SCHOOL OF INNOVATIVE TECHNOLOGIES & ENGINEERING

Module Information Pack

BEng (Hons.) Electronic Engineering
BEng (Hons.) Telecommunications Engineering

Module name: Digital Electronics

Module code: ELEC2102C

Academic Year 2022/2023 – Semester 1

Blended Mode (Online and F2F)

Programme Coordinator (BTEL):	Dr. Vinaye Armoogum
Programme Coordinator (BEE):	Mr. Dudley Tse Kai Wai
Module Coordinator:	Mr. Rishi H. Heerasing
Module Convenor:	Mr. Rishi H. Heerasing
1. **Office:**	Room G2.14 Level 2 SITE BLOCK
2. **Phone:**	207 5250 ext. 124
3. **E-mail:**	rheerasing@utm.ac.mu
Academic Tutoring	None
Classes Day and Time:	Wednesdays 13:00 – 16:00 DEL
Credits and Level:	6 Credits, Level 2
Pre-requisite (if applicable):	Analogue Electronics
Co-requisite (if applicable):	None
Method of Delivery & Frequency:	15 x 3 Hours blended classes
Method & Criteria of Assessment:	50% Coursework and 50% Exams

Summary of Module Content:

Topics include encoding information in various digital formats, manipulation of real data using arithmetic, logic and relational operations, basic logic components, design, analysis and implementation of combinatorial logic circuits, principles of sequential logic and developing an understanding of memory elements and simple sequential circuits.

Module Aims:

- To introduce number systems and codes
- To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
- To introduce the methods for simplifying Boolean expressions
- To learn the basic methods for the design of digital circuits and provide the fundamental concepts used in the design of digital systems.
- To outline the formal procedures for the analysis and design of combinatorial circuits and sequential circuits

Learning Outcomes:

By the end of this module, students are expected:

- To understand the main building blocks of digital circuits and the practical aspects of digital electronics.
- Manipulate number systems used in digital system.
- Use basic gates and MSI circuits in digital system design.
- Analyze counter and register circuits.
- Design synchronous counters.
- Understand the operation of memory circuits.

Tentative Module Schedule

NOTE: All classes will be in blended mode until further notice.

Wk	Dates	Topics Covered
1	19/10/22 *F2F*	Digital Concepts, Number Systems, Operations, and Codes
2	26/10/22 *Online*	Logic Gates and Boolean Algebra
13	***02/11/22*** *F2F*	***Public Holiday***
4	09/11/22 *Online*	Digital Logic Families
5	16/11/22 *F2F*	Combinatorial Logic Analysis
6	23/11/22 *Online*	Functions of Combinational Logic
7	30/11/22 *F2F*	Functions of Combinational Logic
8	07/12/22 *Online*	Latches, Flip-Flops, and Timers
9	14/12/22 *F2F*	Latches, Flip-Flops, and Timers
10	11/01/23 *Online*	Counters
11	18/01/23 *F2F*	Counters
12	25/01/23 Online	Shift Registers, Memory and Storage
13	***01/02/23*** *F2F*	***Public Holiday***
14	*08/02/23* *F2F*	***Class Test (40% of coursework weight)***
15	15/02/23 *Online*	*Buffer*

Reading List

Recommend Texts (as per availability in the UTM Resource Centre)**:**

- **Floyd T. L. (2005) *Digital Fundamentals: 9th Ed.*, Prentice-Hall ***
- **Tokheim R. (2007) *Digital Electronics – Principles & Applications: 7th Ed.*, Mc-Graw-Hill. (D6.11 TOK)**
- **Predko M. (2005) *Digital Electronics Demystified*, Mc-Graw Hill ***

ж*You can download a copy of these books in e-book format on my site at* http://www.rishiheerasing.net/modules/elec2102/readings.html

Other Reading Materials e.g. Articles/Papers/Websites:

- Prentice-Hall Companion Website for *Digital Fundamentals*, 10*th Edition* at http://wps.prenhall.com/chet_floyd_digitalfun_10/

Past Exam Papers

○ Given at the end of MIP and also from the Resource Centre at http://www.utm.ac.mu/Resource/

Module Assets

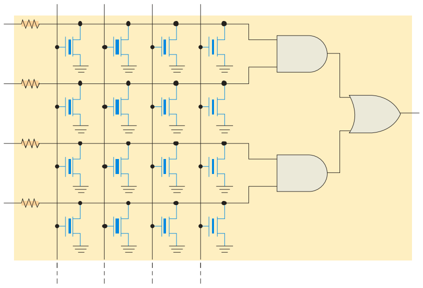
The assets are available on **Nefertum's Shrine** at http://www.rishiheerasing.net/modules/elec2102/ln.html

The lectures notes are in .pdf format so you will need Adobe Acrobat® Reader to view them. This reader can also be downloaded from the above-mentioned site.

Software needed will be National Instruments MultiSim Electronic Workbench Version 10.0 and is available on my site at http://www.rishiheerasing.net/modules/elec2102/tools.html

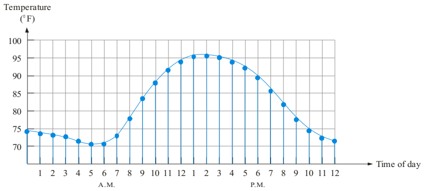
Digital Electronics

Digital Concepts



Analogue Quantities

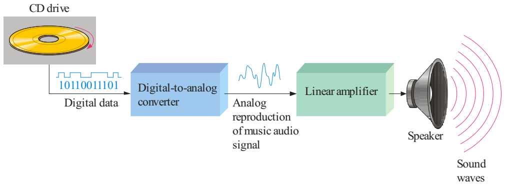
Most natural quantities that we see are **analogue** and vary continuously. Analogue systems can generally handle higher power than digital systems.



Digital systems can process, store, and transmit data more efficiently but can only assign discrete values to each point.

Analogue and Digital Systems

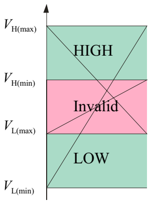
Many systems use a mix of analogue and digital electronics to take advantage of each technology. A typical CD player accepts digital data from the CD drive and converts it to an analogue signal for amplification.


CD drive
10110011101
Digital data
Digital-to-analog converter
Analog reproduction of music audio signal
Linear amplifier
Speaker
Sound waves

Binary Digits and Logic Levels

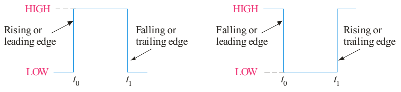
Digital electronics uses circuits that have two states, which are represented by two different voltage levels called HIGH and LOW. The voltages represent numbers in the binary system.

In binary, a single number is called a *bit* (for *b*inary dig*it*). A bit can have the value of either a 0 or a 1, depending on if the voltage is HIGH or LOW.



Digital Waveforms

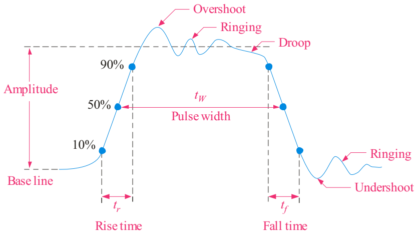
Digital waveforms change between the LOW and HIGH levels. A positive going pulse is one that goes from a normally LOW logic level to a HIGH level and then back again. Digital waveforms are made up of a series of pulses.



(a) Positive–going pulse (b) Negative–going pulse

Pulse Definitions

Actual pulses are not ideal but are described by the rise time, fall time, amplitude, and other characteristics.



Periodic Pulse Waveforms

Periodic pulse waveforms are composed of pulses that repeats in a fixed interval called the **period**. The **frequency** is the rate it repeats and is measured in hertz.

$$f = \frac{1}{T} \qquad T = \frac{1}{f}$$

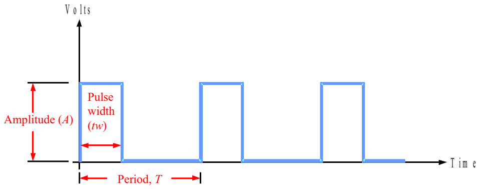
The **clock** is a basic timing signal that is an example of a periodic wave.

What is the period of a repetitive wave if $f = 3.2$ GHz?

$$T = \frac{1}{f} = \frac{1}{3.2\text{ GHz}} = 313\text{ ps}$$

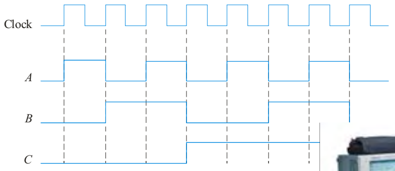
Pulse Definitions

In addition to frequency and period, repetitive pulse waveforms are described by the amplitude (A), pulse width (t_W) and duty cycle. Duty cycle is the ratio of t_W to T.



Timing Diagrams

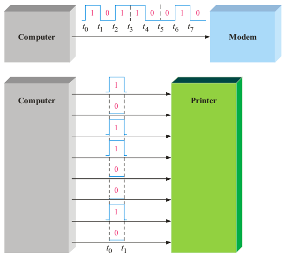
A timing diagram is used to show the relationship between two or more digital waveforms,



A diagram like this can be observed directly on a logic analyzer.

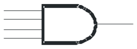
Serial and Parallel Data

Data can be transmitted by either serial transfer or parallel transfer.

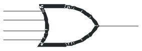


Basic Logic Functions

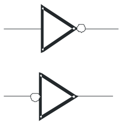
AND True only if ***all*** input conditions are true.



OR True only if ***one or more*** input conditions are true.



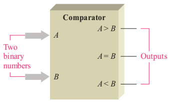
NOT Indicates the ***opposite*** condition.



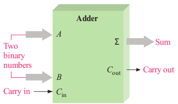
Basic System Functions

And, **or**, and **not** elements can be combined to form various logic functions. A few examples are:

The comparison function

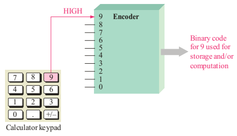


Basic arithmetic functions

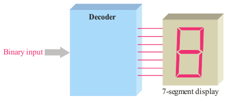


Basic System Functions

The encoding function

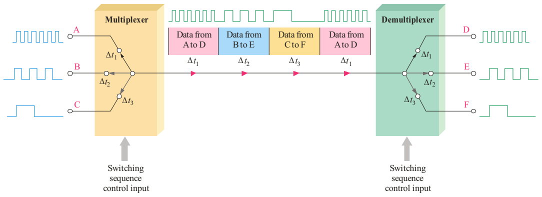


The decoding function



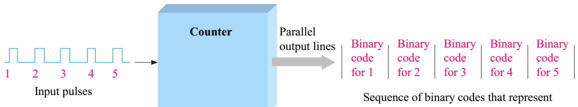
Basic System Functions

The data selection function



Basic System Functions

The counting function

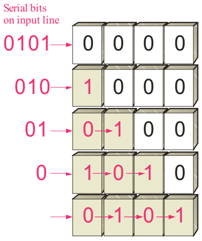


Sequence of binary codes that represent the number of input pulses counted.

…and other functions such as code conversion and storage.

Basic System Functions

One type of storage function is the shift register, that moves and stores data each time it is clocked.



Initially the register contains only invalid data or all zeros as shown here.

First bit (1) is shifted serially into the register.

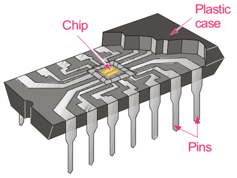
Second bit (0) is shifted serially into register and first bit is shifted right.

Third bit (1) is shifted into register and the first and second bits are shifted right.

Fourth bit (0) is shifted into register and the first, second, and third bits are shifted right. The register now stores all four bits and is full.

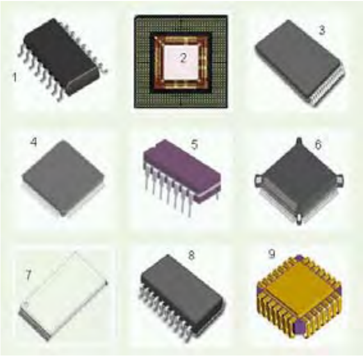
Integrated Circuits

Cutaway view of DIP (Dual-In-line Pins) chip:



The TTL series, available as DIPs are popular for laboratory experiments with logic.

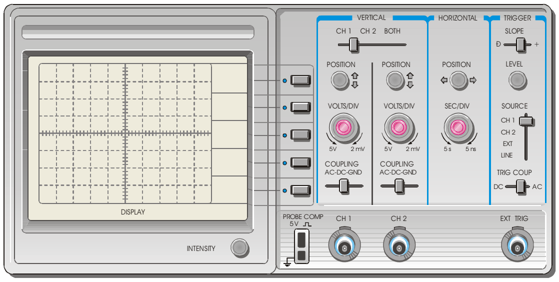
Example of Surface Mount Packaging for Integrated Circuits



1. Small Outline IC (SOIC)
2. Ball Grid Array (BGA)
3. Thin Small Outline Package(TSOP)
4. Thin Quad Flat Pack (TQFP)
5. Dual In-Line Package (DIP)
6. Quad Flat Pack (QFP)
7. Small Outline Package (SOP)
8. Shrink Small Outline Package (SSOP)
9. Ceramic Leadless Chip Carrier (CLCC)

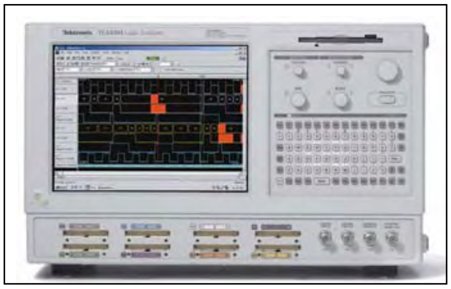
Test and Measurement Instruments

The front panel controls for a general-purpose oscilloscope can be divided into four major groups.



Test and Measurement Instruments

The logic analyzer can display multiple channels of digital information or show data in tabular form.



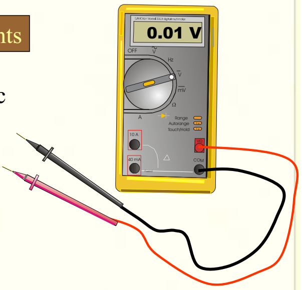
Test and Measurement Instruments

The DMM can make three basic electrical measurements.

Voltage

Resistance

Current

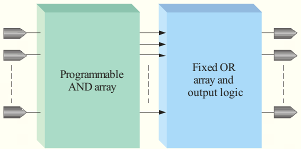


In digital work, DMMs are useful for checking power supply voltages, verifying resistors, testing continuity, and occasionally making other measurements.

Programmable Logic

Programmable logic devices (PLDs) are an alternative to fixed function devices. The logic can be programmed for a specific purpose. In general, they cost less and use less board space that fixed function devices.

A PAL device is a form of PLD that uses a combination of a programmable AND array and a fixed OR array:



Selected Key Terms

Analogue Being continuous or having continuous values.

Digital Related to digits or discrete quantities; having a set of discrete values.

Binary Having two values or states; describes a number system that has a base of two and utilizes 1 and 0 as its digits.

Bit A binary digit, which can be a 1 or a 0.

Pulse A sudden change from one level to another, followed after a time, called the pulse width, by a sudden change back to the original level.

Clock A basic timing signal in a digital system; a periodic waveform used to synchronize actions.

Selected Key Terms

Gate A logic circuit that performs a basic logic operations such as AND or OR.

NOT A basic logic function that performs inversion.

AND A basic logic operation in which a true (HIGH) output occurs only when all input conditions are true (HIGH).

OR A basic logic operation in which a true (HIGH) output occurs when when one or more of the input conditions are true (HIGH).

Fixed-function logic A category of digital integrated circuits having functions that cannot be altered.

Programmable logic A category of digital integrated circuits capable of being programmed to perform specified functions.

1. Compared to analogue systems, digital systems

a. are less prone to noise

b. can represent an infinite number of values

c. can handle much higher power

d. all of the above

2. The number of values that can be assigned to a bit are

a. one

b. two

c. three

d. ten

3. The time measurement between the 50% point on the leading edge to the 50% point on the trailing edge of the pulse is the

a. rise time

b. fall time

c. period

d. pulse width

4. The time measurement between the 90% point on the trailing edge to the 10% point on the trailing edge of the pulse is the

a. rise time

b. fall time

c. period

d. pulse width

5. The reciprocal of the frequency of a clock signal is the

a. rise time

b. fall time

c. period

d. pulse width

6. If the period of a clock signal is 500 ps, the frequency is

a. 20 MHz

b. 200 MHz

c. 2 GHz

d. 20 GHz

7. AND, OR, and NOT gates can be used to form

a. storage devices

b. comparators

c. data selectors

d. all of the above

8. A shift register is an example of a

a. storage device

b. comparator

c. data selector

d. counter

9. A device that is used to switch one of several input lines to a single output line is called a

a. comparator

b. decoder

c. counter

d. multiplexer

10. For most digital work, an oscilloscope should be coupled to the signal using

a. ac coupling

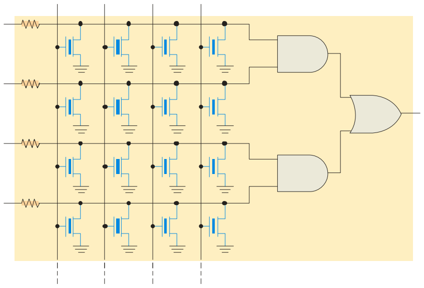
b. dc coupling

c. GND coupling

d. none of the above

Digital Electronics

Number Systems, Operations & Codes



Decimal Numbers

The position of each digit in a weighted number system is assigned a weight based on the **base** or **radix** of the system. The radix of decimal numbers is ten, because only ten symbols (0 through 9) are used to represent any number.

The column weights of decimal numbers are powers of ten that increase from right to left beginning with $10^0 = 1$:

For fractional decimal numbers, the column weights are negative powers of ten that decrease from left to right:

$10^2\ 10^1\ 10^0.\ 10_{-1}\ 10_{-2}\ 10_{-3}\ 10_{-4}\ \ldots$

Decimal Numbers

Decimal numbers can be expressed as the sum of the products of each digit times the column value for that digit. Thus, the number 9240 can be expressed as

$(9 \times 10^3) + (2 \times 10^2) + (4 \times 10^1) + (0 \times 10^0)$

or

$9 \times 1{,}000 + 2\ \ \times 100 + 4 \times 10 + 0 \times 1$

Express the number 480.52 as the sum of values of each digit.

$480.52 = (4 \times 10_2) + (8 \times 10_1) + (0 \times 10_0) + (5 \times 10_{-1}) + (2 \times 10_{-2})$

Binary Numbers

For digital systems, the binary number system is used. Binary has a radix of two and uses the digits 0 and 1 to represent quantities.

The column weights of binary numbers are powers of two that increase from right to left beginning with $2^0 = 1$:

For fractional binary numbers, the column weights are negative powers of two that decrease from left to right:

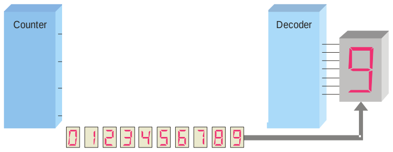
$2^2\ 2^1\ 2^0.\ 2_{-1}\ 2_{-2}\ 2_{-3}\ 2_{-4}\ \ldots$

Binary Numbers

A binary counting sequence for numbers from zero to fifteen is shown.

Notice the pattern of zeros and ones in each column.

Digital counters frequently have this same pattern of digits:



Decimal Number	Binary Number
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Binary Conversions

The decimal equivalent of a binary number can be determined by adding the column values of all of the bits that are 1 and discarding all of the bits that are 0.

Convert the binary number 100101.01 to decimal.

Start by writing the column weights; then add the weights that correspond to each 1 in the number.

$2_5\ \ 2_4\ \ 2_3\ \ 2_2\ \ 2_1\ 2_0.\ 2_{-1}\ 2_{-2}$
$32\ 16\ \ 8\ \ \ 4\ \ \ 2\ \ \ 1\ .\ \tfrac{1}{2}\ \ \tfrac{1}{4}$

$1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1.\ \ 0\ \ 1$
$32\ \qquad +4\ \ \ +1\ \qquad +\tfrac{1}{4} =\ \ 37\tfrac{1}{4}$

Binary Conversions

You can convert a decimal whole number to binary by reversing the procedure. Write the decimal weight of each column and place 1's in the columns that sum to the decimal number.

Convert the decimal number 49 to binary.

The column weights double in each position to the right. Write down column weights until the last number is larger than the one you want to convert.

2_6	2_5	2_4	2_3	2_2	2_1	2_0.
64	32	16	8	4	2	1.
0	**1**	**1**	**0**	**0**	**0**	**1.**

Binary Conversions

You can convert a decimal fraction to binary by repeatedly multiplying the fractional results of successive multiplications by 2. The carries form the binary number.

Convert the decimal fraction 0.188 to binary by repeatedly multiplying the fractional results by 2.

0.188 x 2 = 0.376 carry = 0 MSB
0.376 x 2 = 0.752 carry = 0
0.752 x 2 = 1.504 carry = 1
0.504 x 2 = 1.008 carry = 1
0.008 x 2 = 0.016 carry = 0

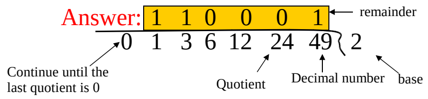
Answer = .00110 (for five significant digits)

Binary Conversions

You can convert decimal to any other base by repeatedly dividing by the base. For binary, repeatedly divide by 2:

Convert the decimal number 49 to binary by repeatedly dividing by 2.

You can do this by "reverse division" and the answer will read from left to right. Put quotients to the left and remainders on top.


Answer: 1 1 0 0 0 1

Binary Addition

The rules for binary addition are

0 + 0 = 0 Sum = 0, carry = 0
0 + 1 = 0 Sum = 1, carry = 0
1 + 0 = 0 Sum = 1, carry = 0
1 + 1 = 10 Sum = 0, carry = 1

When an input carry = 1 due to a previous result, the rules are

1 + 0 + 0 = 01 Sum = 1, carry = 0
1 + 0 + 1 = 10 Sum = 0, carry = 1
1 + 1 + 0 = 10 Sum = 0, carry = 1
1 + 1 + 1 = 10 Sum = 1, carry = 1

Binary Addition

Add the binary numbers 00111 and 10101 and show the equivalent decimal addition.

```
 0 1 1 1
00111      7
10101     21
-----    ---
11100  = 28
```

Binary Subtraction

The rules for binary subtraction are

0 – 0 = 0
1 – 1 = 0
1 – 0 = 1
10 – 1 = 1 with a borrow of 1

Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

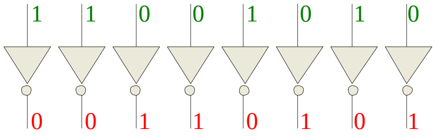
```
 /1/1/1
10101     21
00111      7
-----    ---
01110  = 14
```

1’s Complement

The 1’s complement of a binary number is just the inverse of the digits. To form the 1’s complement, change all 0’s to 1’s and all 1’s to 0’s.

For example, the 1’s complement of 11001010 is
00110101

In digital circuits, the 1’s complement is formed by using inverters:

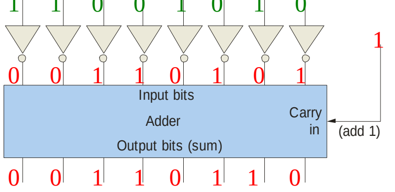


2’s Complement

The 2’s complement of a binary number is found by adding 1 to the LSB of the 1’s complement.

Recall that the 1’s complement of 11001010 is

```
                                   00110101 (1’s complement)
To form the 2’s complement, add 1:       +1
                                   --------
                                   00110110 (2’s complement)
```



Signed Binary Numbers

There are several ways to represent signed binary numbers. In all cases, the MSB in a signed number is the sign bit, that tells you if the number is positive or negative.

Computers use a modified 2’s complement for signed numbers. Positive numbers are stored in *true* form (with a 0 for the sign bit) and negative numbers are stored in *complement* form (with a 1 for the sign bit).

For example, the positive number 58 is written using 8-bits as 00111010 (true form).

Sign bit Magnitude bits

Signed Binary Numbers

Negative numbers are written as the 2’s complement of the corresponding positive number.

The negative number –58 is written as:

–58 = 11000110 (complement form)

Sign bit Magnitude bits

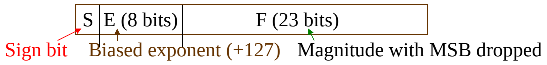
An easy way to read a signed number that uses this notation is to assign the sign bit a column weight of –128 (for an 8-bit number). Then add the column weights for the 1’s.

Assuming that the sign bit = –128, show that 11000110 = –58 as a 2’s complement signed number:

```
Column weights: –128 64 32 16  8  4  2  1.
                   1  1  0  0  0  1  1  0
                –128 +64           +4 +2    = –58
```

Floating Point Numbers

Floating point notation is capable of representing very large or small numbers by using a form of scientific notation. A 32-bit single precision number is illustrated.



Express the speed of light, *c*, in single precision floating point notation. (c = 0.2998 x 10^9)

In binary, c = 0001 0001 1101 1110 1001 0101 1100 0000_2.

In scientific notation, c = 1.001 1101 1110 1001 0101 1100 0000 x 228.

S = 0 because the number is positive. E = 28 + 127 = 15510 = 1001 10112. F is the next 23 bits after the first 1 is dropped.

