Digital and Analog Quantities

An analog quantity has continuous values.

\[ \text{Temperature} \]

\[ 90^\circ \quad 75^\circ \]

0 1 2 3 4 5 6 7 Time

Hours

A digital quantity has discrete values.

\[ \text{Temperature} \]

\[ 90^\circ \quad 75^\circ \]

0 1 2 3 4 5 6 7 Time

The analog signal has been digitized by sampling it each hour and it can be stored as a set of digital codes.
An analog Electronic system:

An audio cassette tape recorder/player would also be an analog system.

A compact disc (CD) player incorporates both digital and analog systems.
A binary system has two digits: 1 and 0. A binary digit is called a bit.

In positive logic a: high voltage level = 1
low voltage level = 0

Groups of bits are called codes.

A byte consists of a group of 8 bits.

Logic Levels

Typical Values for TTL logic: 

\[ V_{H_{(\text{max})}} = 5.5\, \text{V}, \quad V_{H_{(\text{min})}} = 2.4\, \text{V}, \quad V_{L_{(\text{max})}} = 0.8\, \text{V}, \quad V_{L_{(\text{min})}} = 0\, \text{V}. \]
The **rise time** is the time required for the pulse to go from 10% to 90% of its final value.

The **pulse width** is measured between the 50% points on the rising and falling edges.
A periodic waveform repeats itself at a fixed interval called a period \((T)\).

The rate at which a periodic waveform repeats itself is called its frequency \((f)\).

**Units of \(T\)** are: seconds, milliseconds, microseconds.

**Units of \(f\)** are: hertz (cycles per second), kilohertz, megahertz.

\[
\text{Period} = T_1 = T_2 \\
\frac{1}{T} \quad \frac{1}{f}
\]
The **Duty cycle** is the ratio of pulse width to the period expressed as a percentage.

\[
\text{Duty cycle} = \left( \frac{t_w}{T} \right) \times 100\%
\]

More generally the duty cycle is the percentage of time a device is in "ON".

Ex/ An air conditioner which runs for 45 minutes of each hour would be referred to as having a 75% duty cycle.

Ex/ Find: period, frequency and duty cycle.

Sol: \( T = 10 \text{ ms} \)

\[ f = \frac{1}{T} = \frac{1}{10 \text{ ms}} = \frac{1}{10 \times 10^{-3}} = 100 \text{ Hz} \]

\[ \text{Duty Cycle} = \left( \frac{t_w}{T} \right) \times 100\% = \left( \frac{1 \text{ ms}}{10 \text{ ms}} \right) \times 100\% = 10\% \]
Digital Waveforms carry Binary Information

Each bit in a sequence occupies a defined time interval called a **bit time**.

The system's **clock** is a periodic waveform used to synchronize the other waveforms in the system.

1. Signal A's transitions are synchronized with the clock's rising edge.

2. If A were a received signal containing serial digital data, a bit of data could be received by sampling (reading) A's value at each falling edge of the clock.
A timing diagram is a graph of digital waveform on a common axis to show the time relationship of each waveform with respect to the others.

Example:

- **Clock**
- **A**
- **B**
- **C**

Only around time $t_1$ are all waveforms high.

**Data Transfer**

When bits are transferred in **serial** form, they are sent one bit at a time along a single conductor.

When bits are transferred in **parallel** form, all of the bits in a group are sent simultaneously on several separate lines.
Serial Data Transmission:

![Diagram of serial data transmission with a binary sequence](image)

Serial Transmission:
- Advantage: requires only single conductor or channel
- Disadvantage: Slower than parallel transmission

Parallel Data Transmission:

![Diagram of parallel data transmission](image)

An eight bit byte can be sent all at once.

Parallel Transmission:
- Advantage: faster data rate than serial
- Disadvantage: requires more conductors
If a 100 kHz clock is used, how long would it take to send an 8-bit quantity in a) serial form

b) parallel form

a) Solution: One bit can be sent during each clock period.

For a 100 kHz clock: \( T = \frac{1}{f} = \frac{1}{100 \text{ kHz}} = 10 \mu\text{sec} \)

Total Transfer Time = 8 \( \times 10 \mu\text{sec} = 80 \mu\text{sec} \)

b) In parallel form, all of the bits can be sent in a single clock period.

Total Transfer Time = 10 \( \mu\text{sec} \)
Basic Logic Operations

Inverter - (performs NOT operation i.e. output ≠ input)

AND gate

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(output only Hi when both inputs are Hi)

OR gate

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(output only Lo when both inputs are Lo)

Exclusive-OR gate (XOR)

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(output only Lo when both inputs are equal)
TABLE 5-6
Truth table for the control logic

<table>
<thead>
<tr>
<th>Inputs</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>Shut down</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 0 0 1</td>
<td>0 0 0 1</td>
<td>0 0 0 1</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
<td>0 0 1 1</td>
<td>Unallowed (2 saws on). Shut down</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
<td>Unallowed (No lub with conveyor). Shut down</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
<td>Unallowed (conveyor and cross cut). Shut down</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 1 0</td>
<td>0 1 1 0</td>
<td>0 1 1 0</td>
<td>Unallowed (No lub with conveyor). Shut down</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1 1 1</td>
<td>0 1 1 1</td>
<td>0 1 1 1</td>
<td>Unallowed (2 saws and conveyor). Shut down</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>1 0 0 0</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0 0 1</td>
<td>1 0 0 1</td>
<td>1 0 0 1</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 0 1 0</td>
<td>1 0 1 0</td>
<td>1 0 1 0</td>
<td>Unallowed (2 saws). Shut down</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
<td>1 1 0 0</td>
<td>Unallowed (conveyor and cross cut). Shut down</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1 0 1</td>
<td>1 1 0 1</td>
<td>1 1 0 1</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1 1 0</td>
<td>1 1 1 0</td>
<td>1 1 1 0</td>
<td>Unallowed (all on). Shut down</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>1 1 1 1</td>
<td>Unallowed (all on). Shut down</td>
<td></td>
</tr>
</tbody>
</table>

switches and, as a result, sixteen possible ON/OFF combinations for the motors. The states of the switches are the input variables and the states of the motors are the output variables. The unallowed conditions in this case are not treated as “don’t cares.” They are switch input conditions that should not occur but if they do, the system must be shut down with all motors off.

Design of the Control Logic

There are four separate logic circuits, one for each of the motors. Let’s begin by designing the logic circuit for the lubrication pump motor (output $M_1$). The first step is to transfer the data from the truth table to a Karnaugh map and develop an SOP expression.

The switch variables $S_1$, $S_2$, $S_4$, and $S_5$ are map variables and the states of $M_1$ are plotted and grouped as shown in Figure 5–45(a). The 0s on the map are for switch conditions when the motor is off and the 1s are for switch conditions when the motor is on. The resulting SOP expression for the lubrication pump motor logic results in the NAND implementation shown in part (b).

![Karnaugh map simplification and implementation for the lubrication motor logic.](image)

$$M_1 = S_4 \bar{S}_5 + S_4 S_5$$
Basic Logic Functions

Logic gates and inverters are combined within a single integrated circuit to implement more complex logic functions.

Magnitude Comparator

Has 2 binary inputs: \( A \) & \( B \)
Has 3 outputs: \( A < B \), \( A = B \), \( A > B \)

\[
\begin{array}{c}
\text{Inputs} \\
A_3 A_2 A_1 A_0 \\
B_3 B_2 B_1 B_0
\end{array}
\begin{array}{c}
\text{Outputs} \\
A < B \\
A = B \\
A > B
\end{array}
\]

Arithmetic Functions:

Adder performs addition
Has 3 inputs: \( A \), \( B \), carry-in
Has 2 outputs: sum, carry-out

\[
\begin{array}{c}
\text{Inputs} \\
A \\
B \\
Cy_{in}
\end{array}
\begin{array}{c}
\text{Outputs} \\
\text{Sum} \\
Cy_{out}
\end{array}
\]

\[
\begin{array}{c|ccc|c|cc}
\text{Inputs} & 0 & 0 & 0 & 0 & 0 & 0 \\
A & B & Cy_{in} & Y & Cy_{out}
\end{array}
\]

\[
\begin{array}{c|ccc|c|cc}
\text{Inputs} & 0 & 1 & 0 & 1 & 0 & 1 \\
A & B & Cy_{in} & Y & Cy_{out}
\end{array}
\]
Subtractor performs subtraction

Has 3 inputs: A, B, borrow input
Has 2 outputs: difference, borrow output

Multiplier

Has 2 inputs: A, B
Has 1 output (twice as many bits wide as one of the inputs)

If A and B are 8 bit quantities, the output (product) must hold up to 16 bits

Division

Is performed by a series of subtractions, comparisons and shifts.
Division usually is much slower than addition, subtraction or multiplication.

Code Conversion

A code converter changes a form of coded information into another coded form.

Example: Binary form → BCD form
          Decimal → BCD
Encoder

An encoder converts information such as a decimal number or alphabetic character into some coded form.

Keypad Encoder

When one of the 12 keys is pressed, one of the 12 lines going to the encoder will go HIGH. The encoder will then convert this to a 4-bit binary code corresponding to that key.

Decoder

A decoder converts coded information, such as binary numbers, into a non-coded form, such as a decimal form.

BCD to 7-segment display decoder

MAN 74
Seven segment LED display.
Displays digits 0 thru 9
Data Selectors — (Multiplexers & Demultiplexers)

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

A demultiplexer (DEMUX) is a circuit that switches digital data from one input line to several output lines.
Data Storage Devices

Flip-Flops

These are bistable (2 stable states) logic circuits that can store one bit of data.

Registers

A register is a group of flip-flops used to store a group of bits.

Example: An 8 bit register can store an 8 bit binary number (byte).

Semiconductor Memories

They are used to store a large number of bits of data.

Example: 
ROM is read-only memory which can't be changed. (not affected by removing power)

RAM is random-access memory which can be changed. (data lost when power removed)

Magnetic Memories

They are used for mass storage of binary data. The data is not affected by removing power.

Examples: Floppy Disks, computer hard drives, magnetic tape
Counters

A counter is used to count events represented by changing levels or pulses and store the total.

\[
\text{Count Sequence:} \quad \begin{array}{cccc}
Q_3 & Q_2 & Q_1 & Q_0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & \cdots & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}
\]

Counter "rolls over" automatically to 0000 after 15 counts.
CHAPTER 1
DIGITAL SYSTEM APPLICATION

There are no activities required in this section. The block diagram is given below.

Keypad for entering number of tablets per bottle

Encoder

Decoder A

On-site display

Number of tablets per bottle

Register A

Code converter A

Comp

A

A = B

B

Number of tablets per bottle

On-site display of total tablets bottled

HIGH closes valve and advances conveyor. LOW keeps valve open.

Valve

Sensor

Converter control

One pulse from sensor for each tablet advances counter by 1.

Counter

Adder

A

Σ

B

C_{out}

HIGH causes new sum to be stored.

Register B

Current sum

MUX

Total number of tablets is transferred in serial form along this line for remote display and computer inventory control.

Demux

Decoder C

550

Remote unit

Register C
Integrated Circuit Packages

An I.C. consists of a small silicon chip which contains thousands of transistor circuits. It is encased in a plastic or ceramic package with leads brought out.

Package types:

**DIP** - dual-in-line-pins
mounts thru circuit board holes

**SMT** - surface mount technology.
Mounts on PC board surface and take less space than DIP

**Pin Numbers**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>
Notch or pot

Pin indicated by pot and numbers go CCW around package

**Sizes**:

SSI - small scale integration
MSI -
LSI -
VLSI -
ULSI - ultra large scale integration
IC Technologies

Logic Families:

TTL - most common, uses bipolar transistors
  high speed (25 - 80 MHz)
  high power supply current (mA)

CMOS - widely used, uses MOSFET transistors
  slower speed (5 - 30 MHz)
  very low power supply current (μA)

NMOS - used in VLSI and ULSI chips

ECL - very high speed (100 MHz+)
  high power supply current
  requires special circuit board layouts

Test Instruments

Oscilloscope - displays rapidly changing voltages as a function of time.

Logic Analyzer - like an oscilloscope but also has special formats for displaying several digital signals simultaneously. (Timing diagrams, state tables)

Logic Probe - lamp indicates: Hi, Lo, pulsing, open for quick trouble shooting
Logic Pulser - used to inject logic pulses into digital circuits for trouble shooting.

Current Probe - used to detect current flowing in printed circuit traces.

DC Power Supply - converts 115 VAC line voltage into +5 VDC to power logic IC's.

Function Generator - can produce a wide variety of waveforms for circuit testing.

Digital Multimeter - can accurately measure D.C. voltages and currents. Can measure low frequency AC signals amplitude. Also can test continuity and measure resistance.
Decimal Numbers: contain digits 0 thru 9, also called base 10.

Each digit represents a power of 10

\[
\begin{array}{cccc}
10^2 & 10^1 & 10^0 \\
3 & 4 & 2 \\
\end{array}
\]

\[3 \times 10^2 + 4 \times 10^1 + 2 \times 1 = 300 + 40 + 2\]

Ex. Digits to the right of the decimal point represent negative powers of ten

\[
\begin{array}{cccc}
10^{-2} & 10^{-1} & 10^0 \\
5 & 6 & 7 \\
\end{array}
\]

\[7.65 = \frac{7}{10} + \frac{6}{10} + \frac{5}{10^2}\]

Binary Numbers: contain digits 0 and 1, also called base 2.

Each digit represents a power of 2

\[
\begin{array}{ccc}
2^2 & 2^1 & 2^0 \\
1 & 0 & 1 \\
\end{array}
\]

\[1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 0 + 1 = 5\]
### Counting in Binary

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

With $n$ bits, the largest decimal number which can be represented is given by:

$$\text{Largest decimal number} = 2^n - 1$$

**Ex:** with 4 bits, we can count from 0 to 15, the largest decimal number is $2^4 - 1 = 16 - 1 = 15$.

**Ex:** with 8 bits, we can represent decimal numbers up to:

$$2^8 - 1 = 256 - 1 = 255$$

**Note:** with 8 bits, we can represent 256 different possibilities i.e., the numbers from 1 to 255 and Zero.
Convert 1101101 to decimal

Weights: 2^6 2^5 2^4 2^3 2^2 2^1 2^0

1101101 = 2^6 + 2^5 + 0 + 2^3 + 2^2 + 0 + 2^0

= 64 + 32 + 0 + 8 + 4 + 0 + 1 = 109

Convert 0.1011 to decimal

Weights: 2^-1 2^-2 2^-3 2^-4

0.1011 = 0 + 2^-1 + 0 + 2^-3 + 2^-4

= 0 + 1/2 + 0 + 1/8 + 1/16

= 0 + 0.5 + 0 + 0.125 + 0.0625 = 0.6875

Decimal-to-Binary Conversion

Sum-of-Weights Method:

Determine the set of binary weights whose sum is equal to the decimal number.

To find each member of the set, determine the largest power-of-two less than the number. Subtract it from the number and repeat the process.
Example 1: Convert 53 to binary:

List Binary Weights: $1, 2, 4, 8, 16, 32, 64, 128, 256$

$2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8$

Since 32 is the largest binary weight less than 53, we have $2^5$ power present and will need 6 binary digits.

$53 - 32 = 21$

Since 16 is the largest binary weight less than 21, we have $2^4$ power present.

$21 - 16 = 5$

Since 8 > 5 we have no $2^3$ present.

Since 4 is the largest binary weight less than 5, we have $2^2$ power present.

$5 - 4 = 1$

Since 2 > 1 we have no $2^1$ present.

Since 1 is the largest binary weight less than or equal to 1, we have $2^0$ power present.

Re-capping results gives: $53 = 110101$

Checking: $53 = 32 + 16 + 0 + 4 + 0 + 1$ ✔

Example 2:

$12 = 8 + 4 = 2^3 + 2^2 = 1100$

$82 = 64 + 16 + 2 = 2^6 + 2^4 + 2^1 = 1010010$
Repeated Division-by-2 Method

ex/ Convert 12 to binary:

\[
\begin{align*}
\frac{12}{2} &= 6 & \text{Remainder: } 0 \\
\frac{6}{2} &= 3 & \text{Remainder: } 0 \\
\frac{3}{2} &= 1 & \text{Remainder: } 1 \\
\frac{1}{2} &= 0 & \text{Remainder: } 1 \\
\end{align*}
\]

Stop when the whole-number quotient is 0

\[12 = 1100\]

Sum-of-Weights Method for Fractions

Weights: 0.5, 0.25, 0.125, 0.0625
\[
2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}
\]

ex/ Convert 0.625 to binary

\[0.625 = 0.5 + 0.125\]

So there is a one in the \(2^{-1}\) and \(2^{-3}\) positions

\[0.625 = 0.101\]
Four Basic Rules for Adding Binary Digits:

\[
\begin{align*}
0 + 0 &= 0 & \text{Sum of 0 with a carry of 0} \\
0 + 1 &= 1 & \text{Sum of 1 with a carry of 0} \\
1 + 0 &= 1 & \text{Sum of 1 with a carry of 0} \\
1 + 1 &= 10 & \text{Sum of 0 with a carry of 1} \\
\end{align*}
\]

**Example 1:** Find \(11 + 1\)

\[
\begin{array}{c}
11 \\
+ 01 \\
\hline
100
\end{array}
\]

\(\hat{\uparrow}\) a carry of 1 resulted twice

**Example 2:** Find \(100 + 10\)

\[
\begin{array}{c}
100 \\
+ 10 \\
\hline
110
\end{array}
\]

\(\leftarrow\) no carries resulted

**Example 3:** Find \(11001 + 1101\)

\[
\begin{array}{c}
11001 \\
+ 1101 \\
\hline
100110
\end{array}
\]

\(\leftarrow\) 1 carry
**Binary Subtraction**

Four Basic Rules for Subtracting Binary Digits

\[
\begin{align*}
0 - 0 &= 0 \\
1 - 1 &= 0 \\
1 - 0 &= 1 \\
10 - 1 &= 1 (0 - 1 = 1 \text{ with a borrow of 1})
\end{align*}
\]

**Example 1** Find \(11 - 01\)

\[
\begin{array}{c}
\phantom{-}11 \\
-\ 01 \\
\hline
10 \quad \leftarrow \text{no borrows required}
\end{array}
\]

**Example 2** Find \(111 - 100\)

\[
\begin{array}{c}
\phantom{-}111 \\
-\ 100 \\
\hline
011 \quad \leftarrow \text{no borrows required}
\end{array}
\]

**Example 3** Find \(101 - 011\)

\[
\begin{array}{c}
\phantom{-}101 \\
\text{borrow} \quad \leftarrow \quad \begin{array}{c}
0 \\
\text{a one}
\end{array} \\
\phantom{-}011 \\
-\ 011 \\
\hline
010 \quad = 10 \\
\end{array}
\]

\[10 - 1 = 1 \text{ after the borrow}\]
Binary Multiplication

Four Basic Rules for Multiplying Binary Digits

\[
\begin{align*}
0 \times 0 &= 0 \\
0 \times 1 &= 0 \\
1 \times 0 &= 0 \\
1 \times 1 &= 1
\end{align*}
\]

\[
\begin{array}{cccc}
11 & \times 1 & 11 & \times 10 & 11 & \times 11 & 111 & \times 101 \\
1 & 110 & 11 & 1001 & 111 & 1110 & 10001 & 1
\end{array}
\]

Binary Division

Similar process to division in decimal

\[
\begin{align*}
& \underline{110 \div 11} \\
& 11 \overline{)110} \\
& 11 \\
& \underline{-11} \\
& 0
\end{align*}
\]

\[
\begin{align*}
& \underline{110 \div 10} \\
& 10 \overline{)110} \\
& 10 \\
& \underline{-10} \\
& 10 \\
& \underline{-10} \\
& 0
\end{align*}
\]
The 1's Complement of a binary number is found by changing the zeros to ones and changing the ones to zeros.

\[
\text{ex/} \quad 1\ 0\ 1\ 1\ 1\ 0\ 1 \quad \leftarrow \text{Binary Number} \\
\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
0\ 1\ 0\ 0\ 0\ 1\ 0 \quad \leftarrow 1's\ complement
\]

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

\[
\text{ex/} \quad 1\ 0\ 1\ 1\ 1\ 0\ 1 \quad \leftarrow \text{Binary Number} \\
0\ 1\ 0\ 0\ 0\ 1\ 0 \quad \leftarrow 1's\ complement \\
+\ 1 \\
0\ 1\ 0\ 0\ 0\ 1\ 1 \quad \leftarrow 2's\ complement
\]

Signed Numbers (positive and negative)

The left-most bit is used as the sign bit for binary numbers.

