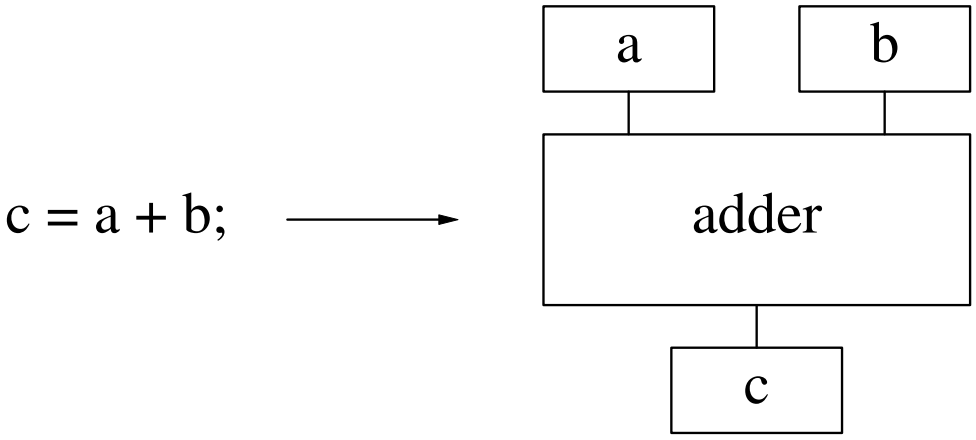
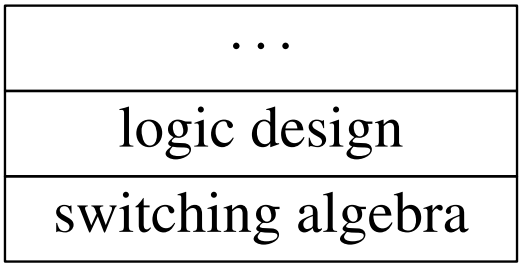
Chapter 1. Introduction

Digital design is about building the hardware part of a (computer) system.
We will cover the basic principles that a digital designer needs for designing hardware.





Physical quantities such as voltage and current assume values that are time-varying and can assume any value in a continuous range of real numbers.

Read:
1.1. About Digital Design
1.2. Analog versus Digital
1.3. Analog Signals

1.4. Digital Logic Signals

A digital signal is modeled as assuming, at any time, only one of two discrete values, called:

0	and	1
LOW	and	HIGH
FALSE	and	TRUE

The table shows the case of positive logic.

With a digital signal, the infinite set of real values of a physical quantity is mapped to the logic values 0 and 1.

Example: CMOS 2-Volt logic

0	1
0-0.5V	1.5-2.0V

Note:
The existence of an undefined area between 0 and 1.
The wide range of physical values that represent the same binary value.

These are important:
For defining 0 and 1 values reliably and unambiguously.
For making digital logic highly immune to component and power-supply variations as well as noise.

In addition, buffer circuits can be used to restore logic values.

For example, a buffer for 2-Volt CMOS logic converts any LOW input voltage into an output very close to 0V.

The same buffer converts any HIGH input voltage into an output very close to 2.0V.

One of the reasons digital circuits have replaced analog circuits in most applications is ease of design.

We will see that it is possible to specify digital designs using hardware description languages (HDL) that are similar to computer programming languages.

Software tools exist that can synthesize the HDL descriptions automatically (translate them into digital circuits).

A logic value is also called a binary digit, or bit.

A set of n bits represents 2^n values.

$n = 1$	$n = 2$		$n = 3$		
b_0	b_1	b_0	b_2	b_1	b_0
0	0	0	0	0	0
1	0	1	0	0	1
	1	0	0	1	0
	1	1	0	1	1
			1	0	0
			1	0	1
			1	1	0
			1	1	1

1.5. Logic Circuits and Gates

At the highest level of abstraction, a logic circuit can be represented as a "black box" with *n* inputs and *m* outputs.

To describe its behavior we only need to list 0 and 1 values for the inputs and outputs.

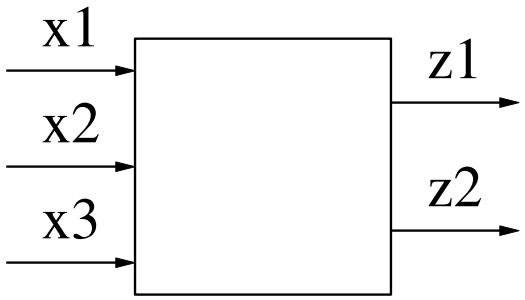
We will start with this description, then look at the basic building blocks of a digital circuit (basic gates), and then into the building blocks of the gates (transistors).

The circuit is *combinational* if the output values depend only on the current input values.

The behavior of a combinational circuit can be described by a table that specifies the output values for every combination of input values.

The table is called a *truth table*.

Example (Full Adder):



x_1	x_2	x_3	z_2	z_1
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A *sequential* circuit has memory.

Its output values depend on the sequence of past and current input values.

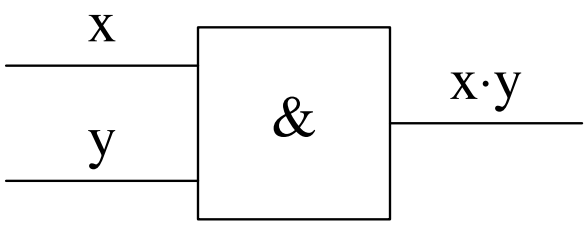
Such a circuit is described by a *state table* that specifies the output values and next state as a function of the current state and input values.

The basic digital device is called a gate.
A gate implements a basic logic function.

Three basic logic functions, AND, OR and NOT, can be used to build any combinational digital logic circuit.

In fact, AND and NOT are sufficient.
OR and NOT are also sufficient.