In floating point notation, c = | 0 | 10011011 | 001 1101 1110 1001 0101 1100 |

Arithmetic Operations with Signed Numbers

Using the signed number notation with negative numbers in 2’s complement form simplifies addition and subtraction of signed numbers.

Rules for **addition**: Add the two signed numbers. Discard any final carries. The result is in signed form.
Examples:

```
 00011110 = +30     00001110 = +14      11111111 =  –1
 00001111 = +15     11101111 = –17      11111000 =  –8
 --------           --------           ---------
 00101101 = +45     11111101 =  –3     ~~1~~11110111 =  –9
```

Discard carry

Arithmetic Operations with Signed Numbers

Note that if the number of bits required for the answer is exceeded, overflow will occur. This occurs only if both numbers have the same sign. The overflow will be indicated by an incorrect sign bit.

Two examples are:

$$\begin{array}{l} 01000000 = +128 \\ 01000001 = +129 \\ \hline 10000001 = -\cancel{126} \end{array} \qquad \text{Discard carry} \rightarrow \begin{array}{r} 10000001 = -127 \\ 10000001 = -127 \\ \hline \cancel{1}00000010 = \cancel{+2} \end{array}$$

Wrong! The answer is incorrect and the sign bit has changed.

Arithmetic Operations with Signed Numbers

Rules for **subtraction**: 2’s complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form.

Repeat the examples done previously, but subtract:

$$\begin{array}{r} 00011110 \;\; (+30) \\ -\;00001111 \;\; -(+15) \\ \hline \end{array} \qquad \begin{array}{r} 00001110 \;\; (+14) \\ -\;11101111 \;\; -(-17) \\ \hline \end{array} \qquad \begin{array}{r} 11111111 \;\; (-1) \\ -\;11111000 \;\; -(-8) \\ \hline \end{array}$$

2’s complement subtrahend and add:

$$\begin{array}{r} 00011110 = +30 \\ 11110001 = -15 \\ \hline \cancel{1}00001111 = +15 \end{array} \qquad \begin{array}{r} 00001110 = +14 \\ 00010001 = +17 \\ \hline 00011111 = +31 \end{array} \qquad \begin{array}{r} 11111111 = -1 \\ 00001000 = +8 \\ \hline \cancel{1}00000111 = +7 \end{array}$$

Discard carry

Discard carry

Hexadecimal Numbers

Hexadecimal uses sixteen characters to represent numbers: the numbers 0 through 9 and the alphabetic characters A through F.

Large binary number can easily be converted to hexadecimal by grouping bits 4 at a time and writing the equivalent hexadecimal character.

Express 1001 0110 0000 1110_2 in hexadecimal:

Group the binary number by 4-bits starting from the right. Thus, 960E

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Hexadecimal Numbers

Hexadecimal is a weighted number system. The column weights are powers of 16, which increase from right to left.

$$\text{Column weights} \left\{ \begin{array}{cccc} 16^3 & 16^2 & 16^1 & .16^0 \\ 4096 & 256 & 16 & .\;1 \end{array} \right.$$

Express $1A2F_{16}$ in decimal.

Start by writing the column weights:

4096 256 16 1

1 A 2 F_{16}

1(4096) + 10(256) +2(16) +15(1) = 6703

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Octal Numbers

Octal uses eight characters the numbers 0 through 7 to represent numbers. There is no 8 or 9 character in octal.

Binary number can easily be converted to octal by grouping bits 3 at a time and writing the equivalent octal character for each group.

Express 1 001 011 000 001 110_2 in octal:

Group the binary number by 3-bits starting from the right. Thus, 113016

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

Octal Numbers

Octal is also a weighted number system. The column weights are powers of 8, which increase from right to left.

$$\text{Column weights} \left\{ \begin{array}{cccc} 8^3 & 8^2 & 8^1 & .8^0 \\ 512 & 64 & 8 & .\;1 \end{array} \right.$$

Express 3702_8 in decimal.

Start by writing the column weights:

512 64 8 1

3 7 0 2_8

3(512) + 7(64) +0(8) +2(1) = 1986

Decimal	Octal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111

BCD

Binary coded decimal (BCD) is a weighted code that is commonly used in digital systems when it is necessary to show decimal numbers such as in clock displays.

The table illustrates the difference between straight binary and BCD. BCD represents each decimal digit with a 4-bit code. Notice that the codes 1010 through 1111 are not used in BCD.

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD

You can think of BCD in terms of column weights in groups of four bits. For an 8-bit BCD number, the column weights are: 80 40 20 10 8 4 2 1.

What are the column weights for the BCD number 1000 0011 0101 1001?

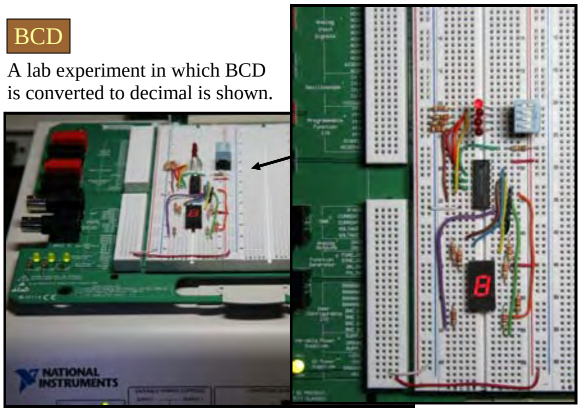
8000 4000 2000 1000 800 400 200 100 80 40 20 10 8 4 2 1

Note that you could add the column weights where there is a 1 to obtain the decimal number. For this case:

8000 + 200 +100 + 40 + 10 + 8 +1 = 8359_{10}

BCD

A lab experiment in which BCD is converted to decimal is shown.



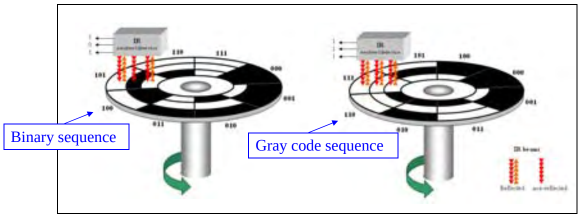
Gray code

Gray code is an unweighted code that has a single bit change between one code word and the next in a sequence. Gray code is used to avoid problems in systems where an error can occur if more than one bit changes at a time.

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Gray code

A shaft encoder is a typical application. Three IR emitter/detectors are used to encode the position of the shaft. The encoder on the left uses binary and can have three bits change together, creating a potential error. The encoder on the right uses gray code and only 1-bit changes, eliminating potential errors.



ASCII

ASCII is a code for alphanumeric characters and control characters. In its original form, ASCII encoded 128 characters and symbols using 7-bits. The first 32 characters are control characters, that are based on obsolete teletype requirements, so these characters are generally assigned to other functions in modern usage.

In 1981, IBM introduced extended ASCII, which is an 8-bit code and increased the character set to 256. Other extended sets (such as Unicode) have been introduced to handle characters in languages other than English.

Parity Method

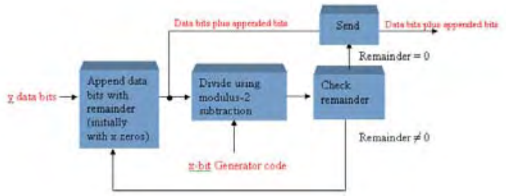
The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits). A parity bit is an “extra” bit attached to a group of bits to force the number of 1’s to be either even (even parity) or odd (odd parity).

The ASCII character for “a” is 1100001 and for “A” is 1000001. What is the correct bit to append to make both of these have odd parity?

The ASCII “a” has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII “A” has an even number of bits that are equal to 1; therefore the parity bit is 1.

Cyclic Redundancy Check

The cyclic redundancy check (CRC) is an error detection method that can detect multiple errors in larger blocks of data. At the sending end, a checksum is appended to a block of data. At the receiving end, the check sum is generated and compared to the sent checksum. If the check sums are the same, no error is detected.


y data bits
Append data bits with remainder (initially with x zeros)
Data bits plus appended bits
Divide using modulus-2 subtraction
x-bit Generator code
Check remainder
Remainder = 0
Remainder ≠ 0
Send
Data bits plus appended bits

Selected Key Terms

Byte A group of eight bits

Floating-point number A number representation based on scientific notation in which the number consists of an exponent and a mantissa.

Hexadecimal A number system with a base of 16.

Octal A number system with a base of 8.

BCD Binary coded decimal; a digital code in which each of the decimal digits, 0 through 9, is represented by a group of four bits.

Selected Key Terms

Alphanumeric Consisting of numerals, letters, and other characters

ASCII American Standard Code for Information Interchange; the most widely used alphanumeric code.

Parity In relation to binary codes, the condition of evenness or oddness in the number of 1s in a code group.

Cyclic redundancy check (CRC) A type of error detection code.

1. For the binary number 1000, the weight of the column with the 1 is

a. 4

b. 6

c. 8

d. 10

2. The 2’s complement of 1000 is

a. 0111

b. 1000

c. 1001

d. 1010

3. The fractional binary number 0.11 has a decimal value of

a. ¼

b. ½

c. ¾

d. none of the above

4. The hexadecimal number 2C has a decimal equivalent value of

a. 14

b. 44

c. 64

d. none of the above

5. Assume that a floating point number is represented in binary. If the sign bit is 1, the

a. number is negative

b. number is positive

c. exponent is negative

d. exponent is positive

6. When two positive signed numbers are added, the result may be larger that the size of the original numbers, creating overflow. This condition is indicated by

a. a change in the sign bit

b. a carry out of the sign position

c. a zero result

d. smoke

7. The number 1010 in BCD is

a. equal to decimal eight

b. equal to decimal ten

c. equal to decimal twelve

d. invalid

8. An example of an unweighted code is

a. binary

b. decimal

c. BCD

d. Gray code

9. An example of an alphanumeric code is

a. hexadecimal

b. ASCII

c. BCD

d. CRC

10. An example of an error detection method for transmitted data is the

a. parity check

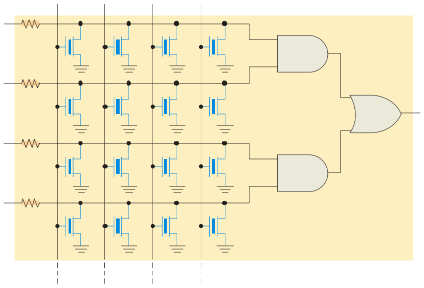
b. CRC

c. both of the above

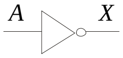
d. none of the above

Digital Electronics

Logic Gates And Boolean Algebra



The Inverter

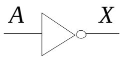


The inverter performs the Boolean **NOT** operation. When the input is LOW, the output is HIGH; when the input is HIGH, the output is LOW.

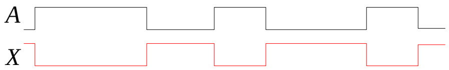
Input	Output
A	*X*
LOW (0)	HIGH (1)
HIGH (1)	LOW(0)

The **NOT** operation (complement) is shown with an over bar Thus, the Boolean expression for an inverter is $X = \overline{A}$.

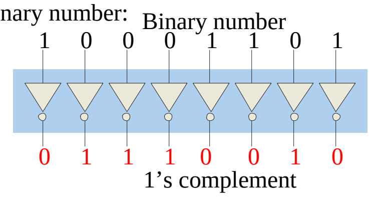
The Inverter



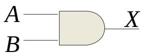
Example waveforms:

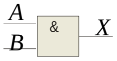


A group of inverters can be used to form the 1's complement of a binary number:



The AND Gate



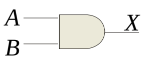


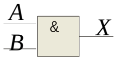
The **AND gate** produces a HIGH output when all inputs are HIGH; otherwise, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

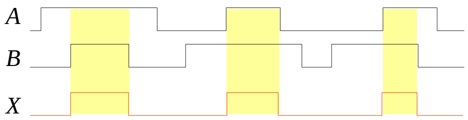
The **AND** operation is usually shown with a dot between the variables but it may be implied (no dot). Thus, the AND operation is written as $X = A \cdot B$ or $X = AB$.

The AND Gate





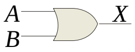
Example waveforms:

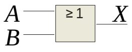


The AND operation is used in computer programming as a selective mask. If you want to retain certain bits of a binary number but reset the other bits to 0, you could set a mask with 1's in the position of the retained bits.

If the binary number 10100011 is ANDed with the mask 00001111, what is the result? 00000011

The OR Gate





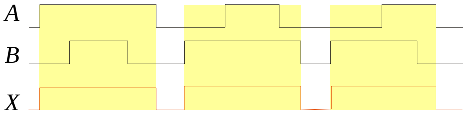
The **OR gate** produces a HIGH output if any input is HIGH; if all inputs are LOW, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

The **OR** operation is shown with a plus sign (+) between the variables. Thus, the OR operation is written as $X = A + B$.

The OR Gate

Example waveforms:

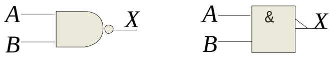


The OR operation can be used in computer programming to set certain bits of a binary number to 1.

ASCII letters have a 1 in the bit 5 position for lower case letters and a 0 in this position for capitals. (Bit positions are numbered from right to left starting with 0.) What will be the result if you OR an ASCII letter with the 8-bit mask 00100000?

The resulting letter will be lower case.

The NAND Gate

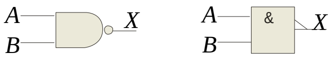


The **NAND gate** produces a LOW output when all inputs are HIGH; otherwise, the output is HIGH. For a 2-input gate, the truth table is

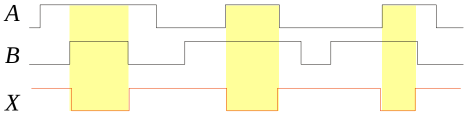
Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

The **NAND** operation is shown with a dot between the variables and an over bar covering them. Thus, the NAND operation is written as $X = \overline{A \cdot B}$ (Alternatively, $X = \overline{AB}$.)

The NAND Gate

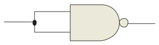


Example waveforms:



The NAND gate is particularly useful because it is a "universal" gate – all other basic gates can be constructed from NAND gates.

How would you connect a 2-input NAND gate to form a basic inverter?



The NOR Gate

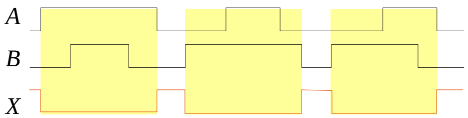
The **NOR gate** produces a LOW output if any input is HIGH; if all inputs are HIGH, the output is LOW. For a 2-input gate, the truth table is

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

The **NOR** operation is shown with a plus sign (+) between the variables and an over bar covering them. Thus, the NOR operation is written as $X = \overline{A + B}$.

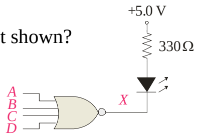
The NOR Gate

Example waveforms:



The NOR operation will produce a LOW if any input is HIGH.

When is the LED is ON for the circuit shown?



The LED will be on when any of the four inputs are HIGH.

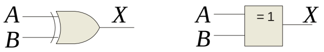
The XOR Gate

The **XOR gate** produces a HIGH output only when both inputs are at opposite logic levels. The truth table is

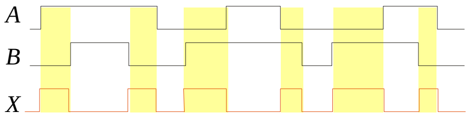
Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

The **XOR** operation is written as $X = \overline{A}B + A\overline{B}$. Alternatively, it can be written with a circled plus sign between the variables as $X = A \oplus B$.

The XOR Gate



Example waveforms:

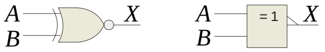


Notice that the XOR gate will produce a HIGH only when exactly one input is HIGH.

If the *A* and *B* waveforms are both inverted for the above waveforms, how is the output affected?

There is no change in the output.

The XNOR Gate

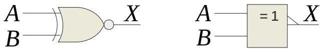


The **XNOR gate** produces a HIGH output only when both inputs are at the same logic level. The truth table is

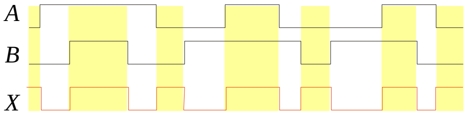
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

The **XNOR** operation shown as $X = \bar{A}\bar{B} + AB$. Alternatively, the XNOR operation can be shown with a circled dot between the variables. Thus, it can be shown as $X = A \odot B$.

The XNOR Gate



Example waveforms:



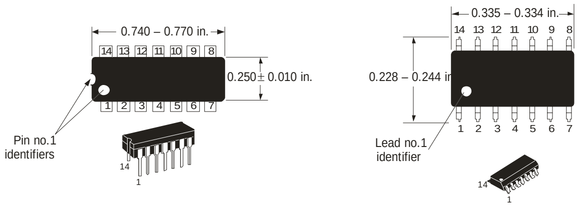
Notice that the XNOR gate will produce a HIGH when both inputs are the same. This makes it useful for comparison functions.

If the *A* waveform is inverted but *B* remains the same, how is the output affected?

The output will be inverted.

Fixed Function Logic

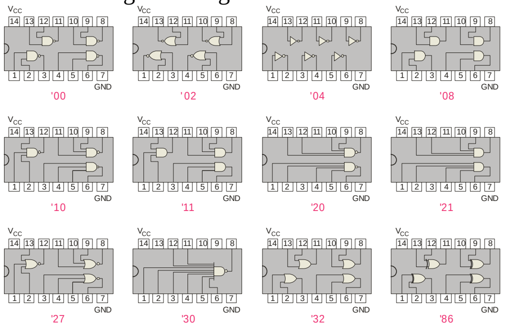
Two major fixed function logic families are TTL and CMOS. A third technology is BiCMOS, which combines the first two. Packaging for fixed function logic is shown.



DIP package
SOIC package

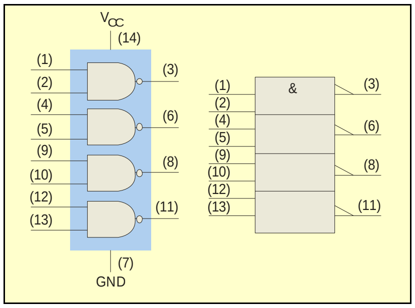
Fixed Function Logic

Some common gate configurations are shown.



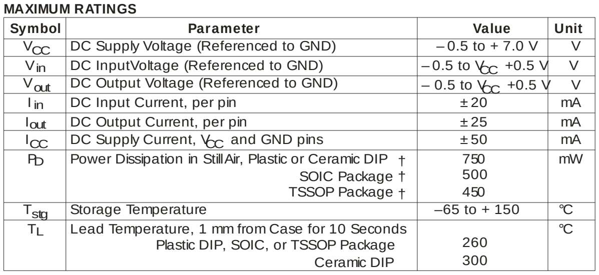
Fixed Function Logic

Logic symbols show the gates and associated pin numbers.



Fixed Function Logic

Data sheets include limits and conditions set by the manufacturer as well as DC and AC characteristics. For example, some maximum ratings for a 74HC00A are:

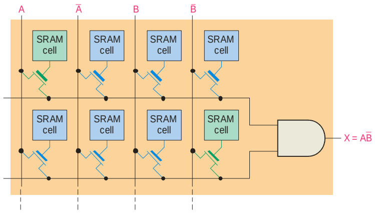


MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	DC Supply Voltage (Referenced to GND)	– 0.5 to + 7.0 V	V
V_{in}	DC InputVoltage (Referenced to GND)	– 0.5 to V_{CC} +0.5 V	V
V_{out}	DC Output Voltage (Referenced to GND)	– 0.5 to V_{CC} +0.5 V	V
I_{in}	DC Input Current, per pin	± 20	mA
I_{out}	DC Output Current, per pin	± 25	mA
I_{CC}	DC Supply Current, V_{CC} and GND pins	± 50	mA
P_D	Power Dissipation in Still Air, Plastic or Ceramic DIP † SOIC Package † TSSOP Package †	750 500 450	mW
T_{stg}	Storage Temperature	–65 to + 150	°C
T_L	Lead Temperature, 1 mm from Case for 10 Seconds Plastic DIP, SOIC, or TSSOP Package Ceramic DIP	 260 300	°C

Programmable Logic

A Programmable Logic Device (PLD) can be programmed to implement logic. There are various technologies available for PLDs. Many use an internal array of AND gates to form logic terms. Many PLDs can be programmed multiple times.



Selected Key Terms

Inverter A logic circuit that inverts or complements its inputs.

Truth table A table showing the inputs and corresponding output(s) of a logic circuit.

Timing diagram A diagram of waveforms showing the proper time relationship of all of the waveforms.

Boolean algebra The mathematics of logic circuits.

AND gate A logic gate that produces a HIGH output only when all of its inputs are HIGH.

Selected Key Terms

OR gate A logic gate that produces a HIGH output when one or more inputs are HIGH.

NAND gate A logic gate that produces a LOW output only when all of its inputs are HIGH.

NOR gate A logic gate that produces a LOW output when one or more inputs are HIGH.

Exclusive-OR gate A logic gate that produces a HIGH output only when its two inputs are at opposite levels.

Exclusive-NOR gate A logic gate that produces a LOW output only when its two inputs are at opposite levels.

1. The truth table for a 2-input AND gate is

a.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

b.

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

c.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

d.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

2. The truth table for a 2-input NOR gate is

a.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

b.

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

c.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

d.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

3. The truth table for a 2-input XOR gate is

a.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

b.

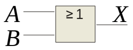
Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

c.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	0
1	0	0
1	1	1

d.

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	1

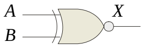
4. The symbol  is for a(n)

a. OR gate

b. AND gate

c. NOR gate

d. XOR gate

5. The symbol  is for a(n)

a. OR gate

b. AND gate

c. NOR gate

d. XOR gate

6. A logic gate that produces a HIGH output only when all of its inputs are HIGH is a(n)

a. OR gate

b. AND gate

c. NOR gate

d. NAND gate

7. The expression $X = A \oplus B$ means

a. *A* OR *B*

b. *A* AND *B*

c. *A* XOR *B*

d. *A* XNOR *B*

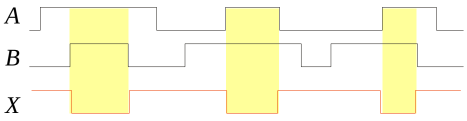
8. A 2-input gate produces the output shown. (*X* represents the output.) This is a(n)

a. OR gate

b. AND gate

c. NOR gate

d. NAND gate



9. A 2-input gate produces a HIGH output only when the inputs agree. This type of gate is a(n)

a. OR gate

b. AND gate

c. NOR gate

d. XNOR gate

Boolean Addition

In Boolean algebra, a **variable** is a symbol used to represent an action, a condition, or data. A single variable can only have a value of 1 or 0.

The **complement** represents the inverse of a variable and is indicated with an overbar. Thus, the complement of *A* is $\overline{A}$.

A **literal** is a variable or its complement.

Addition is equivalent to the OR operation. The sum term is 1 if one or more if the literals are 1. The sum term is zero only if each literal is 0.

Determine the values of *A*, *B*, and *C* that make the sum term of the expression $\overline{A} + B + \overline{C} = 0$?

Each literal must = 0; therefore *A* = 1, *B* = 0 and *C* = 1.

Boolean Multiplication

In Boolean algebra, multiplication is equivalent to the AND operation. The product of literals forms a product term. The product term will be 1 only if all of the literals are 1.

What are the values of the *A*, *B* and *C* if the product term of $A\cdot\overline{B}\cdot\overline{C} = 1$?

Each literal must = 1; therefore *A* = 1, *B* = 0 and *C* = 0.

Commutative Laws

The **commutative laws** are applied to addition and multiplication. For addition, the commutative law states

In terms of the result, the order in which variables are ORed makes no difference.

$\boxed{A + B = B + A}$

For multiplication, the commutative law states

In terms of the result, the order in which variables are ANDed makes no difference.

$\boxed{AB = BA}$

Associative Laws

The **associative laws** are also applied to addition and multiplication. For addition, the associative law states

When ORing more than two variables, the result is the same regardless of the grouping of the variables.

$\boxed{A + (B + C) = (A + B) + C}$

For multiplication, the associative law states

When ANDing more than two variables, the result is the same regardless of the grouping of the variables.

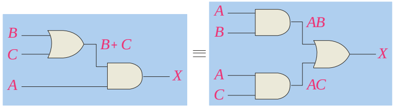
$\boxed{A(BC) = (AB)C}$

Distributive Law

The **distributive law** is the factoring law. A common variable can be factored from an expression just as in ordinary algebra. That is

$\boxed{AB + AC = A(B+ C)}$

The distributive law can be illustrated with equivalent circuits:



$A(B+ C)$ $AB + AC$

Rules of Boolean Algebra

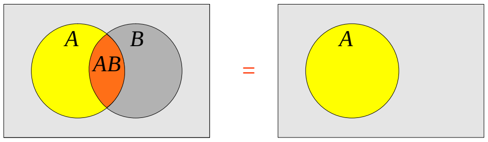
1. $A + 0 = A$ ***(Law of Union)***
2. $A + 1 = 1$ ***(Law of Union)***
3. $A \,.\, 0 = 0$ ***(Law of Intersection)***
4. $A \,.\, 1 = A$ ***(Law of Intersection)***
5. $A + A = A$ ***(Law of Idempotency)***
6. $A \,.\, A = A$ ***(Law of Idempotency)***
7. $A + \overline{A} = 1$ ***(Complementary Law)***
8. $A \,.\, \overline{A} = 0$ ***(Complementary Law)***
9. $\overline{\overline{A}} = A$ ***(Double Negative Law)***
10. $A + AB = A$ ***(Law of Absorption)***
11. $A + \overline{A}B = A + B$ ***(Law of Common Identities)***
12. $A + BC = (A + B)(A + C)$ ***(Distributive Law)***

Rules of Boolean Algebra

Rules of Boolean algebra can be illustrated with *Venn* diagrams. The variable *A* is shown as an area.

