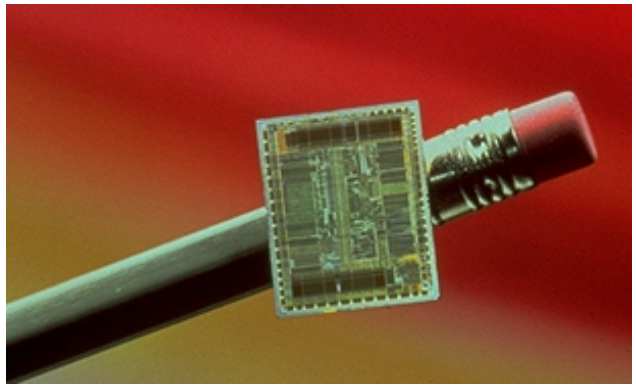


ICS904/CD2IC : Cell Design For Digital Integrated Circuits

L2 : Structural design of digital circuits (1/2))

Yves MATHIEU
yves.mathieu@telecom-paristech.fr

Outline

Outline

Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

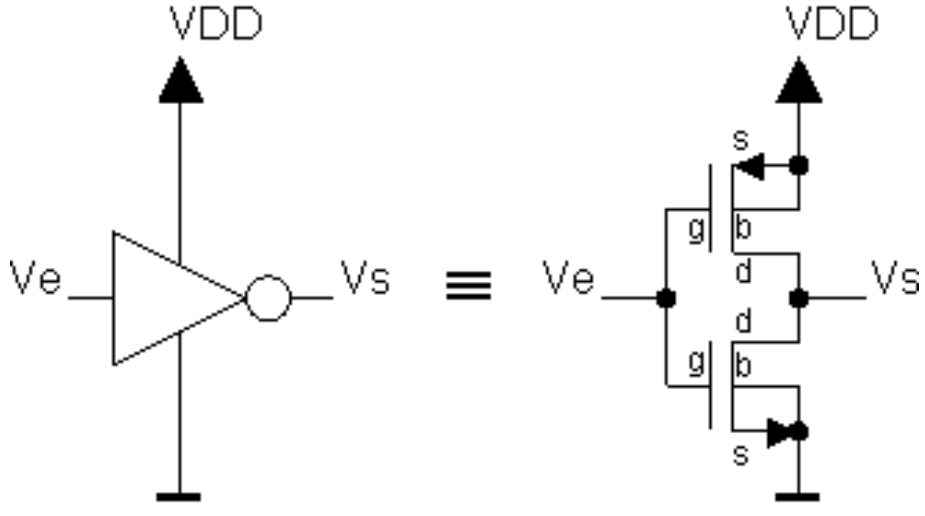
Integrated logic

- How to build logic functions with electronic components ?
- Many possible implementations . . .
  - derived from the manipulation of logical expressions.
  - derived from component properties (MOS transistors, bipolar . . .)
- Arbitrary choice of a physical variable to represent the logic states
- Many paths explored in the years 70-90
- Major classes of solutions stabilized since.
- But research still continues . . .

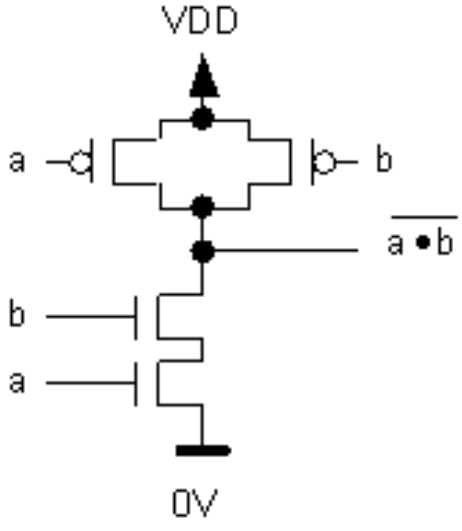
Outline

CMOS invertor

- The selected supply voltage is the reference for defining the Boolean signal
- Harnessing the symetrical behavior of NMOS and PMOS transistors.



The 2 input NAND gate



a	b	Output
0	0	1
0	1	1
1	0	1
1	1	0

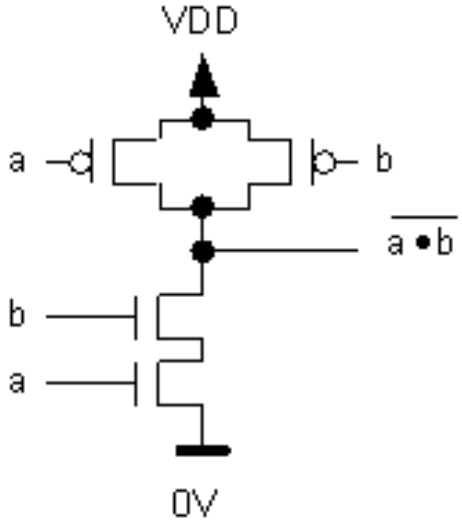
NMOS network

- ON if the two NMOS transistors are "ON"
- OFF if one of the two NMOS transistors is "OFF"

PMOS network

- ON if one of the two PMOS transistors is "ON"
- OFF if the two PMOS transistors are "OFF"

The 2 input NAND gate



a	b	Output
0	0	1
0	1	1
1	0	1
1	1	0

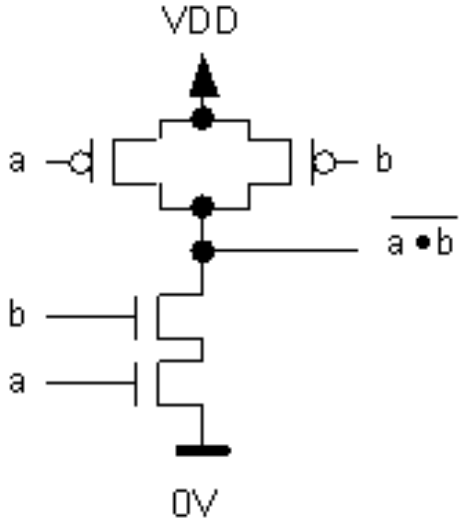
NMOS network

- ON if the two NMOS transistors are "ON"
- OFF if one of the two NMOS transistors is "OFF"

PMOS network

- ON if one of the two PMOS transistors is "ON"
- OFF if the two PMOS transistors are "OFF"

The 2 input NAND gate



a	b	Output
0	0	1
0	1	1
1	0	1
1	1	0

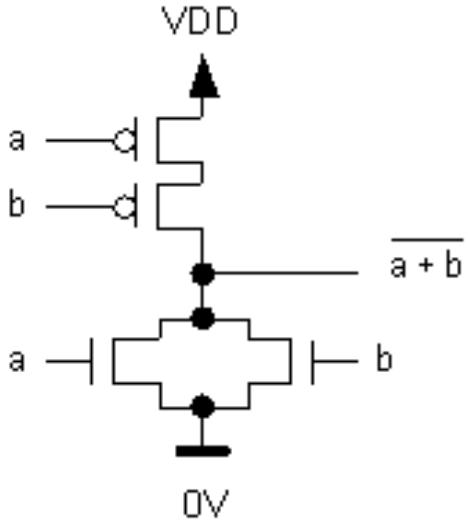
NMOS network

- ON if the two NMOS transistors are "ON"
- OFF if one of the two NMOS transistors is "OFF"

PMOS network

- ON if one of the two PMOS transistors is "ON"
- OFF if the two PMOS transistors are "OFF"

The two input NOR gate



a	b	Output
0	0	1
0	1	0
1	0	0
1	1	0

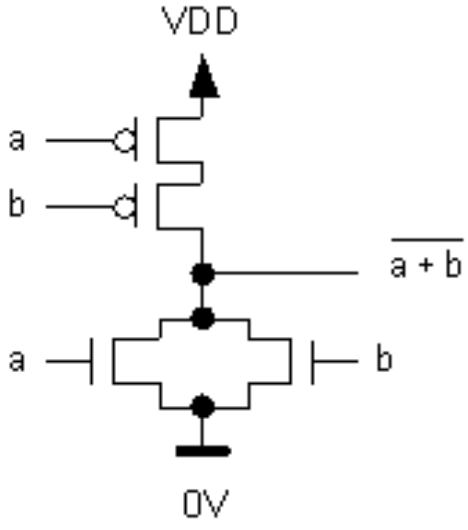
NMOS network

- ON if one of the two NMOS transistor is "ON"
- OFF if the two NMOS transistors are "OFF"

PMOS network

- ON if the two PMOS transistors are "ON"
- OFF if One of the two PMOS transistor is "OFF"

The two input NOR gate



a	b	Output
0	0	1
0	1	0
1	0	0
1	1	0

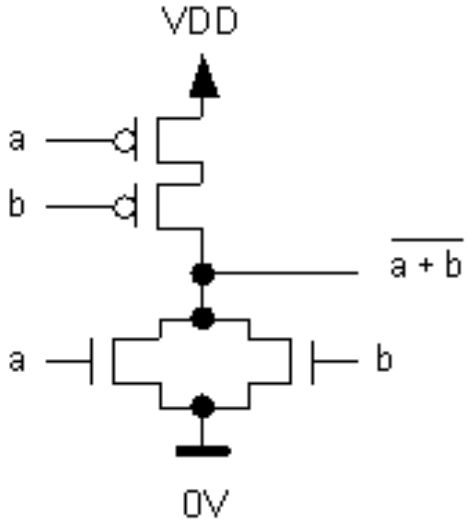
NMOS network

- ON if one of the two NMOS transistor is "ON"
- OFF if the two NMOS transistors are "OFF"

PMOS network

- ON if the two PMOS transistors are "ON"
- OFF if One of the two PMOS transistor is "OFF"

The two input NOR gate



a	b	Output
0	0	1
0	1	0
1	0	0
1	1	0

NMOS network

- ON if one of the two NMOS transistor is "ON"
- OFF if the two NMOS transistors are "OFF"

PMOS network

- ON if the two PMOS transistors are "ON"
- OFF if One of the two PMOS transistor is "OFF"

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Generalization to other boolean functions

- Given the truth-table of boolean function.
- PMOS networks are used for the logic ones of the boolean function.
- NMOS networks are used for the logic zeros of the boolean function.
- A one to one NMOS/PMOS structure
- Implementable gates have a function of the form $F(x_0, x_1, \ldots, x_n) = \overline{\sum \prod x_i}$
- Other boolean functions are assemblies of these primitive gates.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- First, build the NMOS network :
  - Express the $\overline{F(x_0, x_1, \ldots, x_n)}$ function in the form $\sum \prod x_i$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of NMOS transistors (or network of NMOS transistors).
  - Remaining $\sum$ match with NMOS transistors (or network of NMOS transistors) connected in parallel.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- First, build the NMOS network :
  - Express the $\overline{F(x_0, x_1, \ldots, x_n)}$ function in the form $\sum \prod x_i$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of NMOS transistors (or network of NMOS transistors).
  - Remaining $\sum$ match with NMOS transistors (or network of NMOS transistors) connected in parallel.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- First, build the NMOS network :
  - Express the $\overline{F(x_0, x_1, \ldots, x_n)}$ function in the form $\sum \prod x_i$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of NMOS transistors (or network of NMOS transistors).
  - Remaining $\sum$ match with NMOS transistors (or network of NMOS transistors) connected in parallel.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- First, build the NMOS network :
  - Express the $\overline{F(x_0, x_1, \ldots, x_n)}$ function in the form $\sum \prod x_i$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of NMOS transistors (or network of NMOS transistors).
  - Remaining $\sum$ match with NMOS transistors (or network of NMOS transistors) connected in parallel.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- First, build the NMOS network :
  - Express the $\overline{F(x_0, x_1, \ldots, x_n)}$ function in the form $\sum \prod x_i$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of NMOS transistors (or network of NMOS transistors).
  - Remaining $\sum$ match with NMOS transistors (or network of NMOS transistors) connected in parallel.

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Second, build the PMOS network :
  - Express the $F(x_0, x_1, \ldots, x_n)$ function in the form $\sum \prod \overline{x_i}$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of PMOS transistors (or network of PMOS transistors).
  - Remaining $\sum$ match with PMOS transistors (or network of PMOS transistors connected in parallel).

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Second, build the PMOS network :
  - Express the $F(x_0, x_1, \ldots, x_n)$ function in the form $\sum \prod \overline{x_i}$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of PMOS transistors (or network of PMOS transistors).
  - Remaining $\sum$ match with PMOS transistors (or network of PMOS transistors connected in parallel).

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Second, build the PMOS network :
  - Express the $F(x_0, x_1, \ldots, x_n)$ function in the form $\sum \prod \overline{x_i}$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of PMOS transistors (or network of PMOS transistors).
  - Remaining $\sum$ match with PMOS transistors (or network of PMOS transistors connected in parallel).

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Second, build the PMOS network :
  - Express the $F(x_0, x_1, \ldots, x_n)$ function in the form $\sum \prod \overline{x_i}$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of PMOS transistors (or network of PMOS transistors).
  - Remaining $\sum$ match with PMOS transistors (or network of PMOS transistors connected in parallel).

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Second, build the PMOS network :
  - Express the $F(x_0, x_1, \ldots, x_n)$ function in the form $\sum \prod \overline{x_i}$
  - Perform any factorisation/simplification.
  - Remaining $\prod$ match with series of PMOS transistors (or network of PMOS transistors).
  - Remaining $\sum$ match with PMOS transistors (or network of PMOS transistors connected in parallel).

