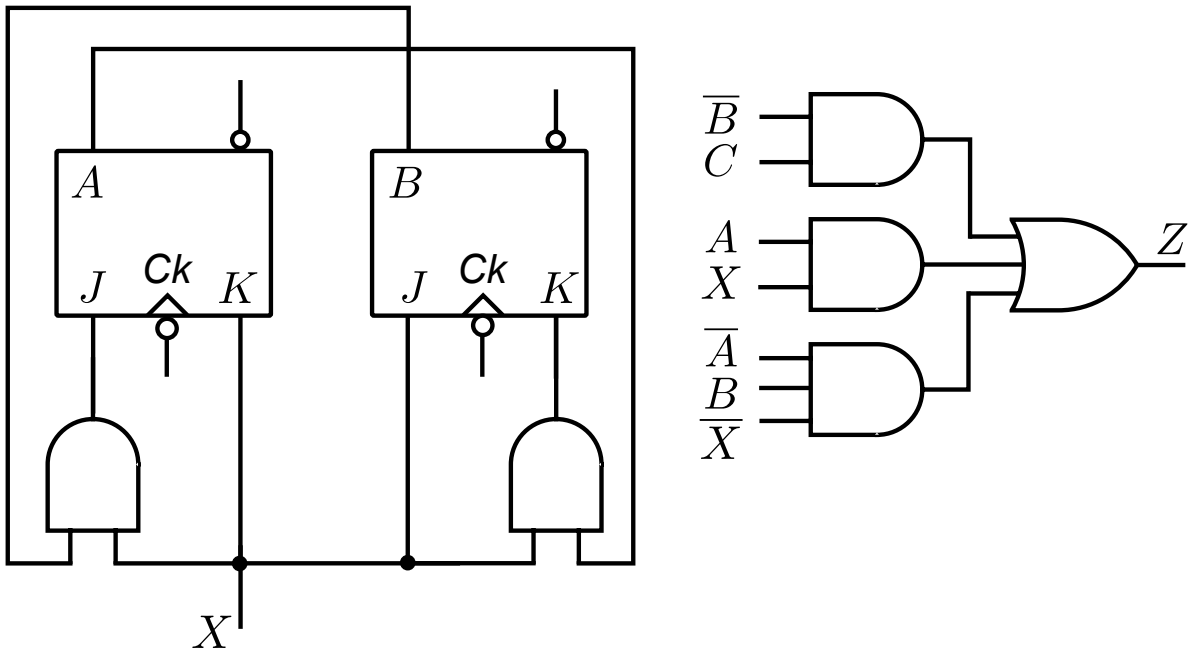
Using state tables and graphs

$\rightarrow$ More convenient for large circuits

Analysis of sequential networks

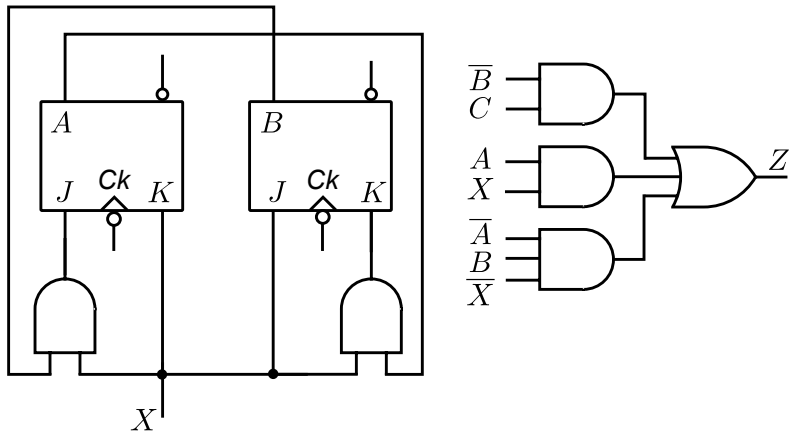
Mearly machine

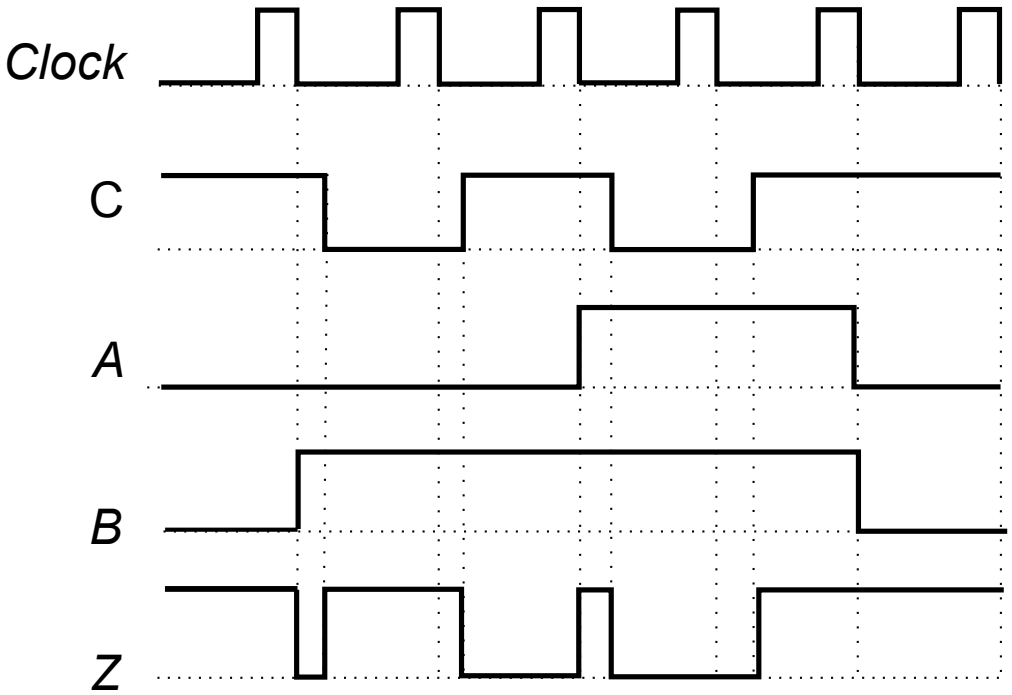
The output is a function of the present state, and of the input



Analysis of sequential networks

Mearly machine: Changes occur after active clock edge and changes in the input





KQ \ J	0	1
00	0	1
01	1	1
11	0	0
10	0	1

Q+

Analysis of sequential networks

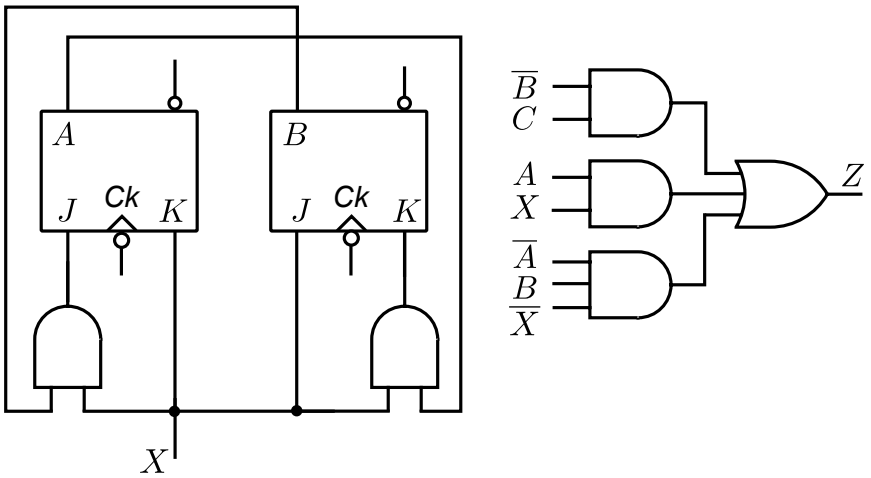
Analysis through state tables and graphs

1. Determine the *FF* input and output equations
2. Derive the next state equations for each *FF*
3. Plot the next state K-map for each *FF*
4. Create the state table and graph

Analysis of sequential networks

Using state tables and graphs

Example using Moore machine



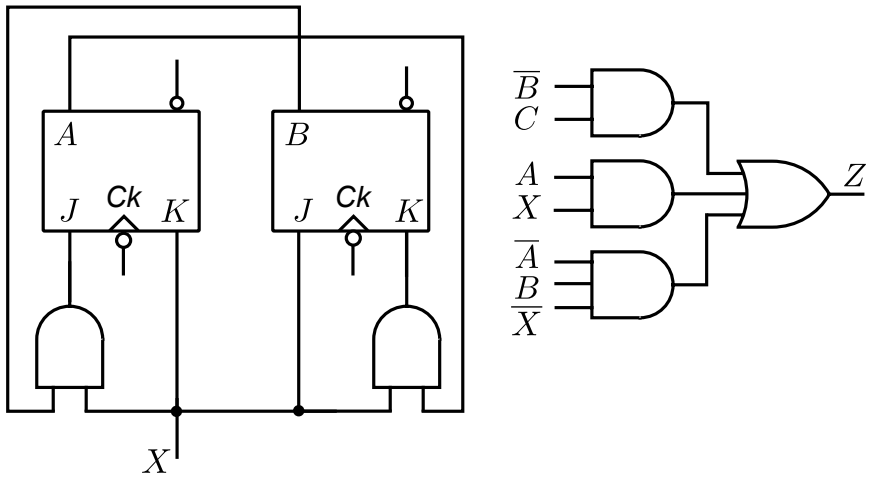
Step 1 - FF input and output equations

$J_A =$ $K_A =$

$J_B =$ $K_B =$

$Z =$

Analysis of sequential networks



Using state tables and graphs

Example using Moore machine

Step 2 - FF next state equations

$A^+ = J_A\overline{A} + \overline{K}_A A$

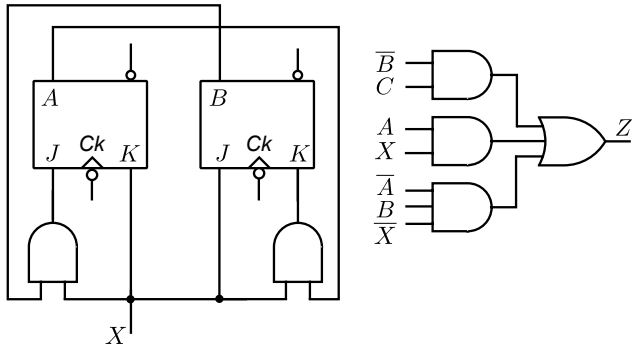
$A^+ =$

$B^+ =$

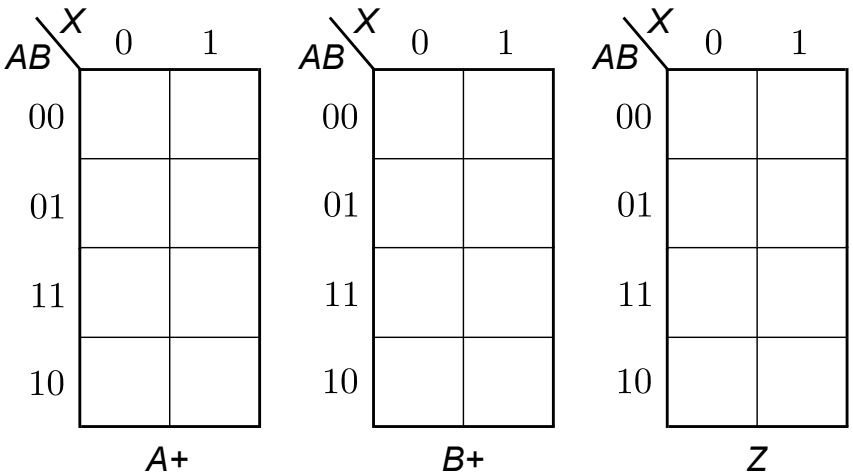
Analysis of sequential networks

Using state tables and graphs

Example using Mealy machine



Step 3 - Next state K-map for each FF



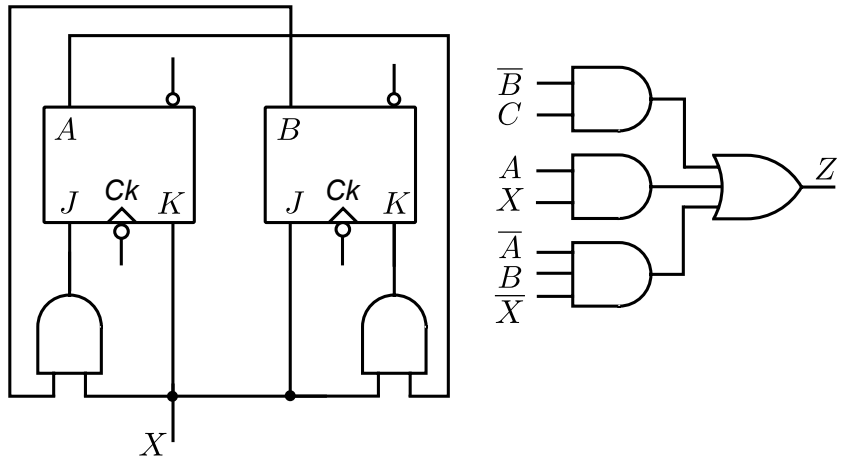
$$A^+ = XB\overline{A} + \overline{X}A$$

$$B^+ = X\overline{B} + \overline{X}B + \overline{A}B$$

$$Z = \overline{A}B\overline{X} + X\overline{B} + XA$$

Analysis of sequential networks

Using state tables and graphs



Step 4 - Transition table

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
00				
01				
11				
10				

AB \ X	0	1
00	0	0
01	0	1
11	1	0
10	1	0

A+

AB \ X	0	1
00	0	1
01	1	1
11	1	0
10	0	1

B+

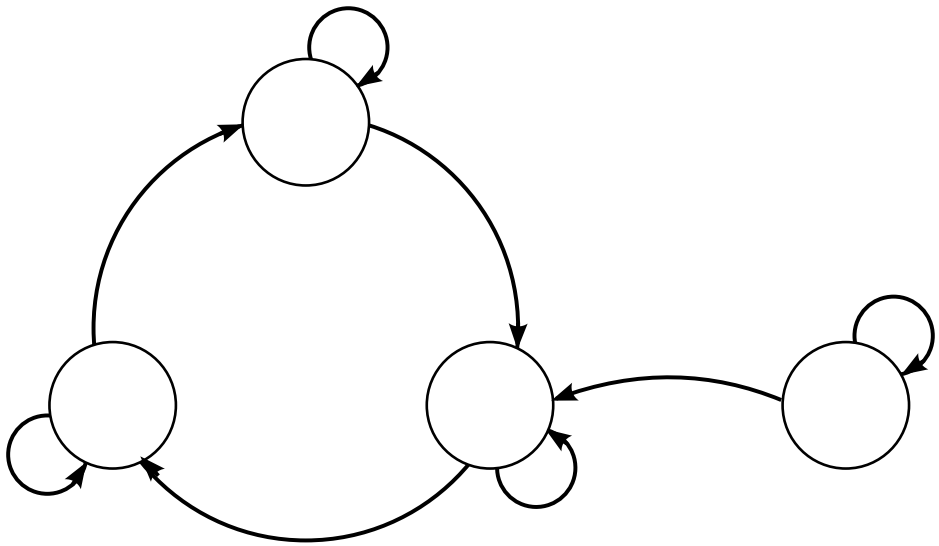
AB \ X	0	1
00	0	1
01	1	0
11	0	1
10	0	1

Z

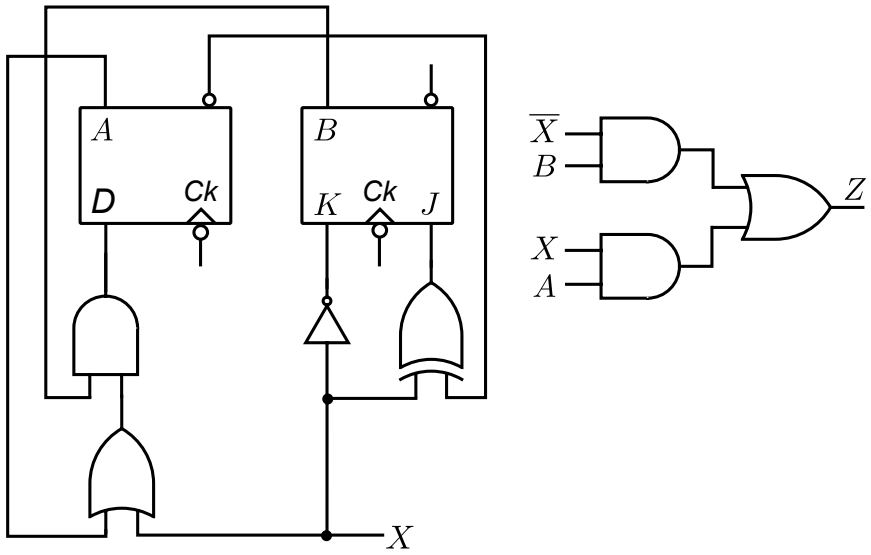
Analysis of sequential networks

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
S_0			0	1
S_1			1	0
S_2			0	1
S_3			0	1



Analysis of sequential networks



FF input and output equations

$J =$ $K =$

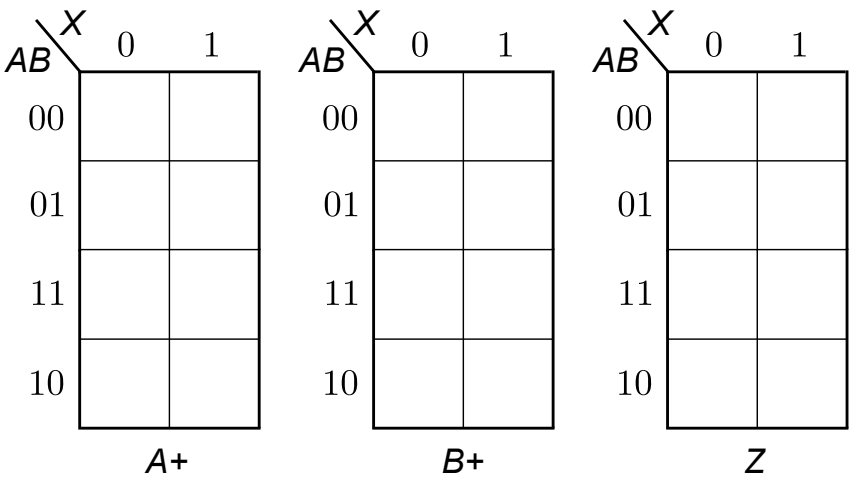
$D =$

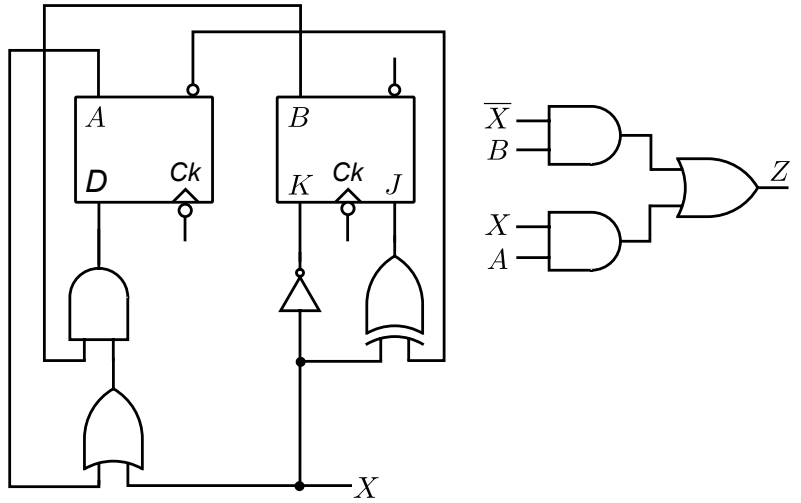
$Z =$

$A^+ =$

$B^+ =$

Analysis of sequential networks





$$A^+ = B(A + X)$$

$$B^+ = (\overline{A} \oplus X)\overline{B} + XB = \bar{A}\bar{B}\bar{X} + A\overline{B}X + XB$$

$$Z = B(A + X)$$

Analysis of sequential networks

AB \ X	0	1
00		
01		1
11	1	1
10		

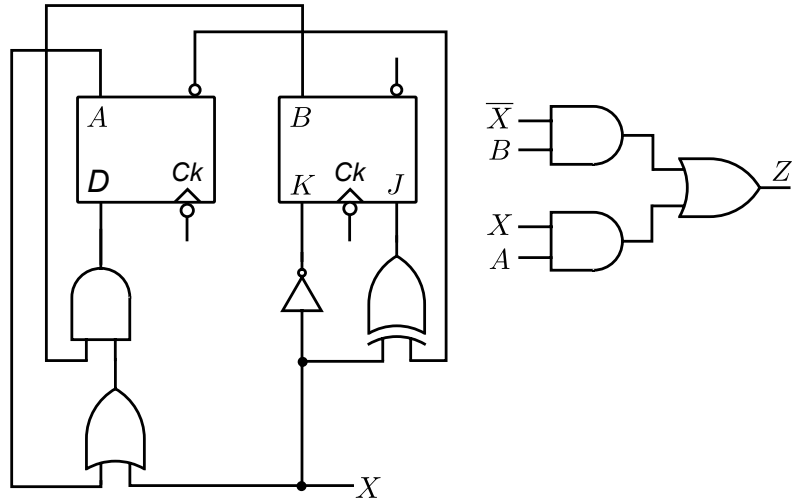
$A+$

AB \ X	0	1
00	1	
01		1
11		1
10		1

$B+$

AB \ X	0	1
00		
01	1	
11	1	1
10		1

Z

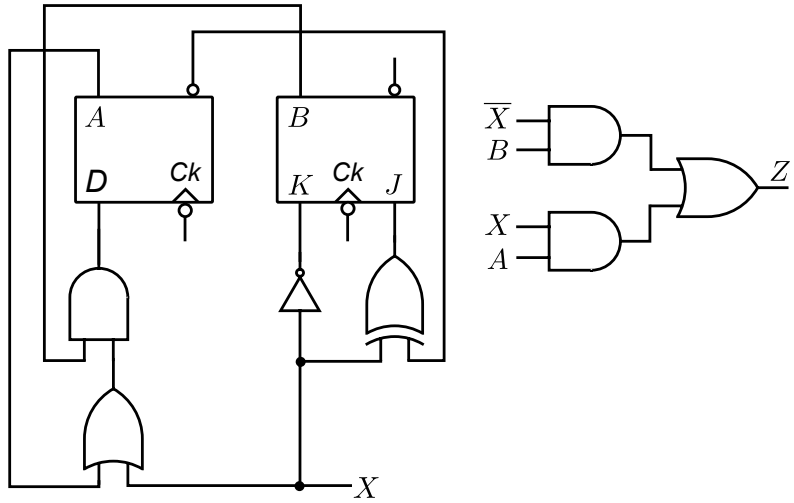


AB	A^+B^+ $X=0$	A^+B^+ $X=1$	output Z $X=0$	output Z $X=1$
00				
01				
11				
10				

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	output Z $X=0$	output Z $X=1$
S_0				
S_1				
S_2				
S_3				

Analysis of sequential networks

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
S_0	S_1	S_1	0	0
S_1	S_0	S_2	1	0
S_2	S_3	S_2	1	1
S_3	S_0	S_1	0	1



Next class...

