Logic Design

Implementation Technology

McMaster University

Outline

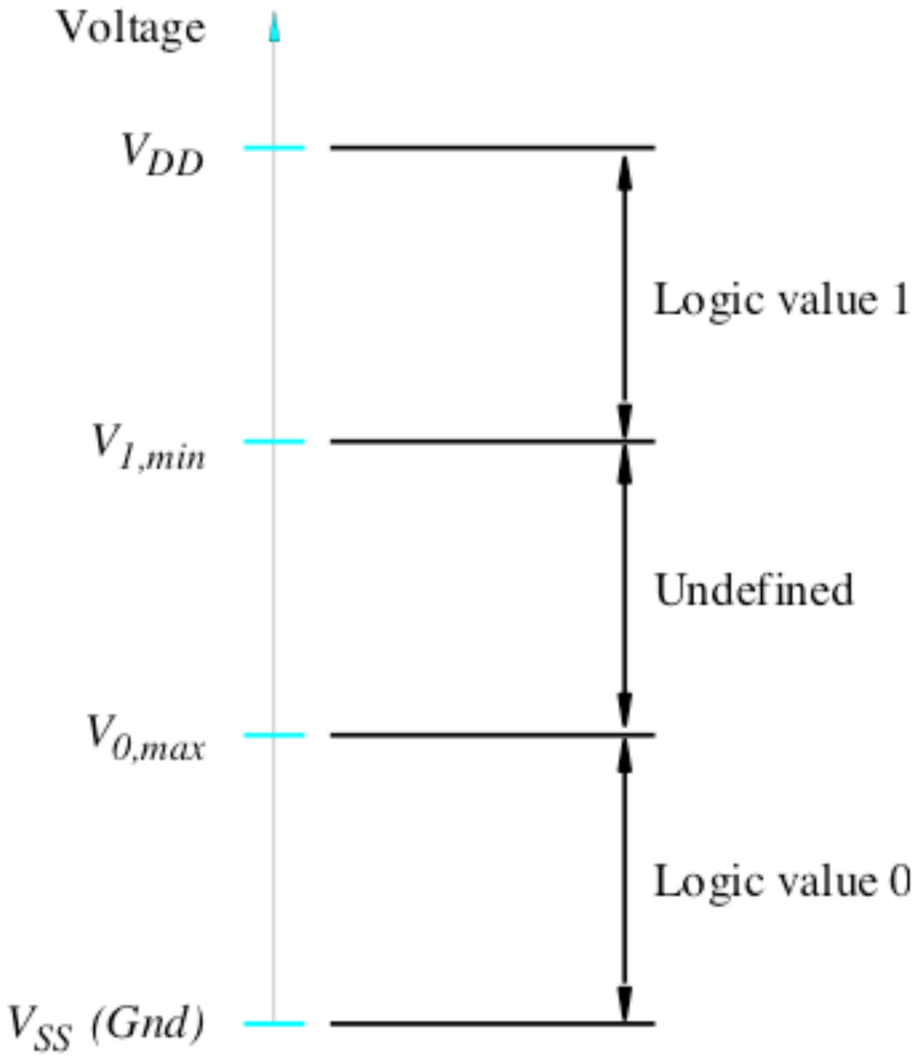
- Implementation of logic gates using transistors
- Programmable logic devices
  - Complex Programmable Logic Devices (CPLD)
  - Field Programmable Gate Arrays (FPGA)
- Dynamic operation of logic gates
- Transmission gates

McMaster University

Transistor as a Switch

- Logic circuits are implemented using transistors
- Logic variables (0 and 1) are represented either as levels of current or voltage.
- We will use voltage.
- A threshold is define and any voltage below threshold is one logic value and any voltage above threshold is the other logic value
- Positive logic system: logic zero is represented by low voltage and logic 1 is represented by higher voltage
- Negative logic system: logic zero is represented by high voltage and logic 1 is represented by low voltage

Transistor as a Switch

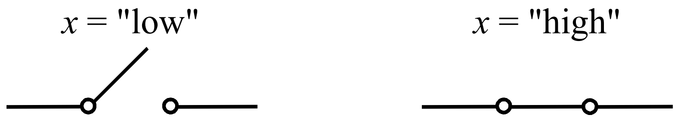


Transistor as a Switch

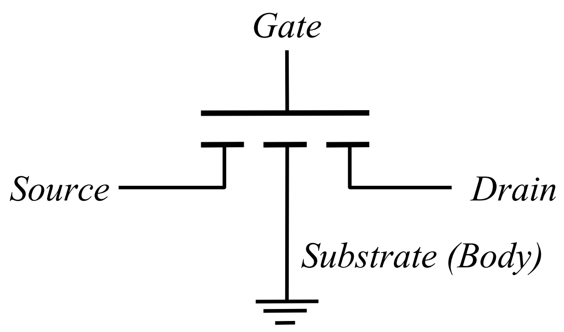
- Transistors in logic circuits operates as switches
- Most popular type of transistors: metal oxide semiconductor field effect transistor (MOSFET)
- Two types of MOSFET: n-channel (NMOS) and p-channel (PMOS)
- Terminals: source, drain, gate and substrate
- A MOSFET is controlled by the gate voltage
- NMOS:
  - V_G high -> transistor on
  - V_G low -> transistor off

McMaster University

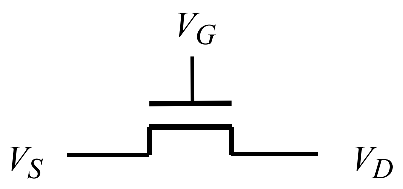
Transistor as a Switch



(a) A simple switch controlled by the input x



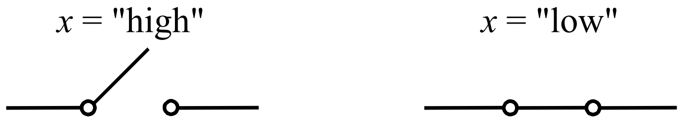
(b) NMOS transistor



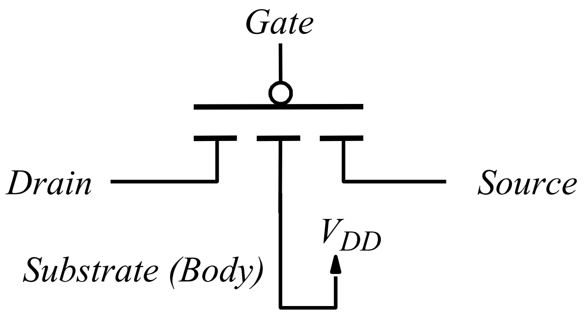
(c) Simplified symbol for an NMOS transistor

Figure 3.2. NMOS transistor as a switch.

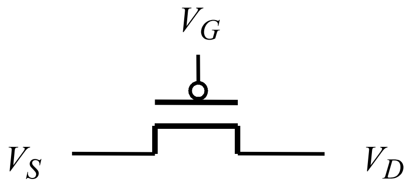
Transistor as a Switch



(a) A switch with the opposite behavior of Figure 3.2 *a*



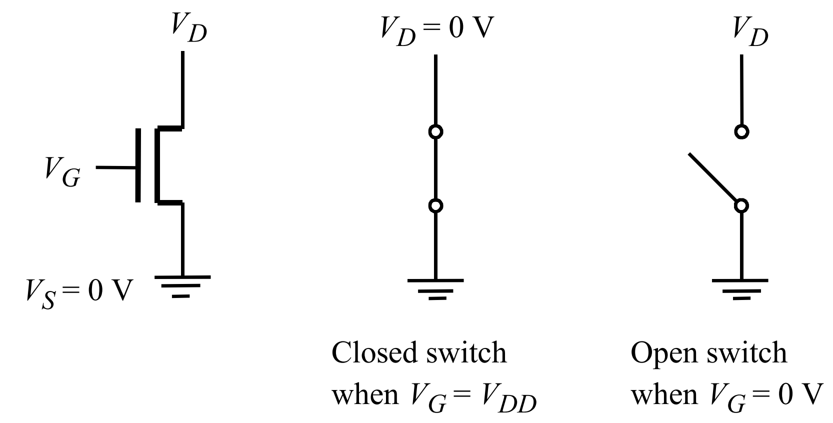
(b) PMOS transistor



(c) Simplified symbol for a PMOS transistor

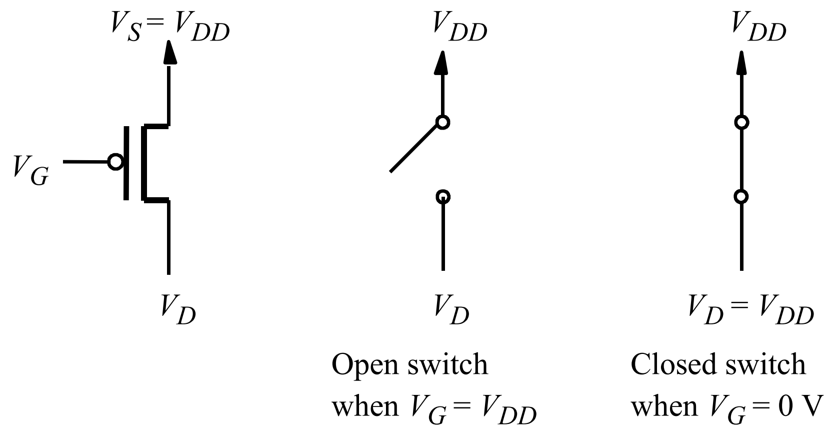
Figure 3.3. PMOS transistor as a switch.

Transistor as a Switch



V_D
V_G
$V_S = 0$ V
$V_D = 0$ V
V_D
Closed switch when $V_G = V_{DD}$
Open switch when $V_G = 0$ V

(a) NMOS transistor



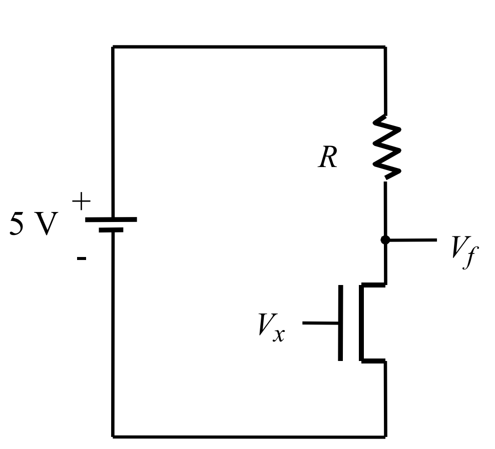
$V_S = V_{DD}$
V_G
V_D
V_{DD}
V_D
V_{DD}
$V_D = V_{DD}$
Open switch when $V_G = V_{DD}$
Closed switch when $V_G = 0$ V

(b) PMOS transistor

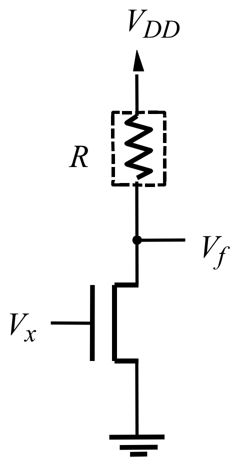
Figure 3.4. NMOS and PMOS transistors in logic circuits.

NMOS Logic Gates

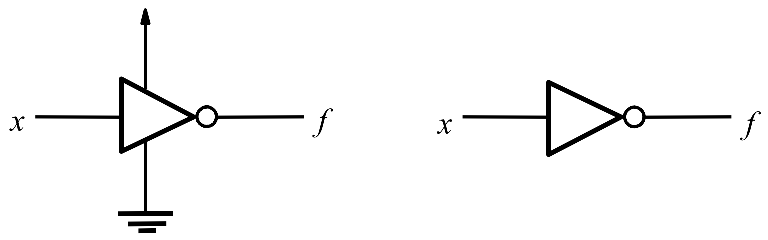
- Consider V_f as a function of V_x



(a) Circuit diagram



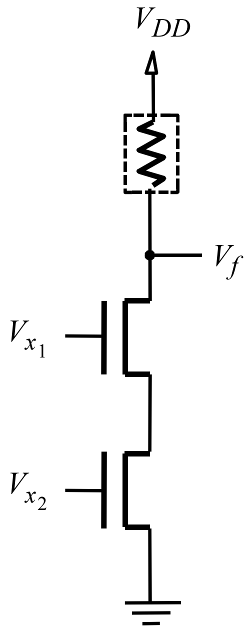
(b) Simplified circuit diagram



(c) Graphical symbols

Figure 3.5. A NOT gate built using NMOS technology.

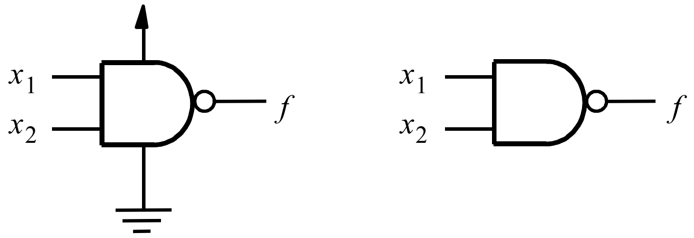
NMOS Logic Gates



(a) Circuit

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

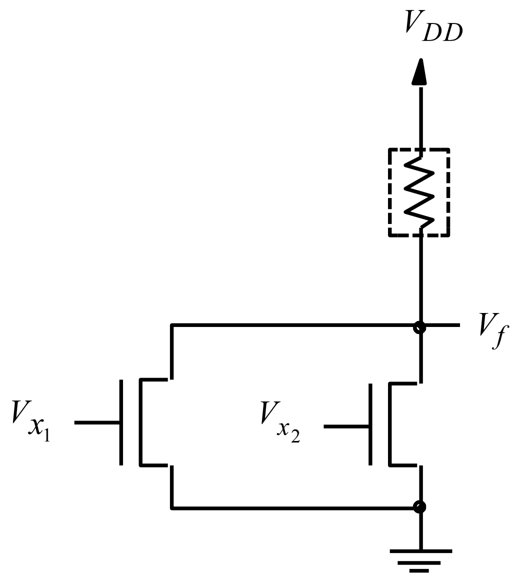
(b) Truth table



(c) Graphical symbols

Figure 3.6. NMOS realization of a NAND gate.