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Two remarks :
  - Simplest CMOS gates are inverting gates (NAND is smaller than AND)
  - Dual PMOS and NMOS networks may be used :
    - Use parallel PMOS network when series of NMOS networks are used
    - Use series of PMOS network when parallel NMOS networks are used
  - This latter method often leads to a non optimal gate (from electrical point of view)

CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Two remarks :
  - Simplest CMOS gates are inverting gates (NAND is smaller than AND)
  - Dual PMOS and NMOS networks may be used :
    - Use parallel PMOS network when series of NMOS networks are used
    - Use series of PMOS network when parallel NMOS networks are used
  - This latter method often leads to a non optimal gate (from electrical point of view)

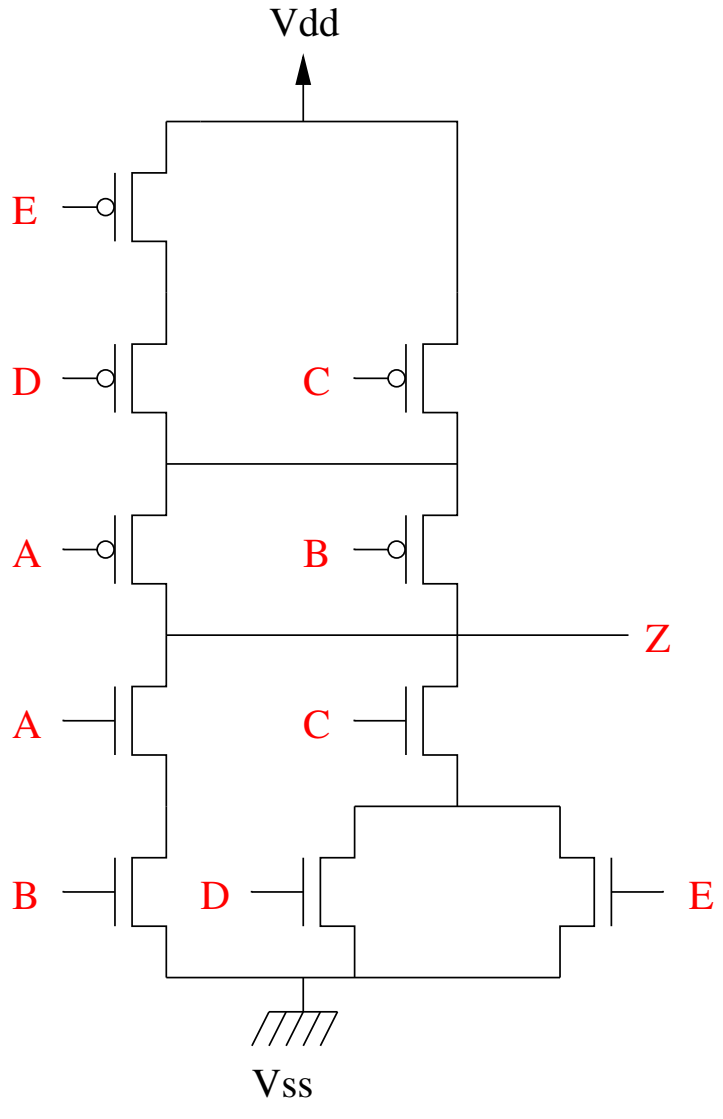
CMOS logic

Construction methods for functions of the form $F = \overline{\sum \prod x_i}$

- Two remarks :
  - Simplest CMOS gates are inverting gates (NAND is smaller than AND)
  - Dual PMOS and NMOS networks may be used :
    - Use parallel PMOS network when series of NMOS networks are used
    - Use series of PMOS network when parallel NMOS networks are used
  - This latter method often leads to a non optimal gate (from electrical point of view)

CMOS logic

$$Z = \overline{A \cdot B + C \cdot (D + E)}$$



CMOS logic

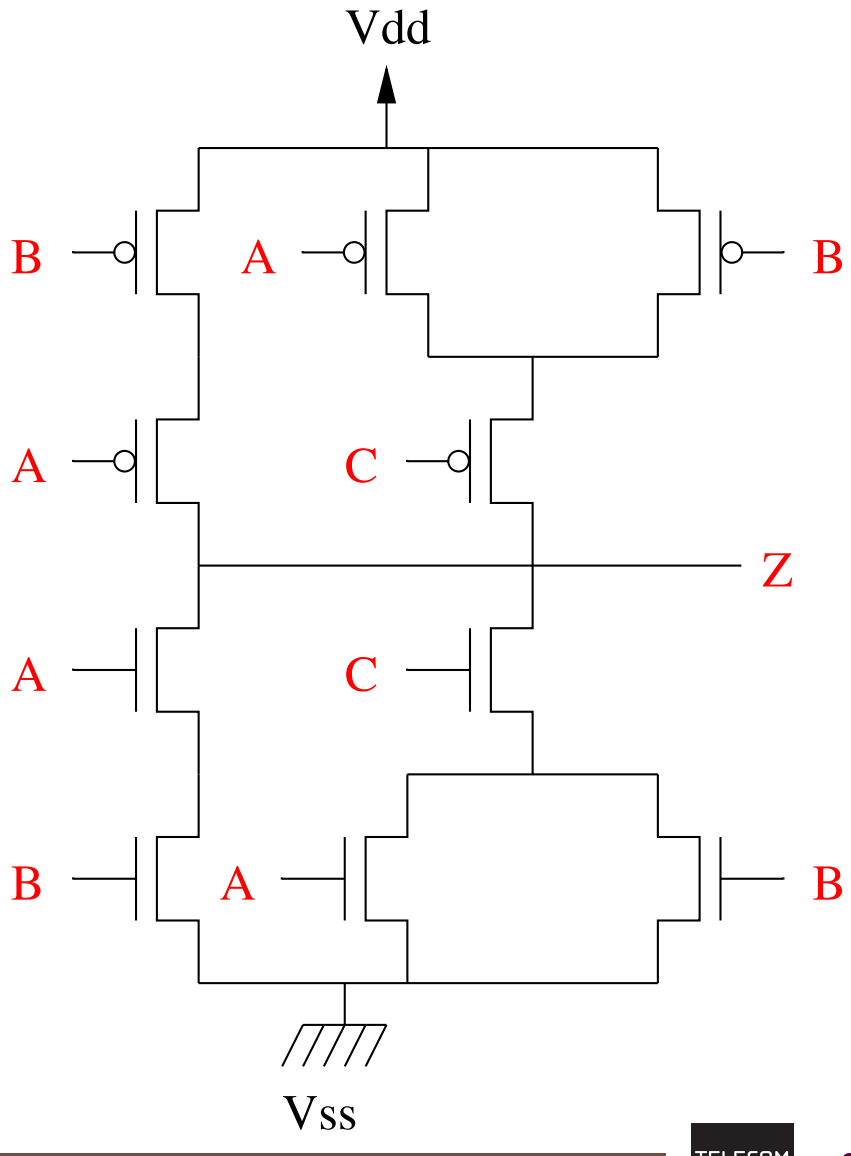
Optimum network : $F = \overline{A \cdot B + B \cdot C + A \cdot C}$

- Q1 : Build the gate using optimum optimisation methods
- Q2 : Exchange "C" NMOS transistor with "A/B" NMOS network : Does it make any difference ?
- Q3 : Build the gate using dual PMOS and NMOS networks
- Q4 : What are the potentials flaws of this later gate ?

CMOS logic

Optimum network : $F = \overline{A \cdot B + B \cdot C + A \cdot C}$

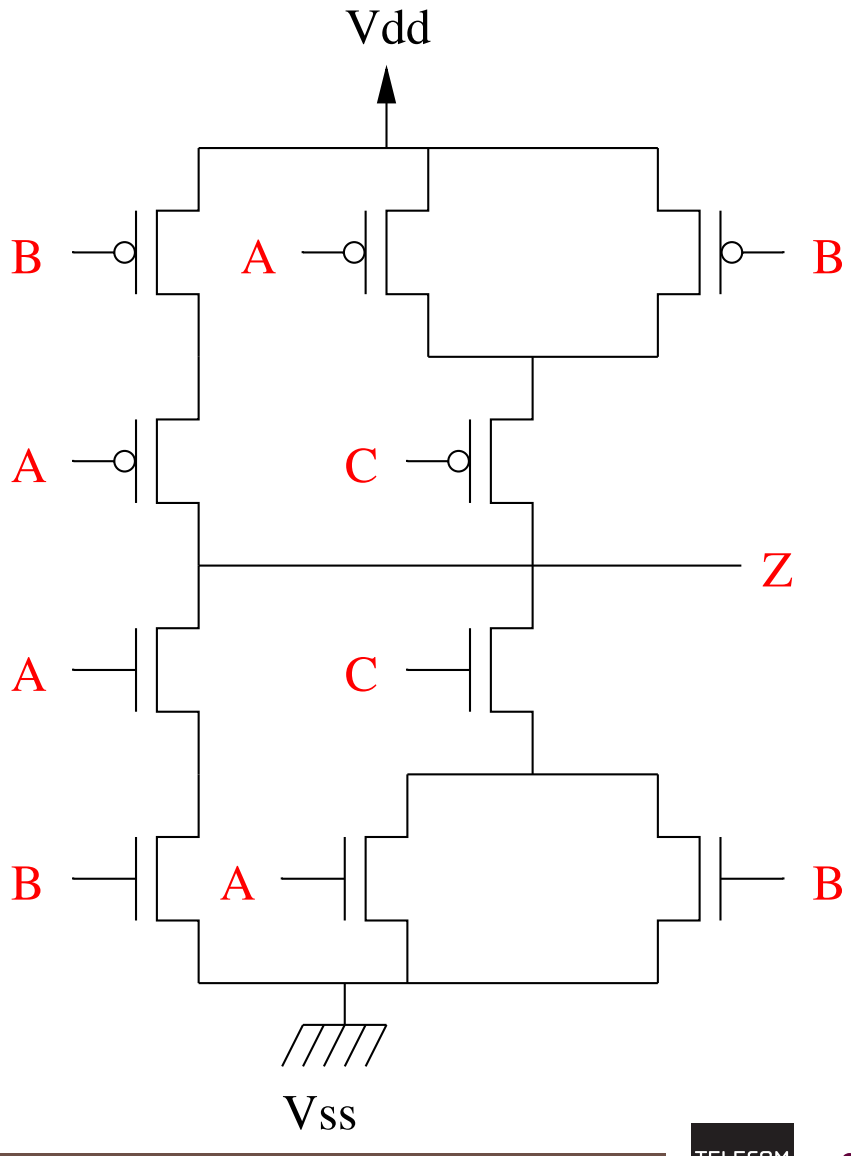
- Q1 : Build the gate using optimum optimisation methods
- Q2 : Exchange "C" NMOS transistor with "A/B" NMOS network : Does it make any difference ?
- Q3 : Build the gate using dual PMOS and NMOS networks
- Q4 : What are the potentials flaws of this later gate ?



CMOS logic

Optimum network : $F = \overline{A \cdot B + B \cdot C + A \cdot C}$

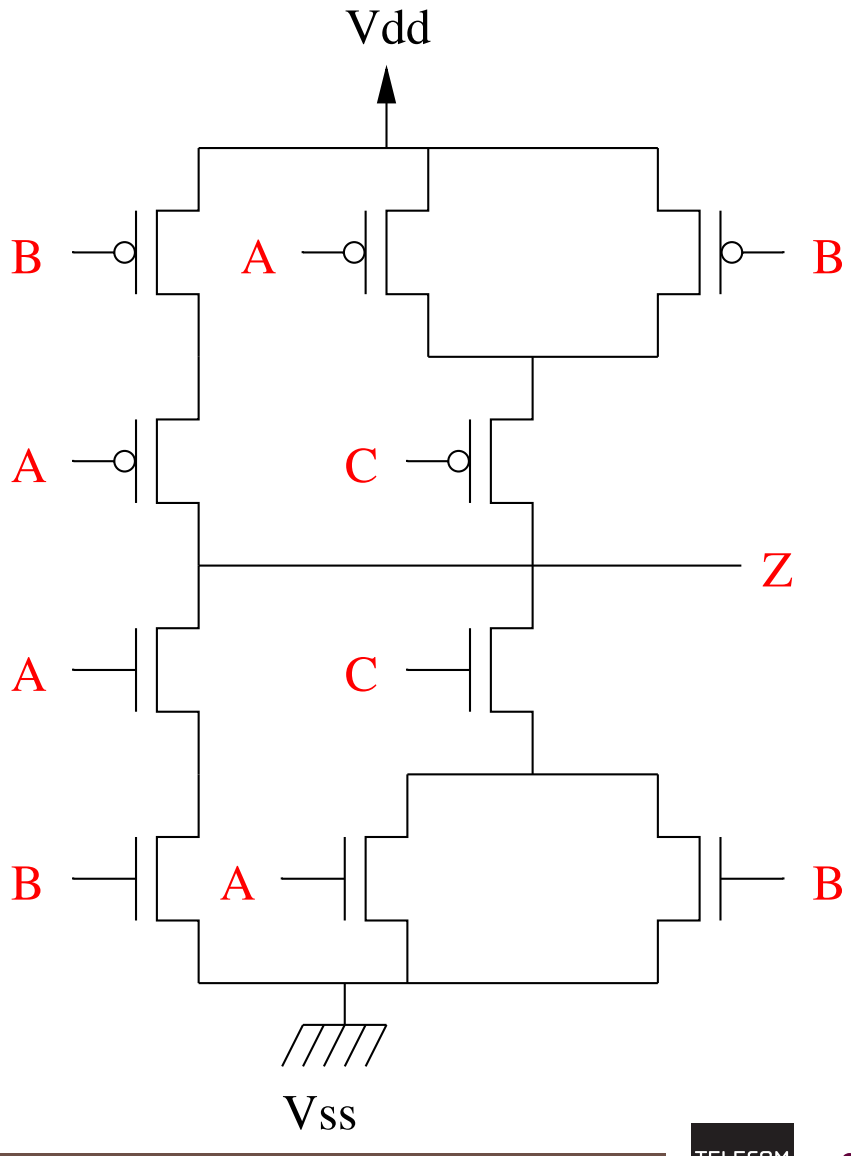
- Q1 : Build the gate using optimum optimisation methods
- Q2 : Exchange "C" NMOS transistor with "A/B" NMOS network : Does it make any difference ?
- Q3 : Build the gate using dual PMOS and NMOS networks
- Q4 : What are the potentials flaws of this later gate ?