- Sequence detector
- Please read Lecture 29

ECE 210
Introduction to Digital Logic Design

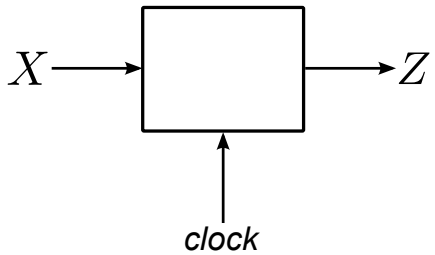
Lecture 29
Design of a Sequence Detector

Sequence detector

Single input X, single output Z sequential circuit

Design a network so that any input sequence ending in 101 will produce an output $Z = 1$ coincident with the last 1.

The network should not reset



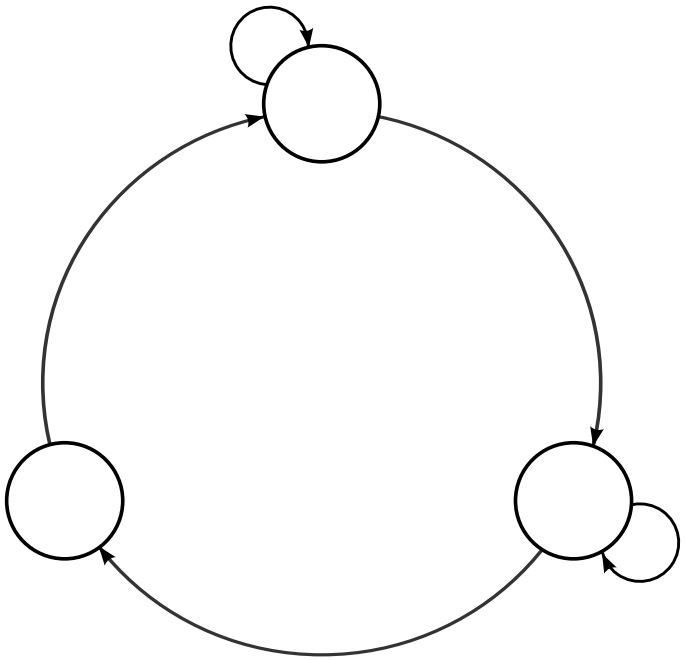
$X =$	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
$Z =$	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0

State graph

→ The circuit assumes Mealy network representation

→ Show the sequence of states and outputs in response to inputs

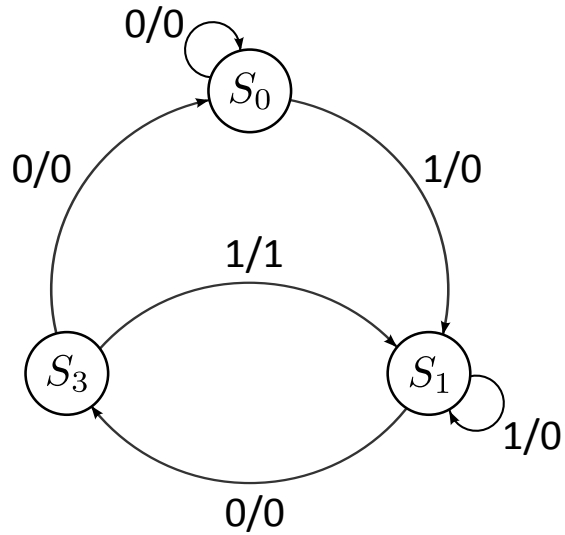
$$X = \quad 0011011001010100$$
$$Z = \quad 0000010000010100$$



Next state tables

	Next state		output Z	
AB	$X=0$	1	$X=0$	1
S_0				
S_1				
S_2				1

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
00			0	0
01			0	0
10			0	1



Now we are ready to design a circuit which has the behaviour described above

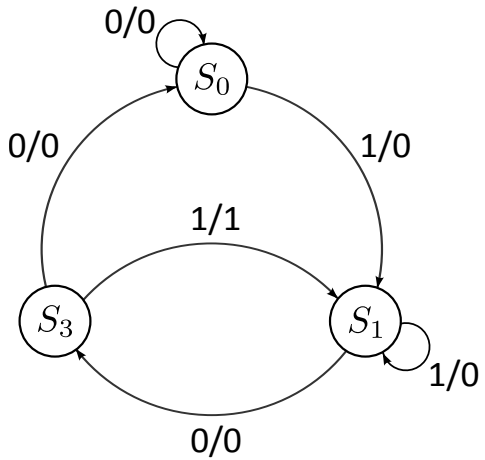
The system has 3 states: How many *FF* are needed?

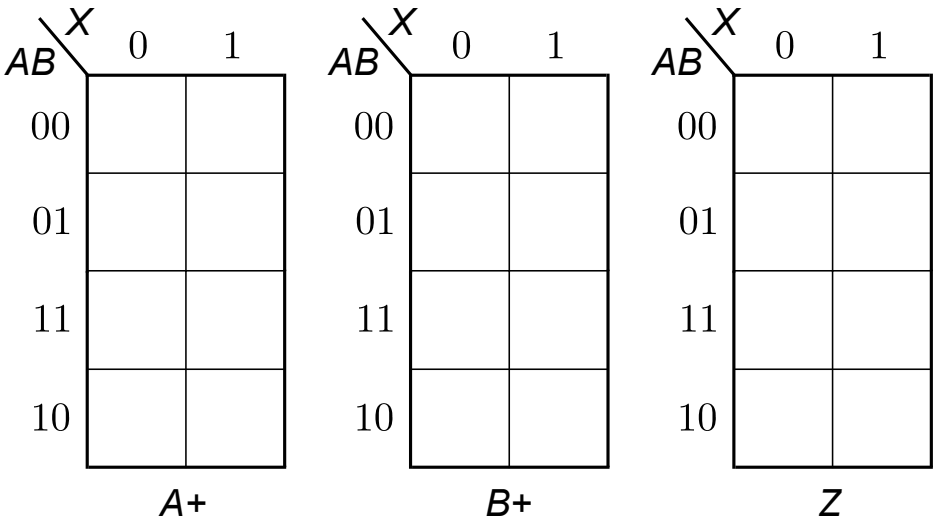
Next state tables

Next-state maps for each FF and the output function Z.

If $D - FF$ is used, $Q^+ = D$.

	A^+B^+		output Z	
AB	$X=0$	1	$X=0$	1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1



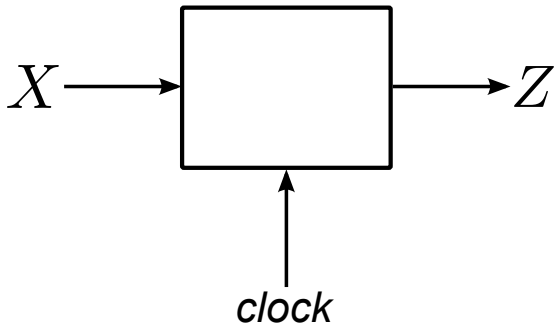


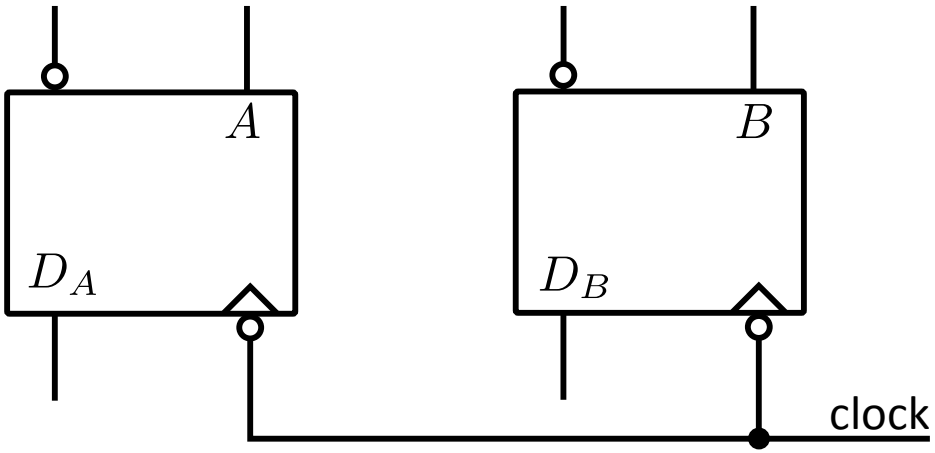
$A^+ =$

$B^+ =$

$Z =$

Sequence detector circuit



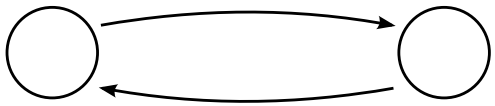


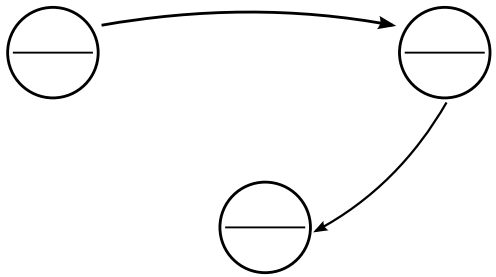
Sequence detector with Moore machine

Let us now rework the previous example as a Moore machine.

The circuit should output 1 only if the input 101 has occurred.

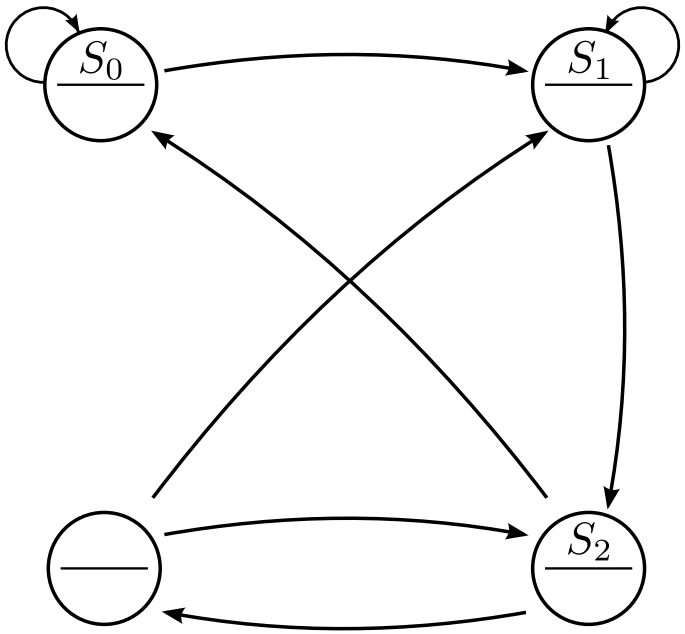
Moore *vs* Mealy state representation





State graph - Moore representation

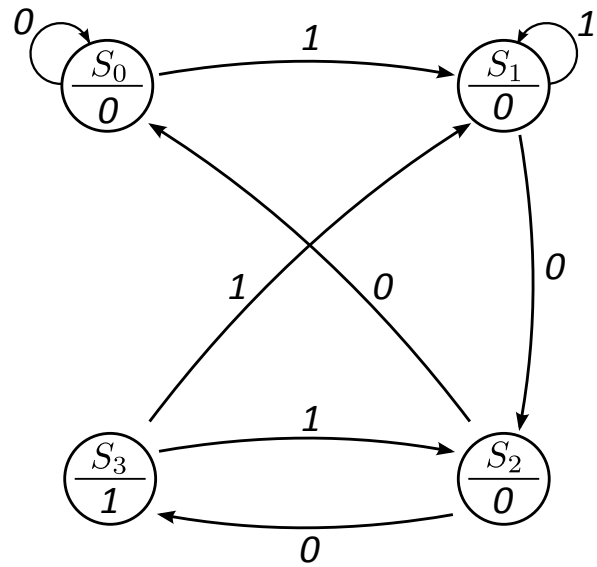
For a Moore machine, the output is written with the state instead of with the transition between states.



State tables

	Next state		output
AB	$X=0$	1	Z
S_0			0
S_1			0
S_2			0
S_3			1

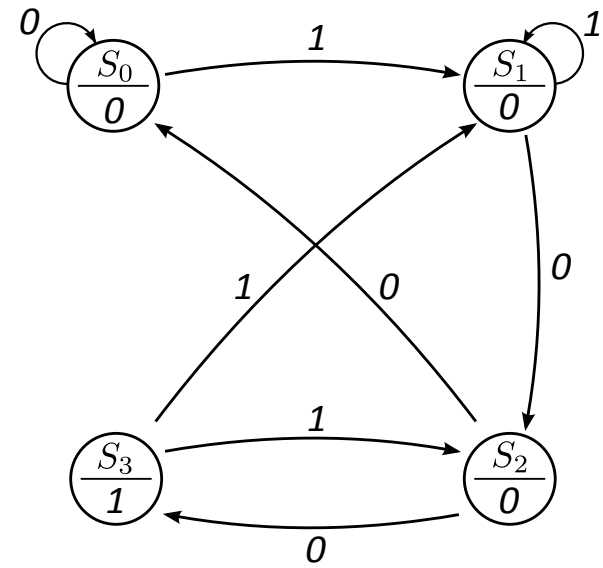
	A^+B^+		output
AB	$X=0$	1	Z
00			0
01			0
11			0
10			1

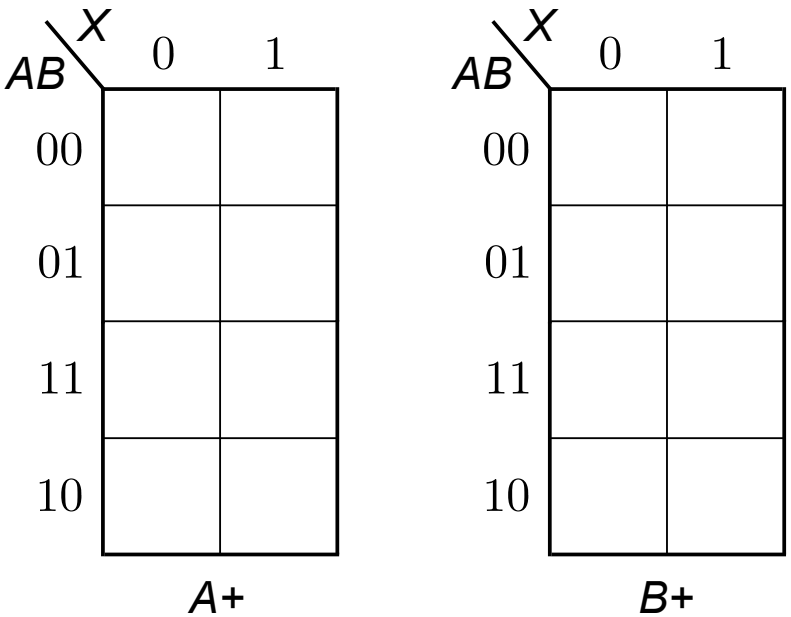


State maps

	A^+B^+		output
AB	$X=0$	1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

If $D-FF$ is used, $Q^+ = D$.



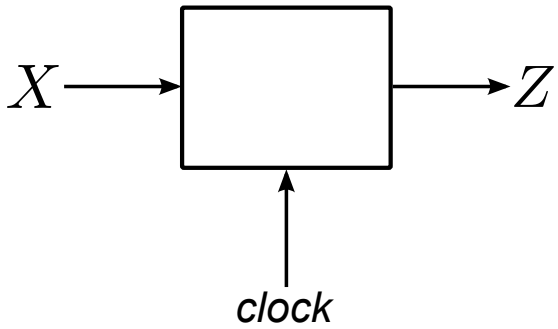


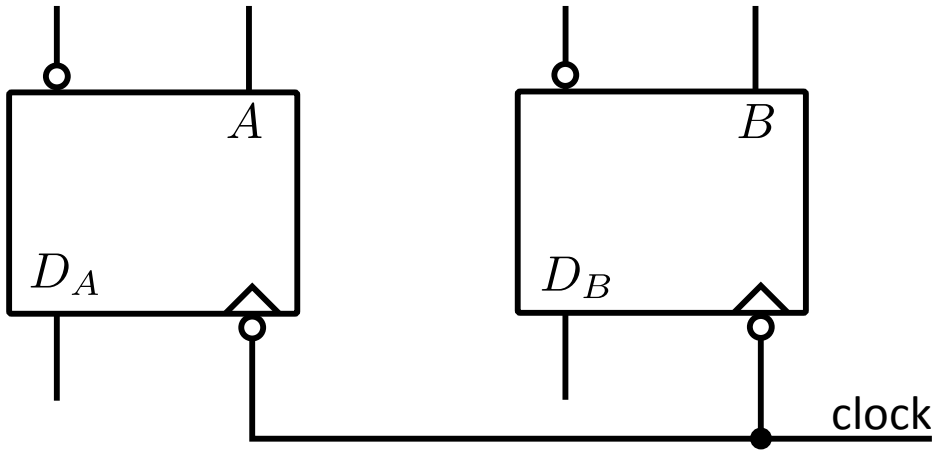
$A^+ =$

$B^+ =$

$Z \ =$

Sequence detector circuit - Moore machine





Rework the example using $JK - FF$

If $JK - FF$ is used, $Q^+ = J\overline{Q} + \overline{K}Q$.

