"States"

HIGH $\leftrightarrow$ TRUE $\leftrightarrow$ 1
LOW $\leftrightarrow$ FALSE $\leftrightarrow$ 0

A state can be indicated by a symbol, e.g. $Q$

negation:

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$\bar{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

This diagram is called a "truth table"

LEDs are often used to indicate state

always use a current-limiting resistor

$\text{typ. } 150 \text{ to } 330 \Omega$
Families of digital logic

<table>
<thead>
<tr>
<th></th>
<th>74LSxx</th>
<th>CMOS (74Cxx)</th>
<th>CMOS (74HCxx)</th>
<th>High-Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Fastest</td>
<td>Slowest</td>
<td>Almost as fast as 74LS</td>
<td></td>
</tr>
<tr>
<td>Power Consumed</td>
<td>Most</td>
<td>Least (Best for battery-powered circuits)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>True Voltage</td>
<td>7x +3.4 V</td>
<td>7x +4.9 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False Voltage</td>
<td>Fx +0.2 V</td>
<td>Fx +0.1 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power-Supply</td>
<td>Vcc = +5 V</td>
<td>+5V &lt; Vdd &lt; +15 (9V battery is okay)</td>
<td>2V &lt; Vdd &lt; 6V</td>
<td></td>
</tr>
</tbody>
</table>

For all 3 families, the most common power-supply voltage is +5V & GND.
### Logic Gates

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Boolean</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>![Inverter Symbol]</td>
<td>$Q = \overline{A}$</td>
<td>![Inverter Truth Table]</td>
</tr>
<tr>
<td>OR</td>
<td>![OR Symbol]</td>
<td>$Q = A + B$</td>
<td>![OR Truth Table]</td>
</tr>
<tr>
<td>AND</td>
<td>![AND Symbol]</td>
<td>$Q = A \cdot B$</td>
<td>![AND Truth Table]</td>
</tr>
<tr>
<td>NOR</td>
<td>![NOR Symbol]</td>
<td>$Q = \overline{A+B}$</td>
<td>![NOR Truth Table]</td>
</tr>
</tbody>
</table>
NAND $\overline{A \cdot B}$ \quad $Q = \overline{A \cdot B}$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$Q$</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>

XOR $A \oplus B$ \quad $Q = A \oplus B$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$Q$</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>

Buffer $A \rightarrow Q$ \quad $Q = A$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Math

You can work problems by making a big truth table:

inputs → intermediate steps → output

ex  

\[ Q = \overline{A} \cdot \overline{B} \]

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

\[
\uparrow \quad \text{result: same as NOR} \quad \uparrow
\]

\[ \Rightarrow \overline{A} \cdot \overline{B} = \overline{A + B} \]
Digital Chips

Part no.

74 LS 02

TTL

"Quadruple 2-input NOR gate"

Pin Diagram

Pin 14 13 12 11 10 9 8

1 2 3 4 5 6 7 GND
Design reminder

*It is OK to connect one output to multiple inputs*

[Diagram showing an AND gate with a single output connected to two inputs.]

*OK: two inputs connected to one output*

*It is NOT OK to connect multiple outputs together*

[Diagram showing two outputs from separate AND gates connected together.]

*NOT OK: two outputs are connected*
"Fanout"

When one circuit's output drives multiple inputs, that's called "fanout"

ex. "Fanout of 3"

A digital gate can typically source/sink enough current for a fanout of 10 or more.
Driving LEDs

many student projects use LEDs

it's easy to use too many LEDs

because on LED is a big power consumer

a gate can drive one LED

if you need to drive more, use a transistor switch
How to use another gate as an inverter

Recall NAND truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Q</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>

Note: Always use a resistor when connecting a digital input directly to 5V.

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

Inverter

Multiplexer concept:
You'll use in Lab 8 - it's like a switch to choose an input.

Multiple data inputs

One output, the same as the input that is specified by the "data select input"
Mechanical Switches

a user interface to enter data

\[ +5V \quad \text{SWITCH CLOSED} \]

↑ the only trouble with this is "bounce"

\[ \text{SWITCH CLOSED} \]

↑ time

↑ time

switch is closed

cure for bounce later & lab
**Binary - Coded Decimal**

**LAB 9**

Four binary bits:
- called A, B, C, D
- represent numbers from 0 to 9

<table>
<thead>
<tr>
<th>*</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>7</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>8</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>9</td>
<td>1 0 0 1</td>
</tr>
</tbody>
</table>
Divide-by-10 Counter (LAB 9)

One use: reduce frequency by 10

Another use:

use 4 outputs D C B A

which advance (as in # above) at each clock input

when it gets to 9, it starts again at 0,

(like a digit in an odometer)
One-Shot
aka "monostable multivibrator"

A device widely used to:
- Produce a trigger pulse
- Produce a pulse of a desired duration
- Analog-to-digital conversion

74121

Input B

Output Q

Start of pulse is triggered when input crosses threshold
Pulse width determined by external RC
One-shot (cont.)

