



**The Abdus Salam  
International Centre for Theoretical Physics**



**2177-14**

**ICTP Latin-American Basic Course on FPGA Design for Scientific  
Instrumentation**

*15 - 31 March 2010*

**Digital Design III**  
**(more complex elements: RAM, ROM, buses, pipeline concept, etc)**

BAZARGAN SABET Pirouz  
*LIP6, University Pierre et Marie Curie  
Paris  
France*

# Outline

## ■ Digital CMOS design

- Boolean algebra
- Basic digital CMOS gates
- **Combinational and sequential circuits**
- Coding - Representation of numbers

# CMOS Circuits

How to implement Boolean functions  
in CMOS technology ?

- A complex function cannot be implemented using a single gate
- Use a network of gates

Boolean network



# CMOS Circuits

Example :

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$f = \bar{x}.y.z + x.\bar{y}.\bar{z} + x.\bar{y}.z + x.y.z$$

$$f = (x+y+z) . (x+y+\bar{z}) .$$

$$(x+\bar{y}+z) . (\bar{x}+\bar{y}+z)$$

$$f = x.(yz+\bar{y}) + \bar{x}.(y.z)$$

$$f = \bar{x}.y.z + x.\bar{y}.\bar{z} + x.z$$

$$f = x.\bar{y} + y.z$$

There is not a unique expression



# CMOS Circuits

Example :

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Which gates should I use ?

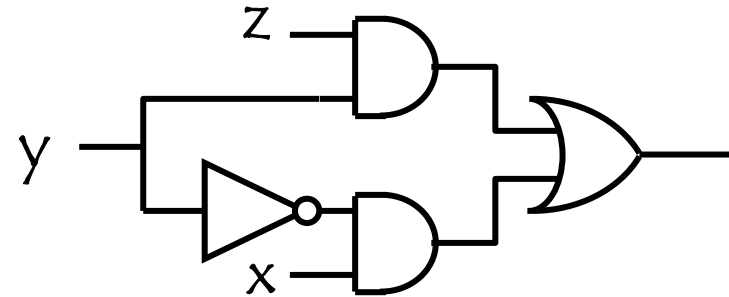
Cell library

$$f = x \cdot \bar{y} + y \cdot z$$

# CMOS Circuits

Example :

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



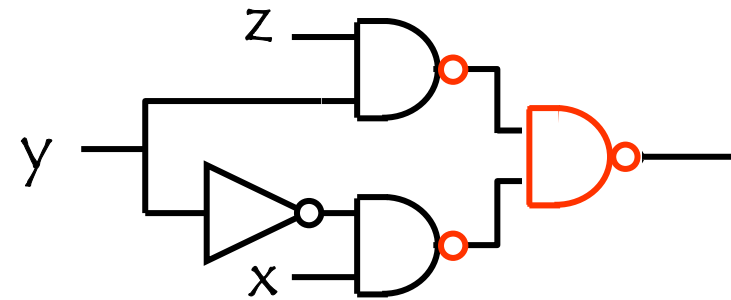
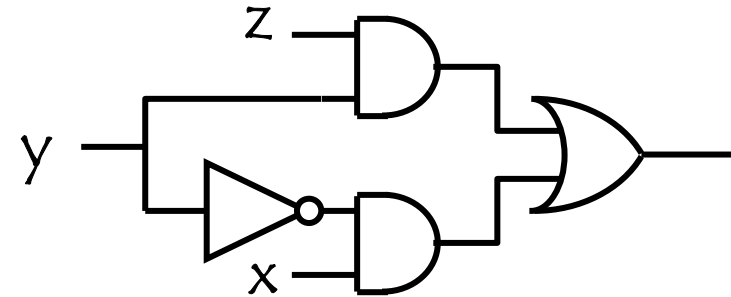
Non-inverting gates do NOT exist

$$f = x \cdot \bar{y} + y \cdot z$$

# CMOS Circuits

Example :

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



# CMOS Circuits

Example :

$$f = (\bar{x} + y) \oplus x$$

$$f = \overline{(\bar{x} + y)} \oplus \bar{x}$$

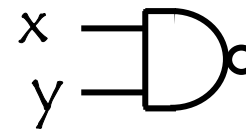
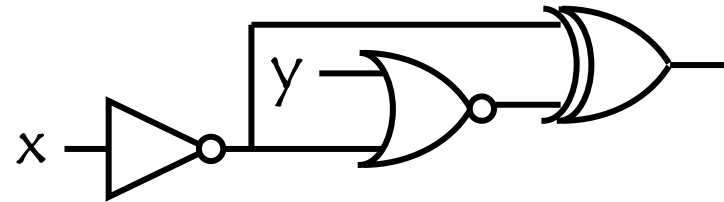
$$f = (\bar{x} + y) \cdot \bar{x} + \overline{(\bar{x} + y)} \cdot x$$

$$f = \bar{x} + y \cdot \bar{x} + x \cdot \bar{y}$$

$$f = \bar{x} + x \cdot \bar{y}$$

$$f = \bar{x} + \bar{y}$$

$$f = \overline{x \cdot y}$$





# CMOS Circuits

How to implement Boolean functions  
with a gate network ?

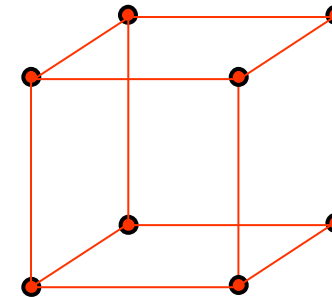
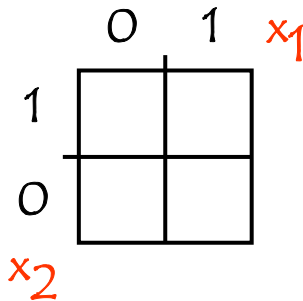
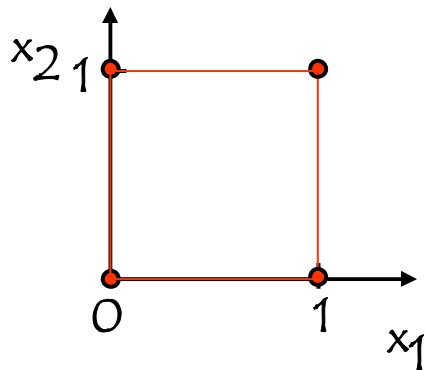
- Which expression ?

# CMOS Circuits

A function can be defined by its Truth table

- **Karnaugh representation gives a minimal expression**

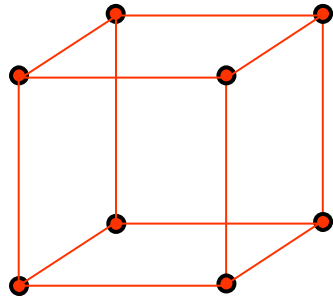
Representation of the function in a space of dimension  $n$



Representation of vectors' adjacency

# CMOS Circuits

A Karnaugh table is represented in a flat 2-dimension table



	00	01	11	10
0		○		○
1				

The table shows a 2x4 grid of cells. The top row is labeled with binary vectors 00, 01, 11, and 10. The left column is labeled with 0 and 1. The cell at (0, 01) contains a small circle with a red line extending downwards to the cell at (1, 01). The cell at (0, 10) contains a small circle with a blue line extending downwards to the cell at (1, 10). A blue oval highlights the top row (00, 01, 11, 10), and a blue line connects the circles in the 0 row to the circles in the 1 row.

vectors' adjacency

# CMOS Circuits

Example :

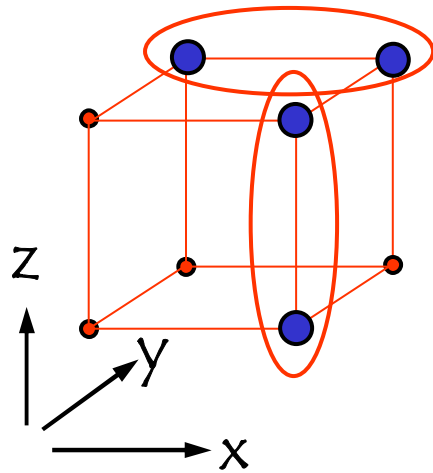
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

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

z

# CMOS Circuits

Example :



	00	01	11	10	xy
0	0	0	0	1	
1	0	1	1	1	
z					

$$f = x.\bar{y} + yz$$

# CMOS Circuits

Example :

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

zV

$$\bar{x}.y + x.\bar{v} + y.\bar{z}$$

# CMOS Circuits

Example :

	000	001	011	010	110	111	101	100	xyv
0	0	1	0	0	0	0	1	0	
1	0	1	1	1	1	0	1	0	

z

The Karnaugh map shows a 2x8 grid of cells. The columns are labeled 000, 001, 011, 010, 110, 111, 101, 100. The rows are labeled 0 and 1. The output values are 0 or 1. Red circles highlight the 1s in the 001, 011, 010, 110, and 101 columns. Red arrows point from these circles to the terms in the Boolean expression below. Blue and black curved arrows are also present at the top of the map, indicating wrap-around connections.