AND:
Logic symbol

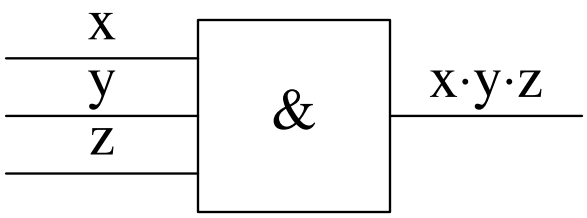


Truth table

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

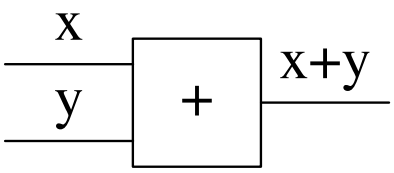
Definition
An AND gate (with any number of inputs) produces a 1 output if and only if all of its inputs are 1.

A 3-input AND gate:



x	y	z	x·y·z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

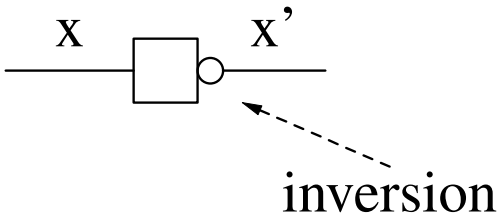
OR:



x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

An OR gate (with any number of inputs) produces a 1 output if and only if at least one of its inputs is 1.

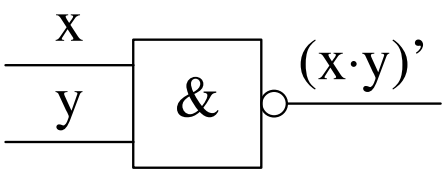
NOT (also called an inverter):



x	x'
0	1
1	0

Two additional gates can be obtained by combining AND and NOT, or OR and NOT.

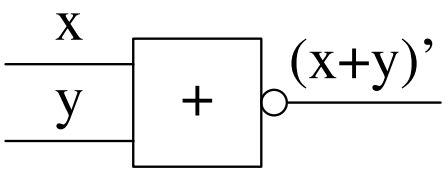
NAND:



x	y	(x·y)’
0	0	1
0	1	1
1	0	1
1	1	0

A NAND gate (with any number of inputs) produces a 0 output if and only if all of its inputs are 1.

NOR:



x	y	(x+y)'
0	0	1
0	1	0
1	0	0
1	1	0

A NOR gate (with any number of inputs) produces a 0 output if and only if at least one of its inputs is 1.

It is possible to connect gates to form more complex circuits that implement more complex functions.

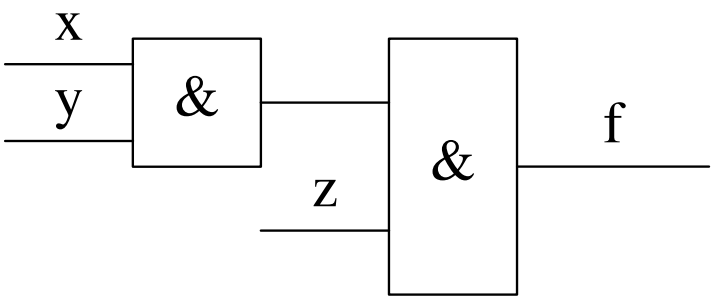
By convention, signals flow from left to right.

If the output of gate1 drives the input of gate2, gate1 is drawn to the left of gate2.

The inputs of the circuit are drawn on the left.

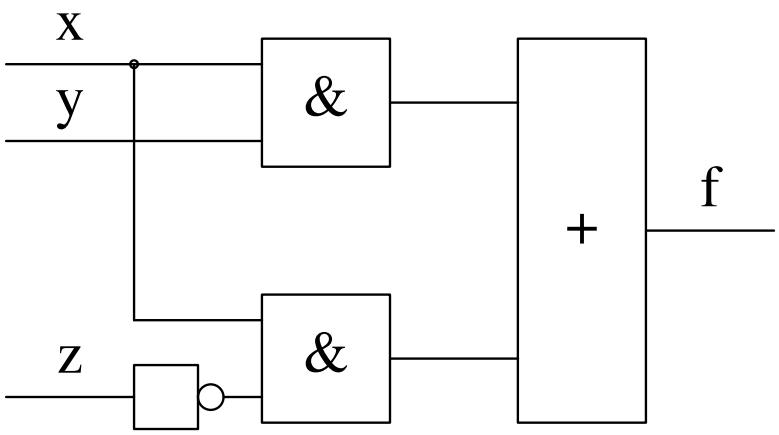
The outputs of the circuit are drawn on the right.

Example 1: A three-input AND gate



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

Example 2:



x	y	z	g_1	g_2	f
0	0	0			0
0	0	1			0
0	1	0			0
0	1	1			0
1	0	0			1
1	0	1			0
1	1	0			1
1	1	1			1

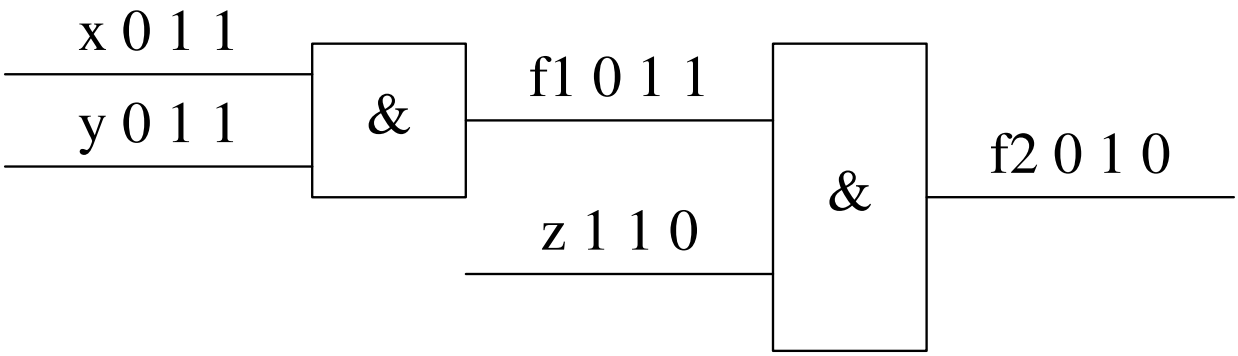
When the value of an input to a gate or circuit changes, it takes time before the gate or circuit outputs change.