	A^+B^+		output
AB	$X=0$	1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

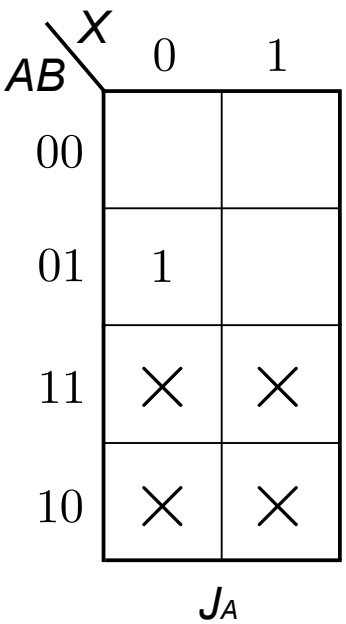
$JK - FF$ transition table

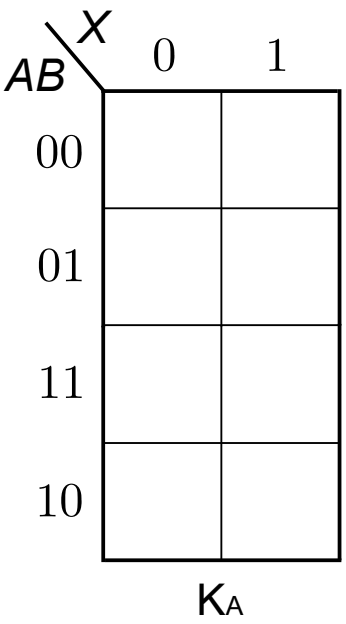
Q	Q^+	J	K
0	0	0	$\times$
0	1	1	$\times$
1	0	$\times$	1
1	1	$\times$	0

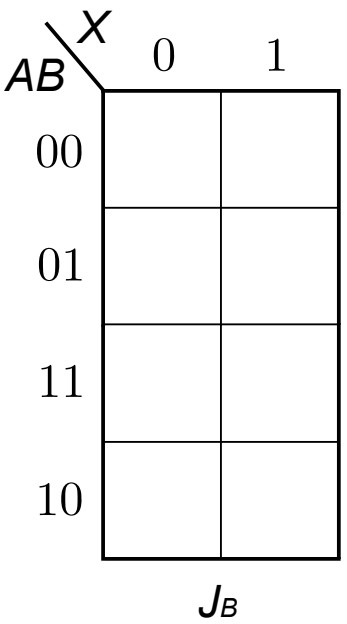
	A^+B^+			
	$X=0$		$X=1$	
AB	J_A K_A	J_B K_B	J_A K_A	J_B K_B
00				
01				
11				
10				

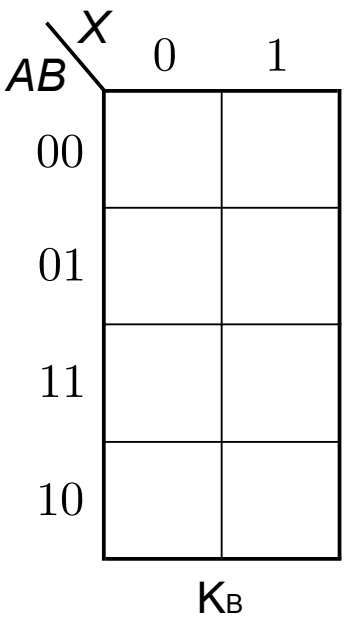
Transition maps - $JK - FF$

	A^+B^+			
	$X=0$		$X=1$	
AB	J_A K_A	J_B K_B	J_A K_A	J_B K_B
00	0 ×	0 ×	0 ×	1 ×
01	1 ×	× 0	0 ×	× 0
11	× 1	× 1	× 0	× 1
10	× 0	1 ×	× 1	1 ×



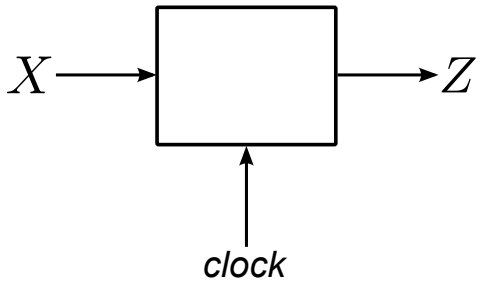


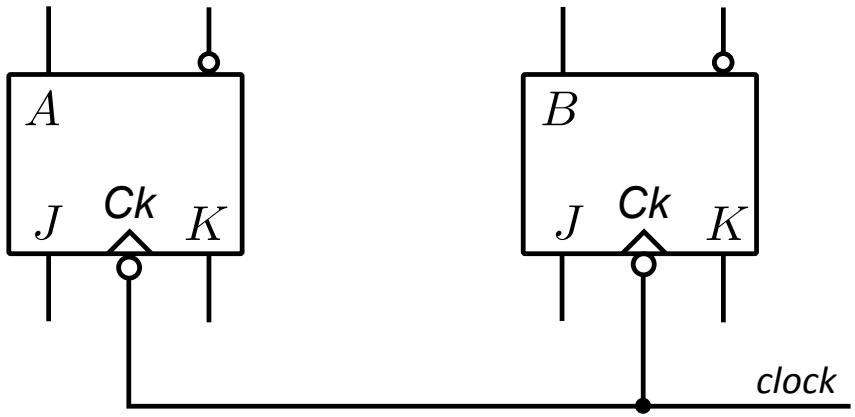




Sequence detector circuit - Moore machine

$$J_A = \overline{X}B \qquad K_A = X \oplus B$$
$$J_B = X + A \qquad K_B = A$$
$$Z = A\overline{B}$$





Next class...

- Multiple input/output networks
- Please read Lecture 30

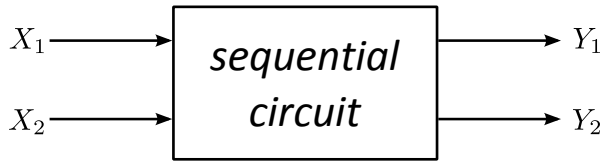
ECE 210
Introduction to Digital Logic Design

Lecture 30
Multiple Inputs and Outputs

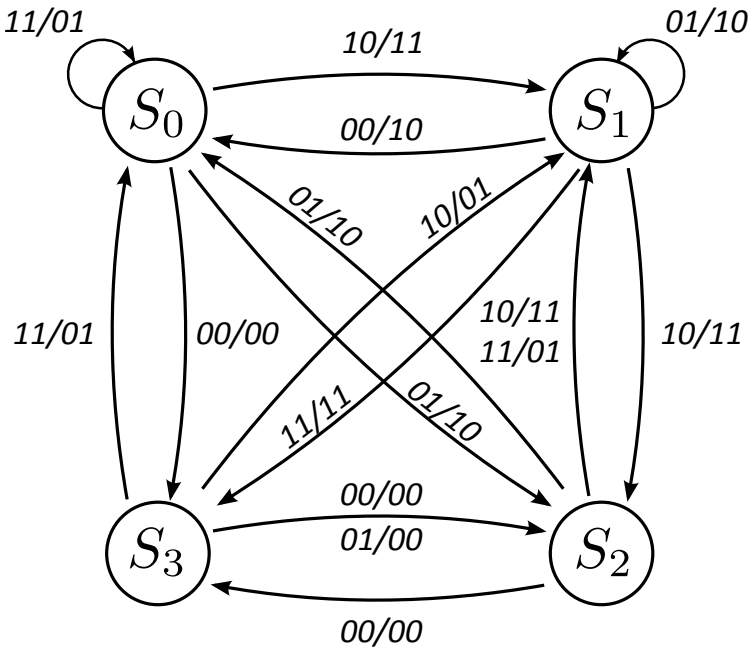
Multiple Inputs and Outputs

Sequential circuits may have multiple inputs and outputs

Consider as an example the following circuit



whose state graph is



Multiple Inputs and Outputs

Moore machine $\rightarrow$ State table has single output column

Mealy machine $\rightarrow$ State table has multiple output columns

State	next state				present output			
	$X_0X_1 = 00$	01	10	11	$X_0X_1 = 00$	01	10	11
S_0	S_3	S_2	S_1	S_0	00	10	11	01
S_1	S_0	S_1	S_2	S_3	10	10	11	11
S_2	S_3	S_0	S_1	S_1	00	10	11	01
S_3	S_2	S_2	S_1	S_0	00	00	01	01

Multiple Inputs and Outputs

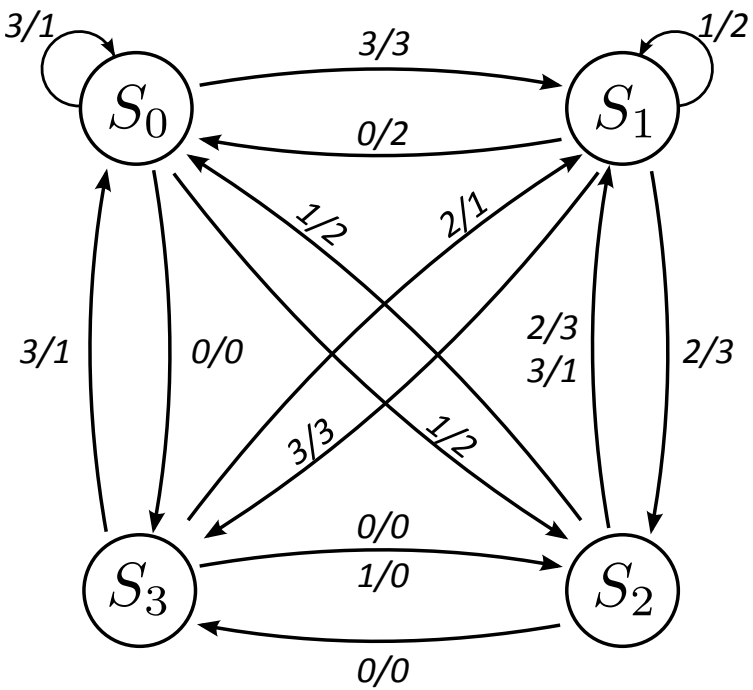
State	next state $X_0X_1 = 00$	01	10	11	present output $X_0X_1 = 00$	01	10	11
S_0	S_3	S_2	S_1	S_0	00	10	11	01
S_1	S_0	S_1	S_2	S_3	10	10	11	11
S_2	S_3	S_0	S_1	S_1	00	10	11	01
S_3	S_2	S_2	S_1	S_0	00	00	01	01

The state table can be simplified by using decimal inputs and outputs

State	next state $X_0X_1 = 00$	01	10	11	present output $X_0X_1 = 00$	01	10	11
S_0	S_3	S_2	S_1	S_0	0	2	3	1
S_1	S_0	S_1	S_2	S_3	2	2	3	3
S_2	S_3	S_0	S_1	S_1	0	2	3	1
S_3	S_2	S_2	S_1	S_0	0	0	1	1

Multiple Inputs and Outputs

State	next state $X_0X_1 = 00$	01	10	11	present output $X_0X_1 = 00$	01	10	11
S_0	S_3	S_2	S_1	S_0	0	2	3	1
S_1	S_0	S_1	S_2	S_3	2	2	3	3
S_2	S_3	S_0	S_1	S_1	0	2	3	1
S_3	S_2	S_2	S_1	S_0	0	0	1	1



Example: Traffic light control

Design a circuit to control a pedestrian crosswalk traffic light.

Default state: green signal for cars

If "walk button" is pressed:

$\rightarrow$ Signal turns yellow,

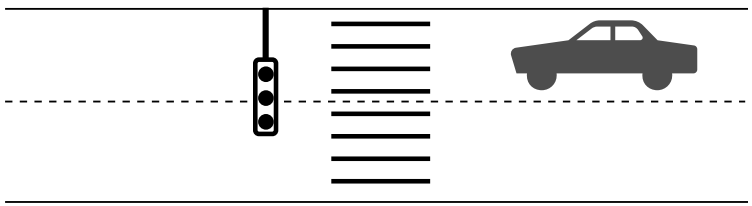
$\rightarrow$ Hold yellow signal for 5 seconds

$\rightarrow$ Signal turns red

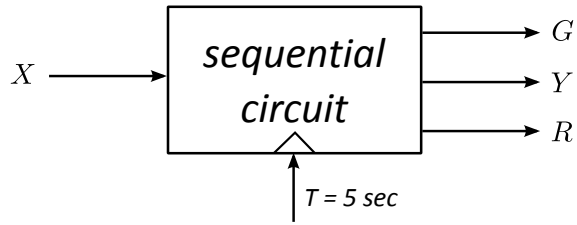
$\rightarrow$ Hold red for 15 seconds

$\rightarrow$ Return to green

Reset 15s timer whenever walk button is pressed



Example - Continued



Circuit inputs

$\rightarrow$ walk button signal X.

Outputs: 3 traffic light outputs:

$\rightarrow$ green (G),

$\rightarrow$ yellow (Y),

$\rightarrow$ red (R).

Assume that a clock signal with a period of 5 seconds is available.

How many states are needed?

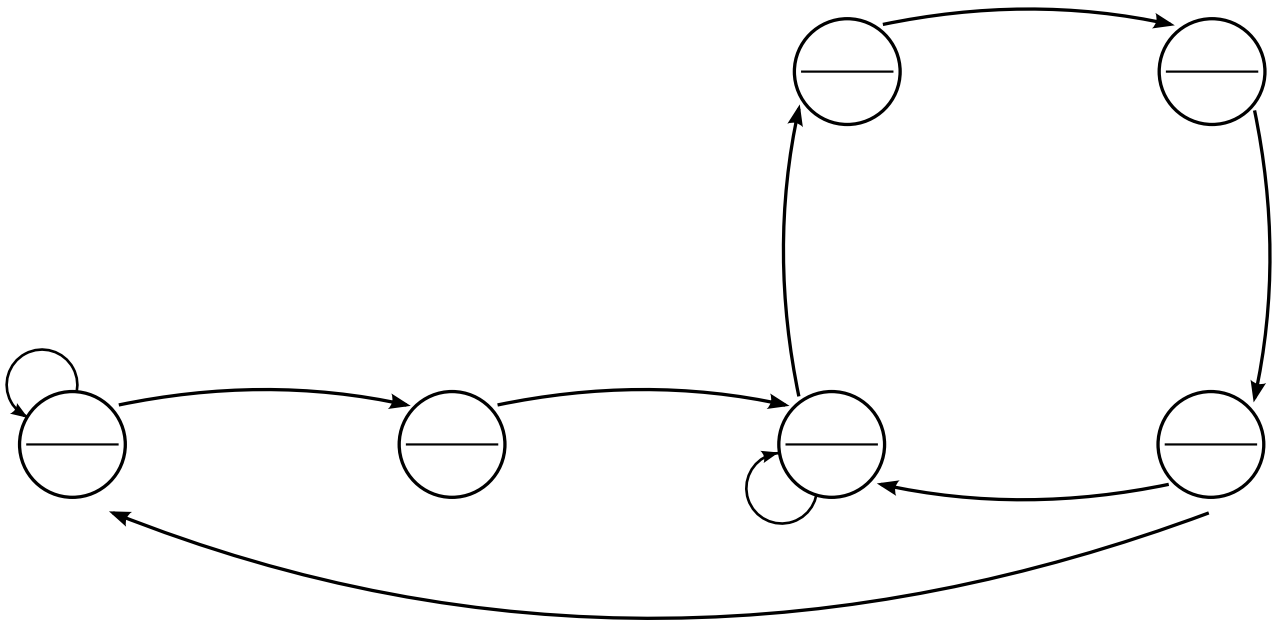
State graph

Input: Signal X.

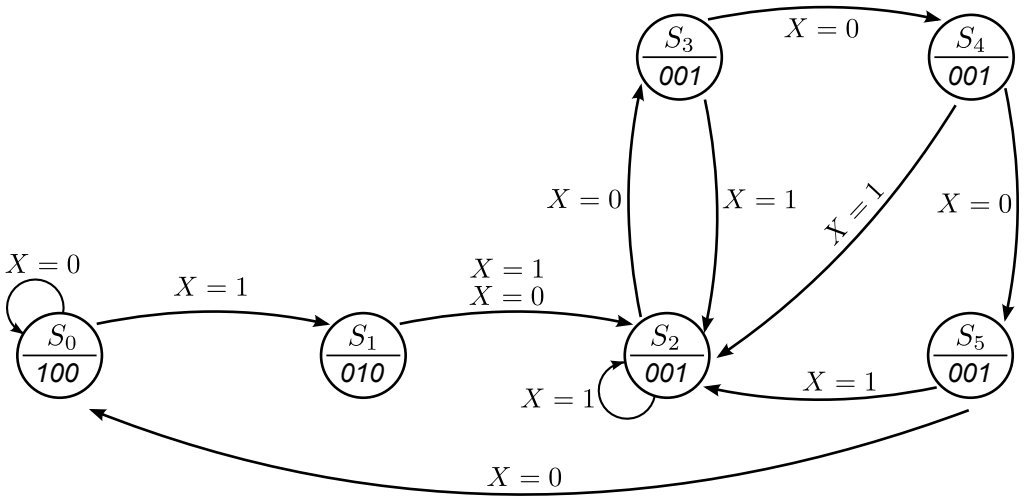
Output: GYR

Clock signal period: 5 seconds.

Reset 15s timer whenever $X = 1$.



State table



State	Next state $X = 0$	Next state $X = 1$	output $G\ Y\ R$
S_0	S_0	S_1	1 0 0
S_1	S_2	S_2	0 1 0
S_2	S_3	S_2	0 0 1
S_3	S_4	S_2	0 0 1
S_4	S_5	S_2	0 0 1
S_5	S_0	S_2	0 0 1

State table

State	Next state $X=0$	Next state $X=1$	output $G\ Y\ R$
S_0	S_0	S_1	1 0 0
S_1	S_2	S_2	0 1 0
S_2	S_3	S_2	0 0 1
S_3	S_4	S_2	0 0 1
S_4	S_5	S_2	0 0 1
S_5	S_0	S_2	0 0 1

ABC	$A^+B^+C^+$ $X=0$	$A^+B^+C^+$ $X=1$	output $G\ Y\ R$
0 0 0	0 0 0	0 0 1	1 0 0
0 0 1	0 1 0	0 1 0	0 1 0
0 1 0	0 1 1	0 1 0	0 0 1
0 1 1	1 0 0	0 1 0	0 0 1
1 0 0	1 0 1	0 1 0	0 0 1
1 0 1	0 0 0	0 1 0	0 0 1
1 1 0	× × ×	× × ×	× × ×
1 1 1	× × ×	× × ×	× × ×

State maps

If T FF's are used, $Q^+ = T \oplus Q$.

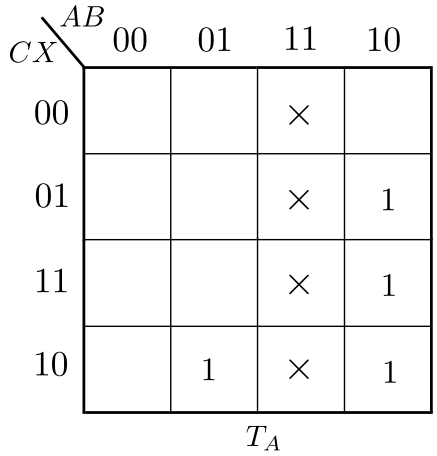
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

	$A^+B^+C^+$		output
ABC	$X=0$	$X=1$	$G\ Y\ R$
0 0 0	0 0 0	0 0 1	1 0 0
0 0 1	0 1 0	0 1 0	0 1 0
0 1 0	0 1 1	0 1 0	0 0 1
0 1 1	1 0 0	0 1 0	0 0 1
1 0 0	1 0 1	0 1 0	0 0 1
1 0 1	0 0 0	0 1 0	0 0 1
1 1 0	× × ×	× × ×	× × ×
1 1 1	× × ×	× × ×	× × ×

	$A^+B^+C^+$		output
ABC	$X=0$	$X=1$	$G\ Y\ R$
	$T_A\,T_B\,T_C$	$T_A\,T_B\,T_C$	
0 0 0	0 0 0	0 0 1	1 0 0
0 0 1	0 1 1	0 1 1	0 1 0
0 1 0	0 0 1	0 0 0	0 0 1
0 1 1	1 1 1	0 0 1	0 0 1
1 0 0			0 0 1
1 0 1			0 0 1
1 1 0	× × ×	× × ×	× × ×
1 1 1	× × ×	× × ×	× × ×

State K-maps

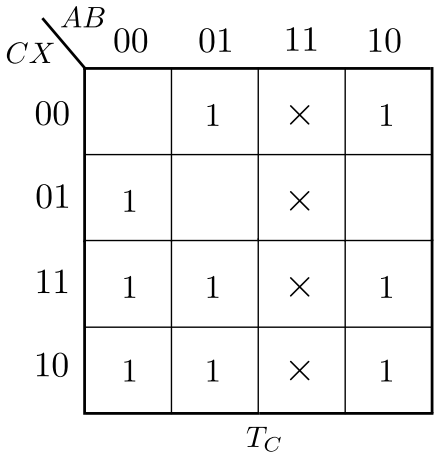
ABC	$A^+B^+C^+$ $X=0$	$A^+B^+C^+$ $X=1$	output $G\ Y\ R$
	$T_A\,T_B\,T_C$	$T_A\,T_B\,T_C$	
0 0 0	0 0 0	0 0 1	1 0 0
0 0 1	0 1 1	0 1 1	0 1 0
0 1 0	0 0 1	0 0 0	0 0 1
0 1 1	1 1 1	0 0 1	0 0 1
1 0 0	0 0 1	1 1 0	0 0 1
1 0 1	1 0 1	1 1 1	0 0 1
1 1 0	× × ×	× × ×	× × ×
1 1 1	× × ×	× × ×	× × ×