The rule *A* + *AB* = *A* can be illustrated easily with a diagram. Add an overlapping area to represent the variable *B*.

The overlap region between A and B represents *AB*.

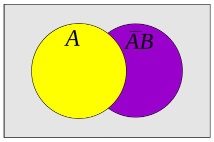


The diagram visually shows that *A* + *AB* = *A*. Other rules can be illustrated with the diagrams as well.

Rules of Boolean Algebra

Illustrate the rule $A + \overline{A}B = A + B$ with a Venn diagram.

This time, $\overline{A}$ is represented by the blue area and B again by the red circle. The intersection represents $\overline{A}B$. Notice that $A + \overline{A}B = A + B$



Rules of Boolean Algebra

Rule 12, which states that $(A + B)(A + C) = A + BC$, can be proven by applying earlier rules as follows:

$$\begin{aligned}(A + B)(A + C) &= AA + AC + AB + BC\\ &= A + AC + AB + BC\\ &= A(1 + C + B) + BC\\ &= A \cdot 1 + BC\\ &= A + BC\end{aligned}$$

This rule is a little more complicated, but it can also be shown with a Venn diagram, as given on the following slide…

Rules of Boolean Algebra

Three areas represent the variables A, B, and C.

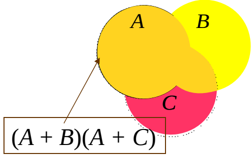
The area representing $A + B$ is shown in yellow.

The area representing $A + C$ is shown in red.

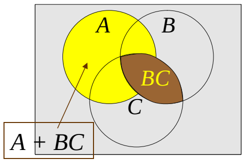
The overlap of red and yellow is shown in orange.

The overlapping area between B and C represents BC.

ORing with A gives the same area as before.



=



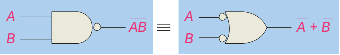
De Morgan's Theorem

DeMorgan's 1st Theorem

The complement of a product of variables is equal to the sum of the complemented variables.

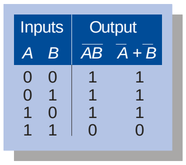
$$\overline{AB} = \overline{A} + \overline{B}$$

Applying DeMorgan's first theorem to gates:



NAND

Negative-OR



Inputs		Output	
A	B	$\overline{AB}$	$\overline{A} + \overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

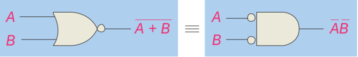
De Morgan's Theorem

DeMorgan's 2nd Theorem

The complement of a sum of variables is equal to the product of the complemented variables.

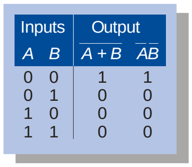
$$\overline{A + B} = \overline{A} \,.\, \overline{B}$$

Applying DeMorgan's second theorem to gates:



NOR

Negative-AND



Inputs		Output	
A	B	$\overline{A + B}$	$\overline{A}\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

De Morgan's Theorem

Apply DeMorgan's theorem to remove the overbar covering both terms from the expression $X = \overline{\overline{C} + D}$.

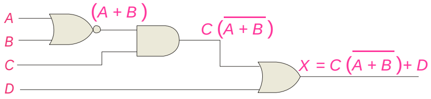
To apply DeMorgan's theorem to the expression, you can break the overbar covering both terms.

Boolean Analysis of Logic Circuits

Combinational logic circuits can be analyzed by writing the expression for each gate and combining the expressions according to the rules for Boolean algebra.

Apply Boolean algebra to derive the expression for *X*.

Write the expression for each gate:



Applying DeMorgan’s theorem and the distribution law:

$$X = C(\overline{A}\ \overline{B}) + D = \overline{A}\,\overline{B}\,C + D$$

SOP and POS forms

Boolean expressions can be written in the **sum-of-products** form (**SOP**) or in the **product-of-sums** form (**POS**). These forms can simplify the implementation of combinational logic, particularly with PLDs. In both forms, an overbar cannot extend over more than one variable.

An expression is in SOP form when two or more product terms are summed as in the following examples:

$\overline{A}\,\overline{B}\,\overline{C} + A\,B$ $\qquad A\,B\,\overline{C} + \overline{C}\,\overline{D}$ $\qquad C\,D + \overline{E}$

An expression is in POS form when two or more sum terms are multiplied as in the following examples:

$(A + B)(\overline{A} + C)$ $\qquad (A + B + \overline{C})(B + D)$ $\qquad (\overline{A} + B)C$

SOP Standard form

In **SOP standard form**, every variable in the domain must appear in each term. This form is useful for constructing truth tables or for implementing logic in PLDs.

You can expand a non-standard term to standard form by multiplying the term by a term consisting of the sum of the missing variable and its complement.

Convert $X = \overline{A}\,\overline{B} + A\,B\,C$ to standard form.

The first term does not include the variable *C*. Therefore, multiply it by the $(C + \overline{C})$, which = 1:

$$X = \overline{A}\,\overline{B}\,(C + \overline{C}) + A\,B\,C$$
$$= \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + A\,B\,C$$

POS Standard form

In **POS standard form**, every variable in the domain must appear in each sum term of the expression.

You can expand a nonstandard POS expression to standard form by adding the product of the missing variable and its complement and applying rule 12, which states that $(A + B)(A + C) = A + BC$.

Convert $X = (\overline{A} + \overline{B})(A + B + C)$ to standard form.

The first sum term does not include the variable *C*. Therefore, add $C\,\overline{C}$ and expand the result by rule 12.

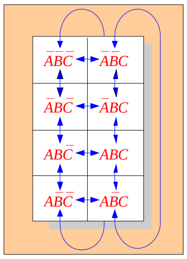
$$X = (\overline{A} + \overline{B} + C\,\overline{C})(A + B + C)$$
$$= (\overline{A} + \overline{B} + C\,)(\overline{A} + \overline{B} + \overline{C})(A + B + C)$$

Karnaugh maps

The Karnaugh map (K-map) is a tool for simplifying combinational logic with 3 or 4 variables. For 3 variables, 8 cells are required (2^3).

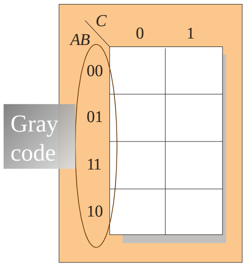
The map shown is for three variables labelled *A*, *B*, and *C*. Each cell represents one possible product term.

Each cell differs from an adjacent cell by only one variable.


$\overline{A}\overline{B}\overline{C}$
$\overline{A}\overline{B}C$
$\overline{A}B\overline{C}$
$\overline{A}BC$
$AB\overline{C}$
ABC
$A\overline{B}\overline{C}$
$A\overline{B}C$

Karnaugh maps

Cells are usually labeled using 0’s and 1’s to represent the variable and its complement.



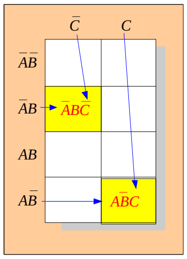
The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

Ones are read as the true variable and zeros are read as the complemented variable.

Karnaugh maps

Alternatively, cells can be labelled with the variable letters. This makes it simple to read, but it takes more time preparing the map.

Read the terms for the yellow cells.

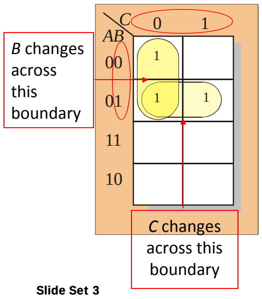


The cells are $\bar{A}B\bar{C}$ and $A\bar{B}C$.

Karnaugh maps

K-maps can simplify combinational logic by grouping cells and eliminating variables that change.

Group the 1’s on the map and read the minimum logic.



B changes across this boundary

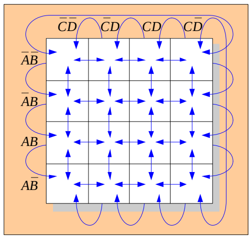
C changes across this boundary

1. Group the 1’s into two overlapping groups as indicated.
1. Read each group by eliminating any variable that changes across a boundary.
1. The vertical group is read $\bar{A}\bar{C}$.
1. The horizontal group is read $\bar{A}B$.

$$X = \bar{A}\bar{C} + \bar{A}B$$

Karnaugh maps

A 4-variable map has an adjacent cell on each of its four boundaries as shown.



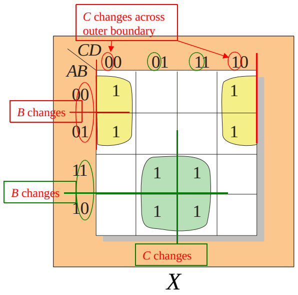
Each cell is different only by one variable from an adjacent cell.

Grouping follows the rules given in the text.

The following slide shows an example of reading a four variable map using binary numbers for the variables…

Karnaugh maps

Group the 1’s on the map and read the minimum logic.



1. Group the 1’s into two separate groups as indicated.
1. Read each group by eliminating any variable that changes across a boundary.
1. The upper (yellow) group is read as $\bar{A}\bar{D}$.
1. The lower (green) group is read as AD.

$$X = \bar{A}\bar{D} + AD$$

Selected Key Terms

Variable A symbol used to represent a logical quantity that can have a value of 1 or 0, usually designated by an italic letter.

Complement The inverse or opposite of a number. In Boolean algebra, the inverse function, expressed with a bar over the variable.

Sum term The Boolean sum of two or more literals equivalent to an OR operation.

Product term The Boolean product of two or more literals equivalent to an AND operation.

Selected Key Terms

Sum-of-products (SOP) A form of Boolean expression that is basically the ORing of ANDed terms.

Product of sums (POS) A form of Boolean expression that is basically the ANDing of ORed terms.

Karnaugh map An arrangement of cells representing combinations of literals in a Boolean expression and used for systematic simplification of the expression.

Quiz

1. The associative law for addition is normally written as

a. $A + B = B + A$

b. $(A + B) + C = A + (B + C)$

c. $AB = BA$

d. $A + AB = A$

2. The Boolean equation $AB + AC = A(B + C)$ illustrates

a. the distribution law

b. the commutative law

c. the associative law

d. De Morgan’s theorem

Quiz

3. The Boolean expression $A \cdot 1$ is equal to

a. A

b. B

c. 0

d. 1

4. The Boolean expression $A + 1$ is equal to

a. A

b. B

c. 0

d. 1

Quiz

5. The Boolean equation $AB + AC = A(B + C)$ illustrates

a. the distribution law

b. the commutative law

c. the associative law

d. DeMorgan’s theorem

Quiz

6. A Boolean expression that is in standard SOP form is

a. the minimum logic expression

b. contains only one product term

c. has every variable in the domain in every term

d. none of the above

7. Adjacent cells on a Karnaugh map differ from each other by

a. one variable

b. two variables

c. three variables

d. answer depends on the size of the map

Quiz

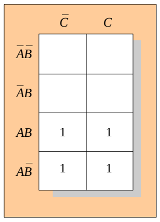
8. The minimum expression that can be read from the Karnaugh map shown is

a. $X = A$

b. $X = \overline{A}$

c. $X = B$

d. $X = \overline{B}$



	$\overline{C}$	C
$\overline{A}\overline{B}$		
$\overline{A}B$		
AB	1	1
$A\overline{B}$	1	1

Quiz

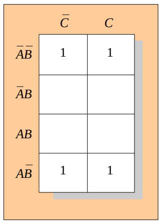
9. The minimum expression that can be read from the Karnaugh map shown is

a. $X = A$

b. $X = \overline{A}$

c. $X = B$

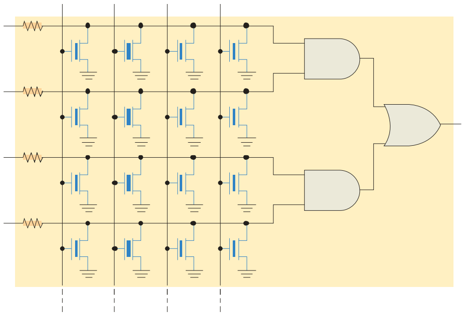
d. $X = \overline{B}$



	$\overline{C}$	C
$\overline{A}\overline{B}$	1	1
$\overline{A}B$		
AB		
$A\overline{B}$	1	1

Digital Electronics

Digital Logic Families

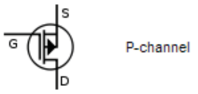


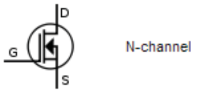
Implementation Technologies

- Two major digital IC technologies
  - CMOS (Complementary Metal Oxide Semiconductor)
  - TTL (Transistor-Transistor Logic)
- The logic operations of NOT, AND, OR, NAND, NOR, XOR, XNOR are technology-independent.
- Different in performance characteristics

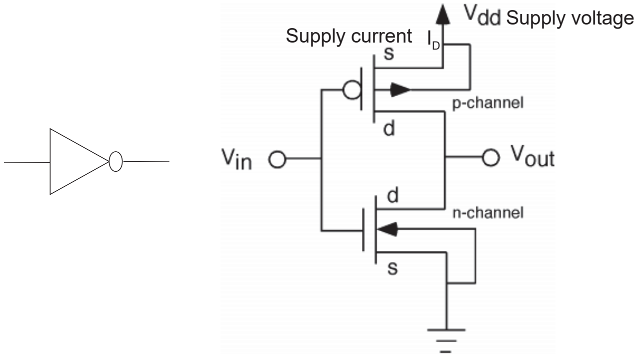
CMOS

- Complementary Metal-Oxide Semiconductor Implemented by Metal-Oxide Semiconductor Field-Effect Transistors (MOSFETs)
- Low Power Dissipation
- High Noise Immunity
- CMOS categories
  - 5 V CMOS
  - 3.3 V CMOS
  - 2.5 V CMOS
  - 1.8 V CMOS



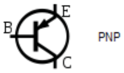


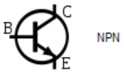
CMOS INVERTER



TTL

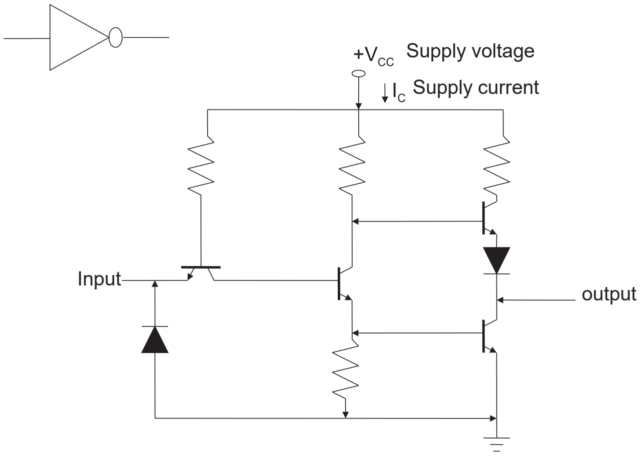
- Transistor-Transistor Logic
- Implemented by Bipolar Junction Transistors (BJTs)





- Not sensitive to electrostatic discharge
- TTL logic gates operate from 5V supply

TTL INVERTER



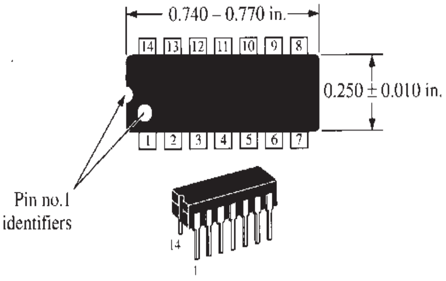
ECL

- Emitter-Coupled Logic
- Bipolar technology
- Much faster than TTL
- High fan-out is possible
- Higher power consumption

CMOS , TTL, ECL

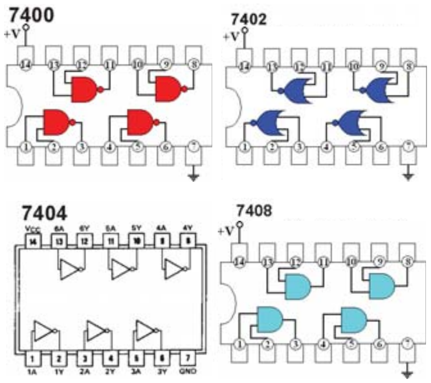
- Speed (high to low)
  - ECL → TTL → CMOS
- Power Dissipation (Low to high)
  - CMOS → TTL → ECL

IC Package

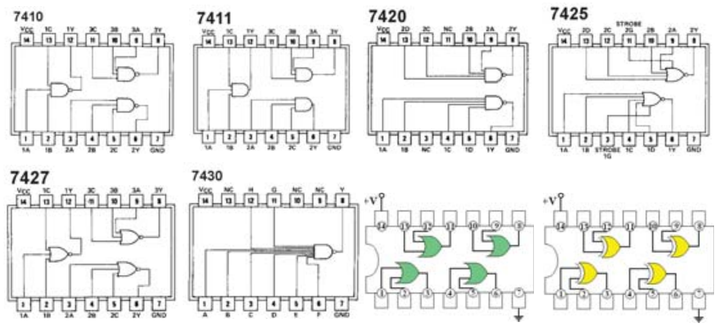


Some common IC gates

Series	Type of logic gate
74XX00	Quad 2-input NAND
74XX02	Quad 2-input NOR
74XX04	Hex inverter
74XX10	Triple 3-input NAND
74XX11	Triple 3-input AND
74XX20	Dual 4-input NAND
74XX25	Dual 4-input NOR
74XX32	Quad 2-input OR
74XX86	Quad 2-input XOR



Some common IC gates



Some common IC gates

- XX (in table 1) indicates the IC series
- For example:
  - HC – High-speed CMOS
  - AC – Advanced CMOS
  - LV – Low-voltage CMOS
  - S – Schottky TTL
  - LS – Low-power Schottky TTL

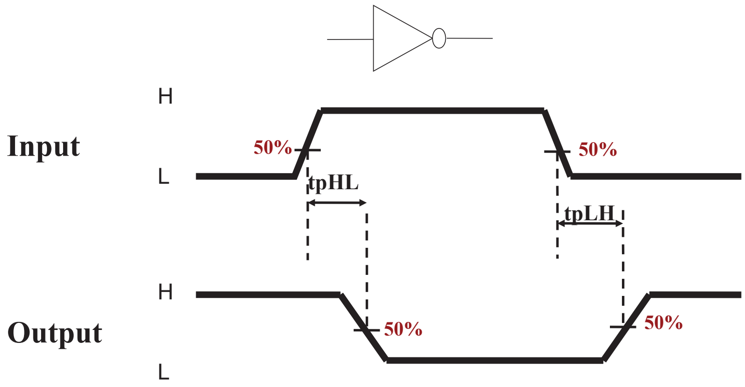
Performance Characteristics and Parameters

- Switching Speed (Propagation delay time)
- Power dissipation
- Fan-in
- Fan-out/drive capability
- Noise Margin
- Packaging Density

Propagation Delay Time, t_p

- Depends on the switching speed that a logic circuit can operate.
- Low speed circuit or high speed circuit
- Definition of t_p
  - Time interval between the application of an input pulse and the occurrence of the resulting output pulse.
- 2 measurements
  - t_{PHL}—t_p with the output changing from HIGH to LOW.
  - t_{PLH}—t_p with the output changing from LOW to HIGH.

Propagation Delay Time, t_p



Power Dissipation, P_D

- Power Dissipation =

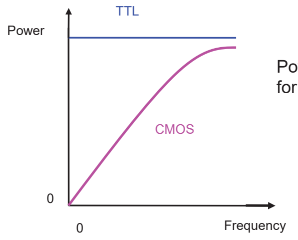
DC supply voltage x average supply current

$$P_D = V_{CC}\left(\frac{I_{CCH} + I_{CCL}}{2}\right)$$

- The above calculation is based on 50% duty cycle.
- VCC—supply voltage
- *ICCL*—supply current for the LOW output state
- *ICCH*—supply current for the HIGH output state

Power Dissipation

- CMOS have very low power dissipation compared to TTL.
- However,

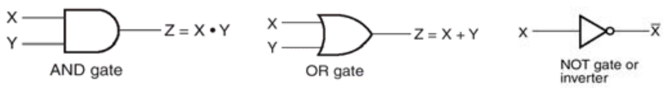


Power versus frequency curves for TTL and CMOS

Fan-in

- The fan-in of a gate – the number of inputs connected to the gate.
- The larger the fan-in is, the larger the propagation delay of the gate will be.

  Fan-in ↑ , speed ↓
- Typical gate has a fan-in of 1 or 2.



Fan-out

- The fan-out of a gate—the maximum number of inputs that are connected to the output of the gate and
  - still maintain the output voltage levels within specified limits. (TTL)
  - Still maintain the operation frequency within specified limit. (CMOS)
-

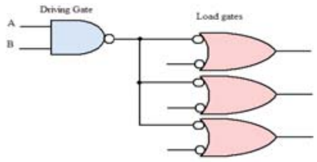
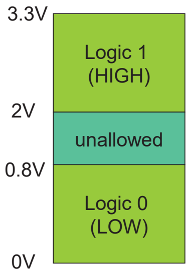


Fig.01: Loading a gate output with gate inputs

Noise Immunity

- Noise –unwanted voltage that is induced in electrical circuits.
- Noise may present a threat to the proper operation of the circuit.

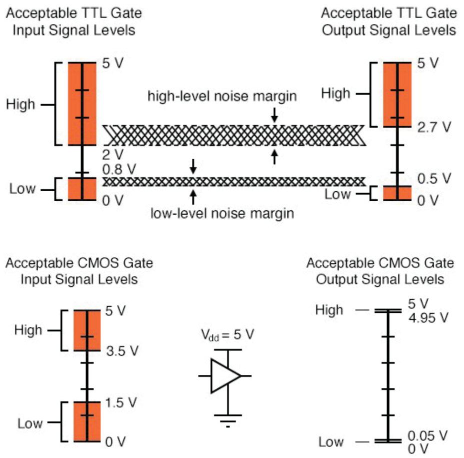


Logic levels

Noise Immunity

- In order not to be affected by noise, the logic circuit must have noise immunity.
- Noise immunity—the ability to tolerate a certain amount of unwanted voltage fluctuation on its inputs without changing its output state.

NOISE MARGIN

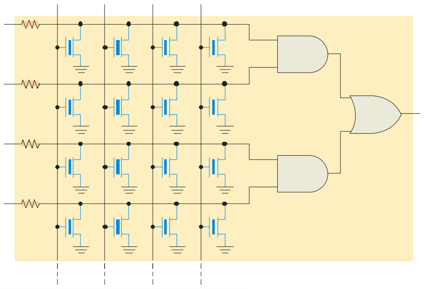


Packaging Densitys

Complexity	Gates per chip
Small-scale integration (SSI)	< 12
Medium-scale integration (MSI)	12 to 99
Large-scale integration (LSI)	100 to 9999
Very large-scale integration (VLSI)	10,000 to 99,999
Ultra large-scale integration (ULSI)	100,000 to 999,999
Giga-scale integration (GSI)	1,000,000 or more

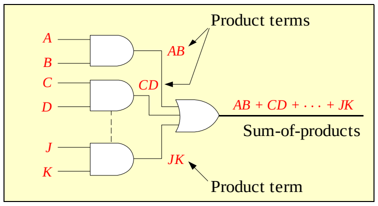
Digital Electronics

Combinational Logic Analysis



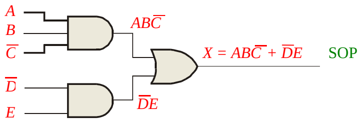
Combinational Logic Circuits

In Sum-of-Products (SOP) form, basic combinational circuits can be directly implemented with AND-OR combinations if the necessary complement terms are available.



Combinational Logic Circuits

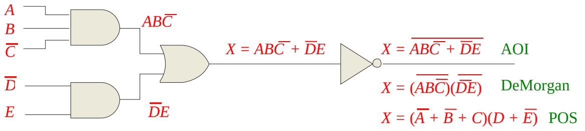
An example of an SOP implementation is shown. The SOP expression is an AND-OR combination of the input variables and the appropriate complements.



Combinational Logic Circuits

When the output of a SOP form is inverted, the circuit is called an AND-OR-Invert circuit. The AOI configuration lends itself to product-of-sums (POS) implementation.

An example of an AOI implementation is shown. The output expression can be changed to a POS expression by applying DeMorgan’s theorem twice.