0 is used for positive

1 is used for negative

\[
\text{ex/} \quad 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 = 54 \\
\quad \begin{array}{c}
\text{Sign bit} \\
\text{Magnitude bits}
\end{array}
\quad 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0 = -54
\]
In the **sign-magnitude** system, a negative number has the same magnitude bits as the corresponding positive number, but the sign bit is a 1.

\[
\begin{align*}
54 &= 00110110 \\
-54 &= 10110110 \\
\end{align*}
\]

In the **1's complement** system, a negative number is the 1's complement of the corresponding positive number.

\[
\begin{align*}
54 &= 00110110 \\
-54 &= 11001010 \\
\end{align*}
\]

In the **2's complement** system, a negative number is the 2's complement of the corresponding positive number.

\[
\begin{align*}
54 &= 00110110 \\
-54 &= 11001010 \\
\end{align*}
\]

In computers the 2's complement system is most widely used because positive and negative numbers can be easily added to get other directly.

\[
\begin{align*}
54 + (-54) &= 0 \\
00110110 \\
+ \quad 11001010 \\
\hline
\quad 00000000 \\
\end{align*}
\]
Evaluation of Signed Numbers:

**Sign Magnitude (sum weights):**

\[ \overline{10010101} = -(2^0 + 2^3 + 2^4) = -(1 + 8 + 16) = -21 \]

- Sign bit indicates negative number.

**1's Complement:**

For positive numbers:

\[ \overline{00010111} = 2^0 + 2^1 + 2^2 + 2^4 = 1 + 2 + 4 + 16 = +23 \]

For negative numbers:

\[ \overline{11101000} = 2^3 + 2^5 + 2^6 - 2^7 \]

Add neg. of sign bits value plus one:

\[ = 8 + 32 + 64 - 128 + 1 = -24 + 1 \]

\[ = -23 \]

**2's Complement:**

For positive numbers:

\[ \overline{01010110} = 2^1 + 2^2 + 2^4 + 2^6 = 2 + 4 + 16 + 64 = +86 \]

For negative numbers:

\[ \overline{10101010} = 2^1 + 2^3 + 2^5 - 2^7 \]

Add neg. of sign bits value:

\[ = 2 + 8 + 32 - 128 = -86 \]
Range of Signed Numbers That Can Be Represented

For $n$ bits,
The Total Number of Combinations = $2^n$

Ex: With a byte, which is 8 bits,
$2^8$ or 256 combinations can be represented.

For 2’s complement signed numbers, the range of values for $n$-bit numbers is:

$-(2^{n-1})$ to $+(2^{n-1}-1)$

Ex: In a microprocessor with 8-bit registers, the range of signed numbers is:

$-(2^7)$ to $+(2^7-1)$
or $-128$ to $+127$

If the microprocessor only has to handle positive numbers the range is:

0 to 255

Ex: If the system has 16-bit registers, the range of signed numbers is:

$-(2^{15})$ to $+(2^{15}-1)$
or $-32,768$ to $+32,767$
Arithmetic Operation with Signed Numbers

In a 2's complement system:

**Addition:**

*ex*/ Both numbers positive:  
\[ \begin{array}{c}
00000111 \\
+ 00000100 \\
\hline
100001011
\end{array} \]

\[ \text{positive sum} \]

*ex*/ Positive number and a negative number with a smaller magnitude:  
\[ \begin{array}{c}
00001111 \\
+ 11111010 \\
\hline
100011001
\end{array} \]

\[ \text{Discard carry} \]

*ex*/ Positive number and a negative number with a larger magnitude:  
\[ \begin{array}{c}
00010000 \\
+ 11101000 \\
\hline
11111000
\end{array} \]

\[ -8 \]

*ex*/ Both numbers negative:  
\[ \begin{array}{c}
11111011 \\
+ 11110111 \\
\hline
111110010
\end{array} \]

\[ \text{Discard Carry} \]

Addition of signed numbers in 2's complement form is done by simply adding the two numbers together and discarding any final carry bit.
Overflow Condition:
When two numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an overflow results causing an error in the sign bit.

\[
\begin{array}{c}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0
\end{array}
\]

\[101101111\]

Sign Incorrect
Magnitude Incorrect

Subtraction:
The sign of a positive or negative binary number is changed by taking its 2's complement.

\[
\begin{array}{c}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\rightarrow & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0
\end{array}
\]

\[2^7 + 2^6 + 2^5 + 2^4 = -4\]

To subtract two signed numbers, take the 2's complement of the subtrahend and then add, discarding any final carry bit.

\[
\begin{array}{c}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
\rightarrow & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

\[2\text{'s compl. of } 3\]

\[100000101 \rightarrow 5\]

Discard Carry
Multiplication:

When two binary numbers are multiplied, both numbers must be in true (uncomplemented) form.

If the signs are the same, the product is positive.
If the signs are different, the product is negative.

\[ \begin{align*}
1101 \times 0011 &= \text{?} \\
A \times B
\end{align*} \]

\[ \text{(3bit magnitude \& sign bit)} \]

\[ A \text{ is negative since its sign bit is one.} \]

\[ \text{Since the signs are different, the product is negative.} \]

1) Convert both numbers to true form:

\[ B \text{ in true form magnitude is 011} \]

\[ \text{Take 2's complement of } A: 1101 \rightarrow 0010+1 = 0011 \quad (A = -3) \]

2) Multiply magnitudes:

\[ \begin{array}{c}
11 \\
\times 11 \\
\hline
11 \\
11 \\
\hline
1001
\end{array} \]

3) Re-insert sign bit: \[ A \times B = 10001001 = -9 \]

\[ \text{(7 bit magnitude \& sign bit)} \]

Note: The magnitude of the product can require as many bits as those contained in both multiplier and multiplicand.
Division:

When two binary numbers are divided, both numbers must be in true (uncomplemented) form.

If the signs are the same, the quotient is positive.
If the signs are different, the quotient is negative.

Division is done by repeated subtraction of the divisor from the dividend until a zero or negative result. Count the number of times subtraction can be done and this is the quotient.

Example: Find $01100100 \div 00011001$

1. Both are positive so result is positive.

2. Subtract by using 2's complement.

   $\begin{array}{c}
   01100100 \leftarrow \text{dividend} \\
   + 11100111 \leftarrow \text{2's complement of divisor} \\
   \hline
   01001011 \leftarrow \text{remainder}
   \end{array}$

   Repeat:

   $\begin{array}{c}
   01001011 \\
   + 11100111 \\
   \hline
   00110010
   \end{array}$

   Repeat:

   $\begin{array}{c}
   00110010 \\
   + 11100111 \\
   \hline
   00011001
   \end{array}$

   Repeat:

   $\begin{array}{c}
   00011001 \\
   + 11100111 \\
   \hline
   00000000 \leftarrow \text{indicates we're done}
   \end{array}$

Since four subtractions were needed, quotient is $00000100$
Hexadecimal Numbers

This is a base 16 number system used as a "short-hand" way of expressing binary numbers. They are often used in microprocessor applications to express 8 and 16 bit binary numbers. (see table pg. 23)

There are 16 digits in the hexadecimal number system: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

To convert from binary to hexadecimal, simply separate the binary number into groups of 4 digits and replace each group with its equivalent hexadecimal character.

Example: Convert to hexadecimal:

1101 → D

1001 1100 → 9A

9 A

0011 1111 0001 0110 → 3F16

3 F 1 6

0111 0101 → 75 since hex 75 could be confused with decimal 75, it is often written as:

75 16, or 75 h

Example: Convert hex to binary:

A 7 → 1010 0111

1010 0111
**Example: Convert hex to decimal:**

Can be done by first converting to binary or directly to decimal by adding together powers of 16.

\[ 1C = 0001 \, 1100 = 16^1 + 8 + 4 = 28_{10} \]

**Powers of 16:**

\[
\begin{array}{c}
16^3 & 16^2 & 16^1 & 16^0 \\
4096 & 256 & 16 & 1 \\
\end{array}
\]

\[ E5 = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = 229_{10} \]

\[ B2F8 = (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1) \]

\[ = (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1) \]

\[ = 45056 + 512 + 240 + 8 = 45816_{10} \]

**Example: Convert Decimal to Hex**

Can be done by using repeated division by 16:

Convert \( 650_{10} \) to hex.

\[
\begin{align*}
\frac{650}{16} & = 40 . \frac{625}{16} \rightarrow 0.625 \times 16 = 10 = A \\
\frac{40}{16} & = 2 . \frac{5}{16} \rightarrow 0.5 \times 16 = 8 = 8 \\
\frac{5}{16} & = 0 . \frac{125}{16} \rightarrow 0.125 \times 16 = 2 = 2 \\
\frac{2}{16} & \text{ Stop when whole number is zero} \\
\end{align*}
\]

**MSD to LSD**

39
Hexadecimal Addition:

\[
\begin{array}{c}
23_{16} \\
+ 16_{16} \\
\hline
39_{16}
\end{array}
\quad
\begin{array}{c}
58_{16} \\
+ 22_{16} \\
\hline
7A_{16}
\end{array}
\quad
\begin{array}{c}
2B_{16} \\
+ 84_{16} \\
\hline
AF_{16}
\end{array}
\]

Note: \( \text{F} + C = 15_{10} + 12_{10} = 27_{10} \)

\[
27_{10} - 16_{10} = 11_{10} = B_{16} \text{ with 1 carry}
\]

\[
D + A + 1 = 13_{10} + 10_{10} + 1_{10} = 24_{10}
\]

\[
24_{10} - 16_{10} = 8_{10} = 8_{16} \text{ with a 1 carry}
\]

Hexadecimal Subtraction:

Convert to negative number by taking 2's complement of equivalent binary number.

Find \( 84_{16} - 2A_{16} \)

1. Convert \( 2A_{16} \) to binary: \( 2A_{16} = 00101010 \)

2. Find 2's complement: \( = \overline{11010101} + 1 = \overline{1101010} \)

3. Add

\[
\begin{array}{c}
84_{16} \\
+ \overline{1101010} \\
\hline
\overline{5A}_{16}
\end{array}
\]

result is: \( 5A_{16} \)

\( \uparrow \) drop carry
Octal Numbers

The octal number system is base 8 and has the following digits: 0, 1, 2, 3, 4, 5, 6, 7

Octal-to-Decimal Conversion:

Weights: 8³ 8² 8¹ 8⁰
(512) (64) (8) (1)

\[ 2374_8 = (2 \times 8^3) + (3 \times 8^2) + (7 \times 8) + (4 \times 8^0) \]

\[ = (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \]

\[ = 1024 + 192 + 56 + 4 = 1276_{10} \]

Decimal-to-Octal Conversion:

ex) Convert \( 359_{10} \) to octal

\[ \frac{359}{8} = 44.875 \rightarrow \text{Integer} = 44 \]

\[ \frac{44}{8} = 5.5 \rightarrow 0.5 \times 8 = 4 \]

\[ \frac{5}{8} = 0.625 \rightarrow 0.625 \times 8 = 5 \]

Stop when whole number quotient is zero

\( 547_8 \)

MSD LSD
Octal-to-binary conversion:

Each octal digit is represented by 3 binary bits.

Octal: 0 1 2 3 4 5 6 7
Binary: 000 001 010 011 100 101 110 111

Example: Convert octal to binary:

\[
\begin{align*}
13_8 &= 0001 011_2 \\
7526_8 &= 111101010110_2
\end{align*}
\]

Example: Convert binary to octal

\[
\begin{align*}
110101_2 &= 65_8 \\
01101000100_2 &= 3204_8
\end{align*}
\]

Binary Coded Decimal (BCD)

BCD is a way to express the decimal digits 0 through 9 with 4 binary bits.

BCD format is useful for interfacing to binary systems from keypads and numeric displays. The most common BCD system is based on an 8421 code.

Decimal Digit: 0 1 2 3 4 5 6 7 8 9
BCD: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001
Invalid Codes:
4 bit binary numbers greater than 9 are considered invalid codes in a BCD system.

**Example: Convert decimal to BCDs**

\[
35 = \frac{3}{0011} \frac{5}{0101} = 0011 0101
\]

\[
170 = \frac{1}{0001} \frac{7}{0111} \frac{0}{0000} = 0001 0111 0000
\]

**Example: Convert BCD to decimal**

\[
\begin{align*}
1000 \ 0110 & \quad 0011 \ 0101 \ 0001 \\
= 8 \ 6 & \quad = 3 \ 5 \ 1
\end{align*}
\]

\[
\begin{align*}
1001 \ 1010 & \quad \text{invalid BCD code (not between 0 and 9)} \\
= 8 & \quad ?
\end{align*}
\]

**BCD Addition:**

Use same rules as binary, except if a 4-bit sum exceeds 9, it must cause a carry into the next higher 4-bit group.

**Example: Add these BCD quantities:**

\[
\begin{align*}
0011 & \quad 1001 & (9+4 = 13) \\
+0100 & \quad +0100 \\
0111 & \quad 1101 & \text{invalid BCD so,} \\
& \quad 0110 & \text{Add 6 and carry will be taken care of} \\
& \quad 0001 \ 0011 & \text{valid BCD Sum}
\end{align*}
\]
The Gray code is unweighted and not an arithmetic code.

The important feature of the Gray code is that it exhibits only a single bit change from one code number to the next.

This makes the Gray code desirable in applications such as shaft position encoders where multiple bit changes increase the chances of error.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray Code</th>
<th>Decimal</th>
<th>Binary</th>
<th>Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>8</td>
<td>1000</td>
<td>1100</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0001</td>
<td>9</td>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0011</td>
<td>10</td>
<td>1010</td>
<td>1111</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0010</td>
<td>11</td>
<td>1011</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0110</td>
<td>12</td>
<td>1100</td>
<td>1010</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>0111</td>
<td>13</td>
<td>1101</td>
<td>1011</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>0101</td>
<td>14</td>
<td>1110</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>0100</td>
<td>15</td>
<td>1111</td>
<td>1000</td>
</tr>
</tbody>
</table>
The Excess-3 code is unweighted and has advantages over BCD for certain arithmetic operations.

Each Excess-3 code value is found by adding 3 to each decimal digit and then converting the result to 4-bit binary.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>BCD</th>
<th>Excess-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>0100</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>0101</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>0110</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>1000</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>1001</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>1010</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>1011</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>1100</td>
</tr>
</tbody>
</table>

The key feature of the excess-3 code is that it is self-complementing. The 1's complement of an excess-3 number is the excess-3 code for the 9's complement of the corresponding decimal number.

\[
\text{Ex-3} \quad 4 \rightarrow 0111 \quad \rightarrow 1000 = 5 = 9\text{'s complement of 4}.
\]

The 9's complement can facilitate subtraction of decimal numbers.
ASCII (American Standard Code for Information Interchange) is an 8-bit code used to universally represent each of the alphanumeric characters and symbols on a keyboard.

Table 2-8 in the text lists the 8-bit binary (and hex) number corresponding to each of the following: small and capital letters, A thru Z, numbers 0 thru 9, punctuation marks, control characters, i.e., CTRL C, non-English letters (in extended ASCII).

The computer uses ASCII to store text in its memory.

Example: The message: PRINT A

Would be stored as the sequence of the following 7 binary numbers, (shown in hex notation):

P = 50_h
R = 52_h
I = 49_h
N = 4E_h
T = 54_h
space = 20_h
A = 41_h
Parity Method for Error Detection

An additional bit is attached to each group of bits before they are sent, which allows the receiver to detect an error of a single bit changing during transmission.

In an even parity system, the attached parity bit will make the total number of ones in the bit group an even number.

Example:

```
<table>
<thead>
<tr>
<th>P</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1001001</td>
</tr>
<tr>
<td>0</td>
<td>11011110</td>
</tr>
</tbody>
</table>
```

For odd parity systems, the total number of ones sent is odd.

Example:

```
Data send over "noisy" path should use error checking scheme.
```

Note: Parity doesn't detect if 2 bits flip erroneously.

Sending a "check sum" is another form of error detection.
Logic Gates

The Inverter

The inverter changes one logic level to the opposite level.

Traditional symbols:

- "0" means "inversion"

ANSI/IEEE Standard 91-1984 symbols:

- "0" means input is "active low" i.e. a logic zero causes specified function to happen.

Timing Diagram:

shows different signals on same horizontal time axis
Logic Expression for the Inverter:

\[ X = \overline{A} \text{  "bar" means complement or "A not"} \]

Example:

Binary number

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0
\end{array}
\]

1's Complement

The AND Gate

The output of an AND gate is high if and only if all of its inputs are high.

\[
\begin{array}{c}
A \\
B \\
X
\end{array}
\]

Output

\[
\begin{array}{ccc}
A & B & X \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1
\end{array}
\]

\[ N = 2^n \]

With \( n \) inputs there are \( 2^n \) possible combinations.

Example: \( n = 2 \)

\[ N = 2^2 = 4 \]

\[
\begin{array}{c}
A \\
B \\
C \\
X
\end{array}
\]

\[
\begin{array}{cccccccc}
A & B & C & X \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1
\end{array}
\]

\[ N = 2^3 = 8 \]
Pulsed Operation:

\[ \begin{align*}
A & \quad \uparrow \\
B & \quad \uparrow \\
X & \quad \uparrow
\end{align*} \]

- \( X \) is high only when both inputs \( A \) and \( B \) are high.

Application:

Frequency Counter

\[ \begin{align*}
A & \quad \uparrow \quad \text{Enable Signal} \\
\downarrow & \quad \text{\( \leq \) 15s.} \\
\uparrow & \quad \text{\( \geq \) 15s.} \quad \text{Reset Signal}
\end{align*} \]

- An AND gate can be used to block the flow of a pulse train when its "control" input is low.

EXIT Seat Belt Alarm System

- Ignition HIGH = ON, LOW = OFF
- Seat Belt HIGH = Buckled, LOW = Unbuckled

Audible Alarm

Timer \( \leq \) High for 30 sec. After ignition.
Logic Expressions for the AND Gate

\[ X = A \cdot B \]

Boolean expression for the AND function looks like multiplication symbol

\[ X = A \cdot B = AB \]

AND function

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X = AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 \cdot 0 = 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 \cdot 1 = 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 \cdot 0 = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1 \cdot 1 = 1</td>
</tr>
</tbody>
</table>
The OR Gate

The output of an OR gate is high if one or more of its inputs are high.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: OR gate uniquely selects the 0 0 input state.

Pulsed Operation:

X is low only when both inputs A and B are low.

Logic Expressions for the OR Gate:

\[ X = A + B \]

Boolean addition is the same as the OR function.
The **NAND Gate**

The output of a **NAND gate** is low if and only if all of its inputs are high.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

NAND gate uniquely selects all high input state
Pulsed Operation:

\[ A \]

\[ B \]

\[ X = t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 \]

A \textit{NAND} gate is equivalent to an \textit{OR} gate with inverters at its inputs.
A NAND gate can be made into an inverter by tying its inputs together:

\[
\begin{align*}
A \quad \text{NAND gate} & \quad \Rightarrow \quad \text{inverter} \\
A \quad \text{NAND gate} & \quad \Rightarrow \quad \text{inverter} \\
\end{align*}
\]

**Example** Design a system to indicate when both tanks are at least \( \frac{1}{4} \) full.

![Diagram of tank A with level sensor](Diagram)

**Diagram**

- Tank A
- Tank B
- Level sensor
- Green LED "ON" when both tanks above \( \frac{1}{4} \) full
- Level sensors = Hi when liquid present
- Sensors = Low when no liquid

**Example** Change above design to indicate when one of the tanks drops below \( \frac{1}{4} \) full

![Diagram of tank A with level sensor](Diagram)

**Diagram**

- Tank A
- Tank B
- Level sensors
- Red LED "ON" when any tank goes below \( \frac{1}{4} \) full
Logic Expressions for the NAND Gate

\[ X = \overline{AB} \]

\[ X = \overline{ABC} \]

Boolean expression for the NAND function looks like multiplication symbol with a complement (inversion) bar over it.

\[ X = \overline{A \cdot B} = \overline{AB} \]

NAND function

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( \overline{AB} ) = X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( \overline{0 \cdot 0} = \overline{0} = 1 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( \overline{0 \cdot 1} = \overline{0} = 1 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( \overline{1 \cdot 0} = \overline{0} = 1 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( \overline{1 \cdot 1} = \overline{1} = 0 )</td>
</tr>
</tbody>
</table>
The NOR Gate

The output of a NOR gate is high if and only if all of its inputs are low.

\[ A \quad B \quad X \equiv A \quad B \quad X \]

\[ \text{equivalent to} \quad \text{OR plus inverter} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Pulsed Operation

\[ A \quad B \quad X \]

A NOR gate is equivalent to a NAND gate with Inverters at its inputs.

\[ \text{NOR} \equiv \text{Negative-NAND} \]
Design a system to monitor the landing gear position of an airplane. A red light should be on if any of the gear is retracted. A green light should be on if all of the gear is extended.

**Logic Expressions for the NOR Gate**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A+B=X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0+0=0=1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0+1=1=0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1+0=1=0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1+1=1=0</td>
</tr>
</tbody>
</table>

A NOR gate can be converted to an inverter by tying its inputs together.
The **Exclusive-OR** and **Exclusive-NOR** Gates

**The Exclusive-OR** (EXOR)

![Diagram of the Exclusive-OR gate](image)

The output of the EXOR gate is high if input A or input B is high, but not both.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The XOR is high if A and B are different i.e. $A = \overline{B}$

**Pulsed Operation**

![Waveform diagrams for A, B, and X](image)
The Exclusive-NOR (XNOR)

The output of the XNOR gate is high if input A equals input B.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The XNOR is useful as an equality checker.

Example: Compare two 4-bit binary numbers to see if they are equal.