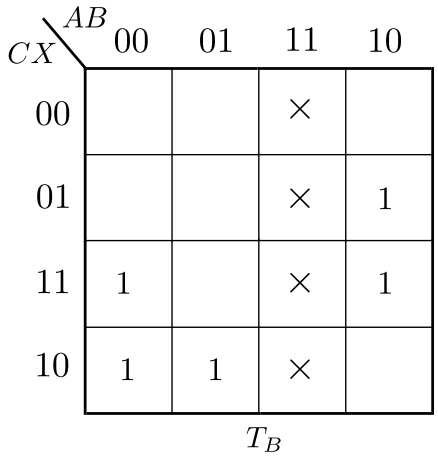


$$T_A = AX + AC + BC\overline{X}$$

$$T_B = AX + \overline{A}C\overline{X} + \overline{B}CX$$

$$T_C = C + A\overline{X} + B\overline{X} + \overline{A}\,\overline{B}X$$





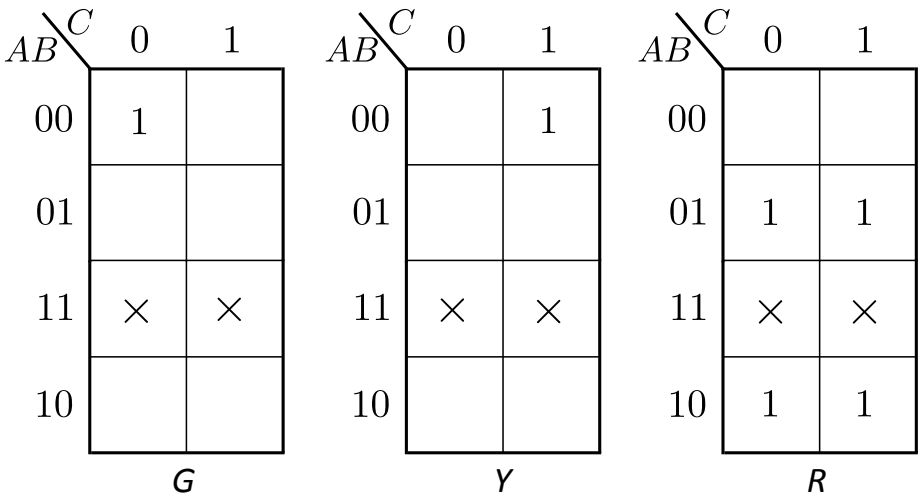
Output K-maps

ABC	$A^+B^+C^+$ $X=0$	$A^+B^+C^+$ $X=1$	output $G\ Y\ R$
	$T_A\,T_B\,T_C$	$T_A\,T_B\,T_C$	
0 0 0	0 0 0	0 0 1	1 0 0
0 0 1	0 1 1	0 1 1	0 1 0
0 1 0	0 0 1	0 0 0	0 0 1
0 1 1	1 1 1	0 0 1	0 0 1
1 0 0	0 0 1	1 1 0	0 0 1
1 0 1	1 0 1	1 1 1	0 0 1
1 1 0	× × ×	× × ×	× × ×
1 1 1	× × ×	× × ×	× × ×

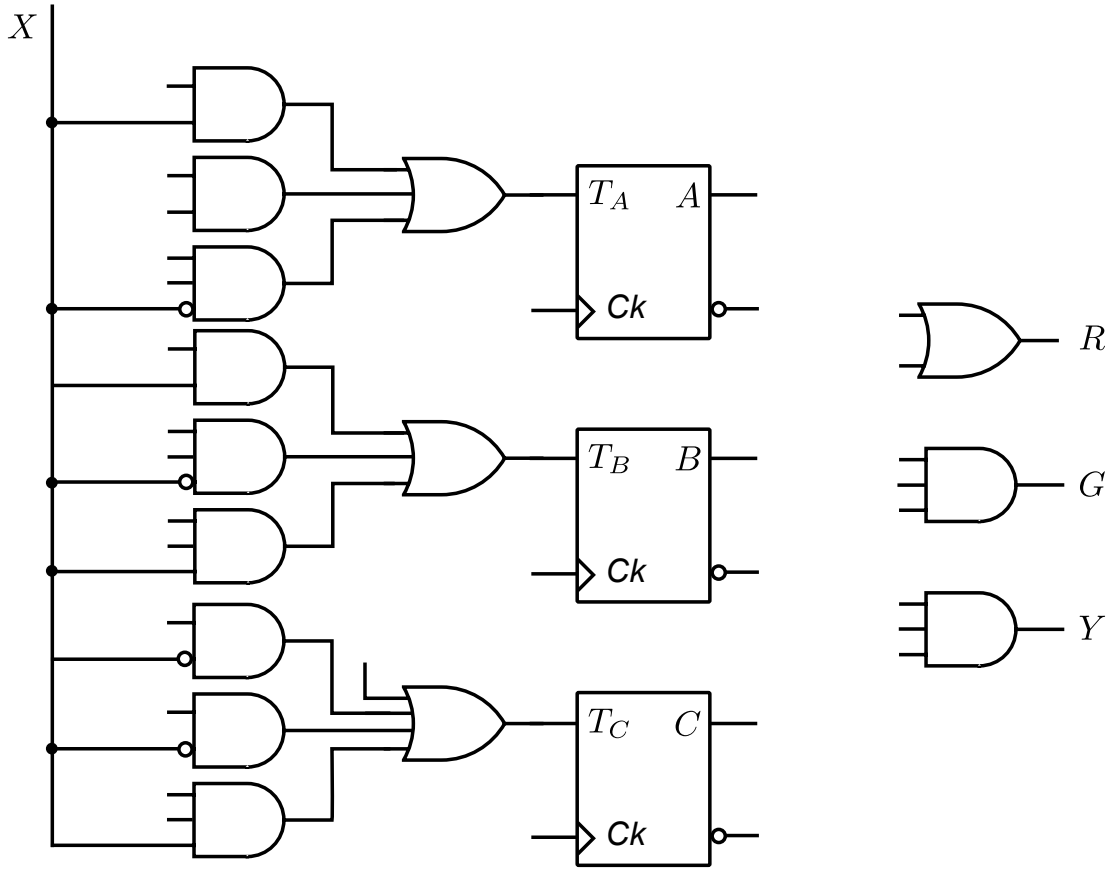
$$G = \overline{A}\,\overline{B}\,\overline{C}$$

$$Y = \overline{A}\,\overline{B}\,C$$

$$R = A + B$$



Traffic light circuit

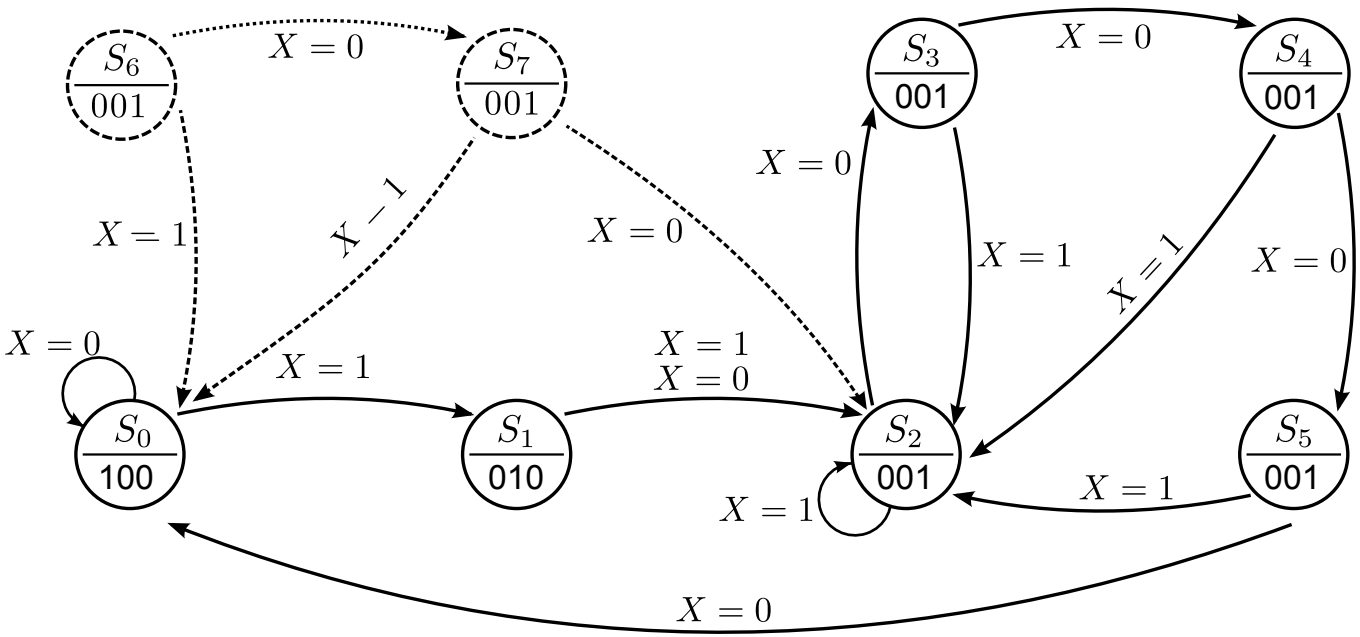


Check the don't care states.

State table

When the *FF*'s are turned on, their initial states may be unpredictable.

Don't care states: S_6 and S_7.



Example 3 - Complex sequence detector

A sequential circuit has one input (X) and two outputs (Z_1 and Z_2).

$Z_1 = 1$ occurs every time the input sequence 100 is completed, provided that the sequence 010 has never occurred,

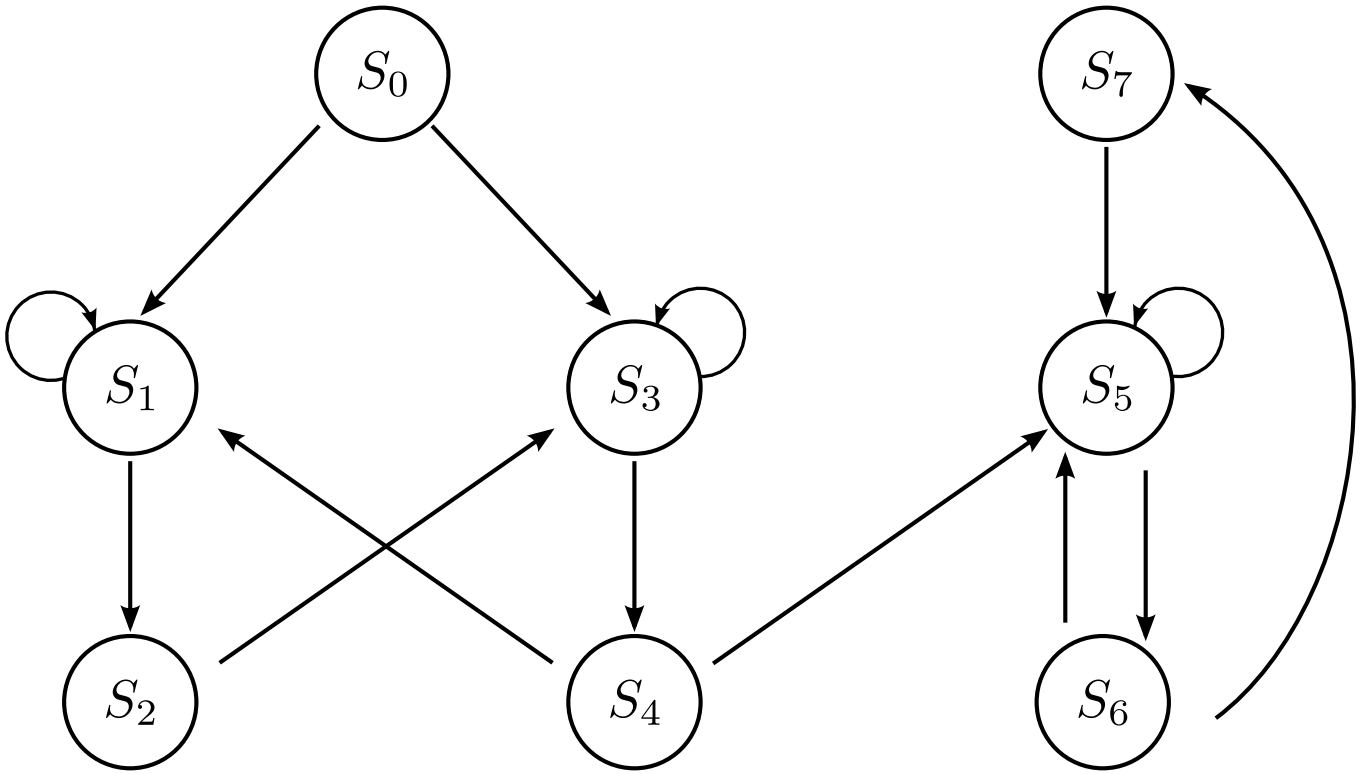
$Z_2 = 1$ occurs every time the sequence 010 is completed.

$$\begin{aligned} X = &\quad 100110010\ 1010010110100 \\ Z_1 = &\quad 001000100\ 0000000000000 \\ Z_2 = &\quad 000000001\ 0101001000010 \end{aligned}$$

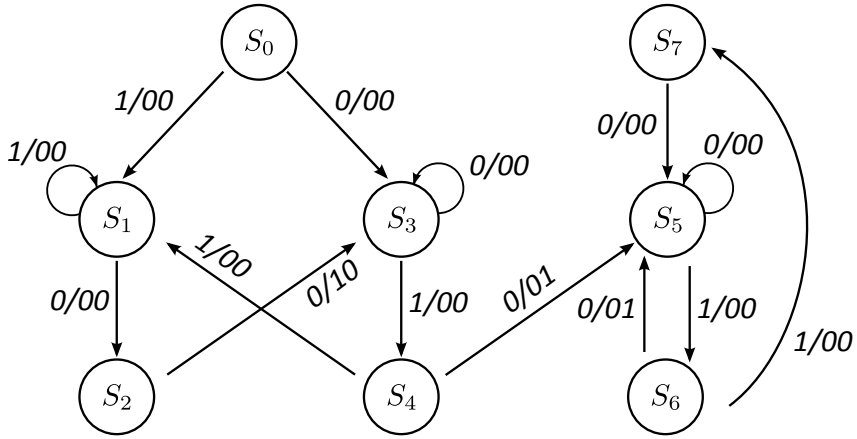
Example 3 - Complex sequence detector

$Z_1 = 1$ if 100 occurs, provided that 010 has never occurred,

$Z_2 = 1$ if sequence 010 occurs.



Example 3 - Complex sequence detector



State	Next state $X=0$	Next state $X=1$	Output Z_1Z_2 $X=0$	Output Z_1Z_2 $X=1$
S_0	S_3	S_1	00	00
S_1			00	00
S_2			10	00
S_3			00	00
S_4			01	00
S_5			00	00
S_6			01	00
S_7	S_5	S_7	00	00

Example 4

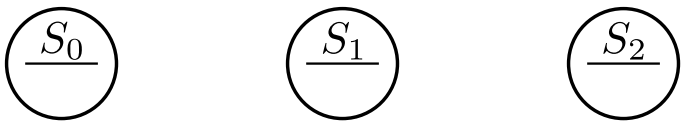
A Moore circuit should have an output of $Z = 1$ if

$\rightarrow$ The total number of 0's received is an even number greater than 0

$\rightarrow$ and provided that two consecutive 1's have never been received.

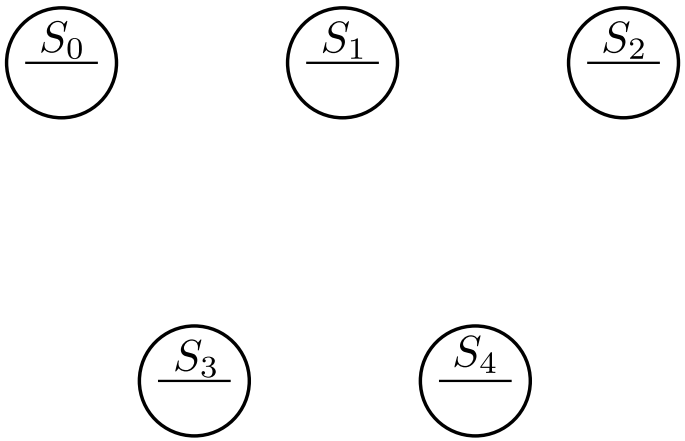
$$X = \quad 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0$$
$$Z =$$

To start, consider only 0 inputs. The graph should give 1 if the total number of 0 is even and greater than 0.



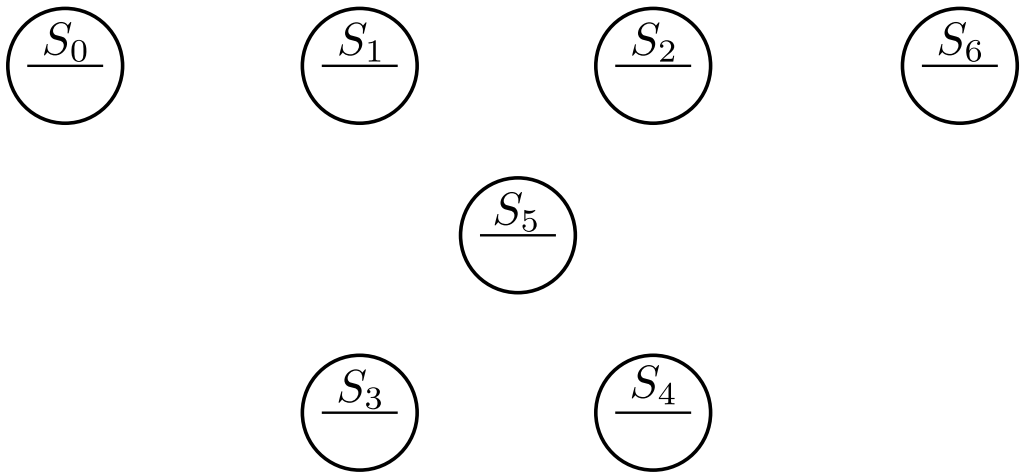
Example 4

Add states to the graph so that starting from S_0, if 2 consecutive 1's are received the output will remain 0.



Example 4

Now complete the graph so that each state has both a 0 and 1 arrow leading away from it.



Next class...

- Reduction of state tables state assignment
- Please read Lecture 31

ECE 210
Introduction to Digital Logic Design

Lecture 31
Minimizing Finite State Machines

Redundant States

The first step in designing a sequential circuit is to derive a sate table.

Before we realize the state table using FF, reduction of the sates table to a minimum number of states is desirable.

A circuit with 9 states needs 4 *FF*,

A circuit with 8 states needs *FF*,

A circuit with 6 states still needs *FF*, but we would introduce don't cares terms.

Redundant States

Design a sequential network that examines groups of 4 bits and produces an output $Z = 1$ if the input sequences 0101 or 1001 occur.

The network resets after 4 input values; typical sequences are:

$$\begin{array}{ll} X = & 0101 \quad 0010 \quad 1001 \quad 0101 \\ Z = & \end{array}$$

We will analyse the problem without being concern with introduction of too many states, and later we will reduce them.

Sequence detector - 0101 or 1001

Input sequence	Current state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
reset	S_0			0	0
0	S_1			0	0
1	S_2			0	0
00	S_3			0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1
011	S_{10}	S_0	S_0	0	0
100	S_{11}	S_0	S_0	0	1
101	S_{12}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Sequence detector

Eliminate redundant states using tow matching technique

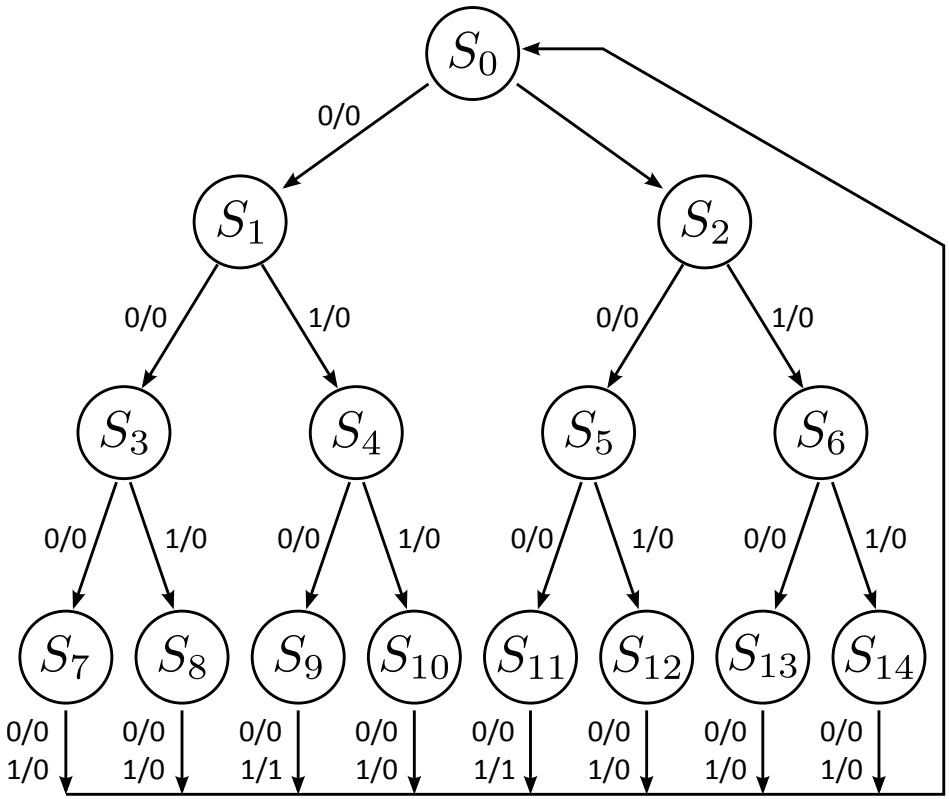
Ex: S_7 and S_8 are identical and can be merged

Input sequence	Current state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1
011	S_{10}	S_0	S_0	0	0
100	S_{11}	S_0	S_0	0	1
101	S_{12}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Sequence detector

Which states are equivalent ?