This is called *propagation delay*.

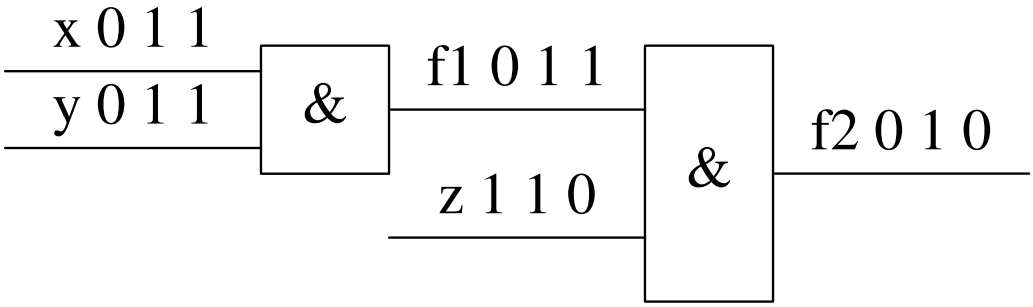
It can be represented by a *timing diagram*.

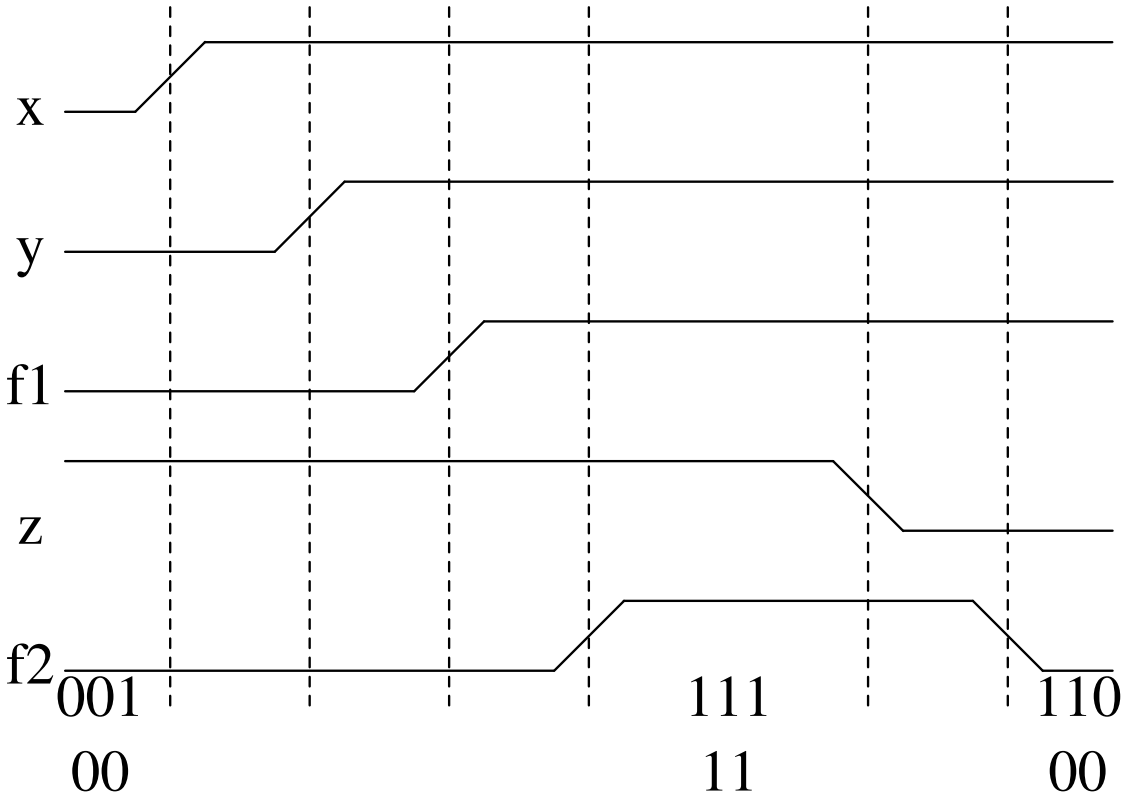
For most of the discussion of digital circuits it is possible to ignore propagation delays. The circuit behavior is well-defined regardless of delay.

Example:



Example:





Read:
1.6. Software Aspects of Digital Design
1.7. Integrated Circuits

1.8. Logic Families

There are many ways to implement logic circuits.

A logic family is a collection of different devices or chips that have similar input, output and internal circuit characteristics, but that perform different logic functions.

Chips from the same family can be interconnected to form circuits.
Chips from different families are typically not compatible.

TTL (Transistor-Transistor Logic) is a logic family based on bipolar junction transistors.

CMOS (Complementary Metal-Oxide Semiconductor) is based on MOSFETs (MOS Field-Effect Transistors).
CMOS is the most commonly used logic family.

