



The Abdus Salam
International Centre for Theoretical Physics



310/1780-3

**ICTP-INFN Advanced Training Course on
FPGA and VHDL for Hardware Simulation and Synthesis
27 November - 22 December 2006**

DIGITAL DESIGN 3

***Pirouz BAZARGAN SABET
Lip 6
University Pierre et Marie Curie (VI)
Department ASIM
4, place Jussieu
75252 Paris Cedex 05
FRANCE***

These lecture notes are intended only for distribution to participants

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 in 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

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

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

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

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

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

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

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 ?

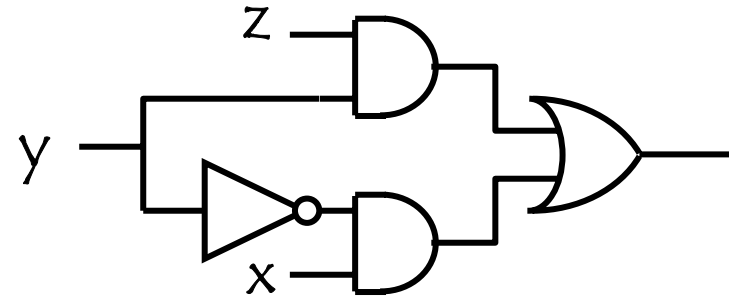
Cell library

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

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



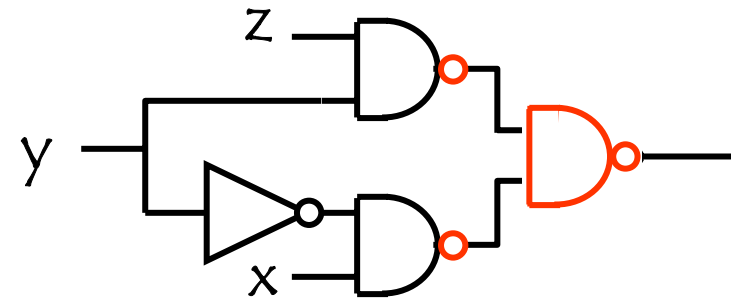
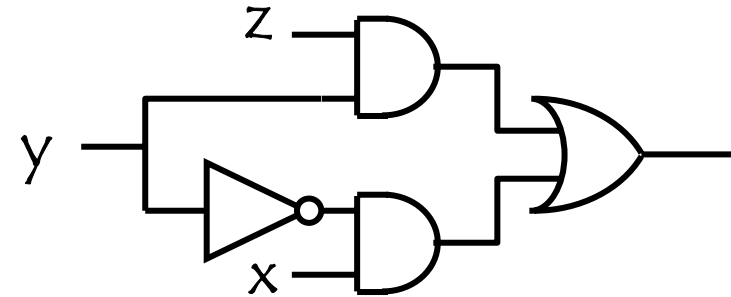
Non-inverting gates do NOT exist

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

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 :

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

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

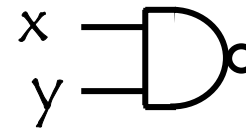
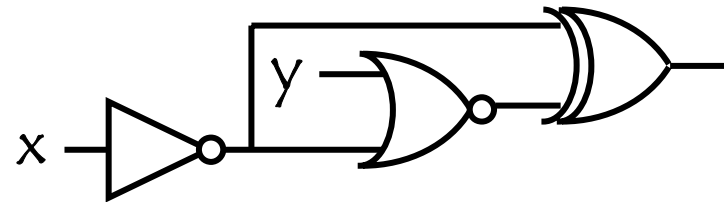
$$f = (\bar{x} + y) \cdot \bar{x} + \overline{(\bar{x} + y)} \cdot x$$

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

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

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

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



CMOS Circuits

How to implement Boolean functions
with a gate network ?

- Which expression ?

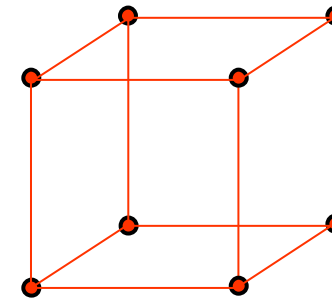
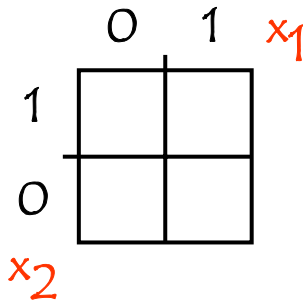
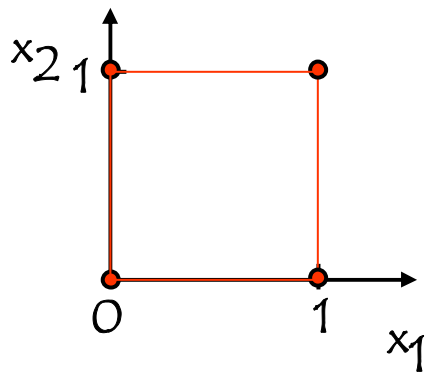


CMOS Circuits

A function can be defined by its Truth table

Karnaugh table

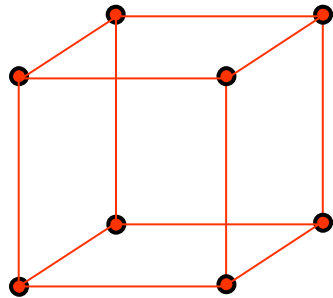
Representation of the function in a space of dimension n



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, and a blue line connects the circles in the 0 row to the circles in the 1 row.

vectors' adjacency

CMOS Circuits

Example :

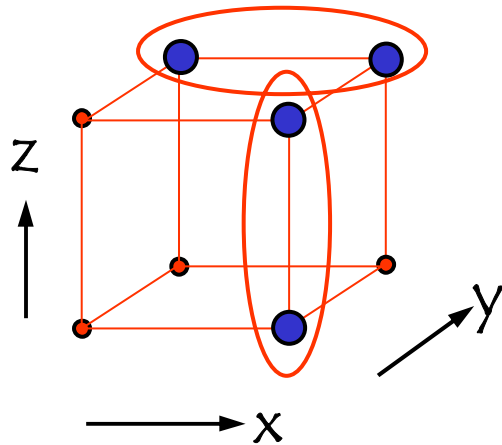
x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

	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 :

A Karnaugh map for the function $g(x,y,z)$. The map is a 2x8 grid with columns labeled 000, 001, 011, 010, 110, 111, 101, 100 and rows labeled 0 and 1. The output variable z is indicated below the rows. The map contains 1s at (0,001), (0,101), (1,001), (1,011), (1,010), (1,110), and (1,101). Blue arrows show groupings of (0,001) and (0,101) into a pair, and (1,001), (1,011), (1,010), and (1,110) into a group of four. Red circles highlight the prime implicants: $\bar{x}y$ (covering (0,001) and (0,101)), $y\bar{v}$ (covering (1,001), (1,011), (1,010), and (1,110)), and $\bar{y}v$ (covering (1,101)).

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

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

CMOS Circuits

How to implement Boolean functions
with a gate network ?