Input seq	S	S^+ 0	S^+ 1	Z 0	Z 1
reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1
011	S_{10}	S_0	S_0	0	0
100	S_{11}	S_0	S_0	0	1
101	S_{12}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0



Example - Continued

Reduced state table

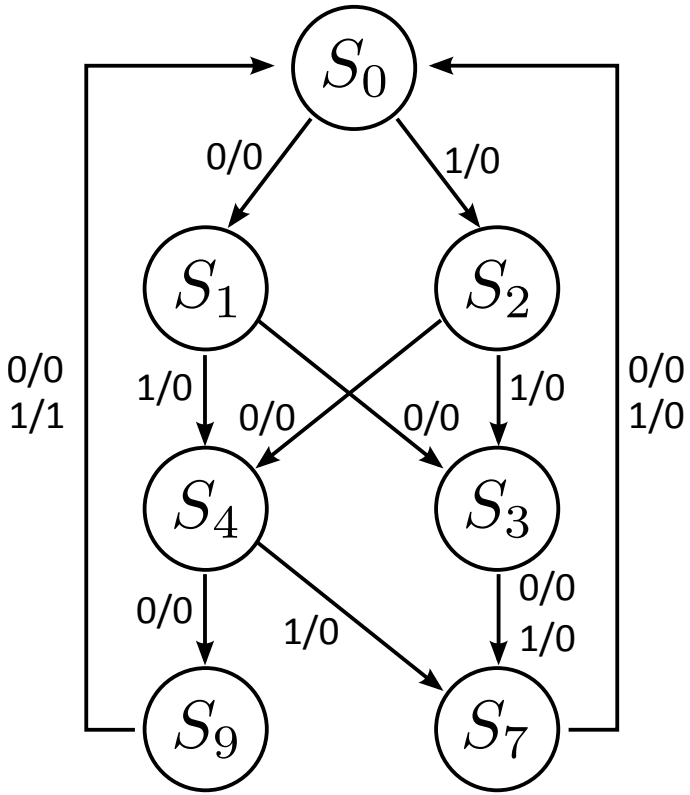
Can it still be simplified ?

Input seq	S	S^+		Z	
		0	1	0	1
reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_7	0	0
01	S_4	S_9	S_7	0	0
10	S_5	S_9	S_7	0	0
11	S_6	S_7	S_7	0	0
000	S_7	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1

$S_3 \equiv S_6$

$S_4 \equiv S_5$

State graph



Input seq	S	S^+ 0	S^+ 1	Z 0	Z 1
reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_4	S_3	0	0
00	S_3	S_7	S_7	0	0
01	S_4	S_9	S_7	0	0
000	S_7	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1

Equivalent states

Two states are equivalent if there is no way of telling them apart by observing their inputs and outputs.

$\rightarrow$ Row matching technique is not sufficient to find all equivalent states

$\rightarrow$ Consider the case when you can observe only the circuit inputs and outputs, not the state diagram.

$\rightarrow$ States are equivalent if for identical inputs they produce identical outputs

Equivalent states

Let $\underline{x} = x_0 x_1 ... x_n$ be an input sequence of arbitrary length n.

States p and q are equivalent if and only if

$$z_p = (p, x) = z_q = (q, x) \quad \forall \ \underline{x}$$

$\rightarrow z_p$ is an output sequence when starting in state p and applying $\underline{x}$

$\rightarrow z_q$ is an output sequence when starting in state q and applying $\underline{x}$

The following input sequences have to be tested:

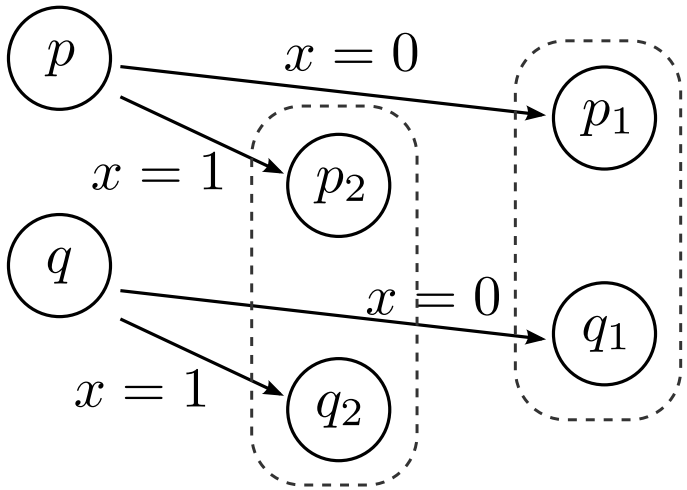
1. 0 and 1
2. 00, 01, 10, and 11
3. 000, 001, 010, 011, . . . , 111
4. 0000, 0001, . . .
5. etc., until the maximum length of the input

Equivalent states

Two states, p and q, are equivalent if for every input x, their **outputs are the same** and their **next states are equivalent**.

$z_p = (p, x) = z_q = (q, x)$ and next state of p for x is equal to next state of q for x

x is an input, and z_p and z_q are output generate by the circuit when applying x for states p and q respectively



If $p_1 \equiv q_1$ and $p_2 \equiv q_2$ then $p \equiv q$

Equivalent states

Two states, p and q, are equivalent if for every input x, their outputs are the same and their next states are equivalent.

Input seq	S	S^+ 0	S^+ 1	Z 0	Z 1
reset	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	1
011	S_{10}	S_0	S_0	0	0
100	S_{11}	S_0	S_0	0	1
101	S_{12}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

S_3 and S_4 are equivalent if $S_7 \equiv S_9$ and $S_8 \equiv S_{10}$. Since S_7 and S_9 have different outputs $\therefore$ $S_3 \not\equiv S_4$.

S_0 and S_1 are equivalent if $S_1 \equiv S_3$ and $S_2 \equiv S_4$ and

$S_2 \equiv S_4$ if $S_5 \equiv S_9$ and $S_6 \equiv S_{10}$.
Since $S_5 \not\equiv S_9$ $\therefore$ $S_0 \not\equiv S_1$.

Implication table

Equivalent states can be determined using an implication table

Aim: Find equivalent states by checking each pair of states

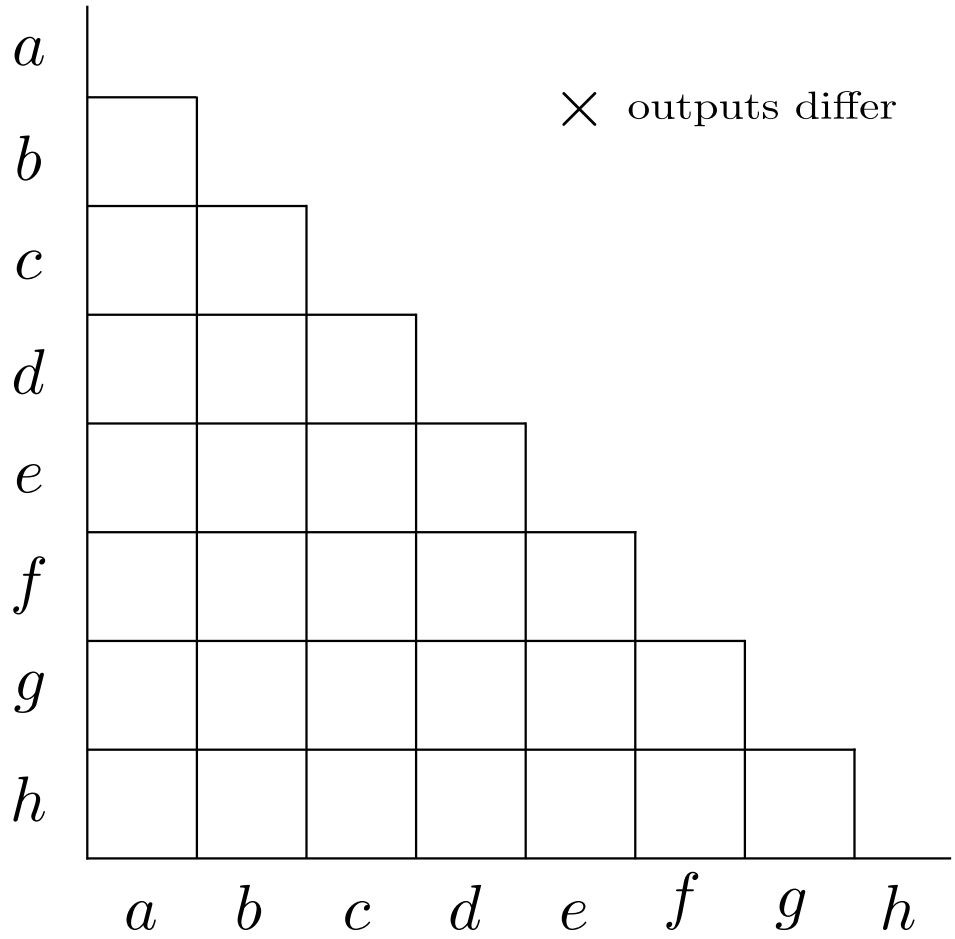
$\rightarrow$ Non-equivalent pairs of states are found and eliminated,

$\rightarrow$ Only equivalent pairs of states remain

Present state	Next state $x = 0$	Next state $x = 1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

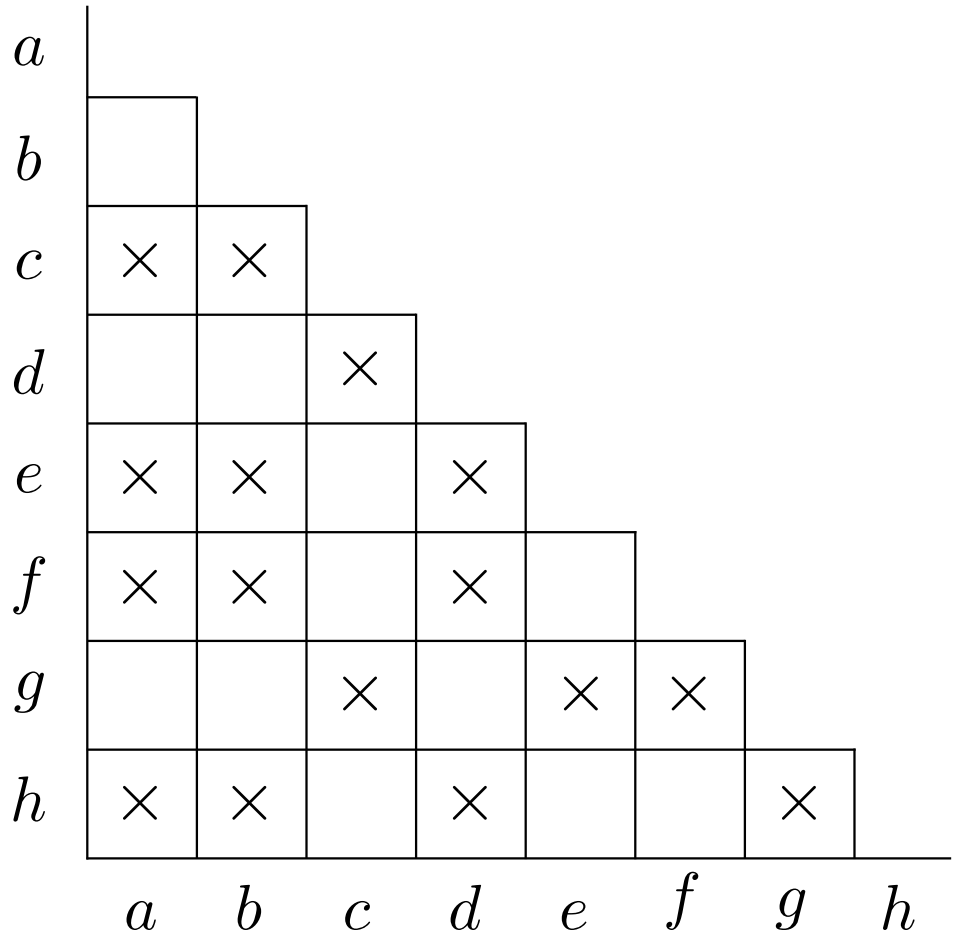
Implication table

Present state	Next state $x = 0$	Next state $x = 1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



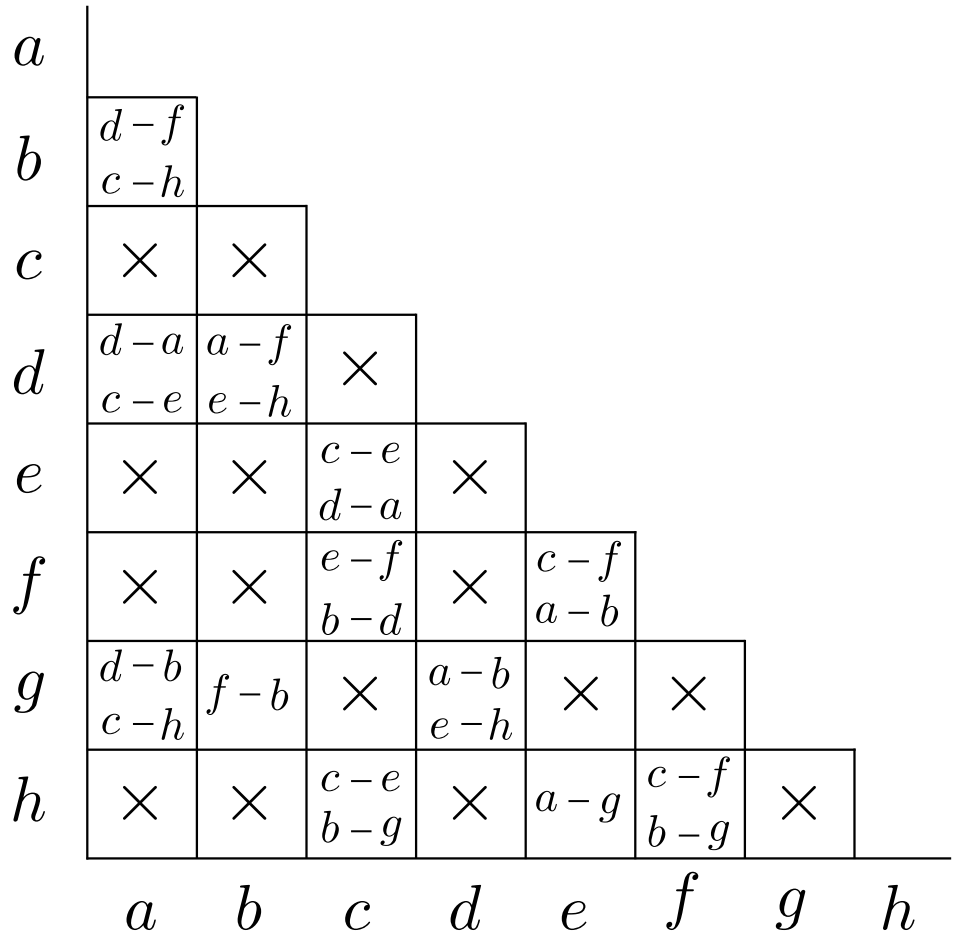
Implication table

Present state	Next state $x = 0$	Next state $x = 1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



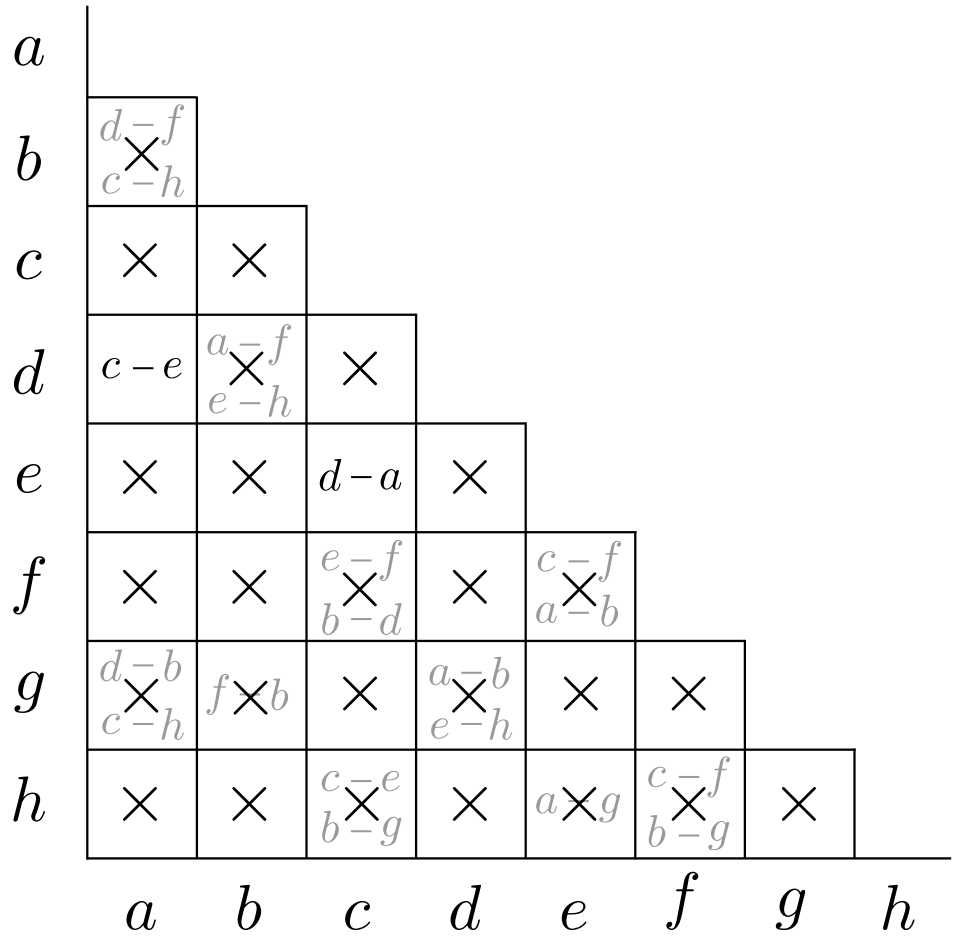
Implication table

Present state	Next state $x=0$	Next state $x=1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Implication table

Present state	Next state $x=0$	Next state $x=1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



$a \equiv d$ and $c \equiv e$

Equivalent States

Use equivalent states to modify the original state table

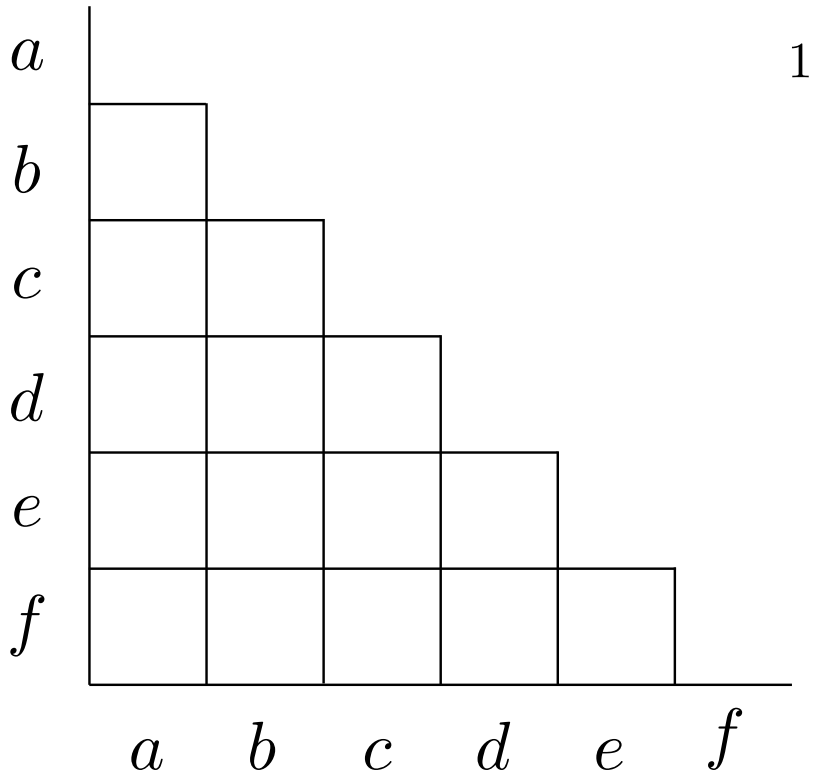
$a \equiv d$ and $c \equiv e$

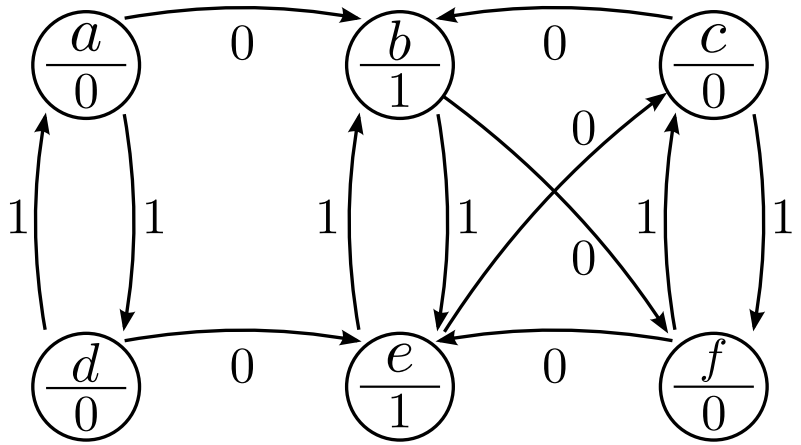
Present state	Next state $x = 0$	Next state $x = 1$	Present output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

