Digital electronics

- Analog: Continuously varying signal (i.e., an analog signal can take any value between \( V_{\text{min}} \) and \( V_{\text{max}} \))
- Digital: a signal can only have two discrete states: \( V_{\text{high}} \) and \( V_{\text{low}} \)

Benefits of digital signals:
- Completely insensitive to noise.
- Signal can be stored, processed, and copied without loss of fidelity (like copying digital music vs. copying analog tapes)

Limitations:
- Most experiments deliver analoge signals.
- The most crucial part of signal acquisition is (and likely always will be) the analog front end of your circuit.
- Digital data acquisition hardware is quite complex.
- Use commercial data acquisition hardware and own analogue front end.

Digital logic can be useful to control an experimental setup and is usually very easy to build (e.g., you want to fire a laser 12.3\( \mu \)s after laser is charged AND a value opened.)
Logic Signals:

There are many different standards for digital circuits. But, in essence, they only differ in the voltage levels used to describe logic high (= "true") and logic low (= "false").

TTL: ("transistor-transistor" logic)

\[ \begin{array}{c}
\text{Signal Voltage} \\
+5V \quad +2.4V \quad +0.8V \quad 0V \\
\end{array} \]

"High" undefined "Low"

The transitions from low to high and vice versa must be fast (no more than 10ns) to avoid oscillations and excessive current draw.

Note: to generate a logic "Low," you need to pull the signal to ground. It is not sufficient to leave it unconnected. Never leave a signal dwell in the "undefined" region.

Digital Signal Processing:

There is a special chip for pretty much any logical operation imaginable (AND, OR, NOT, etc.). Chips are essentially a (very large) collection of such elementary logic gates.
Boolean Algebra: (simplify the math $\rightarrow$ simplify your circuit!)

- **Associative laws:** 
  \[
  (A + B) + C = A + (B + C) \\
  (A \cdot B) \cdot C = A \cdot (B \cdot C)
  \]

- **Distributive law:** 
  \[
  A \cdot (B + C) = A \cdot B + A \cdot C
  \]

- **De Morgan's theorem:** 
  \[
  A \cdot \overline{B} = \overline{A} + \overline{B} \\
  \overline{A + B} = \overline{A} \cdot \overline{B}
  \]

Note: From De Morgan's theorem follows that you actually only need a NOT gate and either an AND or a OR to perform any logic operation. This is important for quantum computing and other "exotic" processing techniques.

\[
\begin{align*}
\text{AND} & : A \cdot B \\
\text{OR} & : A + B \\
\text{NOT} & : \overline{A} \\
\text{NAND} & : \overline{A \cdot B} \\
\text{NOR} & : \overline{A + B} \\
\text{XOR} & : A \oplus B
\end{align*}
\]
Latches and flip-flops: (circuits with two stable states often used to store information)

There are many variations of flip-flops and latches. All of them are built from basic logic gates. The simplest one of these is the SR latch: ("set-reset" latch)

\[
\begin{array}{c}
R \\
\rightarrow \chi \\
Q \\
\leftarrow S \\
\rightarrow \bar{Q} \\
\end{array}
\]

or:

\[
\begin{array}{c}
R \\
\rightarrow Q \\
\leftarrow S \\
\rightarrow \bar{Q} \\
\end{array}
\]

---

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No change</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( Q = 0, \bar{Q} = 1 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( Q = 1, \bar{Q} = 0 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>

---

Example:

Value opens when you push "value open". It stays open until you push "value close". Never push both at the same time!
The JK flip-flop extends the functionality of the SR latch. It defines $S = R = 1$ as a "toggle command." It uses a clock signal to define when the inputs are to be sampled.

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Clear</th>
<th>Clock</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>No Charge</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>No Charge</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$\overline{Q}$ (toggle)</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

'x': "doesn't matter"  "falling edge"

L17 555: An application of an RS-latch (hence used to generate a square wave with adjustable duty-cycle)

$RA + RB \Rightarrow T_H \approx (R_A + R_B) \cdot C$

$RB$ only $\Rightarrow T_L \approx R_B \cdot C$