CMOS logic

Optimum network : $F = \overline{A \cdot B + B \cdot C + A \cdot C}$

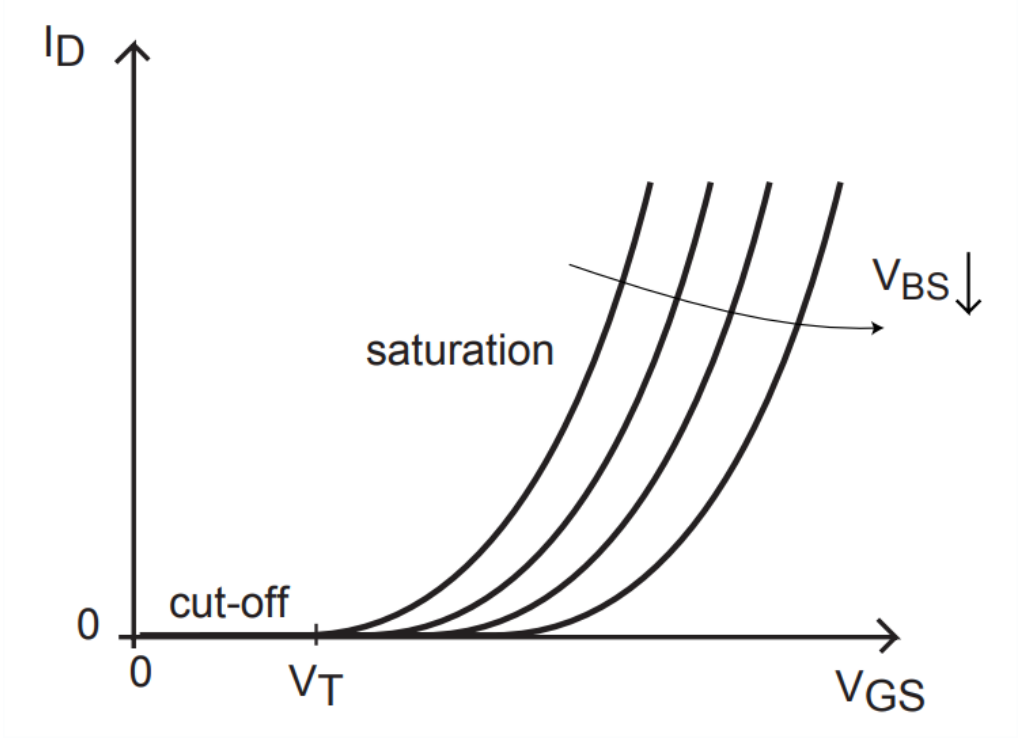
- Q1 : Build the gate using optimum optimisation methods
- Q2 : Exchange "C" NMOS transistor with "A/B" NMOS network : Does it make any difference ?
- Q3 : Build the gate using dual PMOS and NMOS networks
- Q4 : What are the potentials flaws of this later gate ?



CMOS logic

Low/High input counts

- Series of transistors : the body effect (see lecture L1) leads to slow high V_t transistors.

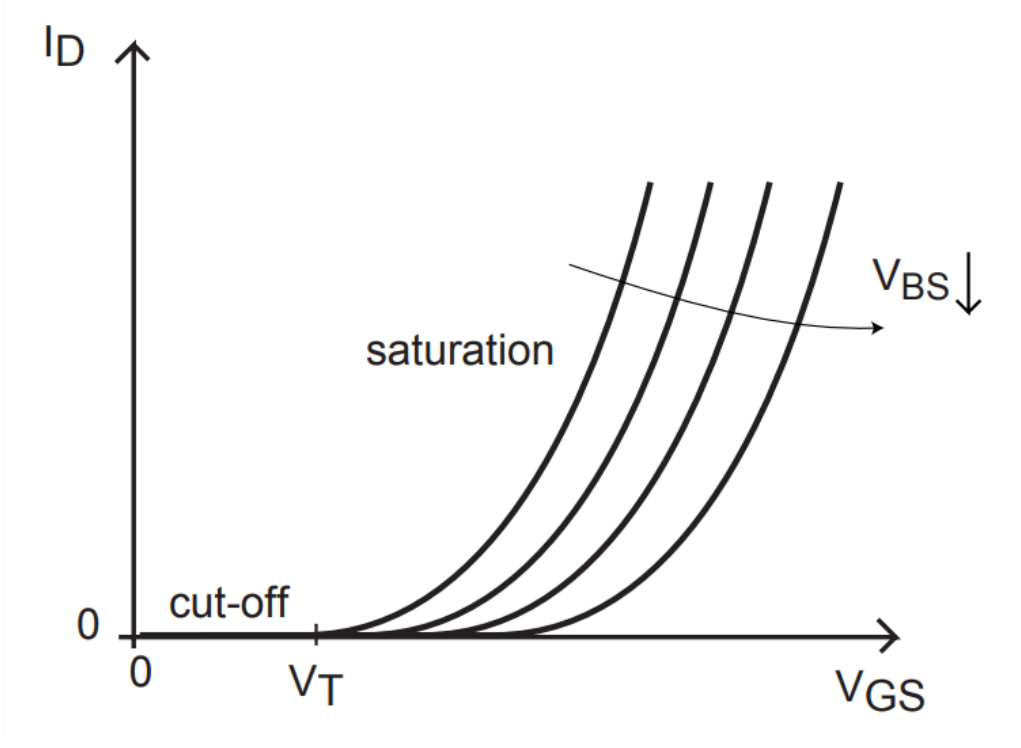


- Assembly of low input count gates may be faster than high input counts gates.

CMOS logic

Low/High input counts

- Series of transistors : the body effect (see lecture L1) leads to slow high V_t transistors.



- Assembly of low input count gates may be faster than high input counts gates.

CMOS logic

Low/High input counts : The NAND6 example

- AND6 gate example, 28nm process, 1v supply voltage, all NMOS transistors of equal size, $w_p/w_n = 1.6$, no output load.
- Test cases : (1) All inputs change, (2) One input changes.

- NAND6+INV
- Tp1R = 53ps
- Tp1F = 18ps
- Tp2R = 33ps
- Tp2F = 11ps

- NAND3+NOR2
- Tp1R = 29ps
- Tp1F = 5ps
- Tp2R = 19ps
- Tp2F = 9ps

- NAND2+NOR3
- Tp1R = 27ps
- Tp1F = 4ps
- Tp2R = 18ps
- Tp2F = 9ps

- Q1 : Setup a simple model of the propagation time, using resistive approximation

CMOS logic

Low/High input counts : The NAND6 example

- AND6 gate example, 28nm process, 1v supply voltage, all NMOS transistors of equal size, $w_p/w_n = 1.6$, no output load.
- Test cases : (1) All inputs change, (2) One input changes.

- NAND6+INV
- Tp1R = 53ps
- Tp1F = 18ps
- Tp2R = 33ps
- Tp2F = 11ps

- NAND3+NOR2
- Tp1R = 29ps
- Tp1F = 5ps
- Tp2R = 19ps
- Tp2F = 9ps

- NAND2+NOR3
- Tp1R = 27ps
- Tp1F = 4ps
- Tp2R = 18ps
- Tp2F = 9ps

- Q1 : Setup a simple model of the propagation time, using resistive approximation

CMOS logic

Low/High input counts : The NAND6 example

- AND6 gate example, 28nm process, 1v supply voltage, all NMOS transistors of equal size, $w_p/w_n = 1.6$, no output load.
- Test cases : (1) All inputs change, (2) One input changes.

NAND6+INV	NAND3+NOR2	NAND2+NOR3
Tp1R = 53ps	Tp1R = 29ps	Tp1R = 27ps
Tp1F = 18ps	Tp1F = 5ps	Tp1F = 4ps
Tp2R = 33ps	Tp2R = 19ps	Tp2R = 18ps
Tp2F = 11ps	Tp2F = 9ps	Tp2F = 9ps

- Q1 : Setup a simple model of the propagation time, using resistive approximation

CMOS logic

Low/High input counts : The NAND6 example

- AND6 gate example, 28nm process, 1v supply voltage, all NMOS transistors of equal size, $w_p/w_n = 1.6$, no output load.
- Test cases : (1) All inputs change, (2) One input changes.

- NAND6+INV
- Tp1R = 53ps
- Tp1F = 18ps
- Tp2R = 33ps
- Tp2F = 11ps

- NAND3+NOR2
- Tp1R = 29ps
- Tp1F = 5ps
- Tp2R = 19ps
- Tp2F = 9ps

- NAND2+NOR3
- Tp1R = 27ps
- Tp1F = 4ps
- Tp2R = 18ps
- Tp2F = 9ps

- Q1 : Setup a simple model of the propagation time, using resistive approximation

CMOS logic

Low/High input counts : The NAND6 example

- AND6 gate example, 28nm process, 1v supply voltage, all NMOS transistors of equal size, $w_p/w_n = 1.6$, no output load.
- Test cases : (1) All inputs change, (2) One input changes.

- NAND6+INV
- Tp1R = 53ps
- Tp1F = 18ps
- Tp2R = 33ps
- Tp2F = 11ps

- NAND3+NOR2
- Tp1R = 29ps
- Tp1F = 5ps
- Tp2R = 19ps
- Tp2F = 9ps

- NAND2+NOR3
- Tp1R = 27ps
- Tp1F = 4ps
- Tp2R = 18ps
- Tp2F = 9ps

- Q1 : Setup a simple model of the propagation time, using resistive approximation

CMOS logic

non complementary PMOS/NMOS networks

- NMOS and PMOS networks may be simultaneously OFF.
- Usage : Input/Output tranceivers for external busses.
- Tristate invertor.

EN	A	Z
0	0	highZ
0	1	highZ
1	0	1
1	1	0

CMOS logic

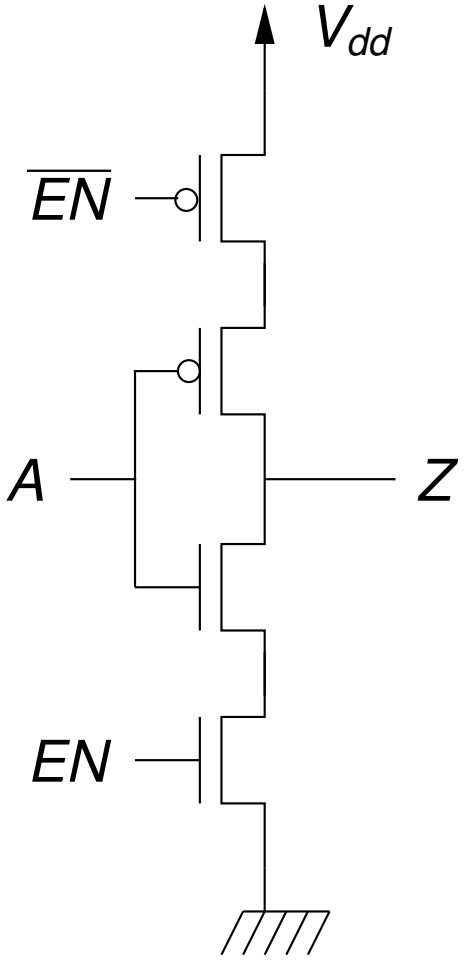
non complementary PMOS/NMOS networks

- NMOS and PMOS networks may be simultaneously OFF.
- Usage : Input/Output tranceivers for external busses.
- Tristate invertor.

EN	A	Z
0	0	highZ
0	1	highZ
1	0	1
1	1	0

CMOS logic

non complementary PMOS/NMOS networks



- NMOS and PMOS networks may be simultaneously OFF.
- Usage : Input/Output tranceivers for external busses.
- Tristate invertor.