Present state	Next state $x = 0$	Next state $x = 1$	Present output
ad	ad	ce	0
b	f	h	0
ce	ce	ad	1
f	f	b	1
g	b	h	0
h	ce	g	1

Example 2

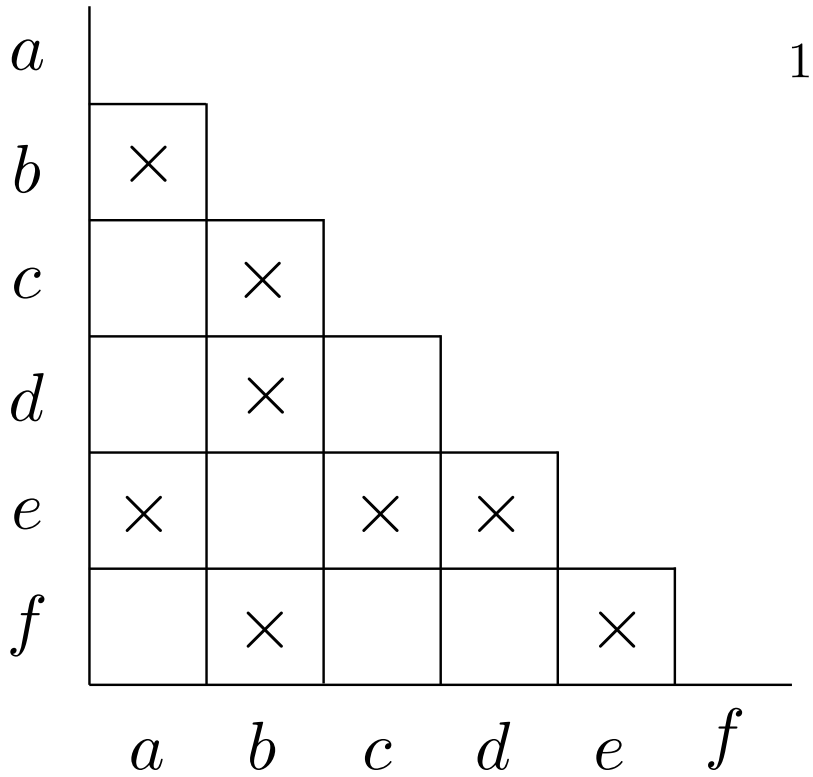
Reduce the state graph to a minimum number of states

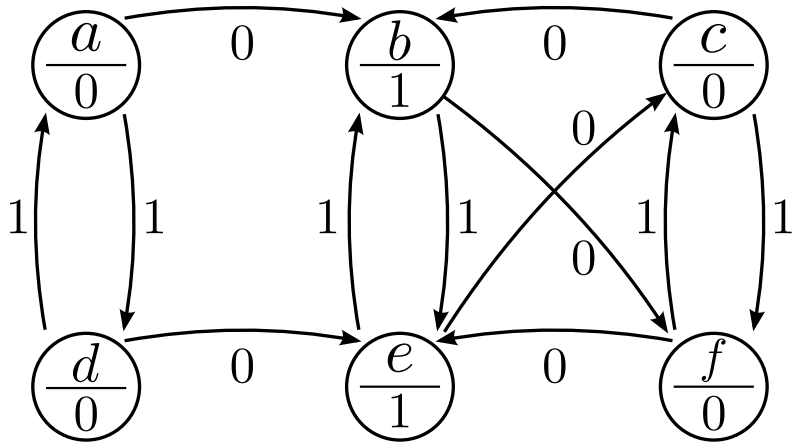




Example 2 - Continued

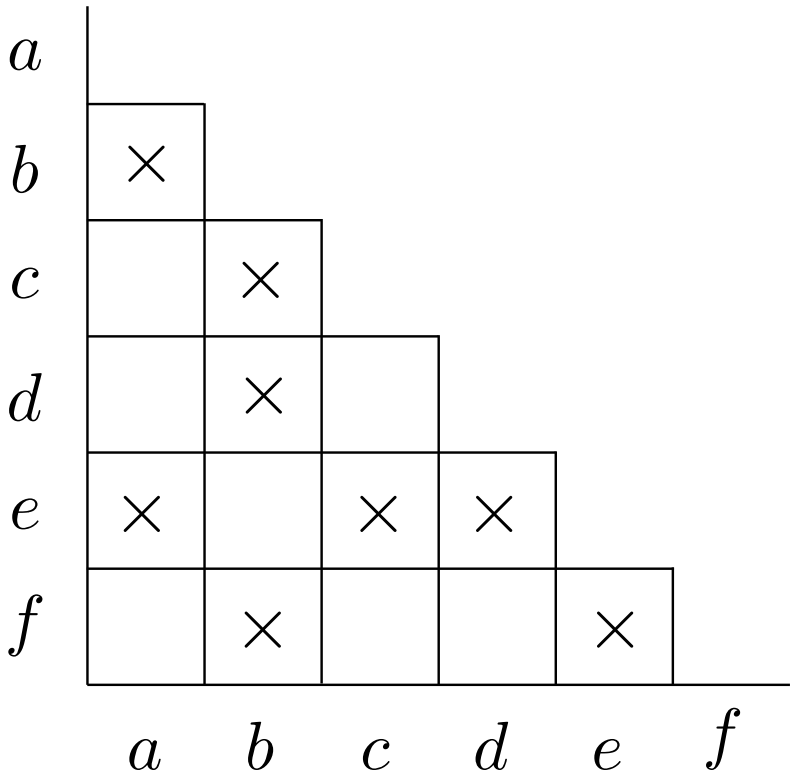
Reduce the state graph to a minimum number of states

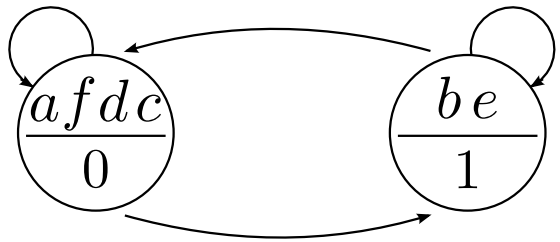




Example 2 - Continued

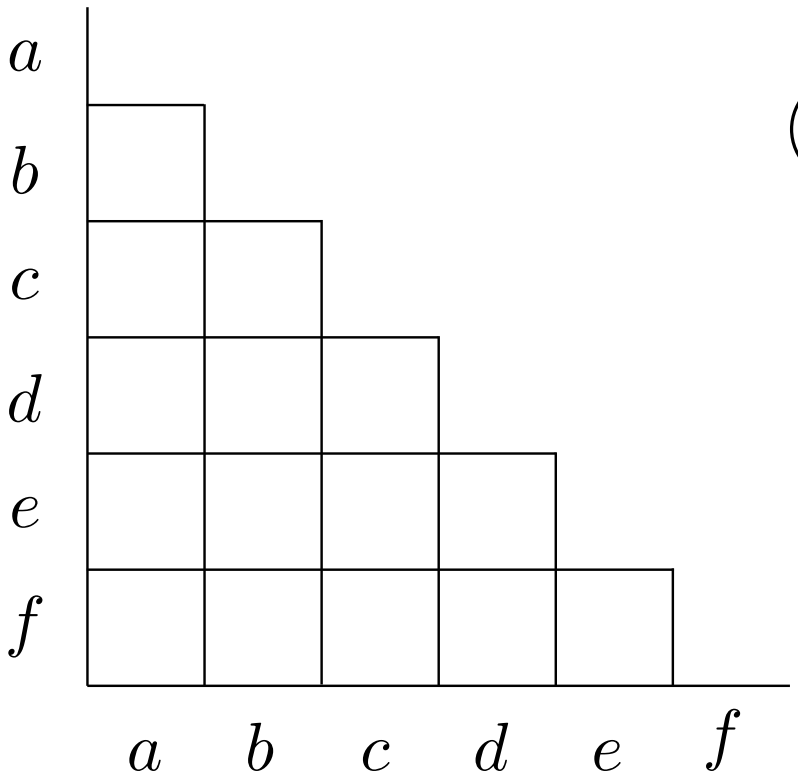
Complete the simplified state graph

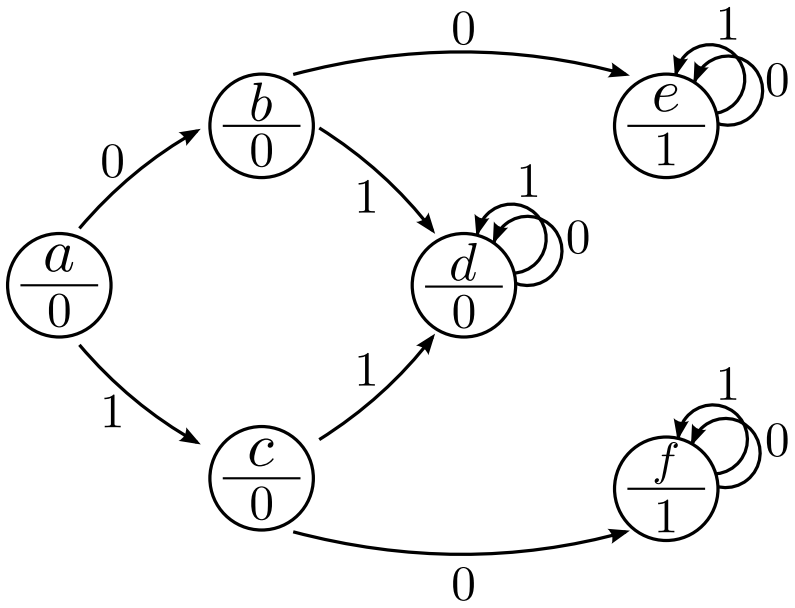




Example 3

Reduce the state graph to a minimum number of states

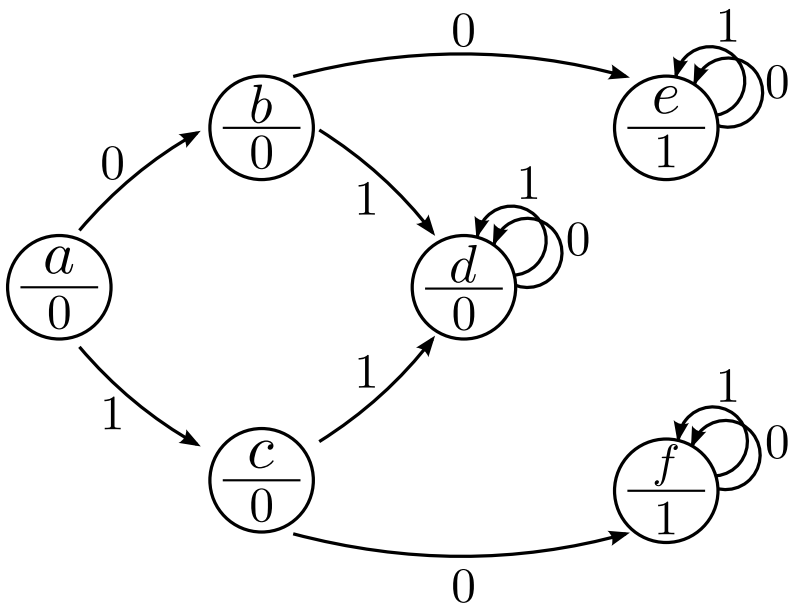




Example 3 - Continued

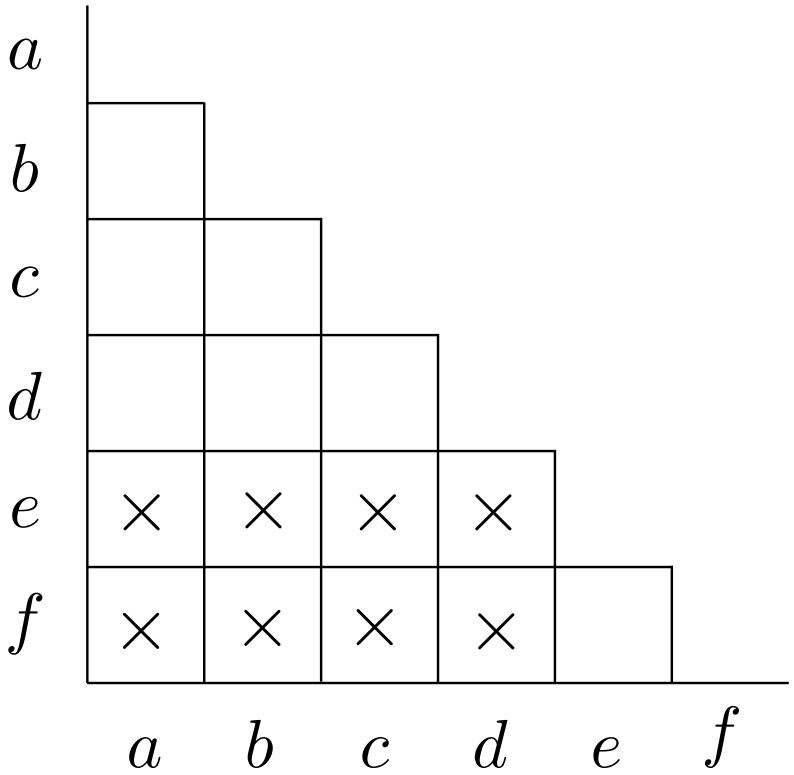
Reduce the state graph to a minimum number of states

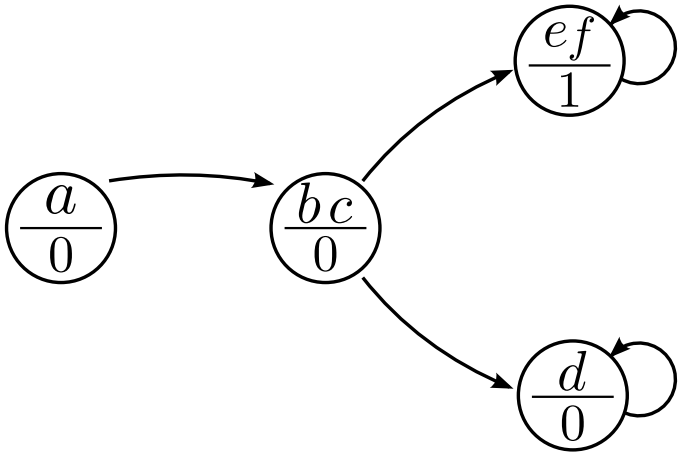
	a	b	c	d	e	f
a						
b						
c						
d						
e	×	×	×	×		
f	×	×	×	×		



Example 3 - Continued

Complete the simplified state graph





Next class...

- Serial code converter
- Please read Lecture 32

ECE 210
Introduction to Digital Logic Design

Lecture 32
Serial Code Converter

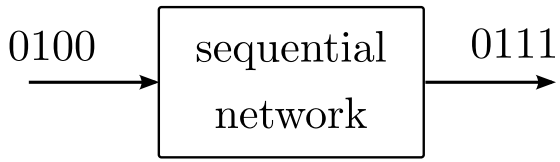
Design Example

We will design a sequential *BCD* to Excess-3 code converter.

4-bits numbers will be converted sequentially

Conversion will be performed bit-by-bit starting with the LSB

decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

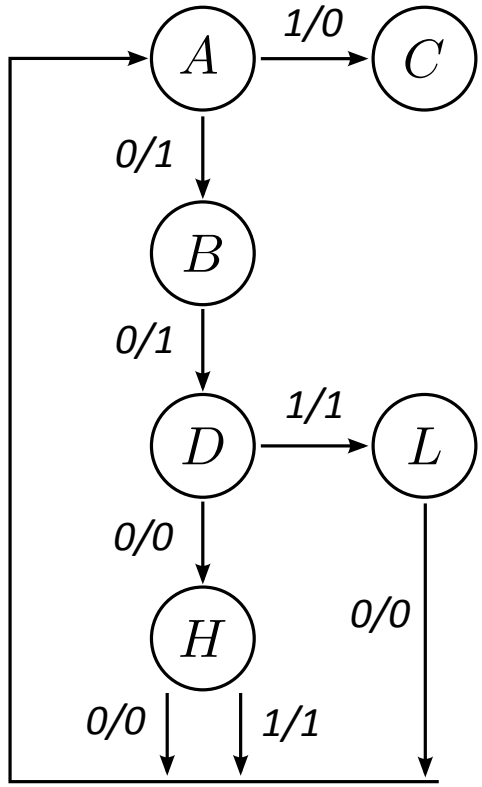


Bit-by-bit conversion example

→ Consider the input 0000

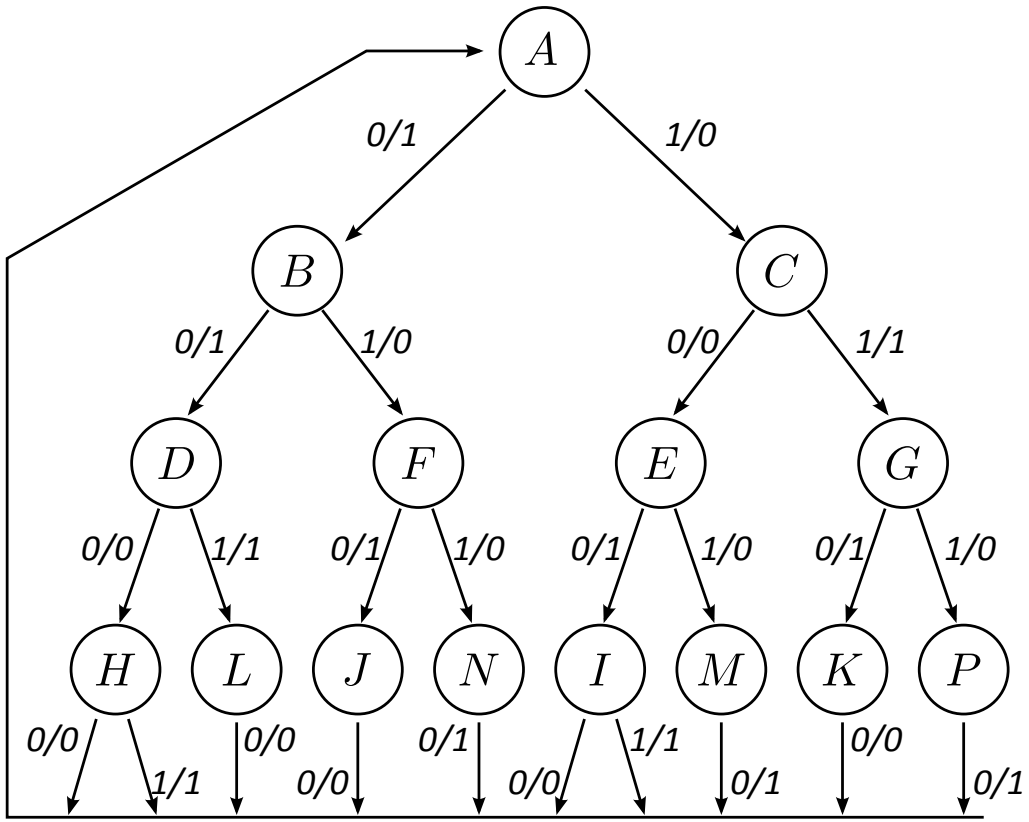
→ A is the reset state

decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100



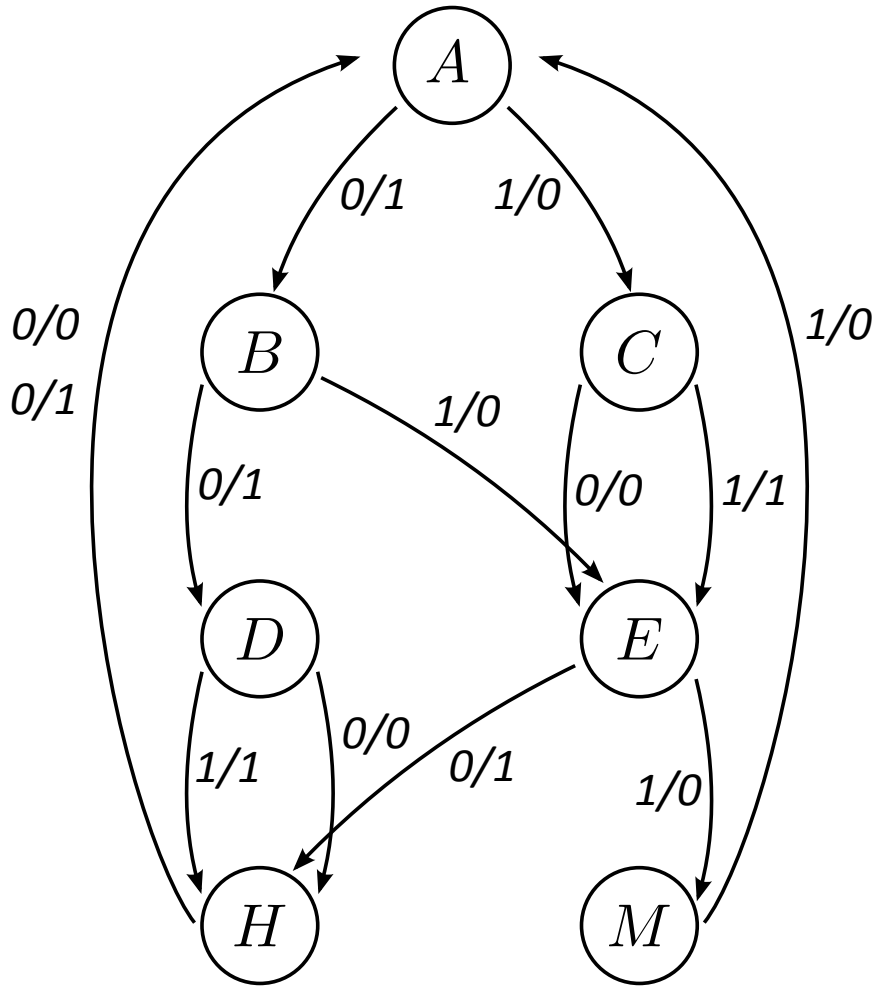
State graph

S	S^+ $x = 0, 1$	Z $x = 0, 1$
A	BC	10
B	DF	10
C	EG	01
D	HL	01
E	IM	10
F	JN	10
G	KP	10
H	AA	01
I	AA	01
J	$A\times$	$0\times$
K	$A\times$	$0\times$
L	$A\times$	$0\times$
M	$A\times$	$1\times$
N	$A\times$	$1\times$
P	$A\times$	$1\times$



Minimized state graph

S	S^+ $x = 0, 1$	Z $x = 0, 1$
A	BC	10
B	DE	10
C	EE	01
D	HH	01
E	HM	10
H	AA	01
M	$A\times$	$1\times$



Transition table for Flip-flops

S	S^+ $x=0,1$	Z $x=0,1$
A	BC	10
B	DE	10
C	EE	01
D	HH	01
E	HM	10
H	AA	01
M	$A\times$	$1\times$

	$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $x=0,1$	Z $x=0,1$
A	000	100 101	10
B	100	111 110	10
C	101	110 110	01
D	111	011 011	01
E	110	011 010	10
H	011	000 000	01
M	010	000 $\times$	$1\times$
–	001	$\times$ $\times$	$\times\times$

Q_1 \ Q_2Q_3	00	01	11	10
0	A		H	M
1	B	C	D	E

← States are given adjacent assignments in order to simplify the next state function.