A
B
$\overline{C}$
$\overline{D}$
E
$AB\overline{C}$
$\overline{D}E$
$X = AB\overline{C} + \overline{D}E$
$X = \overline{AB\overline{C} + \overline{D}E}$
AOI
$X = (\overline{AB\overline{C}})(\overline{\overline{D}E})$
DeMorgan
$X = (\overline{A} + \overline{B} + C)(D + \overline{E})$
POS

Exclusive-OR Logic

Exclusive-OR Logic

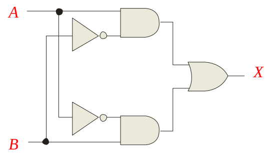
The truth table for an exclusive-OR gate is

Inputs		Output
A	*B*	*X*
0	0	0
0	1	1
1	0	1
1	1	0

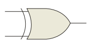
Notice that the output is HIGH whenever *A* and *B* disagree.

The Boolean expression is $X = \overline{A}B + A\overline{B}$

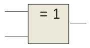
The circuit can be drawn as



Symbols:



Distinctive shape



Rectangular outline

Exclusive-NOR Logic

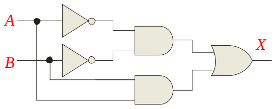
The truth table for an exclusive-NOR gate is

Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	1

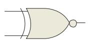
Notice that the output is HIGH whenever *A* and *B* agree.

The Boolean expression is $X = \overline{A}\,\overline{B} + AB$

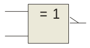
The circuit can be drawn as



Symbols:

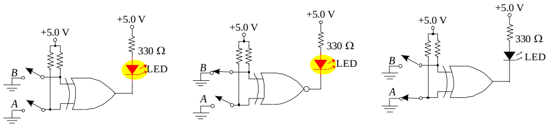


Distinctive shape



Rectangular outline

For each circuit, determine if the LED should be on or off.



(a) (b)

(c)

Circuit (a): XOR, inputs agree, output is LOW, LED is ON.

Circuit (b): XNOR, inputs disagree, output is LOW, LED is ON.

Circuit (c): XOR, inputs disagree, output is HIGH, LED is OFF.

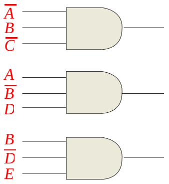
Implementing Combinational Logic

Implementing a SOP expression is done by first forming the AND terms; then the terms are ORed together.

Show the circuit that will implement the Boolean expression $X = ABC + ABD + BD\overline{E}$. (Assume that the variables and their complements are available.)

Start by forming the terms using three 3-input AND gates.

Then combine the three terms using a 3-input OR gate.

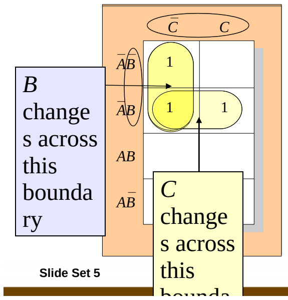


$X = \overline{A}B\overline{C} + A\overline{B}D + B\overline{D}E$

Karnaugh Map Implementation

For basic combinational logic circuits, the Karnaugh map can be read and the circuit drawn as a minimum SOP.

A Karnaugh map is drawn from a truth table. Read the minimum SOP expression and draw the circuit.



B changes across this boundary

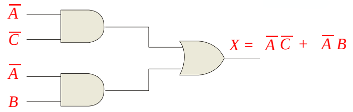
C changes across this bounda

1. Group the 1’s into two overlapping groups as indicated.
1. Read each group by eliminating any variable that changes across a boundary.
1. The vertical group is read $\overline{A}\,\overline{C}$.
1. The horizontal group is read $\overline{A}B$.

The circuit is on the next slide:

continued…

Circuit:



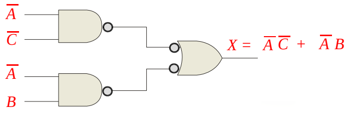
The result is shown as a sum of products.

It is a simple matter to implement this form using only NAND gates as shown in the text and following example.

NAND Logic

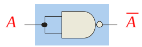
Convert the circuit in the previous example to one that uses only NAND gates.

Recall from Boolean algebra that double inversion cancels. By adding inverting bubbles to above circuit, it is easily converted to NAND gates:

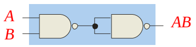


Universal Gates

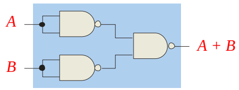
NAND gates are sometimes called **universal** gates because they can be used to produce the other basic Boolean functions.



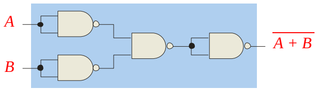
Inverter



AND gate



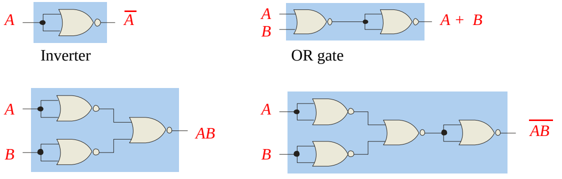
OR gate



NOR gate

Universal Gates

NOR gates are also **universal** gates and can form all of the basic gates.



Inverter

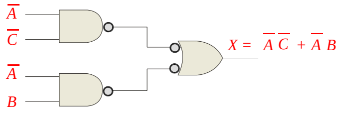
OR gate

AND gate

NAND gate

NAND Logic

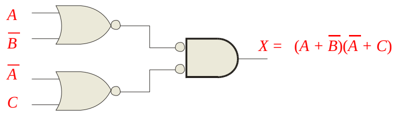
Recall from DeMorgan's theorem that $\overline{AB} = \overline{A} + \overline{B}$. By using equivalent symbols, it is simpler to read the logic of SOP forms. The earlier example shows the idea:



The logic is easy to read if you (mentally) cancel the two connected bubbles on a line.

NOR Logic

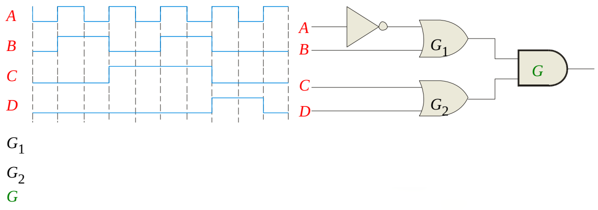
Alternatively, DeMorgan's theorem can be written as $\overline{A + B} = \overline{A}\overline{B}$. By using equivalent symbols, it is simpler to read the logic of POS forms. For example,



Again, the logic is easy to read if you cancel the two connected bubbles on a line.

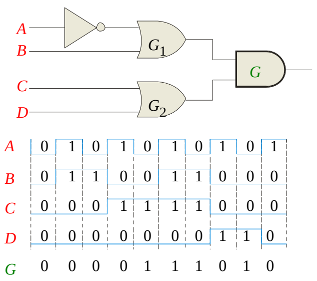
Pulsed Waveform

For combinational circuits with pulsed inputs, the output can be predicted by developing intermediate outputs and combining the result. For example, the circuit shown can be analyzed at the outputs of the OR gates:



Pulsed Waveform

Alternatively, you can develop the truth table for the circuit and enter 0's and 1's on the waveforms. Then read the output from the table.



A	0	1	0	1	0	1	0	1	0	1
B	0	1	1	0	0	1	1	0	0	0
C	0	0	0	1	1	1	1	0	0	0
D	0	0	0	0	0	0	0	1	1	0
G	0	0	0	0	1	1	1	0	1	0

Inputs				Output
A	B	C	D	X
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Selected Key Terms

Universal gate Either a NAND or a NOR gate. The term universal refers to a property of a gate that permits any logic function to be implemented by that gate or by a combination of gates of that kind.

Negative-OR The dual operation of a NAND gate when the inputs are active-LOW.

Negative-AND The dual operation of a NOR gate when the inputs are active-LOW.

1. Assume an AOI expression is $\overline{AB + CD}$. The equivalent POS expression is

a. $(A + B)(C + D)$

b. $(\overline{A} + \overline{B})(\overline{C} + \overline{D})$

c. $(\overline{A + B})(\overline{C + D})$

d. none of the above

2. The truth table shown is for

a. a NAND gate

b. a NOR gate

c. an exclusive-OR gate

d. an exclusive-NOR gate

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

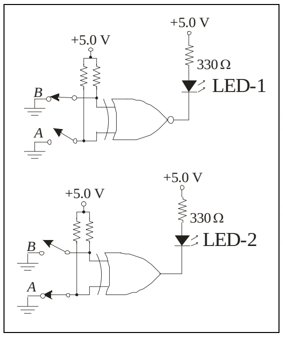
3. An LED that should be ON is

a. LED-1

b. LED-2

c. neither

d. both



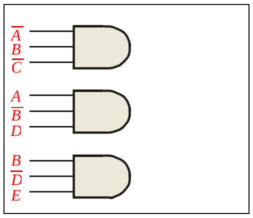
4. To implement the SOP expression $X = \overline{A}B\overline{C} + A\overline{B}D + B\overline{D}E$, the type of gate that is needed is a

a. 3-input AND gate

b. 3-input NAND gate

c. 3-input OR gate

d. 3-input NOR gate



5. Reading the Karnaugh map, the logic expression is

a. $\overline{A}\,\overline{C} + \overline{A}B$

b. $\overline{A}B + A\overline{C}$

c. $A\overline{B} + B\overline{C}$

d. $\overline{A}B + \overline{A}\,\overline{C}$

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	1	
$\overline{A}B$	1	1
AB		
$A\overline{B}$		

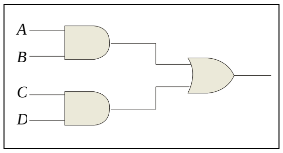
6. The circuit shown will have identical logic out if all gates are changed to

a. AND gates

b. OR gates

c. NAND gates

d. NOR gates



7. The two types of gates which are called *universal gates* are

a. AND/OR

b. NAND/NOR

c. AND/NAND

d. OR/NOR

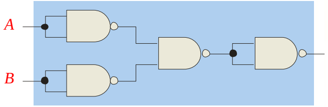
8. The circuit shown is equivalent to an

a. AND gate

b. XOR gate

c. OR gate

d. none of the above



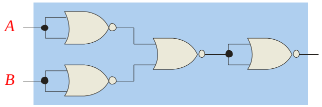
9. The circuit shown is equivalent to

a. an AND gate

b. an XOR gate

c. an OR gate

d. none of the above



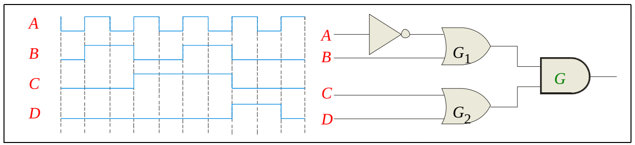
10. During the first *three* intervals for the pulsed circuit shown, the output of

a. *G*1 is LOW and *G*2 is LOW

b. *G*1 is LOW and *G*2 is HIGH

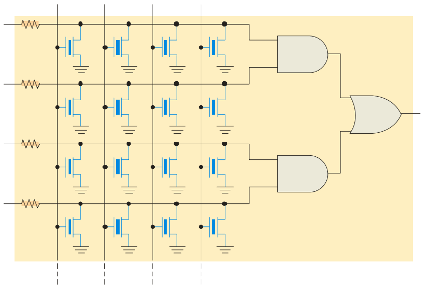
c. *G*1 is HIGH and *G*2 is LOW

d. *G*1 is HIGH and *G*2 is HIGH



Digital Electronics

Functions of Combinational Logic



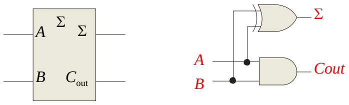
Half-Adder

Basic rules of binary addition are performed by a **half adder**, which has two binary inputs (*A* and *B*) and two binary outputs (Carry out and Sum).

The inputs and outputs can be summarized on a truth table.

Inputs		Outputs	
A	*B*	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The logic symbol and equivalent circuit are:

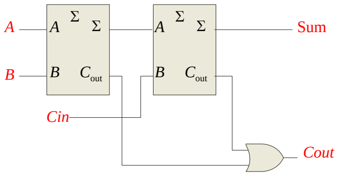


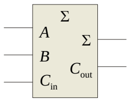
Full-Adder

By contrast, a **full adder** has three binary inputs (*A*, *B*, and Carry in) and two binary outputs (Carry out and Sum). The truth table summarizes the operation.

A full-adder can be constructed from two half adders as shown:

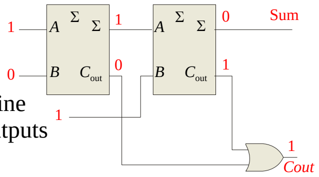
Inputs			Outputs	
A	*B*	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





Symbol

Full-Adder



For the given inputs, determine the intermediate and final outputs of the full adder.

The first half-adder has inputs of 1 and 0; therefore the Sum =1 and the Carry out = 0.

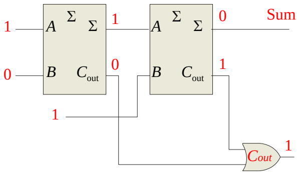
The second half-adder has inputs of 1 and 1; therefore the Sum = 0 and the Carry out = 1.

The OR gate has inputs of 1 and 0, therefore the final carry out = 1.

Full-Adder

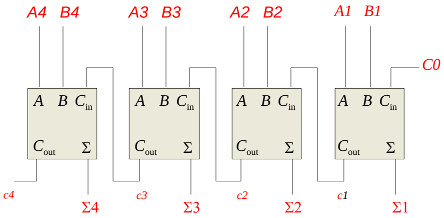
Notice that the result from the previous example can be read directly on the truth table for a full adder.

Inputs			Outputs	
A	*B*	C_{in}	C_{out}	Σ
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



Parallel Adders

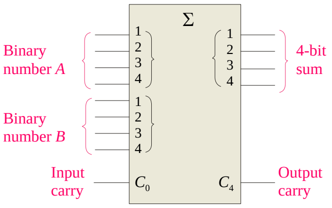
Full adders are combined into parallel adders that can add binary numbers with multiple bits. A 4-bit adder is shown.



The output carry (C_4) is not ready until it propagates through all of the full adders. This is called *ripple carry*, delaying the addition process.

Parallel Adders

The logic symbol for a 4-bit parallel adder is shown. This 4-bit adder includes a carry in (labeled (C_0) and a Carry out (labeled C_4).



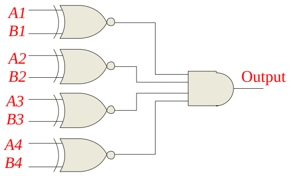
The 74LS283 is an example. It features *look-ahead carry*, which adds logic to minimize the output carry delay. For the 74LS283, the maximum delay to the output carry is 17 ns.

Comparators

The function of a comparator is to compare the magnitudes of two binary numbers to determine the relationship between them. In the simplest form, a comparator can test for equality using XNOR gates.

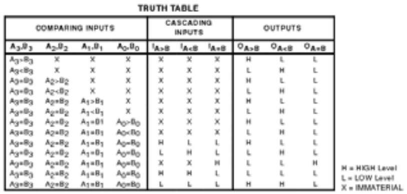
How could you test two 4-bit numbers for equality?

AND the outputs of four XNOR gates.



Comparators

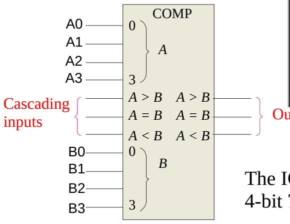
IC comparators provide outputs to indicate which of the numbers is larger or if they are equal. The bits are numbered starting at 0, rather than 1 as in the case of adders. Cascading inputs are provided to expand the comparator to larger numbers.



TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

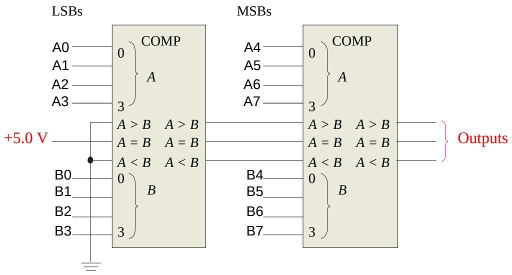
H = HIGH Level
L = LOW Level
X = IMMATERIAL



The IC shown is the 4-bit 74LS85.

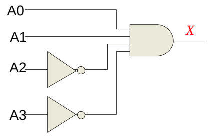
Comparators

IC comparators can be expanded using the cascading inputs as shown. The lowest order comparator has a HIGH on the $A = B$ input.

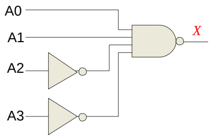


Decoders

A **decoder** is a logic circuit that detects the presence of a specific combination of bits at its input. Two simple decoders that detect the presence of the binary code 0011 are shown. The first has an active HIGH output; the second has an active LOW output.



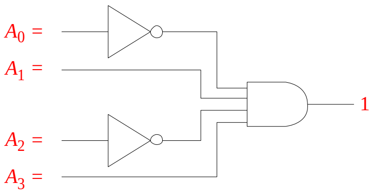
Active HIGH decoder for 0011



Active LOW decoder for 0011

Decoders

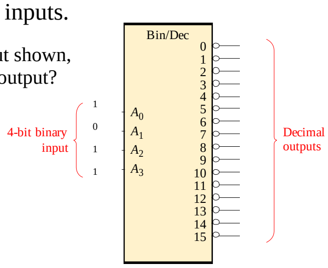
Assume the output of the decoder shown is a logic 1. What are the inputs to the decoder?



Decoders

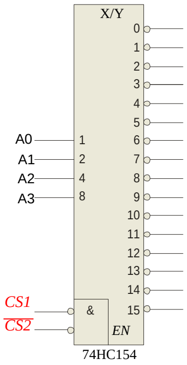
IC decoders have multiple outputs to decode any combination of inputs. For example the binary-to-decimal decoder shown here has 16 outputs – one for each combination of binary inputs.

For the input shown, what is the output?



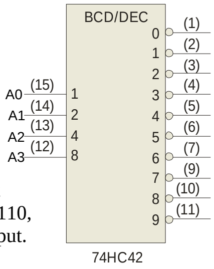
Decoders

A specific integrated circuit decoder is the 74HC154 (shown as a 4-to-16 decoder). It includes two active LOW chip select lines which must be at the active level to enable the outputs. These lines can be used to expand the decoder to larger inputs.



Decoders

BCD-to-decimal decoders accept a binary coded decimal input and activate one of ten possible decimal digit indications.



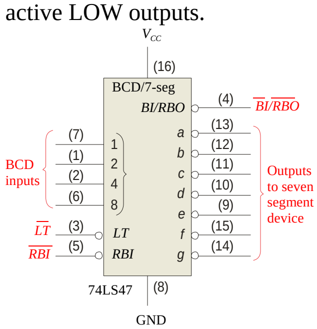
Example Assume the inputs to the 74HC42 decoder are the sequence 0101, 0110, 0011, and 0010. Describe the output.

Solution All lines are HIGH except for one active output, which is LOW. The active outputs are 5, 6, 3, and 2 in that order.

BCD Decoder/Driver

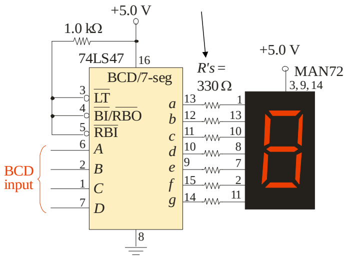
Another useful decoder is the 74LS47. This is a BCD-to-seven segment display with active LOW outputs.

The *a*-*g* outputs are designed for much higher current than most devices (hence the word driver in the name).



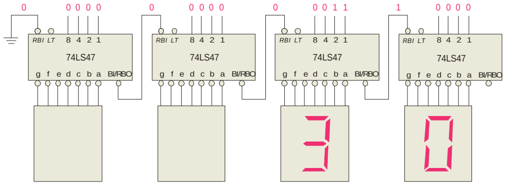
BCD Decoder/Driver

Here the 7447A is an connected to an LED seven segment display. Notice the current limiting resistors, required to prevent overdriving the LED display.



BCD Decoder/Driver

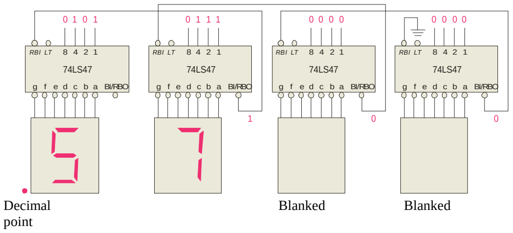
The 74LS47 features leading zero suppression, which blanks unnecessary leading zeros but keeps significant zeros as illustrated here. The $\overline{BI}/\overline{RBO}$ output is connected to the $\overline{RBI}$ input of the next decoder.



Depending on the display type, current limiting resistors may be required.

BCD Decoder/Driver

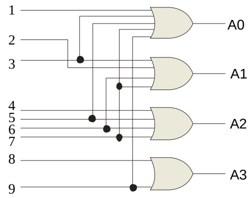
Trailing zero suppression blanks unnecessary trailing zeros to the right of the decimal point as illustrated here. The $\overline{RBI}$ input is connected to the $\overline{BI}/\overline{RBO}$ output of the following decoder.



Encoders

An **encoder** accepts an active logic level on one of its inputs and converts it to a coded output, such as BCD or binary.

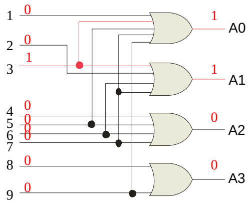
The decimal to BCD is an encoder with an input for each of the ten decimal digits and four outputs that represent the BCD code for the active digit. The basic logic diagram is shown. There is no zero input because the outputs are all LOW when the input is zero.



Encoders

Example Show how the decimal-to-BCD encoder converts the decimal number 3 into a BCD 0011.

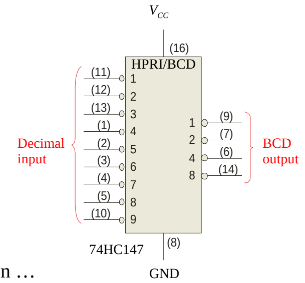
Solution The top two OR gates have ones as indicated with the red lines. Thus the output is 0111.



Encoders

The 74HC147 is an example of an IC encoder. It is has ten active-LOW inputs and converts the active input to an active-LOW BCD output.

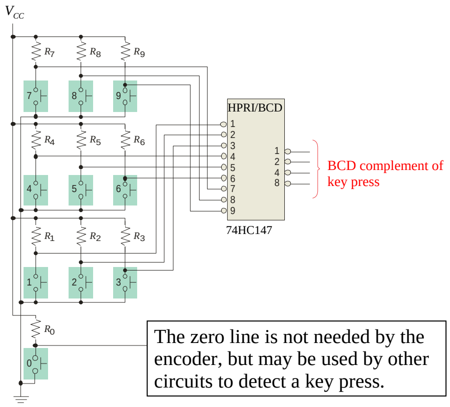
This device offers additional flexibility in that it is a **priority encoder**. This means that if more than one input is active, the one with the highest order decimal digit will be active.



The next slide shows an application …

Encoders

Keyboard encoder



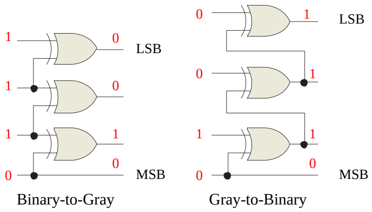
The zero line is not needed by the encoder, but may be used by other circuits to detect a key press.

Code converters

There are various code converters that change one code to another. Two examples are the four bit binary-to-Gray converter and the Gray-to-binary converter.

Example Show the conversion of binary 0111 to Gray and back.

Solution

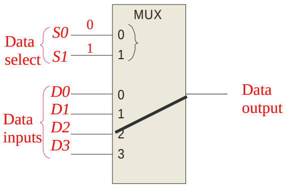


Binary-to-Gray Gray-to-Binary

Multiplexers

A multiplexer (MUX) selects one data line from two or more input lines and routes data from the selected line to the output. The particular data line that is selected is determined by the select inputs.

Two select lines are shown here to choose any of the four data inputs.



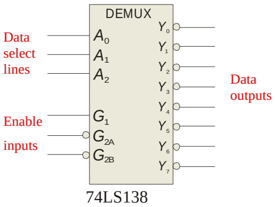
Question

Which data line is selected if S_1S_0 = 10? D2

Demultiplexers

A demultiplexer (DEMUX) performs the opposite function from a MUX. It switches data from one input line to two or more data lines depending on the select inputs.

The 74LS138 was introduced previously as a decoder but can also serve as a DEMUX. When connected as a DEMUX, data is applied to one of the enable inputs, and routed to the selected output line depending on the select variables. Note that the outputs are active-LOW as illustrated in the following example…

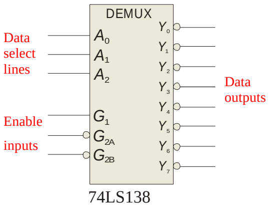


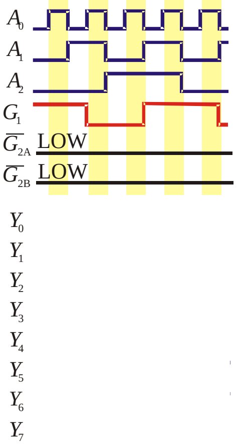
Demultiplexers

Example Determine the outputs, given the inputs shown.

Solution

The output logic is opposite to the input because of the active-LOW convention. (Red shows the selected line).





Parity Generators/Checkers

Parity is an error detection method that uses an extra bit appended to a group of bits to force them to be either odd or even. In even parity, the total number of ones is even; in odd parity the total number of ones is odd.