X = \overline{A \oplus B}

X = \text{high if } A = B
Integrated Circuit Logic Families

**TTL** - Transistor-Transistor Logic

- Uses bipolar junction transistors.
- Medium to high speed: 1.5 - 10 nsec.
- Medium power supply current: a few milliamps per gate.
- Requires 5 volt (±2.5V) supply voltage.
- 74 or 54 prefix in part numbers.

**CMOS** - Complementary Metal-Oxide Semiconductor

- Uses field-effect transistors.
- Slow to medium speed: 8 - 50 nsec.
- Very low power supply current: a few microamps per gate.
- Works over wide supply voltage range: 3 - 18 volts.
- High input resistance requires little current.
- Part numbers: 4000 Series - lowest current.
  - 74C
  - 74HC - higher speed.

**ECL** - Emitter Coupled Logic

- Very high speed: 1 nsec.
- High power supply current.
- Special purpose, higher cost functions.
Performance Characteristics

Propagation Delay Time

This is the time interval between the application of an input pulse and the occurrence of the resulting output pulse.

\[ t_{PHL} \rightarrow \rightarrow k \cdot t_{PLH} \]

The shorter the propagation delay time, the higher the maximum operating frequency.

Power Dissipation

This is the DC supply voltage (Vcc) times the average supply current (Icc).

In general, higher speed logic families dissipate more power and draw higher supply current than slower speed logic.
Fan-out

This is the maximum number of inputs that can be reliably driven from a gate output. (Assumes gates are of the same family)

![Diagram showing fan-out example]

A standard TTL NAND gate is rated to have a fan-out of 10.

Speed-power Product

It is a measure of performance found by multiplying the propagation delay by the gate's power dissipation.

Example: For a 74HC CMOS gate operating at 100 KHz.

\[ SSPP = (8 \text{ ns}) \times (0.17 \text{ mW}) = 1.36 \text{ pJoules} \]
A selection of typical logic gate ICs is now presented. These devices are commonly housed in the dual in-line package (DIP) or a surface-mount (SMT) package. For simplicity, $V_{CC}$ and ground connections to each gate are normally not shown in a logic diagram. On most 14-pin packages $V_{CC}$ is pin 14 and ground is pin 7. On most 16-pin packages $V_{CC}$ is pin 16 and ground is pin 8. Figure 3–51 shows typical 14-pin packages for pin numbering and size comparison. Although not shown to scale, you can see that the SOIC package is significantly smaller than the DIP. Notice that the dimensions are given in inches.

In Figures 3–52 through 3–57, each device is represented by a distinctive shape logic diagram, with the pin numbers indicated in parentheses. Additionally, each device is shown as a rectangular outline logic symbol. The two representations are equivalent. Although the standard 74 series designation is used for illustration, most of these devices are also available in most of the other TTL and CMOS series: 74LS, 74S, 74ALS, 74AS, 74HC, and 74HCT.

FIGURE 3–51
Typical dual in-line and small outline packages showing pin numbers and basic dimensions.
**Hex Inverter**  The 7404 hex inverter is a standard TTL device consisting of six inverters in a 14-pin package, as shown in Figure 3-52.

**AND Gates**  Several configurations of AND gates are available in IC form. The 7408 has four 2-input AND gates (it is called a quad 2-input AND); the 7411 has three 3-input AND gates (a triple 3-input AND); and the 7421 has two 4-input AND gates (a dual 4-input AND). These gates are shown in Figure 3-53.
FIGURE 3-54
NAND gates.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>5400</th>
<th>7400</th>
<th></th>
<th></th>
<th></th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage ((V_{CC}))</td>
<td>Minimum</td>
<td>Typical</td>
<td>Maximum</td>
<td>Minimum</td>
<td>Typical</td>
<td>Maximum</td>
</tr>
<tr>
<td></td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>4.75</td>
<td>5.0</td>
<td>5.25</td>
</tr>
<tr>
<td>Operating free-air</td>
<td>-55</td>
<td>25</td>
<td>125</td>
<td>0</td>
<td>25</td>
<td>70</td>
</tr>
<tr>
<td>temperature range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIGH level output</td>
<td>-400</td>
<td></td>
<td></td>
<td>-400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>current ((I_{OH}))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOW level output</td>
<td>16</td>
<td></td>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>current ((I_{OL}))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Recommended operating conditions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Limits</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum</td>
<td>Typical</td>
</tr>
<tr>
<td>HIGH level input voltage ((V_{IH}))</td>
<td>2.0</td>
<td>V</td>
</tr>
<tr>
<td>LOW level input voltage ((V_{IL}))</td>
<td>0.8</td>
<td>V</td>
</tr>
<tr>
<td>HIGH level output voltage ((V_{OH}))</td>
<td>2.4</td>
<td>3.4</td>
</tr>
<tr>
<td>LOW level output voltage ((V_{OL}))</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>HIGH level input current ((I_{IH}))</td>
<td>40</td>
<td>μA</td>
</tr>
<tr>
<td>LOW level input current ((I_{IL}))</td>
<td>-1.6</td>
<td>mA</td>
</tr>
<tr>
<td>Short-circuit output current ((I_{CM}))</td>
<td>5400</td>
<td>-20</td>
</tr>
<tr>
<td></td>
<td>7400</td>
<td>-18</td>
</tr>
<tr>
<td>Total supply current with</td>
<td>4.0</td>
<td>8.0</td>
</tr>
<tr>
<td>outputs HIGH ((I_{CMH}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total supply current with</td>
<td>12</td>
<td>22</td>
</tr>
<tr>
<td>outputs LOW ((I_{CM}))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Electrical characteristics over operating temperature range (unless otherwise noted)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Limits</th>
<th>Test Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation delay time, LOW-to-HIGH output ((\tau_{PLH}))</td>
<td>11</td>
<td>22</td>
</tr>
<tr>
<td>Propagation delay time, HIGH-to-LOW output ((\tau_{PHL}))</td>
<td>7.0</td>
<td>15</td>
</tr>
</tbody>
</table>

(c) Switching characteristics \((T_a = 25^\circ C)\)

NOTES:
1 For conditions shown as min. or max., use the appropriate value specified under recommended operating conditions for the applicable device type.
2 Typical limits are at \(V_{CC} = 5.0\, V\), 25°C
3 Not more than one output should be shorted at a time. Duration of short not to exceed 1 s.

FIGURE 3-58
Data sheet for the 5400/7400 quad 2-input NAND gate.
Troubleshooting Procedures

When an electronic system consisting of digital I.C.s does not produce the desired output, the following troubleshooting procedures should be performed.

1. Using the Oscilloscope probe, measure the power supply voltages at the Vcc pins of the I.C.s. If voltage incorrect, missing or noisy, check wiring connections and power supply. (Measure at I.C. pins to rule-out bad connections.)

2. Check for proper I.C. ground connections.

3. Check input signals to digital system.

4. Use signal tracing techniques. If an incorrect waveform is present at an output, work backwards by checking for correct signals at the I.C.s input.

5. If all of the input signals to an I.C. are correct and the output is incorrect:
   1) Most likely the I.C. is bad or 2) The I.C.s output is being loaded down excessively by a bad I.C. which it is driving.
Common Circuit Faults

Circuit connection faults:
1.) Open connection due to wires' insulation inserted too far into bread board socket.
2.) Open connection due to unsoldered or "cold" solder connection.
3.) Short circuit due to adjacent wires touching on bread board.
4.) Short circuit due to solder bridge between conductors.

Internal Failures of IC Logic Gates:
1.) Internally open input:
   a.) Acts like high on TTL
   b.) Can act high, low or oscillates on CMOS
2.) Internally shorted input:
   will be stuck high or low in spite of what I.C. driving it puts out.
3.) Internally shorted or open output:
   Will be stuck high, low or floating in between.
Boolean Operations and Expressions

A variable is a symbol used to represent a logical quantity. \( \text{ex} \) \( A \)

The complement is the inverse of a variable. \( \text{ex} \) \( \overline{A} \)

A literal is a variable or the complement of a variable. \( \text{ex} \) \( A \) or \( \overline{A} \)

Boolean Addition is equivalent to the OR operation.

\[
\begin{align*}
0 + 0 &= 0 \\
1 + 0 &= 1 \\
0 + 1 &= 1 \\
1 + 1 &= 1
\end{align*}
\]

A sum term is a sum of literals.

\( \text{ex} \) \( A + B \), \( \overline{A} + B + C + \overline{D} \)

\( \text{ex} \) Determine the values of \( A, B, C \) and \( D \) which make the sum term: \( A + B + C + \overline{D} = 0 \)

**Solution:** For a sum term to equal 0, all of its literals must be zero.

So: \( A = 0, \overline{B} = 0, C = 0 \) and \( \overline{D} = 0 \)

Therefore, \( A = 0, B = 1, C = 0, D = 1 \)
Boolean Multiplication is equivalent to the AND operation.

\[
\begin{align*}
0 \cdot 0 &= 0 \\
0 \cdot 1 &= 0 \\
1 \cdot 0 &= 0 \\
1 \cdot 1 &= 1
\end{align*}
\]

A product term is the product of literals.

\[\text{ex} / AB, \quad ABC \overline{C}\]

\[\text{ex} / \text{Determine the values of } A, B, C \text{ and } D \text{ which make the product term: } A\overline{B}C\overline{D} = 1\]

Solution: For a product term to equal 1, all of its literals must be 1.

So: \( A = 1, \overline{B} = 1, C = 1, \overline{D} = 1 \)

Therefore, \( A = 1, B = 0, C = 1, D = 0 \)

Laws and Rules of Boolean Algebra.

**Commutative Law:** \( A + B = B + A \)

\[
\begin{align*}
A + B &\equiv B + A \\
A \overline{B} &\equiv B \overline{A}
\end{align*}
\]
**Associative Laws:**

\[ A + (B + C) = (A + B) + C \]

\[ A \quad \quad A + (B + C) \quad \equiv \quad A \quad \quad A + B \quad (A + B) + C \]

\[ B \quad \quad B + C \]

\[ C \]

and \[ A (B C) = (A B) C \]

\[ A \quad \quad A (B C) \quad \equiv \quad A \quad \quad A B \quad (A B) C \]

\[ B \quad \quad B C \]

\[ C \]

**Distributive Law:**

Multiplication distributes into addition:

\[ A (B + C) = A B + A C \]

\[ B \quad \quad B + C \]

\[ C \]

\[ A \quad \quad A (B + C) \equiv \quad A \quad \quad A B \quad A B + A C \]

\[ C \]

\[ A \quad \quad A \quad \equiv \quad A \quad \quad A C \]

\[ C \]

\[ A \quad \quad A \quad \equiv \quad A \quad \quad A C \]

\[ C \]

**Note:** This gate implementation performs the same logic function with less gates.
Rules of Boolean Algebra

These rules are useful in manipulating and simplifying Boolean expressions.

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{A} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

Prove Rules by checking for all inputs:

1. $A + 0 = A$

2. $A + 1 = 1$

3. $A \cdot 0 = 0$

4. $A \cdot 1 = A$

5. $A + A = A$

6. $A + \overline{A} = 1$

7. $A \cdot A = A$

8. $A \cdot \overline{A} = 0$

9. $\overline{A} = A$

10. $A + AB = A$

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

12. $(A + B)(A + C) = A + BC$
5. \( A + A = A \)
\[
\begin{array}{c}
A = 1 \\
\bar{A} = 1
\end{array}
\rightarrow 
\begin{array}{c}
x = 1 \\
\bar{x} = 1
\end{array}
\begin{array}{c}
A = 0 \\
\bar{A} = 0
\end{array}
\rightarrow 
\begin{array}{c}
x = 0 \\
\bar{x} = 0
\end{array}
\]
\[x = A + A = A\]

6. \( A + \bar{A} = 1 \)
\[
\begin{array}{c}
A = 1 \\
\bar{A} = 0
\end{array}
\rightarrow 
\begin{array}{c}
x = 1 \\
\bar{x} = 1
\end{array}
\begin{array}{c}
A = 0 \\
\bar{A} = 1
\end{array}
\rightarrow 
\begin{array}{c}
x = 1 \\
\bar{x} = 1
\end{array}
\]
\[x = A + \bar{A} = 1\]

7. \( A \cdot A = A \)
\[
\begin{array}{c}
A = 1 \\
\bar{A} = 1
\end{array}
\rightarrow 
\begin{array}{c}
x = 1 \\
\bar{x} = 1
\end{array}
\begin{array}{c}
A = 0 \\
\bar{A} = 0
\end{array}
\rightarrow 
\begin{array}{c}
x = 0 \\
\bar{x} = 0
\end{array}
\]
\[x = A \cdot A = A\]

8. \( A \cdot \bar{A} = 0 \)
\[
\begin{array}{c}
A = 1 \\
\bar{A} = 0
\end{array}
\rightarrow 
\begin{array}{c}
x = 0 \\
\bar{x} = 0
\end{array}
\begin{array}{c}
A = 0 \\
\bar{A} = 1
\end{array}
\rightarrow 
\begin{array}{c}
x = 0 \\
\bar{x} = 0
\end{array}
\]
\[x = A \cdot \bar{A} = 0\]

9. \( \bar{A} = A \)
\[
\begin{array}{c}
A = 1 \\
\bar{A} = 0
\end{array}
\rightarrow 
\begin{array}{c}
\bar{x} = 0 \\
\bar{\bar{A}} = 1
\end{array}
\begin{array}{c}
A = 0 \\
\bar{A} = 1
\end{array}
\rightarrow 
\begin{array}{c}
\bar{x} = 0 \\
\bar{\bar{A}} = 0
\end{array}
\]
\[\bar{x} = A \rightarrow \bar{\bar{A}} = A\]

10. \( A + AB = A \)

Proof by Truth Table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>AB</th>
<th>A + AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Alternate Proof:

\[A + AB = A(1 + B)\text{ distr. law}
= A \cdot 1 \quad \text{since by rule 2}
= A \quad \text{since by rule 4}
= A \cdot 1 = A\]

\[\text{equal columns } \uparrow \quad \text{so } A + AB = A\]
11. $A + \overline{A}B = A + B$

Proof from earlier rules:

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

$$= (AA + AB) + \overline{A}B$$

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

$$= (A + \overline{A})(A + B)$$

$$= 1 \cdot (A + B)$$

$$= A + B$$

Reason

Rule 10: $A = A + AB$

Rule 7: $A = AA$

Rule 8: Adding $A\overline{A} = 0$

Factoring

Rule 6: $A + \overline{A} = 1$

Rule 4: drop the 1

Proof by truth table:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>$\overline{A}$</td>
<td>$\overline{A}B$</td>
<td>$A + \overline{A}B$</td>
<td>$A + B$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Since their columns are equal, then $A + \overline{A}B = A + B$
12. \((A+B)(A+C) = A + BC\)

**Proof from earlier rules:**

\[
(A+B)(A+C) = AA + AC + AB + BC \\
= A + AC + AB + BC \\
= A(1+C) + AB + BC \\
= A \cdot 1 + AB + BC \\
= A + BC \\
= A + BC
\]

**Reasons**

- Distributive Law
- Rule 7: \(A \cdot A = A\)
- Factoring (Distr. Law)
- Rule 2: \(1 + C = 1\)
- Factoring (Distr. Law)
- Rule 2: \(1 + B = 1\)
- Rule 4: \(A \cdot 1 = A\)

**Proof by truth table:**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A+B</th>
<th>A+C</th>
<th>((A+B)(A+C))</th>
<th>BC</th>
<th>A+BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(\therefore (A+B)(A+C) = A + BC\)
Demorgan's Theorems

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

\[
\overline{XY} = \overline{X} + \overline{Y}
\]

and

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

\[
\overline{X+Y} = \overline{X} \overline{Y}
\]

Proof and gate equivalence:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>\overline{XY}</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c|c|c|c|c}
   X & Y & \overline{XY} & \overline{X+Y} & \overline{XY} & \overline{X+Y} \\
---&---&---&---&---&---
   0 & 0 & 1 & 1 &   &   \\
   0 & 1 & 1 & 1 &   &   \\
   1 & 0 & 1 & 1 &   &   \\
   1 & 1 & 0 & 0 &   &   \\
\end{array}
\]

ex

Apply De Morgan's Thm: to: \( \overline{XYZ} \) and \( \overline{X+Y+Z} \)

Solution:

\[
\overline{XYZ} = \overline{X} + \overline{Y} + \overline{Z}
\]

\[
\overline{X+Y+Z} = \overline{X} \overline{Y} \overline{Z}
\]
Applying De Morgan's Theorems

De Morgan's Thms. can be also applied to groups of variables:

\[ A + BC + D(E+F) \]

ex) Apply De Morgan Thm. to:

1) Treat the terms: \( A + BC \) and \( D(E+F) \) each as single variables. \( A + BC = X \) and \( D(E+F) = Y \)

2) Apply DMT since \( X+Y = \overline{X} + \overline{Y} \), then

\[ \overline{A + BC + D(E+F)} = \overline{A + BC} \overline{D(E+F)} \]

3) Use Rule 9 (\( A + \overline{A} \)) to cancel double bars:

\[ = [A + BC][D(E+F)] \]

4) Let \( Z = E + F \):

\[ = [A + BC][\overline{DZ}] \]

5) Apply DMT: (since \( \overline{DZ} = \overline{D} + \overline{E} \))

\[ = [A + BC][\overline{D} + \overline{(E+F)}] \]

6) Use Rule 9 (\( \overline{A} = \overline{A} \)):

\[ = [A + BC][\overline{D} + (E+F)] \]

\[ = (A + BC)(\overline{D} + E + F) \]
Apply De Morgan's Thm. to: \((A+B+C)D\)

Solution: Let \((A+B+C) = X\) and \(D = Y\)
then since \(XY = X + Y\) we get:

\[
\overline{(A+B+C)D} = \overline{(A+B+C)} + \overline{D}
\]

Now apply DM1 to: \((A+B+C)\) to get

\[
= \overline{ABC} + \overline{D}
\]

Apply De Morgan's Thm. to: \(\overline{AB + CD + EF}\)

Solution:

\[
\overline{AB + CD + EF} = (\overline{AB})(\overline{CD})(\overline{EF})
\]

\[
= (\overline{A+B})(\overline{C+D})(\overline{E+F})
\]

\[
= (A+B)(C+D)(E+F)
\]

The Boolean expression for an XOR gate is: \(\overline{AB} + \overline{AB}\), Use De Morgan's Thm. to find an expression for an XNOR gate.

Solution: Complement the XOR expression and apply DM1

\[
\overline{\overline{AB} + \overline{AB}} = (\overline{\overline{AB}})(\overline{\overline{AB}})
\]

\[
= (\overline{A+B})(\overline{\overline{A+B}})
\]

\[
= (\overline{A+B})(A+B)
\]

\[
= \overline{AA + A\overline{B} + BA + B\overline{B}}
\]

\[
= \overline{AB} + BA
\]
Boolean Analysis of Logic Circuits

To derive a Boolean Expression of a given logic circuit, begin at the left-most inputs and work towards the final output, writing the expression for each gate.

Example:

\[ A(B + CD) \]

Evaluating the Expression: \( A(B + CD) \)

Consider the values of the variables which make the expression equal to 1:

It's 1 only if \( A = 1 \) and \( (B + CD) = 1 \)

Consider what makes \( (B + CD) = 1 \):

It's 1 if either \( B = 1 \) or \( CD = 1 \) or both \( B = 1 \) and \( CD = 1 \)

Consider what makes \( CD = 1 \):

It's 1 only if \( C = 1 \) and \( D = 1 \)

In summary:

The expression \( A(B + CD) = 1 \) when \( A = 1 \) and \( B = 1 \) regardless of the values of \( C \) and \( D \)

or when \( A = 1 \) and \( C = 1 \) and \( D = 1 \) regardless of the value of \( B \)

The expression \( A(B + CD) = 0 \) for all other combinations of the variables.
The evaluation of the expression can also be put in the form of a truth table.

Example: \( A(B + CD) \) has 4 variables and therefore require \( 2^4 \) possible input combinations listed.

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( D )</th>
<th>( A(B+CD) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A simplified Boolean expression uses the fewest gates possible to implement a given expression.