$$g = \bar{x}.y.z + y.\bar{v}.z + \bar{y}.v$$

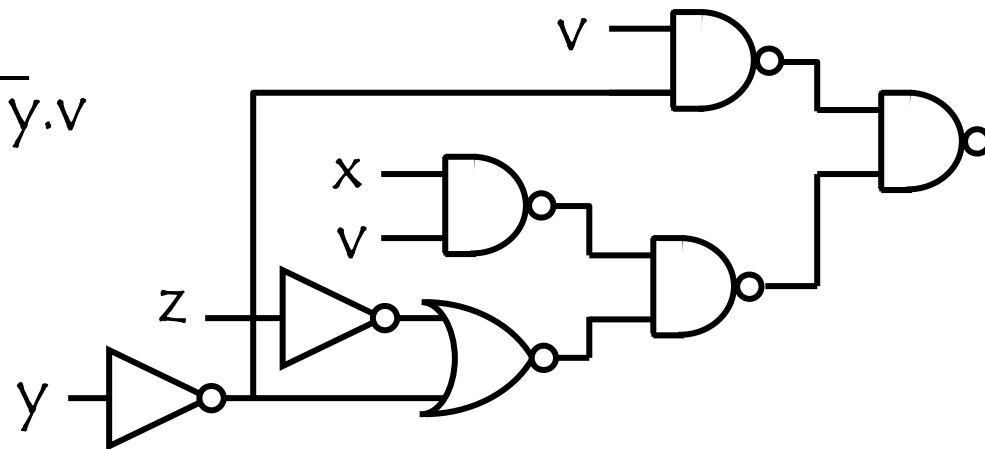
# CMOS Circuits

Example :

$$g = \bar{x}.y.z + y.\bar{v}.z + \bar{y}.v$$

$$g = y.z.(\bar{x}+\bar{v}) + \bar{y}.v$$

$$g = \overline{(\bar{y}+\bar{z})}.(\bar{x}.v) + \bar{y}.v$$





# CMOS Circuits

How to implement Boolean functions  
with a gate network ?

- Local optimization using Karnaugh tables

A design includes several Boolean functions

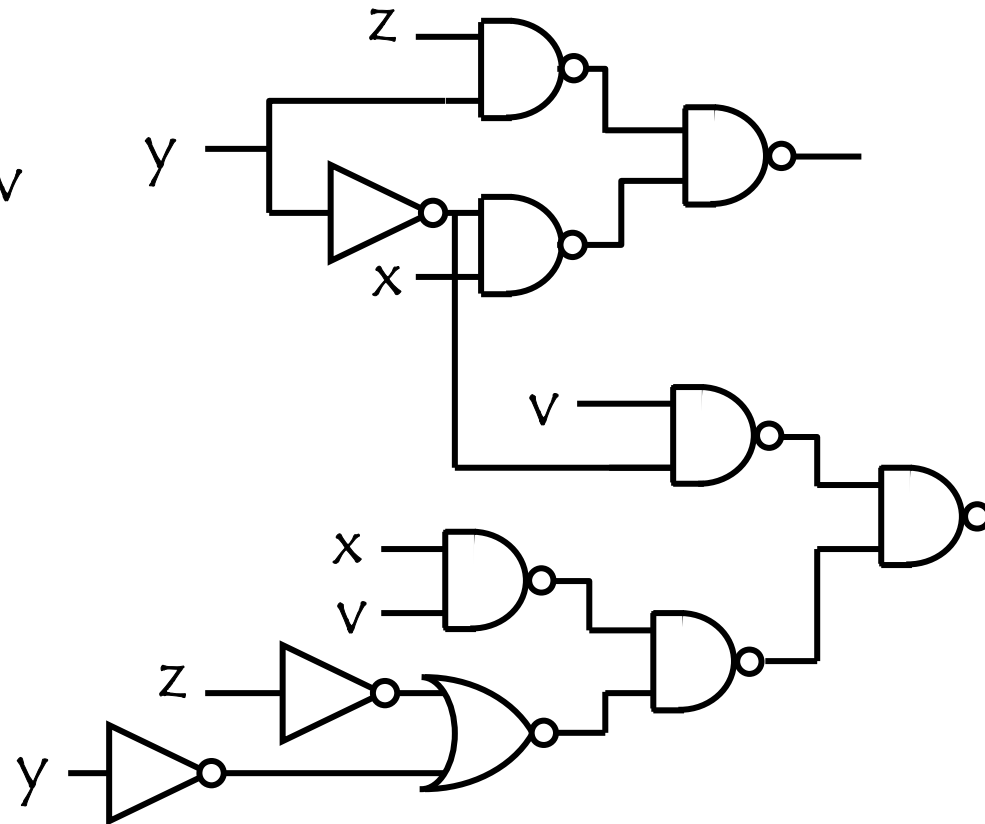


# CMOS Circuits

Example :

$$g = \overline{\overline{\overline{y+z}}}.(\overline{x.v}) + \overline{y}.v$$

$$f = x.\overline{y} + y.z$$

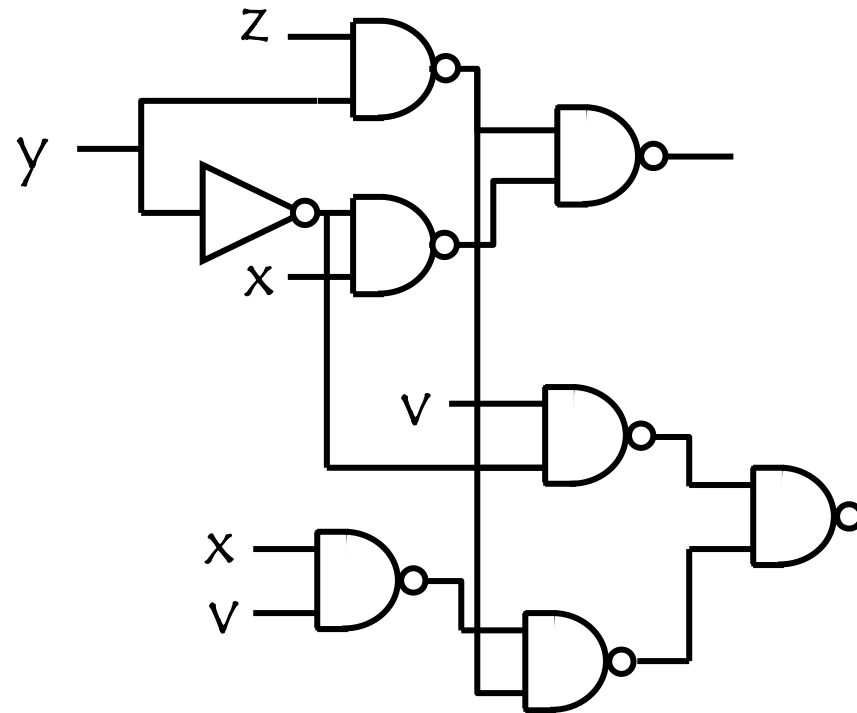


# CMOS Circuits

Example :

$$g = \overline{\overline{\overline{y+z}}}.(\overline{x.v}) + \overline{y}.v$$

$$f = x.\overline{y} + y.z$$



# CMOS Circuits

How to implement Boolean functions  
with a gate network ?

- Local optimization using Karnaugh tables

A design includes several Boolean functions

- Global optimization by sharing sub functions



Synthesis Tool

# CMOS Circuits

- **Combinational logic**

The value of the output can be determined knowing the value of the inputs

- **Sequential logic**

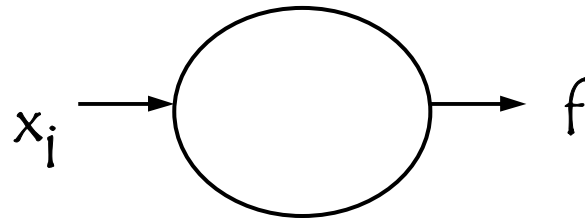
The value of the output depends on the value of the inputs **and the history**

**Notion of memory**



# CMOS Circuits

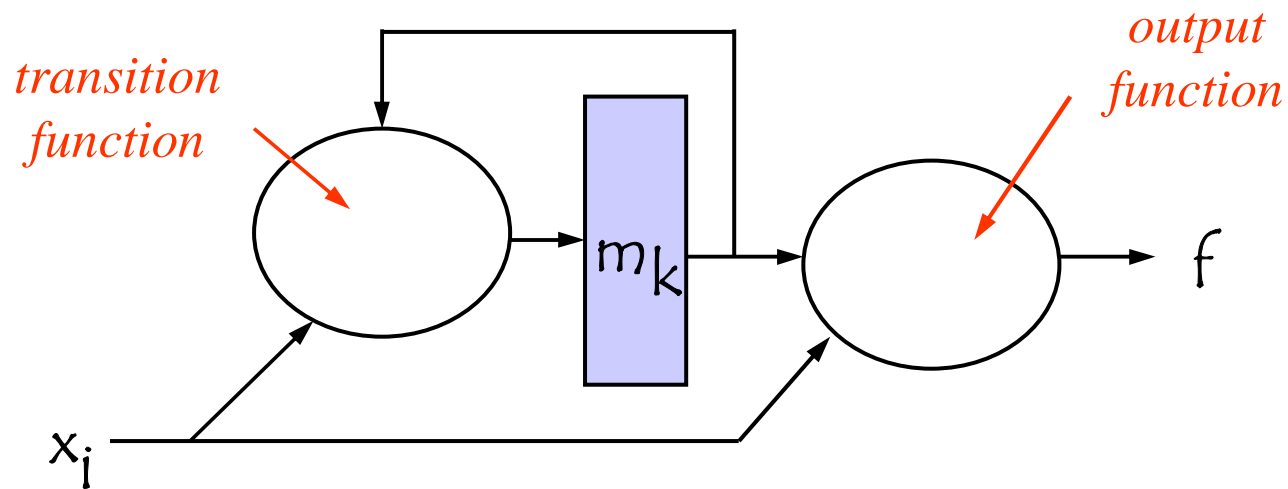
## Sequential logic



$$f(x_1, \dots, x_i, \dots, x_n, m_1, \dots, m_k, \dots, m_p)$$
$$m_k(x_1, \dots, x_i, \dots, x_n, m_1, \dots, m_k, \dots, m_p)$$