EN	A	Z
0	0	highZ
0	1	highZ
1	0	1
1	1	0

CMOS logic

non complementary PMOS/NMOS networks

- NMOS and PMOS networks may be simultaneously ON.
- Taking into account input cases, short-circuits are avoided.
- Only for full custom design (not included in automated methods)

A	B	Z
0	0	1
0	1	forbidden case
1	0	High Z
1	1	0

CMOS logic

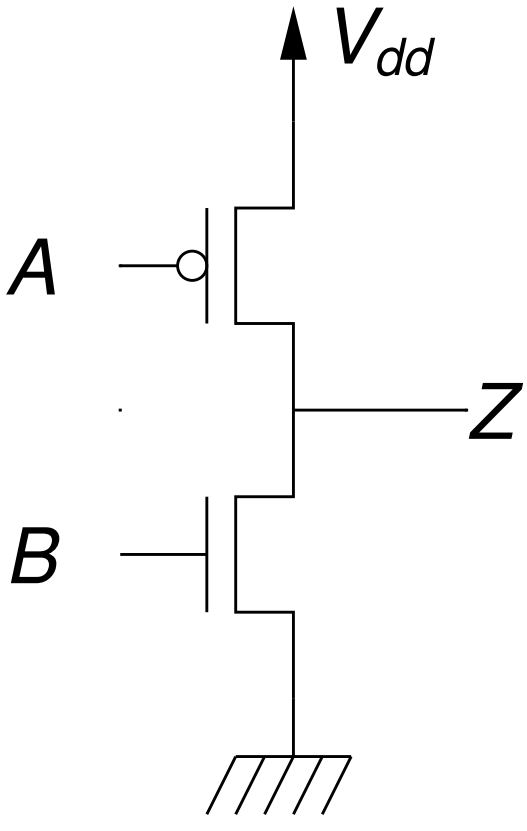
non complementary PMOS/NMOS networks

- NMOS and PMOS networks may be simultaneously ON.
- Taking into account input cases, short-circuits are avoided.
- Only for full custom design (not included in automated methods)

A	B	Z
0	0	1
0	1	forbidden case
1	0	High Z
1	1	0

CMOS logic

non complementary PMOS/NMOS networks



- NMOS and PMOS networks may be simultaneously ON.
- Taking into account input cases, short-circuits are avoided.
- Only for full custom design (not included in automated methods)

A	B	Z
0	0	1
0	1	forbidden case
1	0	High Z
1	1	0

CMOS logic

Summary

- Logic style mainly used in standard cell libraries (see lecture L3)
  - Robustness, Noise immunity
  - The sizing of transistors affects only performance, not functionality.
  - In practical cases series of transistors are limited to 3 to 4 transistors :

CMOS logic

Summary

- Logic style mainly used in standard cell libraries (see lecture L3)
  - Robustness, Noise immunity
  - The sizing of transistors affects only performance, not functionality.
  - In practical cases series of transistors are limited to 3 to 4 transistors :

CMOS logic

Summary

- Logic style mainly used in standard cell libraries (see lecture L3)
  - Robustness, Noise immunity
  - The sizing of transistors affects only performance, not functionality.
  - In practical cases series of transistors are limited to 3 to 4 transistors :

CMOS logic

Summary

- Logic style mainly used in standard cell libraries (see lecture L3)
  - Robustness, Noise immunity
  - The sizing of transistors affects only performance, not functionality.
  - In practical cases series of transistors are limited to 3 to 4 transistors :

Why a so complicated structure ?

- Only one boolean function but two transistor networks . . .
- Truth table of the boolean function :
  - The PMOS network generates the "1" values of the truth table
  - The NMOS network generates the "0" values of the truth table
- Only one network should be enough :
  - The "1" values of the truth table could be implicit values.
  - CMOS logic is not optimal from the standpoint of the complexity.

Why a so complicated structure ?

- Only one boolean function but two transistor networks . . .
- Truth table of the boolean function :
  - The PMOS network generates the "1" values of the truth table
  - The NMOS network generates the "0" values of the truth table
- Only one network should be enough :
  - The "1" values of the truth table could be implicit values.
  - CMOS logic is not optimal from the standpoint of the complexity.

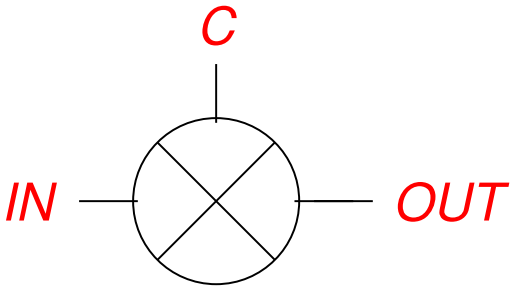
Why a so complicated structure ?

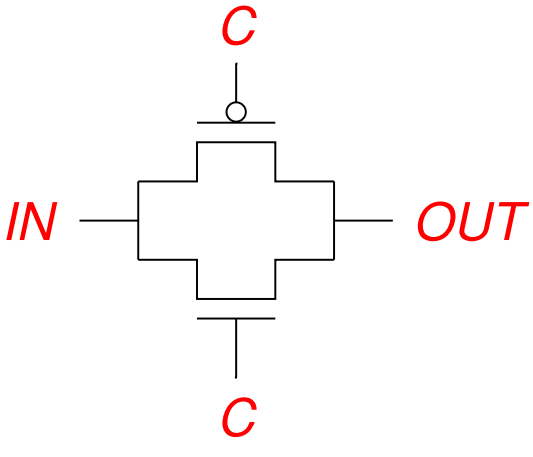
- Only one boolean function but two transistor networks . . .
- Truth table of the boolean function :
  - The PMOS network generates the "1" values of the truth table
  - The NMOS network generates the "0" values of the truth table
- Only one network should be enough :
  - The "1" values of the truth table could be implicit values.
  - CMOS logic is not optimal from the standpoint of the complexity.

Outline

Pass Transistor Logic

An ideal switch

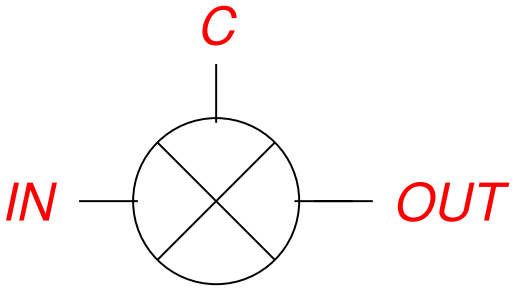


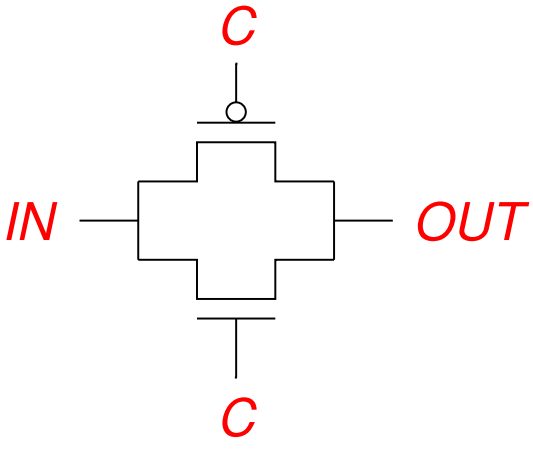


- NMOS transistor efficient for falling transitions.
- PMOS transistor efficient for rising transitions.
- Warning : Passive circuit, no signal regeneration along the path "IN" to "OUT"

Pass Transistor Logic

An ideal switch

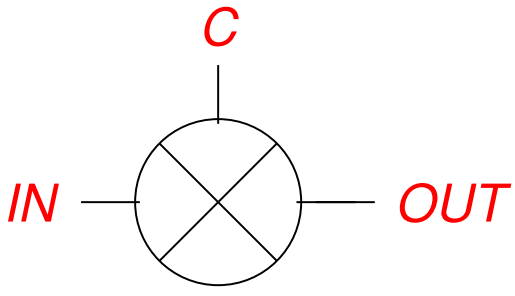


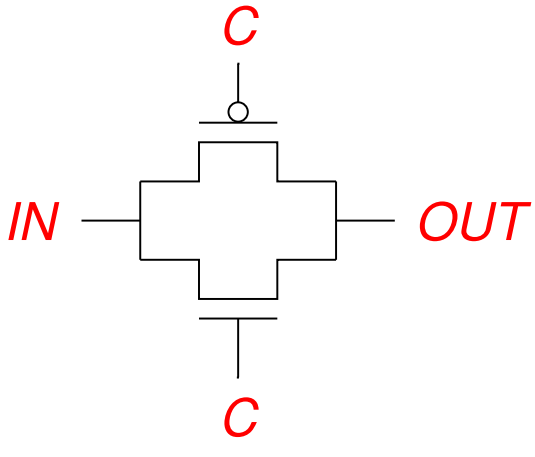


- NMOS transistor efficient for falling transitions.
- PMOS transistor efficient for rising transitions.
- Warning : Passive circuit, no signal regeneration along the path "IN" to "OUT"

Pass Transistor Logic

An ideal switch

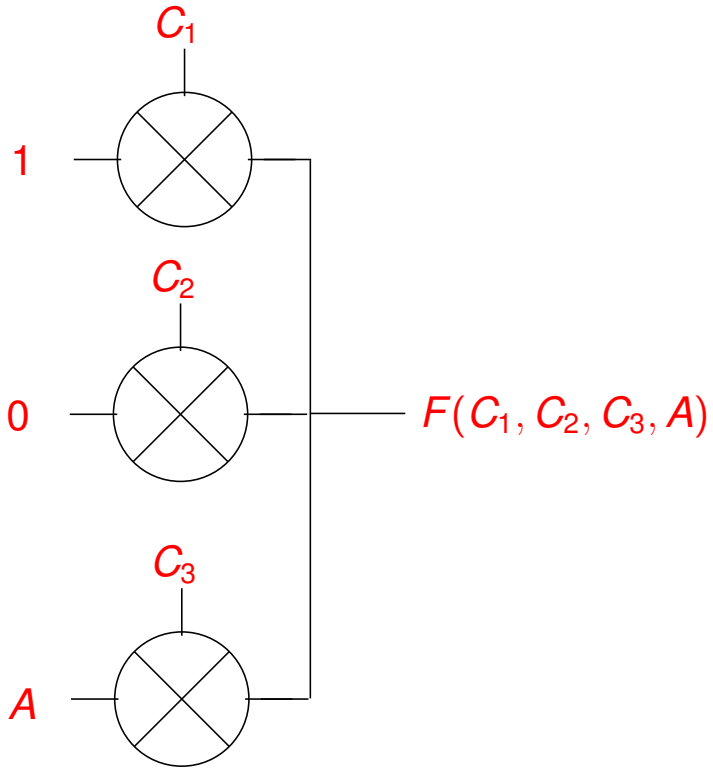




- NMOS transistor efficient for falling transitions.
- PMOS transistor efficient for rising transitions.
- Warning : Passive circuit, no signal regeneration along the path "IN" to "OUT"

Pass transistor logic

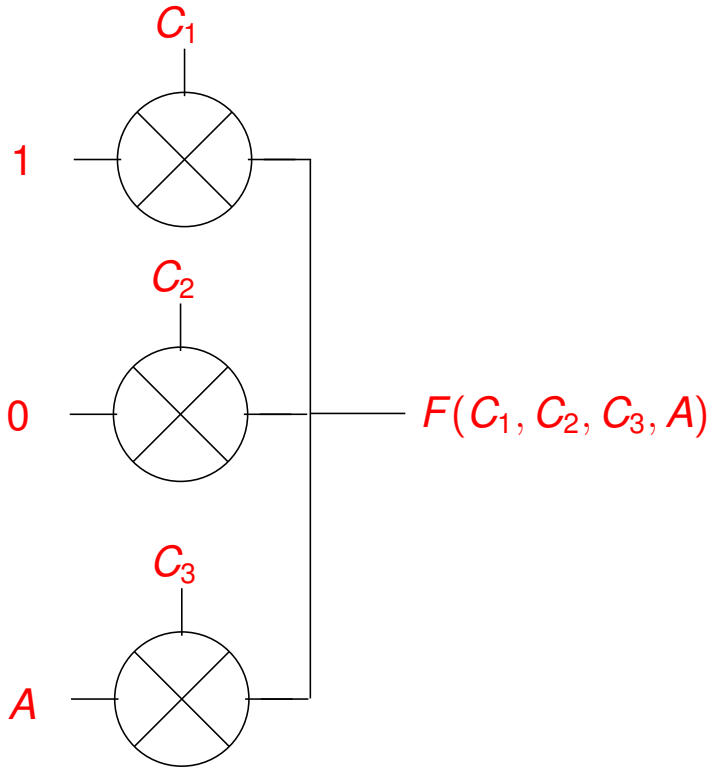
Input signals may be constants or variables



- Input is a constant : CMOS logic
- Input is a variable : Pass logic
- Combination of conditions and inputs should not lead to short-circuits.
- Combination of conditions and inputs should not lead to open-circuits.
- Any form of the Boolean expression.

Pass transistor logic

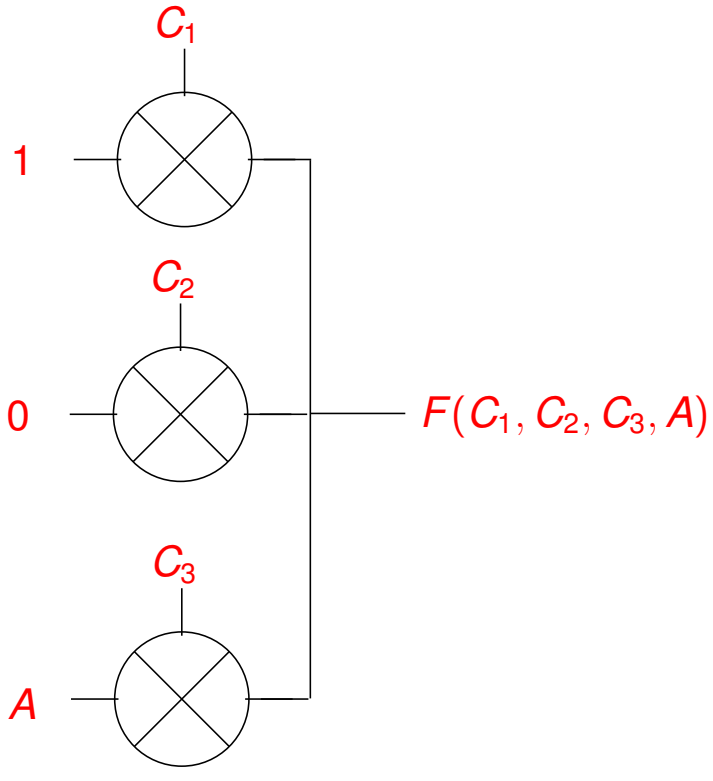
Input signals may be constants or variables



- Input is a constant : CMOS logic
- Input is a variable : Pass logic
- Combination of conditions and inputs should not lead to short-circuits.
- Combination of conditions and inputs should not lead to open-circuits.
- Any form of the Boolean expression.