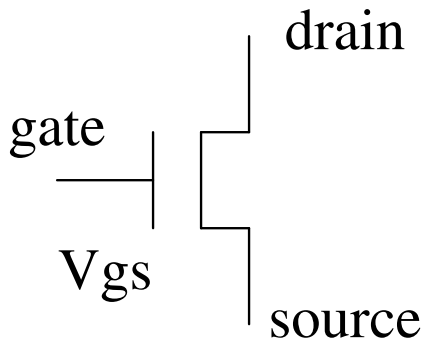
1.9. CMOS Logic Circuits

A MOS transistor can be modeled as a three-terminal voltage-controlled resistance.

The input voltage controls the resistance between the other two terminals.

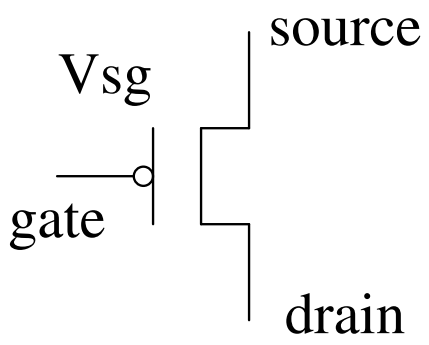
In digital applications, the resistance is either:
very high (the transistor is off), or
very low (the transistor is on).

An n-channel (NMOS) transistor:



When $V_{gs} = 0$ the resistance from drain to source is very high.
As V_{gs} is increased the resistance is decreased to a very low value.

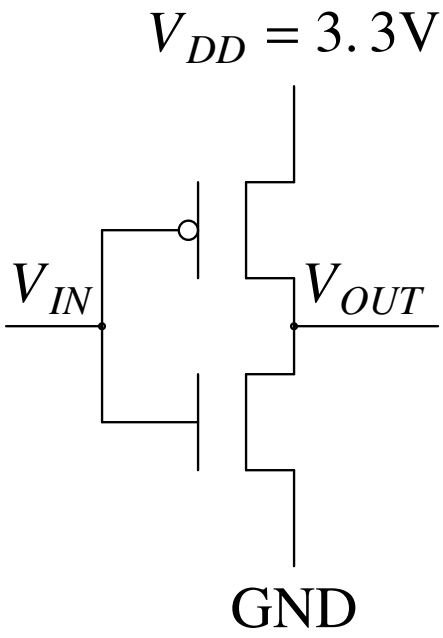
A p-channel (PMOS) transistor:



When $V_{sg} = 0$ the resistance from drain to source is very high.
As V_{sg} is increased the resistance is decreased to a very low value.

NMOS and PMOS transistors are used together in a complementary way to form *CMOS logic*.

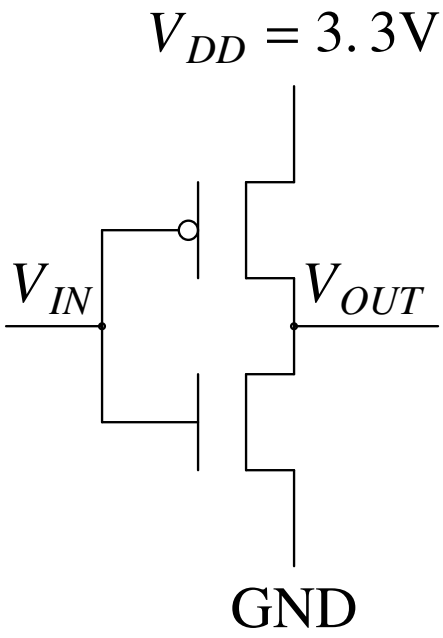
A CMOS inverter:



If $V_{IN} = 0$:
The NMOS transistor with $V_{gs} = 0$ is off.
The PMOS transistor with $V_{sg} = 3.3\text{V}$ is on.
With a very low resistance between V_{DD} and V_{OUT} through the PMOS transistor, we have $V_{OUT} \approx 3.3\text{V}$.