Example: Using Boolean algebra techniques, simplify this expression:

\[ AB + A(B+C) + B(B+C) \]

Solution:

1.) By distributive law:

\[ AB + AB + AC + BB + BC \]

2.) By rule 7: (BB = B)

\[ AB + AB + AC + B + BC \]

3.) By rule 5: (AB + AB = AB)

\[ AB + AC + B + BC \]

4.) By rule 10: (B + BC = B)

\[ AB + AC + B \]

5.) By rule 10: (AB + B = B)

\[ B + AC \]
Simplify the Boolean expression:

\[ (A \overline{B} (C + BD) + \overline{A} \overline{B}) \cdot C \]

Solution:

1. By distributive law:
\[ (A \overline{B} C + A \overline{B} BD + \overline{A} \overline{B}) \cdot C \]

2. By rule 8: \((\overline{B} B = 0)\)
\[ (A \overline{B} C + A \cdot 0 \cdot D + \overline{A} \overline{B}) \cdot C \]

3. By rule 3 \((A \cdot 0 \cdot D = 0)\)
\[ (A \overline{B} C + 0 + \overline{A} \overline{B}) \cdot C \]

4. By rule 1 (drop the 0)
\[ (A \overline{B} C + \overline{A} \overline{B}) \cdot C \]

5. By distributive law:
\[ A \overline{B} C C + \overline{A} \overline{B} C \]

6. By rule 7 \((C C = C)\)
\[ A \overline{B} C + \overline{A} \overline{B} C \]

7. Factor out \( \overline{B} C \):
\[ \overline{B} C (A + \overline{A}) \]

8. By rule 6 \((A + \overline{A} = 1)\)
\[ \overline{B} C \cdot 1 \]

9. By rule 4 (drop the 1)
\[ \overline{B} C \]
Simplify the Boolean Expression:
\[ \overline{A}BC + AB\overline{C} + \overline{A}\overline{B}\overline{C} + ABC + ABC \]

Solution:

1.) Factor out $BC$:
\[ BC(\overline{A} + A) + \overline{A}BC + A\overline{B}\overline{C} + ABC \]

2.) By rule 6 ($\overline{A} + A = 1$) also factor out $A\overline{B}$
\[ BC \cdot 1 + \overline{A}BC + A\overline{B}\overline{C} + ABC \]
\[ BC \cdot 1 + \overline{A}BC + A\overline{B}(\overline{C} + C) \]

3.) By rule 4 (drop 1) By rule 6 ($\overline{C} + C = 1$)
\[ BC + \overline{A}BC + A\overline{B}.1 \]

4.) By rule 4 (drop 1)
\[ BC + \overline{A}BC + A\overline{B} \]

5.) Factor out $\overline{B}$
\[ BC + \overline{B}(\overline{A}\overline{C} + A) \]

6.) By rule 11:
\[ (A + \overline{A}\overline{C} = A + \overline{C}) \]
\[ BC + \overline{B}(A + \overline{C}) \]

7.) By distributive law
\[ BC + A\overline{B} + \overline{B}\overline{C} \]
Exercise: Simplify the Boolean Expression:
\[ AB + AC + \overline{A} \overline{B} C \]

Solution:
1. By De Morgan's Thm: (DMT)
   \[ (\overline{AB})(\overline{AC}) + \overline{A} \overline{B} C \]
2. By DMT:
   \[ (\overline{A} + \overline{B})(\overline{A} + \overline{C}) + \overline{A} \overline{B} C \]
3. By distributive law:
   \[ \overline{A} \overline{A} + \overline{A} \overline{C} + \overline{A} \overline{B} + \overline{B} \overline{C} + \overline{A} \overline{B} \overline{C} \]
4. By rule 7 (\(\overline{AA} = \overline{A}\))
   By rule 10:
   \[ \overline{AB} + \overline{AB} \overline{C} = \overline{AB}(1 + \overline{C}) = \overline{AB} \]
5. By rule 10:
   \[ \overline{A} + \overline{A} \overline{C} = \overline{A}(1 + \overline{C}) = \overline{A} \]
6. By rule 10:
   \[ \overline{A} + \overline{A} \overline{B} = \overline{A}(1 + \overline{B}) = \overline{A} \]

\[ \boxed{\overline{A} + \overline{B} \overline{C}} \]
Standard Forms of Boolean Expressions

**Sum-of-Products (sop) Form**

When two or more product terms are summed by Boolean addition, the resulting expression is a sum-of-products.

\[ AB + ABC \]
\[ ABC + CDE + BCD \]
\[ \overline{AB} + \overline{ABC} + AC \]

**ex/** An sop expression can also contain a single-variable term:

\[ A + \overline{ABC} + BCD \]

**ex/** In an sop expression, a single over-bar cannot extend over more than one variable.

\[ \overline{ABC} \leftarrow \text{valid sop term} \]
\[ ABC \leftarrow \text{invalid sop term} \]

The **Domain** of a Boolean expression is the set of variables contained in it.

**ex/** Domain of \( \overline{A}B + A\overline{BC} \) is \( A, B, C \)
Implementation of an SOP Expression

Implementing an SOP expression simply requires ORing the outputs of two or more AND gates.

Example: Implement the expression $AB + BCD + AC$ with gates:

![Diagram](image)

$X = AB + BCD + AC$

Any logic expression can be changed to SOP form by applying the distributive law to any terms in parenthesis.

Example: Convert to SOP form

a) $AB + B(CD + EF) = AB + BCD + BEF$

b) $(A+B)(B+C+D) = AB + AC + AD + BB + BC + BD$

c) $\overline{(A+B)} + C = \overline{(A+B)} \overline{C} = (A+B) \overline{C} = A \overline{C} + B \overline{C}$
The Standard SOP Form

A standard SOP expression is one in which all the variables in the domain appear in each product term.

\[ \overline{ABC} + AB + A\overline{BC} \leq \text{not a standard SOP form} \]
\[ \overline{ABC} + ABC + \overline{AB}C \leq \text{a standard SOP form} \]

Converting Product Terms to Standard SOP

Step 1: Multiply each non-standard term by \( A + \overline{A} \) where \( \overline{A} \) is the missing variable. (note: \( A + \overline{A} = 1 \))

Step 2: Repeat step 1 until all terms contain all of the variables.

Example: Convert to standard SOP form:
\[ \overline{ABC} + \overline{A}B + A\overline{BCD} \]
Solution: Multiply first term by \( D + \overline{D} \):
\[ \overline{ABC}(D + \overline{D}) = \overline{ABCD} + \overline{ABC\overline{D}} \]
Multiply 2nd term by \( C + \overline{C} \):
\[ \overline{A}B(C + \overline{C}) = \overline{ABC} + \overline{A}BC \]
Repeat again:
\[ \overline{ABC} + \overline{A}BC = \overline{ABC}(D + \overline{D}) + \overline{ABC}(D + \overline{D}) = \overline{ABCD} + \overline{ABCD} + \overline{ABC\overline{D}} + \overline{A}BCD + \overline{A}BCD \]
Now take the results 1 and 2 from operating on the first two terms and list them together with the third term.

\[ \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C}\overline{D} + A\overline{B}CD + ABCD \]

Now it's in standard SOP form.

A standard product term is equal to one for only one combination of variable values.

ex/ The term \( \overline{A}B\overline{C}\overline{D} \) is equal to 1 when:

\[ A=1, B=0, C=1, D=0 \]

then \( \overline{A}B\overline{C}\overline{D} = 1 \cdot 0 \cdot 1 \cdot 0 = 1 \cdot 1 = 1 \)

An SOP expression is equal to 1 if and only if one or more of the product terms in the expression is equal to 1.

ex/ Determine the binary values for which the following standard SOP expression is equal to 1:

\[ ABCD + \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}D \]

Solution: The term \( ABCD \) is equal to 1 when:

\[ A=1, B=1, C=1, D=1 \]

The term \( \overline{A}\overline{B}C\overline{D} \) is equal to 1 when:

\[ A=1, B=0, C=0, D=1 \]

The term \( \overline{A}\overline{B}\overline{C}D \) is equal to 1 when:

\[ A=0, B=0, C=0, D=0 \]

The SOP expression equals 1 when any or all of the three product terms is 1.

i.e. when \( ABCD = 1111, 1001, 0000 \)
**Product-of-Sums (POS) Form**

When two or more sum terms are multiplied, the resulting expression is a product-of-sums.

\[ (\bar{A}+B)(A+B+C) \]
\[ (A+B)(A+B+C)(\bar{A}+C) \]

A POS expression can also contain a single-variable term:

\[ \bar{A} (A+B+C) (\bar{B}+\bar{C}+D) \]

In a POS expression, a single overbar cannot extend over more than one variable.

\[ \bar{A}+\bar{B}+\bar{C} \leq \text{valid POS term} \]
\[ \bar{A}+\bar{B}+\bar{C} \leq \text{invalid POS term} \]

**Implementation of a POS Expression**

Implementing a POS expression simply requires ANDing the outputs of two or more OR gates.

Implement the expression: \((A+B)(B+C+D)(A+C)\)

Solution:

\[ X = (A+B)(B+C+D)(A+C) \]
The Standard POS Form

A standard POS expression is one in which all the variables in the domain appear in each sum term.

Example:

\[(A + \overline{B} + C)(A + \overline{B} + \overline{C} + D) \neq \text{not a standard POS form}
\]

\[\text{no } \overline{D} \text{ variable in this term}\]

Example:

\[(A + \overline{B} + \overline{C} + \overline{D})(A + \overline{B} + \overline{C} + D) \neq \text{is a standard POS expression}\]

Converting a Sum Term to Standard POS

Step 1: Add to each non-standard product term \(A \cdot \overline{A}\) where \(\overline{A}^4\) is the missing variable (note: \(A \cdot \overline{A} = 0\))

Step 2: Apply rule 12: \(A + B C = (A + B)(A + C)\)

Repeat steps 1 and 2 until all product terms contain all variables.

Example:

Convert to standard POS form:

\[(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)\]

Solution:

Add \(D \cdot \overline{D}\) to first terms:

\[A + \overline{B} + C = A + \overline{B} + C + D \cdot \overline{D} \neq (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D}) \neq (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D}) \neq (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D}) \]

Add \(A \cdot \overline{A}\) to 2nd term:

\[\overline{B} + C + \overline{D} = \overline{B} + C + \overline{D} + A \cdot \overline{A} = (A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D}) \neq (A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D}) \]

Substitute results 1 and 2 into original expression:

\[(A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D})(A + \overline{B} + C + \overline{D})\]

Now its in standard POS form.
A standard sum term is equal to zero for only one combination of variable values.

Ex/ The term: \( A \cdot B \cdot C \cdot D \) is equal to 0 only when: \( A=0, B=1, C=0, D=1 \)

A POS expression is equal to 0 if and only if one or more of the sum terms in the expression is equal to 0.

Ex/ Determine the binary values of the variables for which the following standard POS expression is equal to 0:
\[(A+B+C+D)(\overline{A}+\overline{B}+\overline{C}+\overline{D})\]

Solution:
The term \( A \cdot B \cdot C \cdot D \) is equal to 0 when:
\( A=0, B=0, C=0, D=0 \)
The term \( A+\overline{B}+\overline{C}+D \) is equal to 0 when:
\( A=0, B=1, C=1, D=0 \)
The term \( \overline{A}+\overline{B}+\overline{C}+\overline{D} \) is equal to 0 when:
\( A=1, B=1, C=1, D=1 \)
The POS expression equals 0 when any of the three sum terms equals 0.

i.e. The expression equals 0 for the following input codes:

<table>
<thead>
<tr>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Converting Standard SOP to Standard POS

Step 1.) Determine the binary numbers which represent each of the product terms.

Step 2.) Determine all the possible binary numbers not included in step 1.

Step 3.) Write the equivalent sum term for each binary number from step 2, and express in POS form.

Example: Convert the SOP expression to POS form:
\[ \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + ABC \]

Solution:
\[
\begin{array}{cccc}
\bar{A} & \bar{B} & \bar{C} \\
1 & 1 & 1 \\
\end{array}
\quad \text{(bar = 0)}
\]

Step 1.) 000 + 010 + 011 + 101 + 111

Step 2.) With 3 digits there are \(2^3\) possible combinations. Those not included in step 1, are:
001, 100 and 110 \(\leftarrow\) These are the values that make the sum term 0

Step 3.) Write equivalent POS terms.
\[
\begin{array}{cccc}
001 & 100 & 110 & \bar{A} \bar{B} \bar{C} \\
\text{Y} & \text{Y} & \text{Y} & \text{Y} \\
\end{array}
\quad (1 = \text{bar})
\]

\( (A + B + \bar{C}) \quad (\bar{A} + B + C) \quad (A + \bar{B} + C) \)

A similar procedure can be used to go from POS to SOP.
Boolean Expressions and Truth Tables

All standard Boolean expressions can be easily converted into truth table format using binary values for each term in the expression.

Converting SOP expressions to Truth Table Format

1.) List all possible input combinations

2.) Convert SOP to standard form if necessary

3.) Place a 1 in the output column for each binary value that makes an SOP term 01/1, place a 0 in all remaining outputs

Example: Develop a truth table for the standard SOP expressions: \( \overline{A} \overline{B} \overline{C} + A \overline{B} \overline{C} + ABC \)

Solution: To make the following terms 1:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

93
Converting POS expressions to Truth Table Format

A POS expression is equal to 0 only if one of the sum terms is equal to 0.

1.) List all possible input combinations

2.) Convert POS to standard form if necessary

3.) Place a 0 in the output column for each binary value that makes a POS term a 0, place a 1 in all remaining outputs.

Example: Determine the truth table for the following standard POS expression:

\[(\bar{A} + B + C)(A + \bar{B} + C)(\bar{A} + \bar{B} + \bar{C})(\bar{A} + B + \bar{C})\]

Solution:

values to make each term = 0:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

Insert 0's for these codes.
Determining Standard Expressions from a Truth Table

To determine the standard SOP expression from a Truth Table:

1.) List the binary input codes for which the output = 1

2.) Convert these codes to product terms by replacing each 1 with the corresponding variable and each 0 with the complement of the corresponding variable.
   
   \[ \text{ex/ } 1010 \rightarrow A\overline{B}C\overline{D} \]

3.) Sum these product terms to make a SOP expression.

\[ \text{ex/ From the truth table determine a standard SOP expression:} \]

<table>
<thead>
<tr>
<th>Inputs ABC</th>
<th>Output X</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td></td>
<td>\rightarrow 011 \rightarrow \overline{A}BC</td>
</tr>
<tr>
<td>1 0 0</td>
<td></td>
<td>\rightarrow 100 \rightarrow A\overline{B}C</td>
</tr>
<tr>
<td>1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td></td>
<td>\rightarrow 110 \rightarrow ABC</td>
</tr>
<tr>
<td>1 1 1</td>
<td></td>
<td>\rightarrow 111 \rightarrow ABC</td>
</tr>
</tbody>
</table>

Therefore,

\[ X = \overline{A}BC + A\overline{B}C + ABC + ABC \]
To determine the **standard POS expressions** from a Truth Table:

1.) List the binary input codes for which **output = 0**

2.) Convert these codes to sum terms by replacing each **0** with the corresponding **variable** and each **1** with the complement of the corresponding variable.
   
   **ex:** $1001 \rightarrow \bar{A} + B + C + \bar{D}$

3.) Multiply these sum terms to make a POS expression.

**ex:** From the truth table determine a standard POS expression:

<table>
<thead>
<tr>
<th>Inputs A BC</th>
<th>Output X</th>
<th>Solution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>$000 \rightarrow A + B + C$</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>$001 \rightarrow A + B + \bar{C}$</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>$010 \rightarrow A + \bar{B} + C$</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>$101 \rightarrow \bar{A} + B + \bar{C}$</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Therefore,

\[ X = (A + B + C)(A + B + \bar{C})(A + \bar{B} + C)(\bar{A} + B + \bar{C}) \]
The Karnaugh Map

The Karnaugh map provides a systematic method for simplifying Boolean expressions.

The 3-Variable K-Map

A 3-Variable K-Map consists of an array of 8 cells.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note sequence

The 4-Variable K-Map

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Standard product terms represented by each cell
Cell Adjacency

The cells in a k-map are arranged so that there is only a single-variable change between adjacent cells.

Adjacency is defined by a single variable change.

Arrows indicate adjacent cells.

Note the "wrap around" adjacency between the cells at opposite ends of each row and column.
Karnaugh Map SOP Minimization

Mapping a Standard SOP Expression:
A 1 is placed in a K-map cell for each product term in the expression.

Step 1) Determine the binary value of each product term in the expression.
Step 2) Place a 1 in the corresponding K-map cell for each product term.

Map the expression:

$$
\overline{ABC} + \overline{ABC} + ABC + \overline{ABC}
$$

Example:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Mapping a Nonstandard SOP Expression:
The Boolean expression must first be put in standard form (no incomplete terms) before mapping.

Numerical Expansion of a Non-standard Product Term:
1) Write the binary value of the non-standard term.
2) Attach additional binary digits to construct all possible combinations of new product terms.

Example: For 3-variable map with term $A\overline{B}$
1) $A\overline{B} \rightarrow 10$
2) Add missing C variable: $101 \rightarrow A\overline{B}C$

99
For a 3-variable map with non-standard term B.

1.) $B \rightarrow 1$

2.) $010 \rightarrow \overline{A} \overline{B} \overline{C}$

Attach

$011 \rightarrow ABC$

A and C

2 digits

$110 \rightarrow ABC$

with all possible values

required additional SOP terms to convert expression to standard SOP form

**Ex:** Map the SOP expression: $\overline{A} + AB + ABC$

**Solution:** It's not in standard form because each product term does not have all 3 variables.

First expand the incomplete terms:

$\overline{A} + AB + ABC$

000 100 110

001 101

010

011 1 additional digit attached

2 additional digits attached

<table>
<thead>
<tr>
<th>$AB$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>(</td>
</tr>
<tr>
<td>11</td>
<td>)</td>
</tr>
<tr>
<td>10</td>
<td>(</td>
</tr>
<tr>
<td>11</td>
<td>(</td>
</tr>
</tbody>
</table>

100
Example: Map the SOP expression:
\[ \overline{BC} + \overline{AB} + ABC + A \overline{BCD} + \overline{ABCD} + A \overline{BCD} \]

Solution: It's not in standard form because each product term does not have all 3 variables.
First expand the incomplete terms:

\[ \overline{BC} + \overline{AB} + ABC + A \overline{BCD} + \overline{ABCD} + A \overline{BCD} \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Place 1's in cells for each term present. (Some are redundant)
2 additional digits attached
1 additional digit attached
Karnaugh Map Simplification of SOP Expressions

Minimization will result in the fewest possible terms with the fewest possible variables.

The 3 steps in minimization are: grouping the 1's, determining the product term for each group, and summing the resulting product terms.

Grouping the 1's:

1.) A group must contain 1, 2, 4, 8 or 16 cells.

2.) Each cell in a group must be adjacent to at least one other cell in the group.

3.) Make the groups as large as possible.

4.) Each 1 on the map must be included in at least 1 group. The 1's already in a group can be included in another group as long as the overlapping groups include non-common 1's.

Example:

\begin{array}{c|cc}
  & 0 & 1 \\
\hline
  00 & 1 \\
  01 & & \\
  11 & & \\
  10 & & \\
\end{array}

\begin{array}{c|cc}
  & 0 & 1 \\
\hline
  00 & 1 \\
  01 & & \\
  11 & & \\
  10 & & \\
\end{array}

Red group of 4 Cells uses "Wrap Around" adjacency
Determining the Minimum SOP Expression from the Map:

After grouping the 1's follow these rules:

1.) Each group of cells containing 1's creates one product term composed of all variables that occur in only one form (either complemented or non complemented) within the group. Variables that occur in both form (i.e. \( A \) and \( \overline{A} \)) within the group are eliminated. Those are called contradictory variables.

2.) Determine the minimum product terms for each group:
   a.) For a 3-variable map:
      A 1-cell group yields a 3-variable product term
      - 2-cell
      - 4-cell
      - 8-cell
      - Value of 1 for the expression

   b.) For a 4-variable map:
      A 1-cell group yields a 4-variable product term
      - 2-cell
      - 4-cell
      - 8-cell
      - 16-cell
      - Value of 1 for the expression

3.) When all of the minimum product terms are derived from the K-map they are summed to form a minimum SOP expression.

\[
\begin{array}{c|cc|c}
 & 0 & 1 & \overline{A} \overline{B} \\
\hline
0 & 1 & & \\
1 & & 1 & \\
1 & & & \\
0 & & & \\
\end{array}
\]

- Discard A since \( A \) and \( \overline{A} \) are in the group
- Discard C since C and \( \overline{C} \) are in the group

Minimized expression is:
\[
AB + BC + \overline{A} \overline{B} \overline{C}
\]
Use a K-map to minimize the standard SOP expression: \( \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} + \overline{ABC} \)

**Solution:**

1. Get binary values: \(101 + 011 + 000 + 100 + 001\)
2. Route K-map: 3.) group 1's 4.) determine terms from groups

**minimized expression is:** \( \overline{B} + \overline{AC} \)
Use a K-map to minimize the SOP expression

\[ \overline{BCD} + \overline{A} \overline{BCD} + ABCD + \overline{A} \overline{BCD} + \overline{A} \overline{BCD} + ABCD + \overline{ABC} D + A \overline{BCD} + A \overline{BCD} + \overline{ABC} D \]

Solution:

0000 0000 1100 0011 1011 0010 0110 1110 1010

\[ D + \overline{BC} \]

minimized expression is: \( D + \overline{BC} \)
Mapping Directly from a Truth Table:

The 1's in the output column are mapped directly onto a K-map into the cells corresponding to their associated input variable combinations.

Example: Map the truth table on a K-map

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>X</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The expression corresponding to the truth table is:

\[ X = \overline{A} \overline{B} \overline{C} + A \overline{B} \overline{C} + A \overline{B} \overline{C} + A B C \]

The minimized expression read from the K-map is:

\[ X = \overline{A} \overline{B} \overline{C} + \overline{A} \overline{C} + A B \]
"Don't Care" Conditions

Sometimes a situation arises in which some input variable combinations are not allowed. Since these input combinations will never occur in this application, they can be treated as "don't care" terms with respect to their effect upon the output.

For these "don't care" terms either a 1 or a 0 may be assigned to the output; it really doesn't matter since they will never occur.