Transition maps

Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	1	1
01	×	1	1	×
11	0	0	0	0
10	0	0	0	×

$$D_1 = Q_1^+ = \overline{Q_2}$$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	1	0
01	×	1	1	×
11	0	1	1	0
10	0	1	1	×

$$D_2 = Q_2^+ = Q_1$$

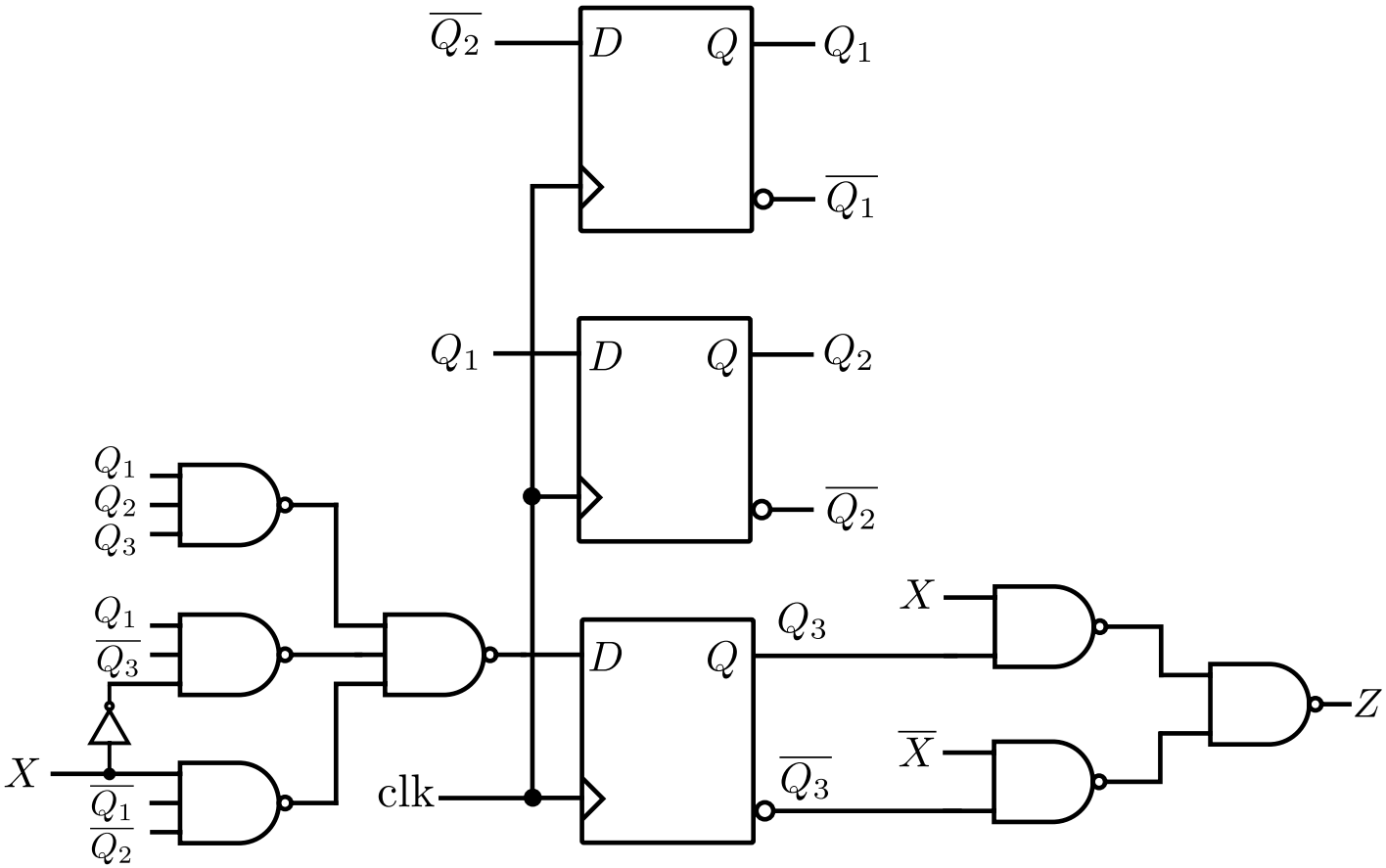
Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	0	1
01	×	0	0	×
11	0	1	1	0
10	0	1	0	×

$$D_3 = Q_3^+ = Q_1Q_2Q_3 + \overline{X}Q_1\overline{Q_3} + X\overline{Q_1}\,\overline{Q_2}$$

Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	0	0
01	×	0	1	×
11	0	0	1	1
10	1	1	0	×

$$Z = \overline{X}\,\overline{Q_3} + XQ_3$$

Code converter circuit



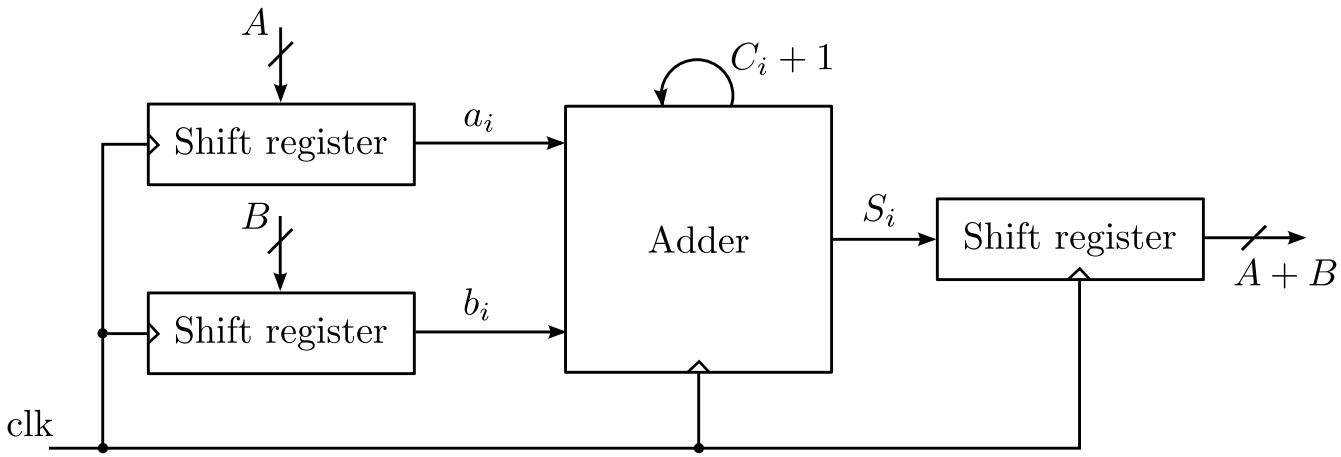
Serial Adder

We will design a circuit that adds two numbers A and B, bit-by-bit.

$\rightarrow A = a_n a_{n-1} a_{n-2} \ldots a_0$

$\rightarrow B = b_n b_{n-1} b_{n-2} \ldots b_0$

The circuit outputs the sum $S_i = a_i + b_i$.

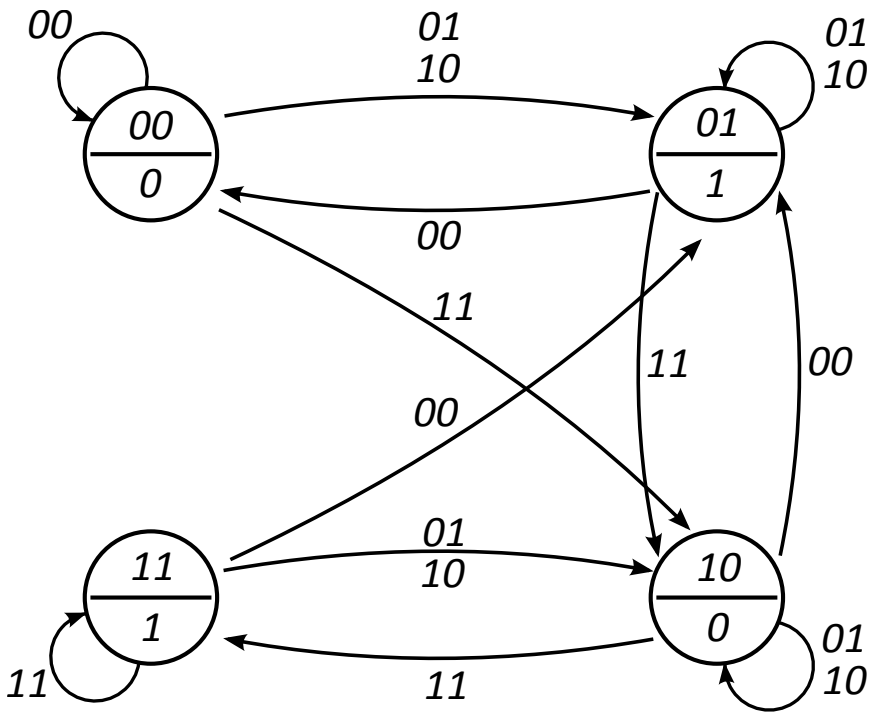


State diagram

State diagram of a Moore sequential adder.

Inputs: a_i and b_i.

Output: Sum.



State table and map

Current state CS	Next state C^+S^+ $ab = 00$	01	11	10	Output z
00	00	01	10	01	0
01	00	01	10	01	1
11	01	10	11	10	1
10	01	10	11	10	0

K-maps for D-flip-flops

CS \ ab	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	1
10	0	1	1	1

S

$D_0 = Q_0^+ = ab + aC + bC$

CS \ ab	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

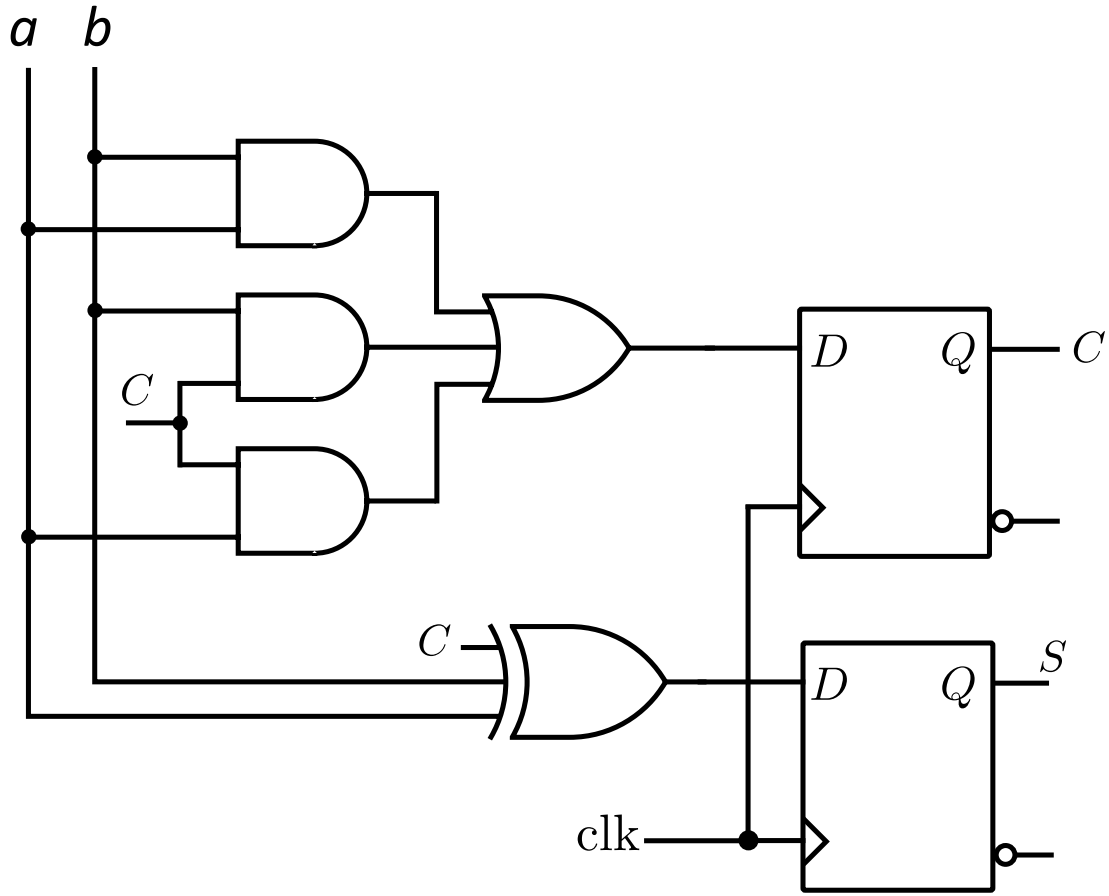
C

$D_1 = Q_1^+ = a \oplus b \oplus C$

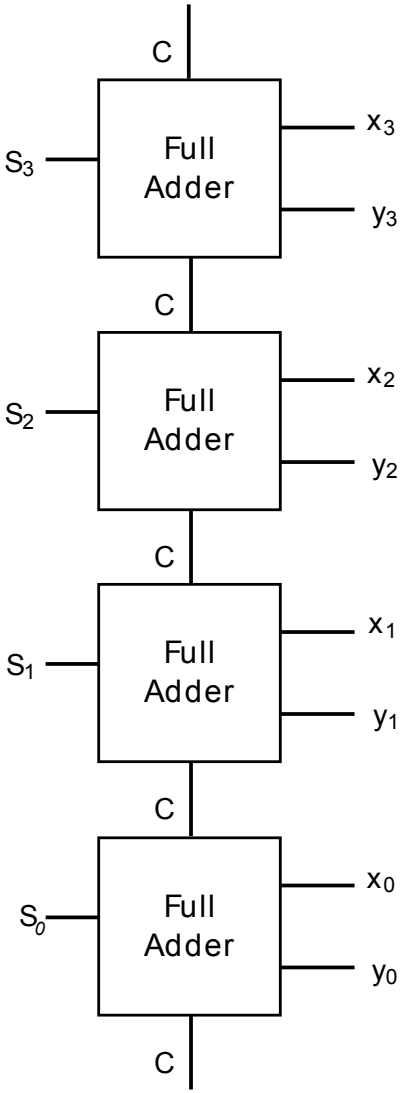
S \ C	0	1
0	0	0
1	1	1

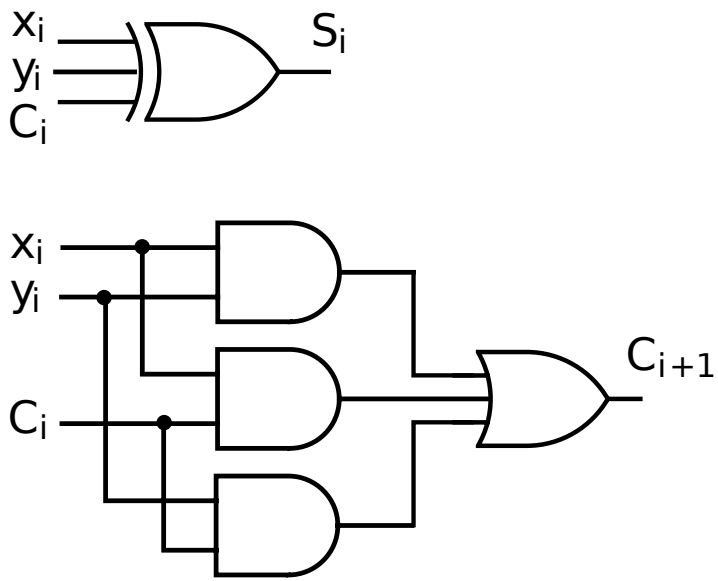
$z = S$

Serial adder circuit

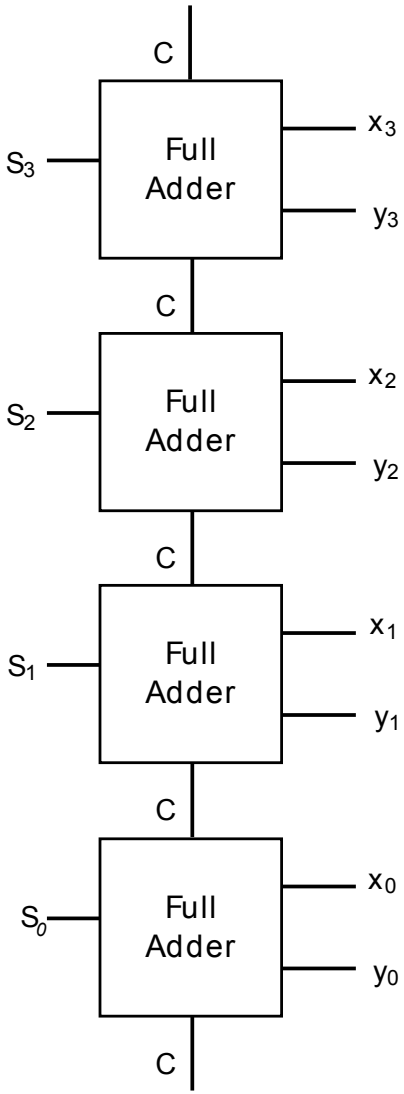


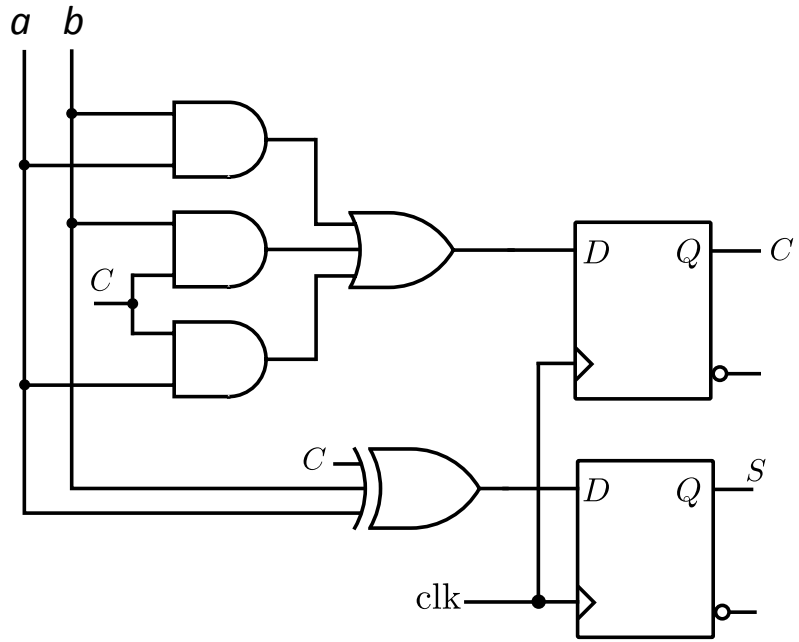
Parallel adder implementation - From Lecture 19

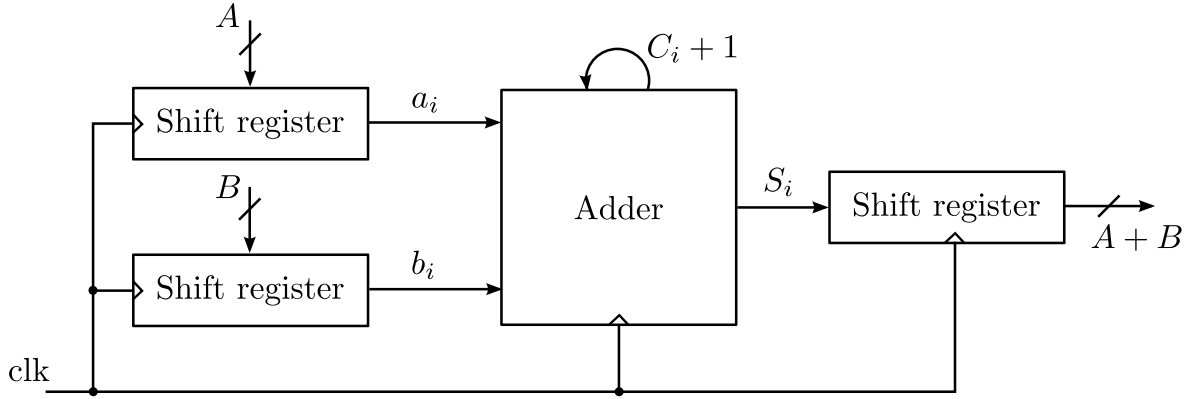




Serial *vs* parallel adder







Next class...