x	x'
0	1
1	0

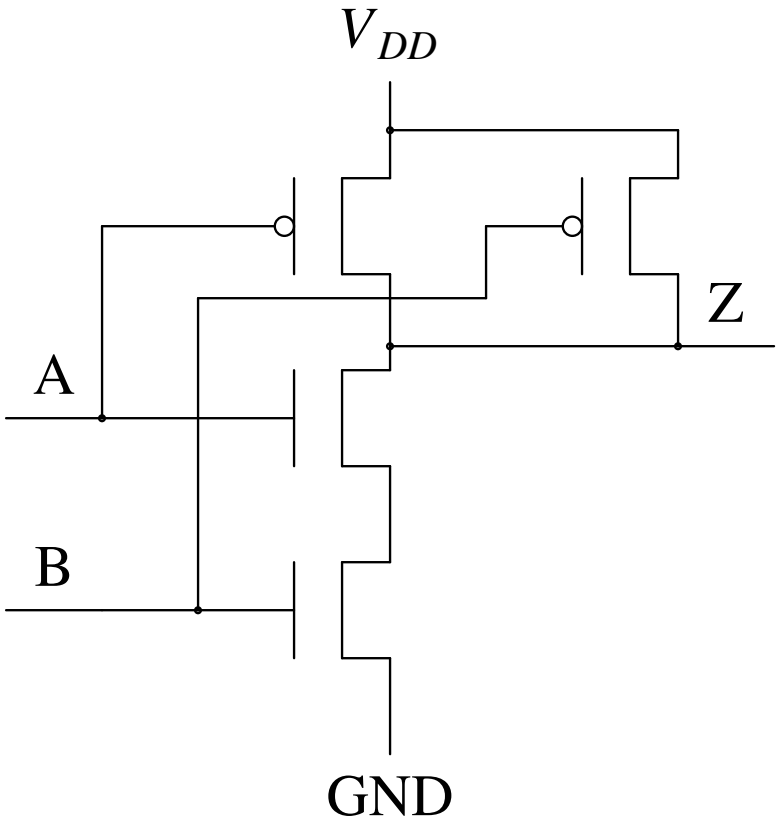
checked



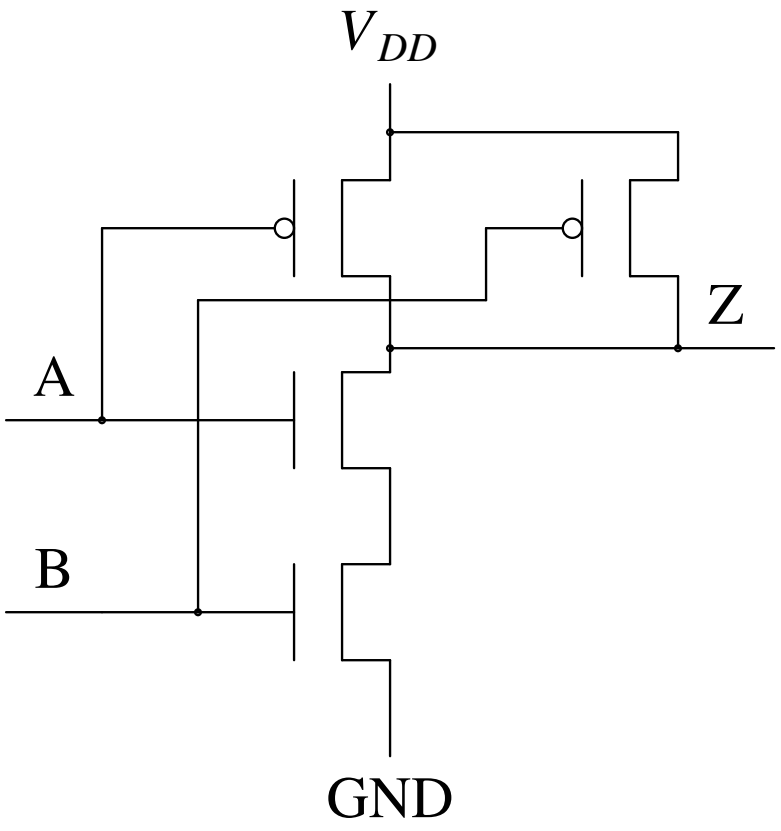
If $V_{IN} = 3.3\text{V}$:
The PMOS transistor with $V_{sg} = 0$ is off.
The NMOS transistor with $V_{gs} = 3.3\text{V}$ is on.
With a very low resistance between V_{OUT} and ground through the NMOS transistor, we have $V_{OUT} \approx 0$.

x	x’	
0	1	checked
1	0	checked

A CMOS NAND gate:



If A = LOW or B = LOW (or both):
One or both of the NMOS transistors are off.
One or both of the PMOS transistors are on, creating a low resistance between V_{DD} and Z.
The result is Z = HIGH.



If A = HIGH and B = HIGH:
Both PMOS transistors are off.
Both NMOS transistors are on, creating a low resistance between Z and ground.
The result is Z = LOW.

x	y	(x·y)’
0	0	1
0	1	1
1	0	1
1	1	0

In a CMOS NOR gate, two NMOS transistors are connected in parallel, and two PMOS transistors are connected in series.

For *n*-input gates, connect *n* NMOS and *n* PMOS transistors similar to the connections above.

The number of inputs to a gate is called its fan-in.

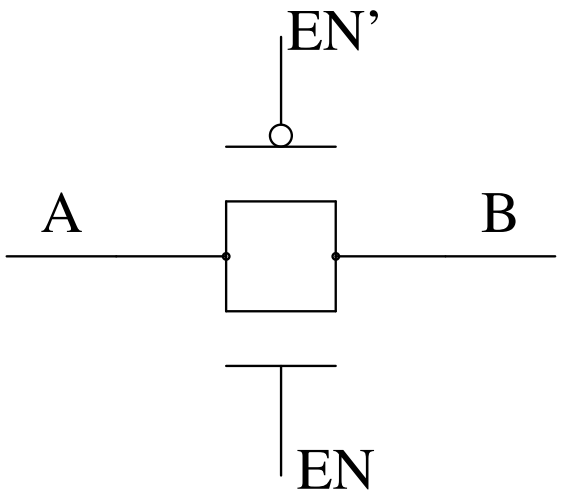
As the fan-in is increased, the resistances of the series transistors increase, and the delay of the gate increases as a result.

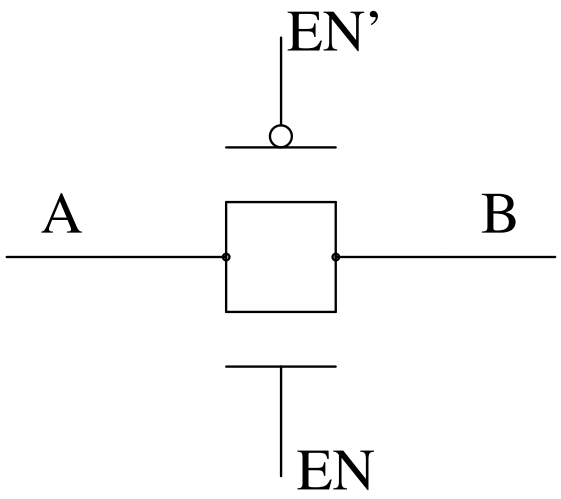
Instead of using gates with large fan-ins, it is possible to connect several gates with lower fan-ins to implement the same function.

NOT, NAND and NOR gates have the simplest implementations.
To obtain an AND or OR gate, connect the output of a NAND or NOR gate to an inverter.

Transmission Gates

A transmission gate acts as a logic-controlled switch.
It passes or blocks a CMOS input signal.
It consists of a PMOS and an NMOS transistor that are connected together with complementing control signals.