# CMOS Circuits

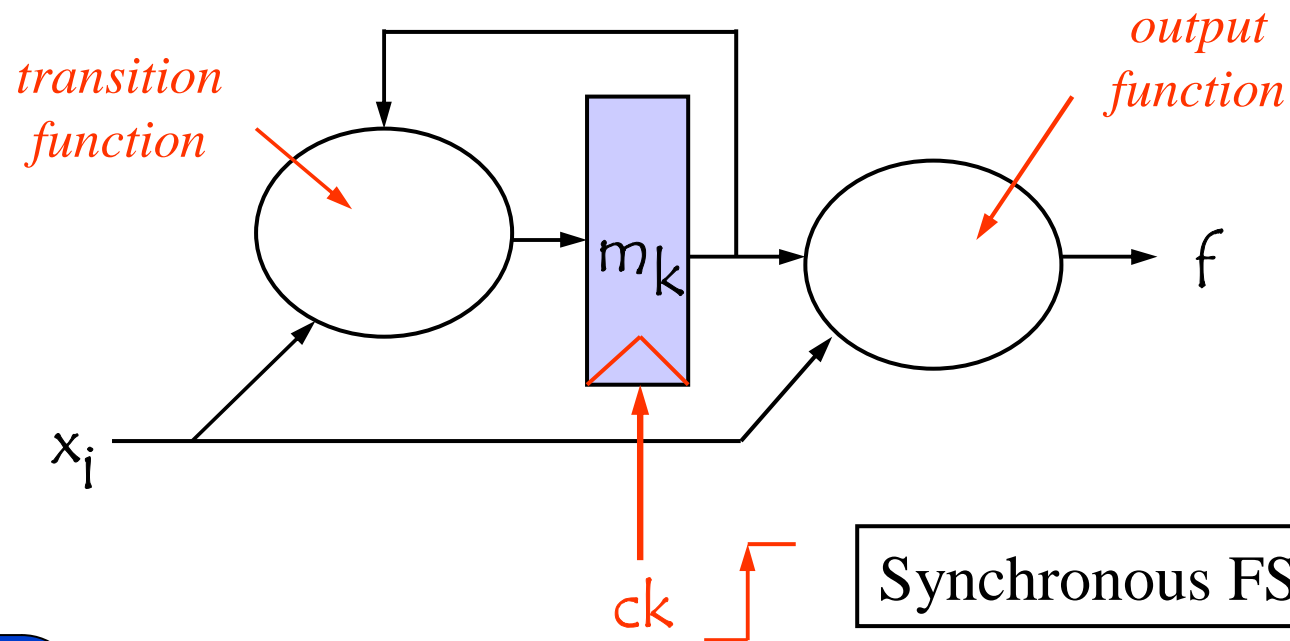
## Sequential logic



Finite State Machine (FSM)

# CMOS Circuits

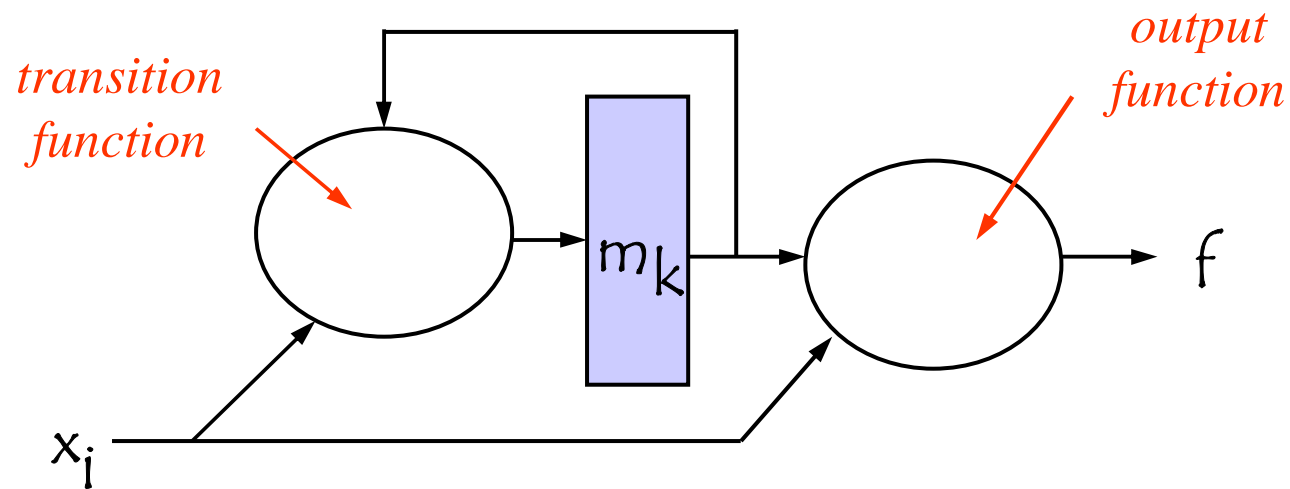
## Finite State Machine





# CMOS Circuits

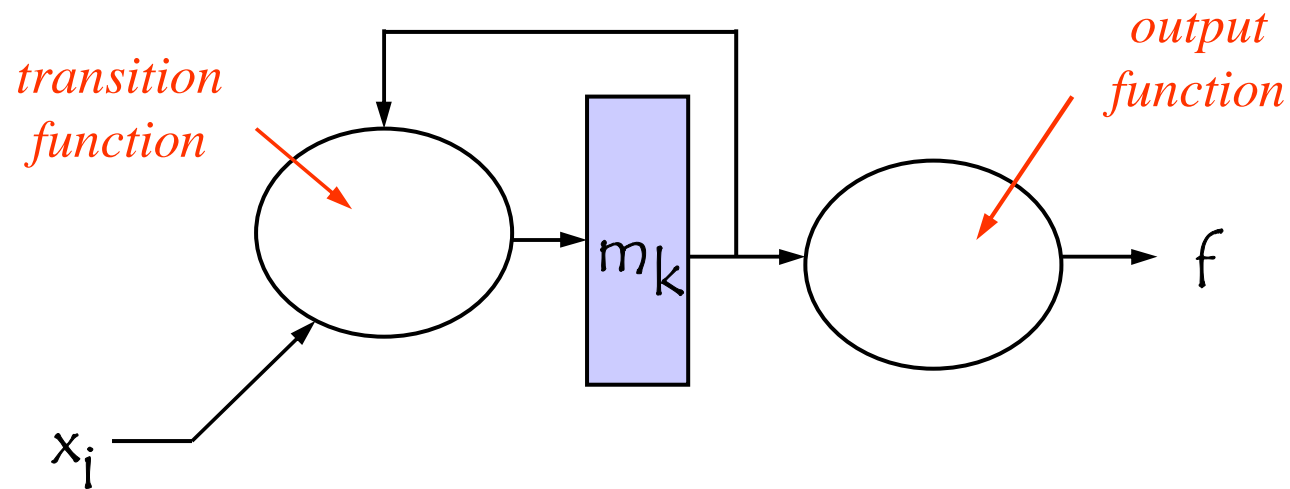
## Finite State Machine



Mealy FSM

# CMOS Circuits

## Finite State Machine



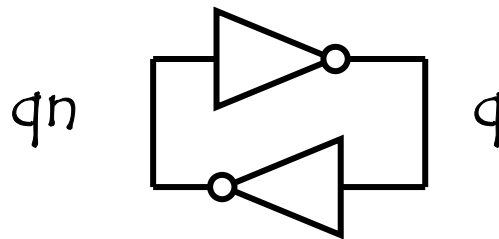
Moore FSM

# CMOS Circuits

Memory :

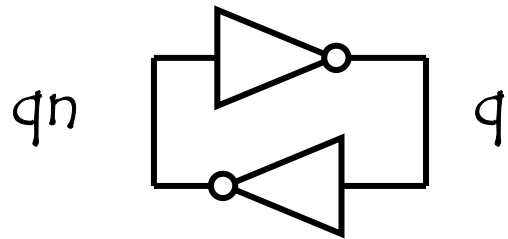
Hold a data (0 or 1)

Write a data (0 or 1)