Example The ASCII letter S is 1010011. Show the parity bit for the letter S with odd and even parity.

Solution

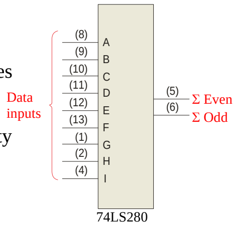
S with odd parity = 11010011
S with even parity = 01010011

Parity Generators/Checkers

The 74LS280 can be used to generate a parity bit or to check an incoming data stream for even or odd parity.

Checker: The 74LS280 can test codes with up to 9 bits. The even output will normally be HIGH if the data lines have even parity; otherwise it will be LOW. Likewise, the odd output will normally be HIGH if the data lines have odd parity; otherwise it will be LOW.

Generator: To generate even parity, the parity bit is taken from the odd parity output. To generate odd parity, the output is taken from the even parity output.



Selected Key Terms

Full-adder A digital circuit that adds two bits and an input carry bit to produce a sum and an output carry.

Cascading Connecting two or more similar devices in a manner that expands the capability of one device.

Ripple carry A method of binary addition in which the output carry from each adder becomes the input carry of the next higher order adder.

Look-ahead carry A method of binary addition whereby carries from the preceding adder stages are anticipated, thus eliminating carry propagation delays.

Selected Key Terms

Decoder A digital circuit that converts coded information into a familiar or noncoded form.

Encoder A digital circuit that converts information into a coded form.

Priority encoder An encoder in which only the highest value input digit is encoded and any other active input is ignored.

Multiplexer (MUX) A circuit that switches digital data from several input lines onto a single output line in a specified time sequence.

Demultiplexer (DEMUX) A circuit that switches digital data from one input line onto a several output lines in a specified time sequence.

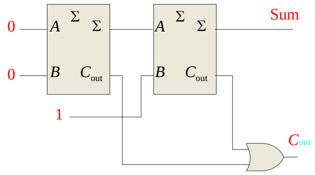
1. For the full-adder shown, assume the input bits are as shown with *A* = 0, *B* = 0, *C*in = 1. The Sum and *C* will be

a. Sum = 0 Cout = 0

b. Sum = 0 Cout = 1

c. Sum = 1 Cout = 0

d. Sum = 1 Cout = 1



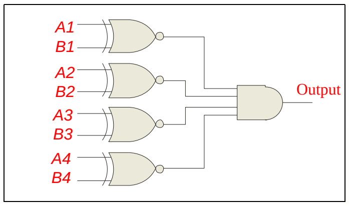
2. The output will be LOW if

a. *A* < *B*

b. *A* > *B*

c. both a and b are correct

d. *A* = *B*



3. If you expand two 4-bit comparators to accept two 8-bit numbers, the output of the least significant comparator is

a. equal to the final output

b. connected to the cascading inputs of the most significant comparator

c. connected to the output of the most significant comparator

d. not used

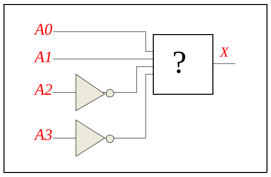
4. Assume you want to decode the binary number 0011 with an active-LOW decoder. The missing gate should be

a. an AND gate

b. an OR gate

c. a NAND gate

d. a NOR gate



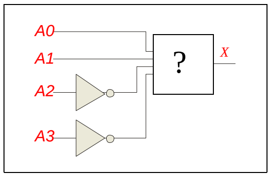
5. Assume you want to decode the binary number 0011 with an active-HIGH decoder. The missing gate should be

a. an AND gate

b. an OR gate

c. a NAND gate

d. a NOR gate



Quiz

6. The 74138 is a 3-to-8 decoder. Together, two of these ICs can be used to form one 4-to-16 decoder. To do this, connect

a. one decoder to the LSBs of the input; the other decoder to the MSBs of the input

b. all chip select lines to ground

c. all chip select lines to their active levels

d. one chip select line on each decoder to the input MSB

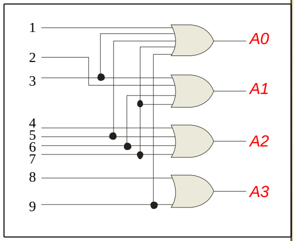
7. The decimal-to-binary encoder shown does not have a zero input. This is because

a. when zero is the input, all lines should be LOW

b. zero is not important

c. zero will produce illegal logic levels

d. another encoder is used for zero



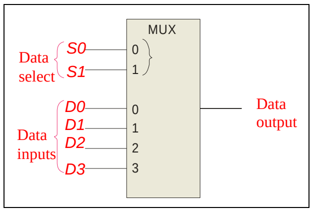
8. If the data select lines of the MUX are $S1S0$ = 11, the output will be

a. LOW

b. HIGH

c. equal to $D0$

d. equal to $D3$



9. The 74138 decoder can also be used as

a. an encoder

b. a DEMUX

c. a MUX

d. none of the above

10. The 74LS280 can generate even or odd parity. It can also be used as

a. an adder

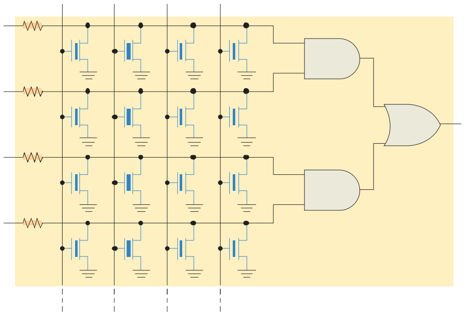
b. a parity tester

c. a MUX

d. an encoder

Digital Electronics

Latches, Flip-Flops, Timers

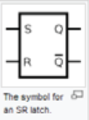


Latches

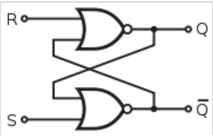
A **latch** is a temporary storage device that has two stable states (bistable). It is a basic form of memory.

The S-R (Set-Reset) latch is the most basic type. It can be constructed from NOR gates or NAND gates. With NOR gates, the latch responds to active-HIGH inputs; with NAND gates, it responds to active-LOW inputs.

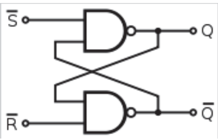
SR latch operation		
S	R	Action
0	0	Keep state
0	1	Q = 0
1	0	Q = 1
1	1	Restricted combination



The symbol for an SR latch.

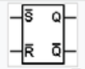


NOR Active-HIGH Latch



NAND Active-LOW Latch

$\overline{S}\overline{R}$ latch operation		
$\overline{S}$	$\overline{R}$	Action
0	0	Restricted combination
0	1	Q = 1
1	0	Q = 0
1	1	Keep state

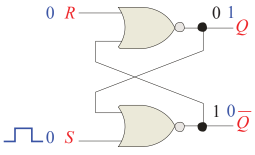


Symbol for an $\overline{SR}$ NAND latch

Latches

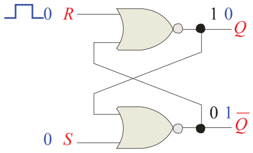
The active-HIGH *S*-*R* latch is in a stable (latched) condition when both inputs are LOW.

Assume the latch is initially RESET ($Q = 0$) and the inputs are at their inactive level (0). To SET the latch ($Q = 1$), a momentary HIGH signal is applied to the *S* input while the *R* remains LOW.



Latch initially RESET

To RESET the latch ($Q = 0$), a momentary HIGH signal is applied to the *R* input while the *S* remains LOW.

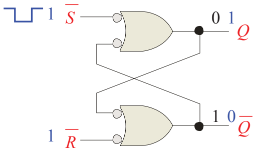


Latch initially SET

Latches

The active-LOW $\overline{S}$-$\overline{R}$ latch is in a stable (latched) condition when both inputs are HIGH.

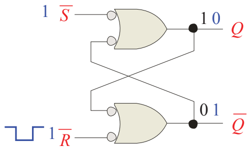
Assume the latch is initially RESET ($Q = 0$) and the inputs are at their inactive level (1). To SET the latch ($Q = 1$), a momentary LOW signal is applied to the $\overline{S}$ input while the $\overline{R}$ remains HIGH.



Latch initially RESET

To RESET the latch a momentary LOW is applied to the $\overline{R}$ input while $\overline{S}$ is HIGH.

Never apply an active set and reset at the same time (invalid).



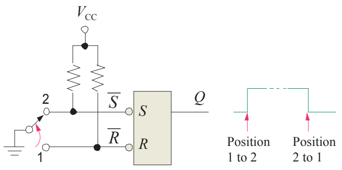
Latch initially SET

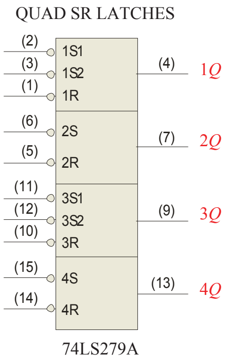
Latches

The active-LOW $\overline{S}$-$\overline{R}$ latch is available as the 74LS279A IC.

It features four internal latches with two having two $\overline{S}$ inputs. To SET any of the latches, the $\overline{S}$ line is pulsed low. It is available in several packages.

$\overline{S}$-$\overline{R}$ latches are frequently used for switch debounce circuits as shown:

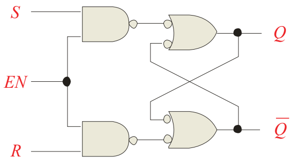




Latches

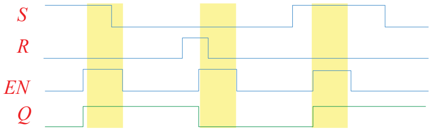
A gated latch is a variation on the basic latch.

The gated latch has an additional input, called enable (*EN*) that must be HIGH in order for the latch to respond to the *S* and *R* inputs.



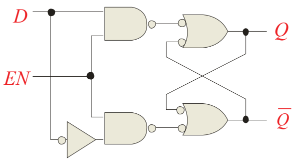
Example Show the *Q* output with relation to the input signals. Assume *Q* starts LOW.

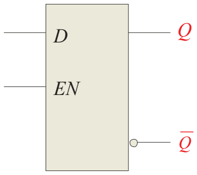
Solution Keep in mind that *S* and *R* are only active when *EN* is HIGH.



Latches

The *D* latch is an variation of the *S*-*R* latch but combines the *S* and *R* inputs into a single *D* input as shown:





A simple rule for the *D* latch is:

Q follows *D* when the Enable is active.

Latches

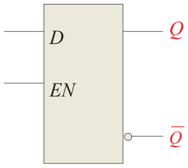
The truth table for the *D* latch summarizes its operation. If *EN* is LOW, then there is no change in the output and it is latched.

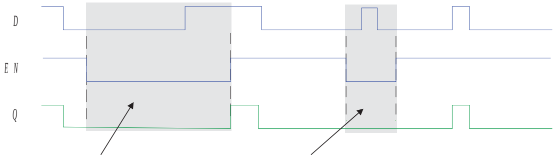
Inputs		Outputs		
D	*EN*	*Q*	$\overline{Q}$	Comments
0	1	0	1	RESET
1	1	1	0	SET
X	0	Q_0	$\overline{Q}_0$	No change

Latches

Example

Determine the *Q* output for the *D* latch, given the inputs shown.



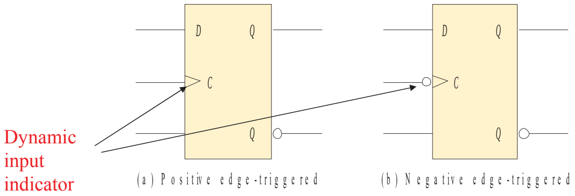


Notice that the Enable is not active during these times, so the output is latched.

Flip-flops

A flip-flop differs from a latch in the manner it changes states. A flip-flop is a clocked device, in which only the clock edge determines when a new bit is entered.

The active edge can be positive or negative.



(a) Positive edge-triggered (b) Negative edge-triggered

Flip-flops

The truth table for a positive-edge triggered D flip-flop shows an up arrow to remind you that it is sensitive to its *D* input only on the rising edge of the clock; otherwise it is latched. The truth table for a negative-edge triggered D flip-flop is identical except for the direction of the arrow.

Inputs		Outputs		
D	CLK	*Q*	$\overline{Q}$	Comments
1	↑	1	0	SET
0	↑	0	1	RESET

(a) Positive-edge triggered

Inputs		Outputs		
D	CLK	*Q*	$\overline{Q}$	Comments
1	↓	1	0	SET
0	↓	0	1	RESET

(b) Negative-edge triggered

Flip-flops

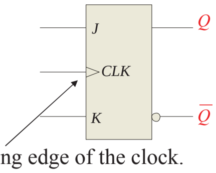
The J-K flip-flop is more versatile than the D flip flop. In addition to the clock input, it has two inputs, labeled *J* and *K*. When both *J* and *K* = 1, the output changes states (toggles) on the active clock edge (in this case, the rising edge).

Inputs			Outputs		
J	*K*	CLK	*Q*	$\overline{Q}$	Comments
0	0	↑	Q_0	$\overline{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\overline{Q}_0$	Q_0	Toggle

Flip-flops

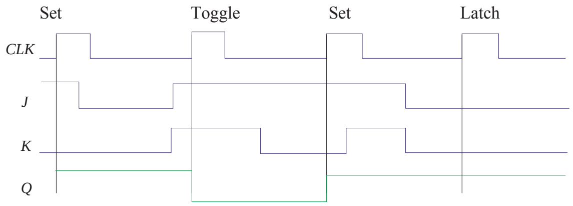
Example

Determine the Q output for the J-K flip-flop, given the inputs shown.



Notice that the outputs change on the leading edge of the clock.

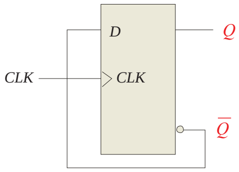
Solution



Flip-flops

A D-flip-flop does not have a toggle mode like the J-K flip-flop, but you can hardwire a toggle mode by connecting $\overline{Q}$ back to D as shown. This is useful in some counters as you will see in Chapter 8.

For example, if Q is LOW, $\overline{Q}$ is HIGH and the flip-flop will toggle on the next clock edge. Because the flip-flop only changes on the active edge, the output will only change once for each clock pulse.

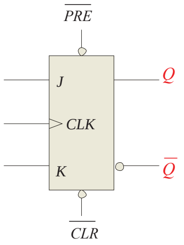


D flip-flop hardwired for a toggle mode

Flip-flops

Synchronous inputs are transferred in the triggering edge of the clock (for example the D or J-K inputs). Most flip-flops have other inputs that are *asynchronous*, meaning they affect the output independent of the clock.

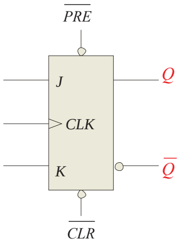
Two such inputs are normally labeled preset (PRE) and clear (CLR). These inputs are usually active LOW. A J-K flip flop with active LOW preset and CLR is shown.



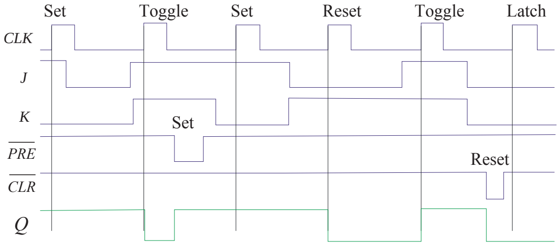
Flip-flops

Example

Determine the Q output for the J-K flip-flop, given the inputs shown.

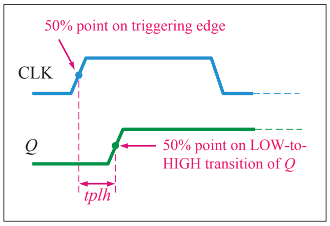


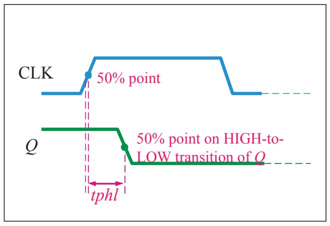
Solution



Flip-flop Characteristics

Propagation delay time is specified for the rising and falling outputs. It is measured between the 50% level of the clock to the 50% level of the output transition.

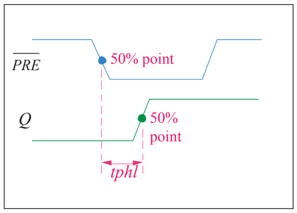

50% point on triggering edge
CLK
Q
50% point on LOW-to-HIGH transition of Q
tplh

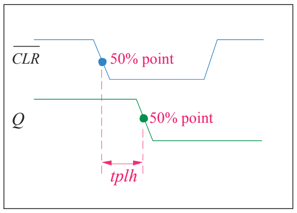


The typical propagation delay time for the 74AHC family (CMOS) is 4 ns. Even faster logic is available for specialized applications.

Flip-flop Characteristics

Another **propagation delay time** specification is the time required for an *asynchronous* input to cause a change in the output. Again it is measured from the 50% levels. The 74AHC family has specified delay times under 5 ns.

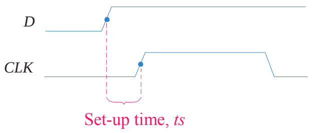




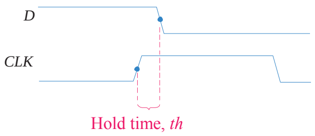
Flip-flop Characteristics

Set-up time and **hold time** are times required before and after the clock transition that data must be present to be reliably clocked into the flip-flop.

Setup time is the minimum time for the data to be present *before* the clock.



Hold time is the minimum time for the data to *remain* after the clock.



Flip-flop Characteristics

Other specifications include maximum clock frequency, minimum pulse widths for various inputs, and power dissipation. The power dissipation is the product of the supply voltage and the average current required.

A useful comparison between logic families is the **speed-power product** which uses two of the specifications discussed: the average propagation delay and the average power dissipation. The unit is energy.

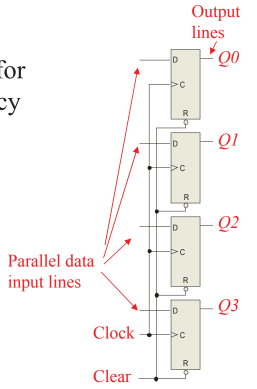
Example What is the speed-power product for 74AHC74A? Use the data from Table 9-5 to determine the answer.

Solution From Table 9-5, the average propagation delay is 4.6 ns. The quiescent power dissipated is 1.1 mW. Therefore, the speed-power product is 5 pJ

Flip-flop Applications

Principal flip-flop applications are for temporary data storage, as frequency dividers, and in counters (later).

Typically, for **data storage** applications, a group of flip-flops are connected to parallel data lines and clocked together. Data is stored until the next clock pulse.

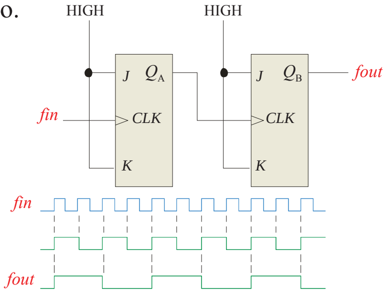


Flip-flop Applications

For **frequency division**, it is simple to use a flip-flop in the toggle mode or to chain a series of toggle flip flops to continue to divide by two.

One flip-flop will divide f_{in} by 2, two flip-flops will divide f_{in} by 4 (and so on). A side benefit of frequency division is that the output has an exact 50% duty cycle.

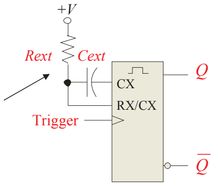
Waveforms:

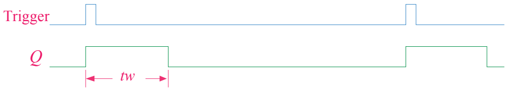


One-Shots

The **one-shot** or **monostable** multivibrator is a device with only one stable state. When triggered, it goes to its unstable state for a predetermined length of time, then returns to its stable state.

For most one-shots, the length of time in the unstable state (t_W) is determined by an external *RC* circuit.



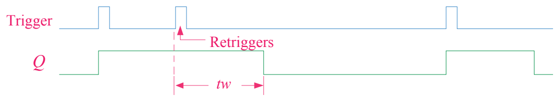


One-Shots

Non-retriggerable one-shots do not respond to any triggers that occur during the unstable state.

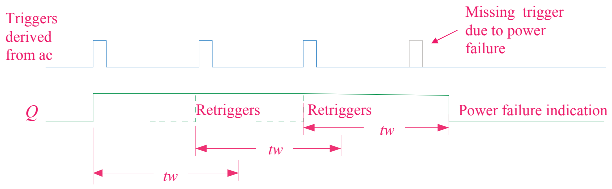
Retriggerable one-shots respond to any trigger, even if it occurs in the unstable state. If it occurs during the unstable state, the state is extended by an amount equal to the pulse width.

Retriggerable one-shot:

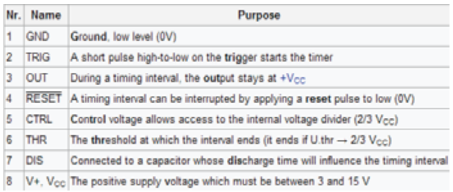


One-Shots

An application for a retriggerable one-shot is a power failure detection circuit. Triggers are derived from the ac power source, and continue to retrigger the one shot. In the event of a power failure, the one-shot is not triggered and an alarm can be initiated.



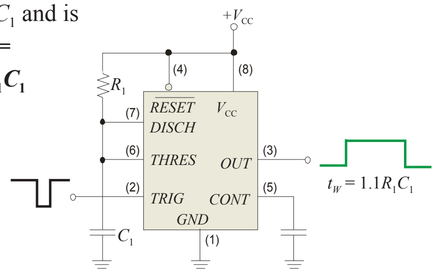
The 555 timer



Nr.	Name	Purpose
1	GND	**Ground**, low level (0V)
2	TRIG	A short pulse high-to-low on the **trigger** starts the timer
3	OUT	During a timing interval, the **output** stays at $+V_{CC}$
4	$\overline{RESET}$	A timing interval can be interrupted by applying a **reset** pulse to low (0V)
5	CTRL	**Control** voltage allows access to the internal voltage divider (2/3 V_{CC})
6	THR	The **threshold** at which the interval ends (it ends if U.thr → 2/3 V_{CC})
7	DIS	Connected to a capacitor whose **discharge** time will influence the timing interval
8	V+, V_{CC}	The positive supply voltage which must be between 3 and 15 V

The 555 timer can be configured in various ways, including as a one-shot. A basic one shot is shown. The pulse width is determined by R_1C_1 and is approximately t_W = $\mathbf{ln(3)}\boldsymbol{R_1C_1} \approx \mathbf{1.1}\boldsymbol{R_1C_1}$

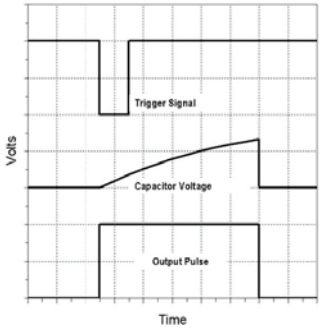
The trigger is a negative-going pulse.

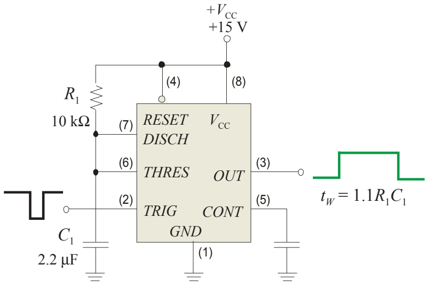


The 555 timer

Example Determine the pulse width for the circuit shown.

Solution $t_W = 1.1R_1C_1 = 1.1(10\ \text{k}\Omega)(2.2\ \mu\text{F}) =$ 24.2 ms

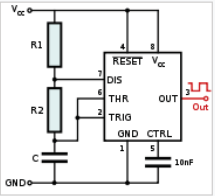




The 555 timer

The 555 can be configured as a basic astable, multivibrator with the circuit shown. In this circuit C_1 charges through R_1 and R_2 and discharges through only R_2. The output frequency is given by:

$$f = \frac{1.44}{(R_1 + 2R_2)C_1}$$



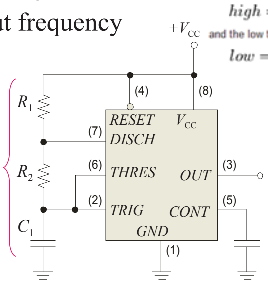
The high time from each pulse is given by

$$high = \ln(2) \cdot (R1 + R2) \cdot C$$

and the low time from each pulse is given by

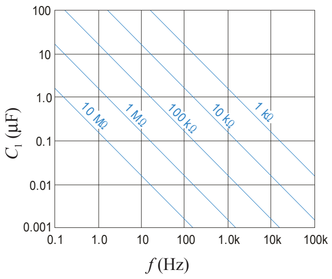
$$low = \ln(2) \cdot R2 \cdot C$$

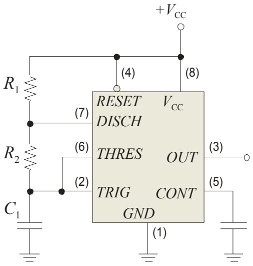
The frequency and duty cycle are set by these components.



The 555 timer

Given the components, you can read the frequency from the chart. Alternatively, you can use the chart to pick components for a desired frequency.