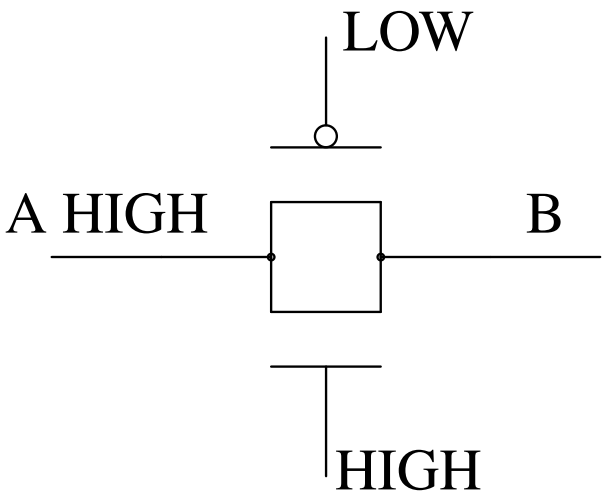


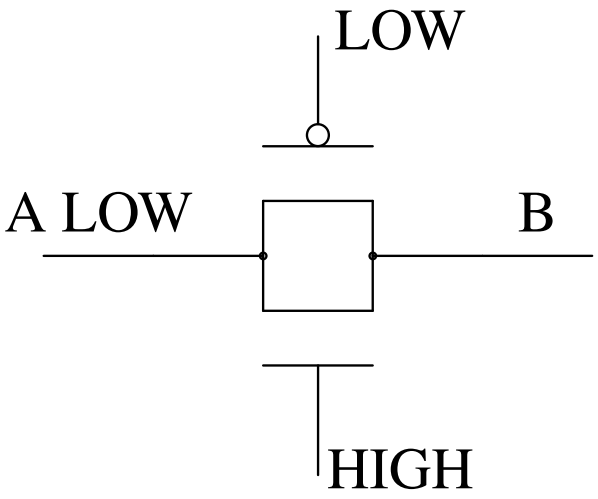
When EN is LOW, both transistors are off, and A and B are disconnected.

When EN is HIGH, both transistors can be on, and A and B are connected.

Hence a switch.

When EN is HIGH, assuming that A is stronger:





Two transistors are needed since:

A typical NMOS transistor cannot transmit a HIGH voltage between A and B very well (first case).

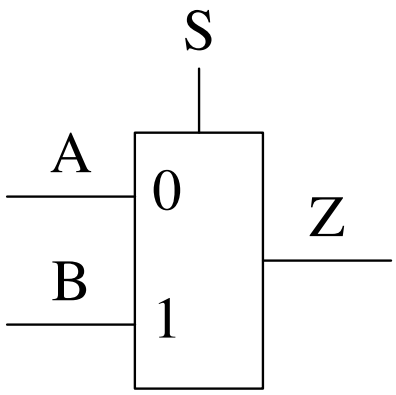
A typical PMOS transistor cannot transmit a LOW voltage between A and B very well (second case).

Together, the two transistors transmit the complete range of voltages well.

Read:

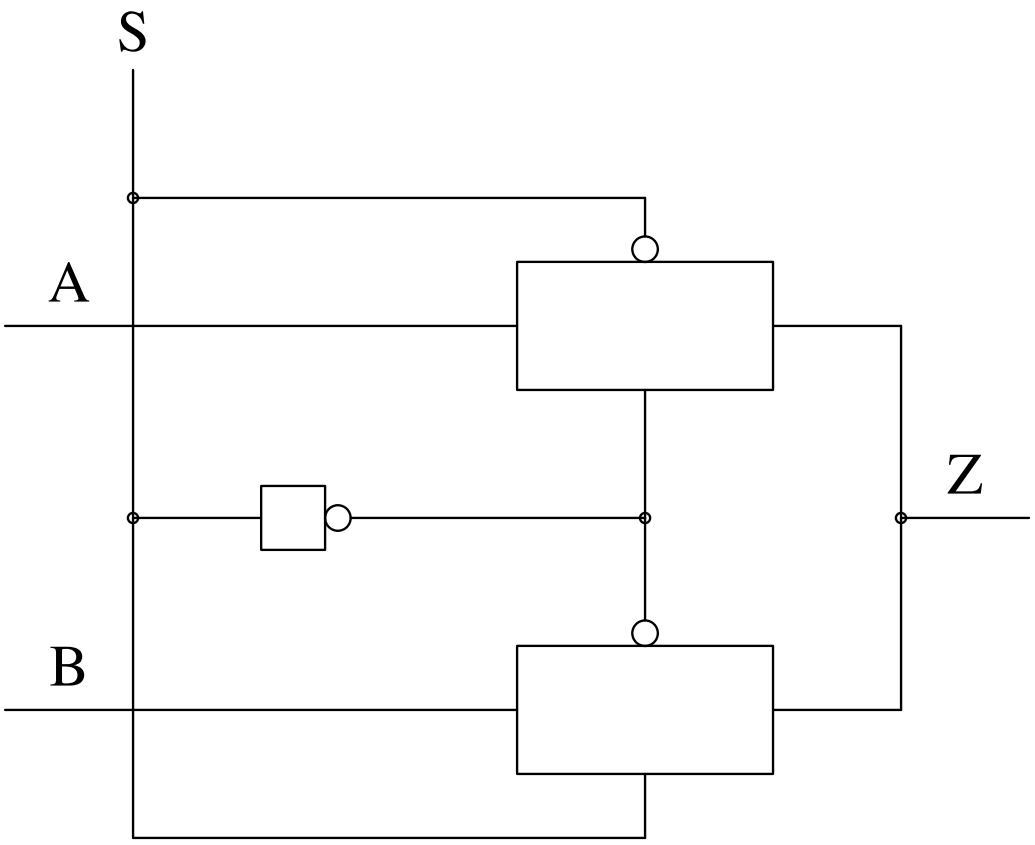
1.10. Programmable Devices
1.11. Application-Specific ICs
1.12. Printed Circuit Boards
1.13. Digital-Design Levels
1.14. Design Objectives

Design Levels Illustrated for a Multiplexers



When $S = 0$, $Z = A$.
When $S = 1$, $Z = B$.