- Local optimization using Karnaugh tables

A design includes several Boolean functions

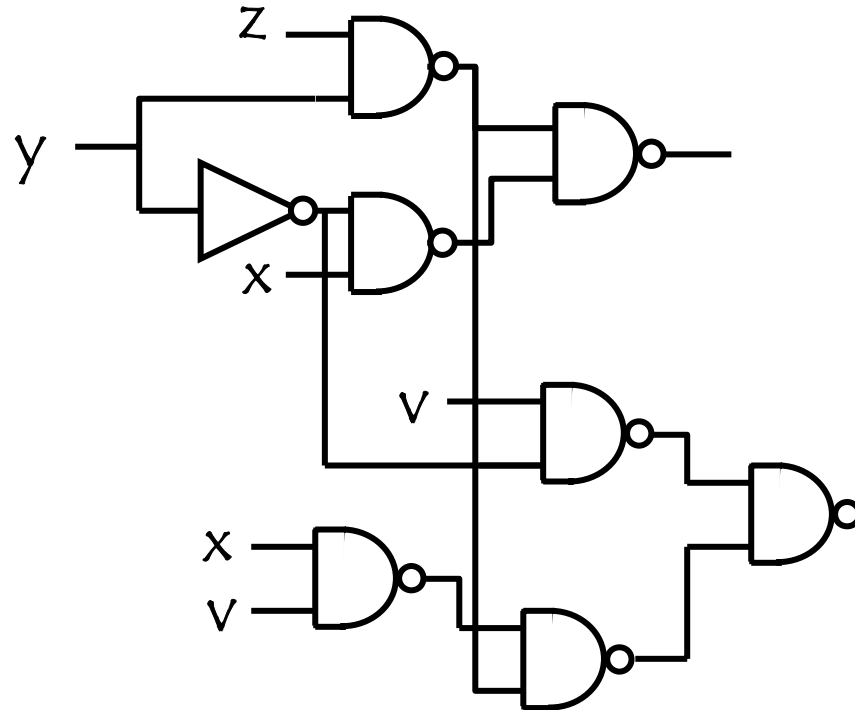


CMOS Circuits

Example :

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

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



CMOS Circuits

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**

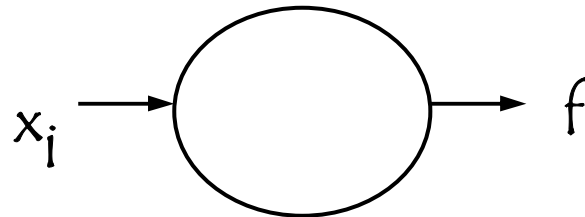
The value of the output depends on the value of the inputs **and the history**

Notion of memory



CMOS Circuits

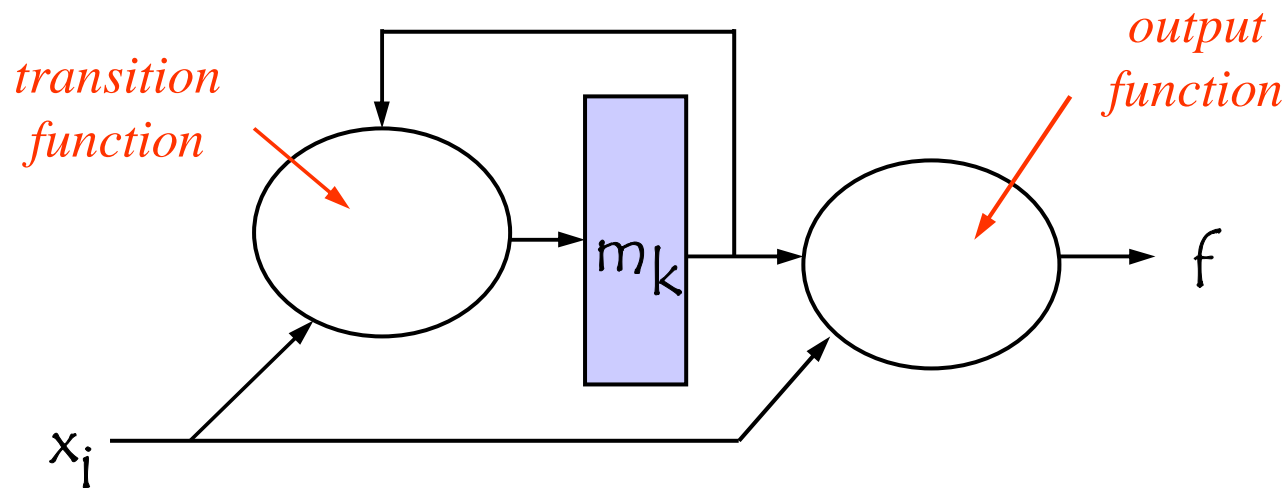
Sequential logic



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

CMOS Circuits

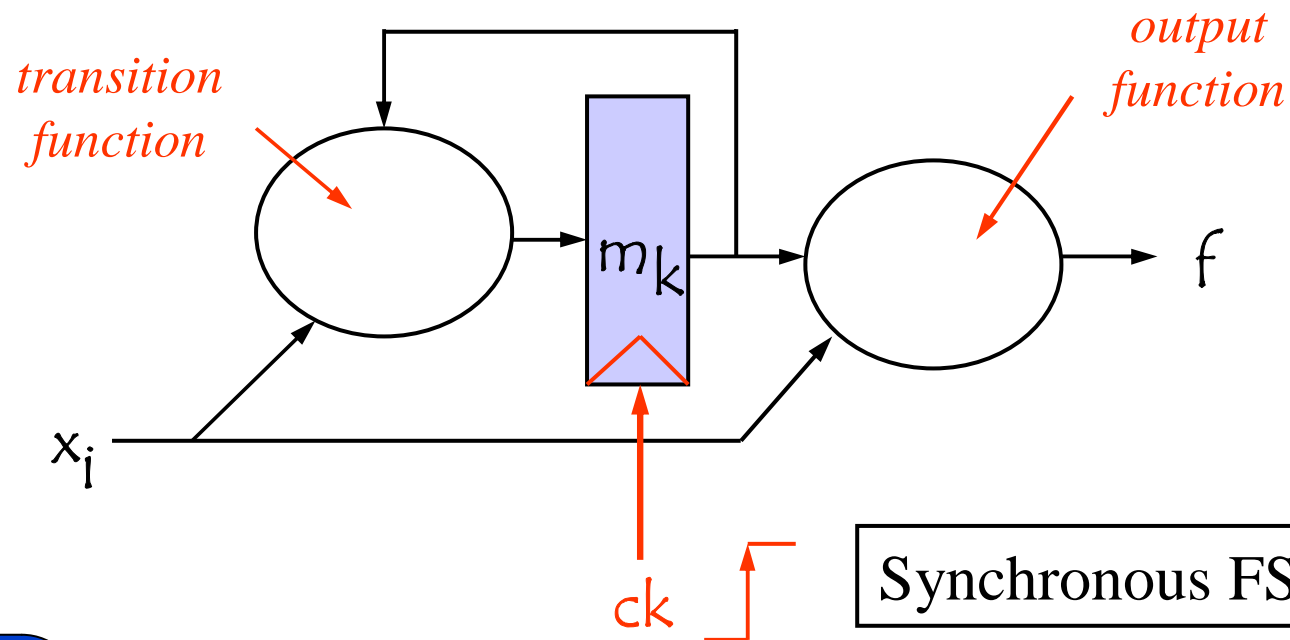
Sequential logic



Finite State Machine (FSM)