The "don't care" terms are mapped as X's on the K-map and can be used as 1's or 0's when grouping (which is more convenient).

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B C D</td>
<td>X</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{without "don't cares": } X = \overline{A}\overline{B}C + \overline{A}BCD \\
\text{with "don't cares": } X = A + BCD
\]
The 7-Segment Display

7 segments are used to display the decimal digits 0 thru 9.

LED displays: emit light.
- Visible in dark.
- Draw relatively high current
- Not visible in bright light.

LCD (Liquid Crystal Displays): reflect light
- Draw very little current.
- Not visible in dark
- Requires a square wave backplane driver voltage.

Segment Decoding:

Example: Segment "a" in "on" to display digits:
0, 2, 3, 5, 6, 7, 8, 9

Diagram:

**BCD** input: A, B, C, D

**7-Seg. Decoding Logic**

Output segments: a, b, c, d, e, f, g
From the truth table:

\[
\begin{array}{cccccccc}
\text{digit} & 0 & 2 & 3 & 5 & 6 & 7 & 8 & 9 \\
BC0 & 0000 & 0010 & 0011 & 0101 & 0110 & 0111 & 1000 & 1001
\end{array}
\]

\[\alpha = \overline{DCBA} + \overline{DCBA} + \overline{DCBA} + \overline{DCBA} + \overline{DCBA} + \overline{BCDA} + \overline{DCBA} + \overline{DCBA}
\]

In order to minimize a decoder design for segment \(\alpha\), a K-map is used.

\[\alpha = D + B + CA + \overline{CA}\]
Special Combinational Logic Circuits

**AND-OR Logic**

ex/ The 74 HC 58 AND-OR IC

\[ Y = AB + CD \]

\[ \text{AND-OR Logic directly implements SOP expressions} \]

**AND-OR Invert Logic**

ex/ The 74L551 AND-OR Invert IC

\[ Y = \overline{AB} + \overline{CD} \]

\[ Y = \overline{AB} \overline{CD} \]

\[ Y = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) \]

\[ \text{AND-OR Invert Logic directly implements POS expressions} \]
Implementing Combinational Logic

Example 1: Implement the Boolean Expression:

\[ X = AB + CDE \]

Solution:

\[ X = AB + CDE \]

Example 2: Implement

\[ X = AB(CD+EF) = ABCD + ABEF \]

SOP Implementation

Example 3: From Truth Table Implement Function:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>X</td>
</tr>
<tr>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>( \overline{ABC} )</td>
</tr>
<tr>
<td>100</td>
<td>( \overline{AB} \overline{C} )</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
</tr>
</tbody>
</table>

In SOP

\[ X = \overline{ABC} + \overline{AB} \overline{C} \]
All of the other logic gates can be implemented using only NAND gates.

\[ \overline{\overline{A}} = A \]

\[ A \overline{B} = \overline{AB} \]

\[ \overline{A+B} = \overline{A} \overline{B} \]

All of the other logic gates can be implemented using only NOR gates.

\[ \overline{\overline{A}} = A \]

\[ A \overline{B} = \overline{AB} \]

\[ \overline{A+B} = \overline{A} \overline{B} \]
A NAND gate can also function as a negative-OR by De Morgan's theorem:

\[
\overline{AB} = \overline{A} + \overline{B}
\]

NAND \quad \text{negative-OR}

Consider the circuit:

\[
x = \overline{(AB)(CD)} = \overline{(\overline{A} + \overline{B})(\overline{C} + \overline{D})} = \overline{(\overline{A} + \overline{B}) + (\overline{C} + \overline{D})}
\]

NAND gates implement AND-OR function (sop)

This can also be shown using NAND's negative or equivalent:

\[
x = AB + CD
\]

\[
\Rightarrow
\]

\[
\text{bubbles cancel}
\]

\[
\text{and}
\]

\[
\text{are dual symbols for the NAND gate}
\]

By using dual symbols for the NAND gate in such a way as to cancel bubbles, a logic diagram's output can be more easily found.
NAND Logic Diagrams

To simplify finding a NAND gate array's output, use the negative-OR symbol at every-other level in the diagram to maximize bubble cancellation opportunities.

\[ x = (\overline{ABC}D)EF \]
\[ = \overline{ABC}D + EF \]
\[ = (AB + \overline{C})D + EF \]

Or by "bubble cancellation" method:

Implement the expression: \( ABC + \overline{D} + \overline{E} \) with NAND gates.

Solution:

\[ x = ABC + \overline{D} + \overline{E} \]
A NOR gate can also function as a negative-AND by De Morgan's theorem:
\[ \overline{A + B} = \overline{A} \overline{B} \]

**NOR** \( \downarrow \) \text{negative-AND}

Consider the circuit:

\[ X = (A + B)(C + D) \]

NOR gates can implement OR-AND function (POS)

This can also be shown using NORE's negative-AND equivalent:

\[ (A + B)(C + D) \Rightarrow \]

NOR gates can implement OR-AND function (POS)

\[ (A + B)(C + D) \Rightarrow \]

NOR and \( \overline{\text{neg.-AND}} \) are dual symbols for the NOR gate

These dual symbols can be use interchangeably to simplify determining a logic diagram's output. (similar to NAND and neg.-OR)
Basic System Operation

The system uses four manual on/off switches, control logic, and motor drive interface to control the conveyor lubrication pump motor, the conveyor motor, the cross-cut saw motor, and the band saw motor, as indicated by the block diagram in Figure 5–44. This system application focuses on the control logic portion of the system.

![Block diagram](image)

**FIGURE 5–44**

Basic block diagram for the control system.

**Operational Requirements**  Switch input $S_1$ controls the lubrication pump motor (output $M_1$). Switch input $S_2$ controls the conveyor motor (output $M_2$). Switch input $S_3$ controls the band saw motor (output $M_3$). Switch input $S_4$ controls the cross-cut saw motor (output $M_4$).

The motor that provides for lubrication of the conveyor must be on ($M_1 = 1$) when the conveyor is on. The motor that drives the conveyor is on ($M_2 = 1$) only when both switch 1 and switch 2 are on ($S_1 = 1$ and $S_2 = 1$). The band saw motor is on ($M_3 = 1$) when switch 3 is on ($S_3 = 1$), and the cross-cut saw motor is on ($M_4 = 1$) when switch 4 is on ($S_4 = 1$). The band saw and cross-cut saw motors require no lubrication, but they must never be on at the same time. If switches 3 and 4 are both turned on at the same time, the system must be completely shut down, including the conveyor and lubrication motors. Also, the cross-cut saw and the conveyor cannot be on at the same time. The control logic controls the motors to prevent any of the unallowed conditions from occurring should the switches be improperly operated.

**Truth Table for the Control Logic**

Table 5–6 is a truth table that shows the states of the switches and motors based on the conditions previously described. A 1 indicates the ON state and a 0 indicates the OFF state. Since there are four switches, there are sixteen possible ON/OFF combinations of the
### Truth Table for the Control Logic

<table>
<thead>
<tr>
<th>Inputs</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Shut down</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>OK</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (2 saws on). Shut down</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (No lub with conveyor). Shut down</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (conveyor and cross cut). Shut down</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (No lub with conveyor). Shut down</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (2 saws and conveyor). Shut down</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>OK</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 0</td>
<td>OK</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>OK</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (2 saws). Shut down</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>OK</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (conveyor and cross cut). Shut down</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1</td>
<td>1 0</td>
<td>1 0</td>
<td>0 0</td>
<td>OK</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>Unallowed (all on). Shut down</td>
</tr>
</tbody>
</table>

Switches and, as a result, sixteen possible ON/OFF combinations for the motors. The states of the switches are the input variables and the states of the motors are the output variables. The unallowed conditions in this case are not treated as "don’t cares." They are switch input conditions that should not occur but if they do, the system must be shut down with all motors off.

### Design of the Control Logic

There are four separate logic circuits, one for each of the motors. Let's begin by designing the logic circuit for the lubrication pump motor (output $M_1$). The first step is to transfer the data from the truth table to a Karnaugh map and develop an SOP expression.

The switch variables $S_1, S_2, S_3$, and $S_4$ are map variables and the states of $M_1$ are plotted and grouped as shown in Figure 5-45(a). The 0s on the map are for switch conditions when the motor is off and the 1s are for switch conditions when the motor is on. The resulting SOP expression for the lubrication pump motor logic results in the NAND implementation shown in part (b).

**FIGURE 5-45**

Karnaugh map simplification and implementation for the lubrication motor logic.

\[
M_1 = S_1 \overline{S}_2 + \overline{S}_1 S_2
\]
Latches

A latch is a type of bistable multivibrator. It has memory capacity to store 1 bit of data. Multivibrators have regenerative feedback.

The S-R Latch

When the SET (S) input is activated the output Q will go HIGH.
When the RESET (R) input is activated the output Q will go LOW.

Active-High input
S-R Latch

Active-Low input
S-R Latch

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

ex/ It is only necessary to provide a pulse to the S input ( ) to set the latch. The latch "remembers" the pulse after the pulse is gone.
The Latch as a Contact-Bounce Eliminator

Alternative bounce eliminator which only requires SPST switch.

\[ t = R \cdot C > 10 \text{ msec.} \]

For CMOS let \( R = R_1 = 100 \text{ k}\Omega \)

Let \( t = 100 \text{ msec} = R_1 \cdot C \Rightarrow \]

\[ C = \frac{100 \text{ msec}}{100 \text{ k}\Omega} = 1 \mu\text{f} \]
The 74LS279 Quad S-R Latch

Inputs
1s1
1s2
1R
2s
2R
3s1
3s2
3R
4s
4R

Outputs
1Q
2Q
3Q
4Q

The Gated S-R Latch

S
EN
R
Q

The ENABLE input (EN) must be HIGH for the S or R input information to reach the latch.

Example waveforms:

S
R
EN
Q
The Gated D Latch

The D (DATA) Latch will load the value on its D input into its storage register when its EN (ENABLE) input is HIGH. When the EN input is low it will ignore its D input.

The Q output follows the D input when EN is HIGH.

Tristate Output

Many ICs are available with a Tristate Output feature. With Tristate Output an IC's output can be switched "OFF" (Placed in a high impedance or open circuit mode) by placing a ZERO on its OE (Output Enable) input. This allows several different IC outputs to share a common data bus.

Ex/

<table>
<thead>
<tr>
<th>OE</th>
<th>D</th>
<th>EN</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Q0</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>HighZ</td>
</tr>
</tbody>
</table>

Outputs multiplexed to Common Data bus
The 74HC373 Octal Latch

This IC contains 8 gated D latches. It also features tristate output.

<table>
<thead>
<tr>
<th>OE</th>
<th>LE</th>
<th>Dn</th>
<th>Qn</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>X</td>
<td>Q_0</td>
</tr>
<tr>
<td>H</td>
<td>X</td>
<td>X</td>
<td>Z</td>
</tr>
</tbody>
</table>

OE is an "output enable" input which allows the Q outputs to share a common data bus with other ICs.

LE is a "latch enable" input which allows the D inputs selectively sample data from a common data bus.

The 74HC373 can effectively Multiplex and Demultiplex data lines in a digital system.
**Edge-Triggered Flip-Flops**

A flip-flop is a synchronous bistable device.

An edge-triggered flip-flop changes state either at the positive (rising) edge or at the negative (falling) edge of the clock pulse and is sensitive to its inputs only at this transition of the clock.

**Example:**

- S-R Flip-Flop
- D Flip-Flop
- J-K Flip-Flop

Synchronous S-R F.F.

**Truth Table for Positive edge-triggered S-R F.F.**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Output changes are only permitted on rising edge of clock pulse

A Method of Edge-Triggering

Positive Edge-Triggered Pulsen

Timing Diagram:

\[ \tau = \text{propagation delay time due to inverter.} \]

Circuit will produce a short output pulse at each rising edge of the CLK input.
Positive Edge Triggered Pulser

Using CMOS I.C.

\[ R \]
\[ C \]

Output pulse width controlled by RC time constant.
\[ \tau \approx RC \]

For CMOS: \( 0 \leq R \leq 1 \text{ Megohm} \)

For \( R = 100 \text{ k}\Omega \) and \( C = 0.1 \mu\text{F} \)
\[ \tau = \left( 100 \times 10^3 \right) \left( 0.1 \times 10^{-6} \right) = 10 \text{ milliseconds} \]

Clock SR Flip-Flop

The S and R inputs will only be acted upon at the instant when the clock input rises from a 0 to a logic 1.
The Edge-Triggered D Flip-Flop

It is used to "latch-on-to" or store a single bit of data. It loads the data on the edge of the clock input.

Truth Table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Q, Q̅</td>
</tr>
<tr>
<td>CLK</td>
<td></td>
</tr>
</tbody>
</table>

1 ↑ 1 0 ← sets output
0 ↑ 0 1 ← resets output

indicates positive-edge triggering

Example waveforms:

CLK: 1 0 1 0 1 0 1 0
D: 1 0 1 0 1 0 1 0
Q: 0 1 0 1 0 1 0 1
DUAL PRECISION RETRIGGERABLE/RESETTABLE MONOSTABLE MULTIVIBRATOR

The MC14538B is a dual, retriggenable, resettable monostable multivibrator. It may be triggered from either edge of an input pulse, and will produce an accurate output pulse over a wide range of widths, the duration and accuracy of which are determined by the external timing components, Cx and Rx. Linear CMOS techniques allow more precise control of output pulse width.

- ±1.0% Typical Pulsewidth Variation from Part to Part
- ±0.5% Typical Pulsewidth Variation over Temperature Range
- New Formula: T = RC (T in seconds, R in ohms, C in farads)
- Pulse Width Range = 10 μs to 10ms
- Symmetrical Output Sink and Source Capability
- Latched Trigger Inputs
- Separate Latched Reset Inputs
- Quiescent Current (Standby) = 5.0 nA/package typical @ 5 Vdc
- 3.0 Vdc to 18 Vdc Operational Limits
- Triggerable from Positive or Negative-Going Edge
- Capable of Driving Two Low-power TTL Loads, One Low-power Schottky TTL Load or Two HTL Loads Over the Rated Temperature Range
- Pin-for-pin Compatible with MC14528B and CD4528B (CD4098)
- For Pulse Widths Less Than 10 μs the MC14528B is Recommended

MAXIMUM RATINGS (Voltages referenced to VSS)

<table>
<thead>
<tr>
<th>Rating</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC Supply Voltage</td>
<td>VDD</td>
<td>-0.5 to +15</td>
<td>Vdc</td>
</tr>
<tr>
<td>Input Voltage, All Inputs</td>
<td>Vin</td>
<td>-0.5 to VDD + 0.5</td>
<td>Vdc</td>
</tr>
<tr>
<td>DC Current Drain per Pin</td>
<td>I</td>
<td>10</td>
<td>mA</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>TA</td>
<td>-55 to +125</td>
<td>°C</td>
</tr>
<tr>
<td>CL/CP Device</td>
<td></td>
<td>-40 to +85</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>Tstg</td>
<td>-65 to +150</td>
<td>°C</td>
</tr>
</tbody>
</table>

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. For proper operation it is recommended that Vin and Vout be constrained to the range VSS ≤ Vin, Vout ≤ VDD.

Unused inputs must always be tied to an appropriate logic voltage level (e.g., either VSS or VDD).
**ELECTRICAL CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>VDD</th>
<th>$T_{\text{low}}^*$</th>
<th>25°C</th>
<th>$T_{\text{high}}^*$</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Vdc</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Typ</td>
</tr>
<tr>
<td>Output Voltage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;0&quot; Level</td>
<td>VOL</td>
<td>5.0</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>$V_{\text{in}} = V_{\text{DD}}$ or 0</td>
<td></td>
<td>10</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>&quot;1&quot; Level</td>
<td>VOH</td>
<td>5.0</td>
<td>4.95</td>
<td>4.95</td>
<td>5.0</td>
<td>4.95</td>
</tr>
<tr>
<td>$V_{\text{in}} = 0$ or $V_{\text{DD}}$</td>
<td></td>
<td>10</td>
<td>9.95</td>
<td>9.95</td>
<td>10</td>
<td>9.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>14.95</td>
<td>14.95</td>
<td>15</td>
<td>14.95</td>
</tr>
<tr>
<td>Input Voltage#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(VO = 4.5 or 0.5 Vdc)</td>
<td>VIL</td>
<td>5.0</td>
<td>1.5</td>
<td>2.25</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>(VO = 9.0 or 1.0 Vdc)</td>
<td></td>
<td>10</td>
<td>3.0</td>
<td>4.50</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>(VO = 13.5 or 1.5 Vdc)</td>
<td></td>
<td>15</td>
<td>4.0</td>
<td>6.75</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>&quot;1&quot; Level</td>
<td>VIH</td>
<td>5.0</td>
<td>3.5</td>
<td>3.5</td>
<td>2.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>7.0</td>
<td>7.0</td>
<td>5.50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>11.0</td>
<td>11.0</td>
<td>8.25</td>
<td></td>
</tr>
<tr>
<td>Output Drive Current (AL Device)</td>
<td>IOH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(VOH = 2.5 Vdc) Source</td>
<td></td>
<td>5.0</td>
<td>-3.0</td>
<td>-2.4</td>
<td>-4.2</td>
<td>-1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>-0.64</td>
<td>-0.51</td>
<td>-0.88</td>
<td>-0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>-1.6</td>
<td>-1.3</td>
<td>-2.25</td>
<td>-0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>-4.2</td>
<td>-3.4</td>
<td>-8.8</td>
<td>-2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0.64</td>
<td>0.51</td>
<td>0.88</td>
<td>0.36</td>
</tr>
<tr>
<td>Output Drive Current (CL/CP Device)</td>
<td>IOL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(VOH = 2.5 Vdc) Source</td>
<td></td>
<td>5.0</td>
<td>-2.5</td>
<td>-2.1</td>
<td>-4.2</td>
<td>-1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>-0.52</td>
<td>-0.44</td>
<td>-0.88</td>
<td>-0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>-1.3</td>
<td>-1.1</td>
<td>-2.25</td>
<td>-0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>-3.6</td>
<td>-3.0</td>
<td>-8.8</td>
<td>-2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0.52</td>
<td>0.44</td>
<td>0.88</td>
<td>0.36</td>
</tr>
<tr>
<td>Input Current, Pin 2 or 14</td>
<td>IIN</td>
<td>15</td>
<td>-0.02</td>
<td>-0.00001</td>
<td>0.05</td>
<td>-0.5</td>
</tr>
<tr>
<td>Input Current, Other Inputs (AL Device)</td>
<td>IIN</td>
<td>15</td>
<td>-0.1</td>
<td>-0.00001</td>
<td>0.1</td>
<td>-1.0</td>
</tr>
<tr>
<td>Input Current, Other Inputs (CL/CP Device)</td>
<td>IIN</td>
<td>15</td>
<td>-0.3</td>
<td>-0.00001</td>
<td>0.3</td>
<td>-1.0</td>
</tr>
<tr>
<td>Input Capacitance, Pin 2 or 14</td>
<td>Cin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>Input Capacitance, Other Inputs (VO = 0)</td>
<td>Cin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.0</td>
</tr>
<tr>
<td>Quiescent Current (AL Device)</td>
<td>IDD</td>
<td>5.0</td>
<td>5.0</td>
<td>0.005</td>
<td>5.0</td>
<td>150</td>
</tr>
<tr>
<td>(Per Package)</td>
<td></td>
<td>10</td>
<td>10</td>
<td>0.010</td>
<td>10</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>20</td>
<td>0.015</td>
<td>20</td>
<td>600</td>
</tr>
<tr>
<td>Quiescent Current (CL/CP Device)</td>
<td>IDD</td>
<td>5.0</td>
<td>20</td>
<td>0.005</td>
<td>20</td>
<td>150</td>
</tr>
<tr>
<td>(Per Package)</td>
<td></td>
<td>10</td>
<td>40</td>
<td>0.010</td>
<td>40</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>80</td>
<td>0.015</td>
<td>80</td>
<td>600</td>
</tr>
<tr>
<td>Quiescent Current, Active State</td>
<td>IDD</td>
<td>5.0</td>
<td>35</td>
<td>80</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>(Q1 = Logic 1)</td>
<td></td>
<td>10</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Q2 = Logic 0)</td>
<td></td>
<td>15</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Total Supply Current at an external load capacitance ($C_L$) and at external timing network ($R_X$, $C_X$)**

<table>
<thead>
<tr>
<th></th>
<th>IT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>(3.5 \times 10^{-2}) R_X C_X f + 4 C_X f + 1 \times 10^{-5} C_L f</td>
</tr>
<tr>
<td></td>
<td>(8 \times 10^{-2}) R_X C_X f + 9 C_X f + 2 \times 10^{-5} C_L f</td>
</tr>
<tr>
<td></td>
<td>(1.25 \times 10^{-1}) R_X C_X f + 12 C_X f + 3 \times 10^{-5} C_L f</td>
</tr>
</tbody>
</table>

Where:
- $f$ in Hz is the input frequency.
- $C_X$ in μF, $C_L$ in pF, $R_X$ in k ohms.