# CMOS Circuits

Memory :



$$q = \overline{q_n}$$

$$q_n = \overline{q}$$

$$f = q = \overline{\overline{q}}$$

$$f(q) = q$$

$$\frac{\partial f^+}{\partial q} = f_{q1} \cdot \overline{f_{q0}} = 1$$

$$\frac{\partial f^-}{\partial q} = \overline{f_{q1}} \cdot f_{q0} = 0$$

$q$  depends always on itself in a **positive** way

**this dependency cannot be disabled**

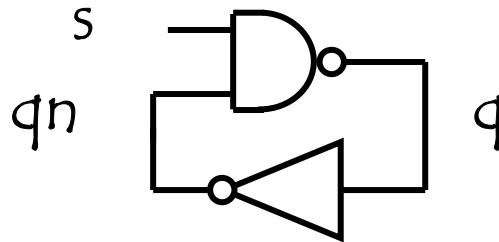


# CMOS Circuits

Memory :

Hold a data (0 or 1)

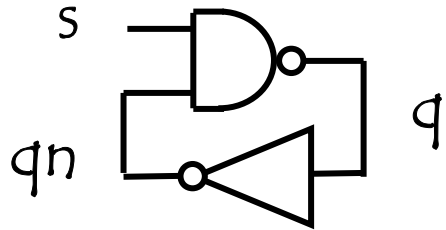
Write a data (0 or 1)



s	q	qn
0	1	0
1	q	qn

# CMOS Circuits

Memory :



$$q = \overline{s \cdot qn} \quad qn = \overline{q}$$

$$f = q = \overline{s \cdot (\overline{q})}$$

$$f = \overline{s} + q \quad f = q + \overline{q} \cdot \overline{s}$$

$$\frac{\partial f^+}{\partial q} = 1 \cdot s = s$$

$$\frac{\partial f^-}{\partial q} = \overline{s} \cdot 0 = 0$$

if  $s = 1$   $q$  is a **positive** function of itself

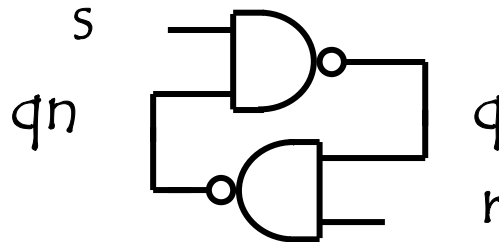
if  $s = 0$   $q$  is a combinational function

# CMOS Circuits

Memory :

Hold a data (0 or 1)

Write a data (0 or 1)

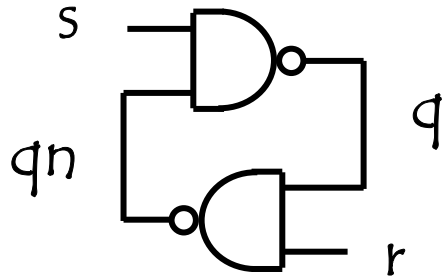


s	r	q	qn
0	1	1	0
1	0	0	1
1	1	q	qn
0	0	1	1

RS flip flop

# CMOS Circuits

Memory :



$$q = \overline{s \cdot qn} \quad qn = \overline{r \cdot q}$$

$$f = q = \overline{s \cdot (r \cdot q)}$$

$$f = \overline{s} + (r \cdot q)$$

$$f = q \cdot (\overline{r+s}) + \overline{q} \cdot \overline{s}$$

$$\frac{\partial f^+}{\partial q} = s \cdot (\overline{s} + r) = s \cdot r$$

$$\frac{\partial f^-}{\partial q} = \overline{s} \cdot \overline{(\overline{s} + r)} = 0$$

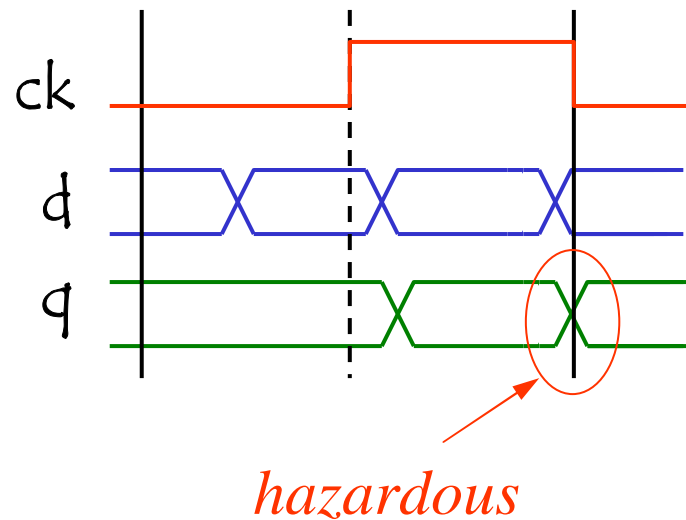
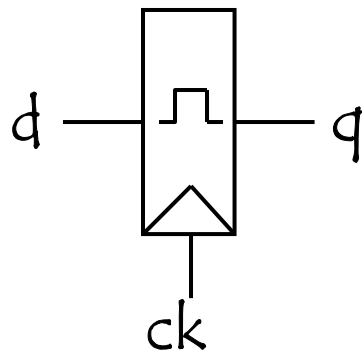




# CMOS Circuits

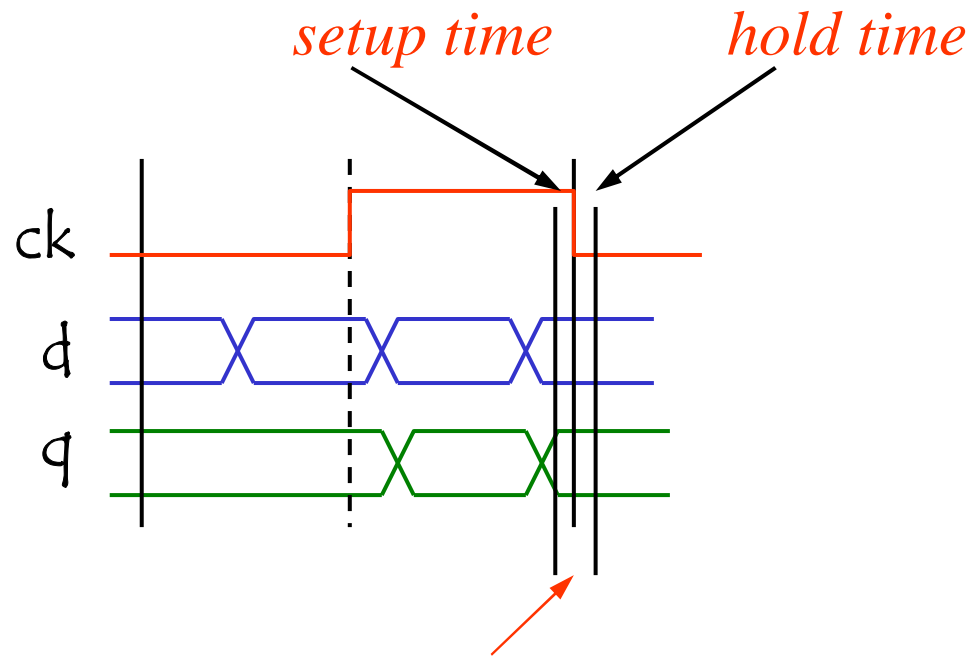
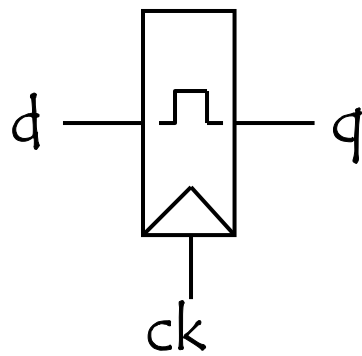
Synchronous Memory :

Write a data  $d$  when the clock  $ck = 1$



# CMOS Circuits

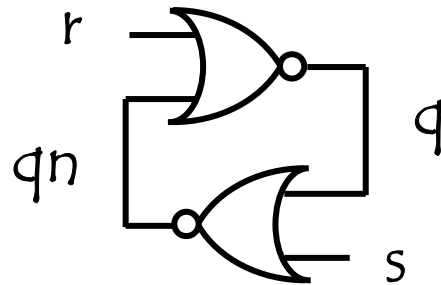
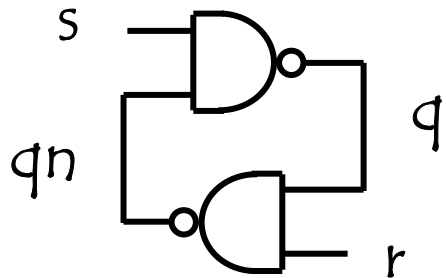
## Synchronous Memory :



data should not change in this period

# CMOS Circuits

Memory :



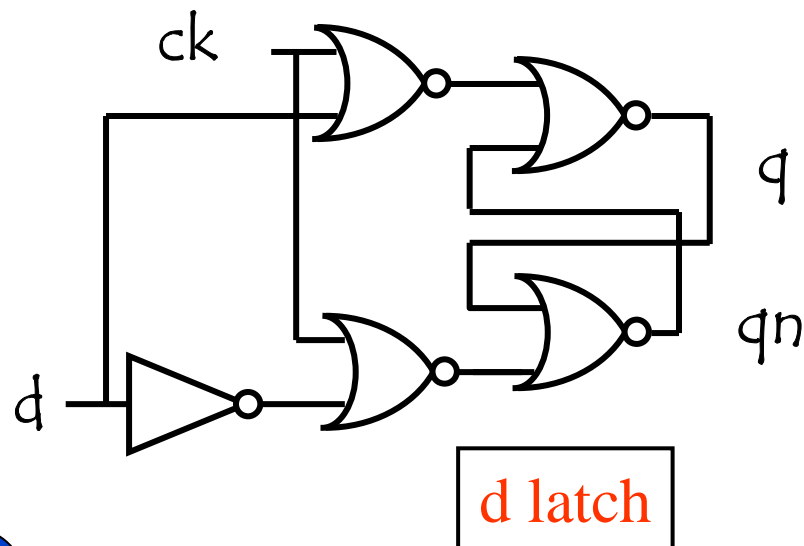
$s$	$r$	$q$	$qn$
0	1	0	1
1	0	1	0
1	1	0	0
0	0	$q$	$qn$