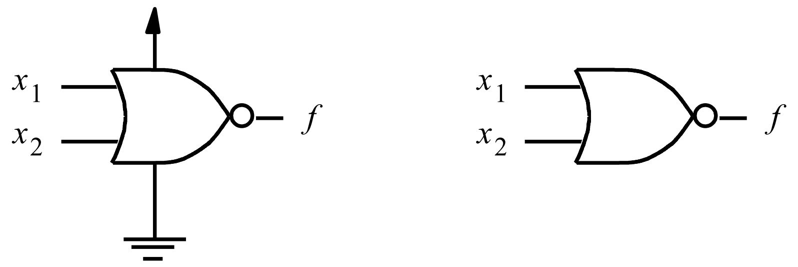
NMOS Logic Gates



(a) Circuit

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

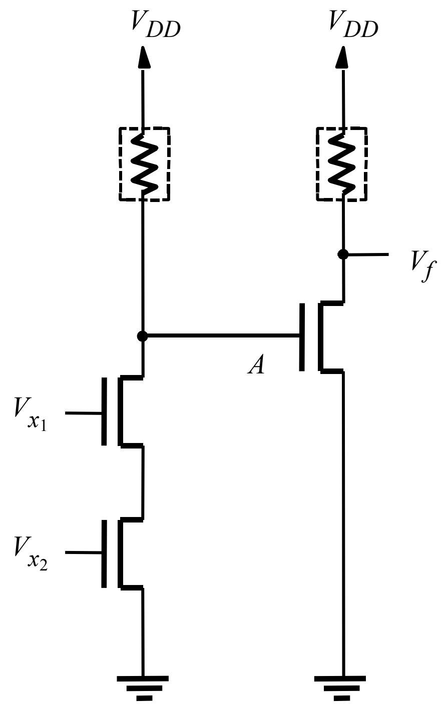
(b) Truth table



(c) Graphical symbols

Figure 3.7. NMOS realization of a NOR gate.

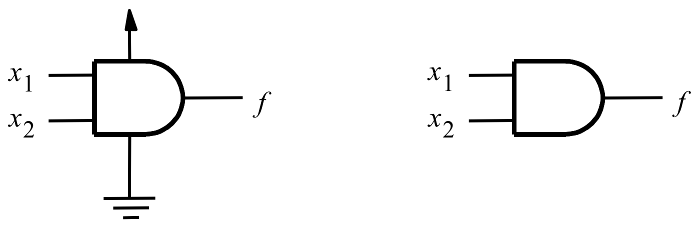
NMOS Logic Gates



(a) Circuit

x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

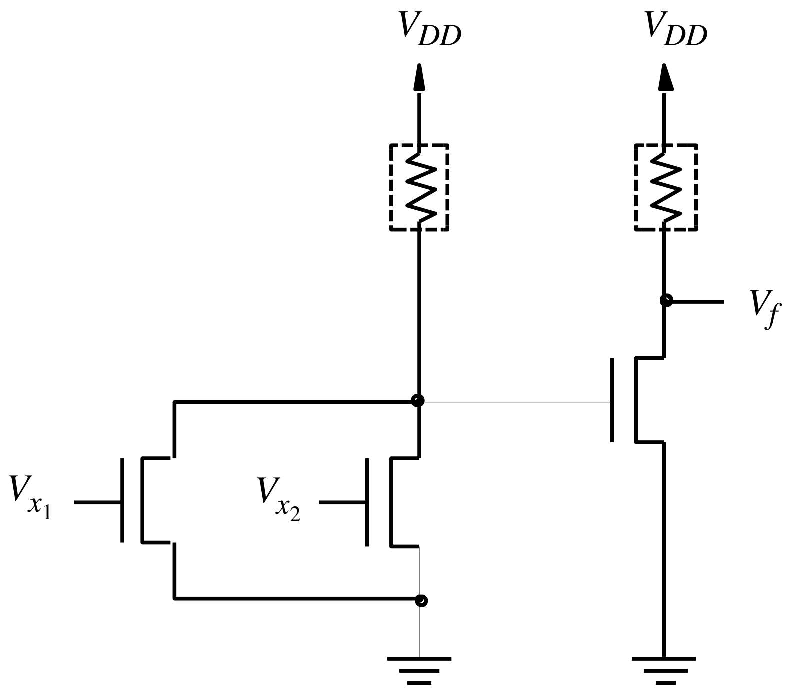
(b) Truth table



(c) Graphical symbols

Figure 3.8. NMOS realization of an AND gate.

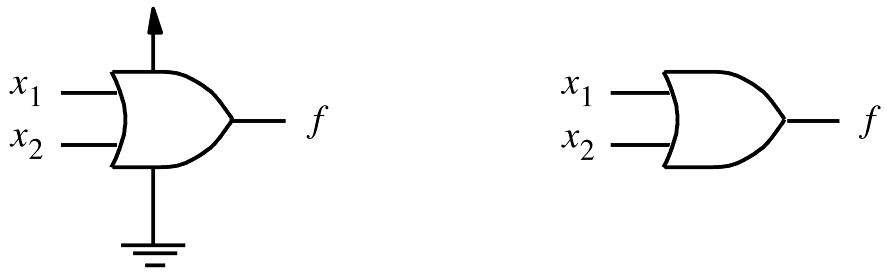
NMOS Logic Gates



(a) Circuit

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

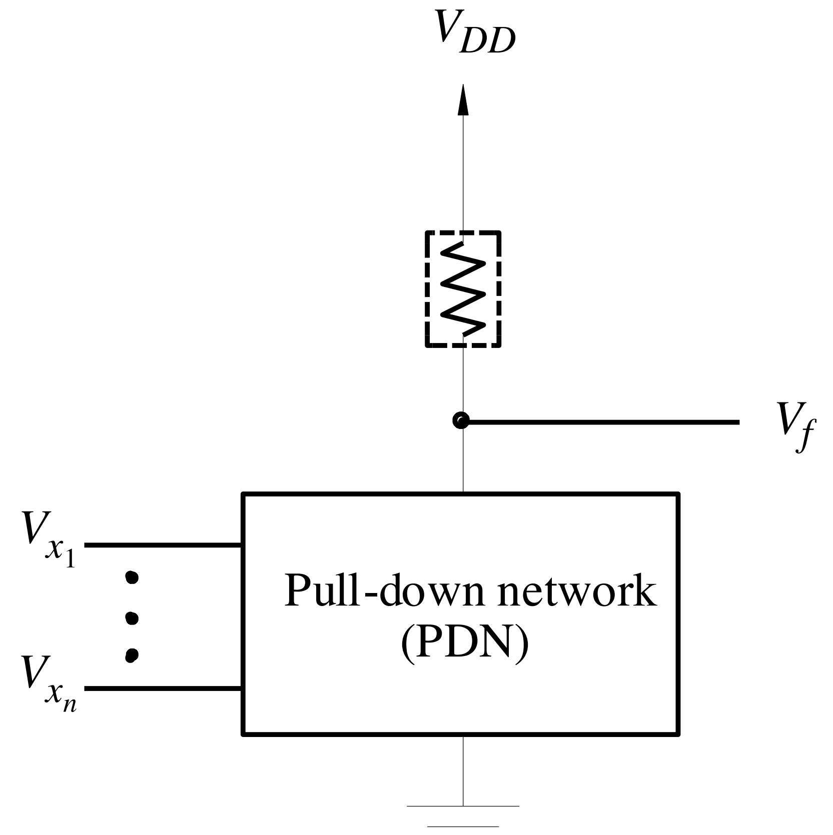
(b) Truth table



(c) Graphical symbols

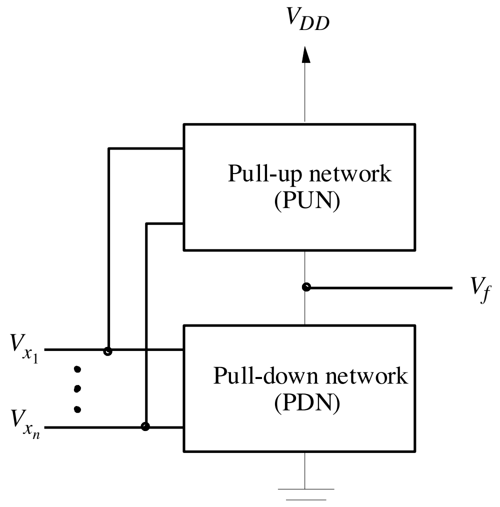
CMOS Gates

- In NMOS the logic functions are realized by arrangements of NMOS transistors combined with a pull-up device that acts as a resistor.

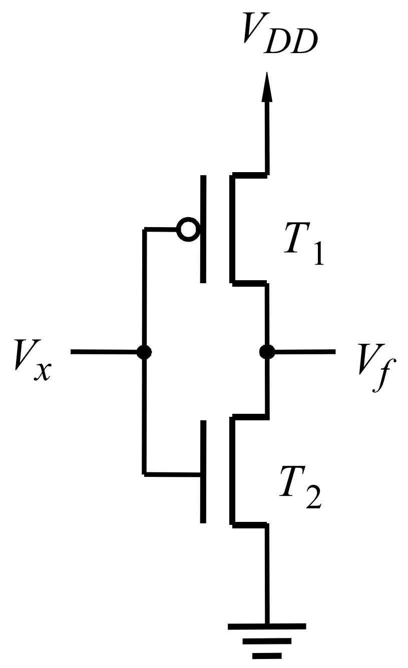


CMOS Gates

- In CMOS circuits the pull-up device is replaced by a pull-up network (PUN) build using PMOS transistors
- For any valuation of inputs, either PDN pulls V_f down to Gnd or PUN pulls V_f up to V_{DD}
- PDN and PUN have equal number of transistors, which are arranged so that the networks are dual



CMOS Gates



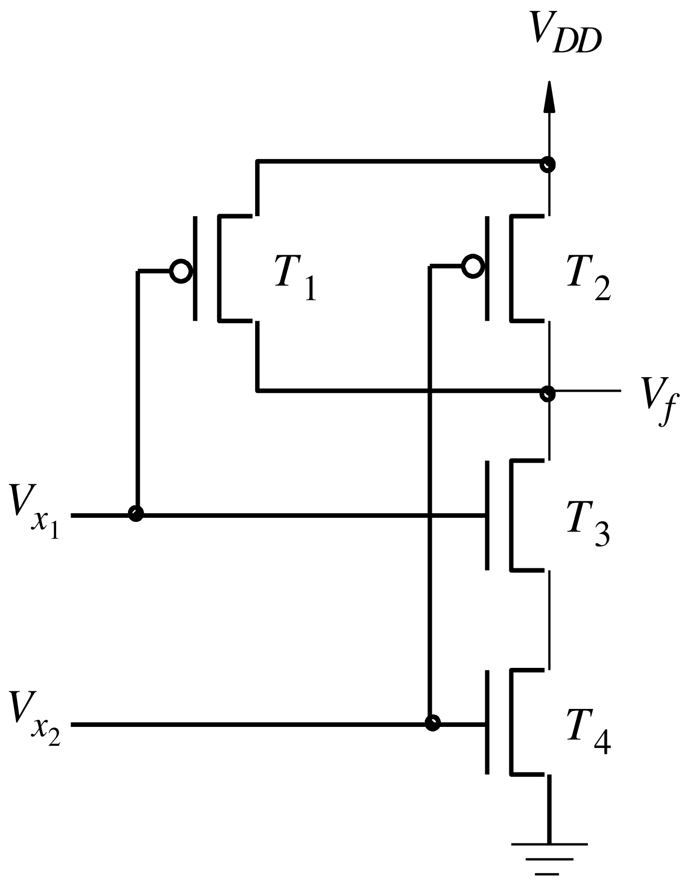
(a) Circuit

x	T_1	T_2	f
0	on	off	1
1	off	on	0

(b) Truth table and transistor states

Figure 3.12. CMOS realization of a NOT gate.

CMOS Gates



(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states

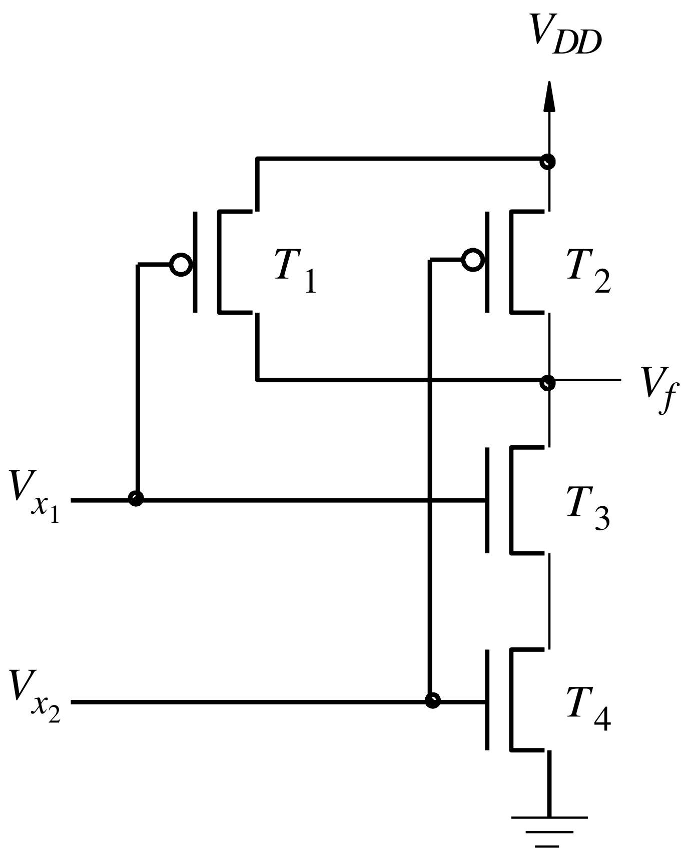
CMOS Gates

- How to design the transistor circuit for a logic function f?
- f=1 gives the PUN
- f=0 gives the PDN

$$f = (x_1 x_2)' = x_1' + x_2'$$

$$f' = x_1 x_2$$

CMOS Gates



(a) Circut

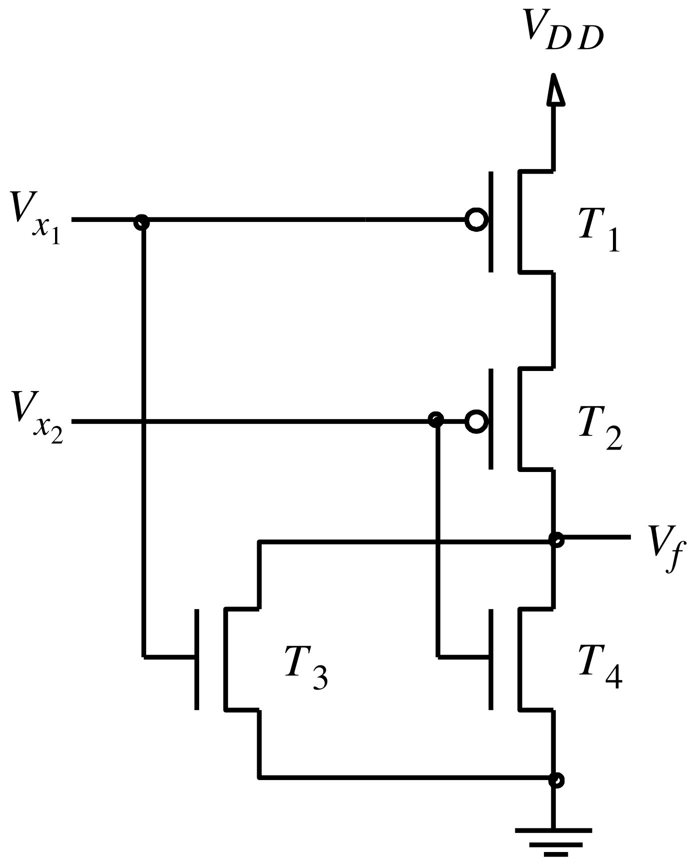
x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states