---

* $T_{\text{low}}^* = -55^\circ\text{C}$ for AL Device, $-40^\circ\text{C}$ for CL/CP Device.
  $T_{\text{high}}^* = +125^\circ\text{C}$ for AL Device, $+85^\circ\text{C}$ for CL/CP Device.
* Noise immunity specified for worst-case input combination.
* Noise Margin both "1" and "0" level = 1.0 Vdc min @ VDD = 5.0 Vdc
  2.0 Vdc min @ VDD = 10 Vdc
  2.5 Vdc min @ VDD = 15 Vdc
* The formulas given are for the typical characteristics only at 25°C.
### SWITCHING CHARACTERISTICS  \( (C_L = 50\, \text{pF}, \, T_A = 25^\circ\text{C}) \)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Symbol</th>
<th>( V_{DD} )</th>
<th>( V_{dc} )</th>
<th>All Types</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Rise Time</td>
<td>( T_{\text{TLH}} )</td>
<td>5.0</td>
<td>-</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>( T_{\text{THL}} = (1.35 , \text{ns}) ) ( C_L + 33 , \text{ns} )</td>
<td></td>
<td>10</td>
<td>-</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>( T_{\text{THL}} = (0.40 , \text{ns}) ) ( C_L + 20 , \text{ns} )</td>
<td></td>
<td>15</td>
<td>-</td>
<td>40</td>
<td>80</td>
</tr>
</tbody>
</table>

| Output: Fall Time           | \( T_{\text{THL}} \) | 5.0          | -            | 100       | 200  | ns   |
| \( T_{\text{THL}} = (1.35 \, \text{ns}) \) \( C_L + 33 \, \text{ns} \) |        | 10           | -            | 50        | 100  |      |
| \( T_{\text{THL}} = (0.40 \, \text{ns}) \) \( C_L + 20 \, \text{ns} \) |        | 15           | -            | 40        | 80   |      |

| Propagation Delay Time      | \( T_{\text{PLH}} \) | 5.0          | -            | 300       | 600  | ns   |
| \( T_{\text{PLH}} = (0.90 \, \text{ns}) \) \( C_L + 255 \, \text{ns} \) |        | 10           | -            | 150       | 300  |      |
| \( T_{\text{PLH}} = (0.36 \, \text{ns}) \) \( C_L + 132 \, \text{ns} \) |        | 15           | -            | 100       | 220  |      |
| \( T_{\text{PLH}} = (0.26 \, \text{ns}) \) \( C_L + 97 \, \text{ns} \) |        |              |              |           |      |      |

| C\(_D\) to Q or \( \bar{Q} \) | \( t_{\text{r(HL)}} \) | 5.0          | -            | 250       | 500  | ns   |
| \( T_{\text{PHL}} = (0.90 \, \text{ns}) \) \( C_L + 205 \, \text{ns} \) |        | 10           | -            | 125       | 250  |      |
| \( T_{\text{PHL}} = (0.36 \, \text{ns}) \) \( C_L + 107 \, \text{ns} \) |        | 15           | -            | 95        | 190  |      |

| Minimum Input Pulse Width   | \( T_{\text{HL}} \) | 5.0          | -            | 35        | 70   | ns   |
| A, B or \( C_D \)           | \( T_{\text{WH}} \) | 10           | -            | 30        | 60   |      |
| \( T_{\text{WL}} \) |        | 15           | -            | 25        | 50   |      |

| Minimum Retrigger Time      | \( t_{\text{rr}} \) | 5.0          | 0            | -         | -    | ns   |
| A, B or \( C_D \)           | \( t_{\text{rr}} \) | 10           | 0            | -         | -    |      |
| \( C_D \) to Q or \( \bar{Q} \) | \( t_{\text{rr}} \) | 15           | 0            | -         | -    |      |

| Output Pulse Width – Q or \( \bar{Q} \) | \( T \) | 5.0          | 210          | 222       | 234  | \( \mu\text{s} \) |
| Refer to Figure 9 for other values of \( R_X \) and \( C_X \). |          | 10           | 212          | 224       | 236  |      |
| \( C_X = 0.002 \, \text{\( \mu \)F}, \, R_X = 100 \, \text{k\Omega} \) | | 15           | 214          | 226        | 238  |      |

| \( C_X = 0.1 \, \text{\( \mu \)F}, \, R_X = 100 \, \text{k\Omega} \) | 5.0          | 9.3          | 9.85        | 10.4     | \( \text{ms} \) |
| \( C_X = 10 \, \text{\( \mu \)F}, \, R_X = 100 \, \text{k\Omega} \) | 10           | 9.5          | 10          | 10.5     |      |
| | 15           | 9.6          | 10.14        | 10.7      |      |

| Pulse Width Match between circuits in the same package. \( C_X = 0.1 \, \text{\( \mu \)F}, \, R_X = 100 \, \text{k\Omega} \) | 100 \((T_1 - T_2)/T_1\) | 5.0          | -          | \( \pm 1 \) | -    | \%   |
| | 10           | -            | \( \pm 1 \)  | -         | -    |      |
| | 15           | -            | \( \pm 1 \)  | -         | -    |      |

### OPERATING CONDITIONS

| External Timing Resistance | \( R_X \) | - | 5.0 | - | * | \( \text{k\Omega} \) |
| External Timing Capacitance | \( C_X \) | - | 0 | - | No Limit | \( \text{pF} \) |

*The maximum usable resistance \( R_X \) is a function of the leakage of the capacitor \( C_X \), leakage of the MC14538B, and leakage due to board layout and surface resistance.

![FIGURE 1 – LOGIC DIAGRAM (1/2 of Device Shown)](image-url)
TYPICAL APPLICATIONS

FIGURE 12 – Retriggerable Monostables Circuitry

FIGURE 13 – Non-retriggerable Monostables Circuitry

FIGURE 14 – Reduction of Power-Up Output Pulse Width

FIGURE 15 – Connection of Unused Sections
The Edge-Triggered J-K Flip-Flop

It has a J and a K input which allow it to be "programmed" for various functions. The output responds to the J-K inputs when the edge of the clock input occurs.

![Diagram of J-K Flip-Flop]

Truth Table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>K</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Waveform Diagram]
Asynchronous Inputs

An asynchronous input is acted upon immediately rather than waiting for the next clock pulse edge.

\[ \text{PRE} \leftarrow \text{A Low on the "Preset bar" input causes } Q \text{ to go High.} \]

\[ \overline{\text{CLR}} \leftarrow \text{A Low on the "Clear bar" input causes } Q \text{ to go Low.} \]

CLK

\[ \overline{\text{PRE}} \]

\[ \overline{\text{CLR}} \]

\[ Q \]
Master-Slave Flip-Flops

Data is entered into the flip-flop on the leading edge of the clock pulse, but the output does not reflect the input state until the trailing edge.

Pulse-triggered Master-Slave JK Flip-Flop

- Postponed output symbol

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>J  K  CLK</td>
<td>Q  Q̅</td>
<td></td>
</tr>
<tr>
<td>0  0  0</td>
<td>Q₀  Q₀̅</td>
<td>No Change</td>
</tr>
<tr>
<td>0  1  0</td>
<td>0  1</td>
<td>Reset</td>
</tr>
<tr>
<td>1  0  0</td>
<td>1  0</td>
<td>Set</td>
</tr>
<tr>
<td>1  1  0</td>
<td>Q₀  Q₀̅</td>
<td>Toggle</td>
</tr>
</tbody>
</table>

CLK pulse waveform

J

K

Q
The Data Lock-Out Master-Slave Flip-Flop

It's similar to the pulse-triggered master-slave flip-flop except it has a dynamic clock input. It is only sensitive to the data inputs at the instant of the clock's leading edge.

Flip-Flop Operating Characteristics

Propagation Delay Times

This is the time required for the output to change after an input signal has been applied.
Set-up Time
The minimum interval required for logic levels to be maintained prior to the triggering edge of the clock.

\[ \times \]
\[ \text{D} \]
\[ \text{CLK} \]
\[ \Rightarrow \text{Set-up time} \]
\[ \hat{t}_S \]

Hold Time
The minimum interval required for logic levels to remain on the inputs after the triggering edge of the clock.

\[ \times \]
\[ \text{D} \]
\[ \text{CLK} \]
\[ \Rightarrow \text{Hold time} \]
\[ \hat{t}_H \]

Maximum Clock Frequency \( - f_{\text{max}} \)
The highest rate at which the flip-flop can be reliably triggered.

Pulse Widths \( - t_w \)
The minimum required pulse widths for the clock, preset and clear inputs for reliable operation.
Flip-Flop Applications

Parallel Data Storage

Collected as a Data Latch or Register

Frequency Division

\[ f_{out} = \frac{f_{in}}{2} \]

Each cascaded flip-flop causes an additional division by 2
Counting

Example Ripple counter

One-Shots

A one-shot is a monostable multivibrator. Its output remains in its one stable state until triggered. Once triggered its output goes high for a fixed period of time and then returns low until re-triggered.

Example simple one-shot ckty

Example One-shot symbols
Non retriggerable One-Shots

They will ignore additional trigger pulses until they completely time-out.

These pulses ignored

ex/  

Trigger

output

\( t_w \)

ex/  74121 Non retriggerable One-Shot

Pulse width set by \( RC \):  \( t_w = 0.7RC_{EXT} \)

Options:  
- a) No External \( RC \), tie \( R_{INT} \) to \( Vcc \) \( \Rightarrow t_w \approx 30 \text{ nsec} \)
- b) Use external \( C \) and \( R = 2k\Omega \) \( R_{INT} \)

\( t_w = 0.7(2k\Omega)C_{EXT} \)

- c) Use external \( C \) and \( R_{EXT} \)

\( Vcc \)
Schmitt-Trigger Inputs

A Schmitt-trigger input produces hysteresis which will "sharpen-up" the edge of a slow-changing waveform. This will permit reliable triggering when a slowly changing signal is applied to an edge-triggered type input.

Example: The B input of the 74121 has a Schmitt-trigger which will allow triggering with signal rise times of 1 Volt/sec.

![Schmitt-trigger schematic]

Schmitt-trigger symbol

Example: 4584 Inverter

Example: 4093 Quad 2-input NAND with Schmitt-trigger inputs
Retriggable One-Shots

A retriggable one-shot can be triggered before it times out.

\[ t_w = 0.32 \frac{R C_{\text{EXT}}}{1 + \frac{0.7}{R}} \]
Determine $R_{\text{ext}}$ and $C_{\text{ext}}$ to produce a pulse width of 1 μsec.

Solution: Choose a standard value capacitor. Let $C_{\text{ext}} = 560 \, \text{pF}$.

Solve the equation for $R_{\text{ext}}$.

$$t_w = 0.32 RC \left(1 + \frac{0.7}{R}\right) = 0.32 RC + 0.7 \left(\frac{0.22 RC}{R}\right)$$

$$t_w = 0.32 RC + (0.7)(0.32) C$$

$$R = \frac{t_w - (0.7)(0.32) C}{0.32 C} = \frac{t_w}{0.32 C} - 0.7$$

Substitute value.

$$R = \frac{1 \, \mu\text{sec}}{(0.32)(560 \, \text{pF})} - 0.7 = 4.89 \, \text{k\Omega}$$

Choose nearest standard value.

Let $R = 4.7 \, \text{k\Omega}$.
A One-Shot Application

One practical one-shot application is a sequential timer that can be used to illuminate a series of lights. This type of circuit can be used, for example, in a lane change directional indicator for highway construction projects or in sequential turn signals on automobiles.

Figure 8–56 shows three 74122 one-shots connected as a sequential timer. This particular circuit produces a sequence of three 1 s pulses. The first one-shot is triggered by a switch closure or a low-frequency pulse input producing a 1 s output pulse. When the first one-shot (OS 1) times out and the 1 s pulse goes LOW, the second one-shot (OS 2) is triggered also producing a 1 s output pulse. When this second pulse goes LOW, the third one-shot (OS 3) is triggered and the third 1 s pulse is produced. The output timing is illustrated in the figure. Variations of this basic arrangement can be used to produce a variety of timed outputs.

FIGURE 8–56
A sequential timing circuit using three 74122 one-shots.
Basic Operation

A functional diagram showing the internal components of a 555 timer is given in Figure 8–57. The comparators are devices whose outputs are HIGH when the voltage on the positive (+) input is greater than the voltage on the negative (−) input and LOW when the − input voltage is greater than the + input voltage. The voltage divider consisting of three 5 kΩ resistors provides a trigger level of $\frac{1}{3}V_{CC}$ and a threshold level of $\frac{2}{3}V_{CC}$. The control voltage input (pin 5) can be used to externally adjust the trigger and threshold levels to other values if necessary. When the normally HIGH trigger input momentarily goes below $\frac{1}{3}V_{CC}$, the output of comparator B switches from LOW to HIGH and SETS the S-R latch, causing the output (pin 3) to go HIGH and turning the discharge transistor $Q_1$ off. The output will stay HIGH until the normally LOW threshold input goes above $\frac{2}{3}V_{CC}$ and causes the output of comparator A to switch from LOW to HIGH. This RESETS the latch causing the output to go back LOW and turning the discharge transistor on. The external reset input can be used to RESET the latch independent of the threshold circuit. The trigger and threshold inputs (pins 2 and 6) are controlled by external components connected to produce either monostable or astable action.

![Functional Diagram of 555 Timer](image)

**FIGURE 8–57**

*Internal functional diagram of a 555 timer (pin numbers are in parentheses).*
FIGURE 8-58
The 555 timer connected as a one-shot.

(a) Prior to trigger

(b) When triggered

(c) At end of charging interval

FIGURE 8-59
FIGURE 8-61
Operation of the 555 timer in the astable mode.

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

(8-4)

FIGURE 8-62
Frequency of oscillation as a function of \( C_1 \) and \( R_1 + 2R_2 \). The sloped lines are values of \( R_1 + 2R_2 \).
By selecting $R_1$ and $R_2$, the duty cycle of the output can be adjusted. Since $C_1$ charges through $R_1 + R_2$ and discharges only through $R_2$, duty cycles approaching a minimum of 50 percent can be achieved if $R_2 >> R_1$ so that the charging and discharging times are approximately equal.

An expression for the duty cycle is developed as follows. The time that the output is HIGH ($t_H$) is how long it takes $C_1$ to charge from $\frac{1}{2}V_{CC}$ to $\frac{3}{2}V_{CC}$. It is expressed as

$$t_H = 0.7(R_1 + R_2)C_1$$  \hspace{1cm} (8-5)

The time that the output is LOW ($t_L$) is how long it takes $C_1$ to discharge from $\frac{3}{2}V_{CC}$ to $\frac{1}{2}V_{CC}$. It is expressed as

$$t_L = 0.7R_2C_1$$  \hspace{1cm} (8-6)

The period, $T$, of the output waveform is the sum of $t_H$ and $t_L$.

$$T = t_H + t_L = 0.7(R_1 + 2R_2)C_1$$

This is the reciprocal of $f$ in Equation (8-4). Finally, the duty cycle is

$$\text{Duty cycle} = \frac{t_H}{T} = \frac{t_H}{t_H + t_L}$$

$$= \left( \frac{R_1 + R_2}{R_1 + 2R_2} \right) 100\%$$ \hspace{1cm} (8-7)

To achieve duty cycles of less than 50 percent, the circuit in Figure 8–60 can be modified so that $C_1$ charges through only $R_1$ and discharges through $R_2$. This is achieved with a diode $D_1$ placed as shown in Figure 8–63. The duty cycle can be made less than 50 percent by making $R_1$ less than $R_2$. Under this condition, the expression for the duty cycle is

$$\text{Duty cycle} = \left( \frac{R_1}{R_1 + R_2} \right) 100\%$$ \hspace{1cm} (8-8)

FIGURE 8–63
The addition of diode $D_1$ allows the duty cycle of the output to be adjusted to less than 50 percent by making $R_1 < R_2$. 

141
Memories

Memories typically store binary data in groups of one to eight bits.

A **byte** is an 8-bit binary quantity.

A **nibble** is a 4-bit binary quantity.

A **word** consists of one or more bytes.

In a memory I.C. data is stored in an array of memory cells. Each cell stores 1 bit.

Examples of a 64-cell memory array can be organized.

- **8x8 array**
- **16x4 array**
- **64x1 array**

A particular bit is addressed by its row and column number.

In an 8x8 array a single row would have one address and could store one byte of data.
1. Address code placed on address bus and address 3 selected
2. Read command is applied
3. The contents of address 3 is placed on the data bus and shifted into data register. The contents of address 3 is not destroyed by the read operation.

1. Address code placed on address bus and address 5 selected
2. Data byte placed on data bus
3. Write command causes data byte to be stored in address 5, replacing previous data
RAM memory
Stands for "Random Access Memory" because all memory addresses are accessible in an equal amount of time.
RAM's have Read and Write capability.
RAM's are volatile memory i.e. they lose their stored data when power is turned off.

ROM memory
Stands for "Read Only Memory".
ROM's do not have the Write capability of RAM's.
ROM's are nonvolatile memory i.e. they retain their stored data even if power is turned off.

A mask ROM is permanently programmed at the factory.
Tristate Output is utilized to allow many I.C's to be connected to the same data bus.
FIGURE 12-6
The semiconductor ROM family.
A control bus is used to drive the I.C. Chip Select inputs which allows many I.C.s to share the same address bus.

**ROM Access Time**

- Address inputs \((A_0 - A_n)\)
- Address transition
- Valid Address on input lines
- Data Outputs \((Q_0 - Q_7)\)
- Valid data on output lines
- Data output transition
- Chip Enable \(E\)

**Example**

In an IBM P.C., a ROM is used to store BIOS (Basic Input/Output Services). This allows machine start-up before DOS is loaded into RAM.

**Example**

A BASIC programming language interpreter is also stored in ROM in the PC.
**PROMs**

A PROM is a type of Read Only Memory that can be programmed in the field with a device called a PROM "burner".

The "burner" blows out selectable fuse-links to store 1's and 0's.

The data is permanently stored (non-volatile) and cannot be erased (one time programming only).

---

**EPROMS**

Erasable Programmable Read Only Memory can be programmed in the field with an EPROM "burner". (Uses higher voltage to program i.e., 25v.)

The entire contents can be erased by an EPROM "eraser" which shines Ultra Violet light into a window on the I.C. (sets all bits Hi)

The data is non-volatile, but the EPROM can be reprogrammed many times.
EEPROMs

Electrically Erasable Programmable Read Only Memory can be programmed in the circuit with normal (+5V) logic level voltages.

EEPROMs behave like non-volatile RAM.

Comparison to RAMs:

EEPROM

Non-volatile

Stores data up to 10 years

A few thousand write cycles before failure.

Requires about 1 ms write time

RAM

Volatile

(Requires Back-Up Battery)

Unlimited number of write cycles

Write times as fast as 10 nanosec.

called a "16 K" ROM holds 16 K bits.

or 2 K bytes

1K = 1024

\[ \text{digital approximation} \]
Static RAM (SRAM)

Data is stored in latches.
Data is lost if power removed. (Volatile)
Used for high speed writing and reading of data.

Disadvantage:

since the SRAM memory cell consists of several transistors for each bit stored, the number of bits which can be stored on a given size IC is smaller than on DRAM.
Higher cost per bit than DRAM

Advantage:

Doesn't require the additional "refresh circuitry" which DRAM needs, simpler to design with.

Dynamic RAM (DRAM)

Data is stored as a charge (voltage) on capacitors.
It requires additional circuitry to continuously (about once every milliseconds) re-charge the capacitors.
Each memory cell has only 1 transistor and 1 capacitor.
Lower cost per bit than SRAM.
Used in large memories (Mbytes) where the low cost per bit more than pays for the refresh circuitry cost.
Logic symbol and block diagram for the MCM6264C SRAM.

**TABLE 12-2**

Truth table for the MCM6264C SRAM (X = don’t care, H = HIGH, L = LOW)

<table>
<thead>
<tr>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$G$</th>
<th>$W$</th>
<th>Mode</th>
<th>Outputs</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Unselected</td>
<td>High-Z</td>
<td>None</td>
</tr>
<tr>
<td>X</td>
<td>L</td>
<td>X</td>
<td>X</td>
<td>Unselected</td>
<td>High-Z</td>
<td>None</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>Output disabled</td>
<td>High-Z</td>
<td>None</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>READ</td>
<td>$D_0-D_7$</td>
<td>Read</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>X</td>
<td>L</td>
<td>WRITE</td>
<td>High-Z</td>
<td>Write</td>
</tr>
</tbody>
</table>

**FIGURE 12-27**
The MCM6264C pin assignments and packages.

(a) Pin diagram
(b) Packages
FIGURE 12–29
The MCM6246 pin assignments and package.

(a) Pin diagram
(b) Package

FIGURE 12–30
Read and write cycles for the MCM6264C (red labels) and the MCM6246 (blue labels) SRAMs.
Flash Memories

They are high density read/write memories that are non-volatile.

Each storage cell consists of a single floating-gate MOS transistor.

A data bit is stored as charge or the absence of charge on the floating gate.