RS flip flop

# CMOS Circuits

## Synchronous Memory :

Write a data  $d$  when the clock  $ck = 0$



s	r	q	qn
0	1	0	1
1	0	1	0
1	1	0	0
0	0	q	qn

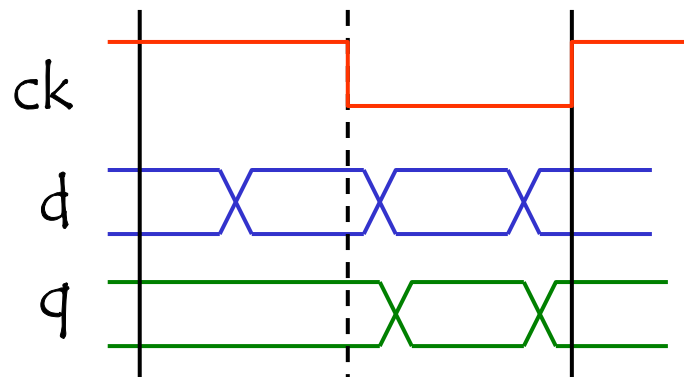
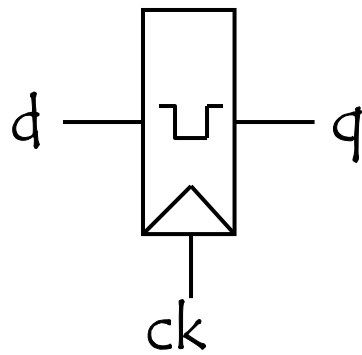
if  $\bar{ck}.d = 1$   $s = 1$

if  $\bar{ck}.\bar{d} = 1$   $r = 1$

# CMOS Circuits

Synchronous Memory :

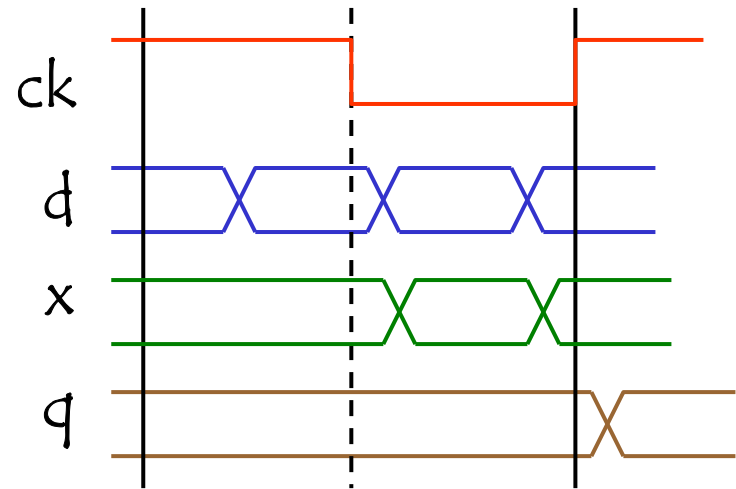
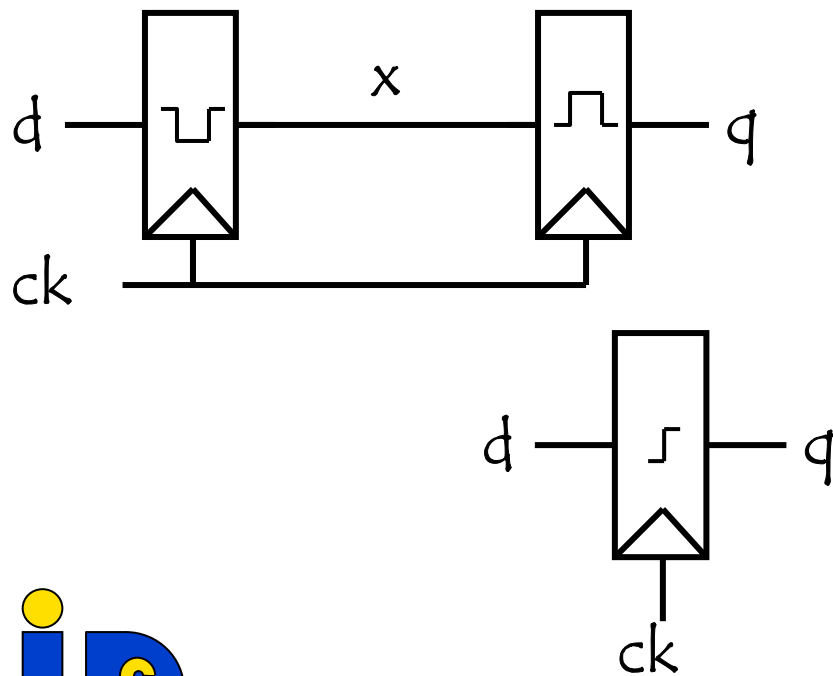
Write a data  $d$  when the clock  $ck = 0$



# CMOS Circuits

## Synchronous Memory :

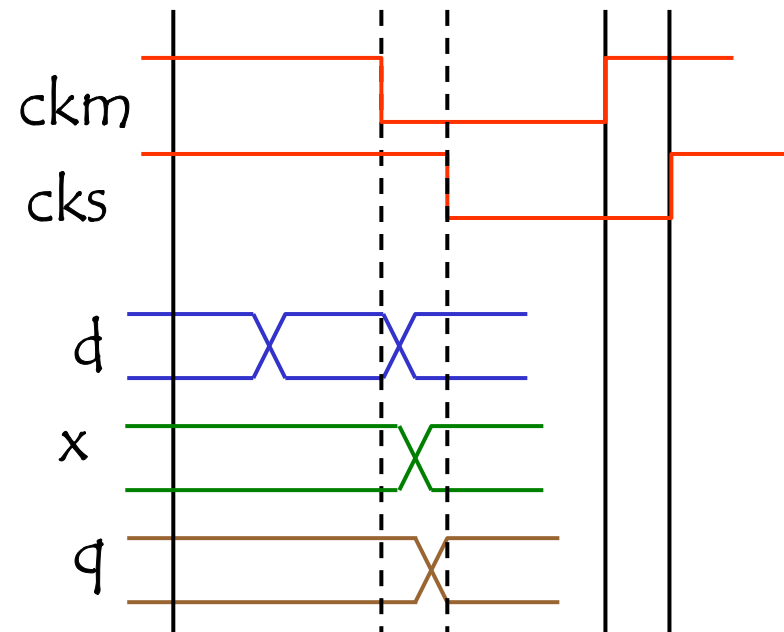
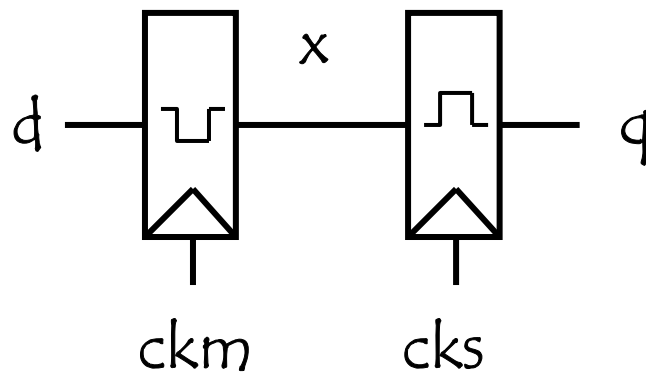
Write a data  $d$  on the rising edge of the clock  $ck$



# CMOS Circuits

Synchronous Memory :