NOR Gate

$$f = (x_1 + x_2)' = x_1' x_2'$$

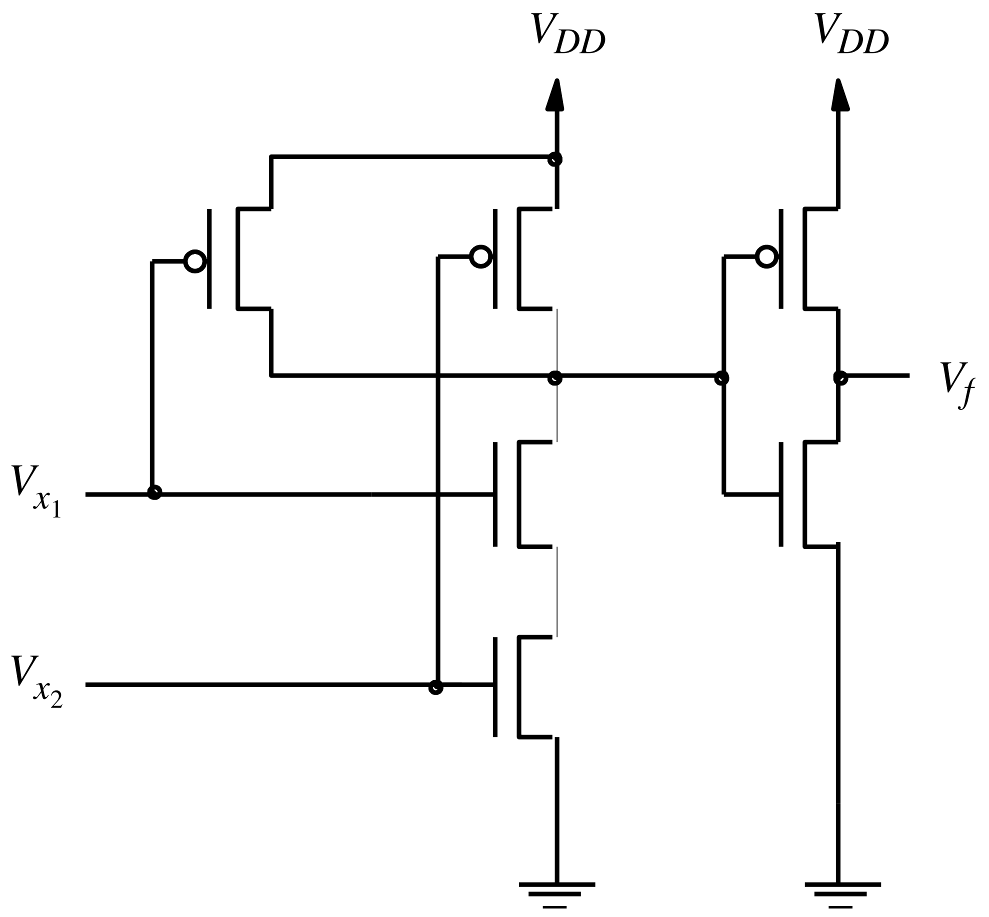
$$f' = x_1 + x_2$$



(a) Circuit

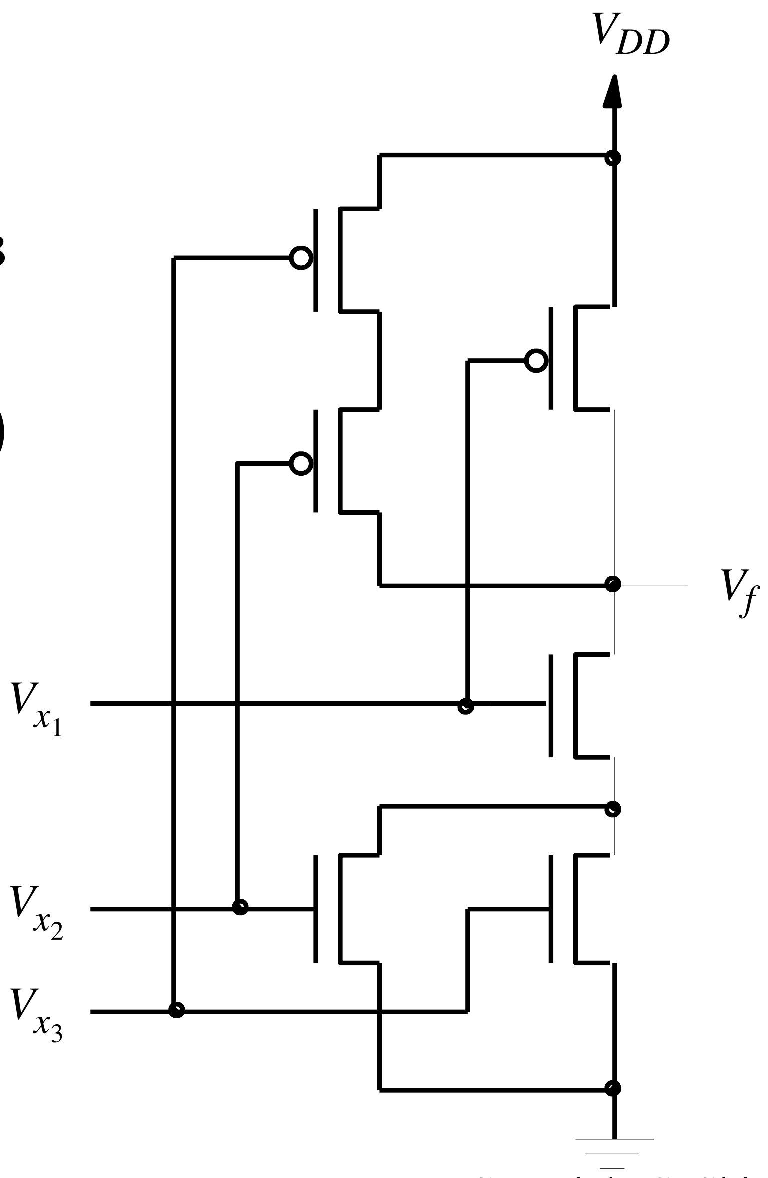
x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

(b) Truth table and transistor states


V_{DD}
V_{DD}
V_f
V_{x_1}
V_{x_2}

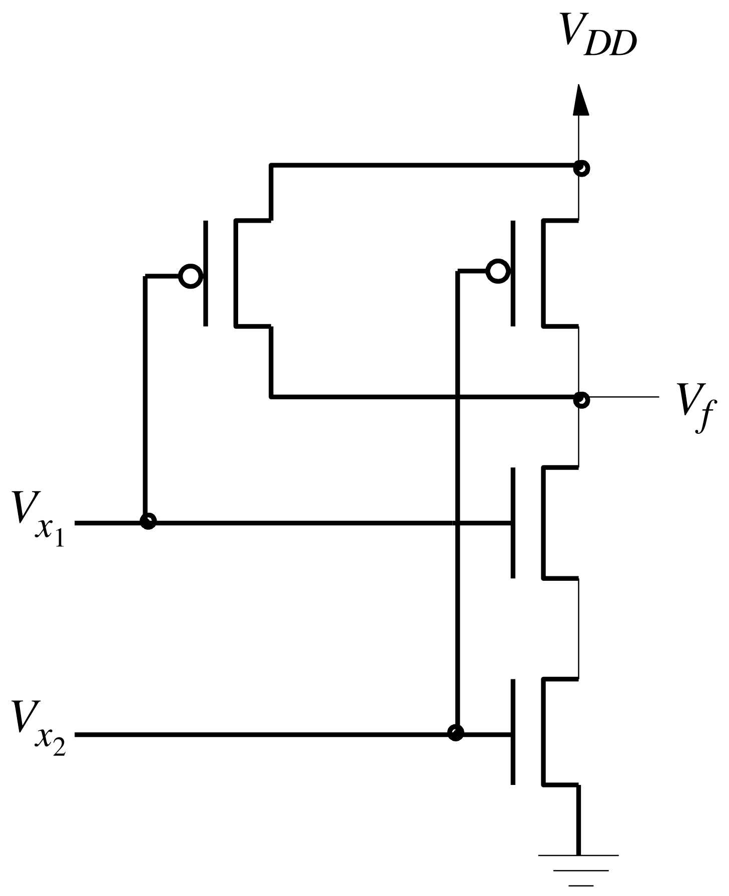
$$f = x_1' + x_2' x_3'$$

$$f' = (x_1' + x_2' x_3')' = x_1(x_2 + x_3)$$



Negative Logic System

- Negative logic system: lower voltage represents 1



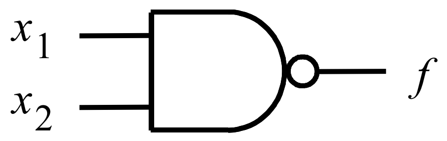
(a) Circuit

V_{x_1}	V_{x_2}	V_f
L	L	H
L	H	H
H	L	H
H	H	L

(b) Voltage levels

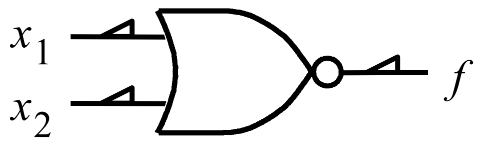
Negative Logic System

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0



(a) Positive logic truth table and gate symbol

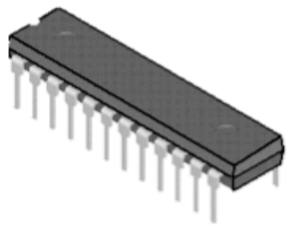
x_1	x_2	f
1	1	0
1	0	0
0	1	0
0	0	1

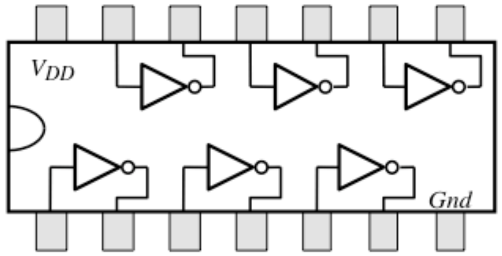


(b) Negative logic truth table and gate symbol

Standard Chips

- An approach used widely until mid-1980s was to connect chips each containing only a few logic gates
- 7400 series: chips with different logic gates


(a) Dual-inline package



(b) Structure of 7404 chip

Figure 3.21. A 7400-series chip.

Standard Chips

- For each specific 7400 series chip several variants are build with different technologies
- 74LS00: build with transistor-transistor logic (TTL)
- 74HC00: build using CMOS technology

Standard Chips

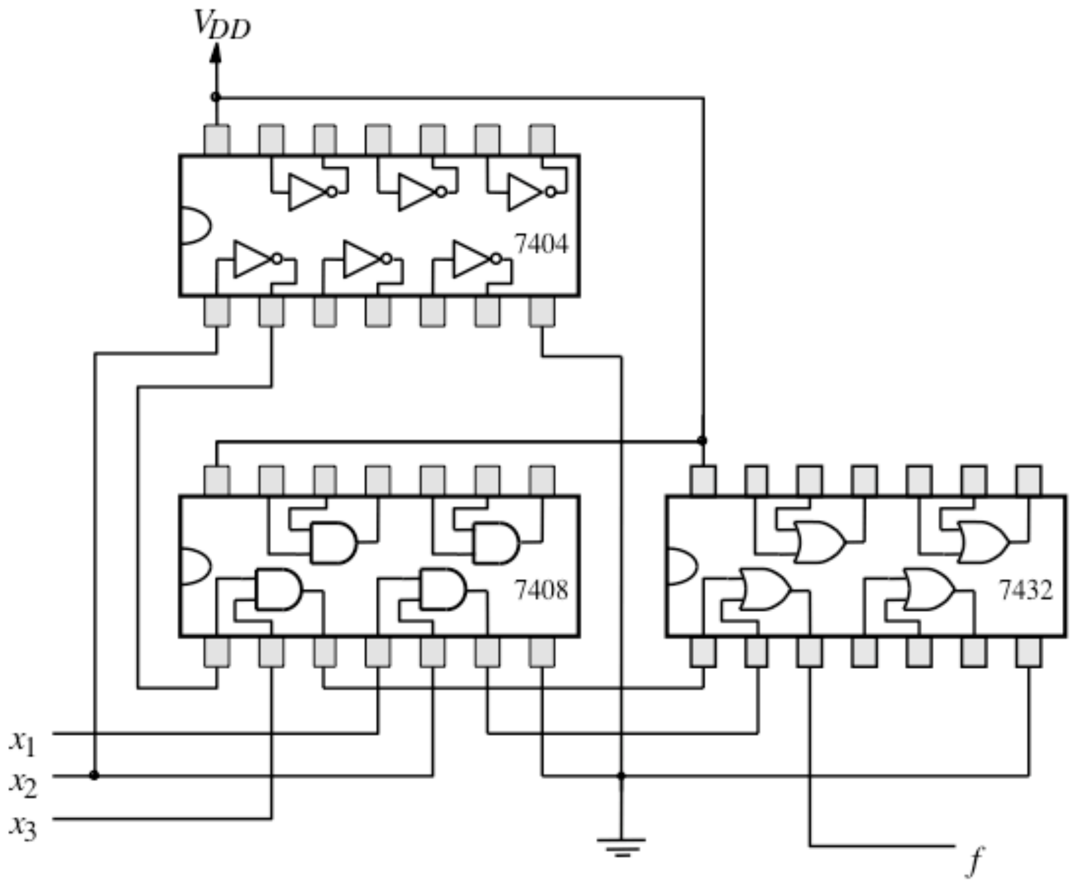
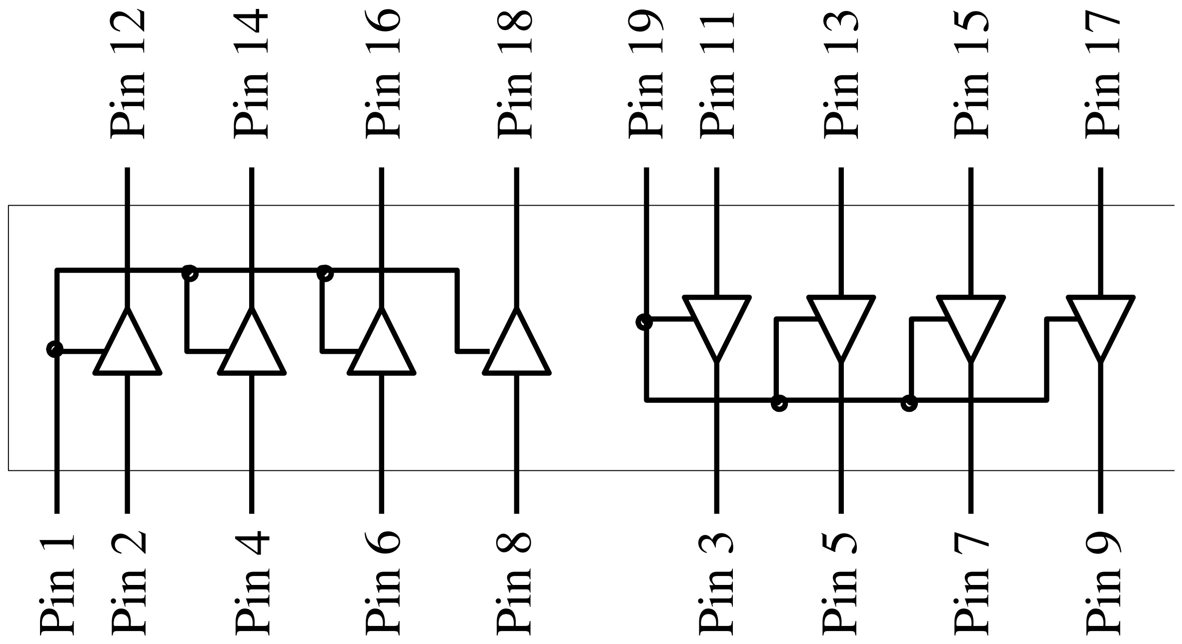


Figure 3.22. An implementation of $f = x_1x_2 + \bar{x}_2x_3$.