Memory Type Comparison:

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Nonvolatile</th>
<th>High Density</th>
<th>One Transistor Cell</th>
<th>In System Writability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SRAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DRAM</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EPROM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

When selecting memory, the designer must also consider:

- Cost
- Access Time
- Number of Erasures Required
**Memory Expansion**

**Word-Length Expansion**

**Example:** Construct a 256 x 8 ROM from two 256 x 4 ROM I.C.s.

**Solution:**

```
+------------------+
| Address Bus      |
+------------------+
| ROM #1           |
|                  |
+------------------+
| Control Bus      |
+------------------+
| ROM #2           |
|                  |
+------------------+
| Data Bus         |
```

Note: Address and Control Bus pins are connected in parallel.

**Word-Capacity Expansion**

**Example:** Construct a 512 x 4 ROM from two 256 x 4 ROM I.C.s

```
+------------------+
| Address Bus      |
+------------------+
| ROM #1           |
|                  |
+------------------+
| Control Bus      |
+------------------+
| EN               |
+------------------+--+
| ROM #2           |
+------------------+
| Data Bus         |
```

Note: Data Outputs are tied together but only one ROM is enabled at a given time.
Single-In-Line Memory Modules (SIMMS)

SIMMS consist of several memory IC's on a small PC board with a single row of I/O pins.

Example:
Eight 4Mx1 bit RAM ICs are on a SIMM to implement a 4Mx8 bit memory module.

Special Types of Memories

First In-First Out (FIFO) Memories

A FIFO memory is equivalent to a long shift register with data entering at one end and leaving at the other end. It's often referred to as a "data pipeline".

Examples of FIFO are printer buffers and keyboard buffers as well as serial communication buffers.

In computer systems, a buffer is some memory where data is temporarily stored before being transferred between two systems or devices.

In electronics or digital logic, a buffer is an amplifier which increases output power capability.
FIFO Applications

One important application area for the FIFO register is the case in which two systems of differing data rates must communicate. Data can be entered into a FIFO register at one rate and taken out at another rate. Figure 12-49 illustrates how a FIFO register might be used in these situations.

(a) Irregular telemetry data can be stored and retransmitted at a constant rate.

(b) Data input at a slow keyboard rate can be stored and then transferred at a higher rate for processing.

(c) Data input at a steady rate can be stored and then output in bursts.

(d) Data in bursts can be stored and reformatted into a steady-rate output.
Last In - First Out (LIFO) Memories

An LIFO memory allows data to be stored and then recalled in reverse order.

An LIFO memory structure is called a stack.

Example: Register Stack

```
17
32
```

"15" pushed on stack

"15" popped off stack

Moved down when new data added

Moved up when data removed

Example: RAM Stack

In a microprocessor system, a portion of RAM is reserved for the stack. The top of the stack is "kept track of" by the address location stored in the stack pointer register.

Example: Stacks are used when calling subroutines to store the necessary return addresses.
FIGURE 12-54
Illustration of the pushing of data onto a RAM stack.

(a) Stack pointer addresses top-of-stack and data byte is pushed onto stack from the data bus.
(b) Stack pointer is decremented to next top-of-stack and second data byte is pushed onto stack.
(c) Stack pointer is decremented to next top-of-stack and third data byte is pushed onto stack.
(d) Stack pointer is decremented to next top-of-stack and waits for fourth data byte.

FIGURE 12-55
Illustration of the pulling of data out of the RAM stack.

(a) Last (fourth) byte in is pulled (read) from top-of-stack.
(b) Stack pointer is incremented and third data byte in is pulled from top-of-stack.
(c) Stack pointer is incremented and second data byte in is pulled from top-of-stack.
(d) Stack pointer is incremented and first data byte in is pulled from top-of-stack.
**CCD (charge-coupled device)**

Data is stored as charge on capacitors.

**Advantages:**
1. High density since no transistor needed at each cell.
2. In CCD camera the charge is injected by incoming photons of light.

**Disadvantages:**
1. Slow access time since data must be serially shifted to output.
2. Charges must be periodically refreshed.

---

**Magnetic Memories** (Non-volatile, high density)

Data is stored by magnetizing particles in the disk or tape as it passes near an electromagnetic read/write head.

**Floppy Disks:** (head contacts disk thru window)
- 5½" holds 720 K bytes
- 3.5" holds 1.44 M bytes

**Adv:**
1. Portable
2. High capacity

**Disadv:**
1. Slow access time
2. Sensitive to stray magnetic fields
The Hard Disk:
- The head rides very close (but not touching) the rotating disk. Requires sealed unit to keep out dust. Organized into tracks and sectors.

**Adv:**
1. High capacity - up to several gigabytes
2. Faster access time than floppy disks

**Disadv:**
1. Sensitive to vibration due to close head spacing
2. Still slower than RAM or solid state memory

Magnetic Tape:
- Magnetic tape wrapped around a reel.

**Adv:**
1. Portable
2. High capacity

**Disadv:**
1. Very slow access time (not random access but in serial form)
2. Largely replaced by disk technology for computer data memory

Magnetic tape useful for audio or video storage where data is accessed serially by nature.
Magneto-Optic Disk
A combination of an electromagnetic head and a high-power laser are used for writing to disk.
A low-power laser is used to read from the disk.

Adv:
1) More storage capacity than magnetic hard disks
2) Data can be erased and re-written
3) Unaffected by stray magnetic fields.
   (only magnetizable when heated by high-power laser)

Disadv:
1) High cost
2) Slower write speed

CD-ROM (Compact Disk Read Only Memory)
Data written at factory as microscopic pits on a plastic disk which is read by reflecting a laser beam off of the disk.

Adv: 1) High density
     2) Non-magnetic

Disadv: 1) Read Only

WORM (Write Once - Read Many Disk)
Data can be written in the field with a high-power laser. Can't be erased and once written it is like a normal CD-ROM

Adv: 1) High Density
     2) Field Programmable

Disadv: 1) Higher cost than CD
        2) Read only after written
10-6 Digital Voltmeters

a) General Characteristics
1) Readout advantages of digital display
2) BCD output available
3) Features/cost ratio improves with LSI advances
4) Automatic: polarity reversal, range selection, overload indication
5) Accuracy: up to ±0.005% of reading
6) Resolution: 1 part in $10^6$ (1μV readable on 1 volt range)
7) Stability: ±0.002% of reading for 24 hrs., ±0.008% for 6 months

b) Ramp Type DVM
1) Works by measuring the time it takes for a linear ramp to change from $V_{IN}$ to 0 volts.
2) The total of the counter is proportional to $V_{IN}$.
3) Sample rate multivibrator initiates the ramp & resets counter.

---

[Diagram showing waveform and circuit diagram related to Ramp Type DVM]
Staircase-ramp DVM

A counter is used to index the digital input of a Digital-to-Analog Converter (DAC or D/A) which generates a staircase ramp voltage. The ramp voltage is compared to $V_{IN}$ and when $V_{Ramp} = V_{IN}$ the ramp is stopped and the counter's address is displayed.

Counter output address = $V_{IN}$ when ramp stops

Additional Features

1) Control logic loads counter address into display latch, then resets counter to begin next sampling period.

2) An over-range indicator is driven by the carry output of the last counter stage.

3) Automatic range switching and polarity indication are accomplished by MOSFET switches.

4) DC input voltage must be free of AC ripple (noise) for accurate readings.

5) Maximum Conversion Time = $(T_o)(\text{Maximum Count Possible})$

Example: Given: Oscillator frequency = 10KHz.

3-digit BCD counter

$T_o = \frac{1}{f_o} = \frac{1}{10kHz} = 0.1 \text{ msec}$

Maximum Count $= 10^3 = 1000$

Maximum Conversion Time = $(0.1 \text{ msec})(1000) = 0.1 \text{ sec}$.
Successive-Approximation DVM

Has a conversion time improvement over the staircase ramp DVM due to quick "homing in" on correct voltage for comparison to \( V_{IN} \).

![Block diagram of Successive-Approximation DVM](image)

Operating Sequence

1. Start pulse sets a 1 in MSB of Control Register (1000 0000) which sets D/A output \( V_i = V_{Ref}/2 \)

2. If \( V_x > V_i \), then retain 1 in MSB of C.R.
   If \( V_x < V_i \), then reset 0 in MSB of C.R.

3. Ring counter advances, shifting a 1 in 2nd MSB of C.R. (1100 0000) which sets D/A output \( V_i = V_{Ref}/4 + V_{Ref}/2 \) or \( V_i = V_{Ref}/4 + 0 \)

4. If \( V_x > V_i \), then retain 1 in 2nd MSB of C.R.
   If \( V_x < V_i \), then retain 0 in 2nd MSB of C.R.

5. Repeat process of successive approximation until the ring counter reaches its last bit, then the Control Register contains the final approximation of \( V_x \).
Successive Approximation DVM (continued)

D/A output quickly "homes in" on $V_x$

The number of "guesses" needed equals the number of bits in the D/A.

For 8-bit resolution, the conversion time = 8 clock periods for a successive approximation DVM.

For a staircase ramp DVM, the conversion time can be as long as $2^8 = 256$ clock periods.

The successive approximation DVM is much faster than the staircase ramp type, but requires more complex logic control circuitry.

Binary-Weighted Resistor D/A Converter

From summing amplifier equation

$$V_{out} = -\left(\frac{R_f}{R_B} V_D + \frac{R_f}{R/4} V_c + \frac{R_f}{R/2} V_B + \frac{R_f}{R} V_A\right)$$

If $R_f = R$

$$V_{out} = -(8V_D + 4V_c + 2V_B + V_A)$$

$\uparrow_2\uparrow_2\uparrow_2\uparrow_2\uparrow_2$

If input = 1111 then $V_{out} = -(8 + 4 + 2 + 1) = -15V$

This type of D/A is not practical for more than about 8 bits because the required resistors become too large or too small.
R-2R Resistor Ladder D/A Converter

\[ E_{out} = E_{Ref} \sum_{i=1}^{n} \frac{a_i}{2^i} \]

R-2R D/A only requires 2 resistor values to implement any number of bits.

D/A Converter Using Controlled Current Sources

For 8-bit D/A: \( V_{out} = E_{Ref} \frac{N}{256} \quad 2^8 = 256 \)

Calibration: Adjust \( R_1 \) to set \( V_{out} = 0 \) (or minimum) when \( N=0 \)
Adjust \( R_f \) to set \( V_{out} = E_{Ref} \frac{255}{256} \) (or maximum) when \( N=11111111 \)

D/A Selection Trade-offs (also true for A/D):
Cost \( \propto \) number of bits (Resolution)
Cost \( \propto \) conversion time (Speed)
Conversion time \( \propto \) number of bits

Note: D/A converters cannot be directly cascaded

4A excl two 8-bit D/A are not equivalent to a 16-bit D/A
Successive Approximation A/D Error Sources

How to interpret manufacturer's specs.

Resolution: 10 bits → potential of $\frac{1}{1024} \approx 0.1\%$

Quantization Uncertainty: $\pm \frac{1}{2} \text{ LSB} \rightarrow$ inherent in A/D process.

Relative Accuracy: $\pm \frac{1}{2} \text{ LSB} \rightarrow$ linearity without temp. drift errors considered

Differential Non-linearity: $\pm \frac{1}{2} \text{ LSB} \rightarrow$ no missing codes

Gain Temp. Coef.: $\pm 10 \text{ ppm of F.S.} / ^\circ \text{C}$

Offset Error: adjustable to zero at $25^\circ \text{C}$.

Offset Temp. Coef.: $\pm 20 \text{ ppm of F.S.} / ^\circ \text{C}$

Summing errors

For 10 bits → Relative Accuracy: $\pm \frac{1}{2} \text{ LSB}$ or $\pm 0.05\%$

For $0^\circ \text{C}$ to $50^\circ \text{C}$, Gain Temp. Coef.: $\pm \frac{1}{2} \text{ LSB}$ or $\pm 0.05\%$

For $0^\circ \text{C}$ to $50^\circ \text{C}$, Offset Temp. Coef.: $\pm 1 \text{ LSB}$ or $\pm 0.1\%$

Worst Case Total Error = $\pm 2 \text{ LSB}$

Typical Error $\approx \pm \frac{1}{2} \text{ LSB}$
Selecting A/D Converters
Most of the data being processed (except info.) consists of physical parameters of an analog nature. e.g. temperature, velocity, light intensity, acceleration, etc.

Decreasing costs: $18,000. mini-computer $400. µP. board
Buy Data Acquisition Module ($400.) Can design a dedicated system from a few building blocks ($50.)

Key Parameters
1) Accuracy — Resolution, number of bits
2) Speed — Conversion Time
3) Cost

Seismic Recording Truck
32 Sensors
scan each sensor every 100 µsec.
explosion shock wave only lasts a few seconds

Furnace Temperature Profile
accuracy to 0.1°C.
thermocouple ΔV for 0.1°C ≈ 10 µV
sample rate of 1 reading per second

Digital Transmission of 5 KHz Audio on phone line
By Sampling Theorem: Sample rate = 2f_max. = 10 KHz
Conversion Time = \( \frac{1}{10 \text{ KHz}} = 100 \text{ µsec} \)
Required Accuracy: for speech → 8 bits OK
Note: For High Fidelity Music → 10 or 12 bits
→ sample rate necessary = 2(15 KHz) = 30 KHz.
Dual-Slope Integrating Converter

Advantages:
- Inherent Accuracy
- Non-critical components
- Excellent Noise Rejection: ex/ 120kHz clock will reject 60Hz.
- Low cost
- No "Sample & Hold" required
- No Missing Codes

Disadvantages:
- Low Speed (3 to 100 readings/sec.)
- ex/ 4½ digit performance $\rightarrow$ (± 1 count in 20,000)

Successive Approximation Converter

Advantages:
- High Speed (100,000 readings/sec.)

Disadvantages:
- Several critical components
- Can have missing codes
- Needs "Sample & Hold"
- Difficult to auto-zero
- High Cost
Dual-Slope Integrating A/D Conversion

\[ V_0 = \text{Integrator Output} = -\frac{1}{RC} \int_{t_1}^{t_2} V \, dt \]

\[ T_1 = V_{\text{IN}} \text{ integrate time} \]

\[ T_2 = V_{\text{Ref}} \text{ integrate time} \]

\[ V_{\text{IN}} \times \frac{T_1}{RC} = V_{\text{Ref}} \times \frac{T_2}{RC} \]

\[ \frac{V_{\text{IN}}}{V_{\text{Ref}}} = \frac{T_2}{T_1} \]

The accuracy of the converter is independent of the integrator time constant and clock frequency. Its accuracy depends only on \( V_{\text{Ref}} \) and the equality of clock cycles within a conversion period.

Conversion process steps:
1) Integrator reset to 0V, \( V_0 = 0 \)
2) Integrate \( V_{\text{IN}} \) for 1000 clock cycles \( (T_1) \)
3) Then apply \( V_{\text{Ref}} \) (polarity opposite of \( V_{\text{IN}} \)) to the input of the integrator and begin counting clock pulses.
4) Count clock pulses until integrator output, \( V_0 = 0 \)
5) Display count which is proportional to the ratio of \( V_{\text{IN}} \) to \( V_{\text{Ref}} \).

\[ \frac{V_{\text{IN}}}{V_{\text{Ref}}} = \frac{T_2}{T_1} \Rightarrow T_2 = \left( \frac{8V}{10V} \right) 1000 \text{ counts} = 800 \text{ counts} \Rightarrow \text{Display} = 8.00 \text{V} \]

Advantages:
1) Integrating feature averages out noise on \( V_{\text{IN}} \)
2) Does not require precision integrator components \((R,C,IC)\)
3) Errors due to gating time and integrator non-linearity tend to cancel out in dual-slope method.
Simultaneous A/D Converter

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$V_{IN}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>0 to $\frac{1}{4} V_{Ref}$</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
<td>$\frac{1}{4}$ to $\frac{1}{2} V_{Ref}$</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>L</td>
<td>$\frac{1}{2}$ to $\frac{3}{4} V_{Ref}$</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>$&gt; \frac{3}{4} V_{Ref}$</td>
</tr>
</tbody>
</table>

Advantages: Simplicity and High Speed
Disadvantages: Low Resolution (for high resolution a very large number of comparators are needed → too costly)

Bar Graph Panel Meters

LED panel meters use above technique.

Example: A single I.C. has 10 comparators to drive LED's.
Available in linear and dB type scales

Advantages: Faster response time than mechanical pointer

14.4.6 Sample and Hold Circuit - used to reduce aperture time.

Aperture Time (window or sample time) - a measure of the uncertainty of when $V_{IN}$ was sampled.
(Important when measuring a rapidly changing $V_{IN}$)

- Sample mode - $S_2 \& S_4$ closed
  - $C$ charges to $V_{IN}$ → Acquisition Time
- Hold mode - $S_1 \& S_3$ closed
  - $C$ holding previous $V_{IN}$ (actually slowly discharging) → Holding Time
  - Aperture time → Switching time

$S/H$ circuit used with non-integrating type (comparison type) A/D converters.
Multiplexing Design Philosophy

Multiplexing allows a single component to be utilized for more than one function through time-sharing techniques. Multiplexing is used when the cost of the additional multiplexing hardware is less than the cost of duplicating the shared component.

Multiplexed LED Displays

LED's appear not to be blinking due to the multiplexing flash rate being faster than our persistence of vision.

The current duty cycle can be used to control LED brightness.

LED's driven by a 100 ma. current with a 10% duty cycle appear as bright as if driven by a 10 ma. constant current.
14.5.1 Digital-to-Analog Multiplexing

This is an example of sequential selection of D/A converters with a common digital input.

Same digital input to all converters on common data bus.

14.5.2 Analog-to-Digital Multiplexing

This is an example of sequential selection of S/H ckt. using a common D/A to produce several analog outputs.

This is an example of how a single A/D converter can be used to sequentially monitor several analog input channels.
Digital Display Types

1. Nixie Tubes
   - Use a separate neon lamp element for each numeral.
   - Disadvantages: Need +400V supply and driver circuit. Shorter life than solid state types.

2. 7-Segment Displays
   a. LED Displays - Supply light
      - Adv: Act as light source → visible in dark
      - Disadv: High current consumption. Not visible in bright light (direct sunlight)
   b. LCD (liquid crystal displays) - Reflect light
      - Adv: Very low current consumption. Visible in bright ambient lighting conditions
      - Disadv: Not visible in dark
      - Require driver circuit with AC source

3. Dot Matrix Alpha-Numeric Display
   - Ex: 5x7 matrix - consists of LED array
   - Adv: Many characters possible
   - Disadv: Requires more complex decoding circuit than 7-segment types.
SELECTING A/D CONVERTERS

Dave Fultagger, Intersil Inc.

One of the popular pastimes of the nineteen sixties was to predict the explosive growth of digital data processing, fed by the newly-developed semiconductor MSI circuits, and the subsequent demise of all analog circuits. The first part of this prediction has certainly come true: the advent of the microprocessor has caused, and will continue to cause, a revolution in data processing which was unthinkable 10 years ago. But far from causing the demise of analog systems, the reverse has occurred. Nearly all the data being processed (with the notable exception of financial data) consists of physical parameters of an analog nature: pressure, temperature, velocity, light intensity and acceleration to name but a few. In every instance this analog information must be converted into its digital equivalent, using some form of A/D converter.

Converter products are thus assuming a key role in the realization of data acquisition systems. Increased use of microprocessors has also caused dramatic cost reductions in the digital components of a typical system. The $5000 mini-computer of a few years ago is being replaced by a $145 dedicated microprocessor board. This trend is being reflected in the analog components. No longer is it possible to justify buying a $400 data acquisition module when a dedicated system, adequate for the task under consideration, can be put together for $50.

Thus many engineers, who in the past have had limited exposure to analog circuits, have no choice but to come to grips with the characteristics of A/D converters, sample & hold circuits, multiplexers and operational amplifiers. Contrary to the propaganda put out by many of the specialty module houses, there is nothing mysterious about these components or the way they interface with one another. Now that many of them are available as one or two chip MSI circuits, a block diagram may be turned into a working piece of hardware with relative ease.

The purpose of this note is to compare and contrast the more popular A/D designs, and provide the reader with sufficient information to select the most appropriate converter for his or her needs.

THE IMPORTANT PARAMETERS

Let's begin by taking a look at some actual systems, since this will illustrate the diversity of performance required of A-D converters.

Case 1: A seismic recording truck is situated over a potential natural gas site. Some 32 recording devices are laid out over the surrounding area. An explosive charge is detonated and in a matter of seconds it is all over. During that time it is necessary to scan each recorder every 100 microseconds. Speed is clearly the most important bit of information. In this instance, 12 bit accuracy is not required, and, since the truck contains many thousands of dollars of electronics, cost is not a critical parameter. The A/D will be a high speed successive approximation design.

Case 2: A semiconductor engineer is measuring the thermal profile of a furnace. It is necessary to make measurements accurate to a few tenths of a degree Centigrade, which is equivalent to a few microvolts of thermocouple output. Sampling rates of a few readings per second are adequate and costs should be kept low. The integrating ("dual slope") technique is used, depending on which manufacturer you go to. A/D is the only type capable of the required precision/cost combination. It has the added advantage of maintaining accuracy in a noisy environment.

Case 3: A businessman is talking to his sales office in Rome. Assuming the phone company is not on strike, his voice will be sampled at a 100 kHz rate, or thereabouts. In order not to lose information in the audio frequency range up to 5 kHz, this requires a medium accuracy (8 bit) A/D with a cycle time of 100 microseconds or less. In this application the integrating technique is not fast enough, so it is necessary to use a slow (for this application) successive approximation design.