Selected Key Terms

Latch A bistable digital circuit used for storing a bit.

Bistable Having two stable states. Latches and flip-flops are bistable multivibrators.

Clock A triggering input of a flip-flop.

D flip-flop A type of bistable multivibrator in which the output assumes the state of the *D* input on the triggering edge of a clock pulse.

J-K flip-flop A type of flip-flop that can operate in the SET, RESET, no-change, and toggle modes.

Selected Key Terms

Propagation delay time The interval of time required after an input signal has been applied for the resulting output signal to change.

Set-up time The time interval required for the input levels to be on a digital circuit.

Hold time The time interval required for the input levels to remain steady to a flip-flop after the triggering edge in order to reliably activate the device.

Timer A circuit that can be used as a one-shot or as an oscillator.

Registered A CPLD macrocell output configuration where the output comes from a flip-flop.

1. The output of a D latch will not change if

a. the output is LOW

b. Enable is not active

c. D is LOW

d. all of the above

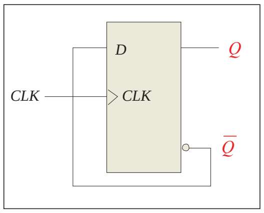
2. The D flip-flop shown will

a. set on the next clock pulse

b. reset on the next clock pulse

c. latch on the next clock pulse

d. toggle on the next clock pulse



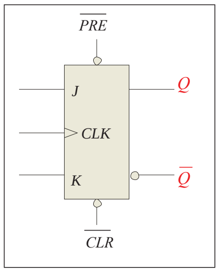
3. For the J-K flip-flop shown, the number of inputs that are asynchronous is

a. 1

b. 2

c. 3

d. 4



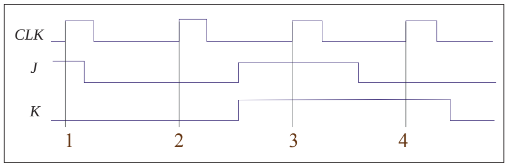
4. Assume the output is initially HIGH on a leading edge triggered J-K flip flop. For the inputs shown, the output will go from HIGH to LOW on which clock pulse?

a. 1

b. 2

c. 3

d. 4



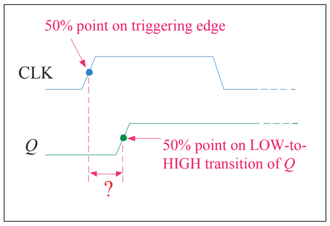
5. The time interval illustrated is called

a. *t*PHL

b. *t*PLH

c. set-up time

d. hold time


50% point on triggering edge
CLK
Q
50% point on LOW-to-HIGH transition of Q
?

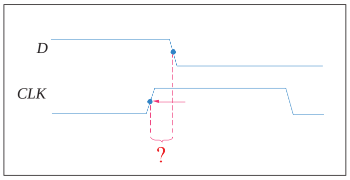
6. The time interval illustrated is called

a. *t*PHL

b. *t*PLH

c. set-up time

d. hold time



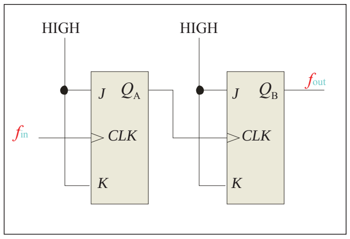
7. The application illustrated is a

a. astable multivibrator

b. data storage device

c. frequency multiplier

d. frequency divider



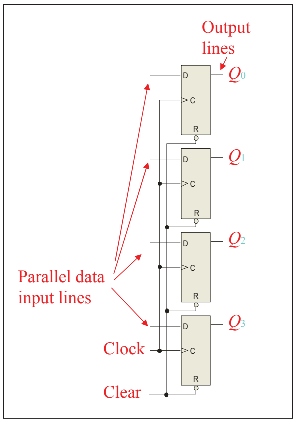
8. The application illustrated is a

a. astable multivibrator

b. data storage device

c. frequency multiplier

d. frequency divider



9. A retriggerable one-shot with an active HIGH output has a pulse width of 20 ms and is triggered from a 60 Hz line. The output will be a

a. series of 16.7 ms pulses

b. series of 20 ms pulses

c. constant LOW

d. constant HIGH

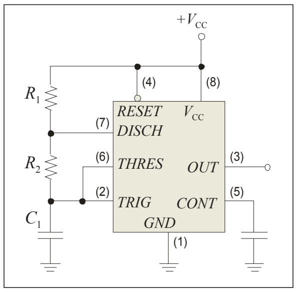
10. The circuit illustrated is a

a. astable multivibrator

b. monostable multivibrator

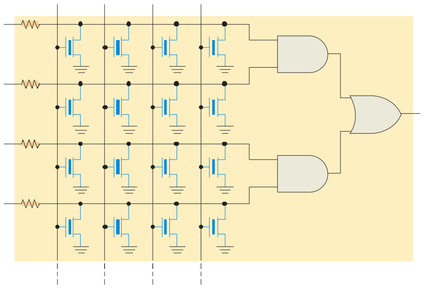
c. frequency multiplier

d. frequency divider



Digital Electronics

Counters



Counting in Binary

As you know, the binary count sequence follows a familiar pattern of 0's and 1's as described in Section 2-2 of the text.

0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

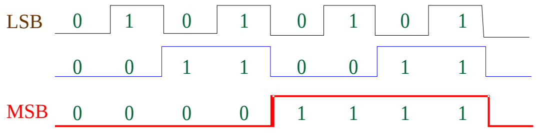
LSB changes on every number.

The next bit changes on every other number.

The next bit changes on every fourth number.

Counting in Binary

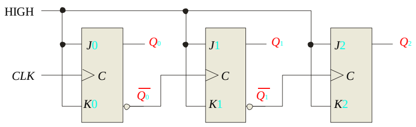
A counter can form the same pattern of 0's and 1's with logic levels. The first stage in the counter represents the least significant bit – notice that these waveforms follow the same pattern as counting in binary.



Three bit Asynchronous Counter

In an asynchronous counter, the clock is applied only to the first stage. Subsequent stages derive the clock from the previous stage.

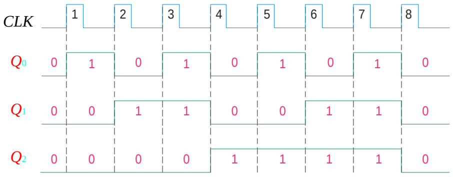
The three-bit asynchronous counter shown is typical. It uses J-K flip-flops in the toggle mode.



Waveforms are on the following slide…

Three bit Asynchronous Counter

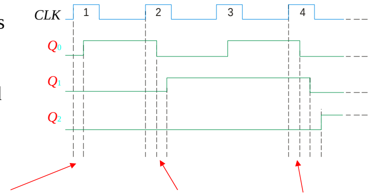
Notice that the Q_0 output is triggered on the leading edge of the clock signal. The following stage is triggered from Q_0. The leading edge of Q_0 is equivalent to the trailing edge of Q_0. The resulting sequence is that of an 3-bit binary up counter.



Propagation Delay

Asynchronous counters are sometimes called **ripple** counters, because the stages do not all change together. For certain applications requiring high clock rates, this is a major disadvantage.

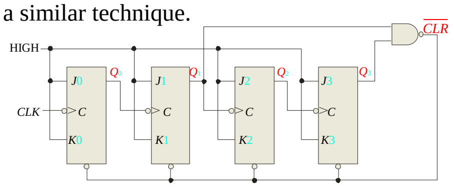
Notice how delays are cumulative as each stage in a counter is clocked later than the previous stage.



Q_0 is delayed by 1 propagation delay, Q_2 by 2 delays and Q_3 by 3 delays.

Asynchronous Decade Counter

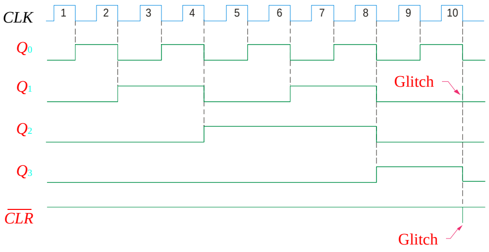
This counter uses partial decoding to recycle the count sequence to zero after the 1001 state. The flip-flops are trailing-edge triggered, so clocks are derived from the Q outputs. Other truncated sequences can be obtained using a similar technique.



Waveforms are on the following slide…

Asynchronous Decade Counter

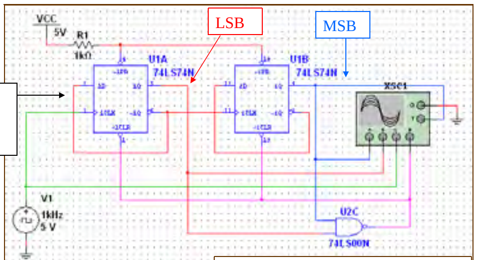
When Q_1 and Q_3 are HIGH together, the counter is cleared by a "glitch" on the $\overline{CLR}$ line.



Asynchronous Counter Using D Flip-flops

D flip-flops can be set to toggle and used as asynchronous counters by connecting $\overline{Q}$ back to D. The counter in this slide is a Multisim simulation of one described in the lab manual. Can you figure out the sequence?

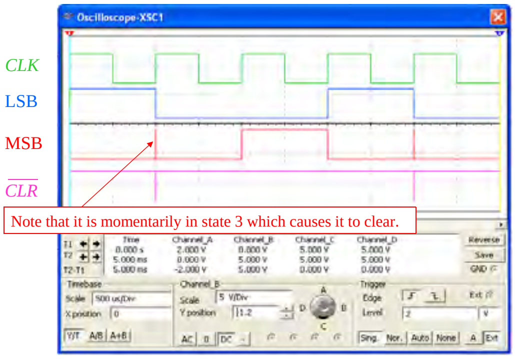
$\overline{Q}$ to D puts D flip-flop in toggle mode



LSB

MSB

The next slide shows the scope…



Note that it is momentarily in state 3 which causes it to clear.

	Time	Channel_A	Channel_B	Channel_C	Channel_D
T1	0.000 s	2.000 V	0.000 V	5.000 V	5.000 V
T2	5.000 ms	0.000 V	5.000 V	5.000 V	5.000 V
T2-T1	5.000 ms	-2.000 V	5.000 V	0.000 V	0.000 V

Timebase: Scale 500 us/Div; X position 0

Channel_B: Scale 5 V/Div; Y position 1.2

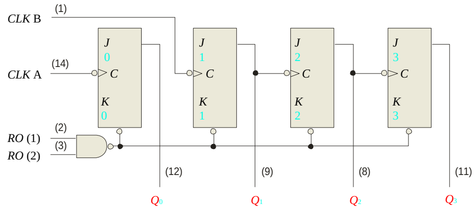
Trigger: Level 2 V

The sequence is 0 – 2 – 1 – ($\overline{CLR}$) (repeat)…

The 74LS93A Asynchronous Counter

The 74LS93A has one independent toggle J-K flip-flop driven by *CLK* A and three toggle J-K flip-flops that form an asynchronous counter driven by *CLK* B.

The counter can be extended to form a 4-bit counter by connecting Q_0 to the *CLK* B input. Two inputs are provided that clear the count.

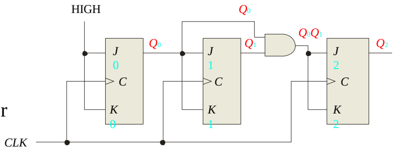


All J and K inputs are connected internally HIGH

Synchronous Counters

In a **synchronous counter** all flip-flops are clocked together with a common clock pulse. Synchronous counters overcome the disadvantage of accumulated propagation delays, but generally they require more circuitry to control states changes.

This 3-bit binary synchronous counter has the same count sequence as the 3-bit asynchronous counter shown previously.



The next slide shows how to analyze this counter by writing the logic equations for each input. Notice the inputs to each flip-flop…

Analysis of Synchronous Counters

A tabular technique for analysis is illustrated for the counter on the previous slide. Start by setting up the outputs as shown, then write the logic equation for each input. This has been done for the counter.

1. Put the counter in an arbitrary state; then determine the inputs for this state.

2. Use the new inputs to determine the next state: Q_2 and Q_1 will latch and Q_0 will toggle.

3. Set up the next group of inputs from the current output.

Outputs | Logic for inputs

$Q_2\,Q_1\,Q_0$	$J_2 = Q_0Q_1$	$K_2 = Q_0Q_1$	$J_1 = Q_0$	$K_1 = Q_0$	$J_0 = 1$	$K_0 = 1$
0 0 0	0	0	0	0	1	1
0 0 1	0	0	1	1	1	1
0 1 0						

4. Q_2 will latch again but both Q_1 and Q_0 will toggle.

Continue like this, to complete the table. The next slide shows the completed table…

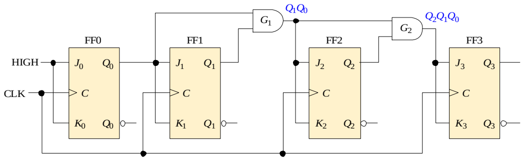
Analysis of Synchronous Counters

Outputs | Logic for inputs

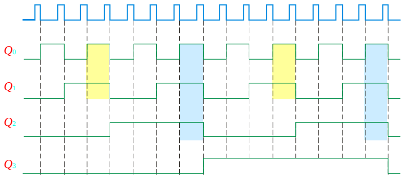
$Q_2\,Q_1\,Q_0$	$J_2 = Q_0Q_1$	$K_2 = Q_0Q_1$	$J_1 = Q_0$	$K_1 = Q_0$	$J_0 = 1$	$K_0 = 1$
0 0 0	0	0	0	0	1	1
0 0 1	0	0	1	1	1	1
0 1 0	0	0	0	0	1	1
0 1 1	1	1	1	1	1	1
1 0 0	0	0	0	0	1	1
1 0 1	0	0	1	1	1	1
1 1 0	0	0	0	0	1	1
1 1 1	1					
0 0 0						

At this points all states have been accounted for and the counter is ready to recycle…

A 4-bit Synchronous Binary Counter



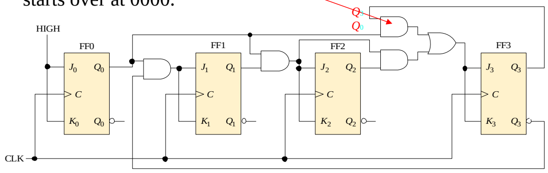
The 4-bit binary counter has one more AND gate than the 3-bit counter just described. The shaded areas show where the AND gate outputs are HIGH causing the next FF to toggle.



BCD Decade Counter

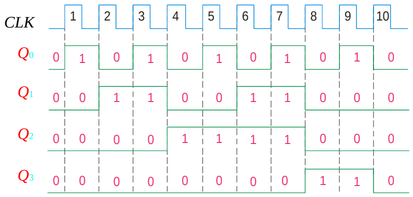
With some additional logic, a binary counter can be converted to a BCD synchronous decade counter. After reaching the count 1001, the counter recycles to 0000.

This gate detects 1001, and causes FF3 to toggle on the next clock pulse. FF0 toggles on every clock pulse. Thus, the count starts over at 0000.



BCD Decade Counter

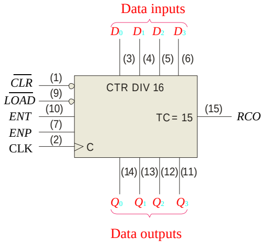
Waveforms for the decade counter:



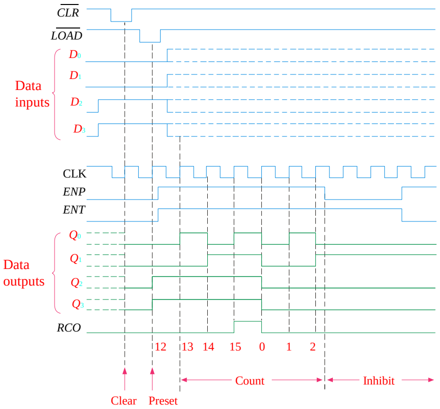
These same waveforms can be obtained with an asynchronous counter in IC form – the 74LS90. It is available in a dual version – the 74LS390, which can be cascaded. It is slower than synchronous counters (max count frequency is 35 MHz), but is simpler.

A 4-bit Synchronous Binary Counter

The 74LS163 is a 4-bit IC synchronous counter with additional features over a basic counter. It has parallel load, a $\overline{CLR}$ input, two chip enables, and a ripple count output that signals when the count has reached the terminal count.

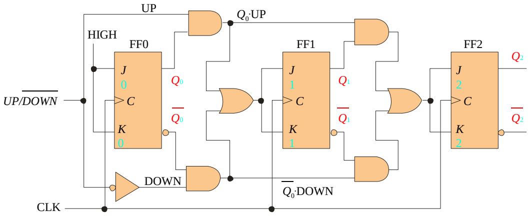


Example waveforms are on the next slide…



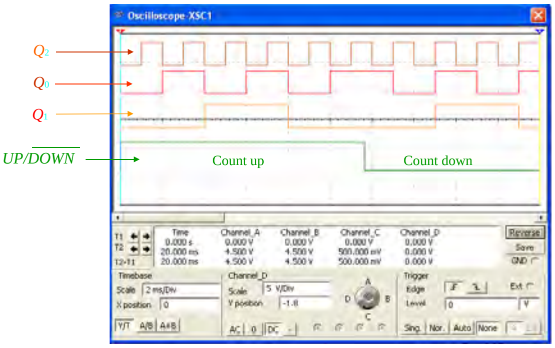
Up/Down Synchronous Counters

An up/down counter is capable of progressing in either direction depending on a control input.



Example waveforms from Multisim are on the next slide…

Up/Down Synchronous Counters



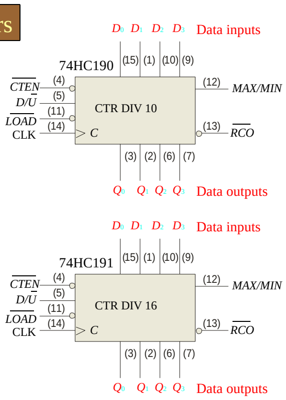
	Time	Channel_A	Channel_B	Channel_C	Channel_D
T1	0.000 s	0.000 V	0.000 V	0.000 V	0.000 V
T2	20.000 ms	4.500 V	4.500 V	500.000 mV	0.000 V
T2-T1	20.000 ms	4.500 V	4.500 V	500.000 mV	0.000 V

Timebase Scale: 2 ms/Div
X position: 0
Channel_D Scale: 5 V/Div
Y position: -1.8
Trigger Level: 0 V

Up/Down Synchronous Counters

The 74HC190 is a high speed CMOS synchronous up/down decade counter with parallel load capability. It also has a active LOW ripple clock output ($\overline{RCO}$) and a *MAX/MIN* output when the terminal count is reached.

The 74HC191 has the same inputs and outputs but is a synchronous up/down binary counter.

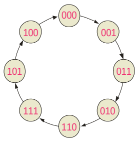


Synchronous Counter Design

Most requirements for synchronous counters can be met with available ICs. In cases where a special sequence is needed, you can apply a step-by-step design process.

The steps in design are described in detail in the text and lab manual. Start with the desired sequence and draw a state diagram and next-state table. The gray code sequence from the text is illustrated:

State diagram:



Next state table:

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

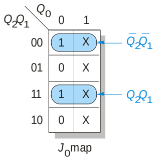
Synchronous Counter Design

The J-K transition table lists all combinations of present output (Q_N) and next output (Q_{N+1}) on the left. The inputs that produce that transition are listed on the right.

Each time a flip-flop is clocked, the *J* and *K* inputs required for that transition are mapped onto a K-map.

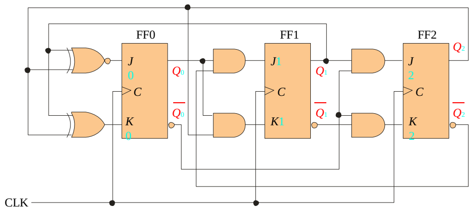
Output Transitions		Flip-Flop Inputs	
Q_N	Q_{N+1}	J	K
0 ⟶	0	0	X
0 ⟶	1	1	X
1 ⟶	0	X	1
1 ⟶	1	X	0

An example of the J_0 map is:

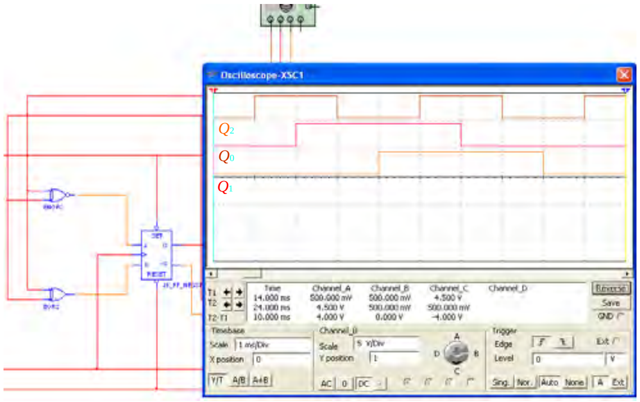


The logic for each input is read and the circuit is constructed. The next slide shows the circuit for the gray code counter…

Synchronous Counter Design



The circuit can be checked with Multisim before constructing it. The next slide shows the Multisim result…

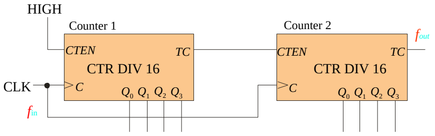


	Time	Channel_A	Channel_B	Channel_C	Channel_D
T1	14.000 ms	500.000 mV	500.000 mV	4.500 V	
T2	24.000 ms	4.500 V	500.000 mV	500.000 mV	
T2-T1	10.000 ms	4.000 V	0.000 V	-4.000 V	

Timebase: Scale 1 ms/Div; X position 0
Channel_D: Scale 5 V/Div; Y position 1
Trigger: Level 0 V

Cascaded counters

Cascading is a method of achieving higher-modulus counters. For synchronous IC counters, the next counter is enabled only when the terminal count of the previous stage is reached.



Example

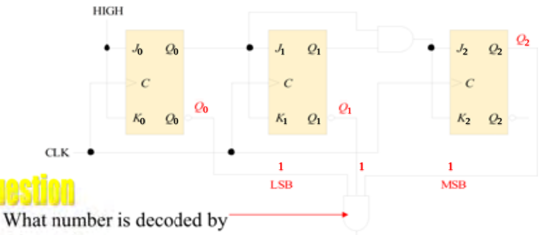
a) What is the modulus of the cascaded DIV 16 counters?
b) If f_{in} =100 kHz, what is f_{out}?

Solution

a) Each counter divides the frequency by 16. Thus the modulus is 16^2 = 256.
b) The output frequency is 100 kHz/256 = 391 Hz

Counter Decoding

Decoding is the detection of a binary number and can be done with an AND gate.



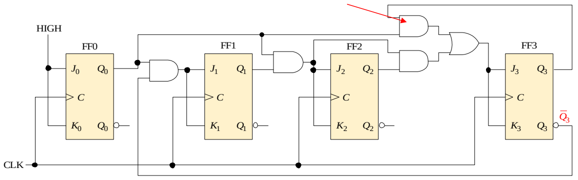
Question

What number is decoded by this gate?

Partial Decoding

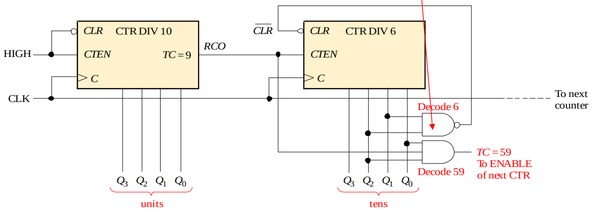
The decade counter shown previously incorporates *partial decoding* (looking at only the MSB and the LSB) to detect 1001. This was possible because this is the first occurrence of this combination in the sequence.

Detects 1001 by looking only at two bits



Resetting the Count with a Decoder

The divide-by-60 counter in the text also uses partial decoding to clear the tens count when a 6 was detected.

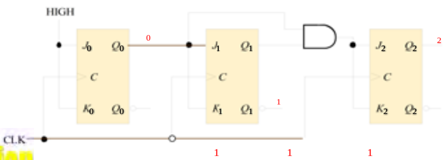


The divide characteristic illustrated here is a good way to obtain a lower frequency using a counter. For example, the 60 Hz power line can be converted to 1 Hz.

Counter Decoding

Example

Show how to decode state 5 with an active LOW output.

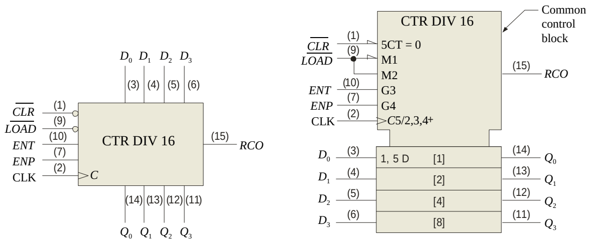


Solution

Notice that a NAND gate was used to give the active LOW output.

Logic Symbols

Dependency notation allows the logical operation of a device to be determined from its logic symbol.



Selected Key Terms

Asynchronous Not occurring at the same time.

Modulus The number of unique states through which a counter will sequence.