Standard Chips

- Because of their low capacity, standard chips are seldom used today except for buffers



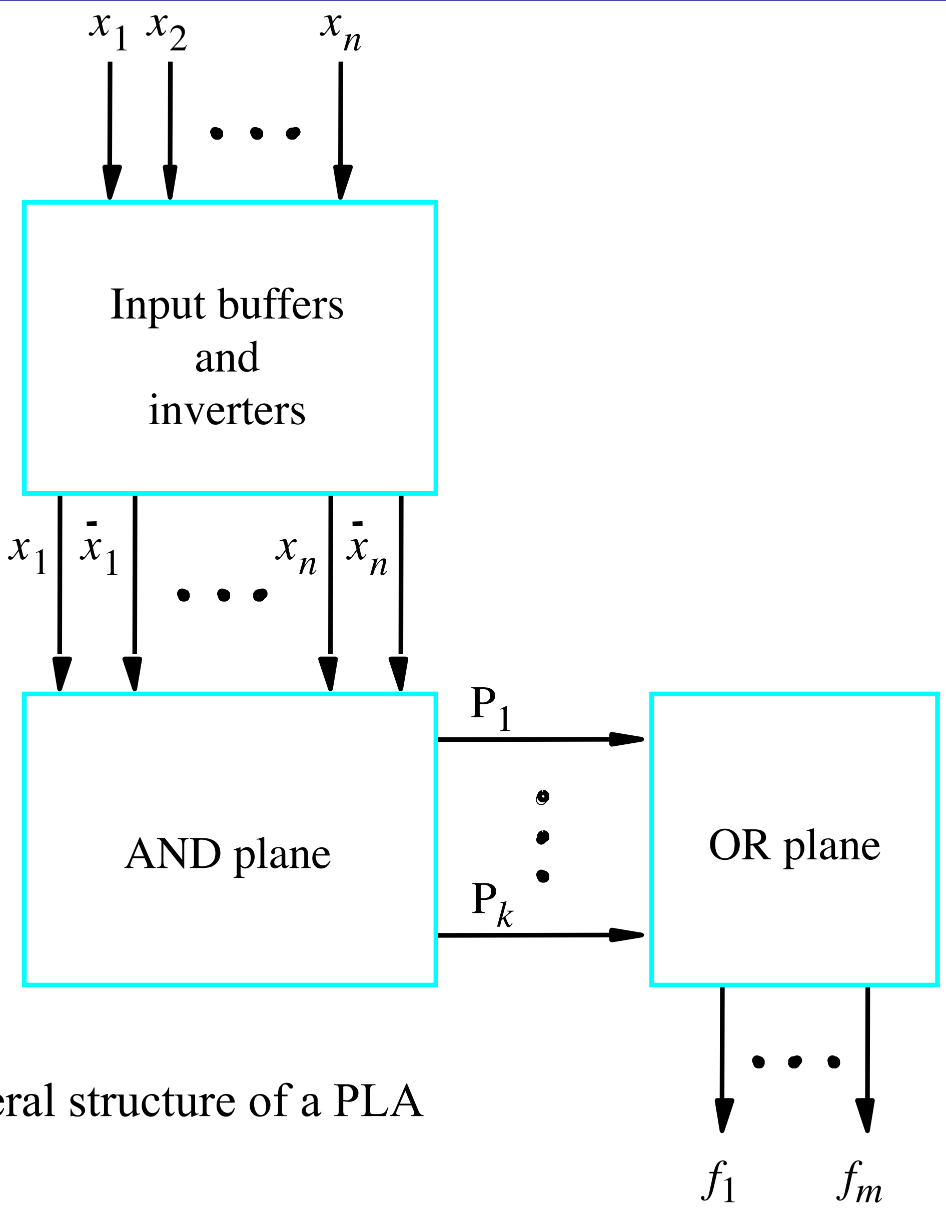
Programmable Logic Devices

- The function provided by each of the 7400 series parts is fixed
- Each chip contains only a few logic gates
- These chips are inefficient for building large circuits
- Solution: manufacture chips that contain relatively large amount of logic with a structure that is not fixed
- These chips are called programmable logic devices (PLDs)
- Several types of PLDs are available: PLA, PAL, CPLD, and FPGA

Programmable Logic Technology

- Simple programmable logic devices (PLDs) such as programmable logic array (PLA) and programmable array logic (PAL) have been in use for over 20 years.
- PLA: the idea is that logic functions can be realized in sum-of products form

McMaster University


x_1 x_2 x_n
Input buffers and inverters
x_1 $\bar{x}_1$ x_n $\bar{x}_n$
AND plane
P_1
P_k
OR plane
f_1 f_m

General structure of a PLA

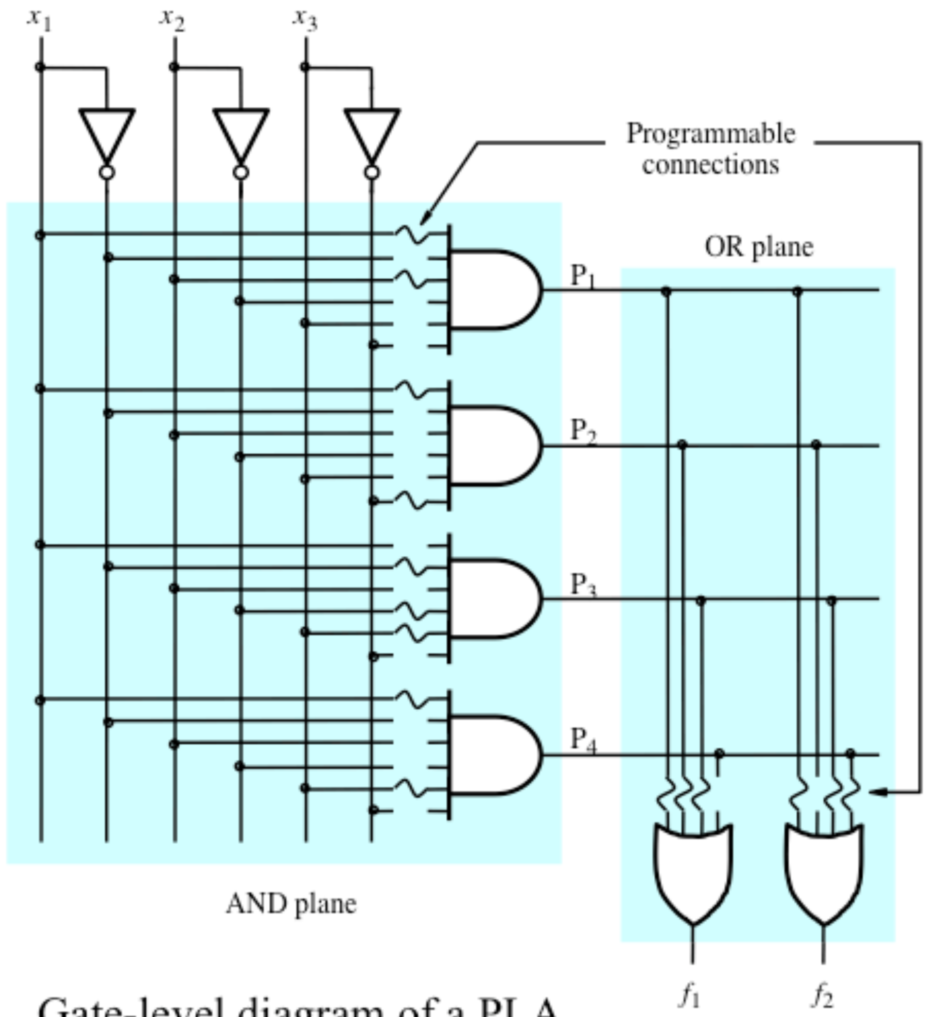


Figure 3.26. Gate-level diagram of a PLA.

McMaster University

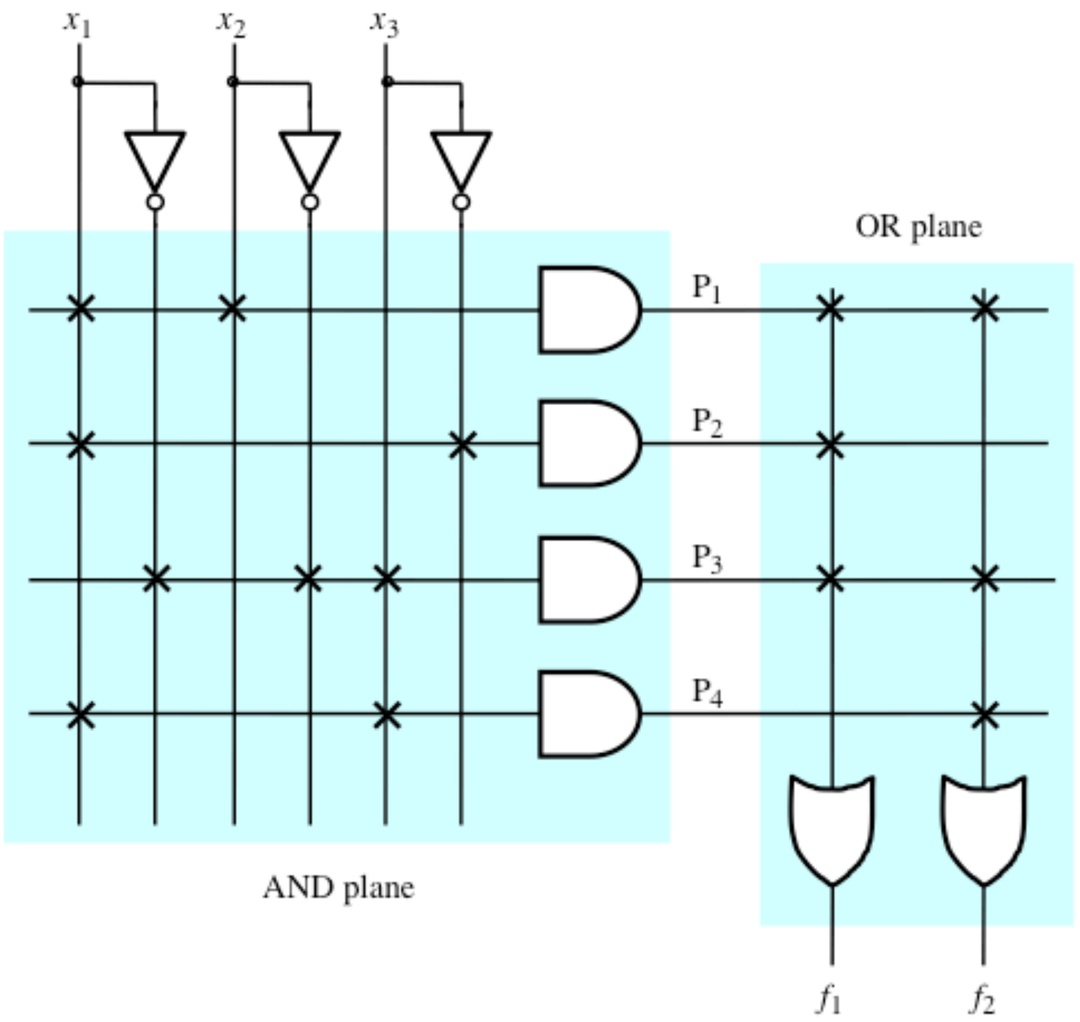


Figure 3.27. Customary schematic for the PLA in Figure 3.26.

Programmable Logic Technology

- Programmable connections (switches) are difficult to fabricate and reduce the speed of circuit
- In PALs the AND plane is programmable but the OR plane is fixed.
- To compensate for reduced flexibility, PALs are manufactured in a range

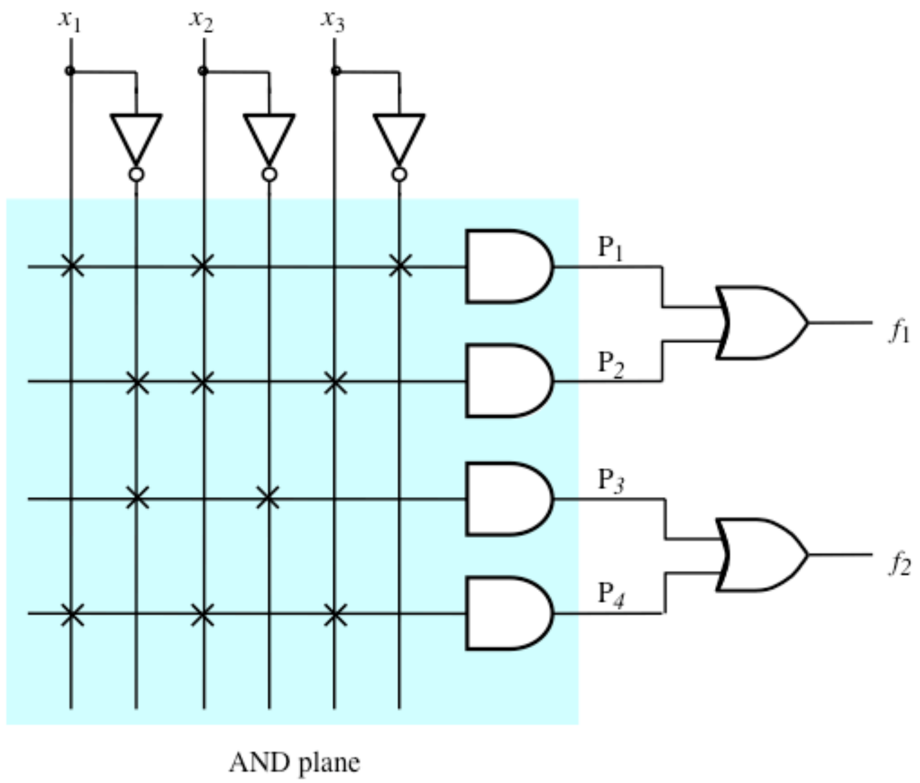


Figure 3.28. An example of a PLA.