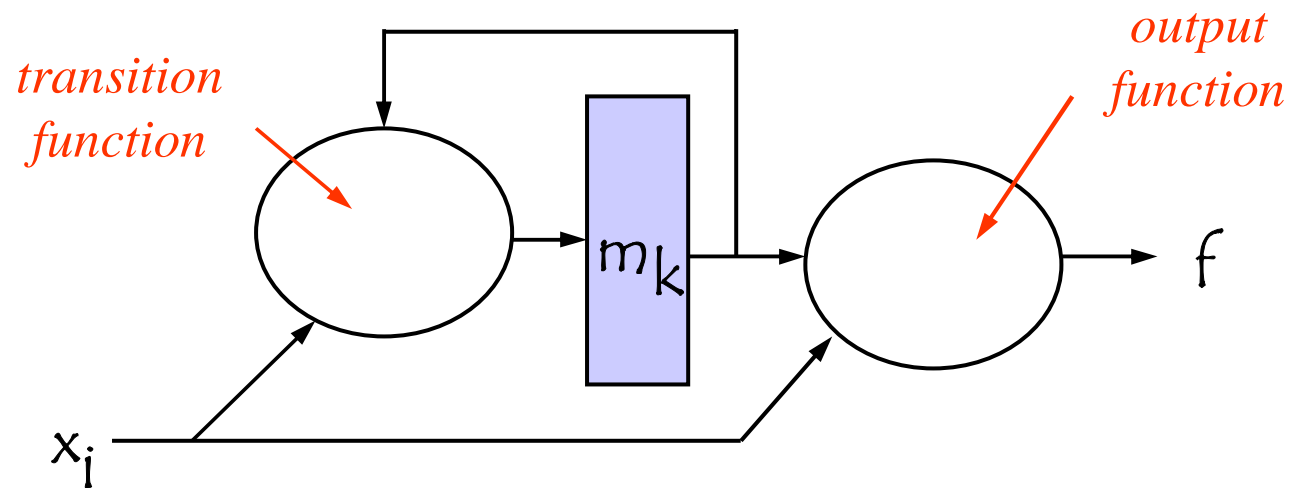
CMOS Circuits

Finite State Machine



CMOS Circuits

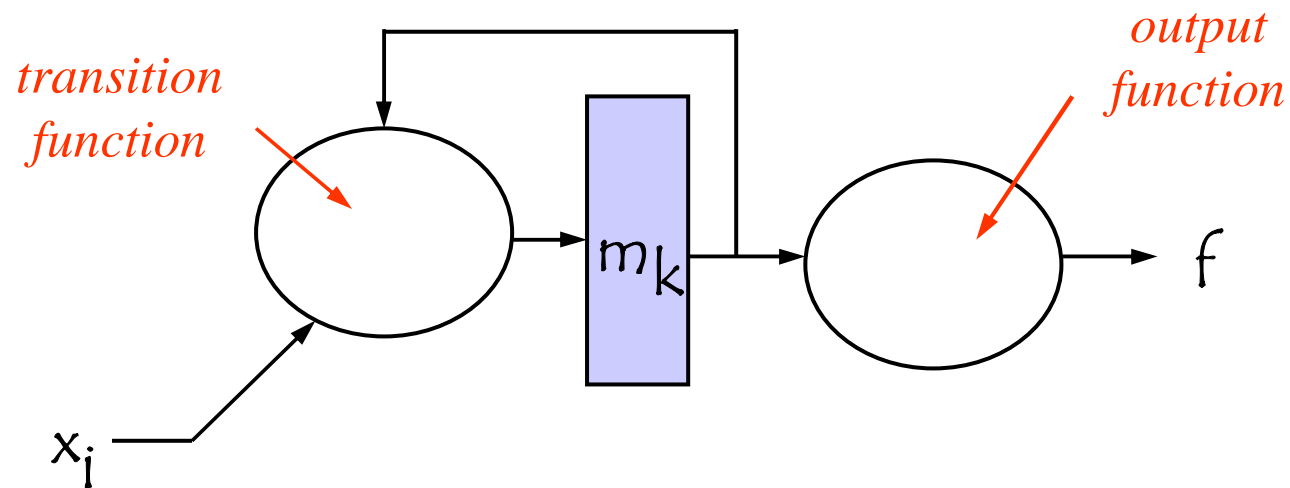
Finite State Machine



Mealy FSM

CMOS Circuits

Finite State Machine



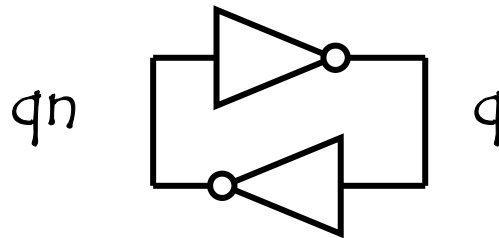
Moore FSM

CMOS Circuits

Memory :

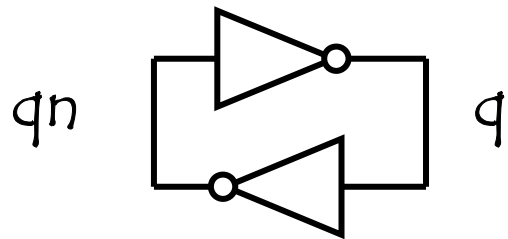
Hold a data (0 or 1)

Write a data (0 or 1)



CMOS Circuits

Memory :



$$q = \overline{qn}$$

$$qn = \overline{q}$$

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

$$f(q) = q$$

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

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

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

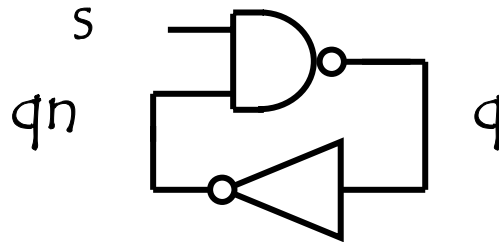
this dependency cannot be disabled

CMOS Circuits

Memory :

Hold a data (0 or 1)

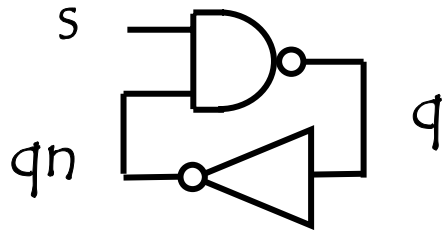
Write a data (0 or 1)



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

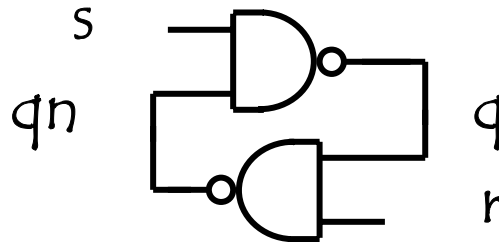
if $s = 0$ q is a combinational function

CMOS Circuits

Memory :

Hold a data (0 or 1)

Write a data (0 or 1)

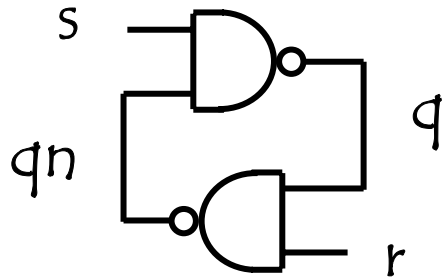


s	r	q	qn
0	1	1	0
1	0	0	1
1	1	q	qn
0	0	1	1

RS flip flop

CMOS Circuits

Memory :