In Lab 10, you will convert an analog input (sine wave) to a digital output (pulse train) using a one-shot.

\[
\begin{align*}
V_i & \quad \text{input} \\
\sin t & \quad \text{trigger} \\
V_o & \quad \text{output}
\end{align*}
\]

Digital Clocks (CLK)

- a pulse train
- used by digital circuits that involve:
  - timing
  - transferring data

Terminology:

- Positive clock edge
- Negative clock edge
Flip Flops

Flip flops are digital devices that:

- have memory
- have two stable states

which state the flip-flop has depends on previous history — this is related to the concept of "memory"

Types of flip-flops:

- SR = set reset, the simplest flip-flop

Clocked Flip Flops:

- D = data flip flop, these are often combined to make a "shift register"
- JK has two data inputs; otherwise much like a D flip flop
Here, we will focus on how an S-R flip-flop functions. In the lab you will also use D and JK flip-flops.

**SR (Set-Reset) Flip Flop**

Has two data inputs, no clock input

![SR Flip Flop Diagram](https://via.placeholder.com/150)

Recall NAND truth table:

<table>
<thead>
<tr>
<th>inputs</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
</tr>
</tbody>
</table>

How it is used:

- Inputs A & B are usually HIGH (and the flip-flop outputs will be in one of two possible stable states).
- One input is momentarily made LOW to SET or RESET the outputs.
first stable state

Suppose inputs A & B are HIGH & output X is HIGH
what is output Y?

\[ A = 1 \quad B = 1 \quad X = 1 \quad Y = ? \]

Examine the NAND truth table (previous page)
for the lower NAND gate
\[ x = 1 \quad y = ? \]
\[ B = 1 \quad \text{both inputs are HIGH, i.e. 1} \]
\[ \Rightarrow \text{output must be low} \]
\[ \Rightarrow y = 0 \]

So, if A & B are HIGH and X is HIGH
then Y must be LOW

second stable state

Suppose inputs A & B are HIGH & output X is LOW
what is output Y?

Again, examine NAND truth table
for the lower NAND gate
\[ x = 0 \]

B = 1 → Y = ?

NAND truth table \( \Rightarrow Y = 1 \)

so, if \( A \& B \) are HIGH & output \( X \) is LOW

then \( Y \) must be HIGH

summary of the two stable states:

- when both inputs \( A, B \) are HIGH,
  there are two stable states of the output \( X, Y \)

  first: \( X = 1, \ Y = 0 \)
  second: \( X = 0, \ Y = 1 \)

- Q1: which of these two states will it be in?

4. For flip-flops in general, this will depend on the history of previous inputs applied to the flip flop. For the SR flip-flop, it depends on whether an input has been "RESET"
**Reset for SR flip-flops**

Suppose the SR flip-flop is initially in the stable state: inputs $A = B = 1$; outputs $x = 1, y = 0$.

Next, let's bring $B$ to low momentarily.

![Diagram showing the process]

After step 1, the inputs look like this:

To find outputs $x$ and $y$, let's start by asking "Can $x$ be HIGH?"

If $x$ is HIGH, then the lower gate looks like:

![Gate diagram]

which is not allowed (see NAND truth table).

$\Rightarrow x$ cannot be HIGH
$\Rightarrow x$ must be LOW
Thus, after step 1, the inputs are $A=1, B=0$.
output are $x=0, y=1$

Note: Output state is now opposite of what it was before step 1.

After step 2:

- The outputs after step 1 (*above*) were $x=0, y=1$.
  Now we also have $A=1, B=1$.

- This combination $A=1, B=1$ inputs $x=0, y=1$ outputs is one of the two stable states.

- So the outputs do not change in step 2.
Summary for SR flip flop "reset":

Usually, you keep both inputs high.
If you momentarily bring input B low, then back to HI, that will "reset" the flip-flop to the state $x=0, y=1$. (If it was already in that state, then nothing changes.)

Timing Diagram (general)

In digital electronics, we need "truth tables" for gates, and for flip-flops (and other devices that involve timing or memory), we also need "timing diagrams".

Discuss:
- for special project: timing diagrams, unlike schematics, diagrams are not graded. But they might help you in designing your project & debugging it.
Timing Diagram

SR flip flop (showing it "resetting" when input B is brought low).

inputs:

```
   A
   B
   X
   Y
```

outputs:

The other possibility for resetting the SR flip flop is to bring input A low (instead of input B). This will reset the flipflop to state X=1, Y=0

Latches

another name for SR flipflop

Application of SR flipflop

switch debouncing

text p. 506 & lab 7
Data D - flip flop (as in lab 9)

- is a clocked flip flop
- clock input is a train of pulses
- indicated on chip with an arrow

\[
\begin{array}{c}
D \\
\rightarrow \\
Q \\
\end{array}
\]

- also has CLR input to "CLEAR" state of flip flop (CLR = 1 forces Q \rightarrow 2)
- requires +5V, GND power supply
what's inside a D-flip flop

D ——— M ——— Q

clk ———— M ———— Q

master ——— slave

Timing:

CLK

causes master flip to do something: M goes to the state of input D

causes slave flip flop to do something: output Q goes to the state of M

so after one clk pulse ————
the data (0 or 1) that was on input D is transferred to output Q
Applications of D flip-flop:

**Shift Register (Lab 9)**

At each clk pulse, the data in the register (a sequence of 1's & 0's) shifts to the right.

Useful for serial memory.

**Divide-by-2 Counter**

IN (clk)

OUT (Q)

D = Q