Implementation using transmission gates:



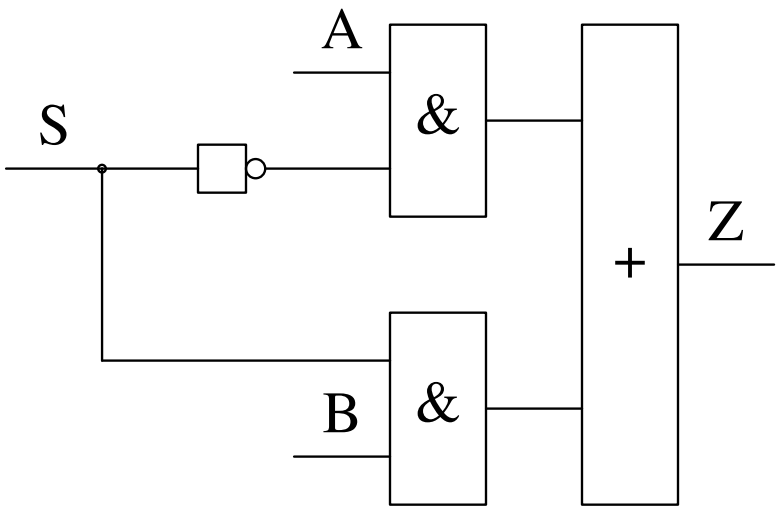
Number of transistors: 6.

Truth table:

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

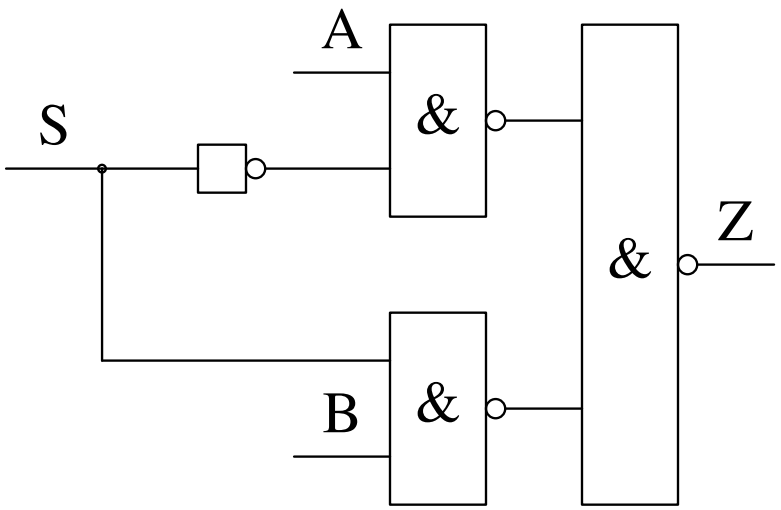
$$Z = S' \cdot A + S \cdot B$$

Implementation using gates:



Number of transistors:
Three gates require 12 transistors.
Four inverters require 8 transistors (three of the inverters are needed for the AND and OR gates).
Total 20 transistors.

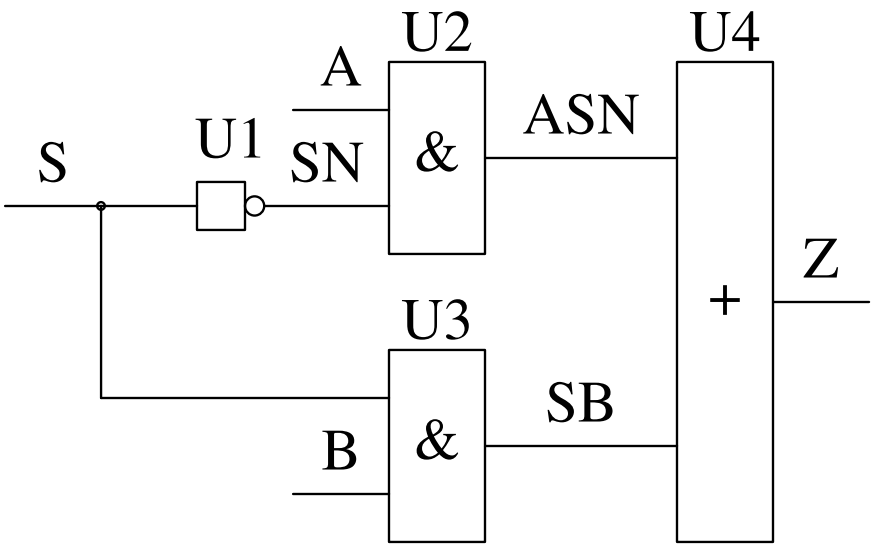
A different implementation using gates:



Number of transistors:
Three gates require 12 transistors.
An inverter requires 2 transitors.
Total 14 transistors.