Programmable Logic Technology

- On many PLAs and PALs the output of the OR gate is connected to a flip flop whose output can then be feedback as an input into the AND gate array.
- This way simple state machines are implemented

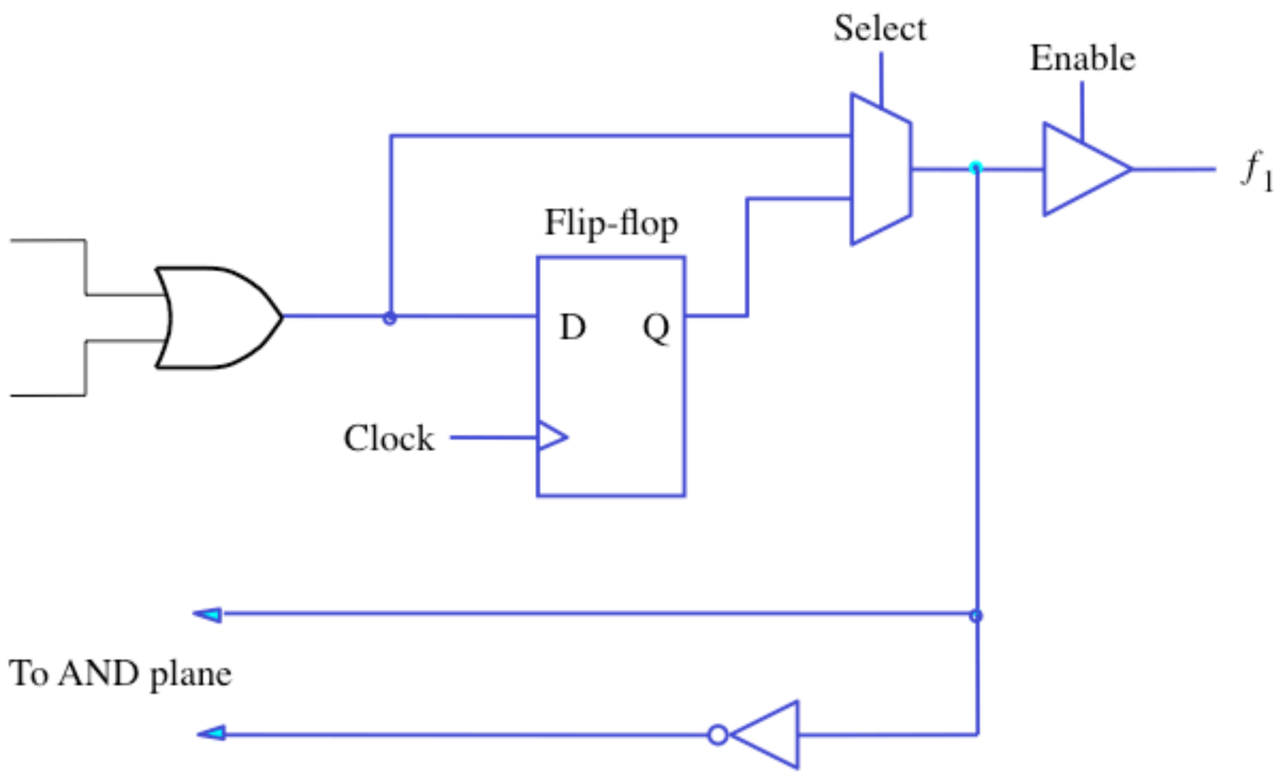


Figure 3.29. Extra circuitry added to OR-gate from Figure 3.28.

Programming PLAs and PALs

- In PLA or PAL switches exist between the inputs and AND gate
- These switches should be programmed to implement a circuit
- There are several thousands programmable switches in commercial chips
- Manual programming is not an option
- CAD systems are employed for this purpose
- CAD is running on a computer that is connected to a programming unit
- CAD generates a file that states how each switch should be to realize the designed circuit

Programming PLAs and PALs

- PLD is placed in the programming unit and the programming file is transferred from the computer system
- Usually PLAs and PALs are part of circuit and are on a printed circuit board
- Usually the chip should be removed from the board for programming
- In system programming is usually not provided for PLAs and PALs but is available for more sophisticated chips

CPLD

- PLAs andPALs are enough for implementing moderate size circuits
- For more sophisticated circuits CPLDs are used
- CPLD: multiple blocks and internal wiring that connects the blocks
- Each block is similar to a PAL or PLA so it is called PAL like block

CPLD

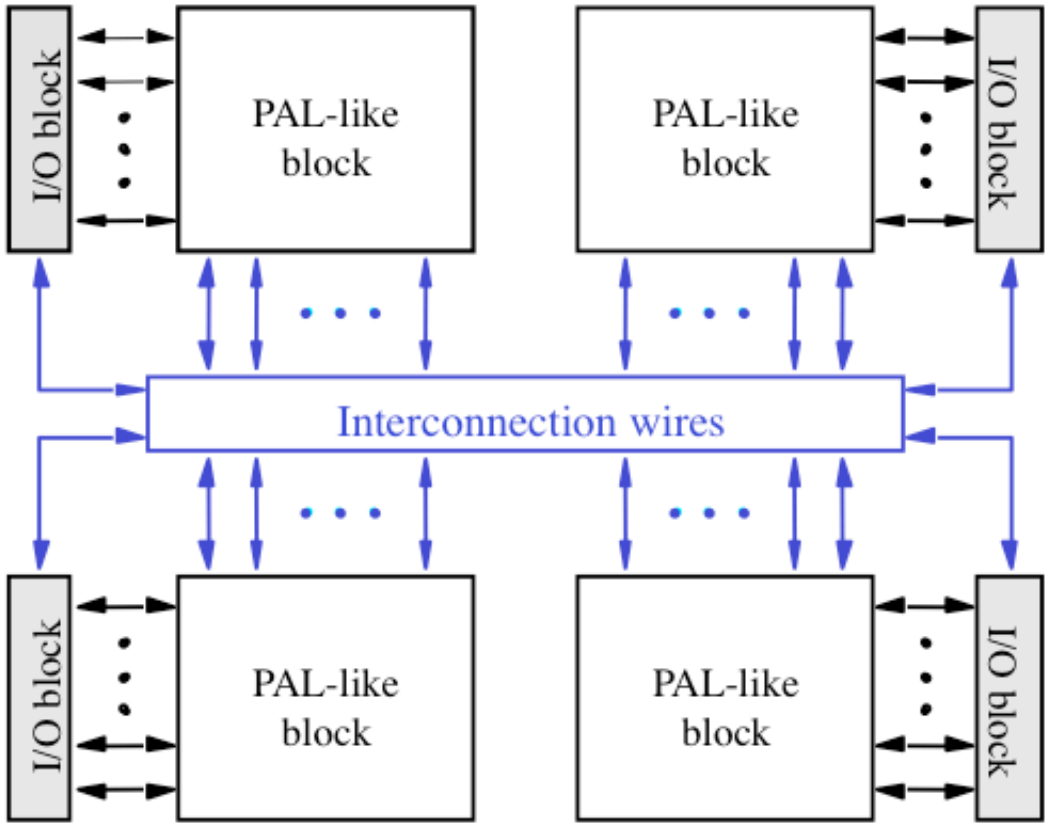


Figure 3.32. Structure of a complex programmable logic device (CPLD).

CPLD

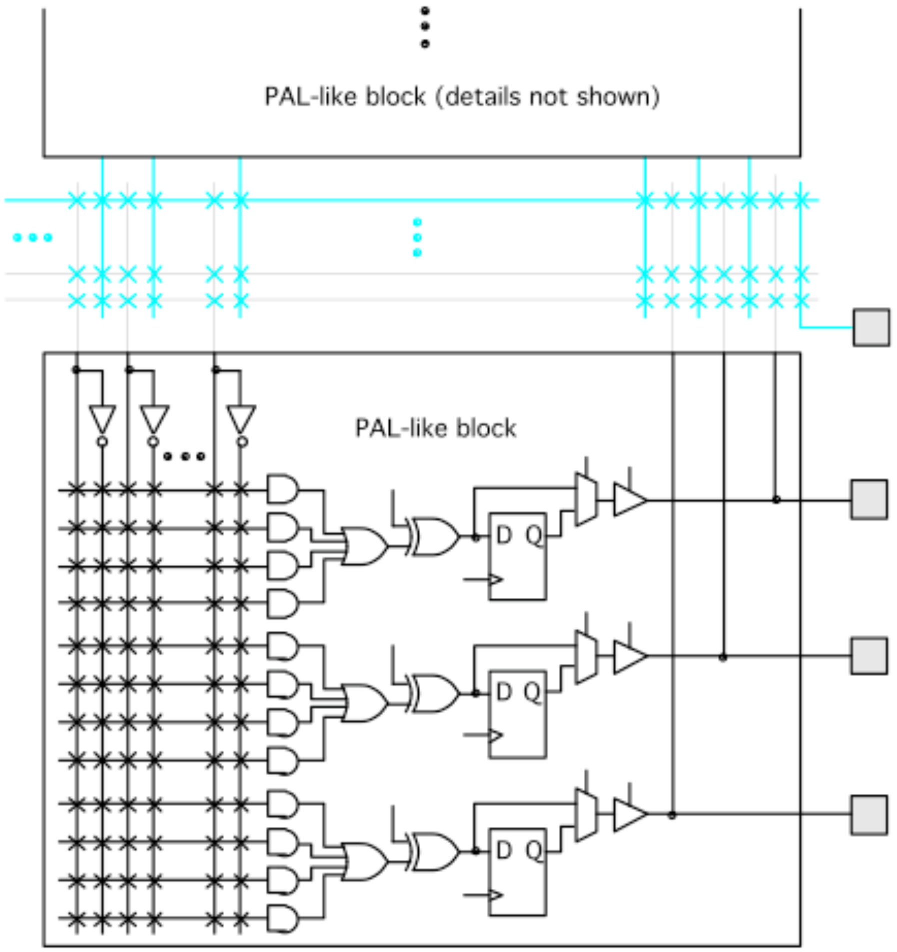
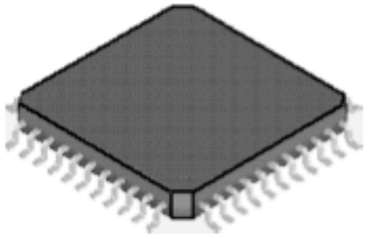


Figure 3.33. A section of the CPLD in Figure 3.32.

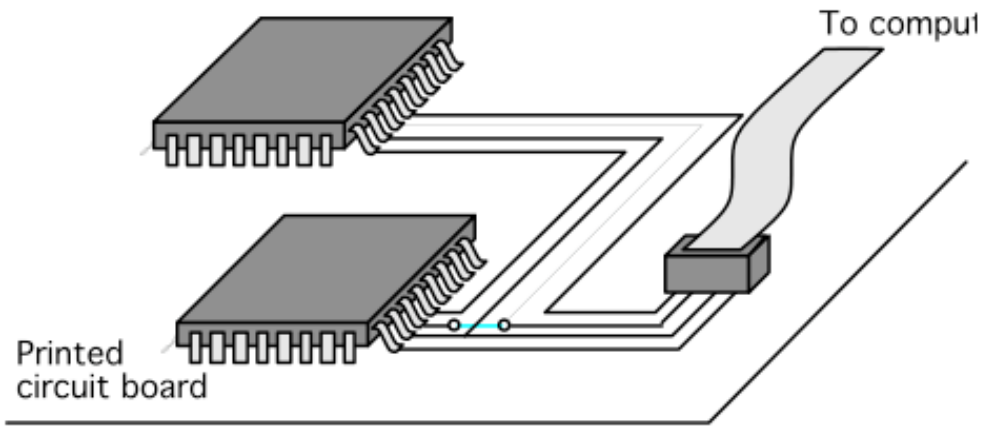
CPLD

- Interconnection wiring contains programmable switches that are used to connect PAL-like blocks
- Commercial CPLDs have 2 to 100 PAL-like blocks
- CPLD devices usually support in system programming
- A small connector is place on the board and for programming it is connected to the computer
- The circuit on CPLD that allows this type of programming is called JTAG (Joint Test Action Group).

CPLD



(a) CPLD in a Quad Flat Pack (QFP) package



(b) JTAG programming

Figure 3.34. CPLD packaging and programming.