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

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

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

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

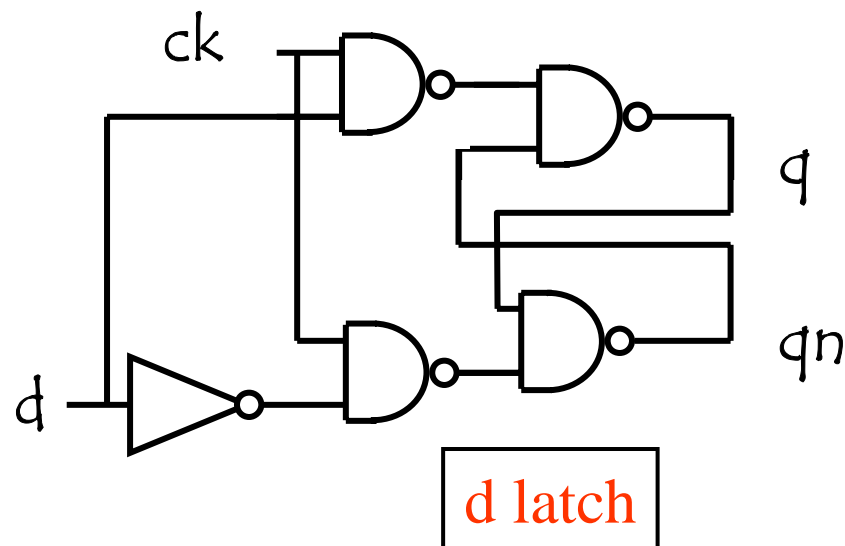
$$\frac{\partial f^+}{\partial q} = s \cdot (\overline{s} + r) = s \cdot r$$

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

CMOS Circuits

Synchronous Memory :

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



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

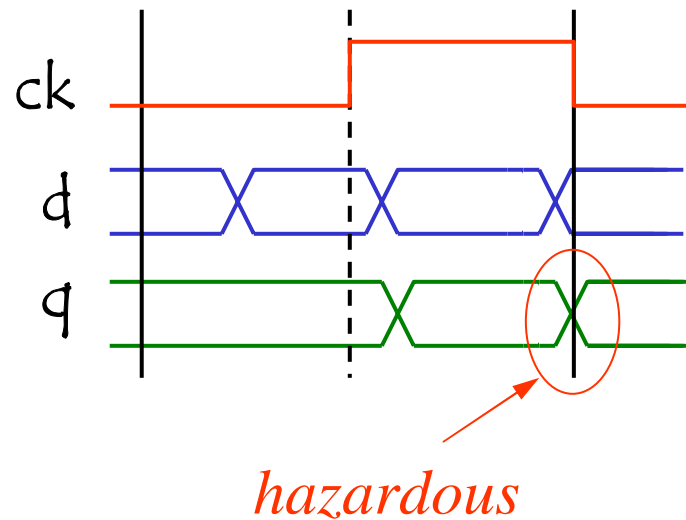
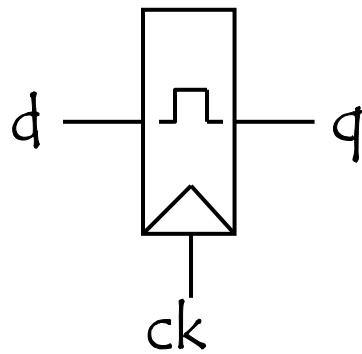
if $ck \cdot d = 1$ $s = 0$

if $ck \cdot \bar{d} = 1$ $r = 0$

CMOS Circuits

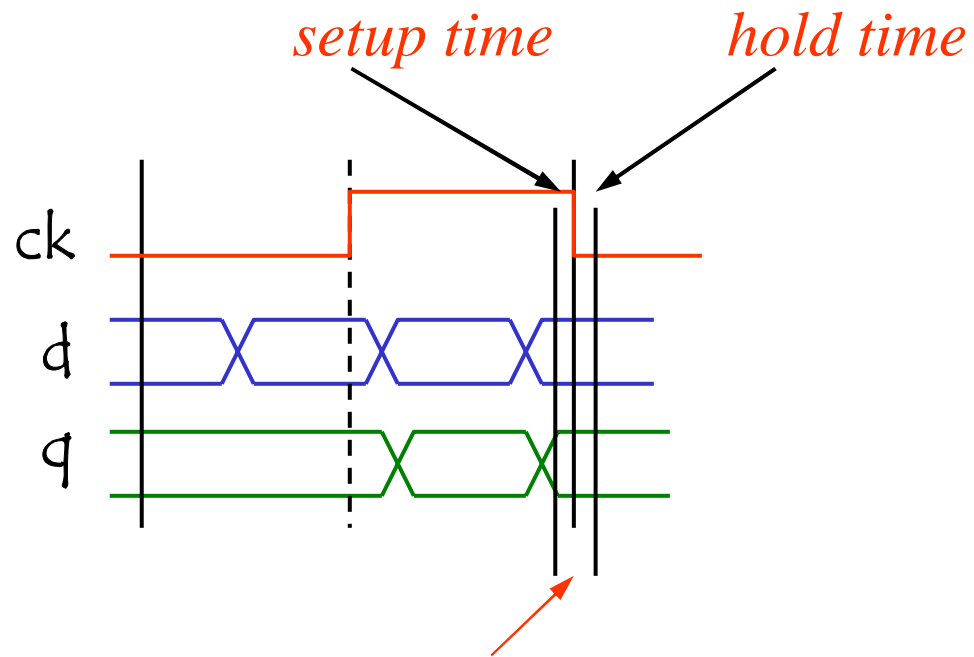
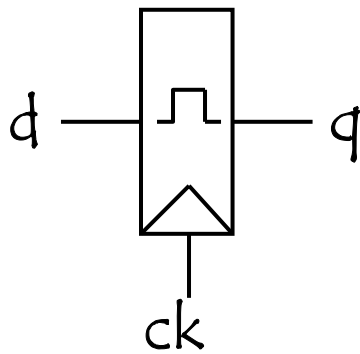
Synchronous Memory :

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



CMOS Circuits

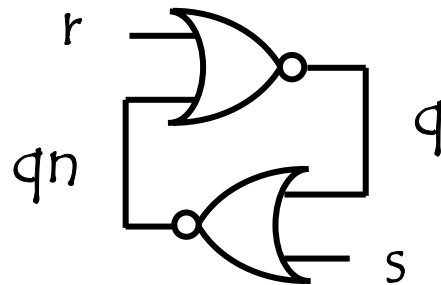
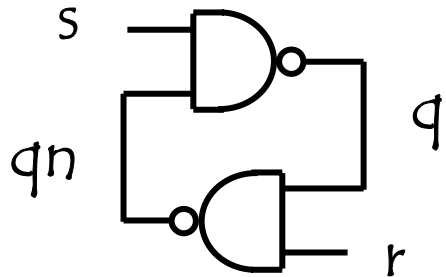
Synchronous Memory :



data should not change in this period

CMOS Circuits

Memory :