Write a data  $d$  on the rising edge of the clock  $ck$

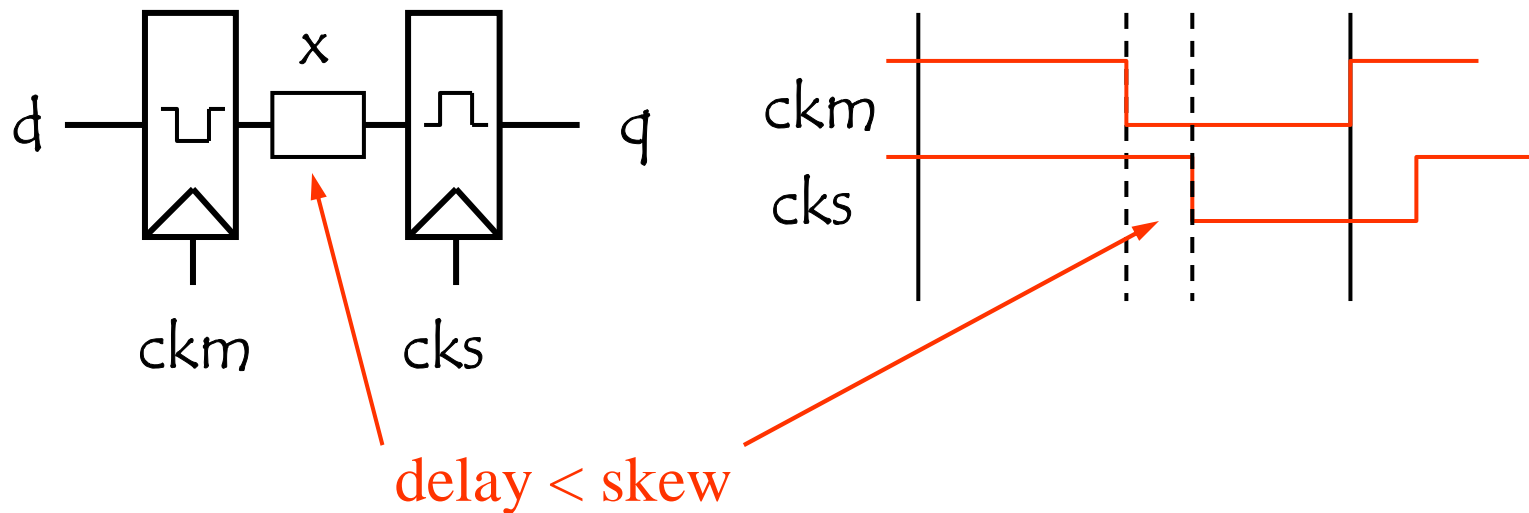




# CMOS Circuits

## Synchronous Memory :

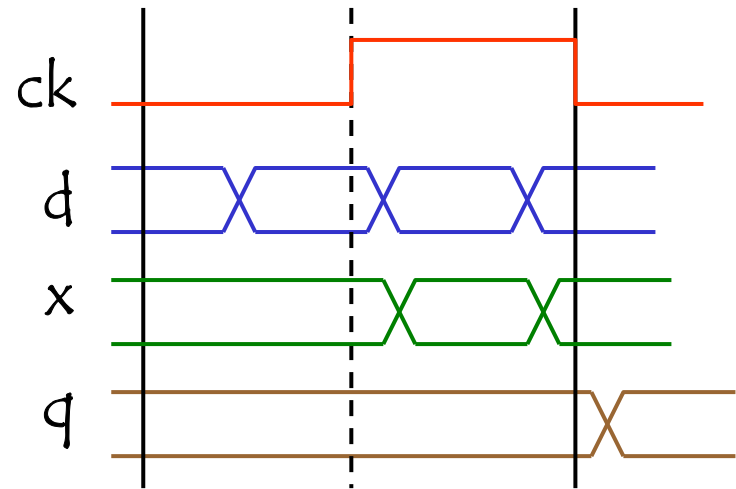
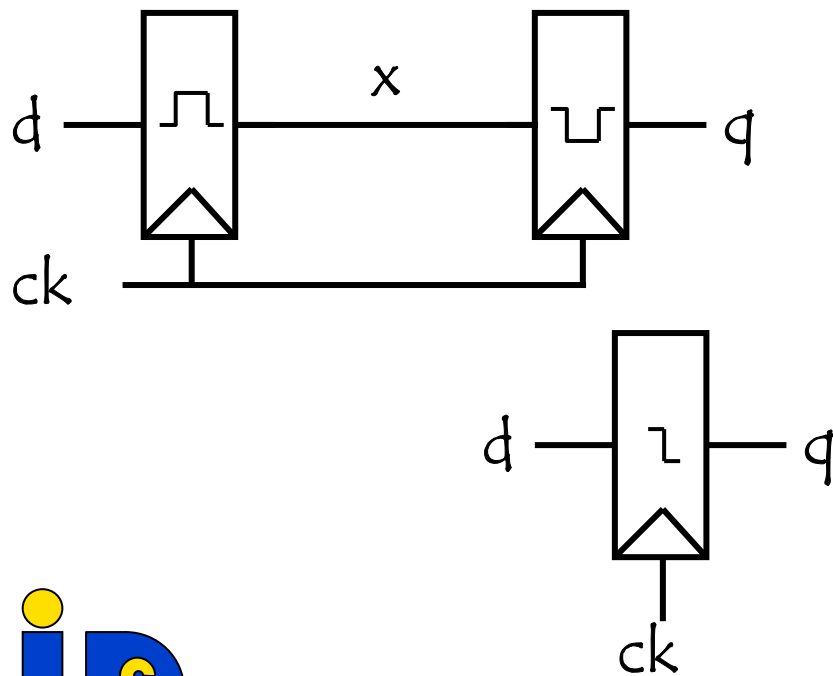
Write a data  $d$  on the rising edge of the clock  $ck$



# CMOS Circuits

## Synchronous Memory :

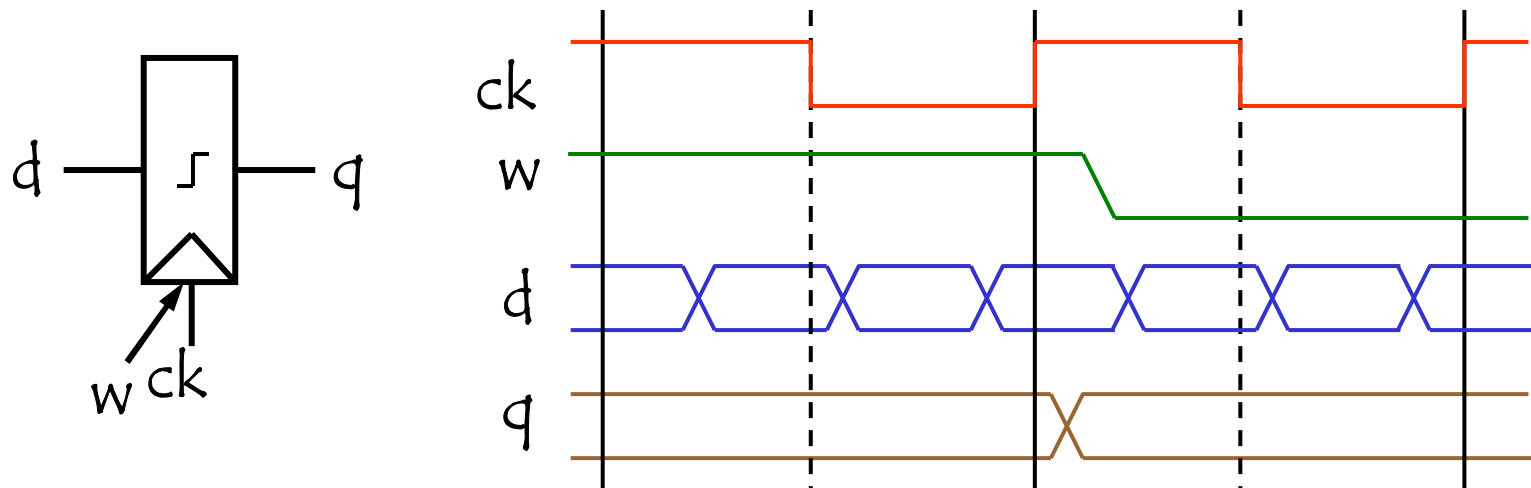
Write a data  $d$  on the falling edge of the clock  $ck$



# CMOS Circuits

## Synchronous Memory :

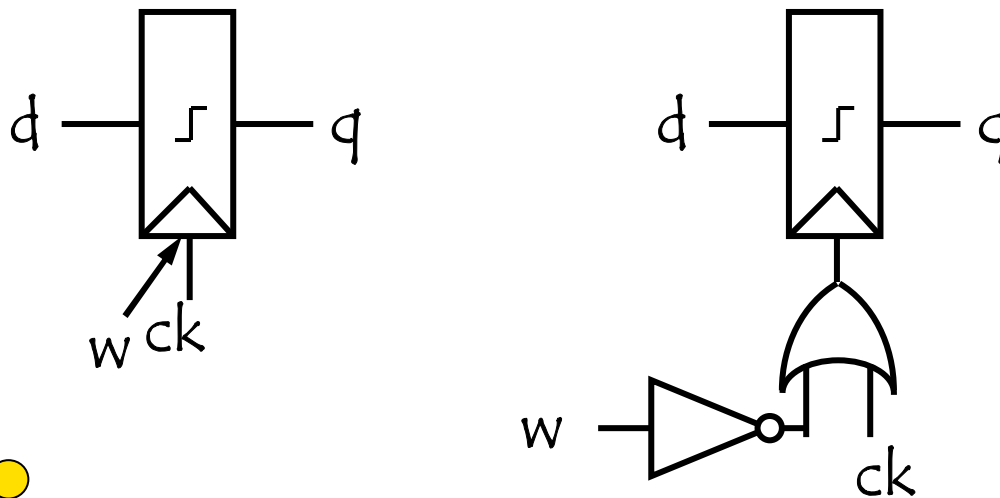
Write a data  $d$  on the rising edge of the clock  $ck$  when a condition is true (write enable)