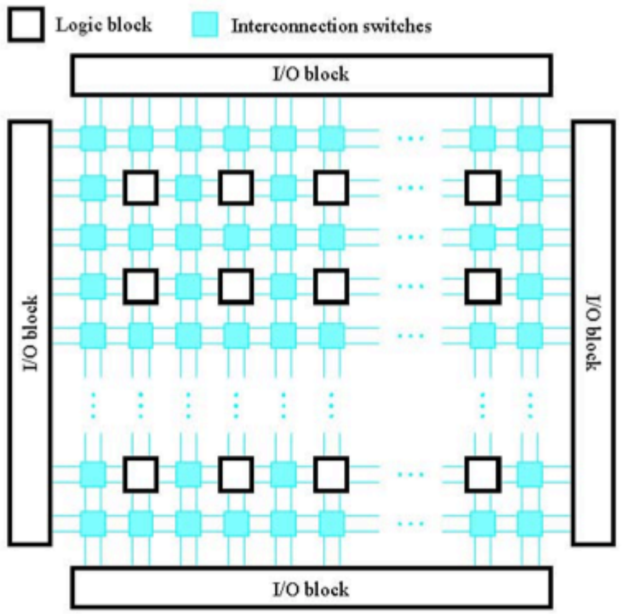
Field Programmable Gate Arrays (FPGA)

- One way to quantify a circuit's size is to assume that the circuits is to be built using only simple gates (e.g., NAND gates) and estimate how many of these gates are needed
- This method is called equivalent gates
- PLA/PAL: 160 gates
- CPLD: 10,000 gates
- Not large by modern standards
- To implement larger circuits we use FPGAs

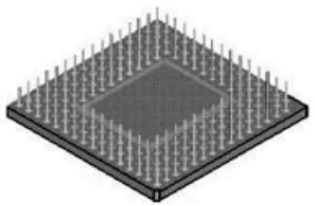
Field Programmable Gate Arrays (FPGA)

- FPGAs: do not contain AND or OR planes
- FPGA contains three main types of resources:
  - logic blocks
  - I/O blocks for connecting to the pins
  - interconnection wires and switches
- Each logic block in an FPGA has a small number of input and outputs
- The most commonly used logic block is a look-up table (LUT)
- LUT: contains storage cells that are used to implement a small logic function

FPGA



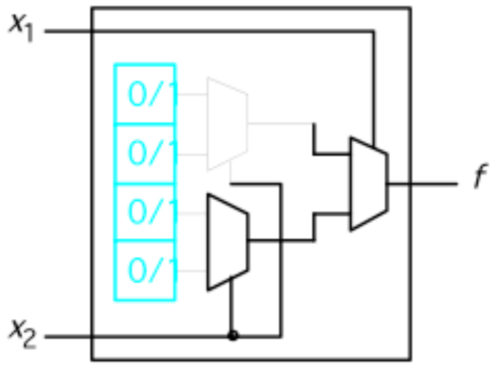
(a) General structure of an FPGA



(b) Pin grid array (PGA) package (bottom view)

Figure 3.35. A field-programmable gate array (FPGA).

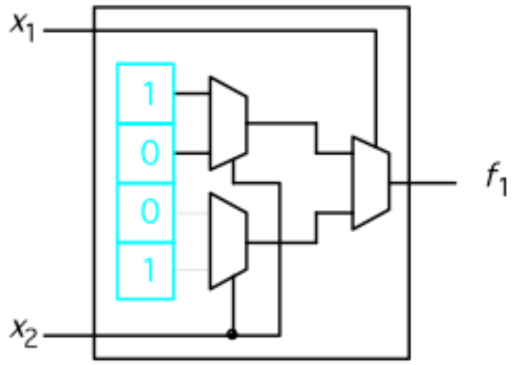
FPGA



(a) Circuit for a two-input LUT

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

(b) $f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$



(c) Storage cell contents in the LUT

Figure 3.36. A two-input lookup table (LUT).

FPGA

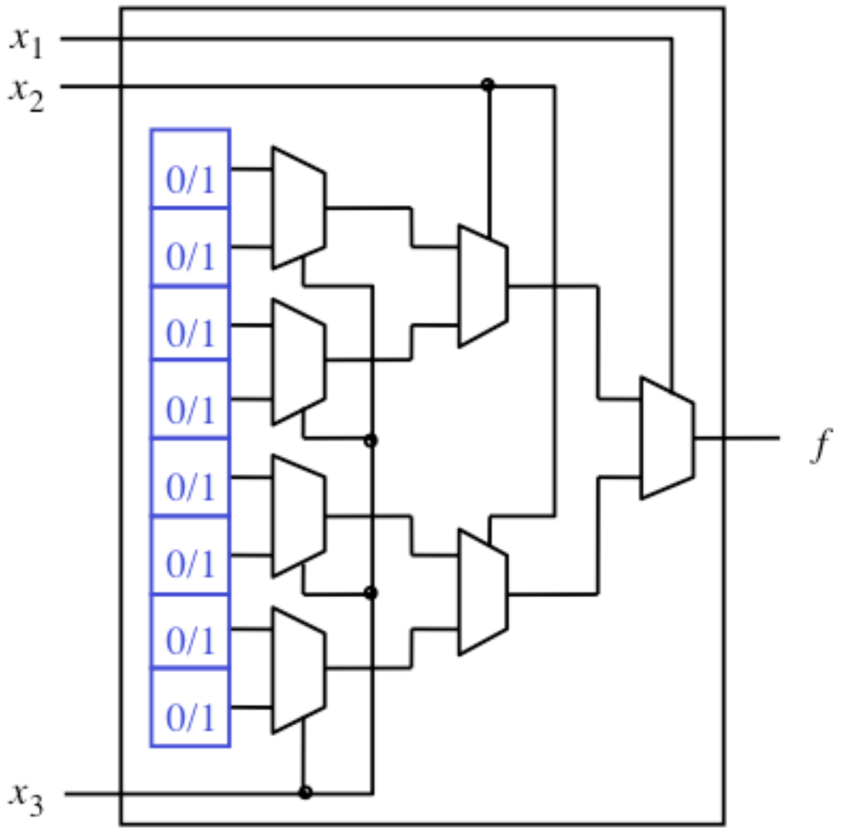


Figure 3.37. A three-input LUT.

FPGA

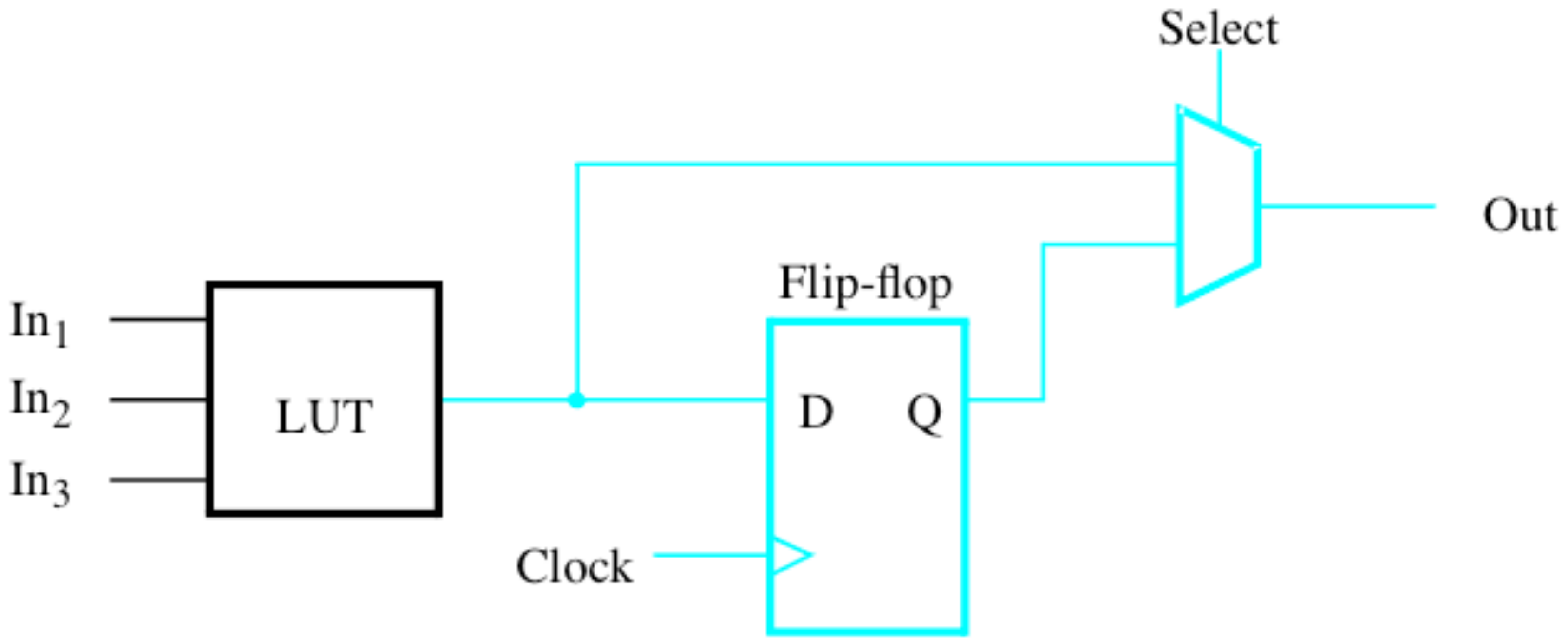


Figure 3.38. Inclusion of a flip-flop in an FPGA logic block.

FPGA

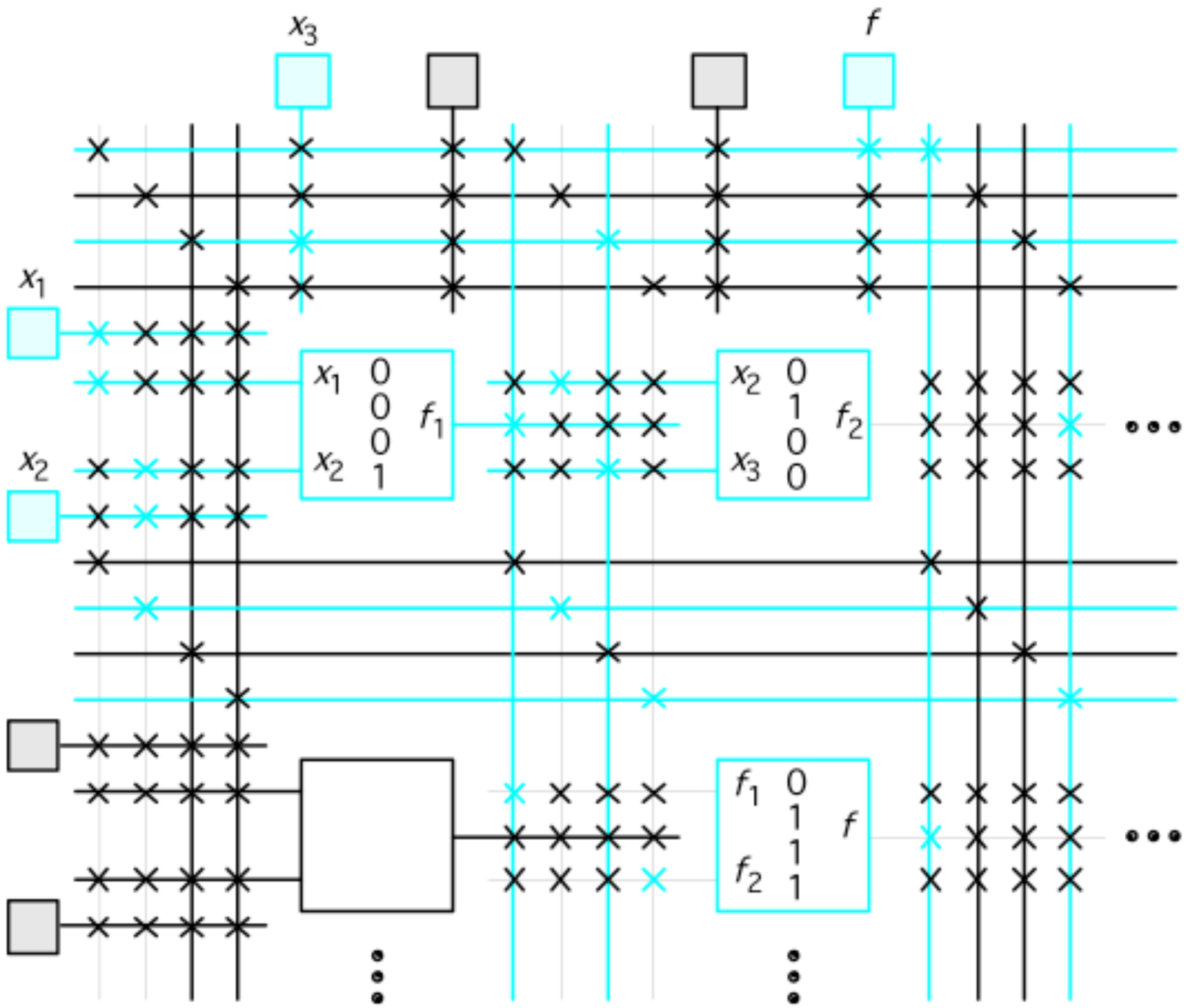
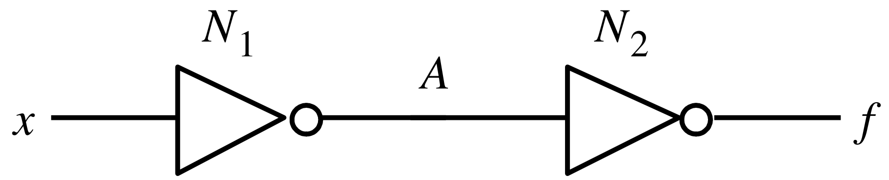


Figure 3.39. A section of a programmed FPGA.

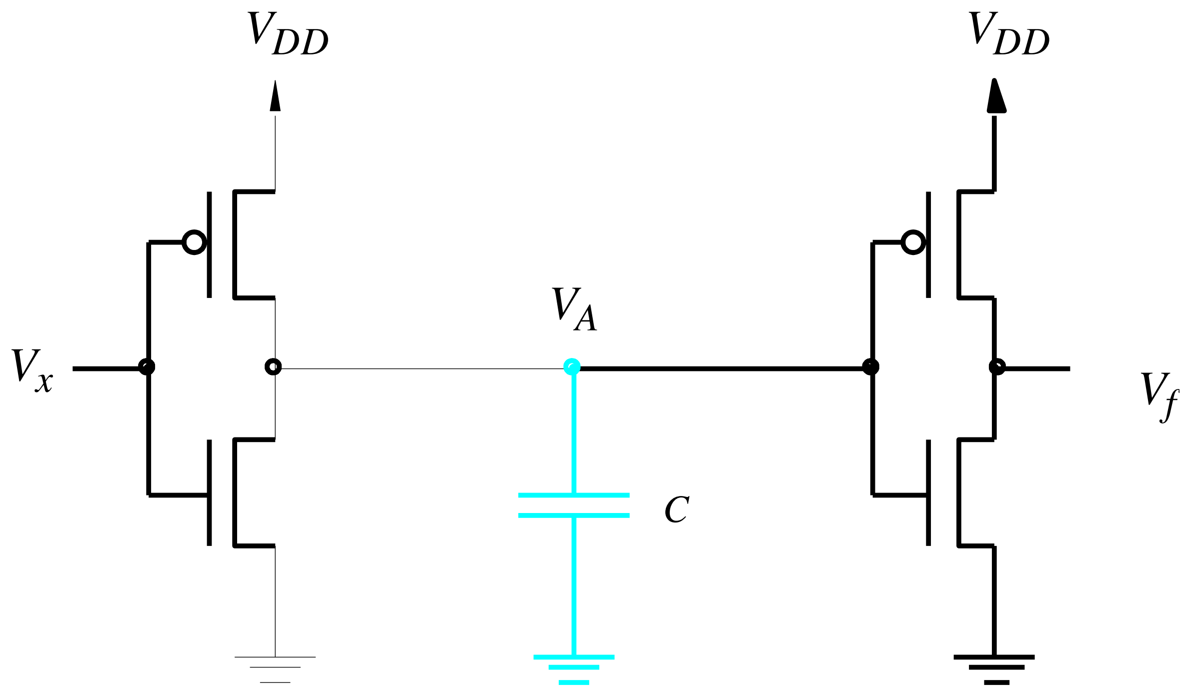
FPGA

- FPGAs are configured by using the in-system programming method
- FPGAs are volatile: they will lose stored contents whenever the power is turned off
- Often a small PROM is included on the board that houses the FPGA and the storage cells are loaded automatically from the PROM when the power is applied