Verilog structural model:

```
module Ch1mux_s(A,B,S,Z)
    input A,B,S;
    output Z;
    wire SN,ASN,SB;
    not U1(SN,S);
    and U2(ASN,A,SN);
    and U3(SB,B,S);
    or U4(Z,ASN,SB);
endmodule
```



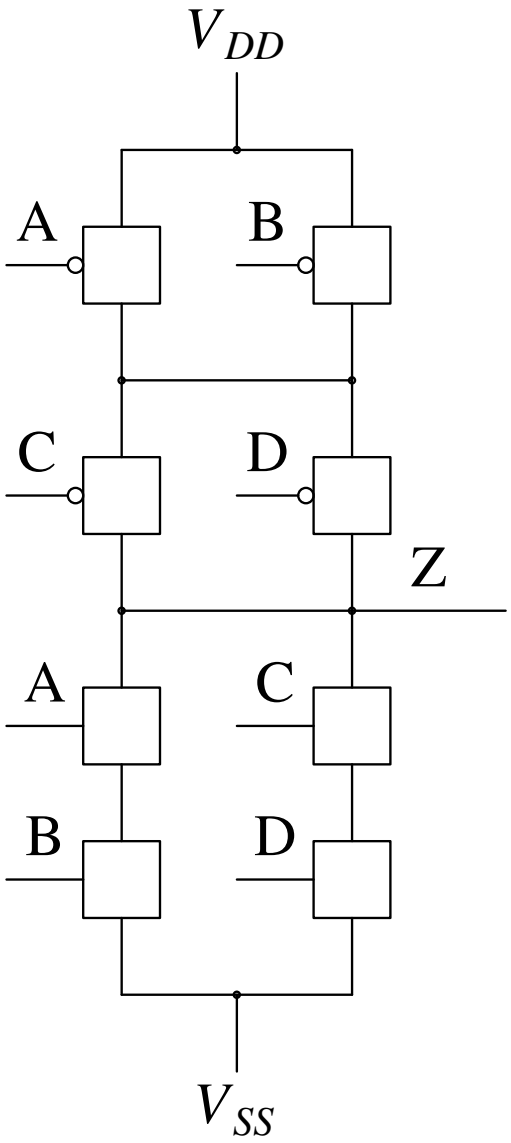
Verilog behavioral model:

```
module Ch1mux_b(A,B,S,Z)
    input A,B,S;
    output reg Z;
    always @(A,B,S)
        if (S==1) Z=B;
        else Z=A;
endmodule
```

The always statement indicates that the inputs A, B and S need to be monitored continously.
If any of them changes, the output Z needs to be updated so it is always equal to B if S is 1, and A otherwise.

Complex CMOS Gates

Example:



From the NMOS transistors:
$Z = 0$ when $A \cdot B + C \cdot D$

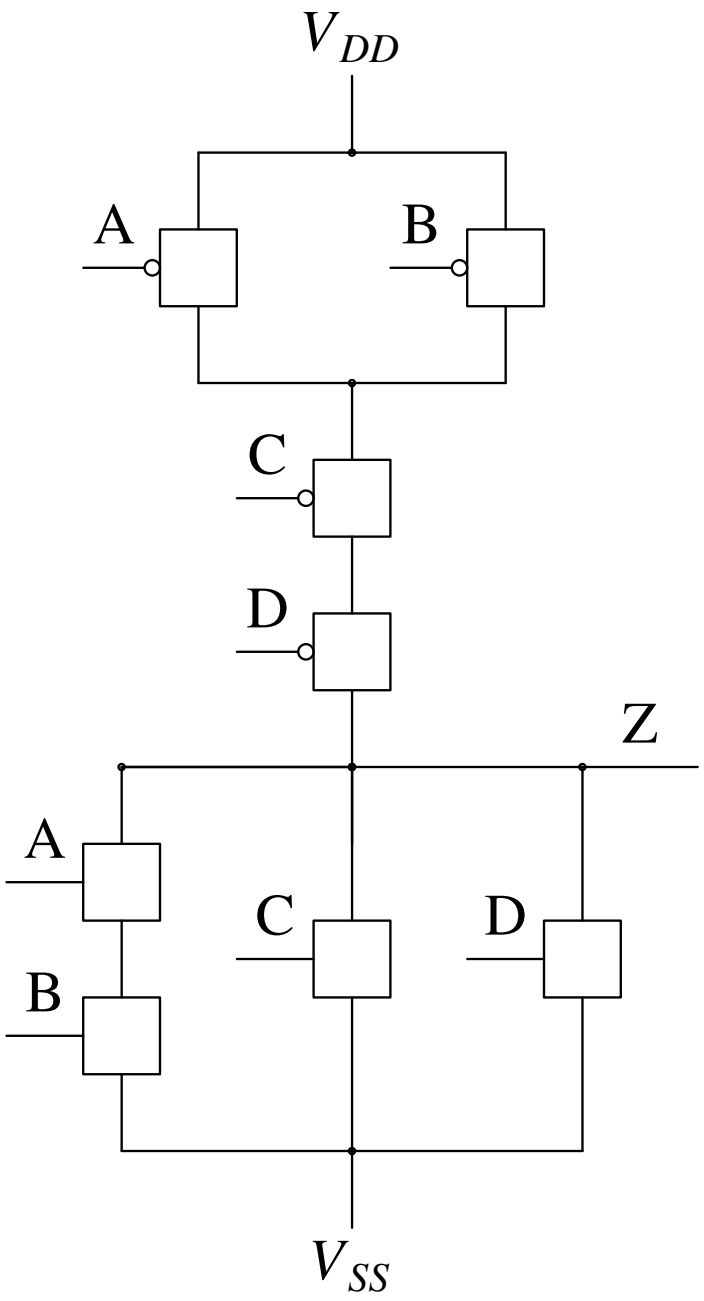
From the PMOS transistors:
$Z = 1$ when $(A' + B') \cdot (C' + D')$

Later on we will see that
$(A' + B') \cdot (C' + D') = (A \cdot B + C \cdot D)'$

$Z = (A \cdot B + C \cdot D)'$

AND-OR-INVERT (AOI) gate.

Example:



From the NMOS transistors:
$Z = 0$ when $A \cdot B + C + D$

From the PMOS transistors:
$Z = 1$ when $(A' + B') \cdot C' \cdot D'$

Later on we will see that
$(A' + B') \cdot C' \cdot D' = (A \cdot B + C + D)'$

$Z = (A \cdot B + C + D)'$