Pass transistor logic

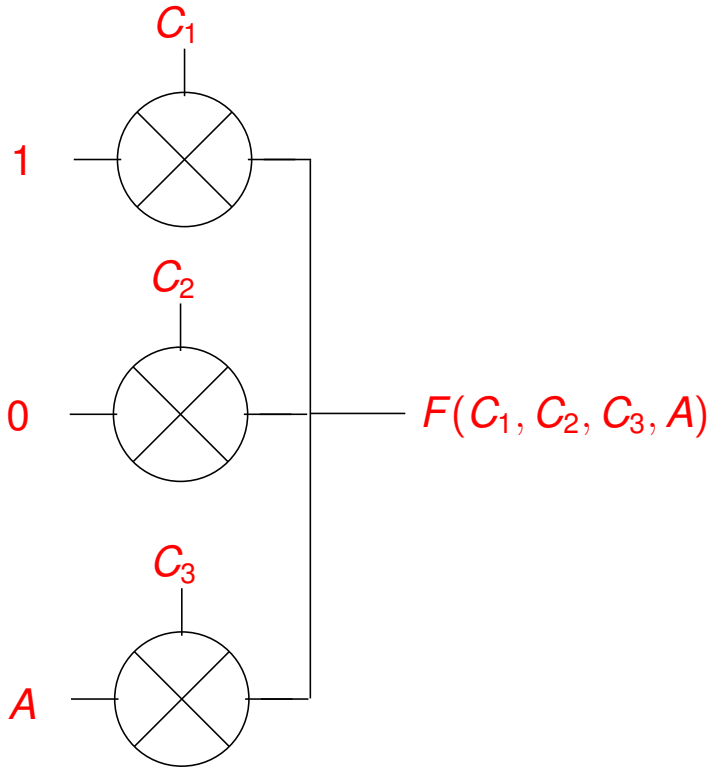
Input signals may be constants or variables



- Input is a constant : CMOS logic
- Input is a variable : Pass logic
- Combination of conditions and inputs should not lead to short-circuits.
- Combination of conditions and inputs should not lead to open-circuits.
- Any form of the Boolean expression.

Pass transistor logic

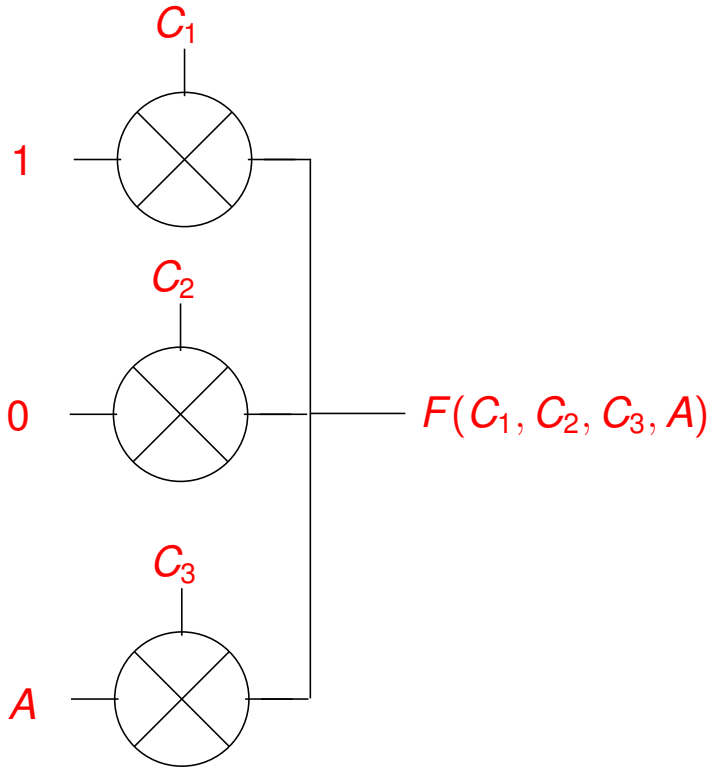
Input signals may be constants or variables



- Input is a constant : CMOS logic
- Input is a variable : Pass logic
- Combination of conditions and inputs should not lead to short-circuits.
- Combination of conditions and inputs should not lead to open-circuits.
- Any form of the Boolean expression.

Pass transistor logic

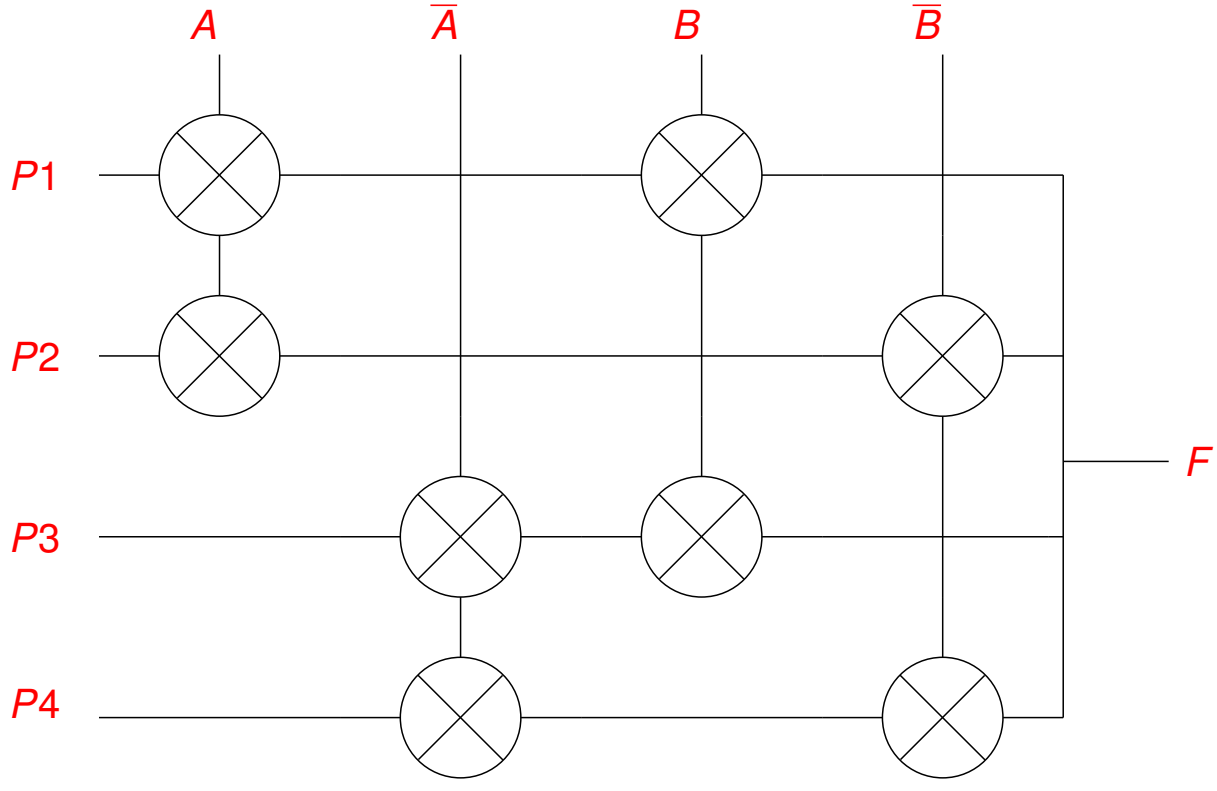
Input signals may be constants or variables



- Input is a constant : CMOS logic
- Input is a variable : Pass logic
- Combination of conditions and inputs should not lead to short-circuits.
- Combination of conditions and inputs should not lead to open-circuits.
- Any form of the Boolean expression.

Pass Transistor Logic

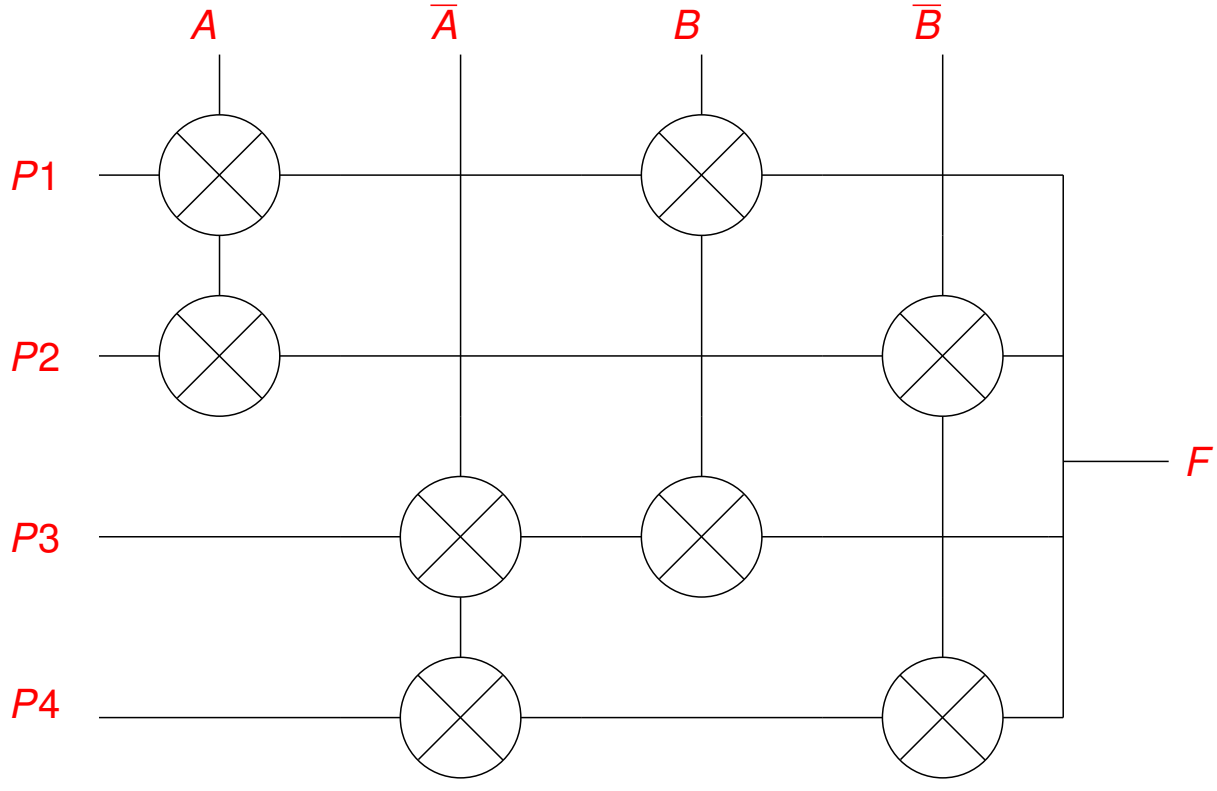
Full decoding



- $F = P_1 \cdot A \cdot B + P_2 \cdot A \cdot \overline{B} + P_3 \cdot \overline{A} \cdot B + P_4 \cdot \overline{A} \cdot \overline{B}$
- Division of the truth table into four sub-tables (P1,P2,P3,P4) based on the full decoding of the 2 inputs A and B.

Pass Transistor Logic

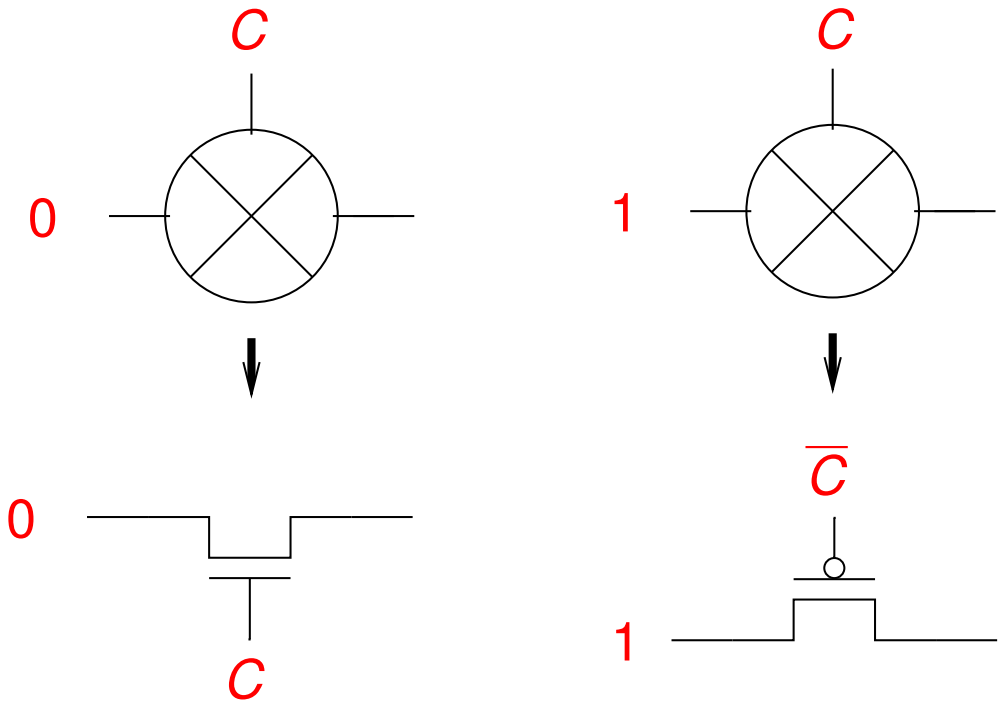
Full decoding



- $F = P_1 \cdot A \cdot B + P_2 \cdot A \cdot \overline{B} + P_3 \cdot \overline{A} \cdot B + P_4 \cdot \overline{A} \cdot \overline{B}$
- Division of the truth table into four sub-tables (P1,P2,P3,P4) based on the full decoding of the 2 inputs A and B.