- Design example
- Please read Lecture 33

ECE 210
Introduction to Digital Logic Design

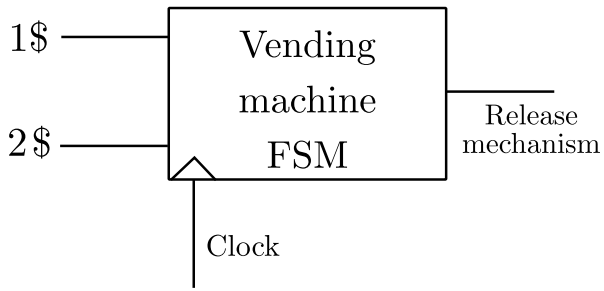
Lecture 33
Design Examples

Example 1 - Vending machine

Release item after $3 are deposited

Single coin slot for loonie (1$) and toonie (2$)

No change



Example 1

Moore machine state graph

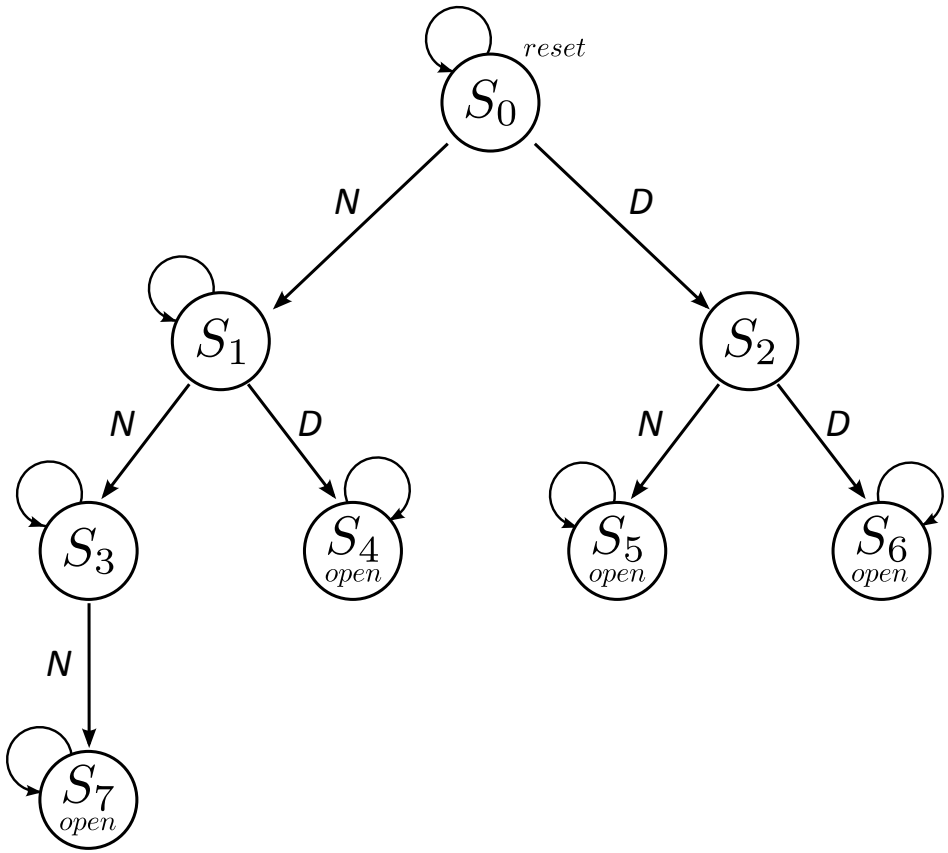
$\rightarrow$ 3 loonies (3 $\times$ \$1)

$\rightarrow$ Loonie + toonie (\$1 + \$2)

$\rightarrow$ Toonie + loonie (\$2 + \$1)

$\rightarrow$ Two loonies (\$2 + \$2)

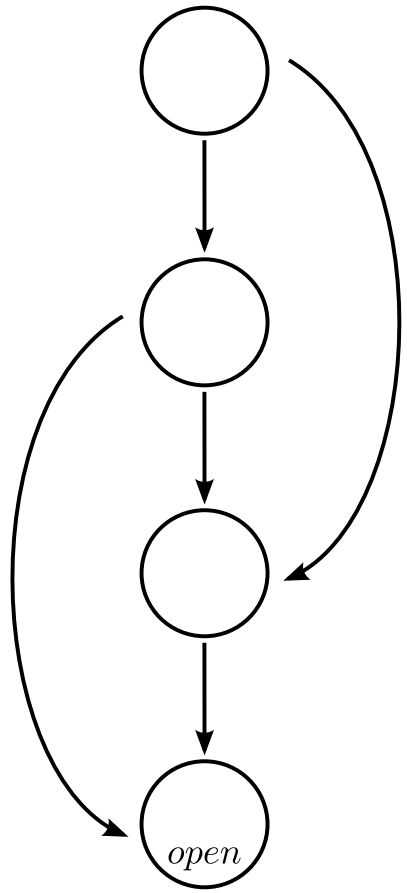
Inputs: L and T



Example 1

Minimize the number of states

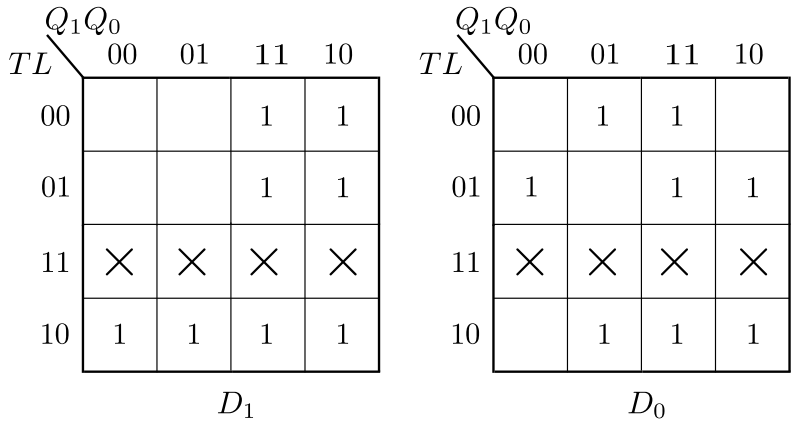
present state	inputs LT	next state	output
0$	00	0$	0
	01	1$	0
	10	2$	0
	11	×	0
1$	00	1$	0
	01	2$	0
	10	3$	0
	11	×	0
2$	00	2$	0
	01	3$	0
	10	2$	0
	11	×	0
3$	×	×	1

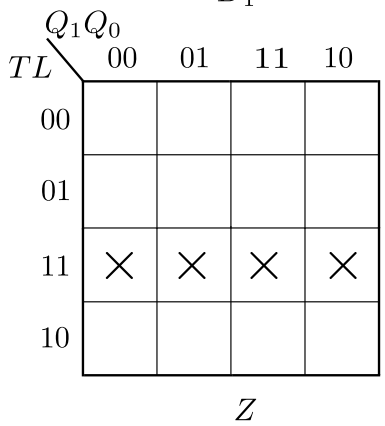


Example 1

Transition maps using D-*FF*

Q_1 Q_0	inputs LT	next Q_1^+ Q_0^+	z
00	00	00	0
	01	01	0
	10	10	0
	11	×	0
01	00	01	0
	01	10	0
	10	11	0
	11	×	0
10	00	10	0
	01	11	0
	10	10	0
	11	×	0
11	×	×	1





$Z =$

$D_0 =$

$D_1 =$

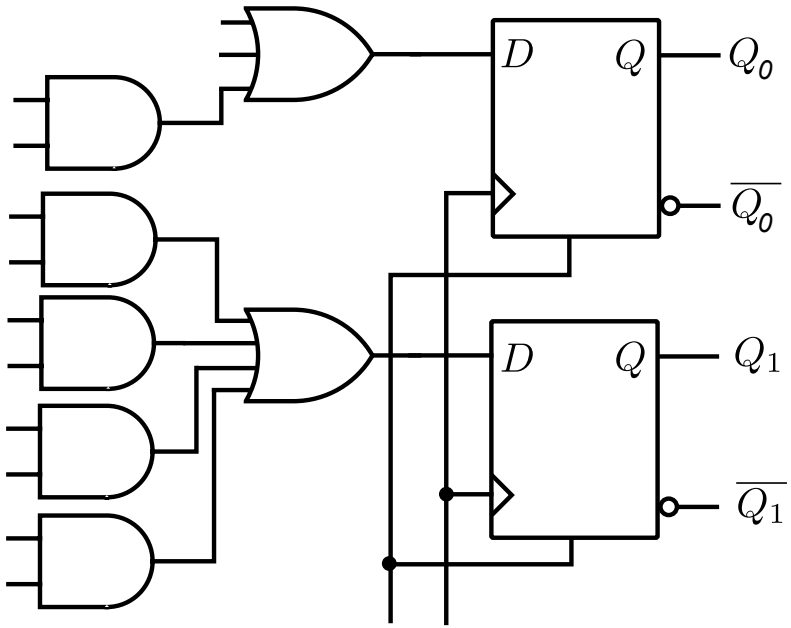
Example 1

Circuit implementation

$$D_1 = Q_1 + D + Q_0 + L$$

$$D_0 = \overline{Q_0} L + Q_0 \overline{L} + Q_1 L + Q_1 T$$

$$Z = Q_0 Q_1$$

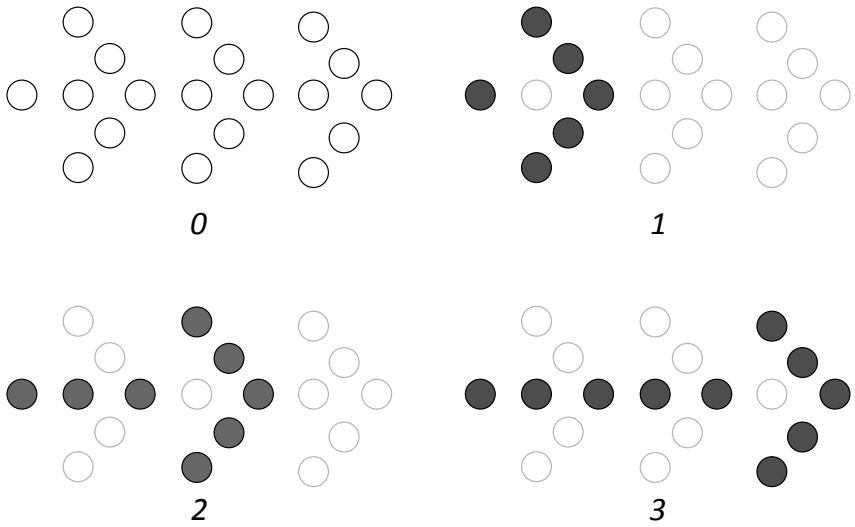


Example 2 - Road construction electric sign

Design a Mealy machine to drive the electric sign used for detours during road constructions.

The sign should display the sequence shown below when the input D is zero.

The arrow should blink (alternate between state 0 and 3) when $D = 1$.

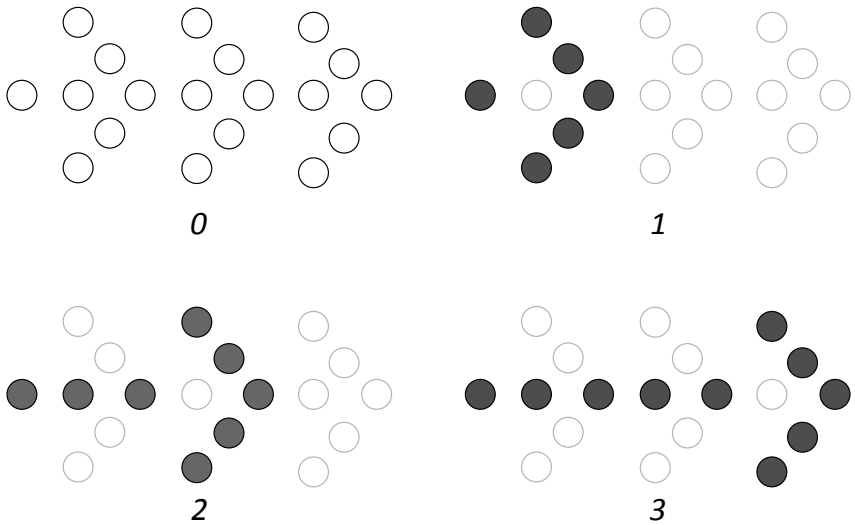


Example 2 - Road construction electric sign

Design a Moore machine to drive the electric sign used for detours during road constructions.

The sign should display the sequence shown below when the input X is zero.

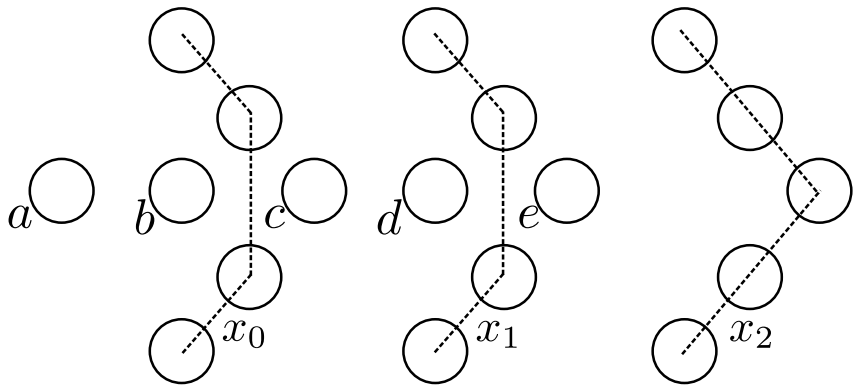
When the input X is 1 and the system is not in state 1, the sequence should be carried on until state 3 is reached. The arrow should then blink (alternate between states 0 and 3).



Example 2

Output assignment

Group the LED's that are only activate simultaneously

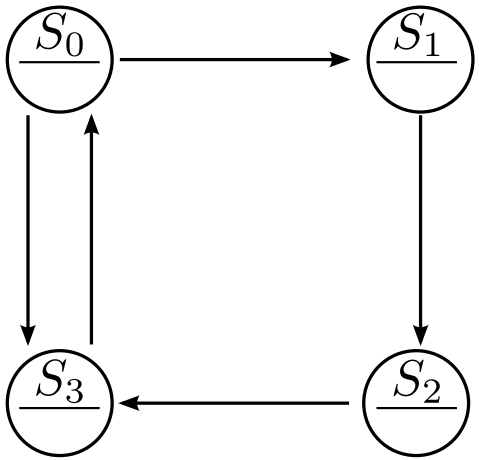


Output $abcdex_0x_1x_2$:

$$00000000 \rightarrow 10100100 \rightarrow 11101010 \rightarrow 11111001$$

$$0 \rightarrow A4 \rightarrow EA \rightarrow F9$$

Example 2



Present state	next state $X = 0, 1$	output
S_0	S_1 S_3	0
S_1	S_2 S_2	A4
S_2	S_3 S_3	EA
S_3	S_0 S_0	F9

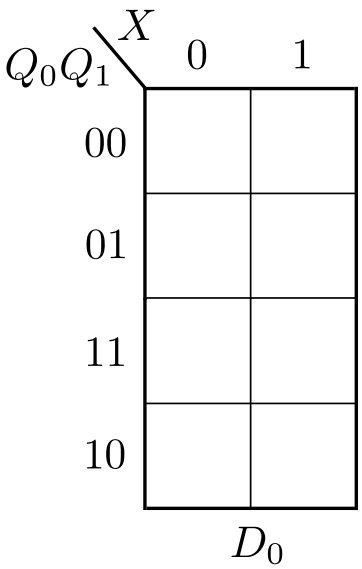
Present state $Q_0 Q_1$	next state $X = 0, 1$	output
00	01 10	0
01	11 11	A4
11	10 10	EA
10	00 00	F9

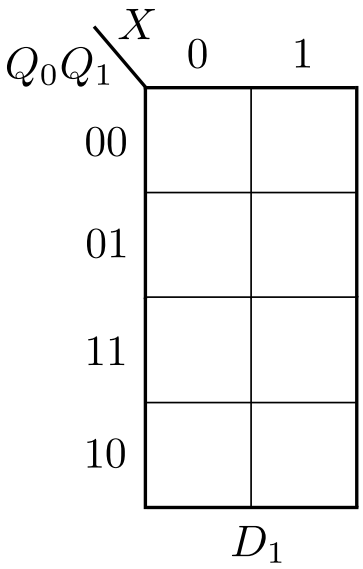
Example 2

Present state $Q_0 Q_1$	next state $X = 0, 1$	output z
00	01 10	0
01	11 11	A4
11	10 10	EA
10	00 00	F9

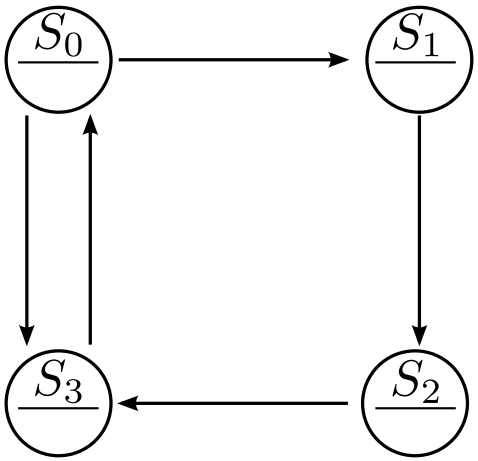
$D_0 =$

$D_1 =$





Example 2



$a =$

$b =$

$c =$

$d =$

$e =$

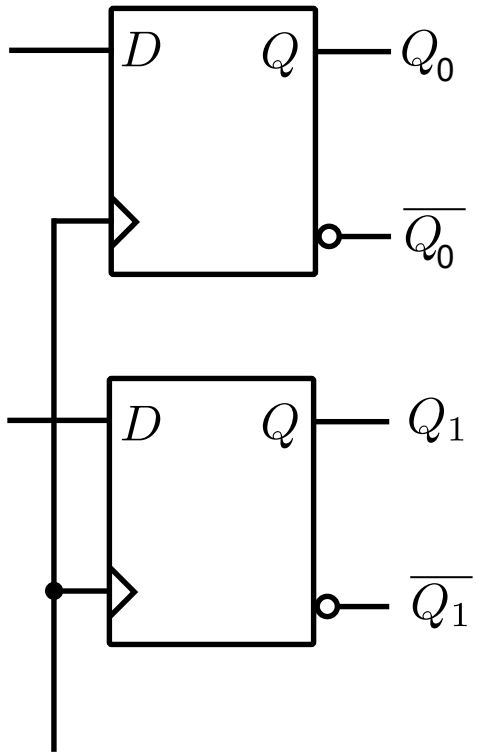
$x_0 =$

$x_1 =$

$x_2 =$

State	output $abcdex_0x_1x_2$
00	00000000
01	10100100
10	11101010
11	11111001

Example 2



The end