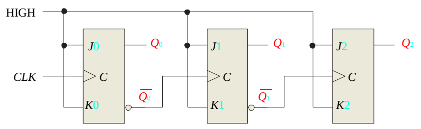
Synchronous Occurring at the same time.

Terminal count The final state in a counter’s sequence.

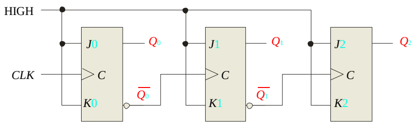
State machine A logic system exhibiting a sequence of states or values.

Cascade To connect “end-to-end” as when several counters are connected from the terminal count output of one to the enable input of the next counter.

1. The counter shown below is an example of
   a. an asynchronous counter
   b. a BCD counter
   c. a synchronous counter
   d. none of the above



2. The *Q*0 output of the counter shown
   a. is present before *Q*1 or *Q*2
   b. changes on every clock pulse
   c. has a higher frequency than *Q*1 or *Q*2
   d. all of the above



3. To cause a D flip-flop to toggle, connect the
   a. clock to the *D* input
   b. *Q* output to the *D* input
   c. *Q* output to the *D* input
   d. clock to the preset input

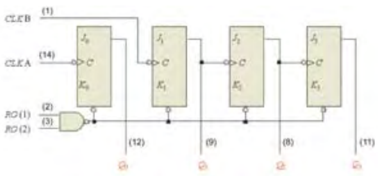
4. The 7493A asynchronous counter diagram is shown (*J*'s and *K*'s are HIGH.) To make the count have a modulus of 16, connect

a. *Q*0 to RO(1) and RO(2) to

b. *Q*3 to RO(1) and RO(2)

c. *CLK* A and *CLK* B together

d. *Q*0 to *CLK* B



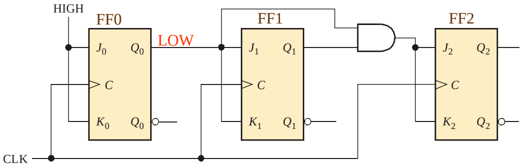
5. Assume *Q*0 is LOW. The next clock pulse will cause

a. FF1 and FF2 to both toggle

b. FF1 and FF2 to both latch

c. FF1 to latch; FF2 to toggle

d. FF1 to toggle; FF2 to latch



6. A 4-bit binary counter has a terminal count of

a. 4

b. 10

c. 15

d. 16

7. Assume the clock for a 4-bit binary counter is 80 kHz. The output frequency of the fourth stage (*Q*3) is

a. 5 kHz

b. 10 kHz

c. 20 kHz

d. 320 kHz

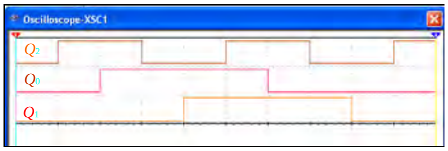
8. A 3-bit count sequence is shown for a counter (*Q*2 is the MSB). The sequence is

a. 0-1-2-3-4-5-6-7-0 (repeat)

b. 0-1-3-2-6-7-5-4-0 (repeat)

c. 0-2-4-6-1-3-5-7-0 (repeat)

d. 0-4-6-2-3-7-5-1-0 (repeat)



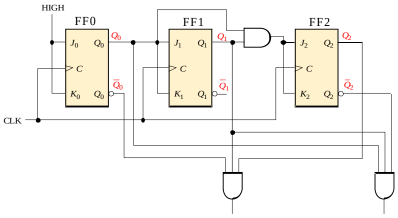
9. FF2 represents the MSB. The counts that are being decoded by the 3-input AND gates are

a. 2 and 3

b. 3 and 6

c. 2 and 5

d. 5 and 6



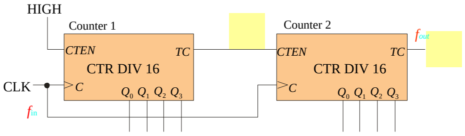
10. Assume the input frequency (*fin*) is 256 Hz. The output frequency (*fout*) will be

a. 16 Hz

b. 1 kHz

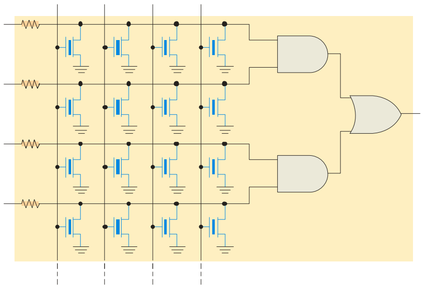
c. 65 kHz

d. none of the above



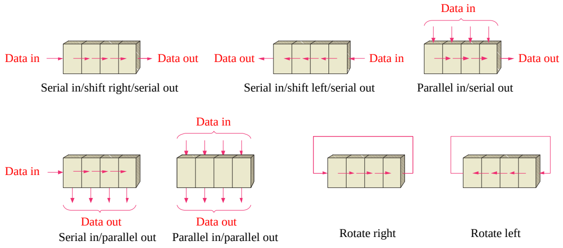
Digital Electronics

Shift Registers



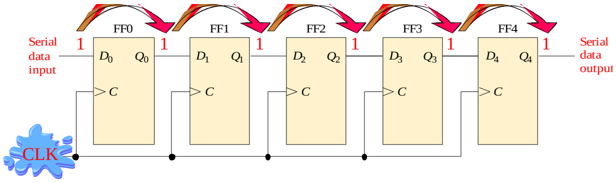
Basic Shift Register Operations

A shift register is an arrangement of flip-flops with important applications in storage and movement of data. Some basic data movements are illustrated here.



Serial-in/Serial out Shift Register

Shift registers are available in IC form or can be constructed from discrete flip-flops as is shown here with a five-bit serial-in serial-out register.
Each clock pulse will move an input bit to the next flip-flop. For example, a 1 is shown as it moves across.

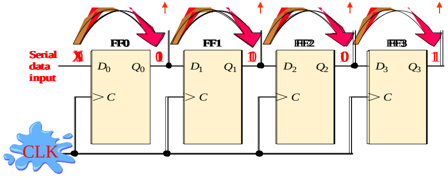


A Basic Application

An application of shift registers is conversion of serial data to parallel form.

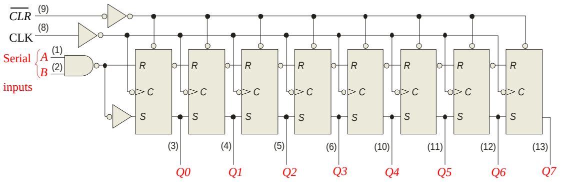
For example, assume the binary number 1011 is loaded sequentially, one bit at each clock pulse.

After 4 clock pulses, the data is available at the parallel output.



The 74HC164A Shift Register

The 74HC164A is a CMOS 8-bit serial in/parallel out shift register. V_{CC} can be from +2.0 V to +6.0 V.

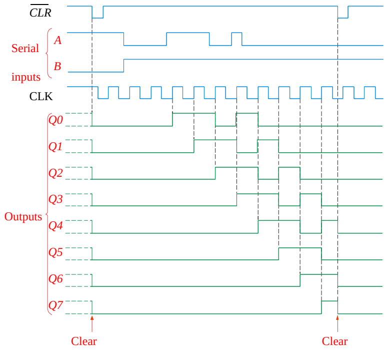


One of the two serial data inputs may be used as an active HIGH enable to gate the other input. If no enable is needed, the other serial input can be connected to V_{CC}. The 74HC164A has an active LOW asynchronous clear. Data is entered on the leading-edge of the clock.

Waveforms for the 74HC164A

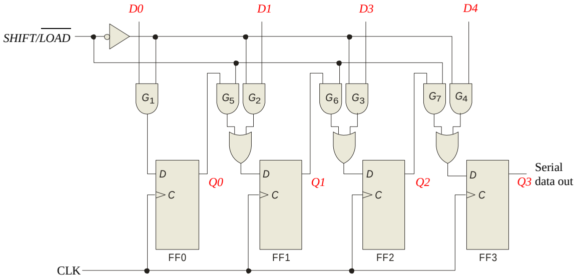
Sample waveforms for the 74HC164A are shown. Notice that *B* acts as an active HIGH enable for the data on *A* as discussed.

As with CMOS devices, unused inputs should *always* be connected to a logic level; unused outputs should be left open.



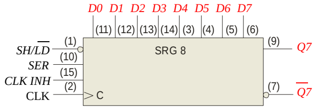
Parallel in/Serial out Shift Register

Shift registers can be used to convert parallel data to serial form. A logic diagram for this type of register is shown:



The 74HC165 Shift Register

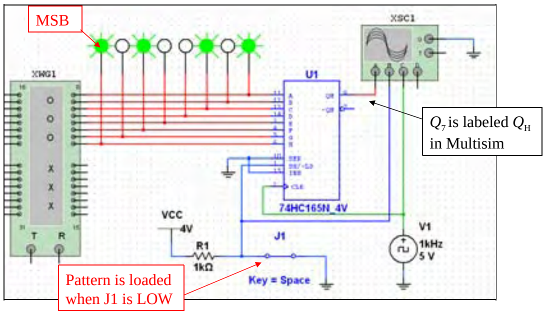
The 74HC165 is a CMOS 8-bit parallel in/serial out shift register. The logic symbol is shown:



The clock (*CLK*) and clock inhibit (*CLK INH*) lines are connected to a common OR gate, so either of these inputs can be used as an active-LOW clock enable with the other as the clock input. Data is loaded *asynchronously* when *SH/$\overline{LD}$* is LOW and moved through the register *synchronously* when *SH/LD* is HIGH and a rising clock pulse occurs.

The 74HC165 Shift Register

A Multisim simulation of the 74165A is shown. The word generator is used as a source for the pattern shown in the green probes.



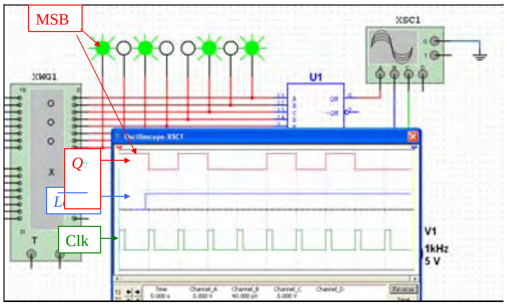
MSB

Q_7 is labeled Q_H in Multisim

Pattern is loaded when J1 is LOW

The 74HC165 Shift Register

Here the scope is opened and you can observe the pattern. The MSB is HIGH and is on the Q_7 output as soon as LOAD is LOW.



MSB

Q_7

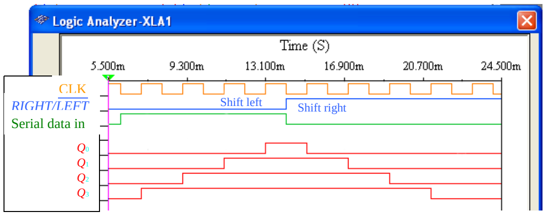
L

Clk

Bidirectional Shift Register

Bidirectional shift registers can shift the data in either direction using a *RIGHT/LEFT* input.

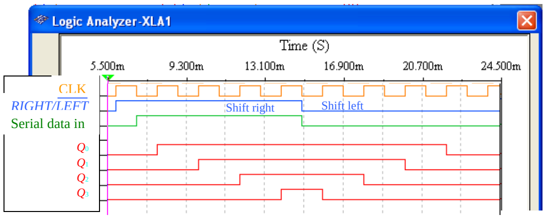
The logic analyzer simulation shows a bidirectional shift register such as the one shown in Figure 9-19 of the text. Notice the HIGH level from the Serial data in is shifted at first from Q_3 toward Q_0.



Bidirectional Shift Register

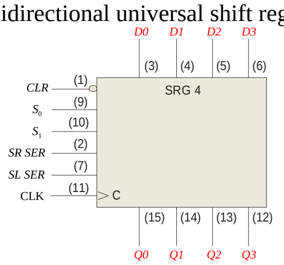
Question How will the pattern change if the *RIGHT/$\overline{LEFT}$* control signal is inverted?

Answer See display



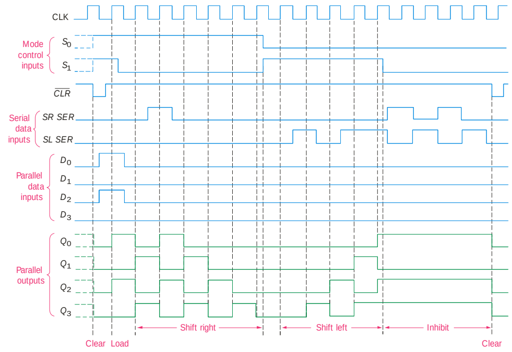
Universal Shift Register

A universal shift register has both serial and parallel input and output capability. The 74HC194 is an example of a 4-bit bidirectional universal shift register.



Sample waveforms are on the following slide…

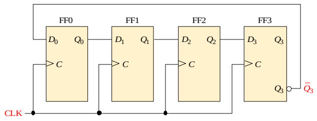
Universal Shift Register



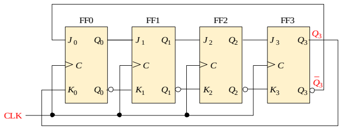
Shift Register Counters

Shift registers can form useful counters by recirculating a pattern of 0's and 1's. Two important shift register counters are the *Johnson counter* and the *ring counter*.

The Johnson counter can be made with a series of D flip-flops



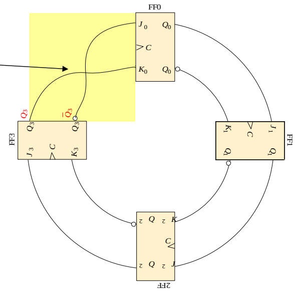
… or with a series of J-K flip flops. Here Q_3 and $\overline{Q}_3$ are fed back to the *J* and *K* inputs with a "twist".



Johnson Counter

Redrawing the same Johnson counter (without the clock shown) illustrates why it is sometimes called as a "twisted-ring" counter.

"twist"



Johnson Counter

The Johnson counter is useful when you need a sequence that changes by only one bit at a time but it has a limited number of states (2*n*, where *n* = number of stages).

The first five counts for a 4-bit Johnson counter that is initially cleared are:

CLK	*Q0*	*Q1*	*Q2*	*Q3*
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1

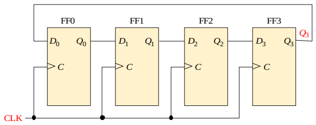
Question

What are the remaining 3 states?

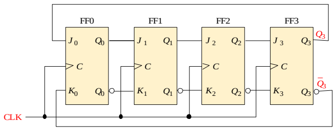
Ring Counter

The ring counter can also be implemented with either D flip-flops or J-K flip-flops.

Here is a 4-bit ring counter constructed from a series of D flip-flops. Notice the feedback.



Like the Johnson counter, it can also be implemented with J-K flip flops.

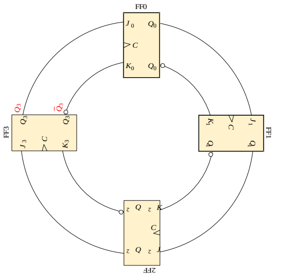


Ring Counter

Redrawing the Ring counter (without the clock shown) shows why it is a “ring”.

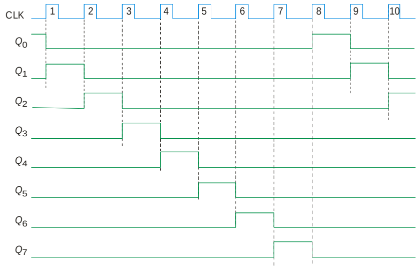
The disadvantage to this counter is that it must be preloaded with the desired pattern (usually a single 0 or 1) and it has even fewer states than a Johnson counter (*n*, where *n* = number of flip-flops.

On the other hand, it has the advantage of being self-decoding with a unique output for each state.



Ring Counter

A common pattern for a ring counter is to load it with a single 1 or a single 0. The waveforms shown here are for an 8-bit ring counter with a single 1.



Shift Register Applications

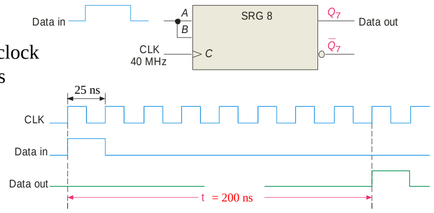
Shift registers can be used to delay a digital signal by a predetermined amount.

Example An 8-bit serial in/serial out shift register has a 40 MHz clock. What is the total delay through the register?

Solution

The delay for each clock is 1/40 MHz = 25 ns

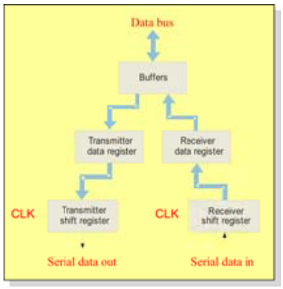
The total delay is 8 x 25 ns = 200 ns

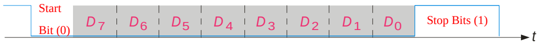


Shift Register Applications

A UART (Universal Asynchronous Receiver Transmitter) is a serial-to-parallel converter and a parallel to serial converter.

UARTs are commonly used in small systems where one device must communicate with another. Parallel data is converted to asynchronous serial form and transmitted. The serial data format is:

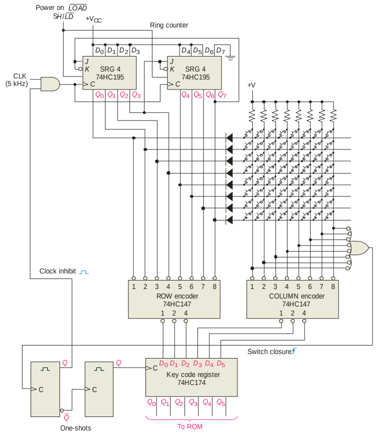




Keyboard Encoder

The keyboard encoder is an example of where a ring counter is used in a small system to encode a key press.

Two 74HC195 shift registers are connected as an 8-bit ring counter preloaded with a single 0. As the 0 circulate in the ring counter, it “scans” the keyboard looking for any row that has a key closure. When one is found, a corresponding column line is connected to that row line. The combination of the unique column and row lines identifies the key. The schematic is shown on the following slide…



Key Terms

Register One or more flip-flops used to store and shift data.

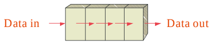
Stage One storage element in a register.

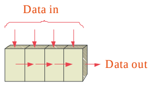
Shift To move binary data from stage to stage within a shift register or other storage device or to move binary data into or out of the device.

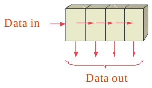
Load To enter data in a shift register.

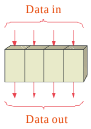
Bidirectional Having two directions. In a bidirectional shift register, the stored data can be shifted right or left.

1. The shift register that would be used to delay serial data by 4 clock periods is

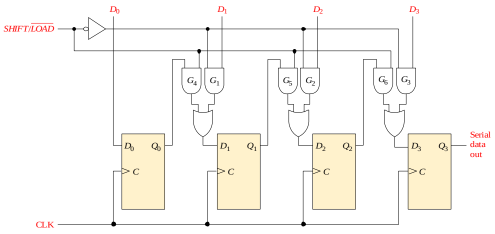
a. 

c. 

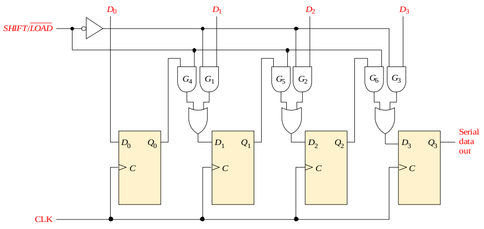
b. 

d. 

2. The circuit shown is a
   a. serial-in/serial-out shift register
   b. serial-in/parallel-out shift register
   c. parallel-in/serial-out shift register
   d. parallel-in/parallel-out shift register

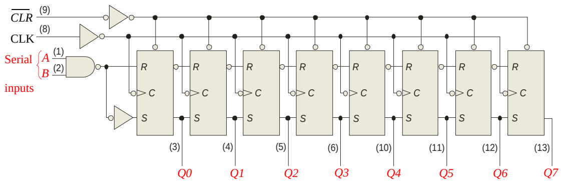


3. If the *SHIFT/$\overline{LOAD}$* line is HIGH, data
   a. is loaded from *D*0, *D*1, *D*2 and *D*3 immediately
   b. is loaded from *D*0, *D*1, *D*2 and *D*3 on the next CLK
   c. shifted from left to right on the next CLK
   d. shifted from right to left on the next CLK



4. A 4-bit parallel-in/parallel-out shift register will store data for
   a. 1 clock period
   b. 2 clock periods
   c. 3 clock periods
   d. 4 clock periods

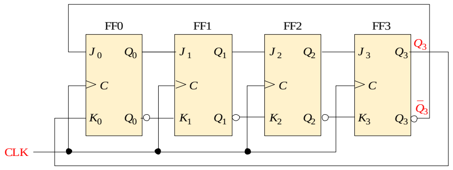
5. The 74HC164 (shown) has two serial inputs. If data is placed on the *A* input, the *B* input
   a. could serve as an active LOW enable
   b. could serve as an active HIGH enable
   c. should be connected to ground
   d. should be left open



6. An advantage of a ring counter over a Johnson counter is that the ring counter
   a. has more possible states for a given number of flip-flops
   b. is cleared after each cycle
   c. allows only one bit to change at a time
   d. is self-decoding

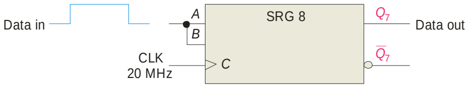
7. A possible sequence for a 4-bit ring counter is
   a. … 1111, 1110, 1101 …
   b. … 0000, 0001, 0010 …
   c. … 0001, 0011, 0111 …
   d. … 1000, 0100, 0010 …

8. The circuit shown is a
   a. serial-in/parallel-out shift register
   b. serial-in/serial-out shift register
   c. ring counter
   d. Johnson counter



9. Assume serial data is applied to the 8-bit shift register shown. The clock frequency is 20 MHz. The first data bit

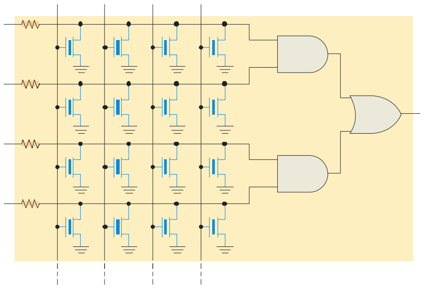
   will show up at the output in
   a. 50 ns
   b. 200 ns
   c. 400 ns
   d. 800 ns



10. For transmission, data from a UART is sent in
   a. asynchronous serial form
   b. synchronous parallel form
   c. can be either of the above
   d. none of the above

Digital Electronics

Memory & Storage



Memory Units

Memories store data in units from one to eight bits. The most common unit is the **byte**, which by definition is 8 bits.

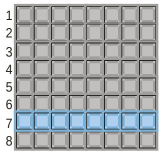
Computer memories are organized into multiples of bytes called words. Generally, a word is defined as the number of bits handled as one entity by a computer. By this definition, a word is equal to the internal register size (usually 16, 32, or 64 bits).

For historical reasons, assembly language defines a word as exactly two bytes. In assembly language, a 32 bit entity is called a double-word and 64 bits is defined as a quad-word.

Memory Units

The location of a unit of data in a memory is called the **address**. In PCs, a byte is the smallest unit of data that can be accessed.

In a 2-dimensional array, a byte is accessed by supplying a row number. For example the blue byte is located in row 7.



Memory Addressing

A 3-dimensional array is arranged as rows and columns. Each byte has a unique row and column address.

Question

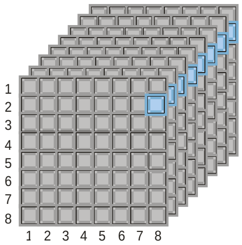
a) How many bytes are shown?

b) What is the location of the blue byte?

Answers

a) 64 B

b) Row 2, column 8

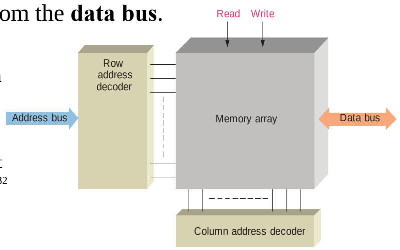


This example is (of course) only for illustration. Typical computer memories have 256 MB or more of capacity.

Memory Addressing

In order to read or write to a specific memory location, a binary code is placed on the **address bus**. Internal decoders decode the address to determine the specific location. Data is then moved to or from the **data bus**.

The address bus is a group of conductors with a common function. Its size determines the number of locations that can be accessed. A 32 bit address bus can access 2^{32} locations, which is approximately 4G.



Memory Addressing

In addition to the address bus and data bus, semiconductor memories have read and write control signals and chip select signals. Depending on the type of memory, other signals may be required.

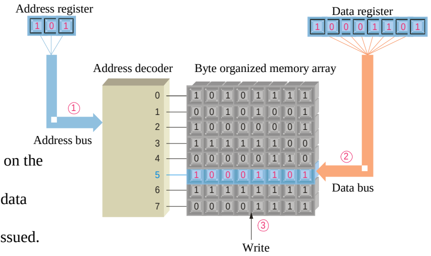
Read Enable ($\overline{RE}$) and **Write Enable** ($\overline{WE}$) signals are sent from the CPU to memory to control data transfer to or from memory.

