Digital Devices & Circuits

A device like the Shmitt Trigger allows us to move to the realm of digital devices.

In the digital domain we restrict ourselves to using just two voltages:

- High / Low
- ∅ / 0
- On / Off

Devices will take high/low inputs and produce high/low outputs.

What are voltages should we use?

Let's use ground as low and say 5V as high.

Can't have exactly 5V as high because of measurement errors (same with V=0 as low).

What about V>2.5V as high
V<2.5V as low

What about noise
What about V=2.5V
Imagine I'm sending a digital signal

\[
\begin{array}{c}
\text{Sender} \\
5V \\
t \\
\end{array} \quad \xrightarrow{\text{add noise}} \quad \begin{array}{c}
\text{Receiver} \\
25V \\
t \\
\end{array}
\]

Put in a Schmitt Trigger

\[
\begin{array}{c}
5V \\
\end{array} \quad \xrightarrow{\text{buffer zone}} \quad \begin{array}{c}
\text{Still use Schmitt but as it normally works: with hysteresis} \\
11V \\
\end{array}
\]

Need a buffer zone

Still use Schmitt but as it normally works: with hysteresis
So really have four voltages

\[ \begin{align*}
\text{IN/OUT} & \quad \exists \quad \text{Acceptable Values} \\
\text{Hi/Low} & \quad \end{align*} \]

\( V > V_{HI} \) considered as high as input
\( V > V_{HO} \) spec for producing high as output

\( V < V_{LI} \)
\( V < V_{LO} \)

Digital Logic Families defined by

\[ \begin{align*}
V_{HI} & \quad V_{HO} & \quad V_{LI} & \quad V_{LO} \\
\text{TTL} & \quad \text{RTL} & \quad \text{DTL} & \quad \text{ECL} & \quad \text{CMOS} & \quad \exists \quad \text{We will be using these} \\
\text{Logic Devices} & \quad \text{Buffer} \rightarrow \text{Schmitt trigger}, \text{restores value} \\
& \quad V_{HI} \rightarrow V_{HO} \quad V_{LI} \rightarrow V_{LO} \\
& \quad \text{NOR Gate} \quad \text{Output opposite of input} \end{align*} \]
Imagine a "voltage switch":

- \( V_{in} > V_{th} \) closes the switch.
- \( V_{in} = 0 \) switch open.
  - \( V_{out} = V_{cc} = V_{ho} \)
  - \( V_{in} > V_{hi} \) \( V_{out} = 0 = V_{lo} \)

**Truth Table**

<table>
<thead>
<tr>
<th>( V_{in} )</th>
<th>( V_{out} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Most transistors can work like this switch:

- \( R_T \gg R_L \) \( V_{out} = 1 \)
- \( R_T \ll R_L \) \( V_{out} = 0 \)

**NAND Gate**

<table>
<thead>
<tr>
<th>( V_A )</th>
<th>( V_B )</th>
<th>( V_O )</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>
NOR

\[
\begin{array}{c}
V_A \\
\downarrow \\
\text{AND} \\
\downarrow \\
V_B \\
\downarrow \\
V_S
\end{array}
\quad
\begin{array}{c|c|c|c}
V_A & V_B & V_S \\
\hline
00 & 0 & 1 \\
01 & 1 & 0 \\
10 & 0 & 0 \\
11 & 1 & 0 \\
\end{array}
\]

Can combine gates in arbitrary fashion

Analyzer using Boolean Algebra

range: 0, 1
operators: AND, OR, NOT

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

with variables:

\[\begin{align*}
0 \cdot A &= 0 & 1 \cdot A &= A & A + A &= A & A + \bar{A} &= 1 \\
0 + A &= A & 1 + A &= 1 & A \cdot A &= A & A \bar{A} &= 0 \\
\end{align*}\]
Commutative: 
\[ A + B = B + A \]
\[ AB = BA \]

Associative: 
\[ (A + B) + C = A + (B + C) \]
\[ (AB)C = A(BC) \]

Distribution: 
\[ A(B + C) = AB + AC \]
\[ A + BC = (A + B)(A + C) \]