Pass Transistor Logic

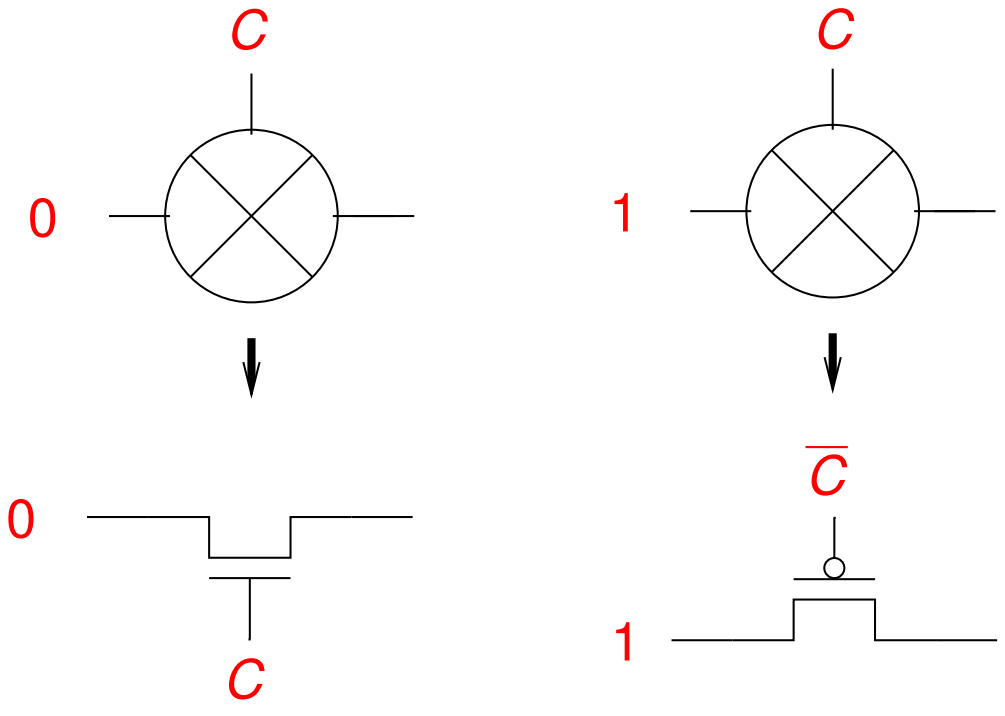
Area minimization for constant inputs



- Constant zero is a connection to V_{ss}
- Constant one is a connection to V_{dd}
- Beware the inverted condition for the PMOS transistor

Pass Transistor Logic

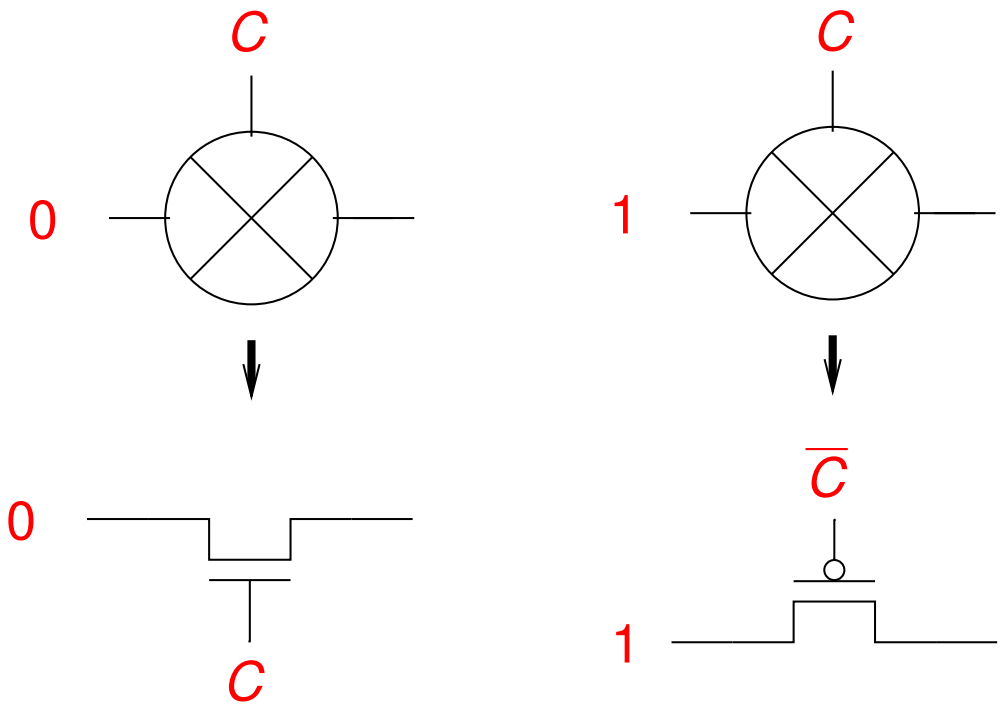
Area minimization for constant inputs



- Constant zero is a connection to V_{ss}
- Constant one is a connection to V_{dd}
- Beware the inverted condition for the PMOS transistor

Pass Transistor Logic

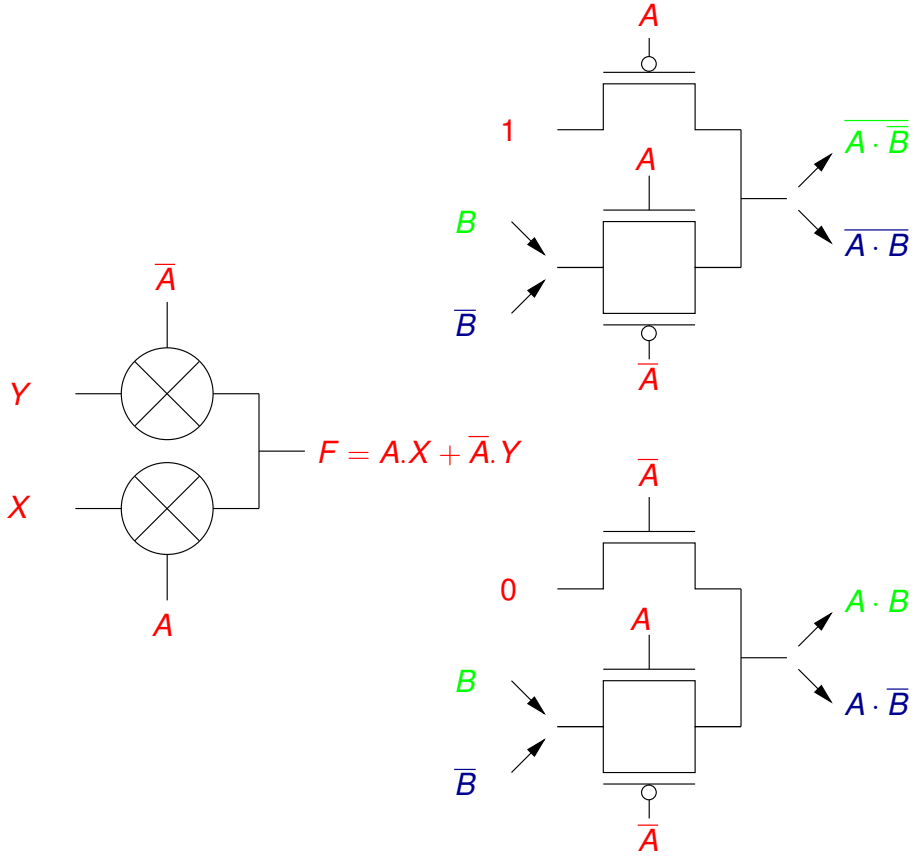
Area minimization for constant inputs



- Constant zero is a connection to V_{ss}
- Constant one is a connection to V_{dd}
- Beware the inverted condition for the PMOS transistor

Pass Transistor Logic

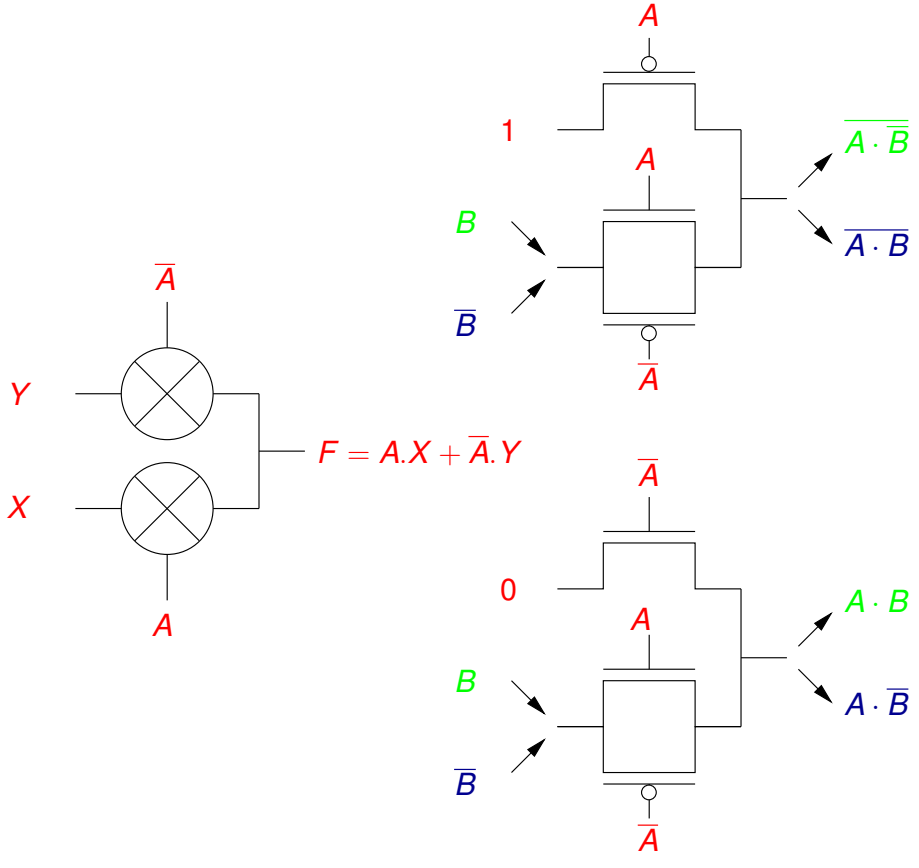
Application to two input boolean functions



- F function is a simple mux between X and Y.
- Q1 : Generate the $F = A + B$ function.
- Q2 : Compare the five functions with CMOS Logic.

Pass Transistor Logic

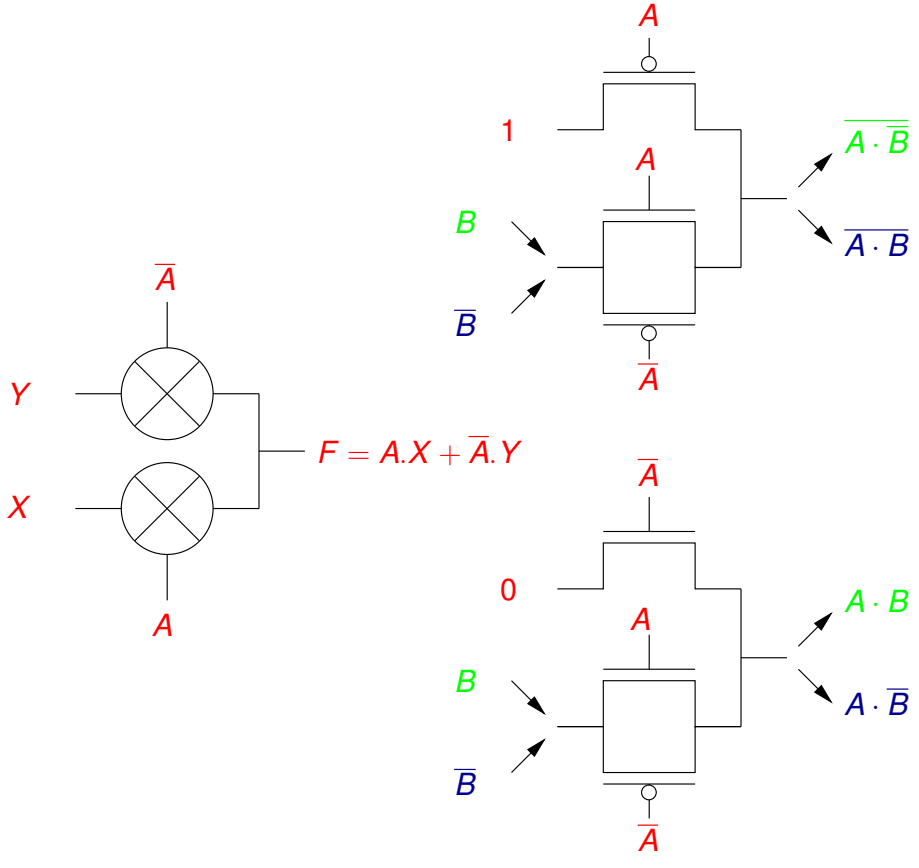
Application to two input boolean functions



- F function is a simple mux between X and Y.
- Q1 : Generate the $F = A + B$ function.
- Q2 : Compare the five functions with CMOS Logic.