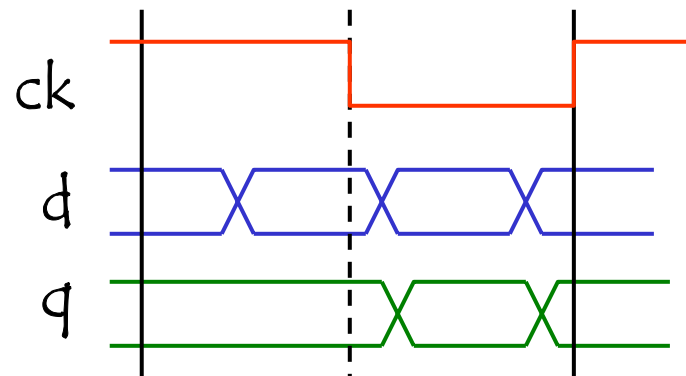
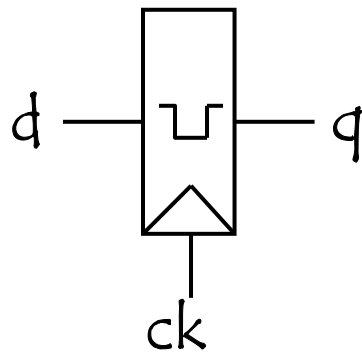
s	r	q	qn
0	1	0	1
1	0	1	0
1	1	0	0
0	0	q	qn

RS flip flop

CMOS Circuits

Synchronous Memory :

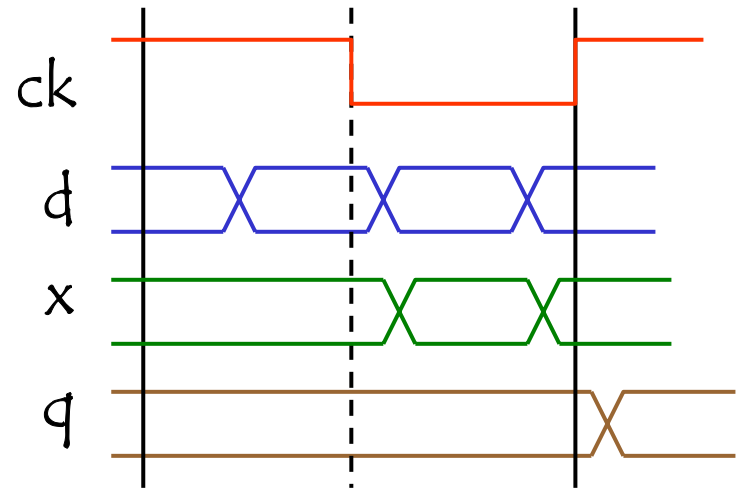
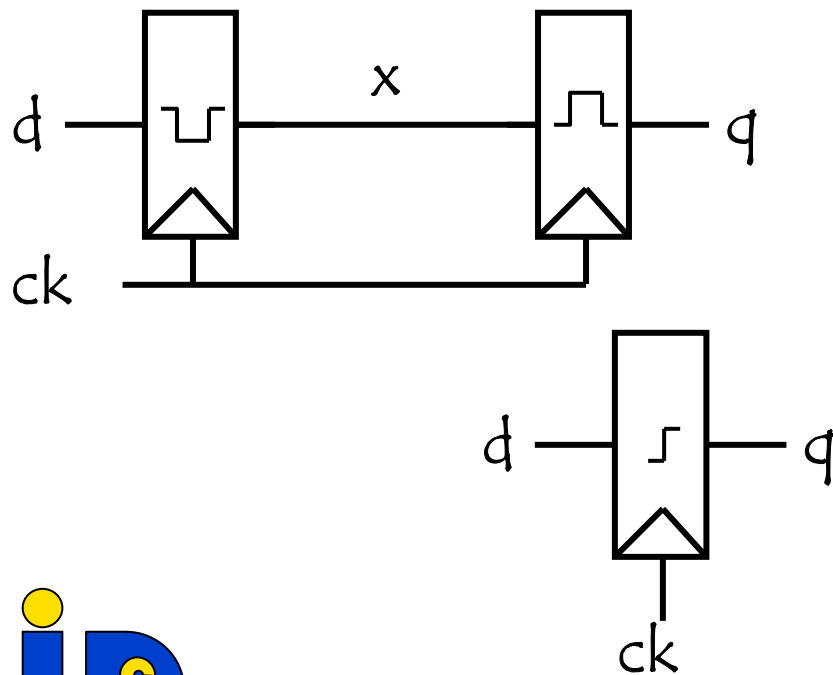
Write a data d when the clock $ck = 0$



CMOS Circuits

Synchronous Memory :

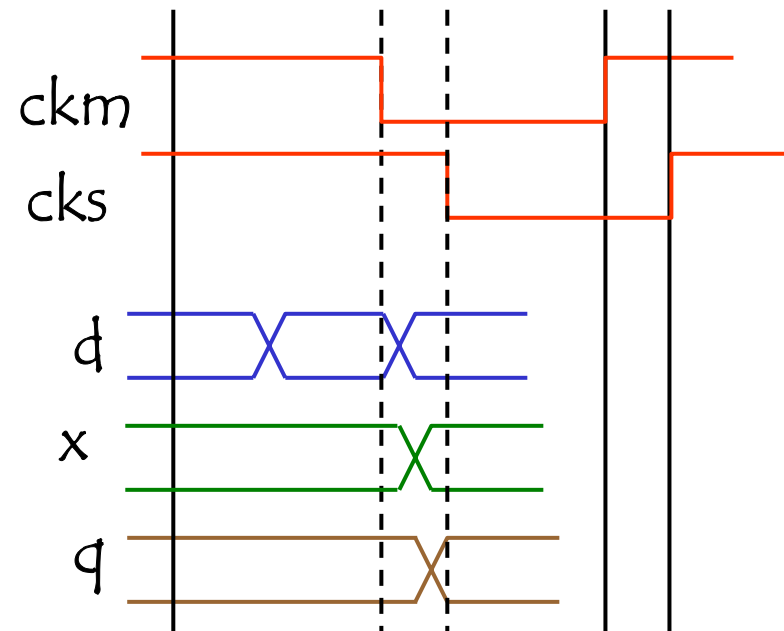
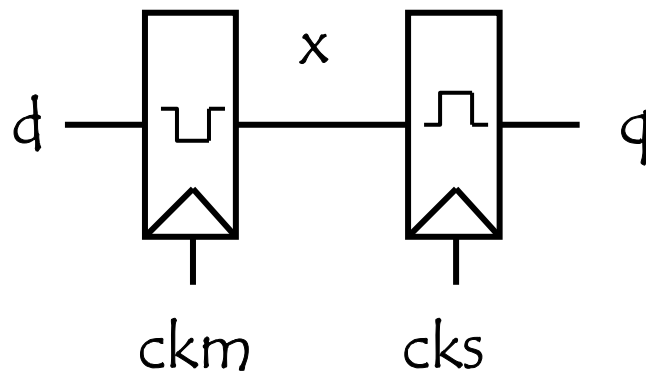
Write a data d on the rising edge of the clock ck



CMOS Circuits

Synchronous Memory :