# CMOS Circuits

## Synchronous Memory :

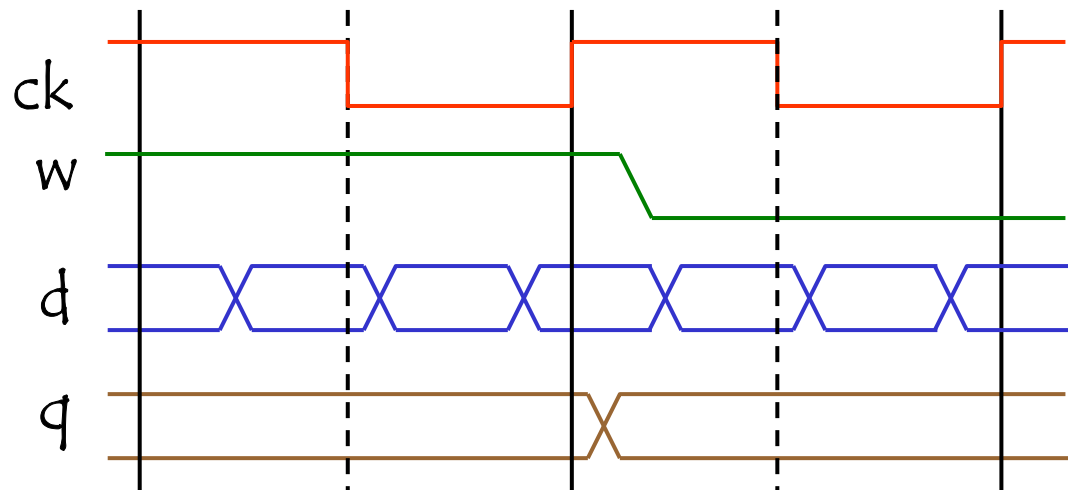
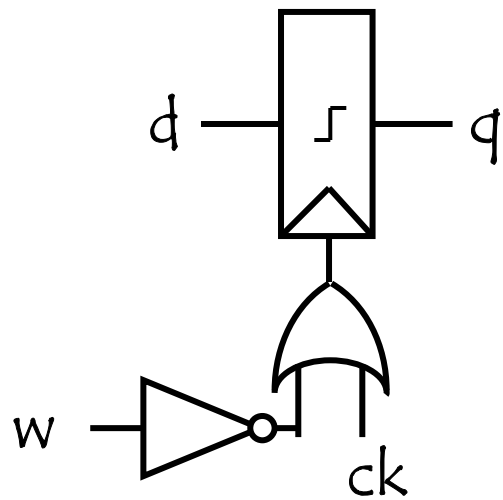
Write a data  $d$  on the rising **edge** of the clock  $ck$  when a condition is true (write enable)



# CMOS Circuits

## Synchronous Memory :

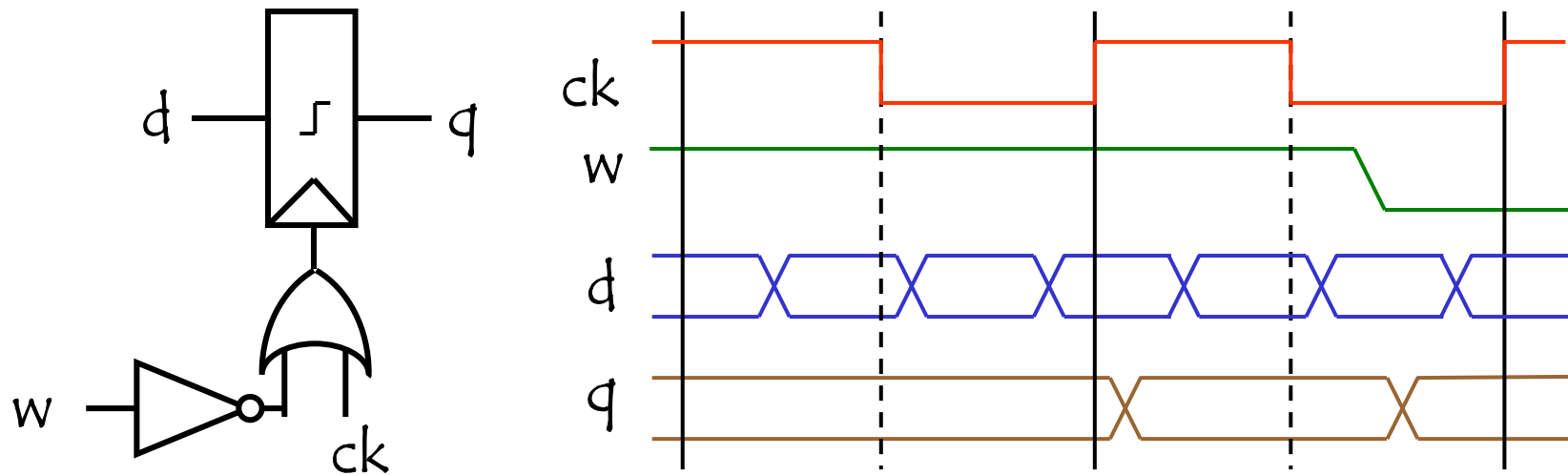
Write a data  $d$  on the rising edge of the clock  $ck$  when a condition is true (write enable)



# CMOS Circuits

## Synchronous Memory :

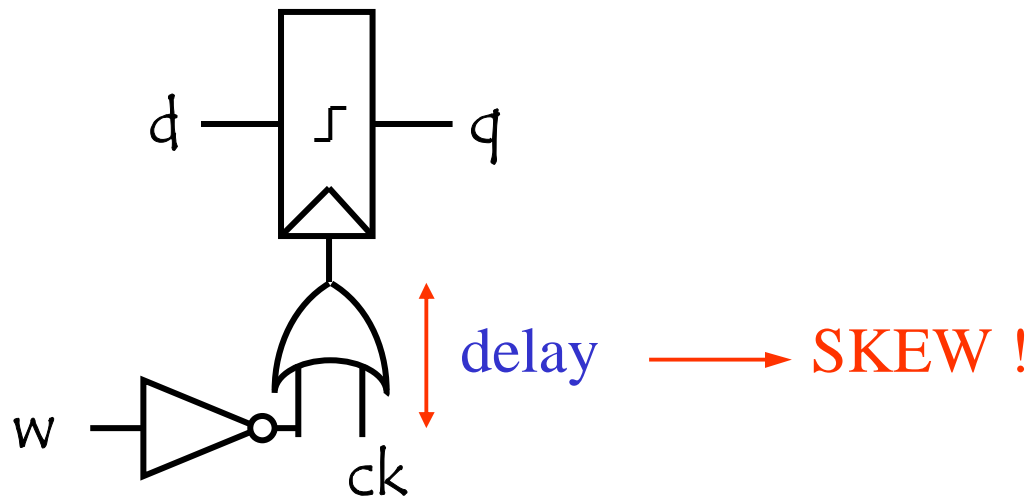
Write a data  $d$  on the rising edge of the clock  $ck$  when a condition is true (write enable)



# CMOS Circuits

## Synchronous Memory :

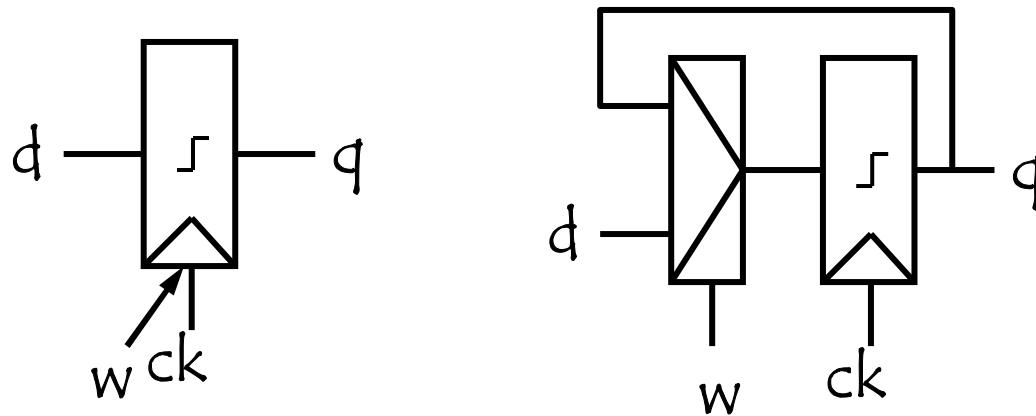
Write a data  $d$  on the rising **edge** of the clock  $ck$  when a condition is true (write enable)



# CMOS Circuits

Synchronous Memory :

Write a data  $d$  on the rising edge of the clock  $ck$  when a condition is true (write enable)

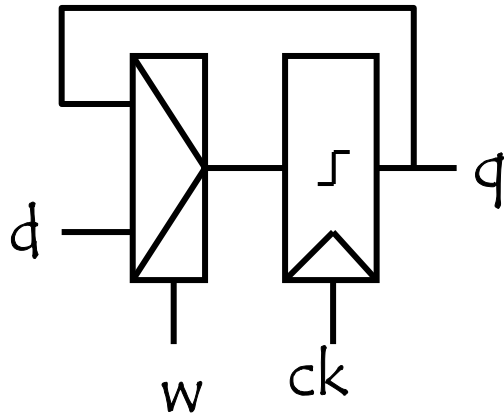




# CMOS Circuits

## Synchronous Memory :

Write a data  $d$  on the rising **edge** of the clock  $ck$  when a condition is true (write enable)



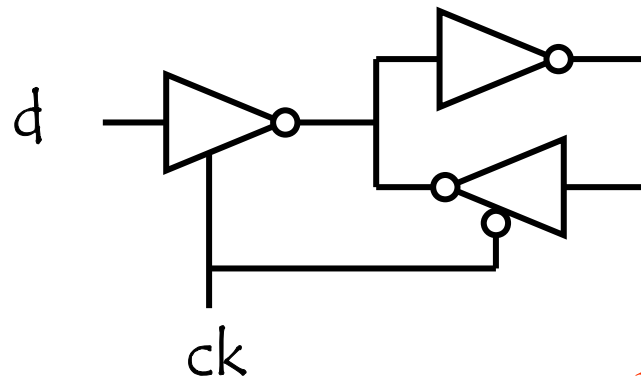
44 tr per register !!