Pass Transistor Logic

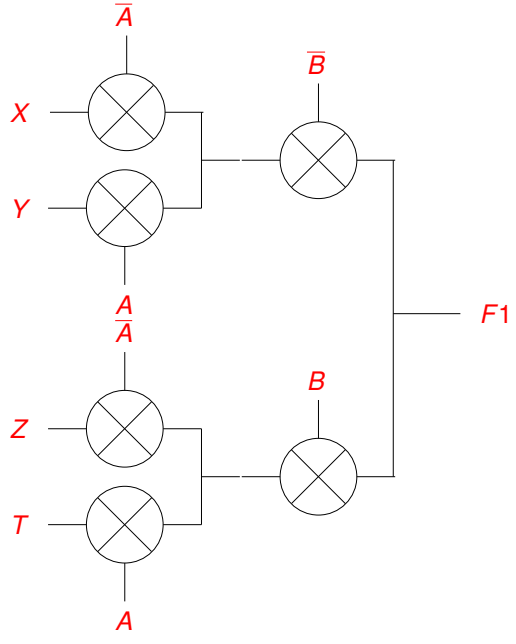
Application to two input boolean functions

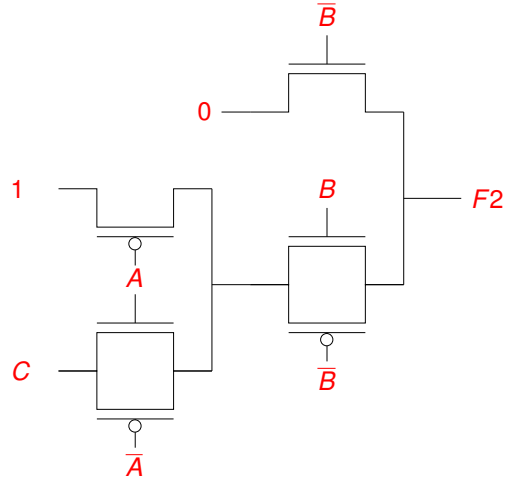


- F function is a simple mux between X and Y.
- Q1 : Generate the $F = A + B$ function.
- Q2 : Compare the five functions with CMOS Logic.

Pass Transistor Logic

Trees of muxes

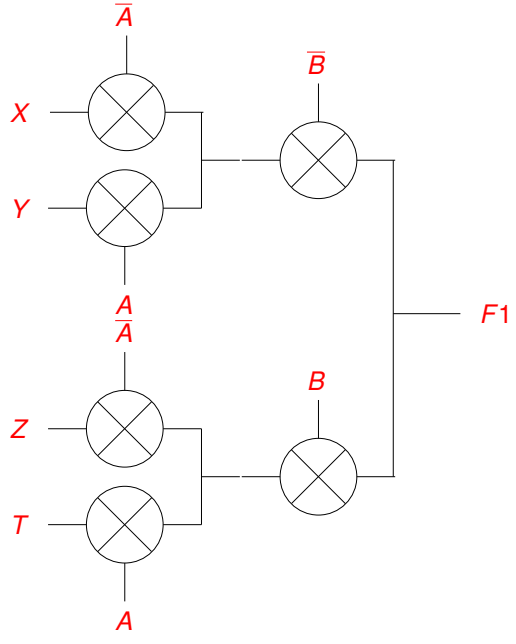


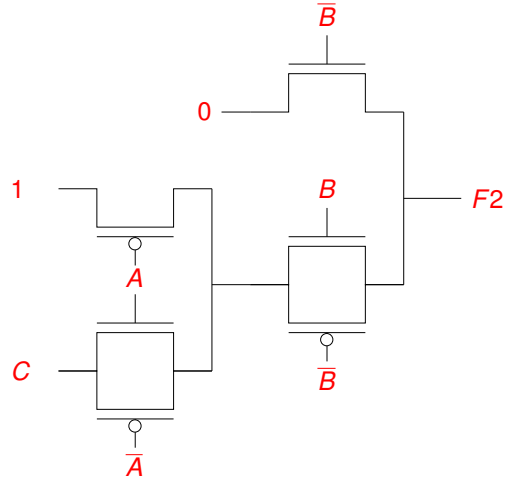


- $F1 = \overline{B} \cdot (\overline{A} \cdot X + A \cdot Y) + B \cdot (\overline{A} \cdot Z + A \cdot T)$
- $F2 = B \cdot A \cdot C + B \cdot \overline{A}$
- Less transistors than Pass Transistor Logic using decoders.
- But less simplifications (branch sharing)
- Heavily used for look-up tables (LUTs) in FPGAs.

Pass Transistor Logic

Trees of muxes

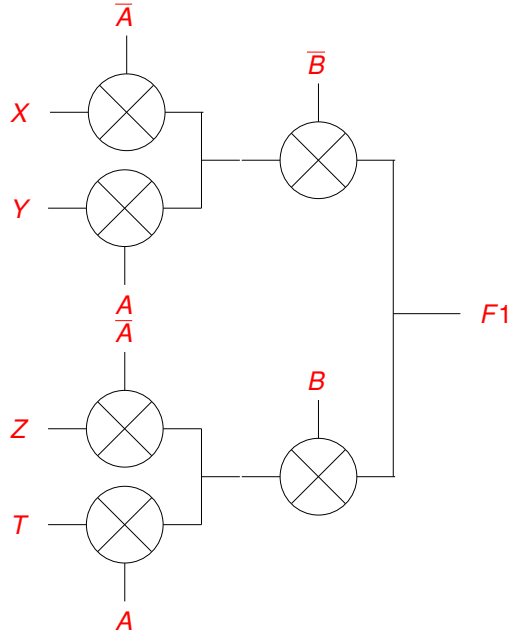


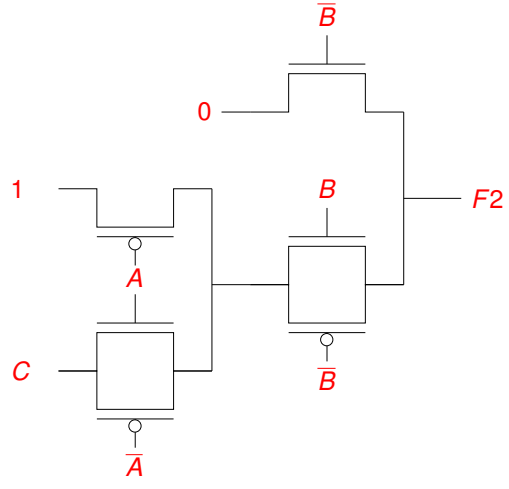


- $F1 = \overline{B} \cdot (\overline{A} \cdot X + A \cdot Y) + B \cdot (\overline{A} \cdot Z + A \cdot T)$
- $F2 = B \cdot A \cdot C + B \cdot \overline{A}$
- Less transistors than Pass Transistor Logic using decoders.
- But less simplifications (branch sharing)
- Heavily used for look-up tables (LUTs) in FPGAs.

Pass Transistor Logic

Trees of muxes

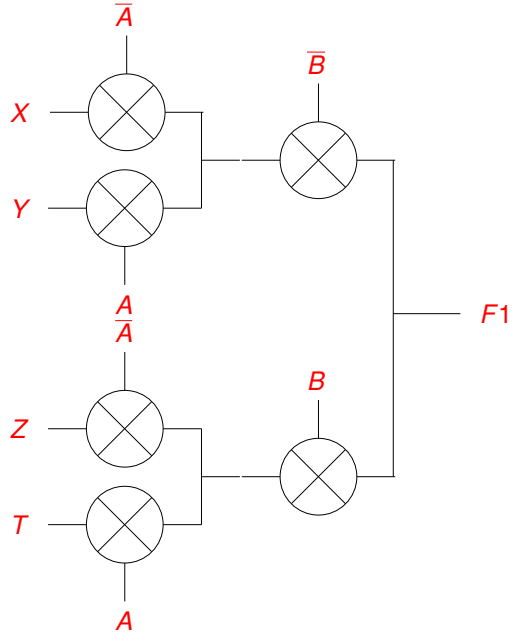


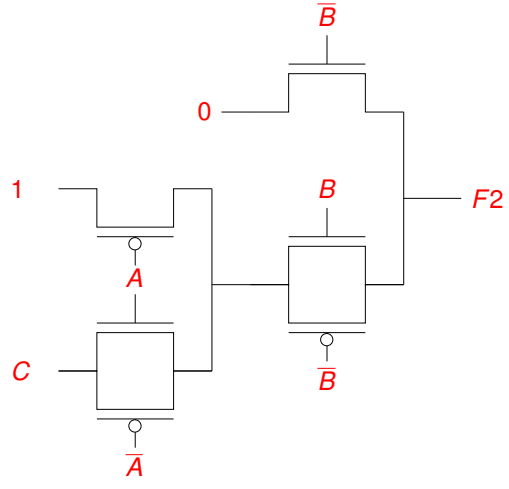


- $F1 = \overline{B} \cdot (\overline{A} \cdot X + A \cdot Y) + B \cdot (\overline{A} \cdot Z + A \cdot T)$
- $F2 = B \cdot A \cdot C + B \cdot \overline{A}$
- Less transistors than Pass Transistor Logic using decoders.
- But less simplifications (branch sharing)
- Heavily used for look-up tables (LUTs) in FPGAs.

Pass Transistor Logic

Trees of muxes

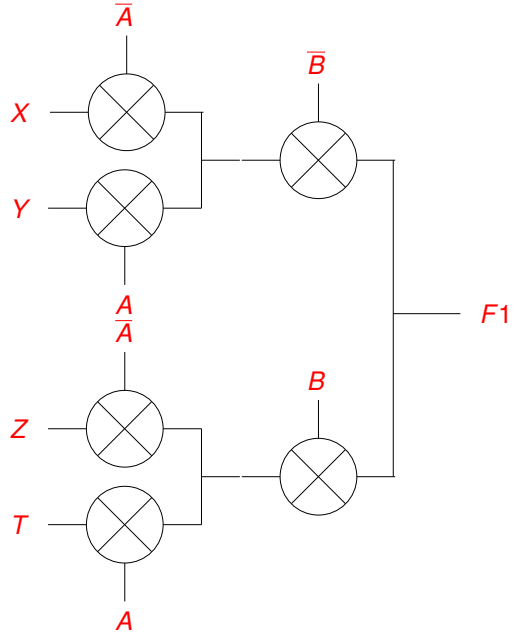


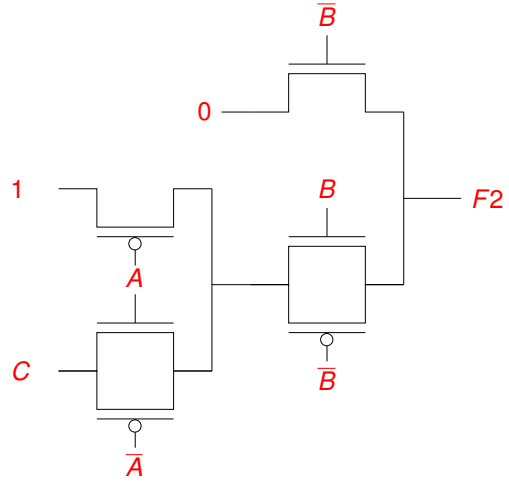


- $F1 = \overline{B} \cdot (\overline{A} \cdot X + A \cdot Y) + B \cdot (\overline{A} \cdot Z + A \cdot T)$
- $F2 = B \cdot A \cdot C + B \cdot \overline{A}$
- Less transistors than Pass Transistor Logic using decoders.
- But less simplifications (branch sharing)
- Heavily used for look-up tables (LUTs) in FPGAs.