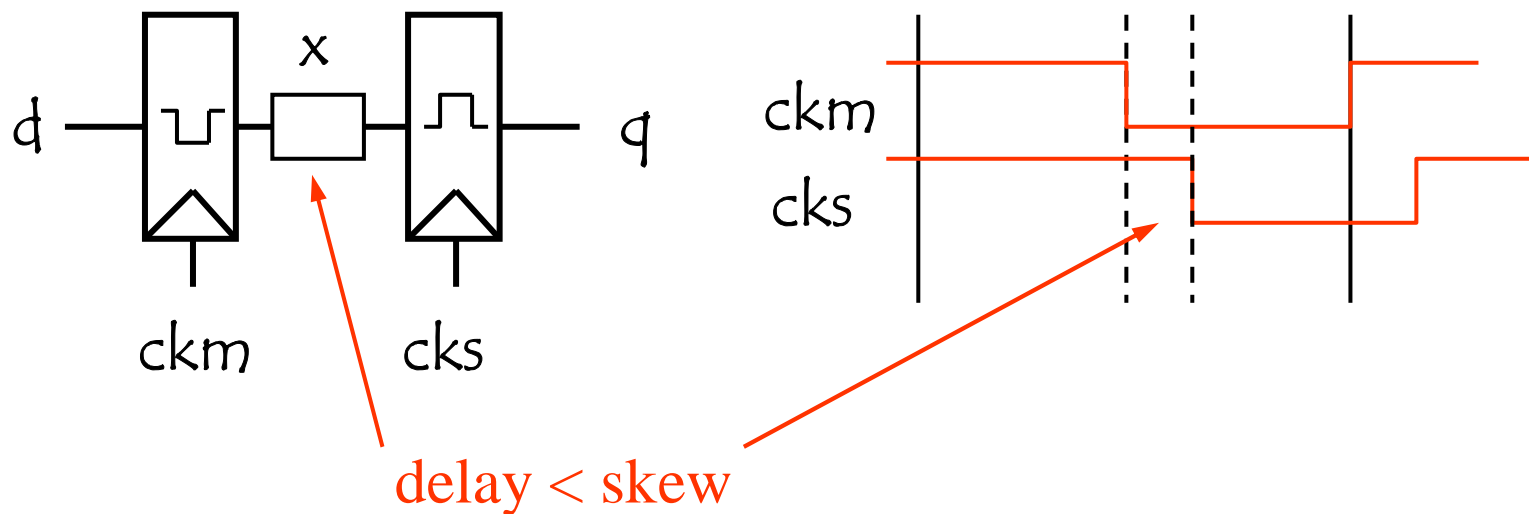
Write a data d on the rising edge of the clock ck



CMOS Circuits

Synchronous Memory :

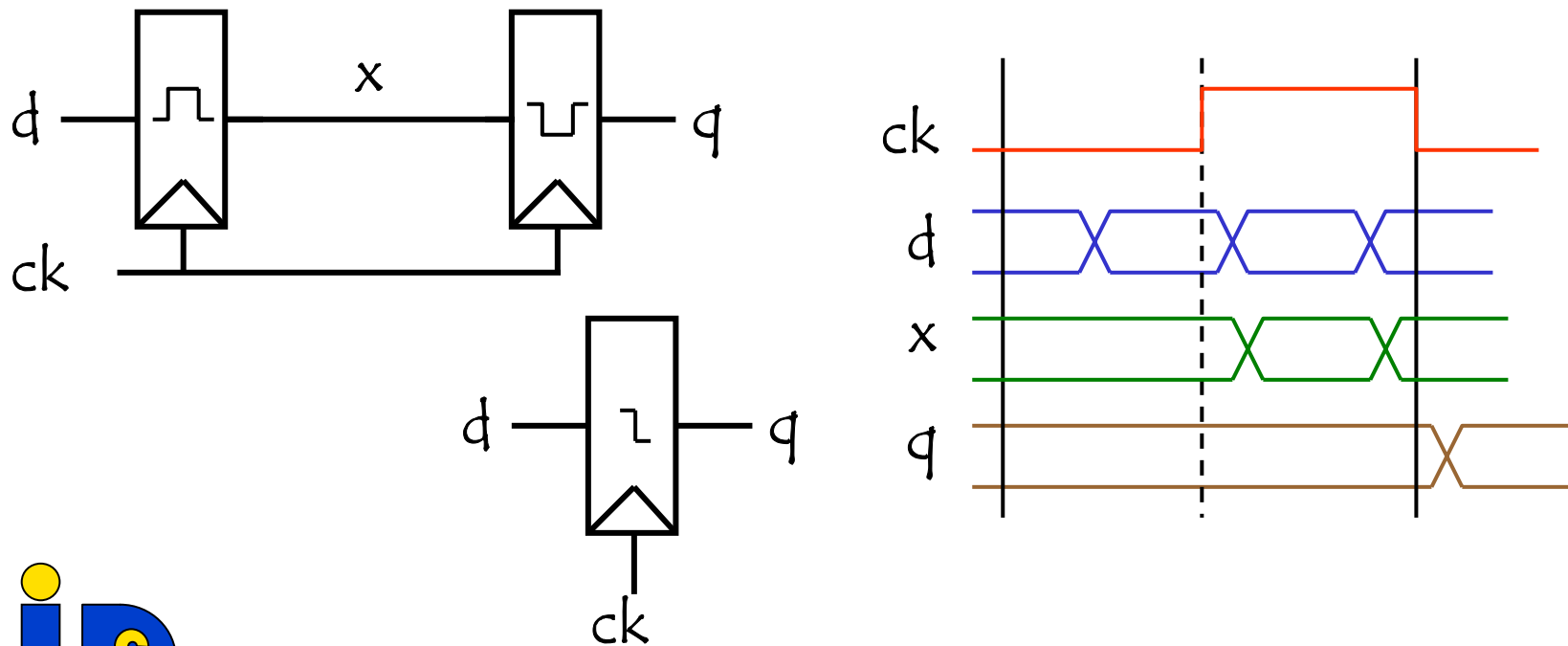
Write a data d on the rising edge of the clock ck



CMOS Circuits

Synchronous Memory :

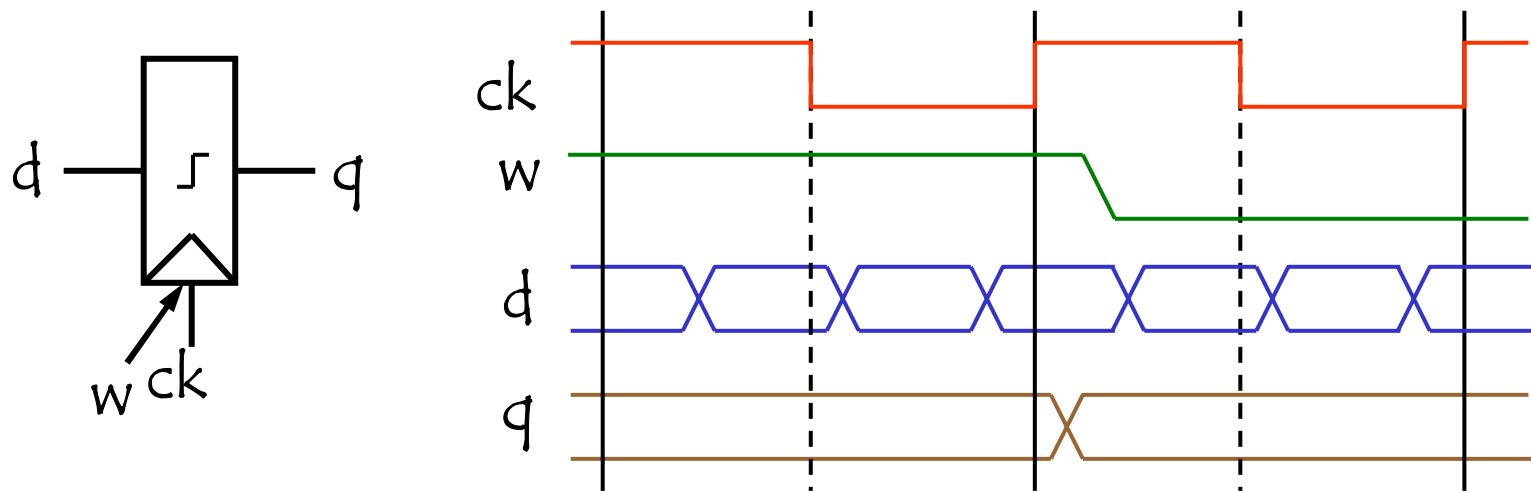
Write a data d on the falling edge of the clock ck