**PROOF:** How can we prove this?
Construct truth table (i.e. consider all cases)

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

Columns identical
\[ \Box \text{ QED} \]
Some simplification rules

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

\[ A(\overline{A} + B) = A(1) \]
\[ A \overline{B} + B = (B + \overline{B})B = (\overline{B} + B)B = A + 0 + B \]
\[ A + AB = A + AB = A \]

De Morgan's Theorems

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

Be careful!

\[ \overline{AB} \neq \overline{A} \cdot \overline{B} \]

Negative input & output exchanged

AND & OR

Universal Gate: \textbf{NOR}

\[ \text{AND} \quad AB = \overline{A + B} \]
\[ \text{NOT} \quad \overline{A} = \frac{0 + A}{0 + B} \]
\[ \text{OR} \quad A + B = A + B = \frac{0 + A + B}{0 + A + B} \]

\[ A \]
\[ \overline{1} \]
\[ \overline{1} \]
\[ A \]
\[ \overline{1} \]
\[ \overline{1} \]
\[ \overline{1} \]
\[ \overline{1} \]
\[ A \]
\[ \overline{1} \]
\[ \overline{1} \]
\[ A \overline{B} \]
\[ \overline{A} \overline{B} \]
\[ \overline{A} \overline{B} \]
\[ A + B \]
Boolean Functions

All Boolean functions have a finite set of responses. Write the truth table...

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

Many ways of writing out algebraically:

\[
F = AB + AC \\
F = A(B+C) \\
\bar{F} = \overline{AB+AC} = \overline{AB}.\overline{AC} = (A+B)\cdot(\overline{A+C})
\]

Many ways of writing out:

\[\bar{F} = \overline{A+B} \cdot \overline{A+C}\]

A B C

\[
F = \overline{A} + (B+C) \\
F = \overline{A+B} \cdot \overline{A+C}
\]
Canonical Form
\[
\begin{align*}
\text{ABC} & + \\
\text{F} & \\
\text{or} & \\
\text{(A+B+C)(D)} & \\
\text{max term} & \\
\text{Sum of products:} & \\
\text{one term for each line of \text{TT}} & \\
\end{align*}
\]

2-Input Functions
16 total
\[
\begin{array}{ccc}
A & B & \text{F} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

0 or 1 in any of the boxes
1 - all zeros (trivial)
4 - one one (atoms) \(A, \overline{A}, \overline{B}, B\) ANDs
6 - two ones (atoms) XORs
4 - three ones ORs
1 - all ones (trivial)

this is \(A\)

\[
\begin{array}{ccc}
0 & 1 & \text{F} \\
0 & 1 & \text{F} \\
\end{array}
\]

this is \(B = AB + \overline{A}B\)
Three & Four input functions
Still have two-layer representations

\[ F = ABC + \ldots \]
\[ F = (A + B + C)(\ldots) \]
But how to simplify? Fewest Gates
Recall \[ B = AB + \overline{AB} \]
Three gates in So.P.
Zero gates in simplest

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

\[ \overline{AB} \quad 00 \quad 01 \quad 10 \quad 10 \]
\[ \quad 0 \quad 1 \quad 1 \quad 0 \]
Bundle

Karnaugh Maps

\[ \begin{array}{c|cccc}
A & 00 & 01 & 11 & 10 \\
B & 0 & x & x & x \\
C & 1 & x & x & x \\
\end{array} \]

Look for bundles of \( 2^n \) contiguous 1's

\[ \overline{A} \quad 0 \\
1 \quad 1 \quad 1 \quad 1 \quad 1 \quad \Rightarrow \quad A \text{ is on}! \]

\[ \overline{B} \quad x \quad x \quad x \\
x \quad x \quad x \quad x \quad x \quad \Rightarrow \quad B \text{ is on} \]
By circling all #1s with f
fewest # of boxes
get simplest representation!

1. Represent F as S.a.P.
   Place terms in map 1 for present
   0 for missing

2. Group adjacent 1's in groups
   Groups as rect. an 2^n
   Start with largest groups

3. When all 1's included
   Write down sum of groups