# CMOS Circuits

## Memory : Latch

Hold a data (0 or 1)

Write a data (0 or 1)



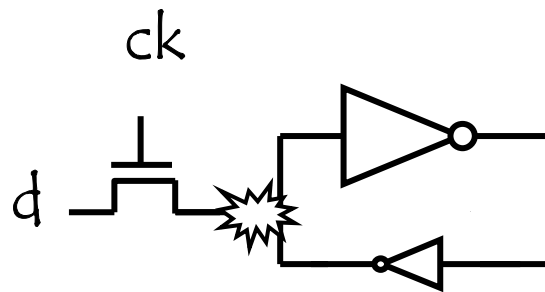
10 tr per latch

# CMOS Circuits

## Memory : Latch

Hold a data (0 or 1)

Write a data (0 or 1)



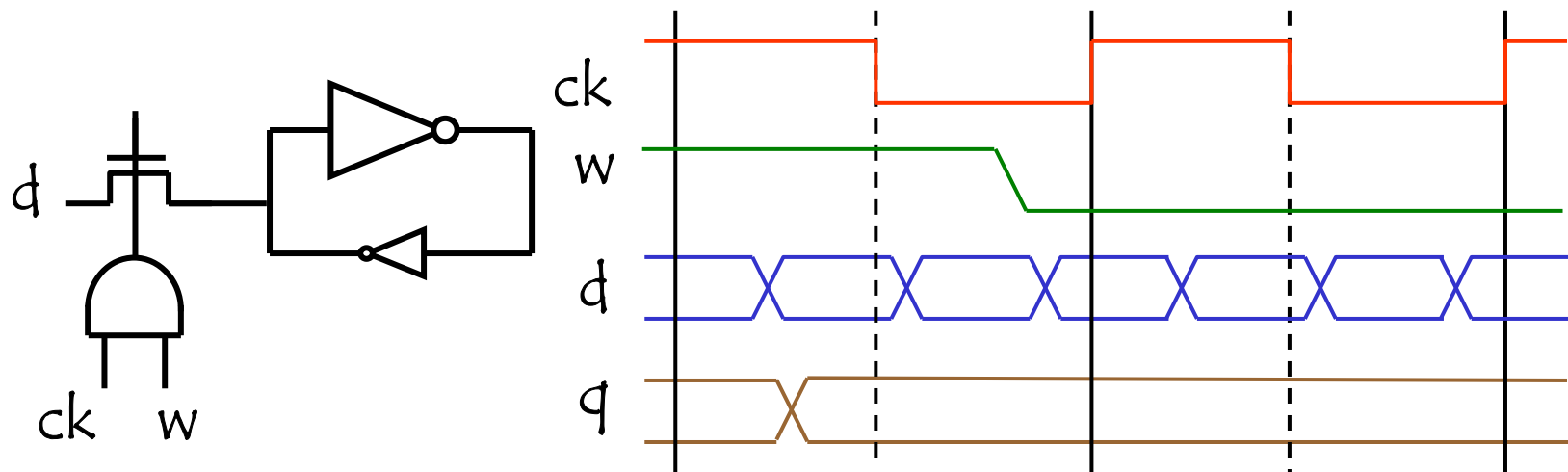
short circuit !

5 tr per latch

# CMOS Circuits

## Synchronous Memory :

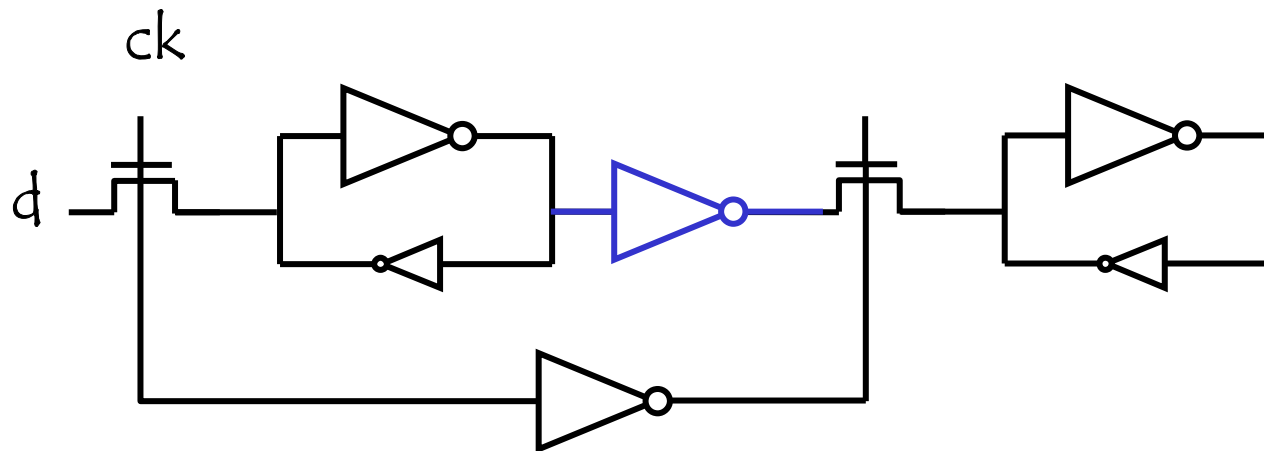
Write a data  $d$  when the clock  $ck = 1$   
when a condition is true (write enable)



# CMOS Circuits

Synchronous Memory :

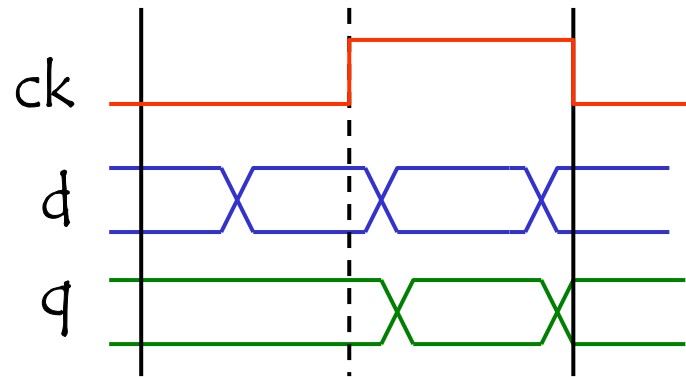
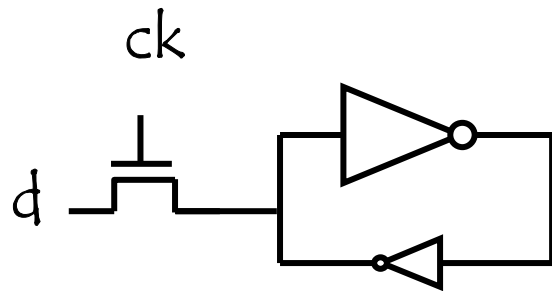
Write a data  $d$  on the falling edge of the clock  $ck$



# CMOS Circuits

Synchronous Memory :

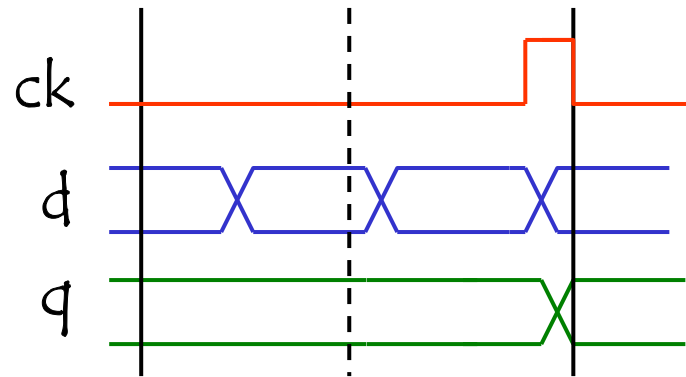
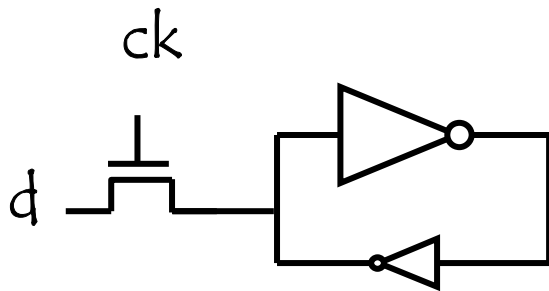
Write a data  $d$  when the clock  $ck = 1$



# CMOS Circuits

Synchronous Memory :

Write a data  $d$  on the rising edge of the clock  $ck$



# CMOS Circuits

Synchronous Memory :

Write a data  $d$  on the rising edge of the clock  $ck$