Chip Select ($\overline{CS}$) or **Chip Enable** ($\overline{CE}$) is used as part of address decoding. All other inputs are ignored if the Chip Select is not active.

Output Enable ($\overline{OE}$) is active during a read operation, otherwise it is inactive. It connects the memory to the data bus.

Read and Write Operations

The two main memory operations are called **read** and **write**. A simplified write operation is shown in which new data overwrites the original data. Data moves *to* the memory.

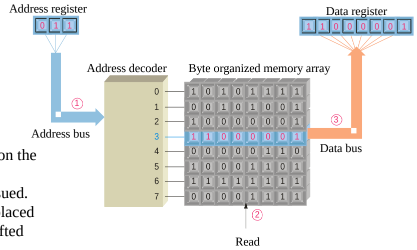


0	1	0	1	0	1	1	1	1
1	0	0	1	0	1	0	0	1
2	1	0	0	0	0	0	0	1
3	1	1	1	1	1	1	0	0
4	0	0	0	0	0	1	1	0
5	1	0	0	0	1	1	0	1
6	1	1	1	1	1	1	1	1
7	0	0	0	0	1	1	1	1

1. The address is placed on the address bus.
2. Data is placed on the data bus.
3. A write command is issued.

Read and Write Operations

The read operation is actually a “copy” operation, as the original data is not changed. The data bus is a “two-way” path; data moves *from* the memory during a read operation.

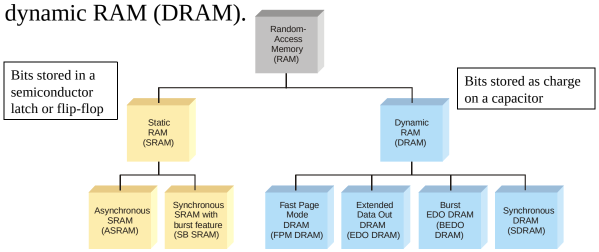


0	1	0	1	0	1	1	1	1
1	0	0	1	0	1	0	0	1
2	1	0	0	0	0	0	0	1
3	1	1	0	0	0	0	0	1
4	0	0	0	0	0	1	1	0
5	1	0	0	0	1	1	0	1
6	1	1	1	1	1	1	1	1
7	0	0	0	0	1	1	1	1

1. The address is placed on the address bus.
2. A read command is issued.
3. A copy of the data is placed in the data bus and shifted into the data register.

Random Access Memory

RAM is for temporary data storage. It is read/write memory and can store data only when power is applied, hence it is *volatile*. Two categories are static RAM (SRAM) and dynamic RAM (DRAM).


Random-Access Memory (RAM)

Bits stored in a semiconductor latch or flip-flop

Bits stored as charge on a capacitor

Static RAM (SRAM)

Dynamic RAM (DRAM)

Asynchronous SRAM (ASRAM)

Synchronous SRAM with burst feature (SB SRAM)

Fast Page Mode DRAM (FPM DRAM)

Extended Data Out DRAM (EDO DRAM)

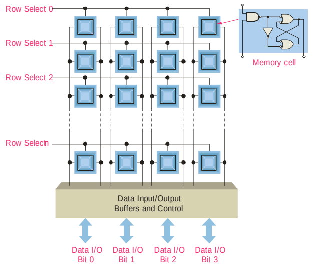
Burst EDO DRAM (BEDO DRAM)

Synchronous DRAM (SDRAM)

Static RAM

SRAM uses semiconductor latch memory cells. The cells are organized into an array of rows and columns.

SRAM is faster than DRAM but is more complex, takes up more space, and is more expensive. SRAMs are available in many configurations – a typical large SRAM is organized as 512 k X 8 bits.



Asynchronous Static RAM

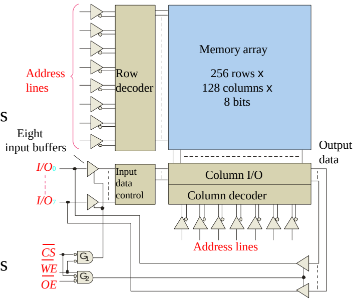
The basic organization of an asynchronous SRAM is shown.

Read cycle sequence:

- A valid address is put on the address bus
- Chip select is LOW
- Output enable is LOW
- Data is placed on the data bus

Write cycle sequence:

- A valid address is put on the address bus
- Chip select is LOW
- Write enable is LOW
- Data is placed on the data bus

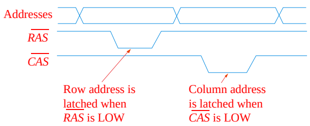


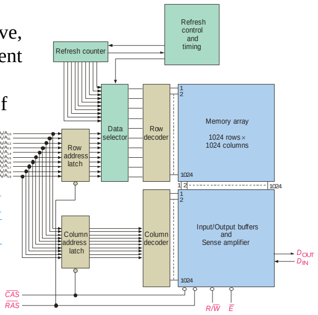
Dynamic RAM (DRAM)

Dynamic RAMs (DRAMs) store data bits as a charge on a capacitor.

DRAMs are simple and cost effective, but require refresh circuitry to prevent losing data. The address lines are multiplexed to reduce the number of address lines.

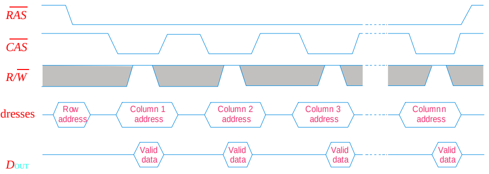
Multiplexed address lines:


Addresses
$\overline{RAS}$
$\overline{CAS}$
Row address is latched when $\overline{RAS}$ is LOW
Column address is latched when $\overline{CAS}$ is LOW



Dynamic RAM (DRAM)

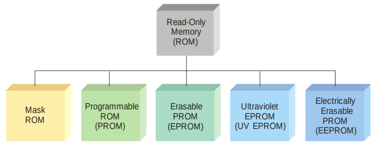
A feature with some DRAMs is fast page mode. Fast page mode allows successive read or write operations from a series of columns address that are all on the same row.



Other types of DRAMs have been developed to speed access and make the processor more efficient. These include EDO DRAMs, BEDO DRAMs and SDRAMs, as described in the text.

Read-Only Memory (ROM)

The ROM family is all considered non-volatile, because it retains data with power removed. It includes various members that can be either permanent memory or erasable.



Read-Only Memory (ROM)

Mask ROM

Programmable ROM (PROM)

Erasable PROM (EPROM)

Ultraviolet EPROM (UV EPROM)

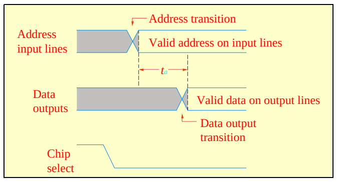
Electrically Erasable PROM (EEPROM)

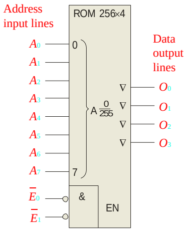
ROMs are used to store data that is never (or rarely) changed such as system initialization files. ROMs are *non-volatile*, meaning they retain the data when power is removed, although some ROMs can be reprogrammed using specialized equipment.

Read-Only Memory (ROM)

A ROM symbol is shown with typical inputs and outputs. The triangles on the outputs indicate it is a tri-stated device.

To read a value from the ROM, an address is placed on the address bus, the chip is enabled, and a short time later (called the access time), data appears on the data bus.

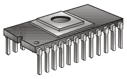




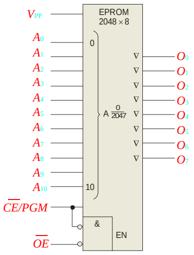
PROMs, EPROMs and EEPROMs

PROMs are programmable ROM, in which a fused link is burned open during the programming process. Once the PROM is programmed, it cannot be reversed.

An EPROM is an erasable PROM and can be erased by exposure to UV light through a window. To program it, a high voltage is applied to V_{PP} and $\overline{OE}$ is brought LOW.



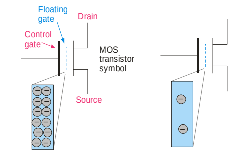
Another type of erasable PROM is the EEPROM, which can be erased and programmed with electrical pulses.



Flash Memory

Flash memories are high density read/write memories that are nonvolatile. They have the ability to retain charge for years with no applied power.

Flash memory uses a MOS transistor with a floating gate as the basic storage cell. The floating gate can store charge (logic 0) when a positive voltage is applied to the control gate. With little or no charge, the cell stores a logic 1.



logic 0 is stored

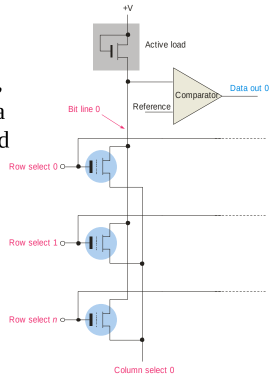
logic 1 is stored

The flash memory cell can be read by applying a positive voltage to the control gate. If the cell is storing a 1, the positive voltage is sufficient to turn on the transistor; if it is storing a 0, the transistor is off.

Flash Memory

Flash memories arranged in arrays with an active load. For simplicity, only one column is shown. When a specific row and column is selected during a read operation, the active load has current.

One drawback to flash memory is that once a bit has been set to 0, it can be reset to a 1 only by erasing an entire block of memory. Another limitation is that flash memory has a large but finite number of read/write cycles.

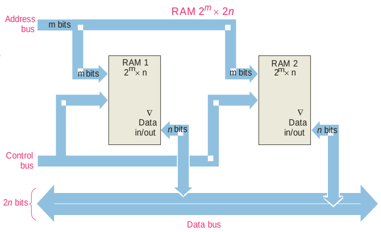


Memory Expansion

Memory can be expanded in either word size or word capacity or both.

To expand word size:

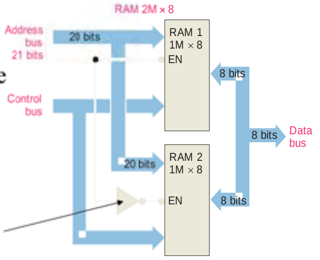
Notice that the data bus size is larger, but the number of address is the same.



Memory Expansion

To expand word capacity, you need to add an address line as shown in this example

Notice that the data bus size does not change.



Question

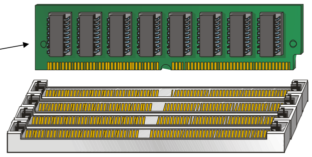
What is the purpose of the inverter?

Answer

Only one of the ICs is enabled at any time depending on the logic on the added address line.

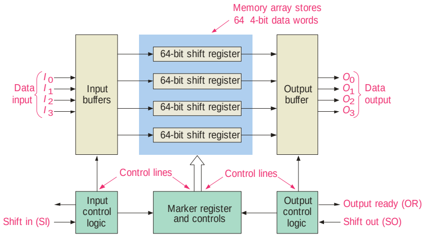
SIMMs and DIMMs

SIMMs (single in-line memory modules) and DIMMs (dual in-line memory modules) are plug-in circuit boards containing the ICs and I/O brought out on edge connectors. SIMMs have a 32-bit data path with I/O on only one side whereas DIMMs have a 64-bit data path with I/O on both sides of the board.



FIFO Memory

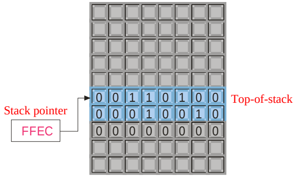
FIFO means first in-first out. This type of memory is basically an arrangement of shift registers. It is used in applications where two systems communicate at different rates.



LIFO Memory

LIFO means last in-first out. In microprocessors, a portion of RAM is devoted to this type of memory, which is called the **stack**. Stacks are very useful for temporary storage of internal registers, so that the processor can be interrupted but can easily return to a given task.

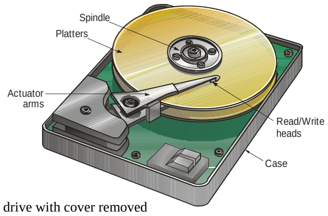
A special register, called the stack pointer, keeps track of the location that data was last stored on the stack. This will be the next data to be taken from the stack when needed.



Magnetic Hard Drive

The magnetic hard drive is the backbone of computer mass storage and is applied to other devices such as digital video recorders. Capacities of hard drives have increased exponentially, with 1 TB (1 trillion bytes!) drives available today.

Platters are arranged in tracks (circular shapes) and sectors (pie shaped). Files are listed in a File Allocation Table, (FAT) that keeps track of file names, locations, size, and more.



Hard drive with cover removed

Optical Storage

The compact disk (CD) uses a laser to burn tiny *pits* into the media. Surrounding the pits are flat areas called *lands*. The CD can be read using a low-power IR laser that detects the difference between pits and lands.

Binary data is encoded with a special method called negative non-return to zero encoding. A change from a pit to a land or a land to a pit represents a binary one, whereas no change represents a zero. A standard 120 mm CD can hold approximately 700 MB of data.

Selected Key Terms

Address The location of a given storage cell or group of cells in memory.

Capacity The total number of data units (bits, nibbles, bytes, words) that a memory can store.

SRAM Static random access memory; a type of volatile read/write semiconductor memory.

DRAM Dynamic random access memory; a type of read/write memory that uses capacitors as the storage elements and is a volatile read/write memory.

PROM Programmable read-only memory; type of semiconductor memory.

Selected Key Terms

EPROM Erasable programmable read-only memory; a type of semiconductor memory device that typically uses ultraviolet light to erase data.

Flash memory A nonvolatile read/write random access semiconductor memory in which data are stored as charge on a floating gate of a certain type of FET.

FIFO First in-first out memory.

LIFO Last in-first out memory

Hard disk A magnetic storage device; typically a stack of two or more rigid disks enclosed in a sealed housing.

1. Static RAM is

   a. nonvolatile read only memory

   b. nonvolatile read/write memory

   c. volatile read only memory

   d. volatile read/write memory

2. A nonvolatile memory is one that

   a. requires a clock

   b. must be refreshed regularly

   c. retains data without power applied

   d. all of the above

3. The advantage of dynamic RAM over static RAM is that

   a. it is much faster

   b. it does not require refreshing

   c. it is simpler and cheaper

   d. all of the above

4. The first step in a read or write operation for a random access memory is to

   a. place a valid address on the address bus

   b. enable the memory

   c. send or obtain the data

   d. start a refresh cycle

5. The output enable signal ($\overline{OE}$) on a RAM is active

   a. only during a write operation

   b. only during a read operation

   c. both of the above

   d. none of the above

6. When data is read from RAM, the memory location is

   a. cleared after the read operation

   b. set to all 1’s after the read operation

   c. unchanged

   d. destroyed

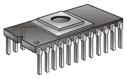
7. An EPROM has a window to allow UV light to enter under certain conditions. The purpose of this is to

a. refresh the data

b. read the data

c. program the IC

d. erase the data



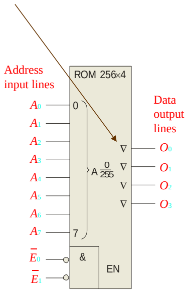
8. The small triangles on the logic diagram indicate that these outputs are

a. not used

b. tri-stated

c. inverted

d. grounded



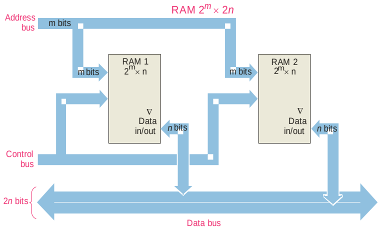
9. Using two ICs as shown will expand

a. the word size

b. the number of words available

c. both of the above

d. none of the above



10. On a hard drive. information about file names, locations, and file size are kept in a special location called the

a. file location list

b. file allocation table

c. disk directory

d. stack

UNIVERSITY of TECHNOLOGY, MAURITIUS

BEng (Hons.) Electronic Engineering

BEng (Hons.) Telecommunications

BEngEE/09/FT - BEngTEL/09/FT

Examinations for 2010-2011 / Semester 1

MODULE: DIGITAL ELECTRONICS

MODULE CODE: ELEC2102

Duration: 2½ Hours

Reading Time: None

Instructions to Candidates:

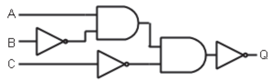
1. Attempt **ALL FOUR** questions
2. Start your answer to each question on a fresh page
3. Questions carry equal marks
4. Total Marks = 100
5. Electronic Calculators are allowed in the Examination Room

This question paper contains 4 questions and 5 pages.

ATTEMPT ALL FOUR QUESTIONS

QUESTION 1 (25 Marks)

(a) Assume the following circuit below:



(i) Derive an equation for the output Q in terms of A,B and C.

(ii) Simplify the equation above using De Morgan's Theorems and draw the resulting equivalent circuit.

*(Assume that complemented inputs are **unavailable**)*

(2+6 marks)

(b) Using Boolean algebra, simplify the following expression:

$$X = AB + A(B+C) + B(B+C)$$

(4 marks)

(c) Convert the following expression in **standard POS** form:

$$X = (\bar{A}+\bar{B})(A+B+C)$$

(4 marks)

(d) Using **K-maps**, simplify the following examples:

(i) $f(A,B,C,D) = \sum m(1,3,4,5,7,10,11,12,14,15)$

(ii) $f(A,B,C) = BC + \overline{A}\,\overline{C}$ and 'don't care' conditions as $A\overline{B} + AB\overline{C} + \overline{A}\,\overline{B}C$

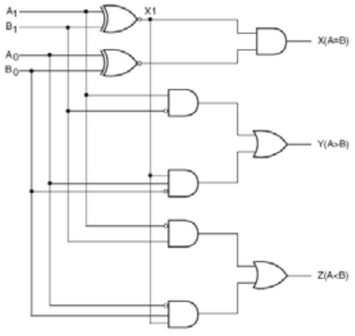
(4+5 marks)

QUESTION 2 (25 MARKS)

(a) Implement $f(A,B,C) = \overline{A}C + A\overline{B}C + AB\overline{C}$ using only **one 4-to-1** multiplexer.

(8 marks)

(b) Hardware-implement a **3-bit comparator** having only one output that goes HIGH when the two 3-bit numbers are equal. Use **only NAND** gates. *(To help you, find below the implementation of a **2-bit magnitude** comparator.)*



(7 marks)

(c) Design a **4-to-1 multiplexer** using **two 2-to-1 multiplexers** each with active-low ENABLE lines and NOR gates only.

(Make sure to label Inputs lines I_0 - I_3, Select lines S_0 & S_1 and EN lines correctly)

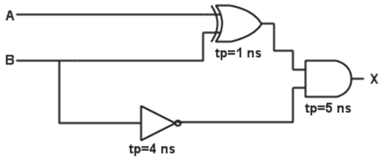
(6 marks)

(d) Design a **4-bit adder** by using **four full-adders** in parallel.

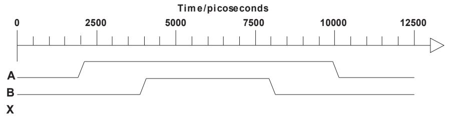
(4 marks)

QUESTION 3 (25 MARKS)

(a) Consider the following circuit, with typical gate propagation delays, as shown:



(i) Sketch the output waveform X for the following input waveforms:

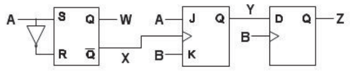


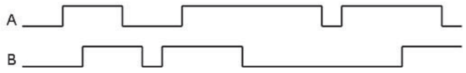
(5 marks)

(ii) Sketch the equivalent circuit using NAND gates only and calculate the overall propagation delay for the new circuit with one NAND delay of **3 ns**.

(6+2 marks)

(b) The signals A and B are applied to the inputs of a SR latch, a JK-FF and a D-FF. Sketch the waveforms at W, X, Y and Z. *(Assume all Q are HIGH initially)*





(2+2+3+5 marks)

QUESTION 4: (25 MARKS)

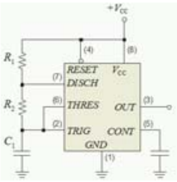
(a) There are three main digital IC technologies: TTL, ECL and CMOS

- (i) What is the meaning of a ECL?
- (ii) Give **two** advantages of ECL over TTL?
- (iii) What do yo understand by the fall-time of a digital pulse?

(2+4+2 marks)

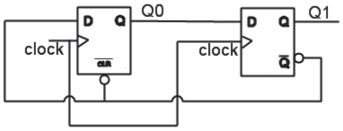
(b) Given the following 555 Timer IC and assuming that R_1 = 50 kΩ and C_1 = 10 μF:

- (i) Evaluate the value of resistor R_2 which will provide a 60% duty-cycle?
- (ii) Hence, calculate the pulse period of the output.



(3+5 marks)

(c) The circuit below contains 2 D-type flip-flops with an asynchronous, active-low $\overline{CLR}$ input. Assume that Q0 and Q1 are both LOW initially, sketch a timing diagram showing the waveforms of CLOCK, Q0 and Q1 for the next six clock pulses.



(3+3+3 marks)

***** End of Exam Paper *****

UNIVERSITY of TECHNOLOGY, MAURITIUS

BEng (Hons.) Telecommunications

BEng (Hons.) Electronic Engineering

BTEL/14/FT- BEE/14/FT

Examinations for 2015-16 / Semester 1

MODULE: DIGITAL ELECTRONICS

MODULE CODE: ELEC2102C

Duration: 2½ Hours

Reading Time: None

Instructions to Candidates:

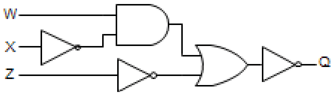
1. Attempt **ALL FOUR** questions
2. Start your answer to each question on a fresh page
3. Questions carry equal marks
4. Total Marks = 100
5. Electronic Calculators are allowed in the Examination Room

This question paper contains 4 questions and 5 pages.

ATTEMPT ALL FOUR QUESTIONS

QUESTION 1 (25 Marks)

(a) Assume the following circuit below:



- (i) Derive an equation for the output Q in terms of W, X and Z.
- (ii) Using the equation above using De Morgan's Theorems and redraw the resulting equivalent circuit using only NOR gates.

*(Assume that complemented inputs are **not available**)*

(2+6 marks)

(b) Using Boolean algebra, simplify the following expression:

$$\mathbf{XYZ + \overline{X}Y + XY\overline{Z}}$$

(4 marks)

(c) Convert the following expression in **standard POS** form:

$$\mathbf{X = (\overline{A}+\overline{B}).(A+B+C)}$$

(4 marks)

(d) Using **K-maps**, simplify the following examples:

- (i) $\mathbf{f(W,X,Y,Z) = m(1,3,4,5,7,10,11,12,14,15)}$
- (ii) $\mathbf{f(A,B,C) = BC + \overline{AC}}$ and 'don't care' conditions as $\mathbf{A\overline{B}+AB\overline{C}+\overline{ABC}}$

(4+5 marks)

Question 2 (25 MARKS)

(a) Implement the function $f(A,B,C,D) = \Sigma\, m(1,2,5,8,10,12,13,14)$ using only **one 8-to-1** multiplexer.

(8 marks)

(b) Show how you can design one **12-bit magnitude** comparator using only **4-bit magnitude** comparators.

(8 marks)

(c) Design a **16-to-1 multiplexer** with an active-high ENABLE line using **two 8-to-1 multiplexer blocks** both with active-high ENABLE lines**.**

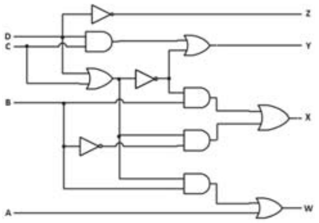
(5 marks)

(d) Design a **full-adder** using **2 half-adders** and an **OR** gate. (Do not use any additional gates.)

(4 marks)

QUESTION 3 (25 MARKS)

(a) Consider the following combinational circuit below:



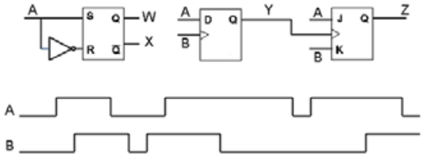
(i) Calculate the propagation delay for **each** output. Assume that there are no hazards and that each gate has a delay of 10 ns.

(4 marks)

(ii) Redraw the above circuit using only **NAND gates** with propagation delay of 5 ns and calculate the new overall propagation delay for each output.

(8 marks)

(b) The signals A and B are applied to the inputs of a SR latch, a D-FF and a JK-FF. Sketch the waveforms at W, X, Y and Z. *(Assume all Q are HIGH initially)*



(2+3+3+5 marks)

QUESTION 4: (25 MARKS)

(a) There are three major digital IC technology: TTL, CMOS and ECL

(i) What is the meaning of a CMOS?

(ii) Give two advantages of CMOS over TTL?

(iii) Give two advantages of ECL over CMOS?

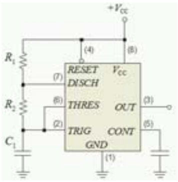
(iv) What do yo understand by the fall-time of a digital pulse?

(3+2+4+2 marks)

(b) Given the following 555 Timer IC and assuming that R_1 = 10 kΩ and C_1 = 5 μF:

(i) Evaluate the value of resistor R_2 which will provide a 50% duty-cycle?

(ii) Hence, calculate the pulse width of the output.



(3+5 marks)

(c) Draw a circuit diagram for a **4-bit asynchronous** counter.

(6 marks)

***** End of Exam Paper *****