These examples serve to introduce both the most popular conversion techniques (successive approximation and integrating), and the three key parameters of a converter: speed, accuracy and cost. In fact the first choice in selecting an A/D is between successive approximation and integrating, since virtually 95% of all converters fall into one of these two categories.

If we look at the whole gamut of available converters, with conversion speeds ranging from 100 ms to less than 1 μs, we see that these two design approaches divide the speed spectrum into two groups with almost no overlap. (Table 1) However, before making a selection solely on the basis of speed, it is important to have an understanding of how the converters work, and how the data sheet specifications relate to the circuit operation.

THE INTEGRATING CONVERTER

Summary of Characteristics

As the name implies, the output of an integrating converter is the average value of an input voltage over a fixed period of time. A sample and hold circuit, therefore, is not required to freeze the input during the measurement period, and noise rejection is excellent. Equally important, the linearity error of integrating converters is small since they use time to quantize the answer. It is relatively easy to hold short-term clock jitter to better than 1 in 10^8.

The most popular integrating converter uses the dual-slope principle, a detailed description of which is given in Ref 1. Its advantages and disadvantages may be summarized as follows:

Advantages:
- Inherent accuracy
- Non-critical components
- Excellent noise rejection
- No sample & hold required
- Low cost
- No missing codes

Disadvantages:
- Low speed (typically 3 to 100 readings/sec)

In a practical circuit, the primary errors (other than reference drift) are caused by the non-ideal characteristics of analog switches and capacitors. In the former, leakage and charge injection are the main culprits; in the latter, distributed absorption is a source of error. All these factors are discussed at length in Ref 1.

A well-designed dual slope circuit such as Intersil's 802A/7103A is capable of 4½ digit performance (1 in 2^20 = 100000) with no critical tweaks or close tolerance components other than a stable reference.

Timing Considerations

In a typical circuit, such as the 802A/7103A referred to above, the conversion takes place in three phases as shown in Fig 1. Note that the input is actually integrated or averaged over a period of 10000 clock pulses (or 8.3 ms with a 120 kHz clock) within a conversion cycle of 40000 clock pulses in total. Also note that the actual business of looking at the input signal does not begin until after 10000 clock pulses, since the circuit first goes into an auto-zero mode. For a 3½ digit product such as the 7101 or 7103, the measurement time is 10000 clock pulses (or 88.3 ms with a 120 kHz clock).

These timing considerations give the dual slope circuit both its strengths and its weaknesses. By making the signal integrate prior to an integral number of line frequency periods, excellent 60 Hz noise rejection can be obtained. At the same time, integrating the input signal for several milliseconds smooths out the effect of high frequency noise.

But in many applications such as transient analysis or sampling high frequency waveforms, averaging the input over several milliseconds is totally unacceptable. In is therefore necessary to use a sample & hold at all the input, but the majority of systems that demand a short measurement window also require high speed conversions.

THE SUCCESSIVE APPROXIMATION CONVERTER

How it Works.

The heart of the successive approximation A/D is a digital-to-analog converter (DAC) in a feedback loop with a comparator and some clever logic referred to as a successive approximation register (SAR). Fig 2 shows a typical system. The DAC output is compared with the analog input, progressing from the most significant bit (MSB) to the least significant bit (LSB) one bit at a time.

The bit in question is set to one if the DAC output is less than the input. If the current bit is set to zero the register then moves on to the next bit. At the completion of the conversion, three bits are left in the one state causing a current to flow at the output of the DAC which should match fig 2 within 1/4 LSB. Performing an 'n' bit converter requires only 'n' triads, making the technique capable of high speed conversion.
Disadvantages:
- Several critical components
- Can have mismatched codes
- Requires sample and hold
- Difficult to auto-zero
- High cost

Error Sources

The error source in the successive approximation converter is more numerous than in the integrating type, due to the contribution from both the DAC and the comparator. The DAC generally relies on a resistor ladder and either current or voltage switching techniques for quantization. Maintaining the correct impedance ratios over the operating temperature range is much more difficult than maintaining clock pulse uniformity in an integrating converter.

The data sheet for a hypothetical ADC might contain the following accuracy related specifications:

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>10 bits</td>
</tr>
<tr>
<td>Quantization Uncertainty</td>
<td>± 0.1% LSB</td>
</tr>
<tr>
<td>Relative Accuracy</td>
<td>± 0.05% LSB</td>
</tr>
<tr>
<td>Differential Non-Linearity</td>
<td>± 0.1% LSB</td>
</tr>
</tbody>
</table>

Gain Error

- Adjustable to zero at 25°C
- ± 10 ppm of Full Scale Reading

Offset Error

- Adjustable to zero at 25°C
- ± 20 ppm of Full Scale Reading

Now, referring to the definition of terms on page 6, what does this tell us about the product? First of all, being told that the quantization uncertainty is ± 0.1% LSB is like being told that binary numbers are comprised of zeros and ones - it's part of the system. The relative accuracy of ± 0.05% LSB, guaranteed over the entire temperature range, tells us that after removing gain and offset errors, the transfer function never deviates by more than ± 0.05% LSB from where it should be. That's a good spec, but note that gain and offset errors have been adjusted prior to making the measurement. Over a finite temperature range, the temperature coefficients of gain and offset must be taken into account.

The offset temperature linearity of ± 0.1% LSB maximum is also guaranteed over temperature, thus ensuring that there are no missing codes.

The gain temperature coefficient is 10 ppm per °C, or 0.001% per °C. Now 1 LSB in a 10-bit system is 1 ppm. In 12024, or approximately 0.1%. So 0.5°C temperature change from the temperature at which the gain was adjusted (i.e. from -25°C to +75°C) could give rise to 0.5 LSB error. This error is separate from, and in the limit could add to, the offset accuracy spec.

The offset temperature coefficient of 20 ppm per °C gives rise to ± 1 LSB error (over a ±25°C to 175°C range) by the same reasoning applied to the gain tempco. The reference contributes an error in direct proportion to its percentage change over the operating temperature range.

We can summarize the effect of the major error sources:

<table>
<thead>
<tr>
<th>Error Source</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Accuracy</td>
<td>± 0.05% LSB</td>
</tr>
<tr>
<td>Gain Temp Coeff.</td>
<td>± 0.05% LSB</td>
</tr>
<tr>
<td>Offset Temp Coeff.</td>
<td>± 0.1% LSB</td>
</tr>
</tbody>
</table>

A straight forward RMS summation shows that the A/D is 10 bits ± 0.1% LSB over a 0°C to +75°C temperature range. However, it is over optimistic to RMS errors with such a small number of variables, and yet we do know that the error cannot exceed ± 1 LSB. An optimistic estimate might place the accuracy at 10 bits ± 1 LSB.

Timing Considerations

The 2502/2503/2504 successive approximation registers are now used in the majority of high speed A/D Converters and the timing diagram shown in Fig. 3 is taken from the 2502 data sheet. However, all successive approximation converters have essentially similar timing characteristics. Holding the start input low for at least a clock period initiates the conversion. The MSB is set low and all the other bits high for the first half. Each bit takes one clock period, proceeding from the MSB to the LSB. Note that, in contrast to a high sampling rate shunt converter, a serial output arises naturally from this conversion technique.

Although the successive approximation A/D is capable of very high conversion speeds, there is an important limitation to the slew rate of the input signal. Unlike integrating designs, no averaging of the input signal takes place. To maintain accuracy to 10 bits, for example, the input should not change by more than ± 1 LSB during the conversion period. Fig. 4a shows maximum allowable DNL as a function of sampling (or aperture) time for various conversion resolutions. Now for a sinusoidal waveform represented by Eqn. (1), the maximum rate of change of voltage (dV/dt) is 2πf E. The amplitude of one LSB is E/2^10. Since the pk-pk amplitude is 2E, the change in input amplitude is given by:

\[ JE = \frac{2E}{2\pi f} \text{ where } J = \text{conversion time} \]

![Figure 3: Typical Timing Diagram for Successive Approximation Converter](image)

![Figure 4: Maximum Input Signal Rate Change (dV/dt) and Simultaneous Frequency (f) as a Function of Sampling or Aperture Time for ±1 LSB Accuracy in 4 bits](image)

![Figure 5: Voltage across a capacitor (as % of full scale) as a function of time (as % of time constants)](image)

**CONVERTER CHECKLIST**

In selecting a converter for a specific application, it will be helpful to go through the following checklist, matching required performance against data sheet guarantees:

a) How many bits?

b) What is total error budget over the temperature range?

c) What is full scale reading and magnitude of LSB?

- Make sure that the 95% noise is substantially less than the magnitude of the LSB if no smoothing techniques are given. Assume that the omission is intentional.

d) What input characteristics are required?

With most successive approximation converters, the input resistance is low (~ 5kΩ) since one is looking into the comparator summing junction in a well damped dual slope circuit, there should be a high input resistance buffer (Rin = 1kΩ) included within the auto-zero loop. However in some designs [Elderyl 8700, Analog Devices AD7556] the input looks directly into the integrating resistor (1 MΩ).

e) What aperture time (or measurement window) is required?

If an averaged value of the input signal (over some milliseconds) is acceptable, use an integrating converter. Refer to Fig. 4 for systems where an averaged value of the input is not acceptable. Remember that successive approximation systems rely on a sample hold to freeze the input while the conversion is taking place. Thus the sample & hold characteristics should be matched to the input signal slew rate, and the A/D converter characteristics matched to the required conversion rate.

f) What measurement frequency is required?

This will determine the maximum allowable conversion time (including auto-zero time for wide input signal types).

g) Is component compatibility important?

Some A/Ds interface easily with microprocessors, others do not. 2 does explore the microprocessor interface in considerable depth.
h) Does the converter form part of a multiplexed data acquisition system? Note that some integrating converters (Motorola MC14433) assess polarity based on the input voltage during the previous conversion cycle. Such designs are clearly unsuitable for multiplexed inputs where the signal polarity bears no relationship to the previously measured value. They can also give trouble with inputs hovering around zero.

i) Is 60Hz rejection important? If the line frequency rejection capabilities of the integrating converter are important, make sure that the duration of the measurement (input integrate) period is a fixed number of clock pulses. In some designs, the input integration time is programmed by the auto-zero information, making rejection of specific frequencies impossible.

**MULTIPLEXED DATA SYSTEMS**

The foregoing discussion has summarized the characteristics of A/D converters as stand-alone components. However, one of the most important applications for A/D converters is as part of a multiplexed data acquisition system. Traditionally, systems of this type have used analog signal transmission between the transducer and a central multiplexer/converter console (Fig 6a). To sample 100 data points 25 times per second requires a 100 input analog multiplexer and an A/D capable of 2500 conversions per second. A successive approximation converter would be the obvious choice.

![Figure 6a](image-url) Data Acquisition using one central A/D

![Figure 6b](image-url) Data Acquisition using several local A/Ds

Another approach, which becomes attractive with the availability of low cost IC converters, is to use localized A/D conversion with digital transmission back to a central console. In the limit one could use a converter per transducer, but it is often more economical to have a local conversion station servicing several transducers (Fig 6b). Several advantages result from this approach. Firstly, digital transmission is more satisfactory in a noisy environment, and lends itself to optical isolation techniques better than analog transmission. Secondly, using local conversion stations significantly reduces the number of interconnections back to the central processor. When one considers that the instrumentation for a typical power plant uses 4.5 million feet of cable, this will result in real cost savings. Finally, by sharing the conversion workload among several A/Ds, it is frequently possible to switch from a successive approximation to a dual slope design.

An example of a local conversion station featuring an 8652/1763 dual slope A/D, a CMOS multiplexer, and an UART for serial data transmission is discussed in Fig 7. The local conversion station concept can be taken a stage further by the addition of a microprocessor. This may be used to reduce the data prior to transmission to the central computer, and/or to look for dangerous conditions, for example.

**DEFINITION OF TERMS**

**Quantization Error.** This is the fundamental error associated with dividing a continuous (analog) signal into a finite number of digital bits. A 10 bit converter, for example, can only identify the input voltage to 1 part in 2^10, and there is an unavoidable output uncertainty of 1.3 LSB (Least Significant Bit). See Fig. 7.

**Linearity.** The maximum deviation from a straight line drawn between the ends of the converter transfer function. Linearity is usually expressed as a fraction of LSB size. The linearity of a good converter is ±1 LSB. See Fig. 8.

**Differential Non-Linearity.** This describes the variation in the analog value between adjacent pairs of digital numbers, over the full range of the digital output. If each transition is equal to 1 LSB, the differential non-linearity is clearly zero. If the transition is 1 LSB ± 1/2 LSB, then there is a differential linearity error of ±1/2 LSB, but no possibility of missing codes. If the transition is 1 LSB ± 1 LSB, then there is the possibility of missing codes. This means that the output may jump from say 001111111111000000000000000 to 0000000000000000000000000000, missing out 100...100. See Fig. 9.

**Relative Accuracy.** The input to output error as a fraction of full scale, with gain and offset errors adjusted to zero. Relative accuracy is a function of linearity, and is usually specified at less than ±1/2 LSB.

**Gain Error.** The difference in slope between the actual transfer function and the ideal transfer function, expressed as a percentage. The error is generally adjustable to zero by adjusting the input resistor in a current-comparing successive approximation A/D. See Fig. 10.

**Gain Temperature Coefficient.** The deviation from zero gain error on a 'zeroed' part which occurs as the temperature moves away by 25°C. See Fig. 10.

**Offset Error.** The mean value of input voltage required to set zero code out. This error can generally be trimmed to zero at any given temperature, or is automatically zeroed in the case of a good integrating design.

**Offset Temperature Coefficient.** The change in offset error as a function of temperature.
DO'S AND DON'TS OF APPLYING A/D CONVERTERS

Peter Bradshaw and Skip Osgood, Intersil Inc.

In many applications, the limitation in the performance of any system lies in how the individual components are used. The Analog to Digital Converter (A/D) can also be considered as a component and, therefore, proper design procedures are necessary in order to obtain the optimum accuracy. Intersil A/D converters are inherently extremely accurate devices. To obtain the optimum performance from them, care should be taken in the hook-up and external components used. Test equipment used in system evaluation should be substantially more accurate and stable than the system needs to be. For the following sections illustrate DO's and DON'Ts to obtain the best results from any system.

1. DON'T INTRODUCE GROUND LOOP ERRORS

Plan your grounding carefully. Probably the most common source of error in any Analog-Digital system is improper grounding. Let's look at Fig 1. All the grounds are tied together; as everything should be, right? WRONG! Almost everything is wrong with this connection.

![FIGURE 1](image)

The power supply currents for the analog and digital sections, together with the output or display currents, all flow through a lead common to the input. Let's analyze some of the errors we have introduced. The average currents flowing in the resistance of the common lead will generate an offset voltage. Even the autozero circuit of an integrating A/D converter cannot remove this error. In addition, this current will have several varying components. The clock oscillator, and the various digital circuits, etc. From it, will show supply currents varying at the clock frequency, and usually at submultiple also. For a successive approximation converter, these will add up to an additional effective offset. For an integrating converter, at least the higher frequency components should average out. In some converters, the analog supply currents will vary with the clock (or a sub-multiple frequency) if the display is multipplexed, that current will vary with the multiplex frequency, usually small. In any system using an integrating converter, both digital and analog section currents will change as the converter goes from one phase at converter to another. (Currents of this type injected into an autozero loop are particularly disastrous.) Another serious source of variation is the change in digital and display section currents with the resultant value. This frequency shows up as an oscillating result and, for missing results, one value being displayed displaces the effective input to a new value, which is converted and displayed, leading to a different placement, a new value and so on. This sequence usually closes after two or three values, which are displayed in sequence. A more serious source of error in this circuit comes from the clock oscillator frequency. For an integrating converter, variations in clock frequency during a single conversion cycle due to varying digital supply voltage or supply currents, or ground loops to a timing capacitor, will lead to incorrect results.

Fig 2 shows a much better arrangement. The digital and analog grounds are connected by a link carrying only the interface currents between sections, and the input section is also fed back by a low current line. The display current loop will not affect the analog section and the clock section is isolated by a decoupling capacitor. Note that external reference return currents and any other analog system currents must also be returned carefully to analog ground.

![FIGURE 2](image)

2. DON'T COUPLE DIGITAL SIGNALS INTO ANALOG LINES

Although Intersil's A/D converter circuits have been designed minimizing the internal coupling of digital signals into analog lines, the external capacitive coupling is controlled by the user. For the best results, it is advisable to keep analog and digital sections separated on PCB boards. A few examples of the results of capacitive coupling follow.

On dual slope converters, the "buzzy" line swings from one state to the other at the end and beginning of the auto-zero cycle. Capacitive coupling from this line to the auto-zero or integrating capacitors will induce an effective input offset voltage. A similar affect occurs with the "Measure/Zero" line on charge balancing converters and for a successive approximation converter with coupling between "End of Conversion" and a sample-and-hold capacitor. For a multi-bit display device, coupling between the multiplex or digit lines and these capacitors can lead to non-linearity of the converter. And coupling from the digital line into a high-impedance input line can lead to errors in any system.

3. DON'T USE ADEQUATE QUALITY COMPONENTS

For successive approximation converters, the resistor used must have excellent line and temperature stability to maintain accuracy. Any adjustment potentiometers, etc. must be of compatible quality (note that in some trimouts, the slider position moves with temperature). For dual slope converters, the component selection is less critical. Long term drifts in the integrating resistor and the capacitors are not important. However, any sensitive divider used on the reference, especially if it is adjustable, must be of sufficient stability not to degrade system accuracy. Dielectric absorption in the integrating capacitor is important (see reference 1) and the integrating resistor must have a negligible voltage coefficient to ensure linearity. Noisy components will lead to noisy performance, whether in the integrator, autozero or clock circuits.

4. DON'T USE A GOOD REFERENCE

Good references are like good wines; nobody is quite sure how to make them but generally the older the technology, the better the result, and the proof lies in the testing (or testing now). It is possible to deal with the old temperature compensated zener with the current flow adjusted to the optimum for each diode. If you aren't into Zainfeld Superior Premier Cru (1972), the Intersil 0052 has a fairly good reference built in. In either case, the division down from what you get to the required reference voltage requires careful attention to accuracy and stability. In the fundamental fact that no converter can be better than its reference voltage.

5. DON'T WATCH OUT FOR THERMAL EFFECTS

All amplifiers have a thermal time constant of a few milliseconds to dissipation changes in the die. These can cause changes in such parameters as offset voltages and VTN matching. For example, the power dissipation in an 8018 quad current switch depends on the digital voltage. Although the die is carefully designed to minimize the effects of this, the resultant temperature changes will affect the matching between current switch values. The power dissipated in a dual slope converter circuit depends on the comparator polarity and hence varies during the conversion cycle. Offset voltage variations due to this cannot be autozeroed out, so they can lead to errors. Again, a poor choice of comparator loading or swing level will enhance this (normally) minor effect. The power dissipation in an output display could be coupled into the sensitive analog sections of a converter, leading to similar problems. To minimize these effects, keep all boards in the processor very close to the IC's, and limit all power dissipation to the IC's. The processor or board should be cooled with low thermal resistance, to ensure that the IC's reach a uniform temperature.

6. DO USE THE MAXIMUM INPUT SCALE

To minimize all other sources of error, it is advisable to use the highest possible full scale input voltage. This is particularly important with successive approximation converters, where offset voltage errors can quickly get well above 1.5V. But even for integrating-type converters, noise and the various other errors discussed above will increase in importance as the nominal full scale range gets low. Pre-conversion gain is usually preferable for small nominal signals. All Intersil's integrating converters have a digital output line that can be used to extend auto-zero or pre-conversion circuits. Using a careful pre-zeroing circuit, noise will never be aliased to the digital signal into the analog system, of course.

Also, DO CHECK THESE AREAS

The digital inputs down (or up) if you are not using them. This will avoid stray input spikes from affecting operation. Bypass all supplies with a large and a small capacitor close to the package. Limit input currents into any IC pin to values within the maximum rating of the device (or a few mA if specified) to avoid damaging the device. Ensure that power supplies do not reverse polarity or supply high voltages when turned on or off. Remember that many digital gates take higher than normal supply currents for inputs between logic levels. And remember also that gates can look like amplifiers under these circumstances. For example, in Fig 3, where stray and internal input-to-output capacitance is multiplied by the gain of the gate at the threshold capacitance, any gain from the output on the 8002 comparator (see reference 1) for the effects of this non-inverting gate could lead to oscillations.

![FIGURE 3](image)

External Adjustment Procedure

Most of the A/D converters now offered by Intersil do not require an offset adjustment. An exception being the all-zero circuits which typically give less than 0.1 uV of offset. Therefore, the only optional adjustment required to obtain optimum accuracy in any application is the full scale or gain reading.

With the A/D converter in a continuous mode of conversion, the following procedure is recommended. The full scale adjustment is made by seeing the input voltage to precisely 1LSB less than full scale or 1/2 LSB down from nominal full scale. (Note that the nominal full scale may be actually never reached but it is always true that 1/2 LSB is exactly 0.) Adjust the full scale control until the converter output does not change from full scale to zero. You should then be calibrating within one count.