Pass Transistor Logic

Trees of muxes

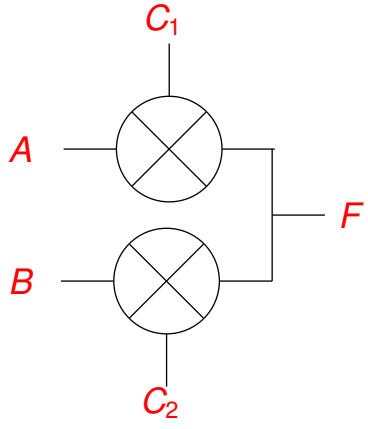


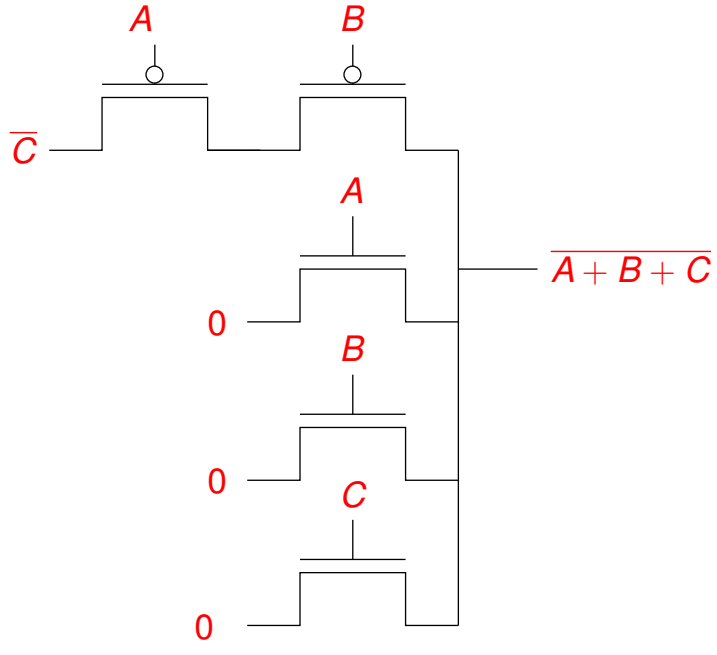


- $F1 = \overline{B} \cdot (\overline{A} \cdot X + A \cdot Y) + B \cdot (\overline{A} \cdot Z + A \cdot T)$
- $F2 = B \cdot A \cdot C + B \cdot \overline{A}$
- Less transistors than Pass Transistor Logic using decoders.
- But less simplifications (branch sharing)
- Heavily used for look-up tables (LUTs) in FPGAs.

Pass Transistor Logic

Non exclusive conditions

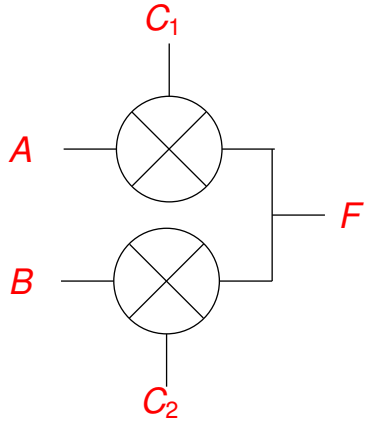


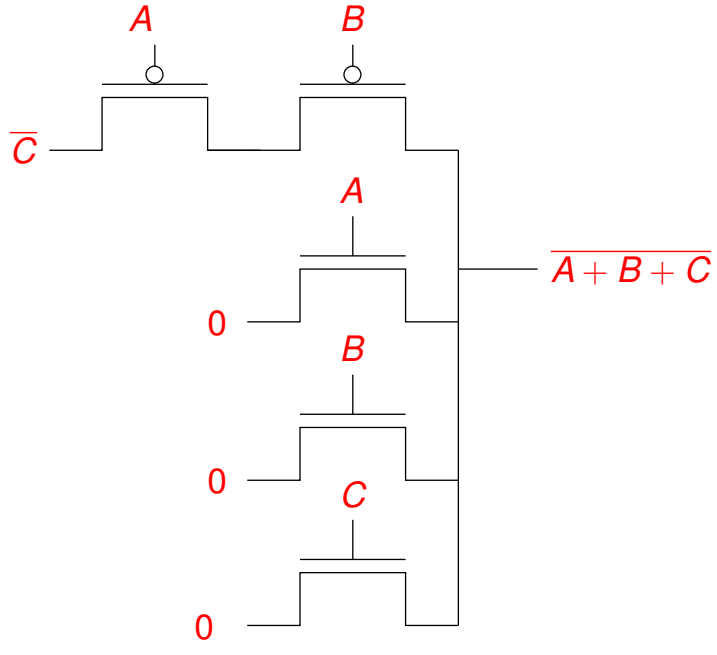


- Several branches may be simultaneously ON.
- Short circuits and High Impedance state should be avoided.
- $F : (C_1 = C_2) \Rightarrow A = B$

Pass Transistor Logic

Non exclusive conditions

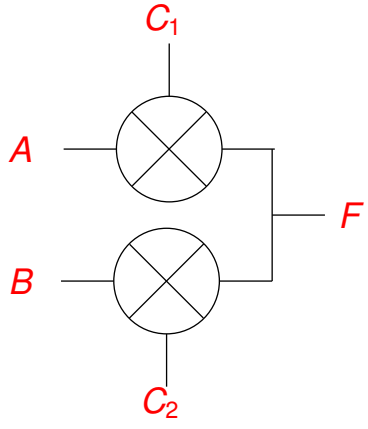


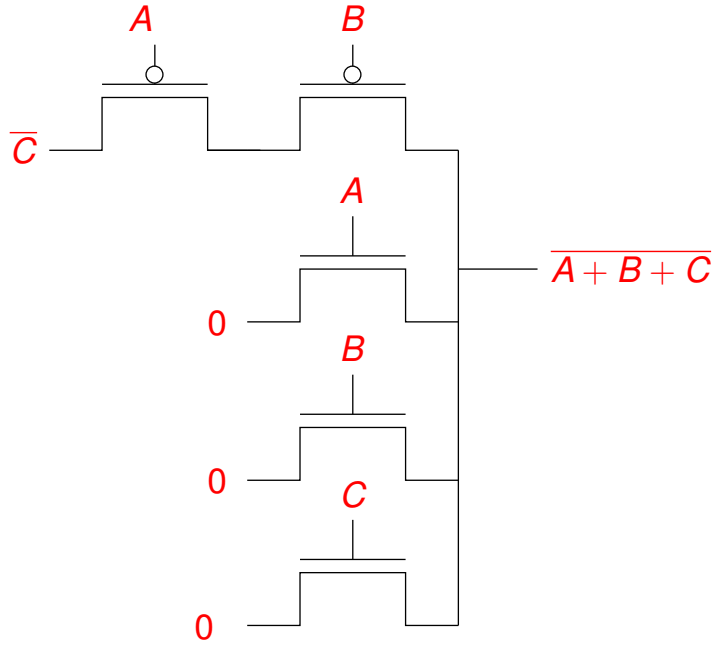


- Several branches may be simultaneously ON.
- Short circuits and High Impedance state should be avoided.
- $F : (C_1 = C_2) \Rightarrow A = B$

Pass Transistor Logic

Non exclusive conditions

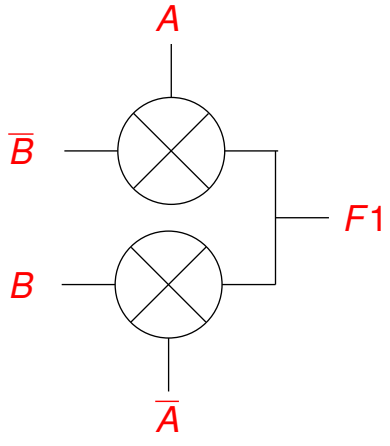


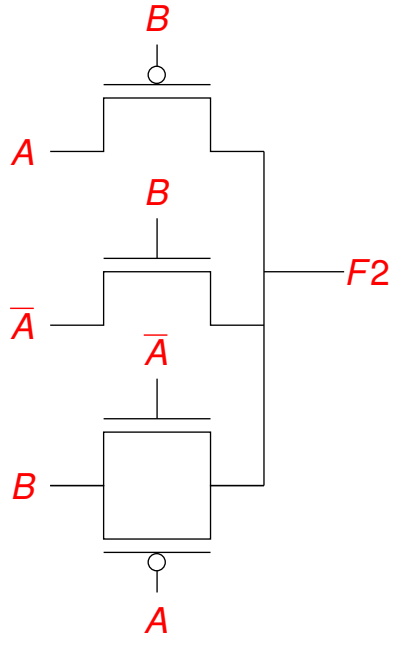


- Several branches may be simultaneously ON.
- Short circuits and High Impedance state should be avoided.
- $F : (C_1 = C_2) \Rightarrow A = B$

Pass Transistor Logic

Two inputs XOR gate

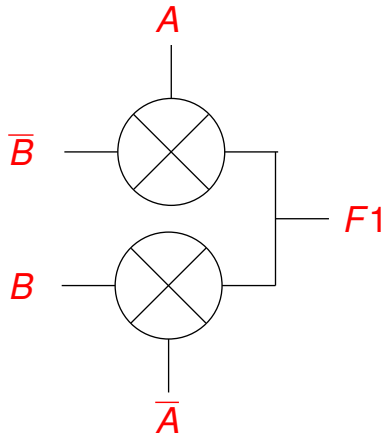


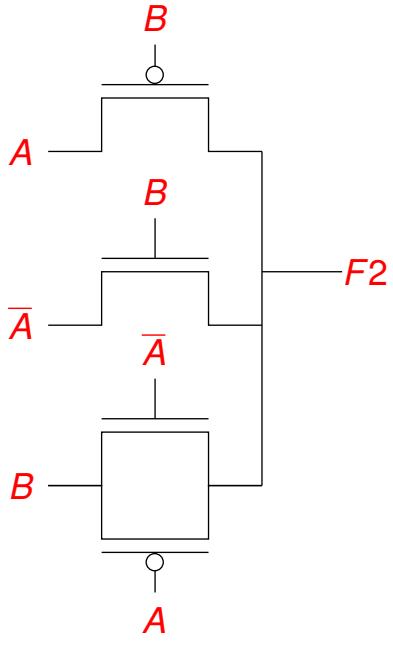


- XOR : $A \cdot \overline{B} + \overline{A} \cdot B$
- F1 : Classical pass transistor logic XOR gate (8 transistors including invertors)
- F2 : Pass transistor logic XOR gate using non exclusive branches (6 transistors including invertors)

Pass Transistor Logic

Two inputs XOR gate

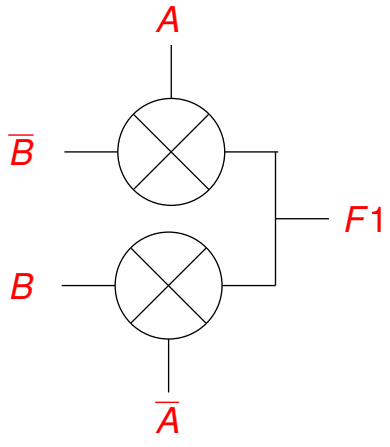


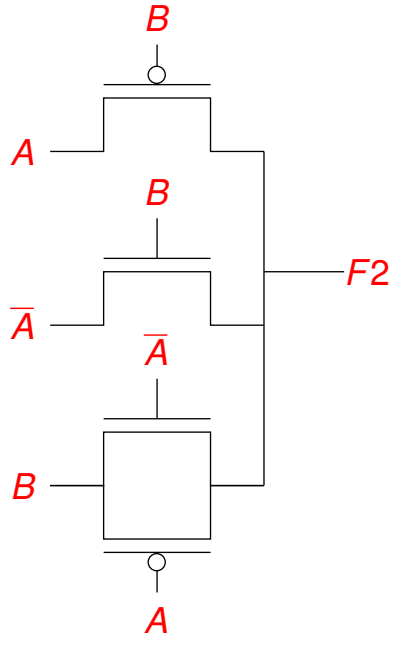


- XOR : $A \cdot \overline{B} + \overline{A} \cdot B$
- F1 : Classical pass transistor logic XOR gate (8 transistors including invertors)
- F2 : Pass transistor logic XOR gate using non exclusive branches (6 transistors including invertors)

Pass Transistor Logic

Two inputs XOR gate





- XOR : $A \cdot \overline{B} + \overline{A} \cdot B$
- F1 : Classical pass transistor logic XOR gate (8 transistors including invertors)
- F2 : Pass transistor logic XOR gate using non exclusive branches (6 transistors including invertors)

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS $28nm$, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS $28nm$, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS 28nm, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS $28nm$, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS $28nm$, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Chains of pass transistor logic / versus chain of invertors

- Passive propagation.
- RC time accumulation
- Test condition : CMOS $28nm$, $V_{DD} = 1V$.
- Test case 1 : 2 invertors separated by N pass gates.
- Test case 2 : 2 invertors separated by N invertors.

N	1	2	3	4	5	6	7
Tp pass chain (ps)	7	11	17	24	32	42	53
Tp inv chain (ps)	7	12	18	23	29	35	41

- Increasing delta propagation time (cumulative RC)
- Use invertors every 2 or 3 pass gate.

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Examples

- Q1 : Design of a 2 inputs XOR gate
  - Q1.1 : How many transistors in CMOS logic.
  - Q1.2 : Compare with pass transistor logic.
- Q2 : Ripple carry adder.
  - Q2.1 : Design the boolean function computing the output carry of a "Full adder" using CMOS logic.
  - Q2.2 : Optimize the carry propagation of a 2N-bits ripple carry adder.
  - In a full-adder, the output carry may be calculated as follows :
    - $C_{OUT} = G + P \cdot C_{IN}$
    - With G (carry generation) : $G = A \cdot B$
    - With P (carry propagation) : $P = A \oplus B$
  - Q2.3 : Design a boolean function computing the output carry using previous equations and pass transistor logic

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .

Pass Transistor Logic

Summary

- Requires less transistors than CMOS logic . . .
- Partialy passive logic. . .
  - Requires buffers every 2 to 4 paths. . .
  - Performances of a gate alone are difficult to assess : Performance evaluation at the assembly level.. . .
- Gate inputs are not only transistor Gates but also transistor Drains or Sources.
  - Input capacitor depends from the state of the pass gate.. . .
- Preferred Use :
  - Big "full-custom" digital blocs optimized for speed. . .
  - Embedded inside CMOS standard-cells ("automated design").. . .