Dynamic operation of logic gates

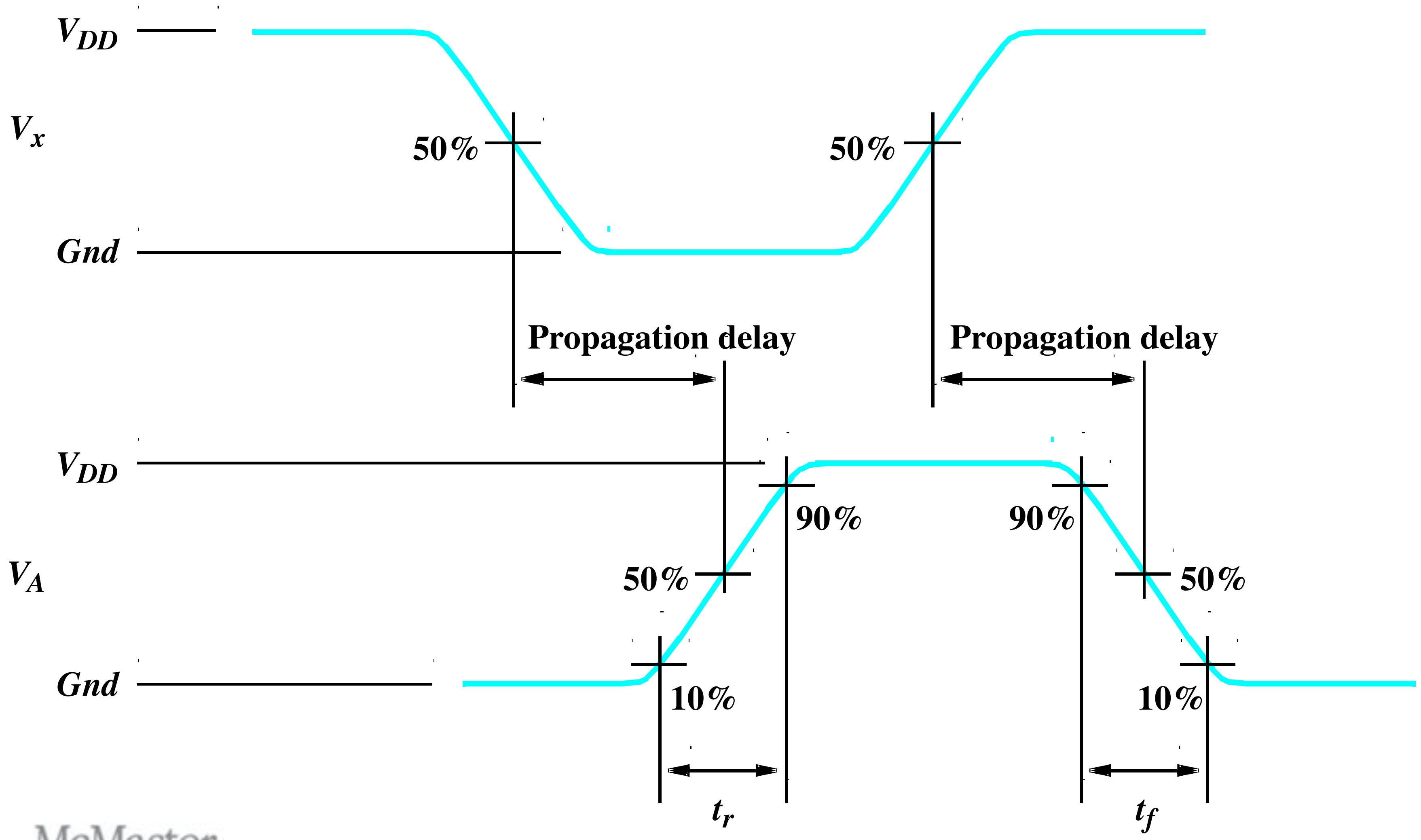


(a) A NOT gate driving another NOT gate

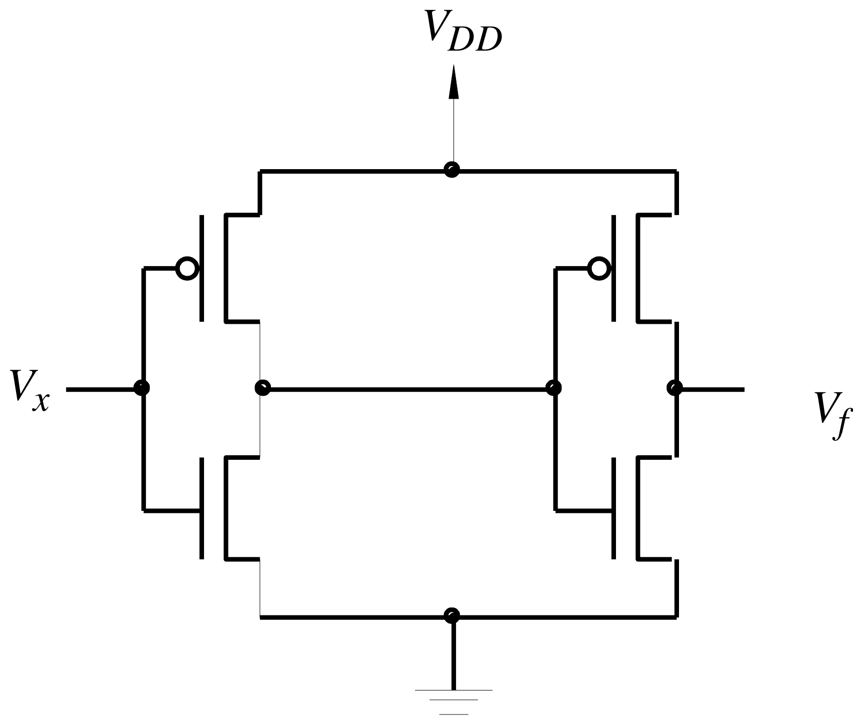


(b) The capacitive load at node A

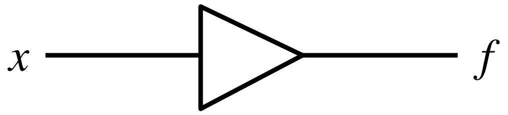
Dynamic operation of logic gates



Buffer

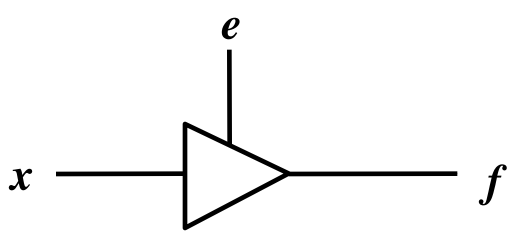


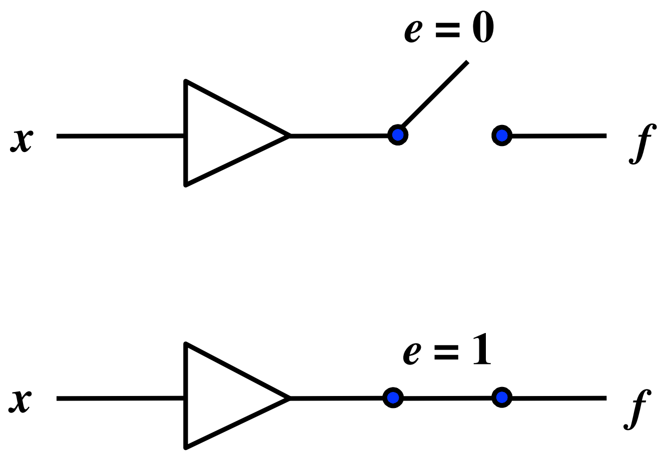
(a) Implementation of a buffer



(b) Graphical symbol

Tri-state Buffer

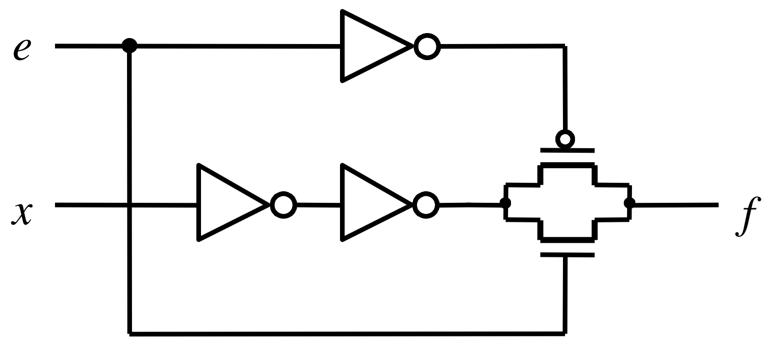




(a) A tri-state buffer **(b) Equivalent circuit**

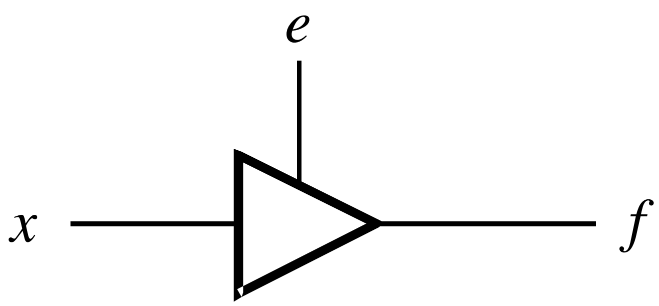
e	*x*	*f*
0	0	Z
0	1	Z
1	0	0
1	1	1

(c) Truth table

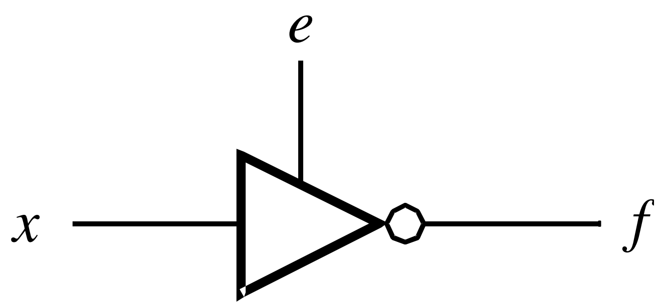


(d) Implementation

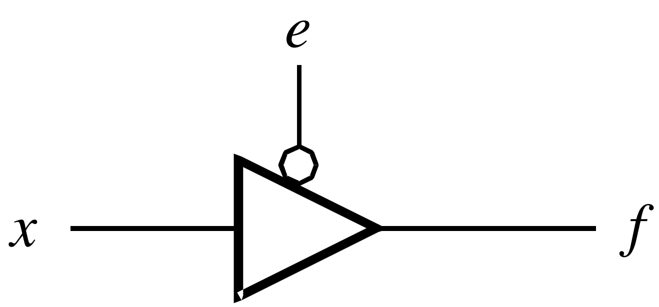
Tri-state Buffer



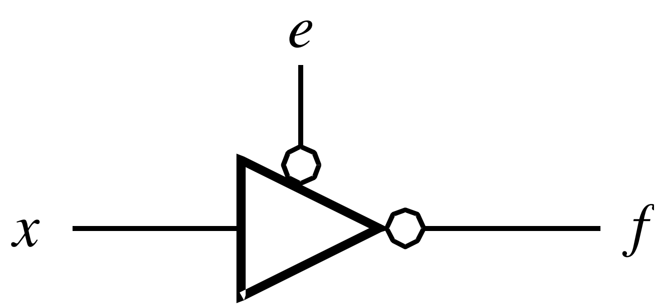
(a)



(b)

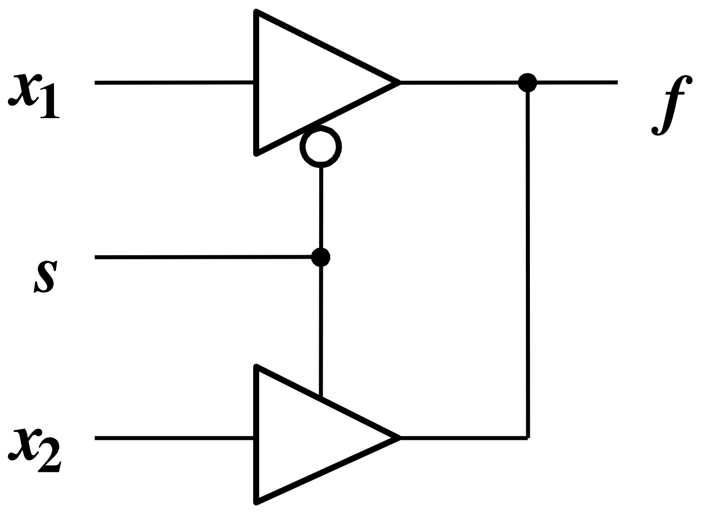


(c)



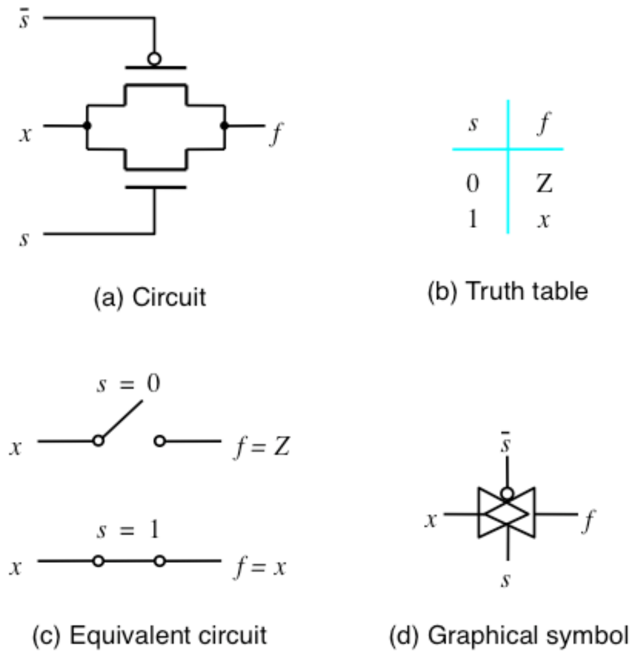
(d)

Tri-state Buffer



Transmission Gate

- It is possible to combine an NMOS and a PMOS transistor into a single switch
- It is called a transmission gate
- Vs=5 and Vs’ =0 turns the switch on.
- When Vx =0, NMOS is on and Vf will be 0
- When Vx=5, PMOS transistor will be on and Vf will be 5

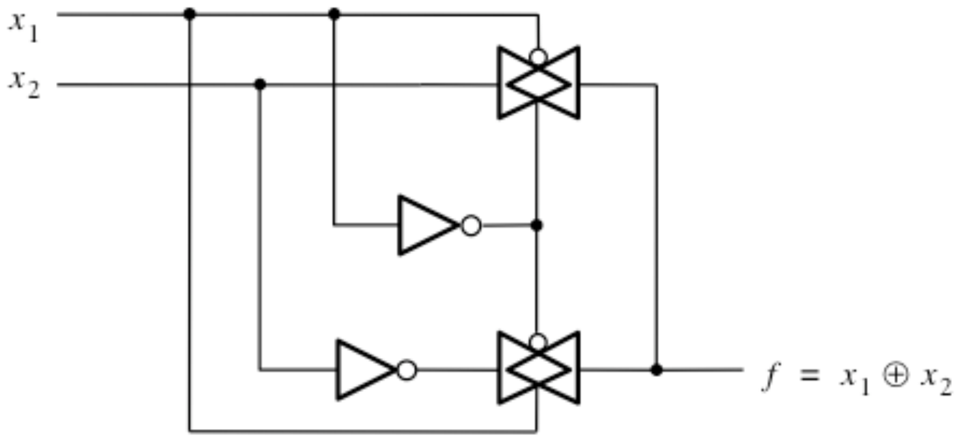


s	f
0	Z
1	x

Figure 3.60. A transmission gate.