CMOS Circuits

Synchronous Memory :

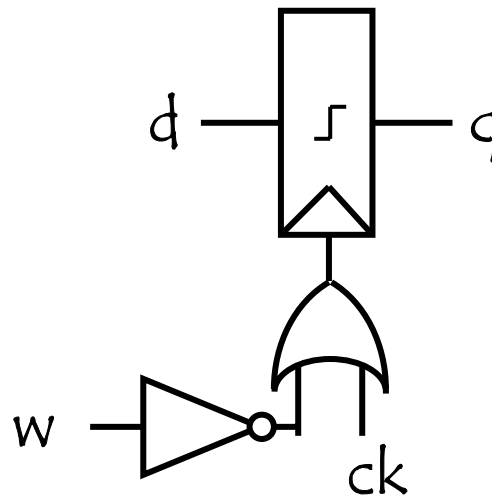
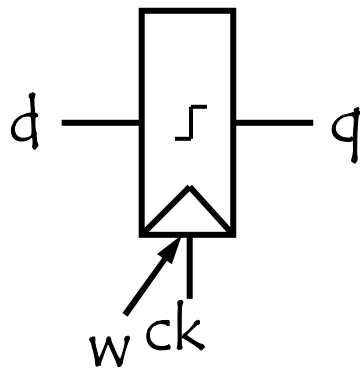
Write a data d on the rising **edge** of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

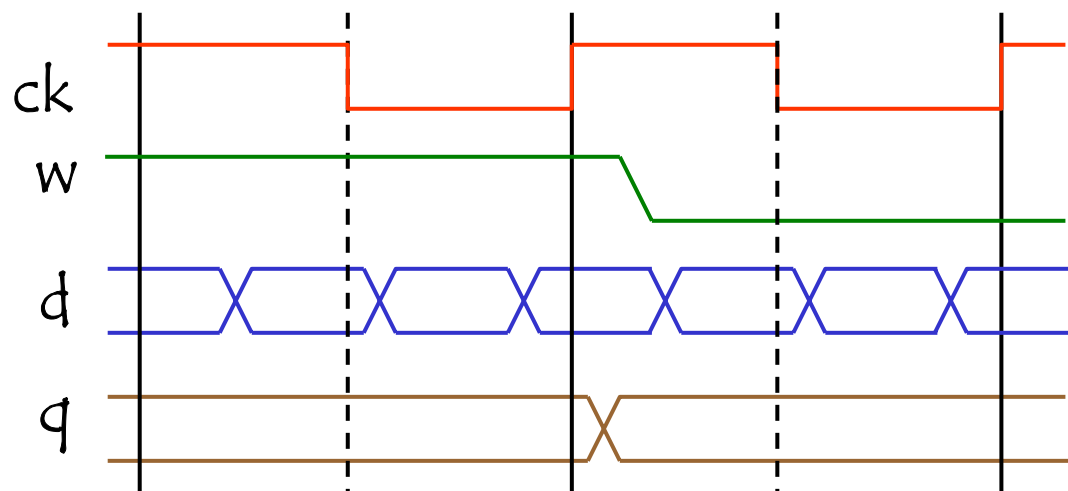
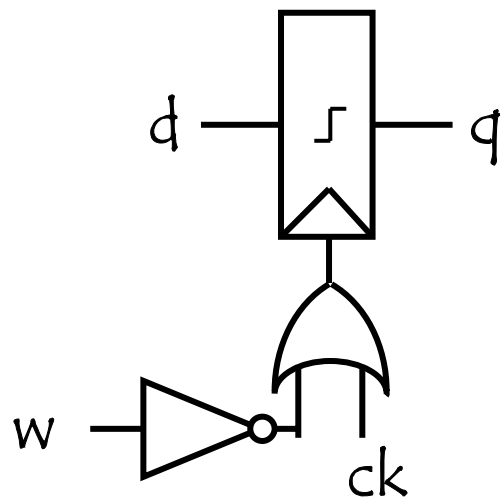
Write a data d on the rising edge of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

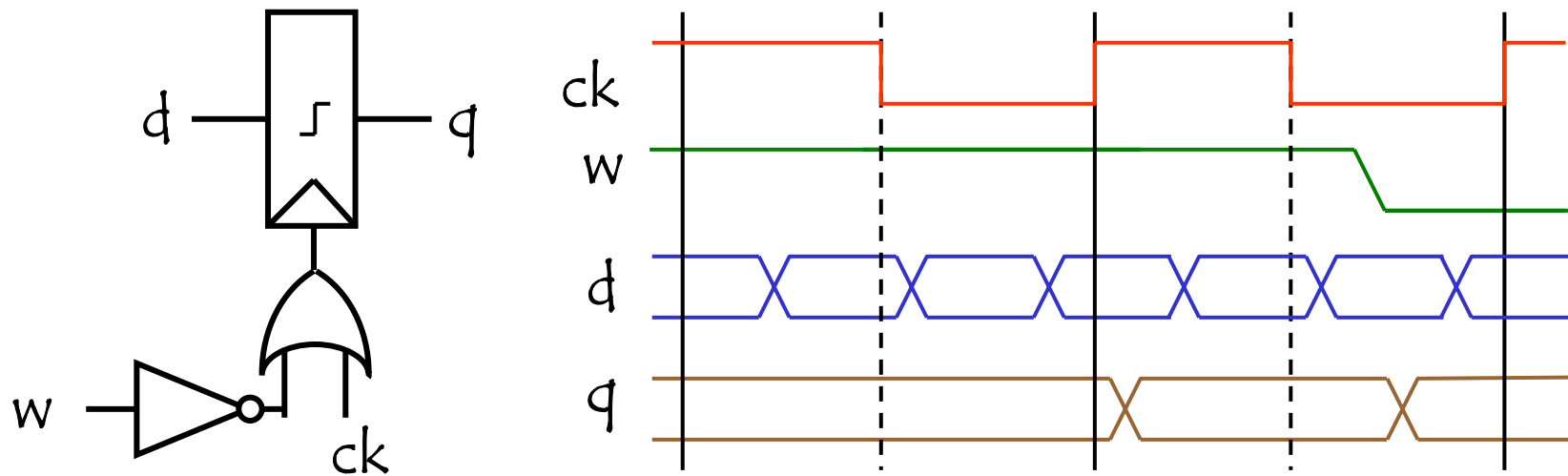
Write a data d on the rising edge of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