XOR

- Implementing XOR using AND and OR and NOT requires 22 transistors.
- This can be reduced using transmission gates



(d) CMOS implementation

Figure 3.61b. CMOS Exclusive-OR gate.

Multiplexer

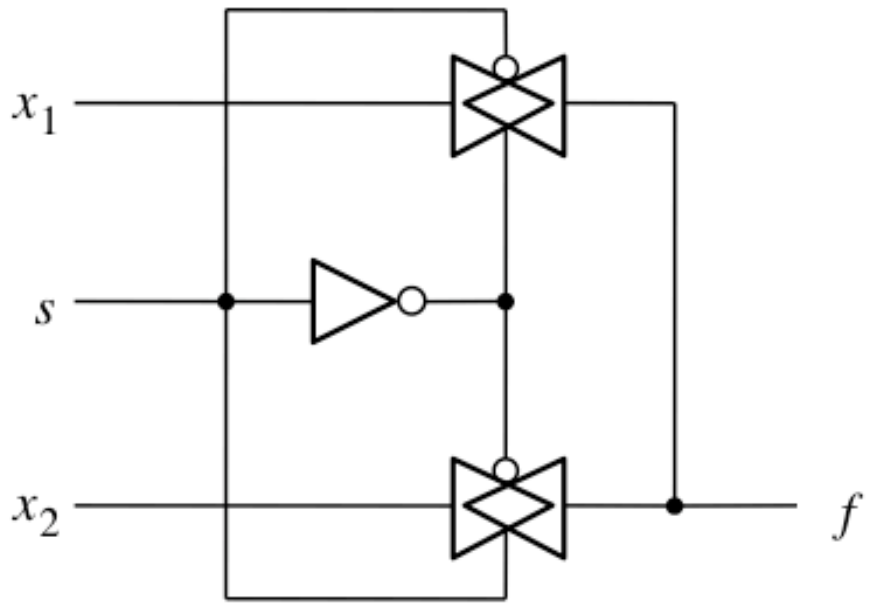


Figure 3.62. A 2-to-1 multiplexer built using transmission gates.

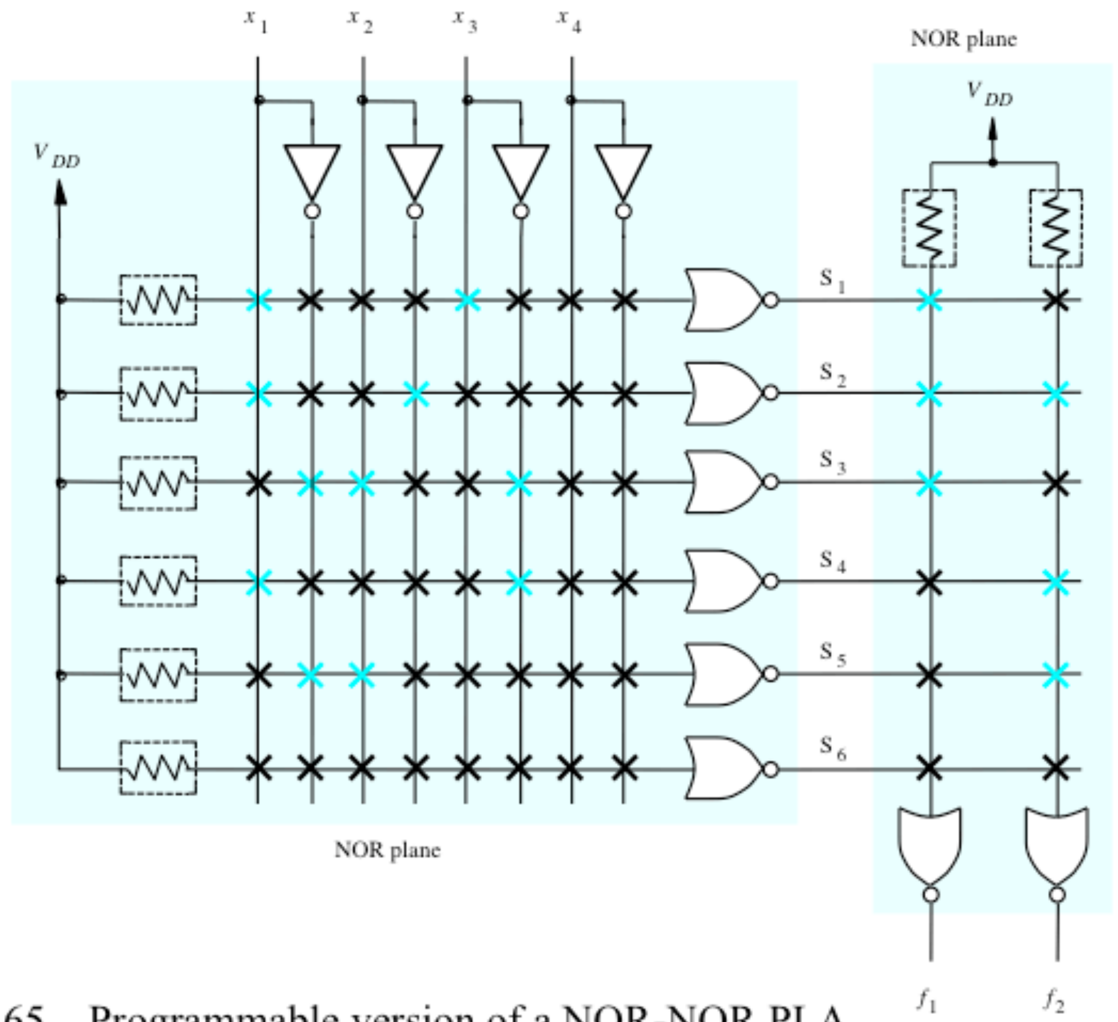


Figure 3.65. Programmable version of a NOR-NOR PLA.

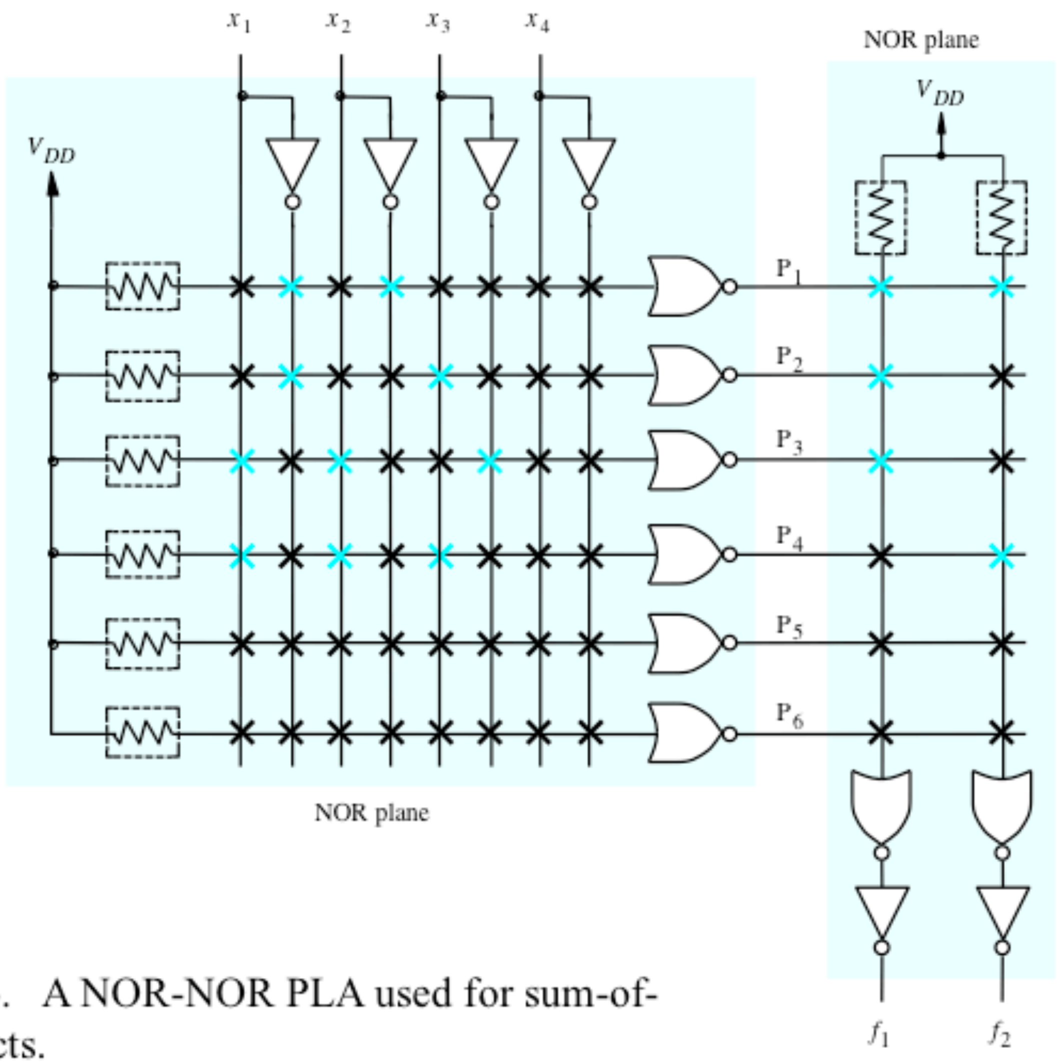


Figure 3.66. A NOR-NOR PLA used for sum-of-products.

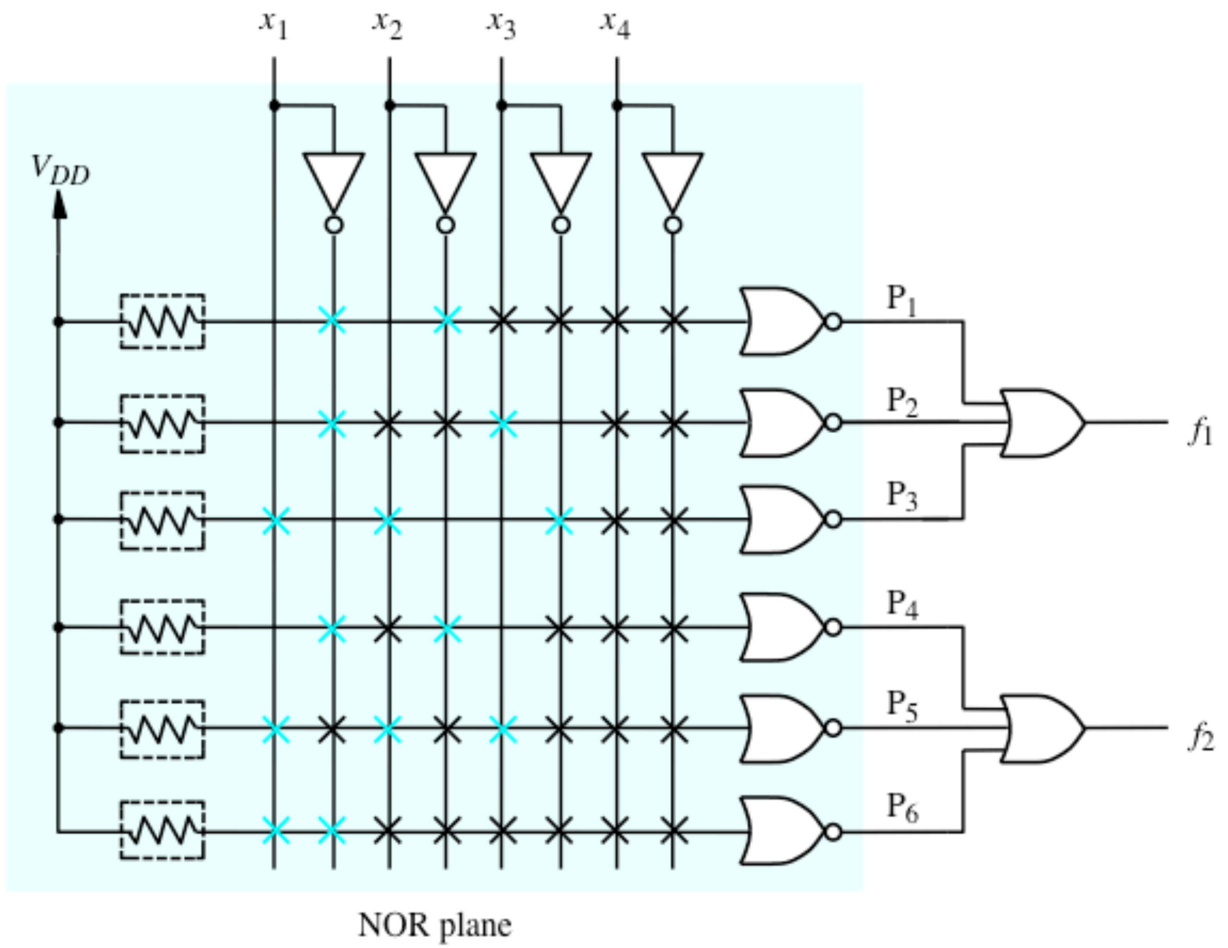


Figure 3.67. PAL programmed to implement two functions in Figure 3.66.