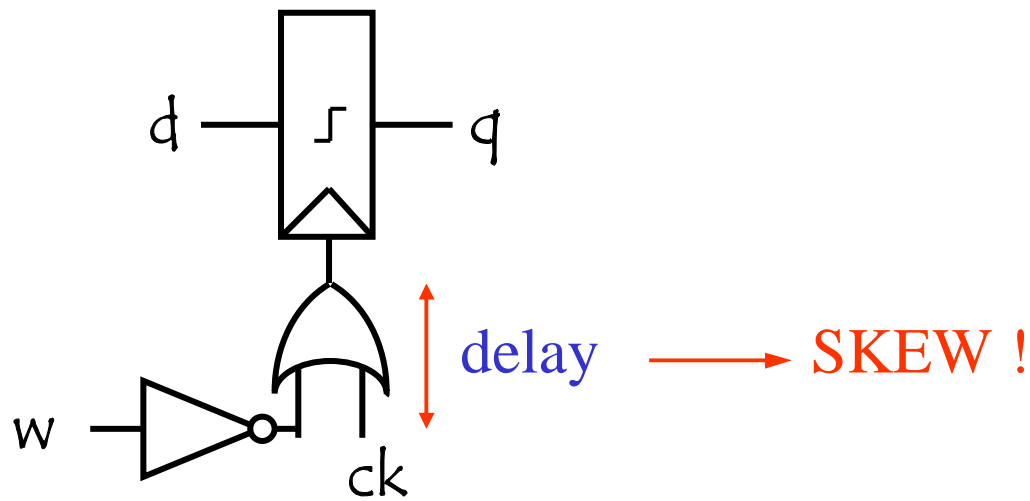
Write a data d on the rising **edge** of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

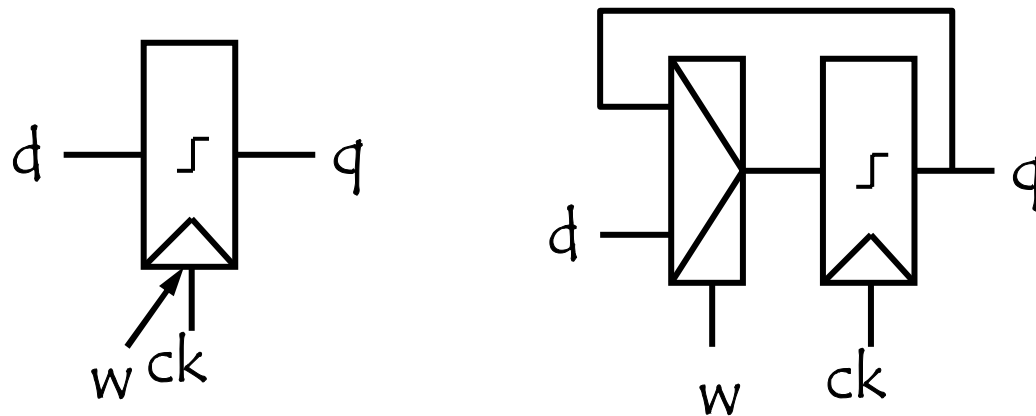
Write a data d on the rising **edge** of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

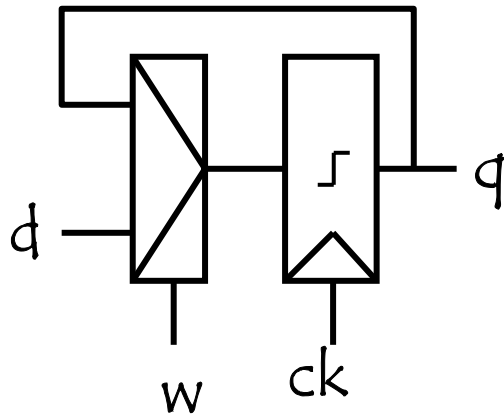
Write a data d on the rising **edge** of the clock ck when a condition is true (write enable)



CMOS Circuits

Synchronous Memory :

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



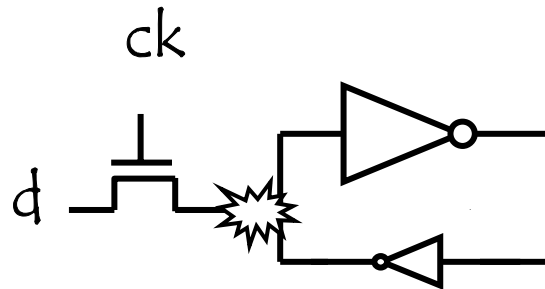
44 tr per register !!

CMOS Circuits

Memory :

Hold a data (0 or 1)

Write a data (0 or 1)

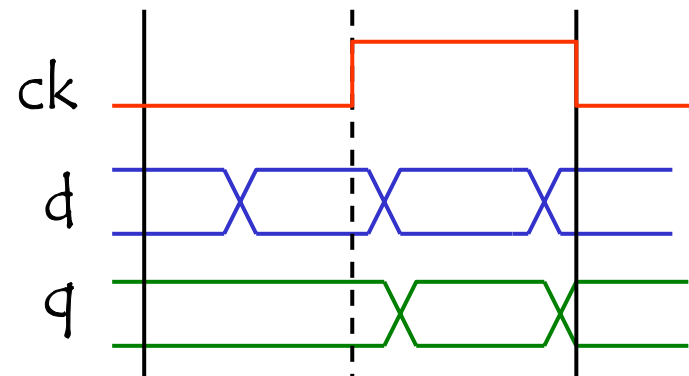
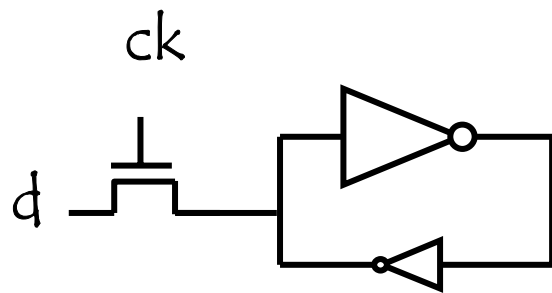


short circuit !

CMOS Circuits

Synchronous Memory :

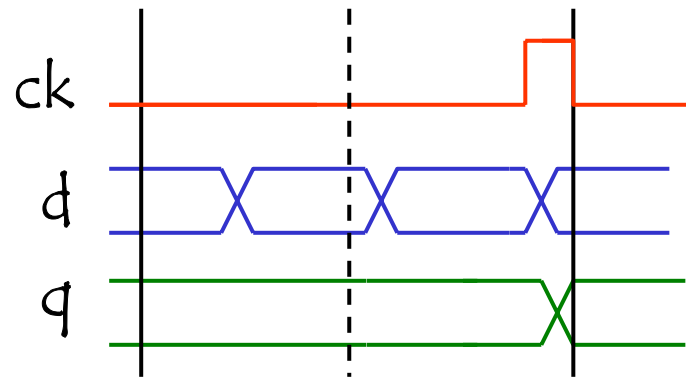
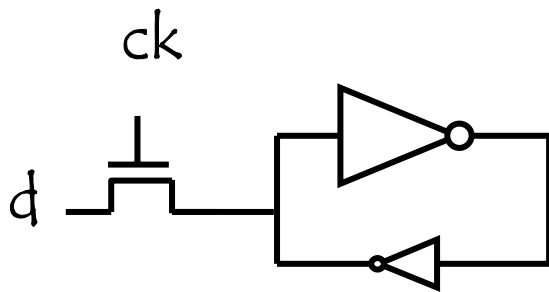
Write a data d when the clock $ck = 1$



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

