Introduction:

- We can classify the building blocks of a circuit or system as being either analog or digital in nature.
  - If we focus on voltage as the circuit parameter of interest:
    - Analog: The voltage can take on a range of voltages, e.g. any value between 0.1 and 2 Volts.
    - Digital: The voltage can have only two values, e.g. 0 or 5 Volts
      - We say the voltage is either on or off
  - Digital circuits are useful when we don’t need a continuous range of voltage or current.
  - Examples: Representing numbers, binary logic, counting circuits.
  - Example: Represent base 10 numbers using the binary system:
    
    \[
    \begin{align*}
    2_{10} & = 10_{2} = 1 \times 2^1 + 0 \times 2^0 \\
    10_{10} & = 1010_{2} = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0
    \end{align*}
    \]

- Digital circuits use standard voltages (or currents) to denote **ON** (high, 1) or **OFF** (low, 0).
  - These standards are called "Logic Families" and there are several families.
  - Two of the most popular families are:
    - TTL (Transistor-Transistor-Logic): ON = 5 Volts, OFF = 0 Volts
    - ECL (Emitter-Coupled-Logic): ON = -1 Volt, OFF = -1.6 Volts
  - For practical reasons both ON and OFF are given by a range of voltages or currents.
  - ON for an input to a circuit might have slightly different voltage than ON for an output to a circuit.
A description of several logic families is given in the table below:

<table>
<thead>
<tr>
<th>Logic Family</th>
<th>Delay (ns)</th>
<th>Max. FF Rate (MHz)</th>
<th>Power/Gate (mW)</th>
<th>High (V)</th>
<th>Low (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard TTL (7400)</td>
<td>10</td>
<td>35</td>
<td>15</td>
<td>3.5</td>
<td>0.2</td>
</tr>
<tr>
<td>Low-power Schottky (74LS00)</td>
<td>9.0</td>
<td>33</td>
<td>2</td>
<td>3.5</td>
<td>0.2</td>
</tr>
<tr>
<td>Fast TTL (74F00)</td>
<td>3.5</td>
<td>125</td>
<td>5.5</td>
<td>2.7</td>
<td>0.5</td>
</tr>
<tr>
<td>CMOS (74C00)</td>
<td>25 @ 10 V</td>
<td>10 @ 10 V</td>
<td>0.01</td>
<td>5-15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>50 @ 25 V</td>
<td>3.5 @ 5 V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High-speed CMOS (74HC00)</td>
<td>8.0</td>
<td>40</td>
<td>0.01</td>
<td>2-6</td>
<td>0.1</td>
</tr>
<tr>
<td>ECL</td>
<td>2</td>
<td>250</td>
<td>25</td>
<td>-0.9</td>
<td>-1.8</td>
</tr>
<tr>
<td>100k ECL</td>
<td>0.75</td>
<td>500</td>
<td>40</td>
<td>-1.0</td>
<td>-1.7</td>
</tr>
</tbody>
</table>

- Advantages of Digital:
  - only deal with two voltage levels (either ON or OFF)
  - voltages (or currents) are standardized
  - do not deal with individual transistors…

- Disadvantages of Digital:
  - too many "black" boxes
  - need good power supplies, clocks etc. for circuits to work properly
Logic Gates:

- We want to make decisions based on digital information.
  - For now consider the basic building blocks with one or two inputs and one output.
- The basic logic units (gates) are: AND, OR, NOT.
  - These functions are defined by their truth tables.

**AND**

<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>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ A \land B = Y \]

**OR**

<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>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ A + B = Y \]

**NOT (inverter)**

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

\[ \overline{A} = Y \]

A and B stand for the inputs
Y stands for the output
0: low input or output
1: high input or output
Boolean Algebra or the Algebra of 1's and 0's

- Circuits consisting of logic gates are described by Boolean algebra.
  - Use of this algebra can greatly simplify circuit design, e.g. minimize the number of components.
- The following theorems can be proved using a truth table and the definition of OR, AND, and NOT.

1) \( A + A = A, \ A + 1 = 1, \ A + 0 = A \)
2) \( A A = A \)
3) \( A B = B A \)
4) \( A B C = (A B)C = A (B C) \)
5) \( A (B + C) = AB + AC \)
6) \( \overline{1} = 0, \ \overline{0} = 1 \)
7) \( A + \overline{A} = 1, \ A \overline{A} = 0, \ A \cdot 1 = A \)
8) \( \overline{A} = A \)
9) \( \overline{A + B} = \overline{A} \cdot \overline{B} \)
10) \( \overline{A B} = \overline{A} + \overline{B} \)

\[ \begin{align*}
\text{DeMorgan's Theorem} \\
\text{Example using Boolean algebra:}
\end{align*} \]

- **Prove:** \( X + YZ = (X + Y) (X + Z) \)
  \[ \begin{align*}
  (X + Y) (X + Z) &= XX + XZ + YX + YZ \quad \text{by 5)} \\
  &= X + X (Z + Y) + YZ \quad \text{by 2) and 5)} \\
  &= X (1 + Z + Y) + YZ \quad \text{by 5)} \\
  &= X + YZ \quad \text{by 1)}
  \end{align*} \]

See Simpson page 540 for more theorems.

For clarity “\( \cdot \)” (AND) is not shown in some theorems.
- We could also have proven the above using a truth table.
  - There are 8 \(2^3\) possible combinations of \(X, Y, Z\).
  - For a large number of inputs using a truth table becomes unwieldy.
  - Example, if there are 10 inputs
    \[2^{10} = 1024\] possible combinations!

- Example: Exclusive OR = XOR = \(A \oplus B\).
  - Output is high if inputs are different.

<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>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- How do we make an exclusive OR with AND, OR, and NOT gates?

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

Brute force method
Can we simplify this circuit with the use of less parts?
- Use logical theorems:
  \[ A \oplus B = AB + \overline{AB} \]
  \[ = A\overline{A} + AB + \overline{AB} + B\overline{B} \quad 7) \text{ and } 1) \]
  \[ = A(\overline{A} + B) + B(\overline{A} + \overline{B}) \quad 5) \]
  \[ = A(\overline{AB}) + B(\overline{AB}) \quad 10) \]
  \[ = (A + B)(\overline{AB}) \quad 5) \]
- The circuit uses only 3 parts (OR, NAND, AND), but each of them is different!
- Usually there are many ways to synthesize the same function (circuit).
- Must decide if you want to minimize:
  - number of components
  - types of components
  - number of connections
  - power consumption
- For example we can make an XOR using only 4 NAND gates:
Final example: Suppose you have a light controlled by 3 switches.
- You want the light to be on if any one of the 3 switches is on or if all 3 switches are on.

\[
\begin{array}{ccc|c}
A & B & C & \text{Light} \\
1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
L = ABC + AB\overline{C} + \overline{A}B\overline{C} + \overline{A}BC
\]
\[
= A(BC + B\overline{C}) + \overline{A}(B\overline{C} + B\overline{C})
\]
\[
= A(B + \overline{C})(\overline{B} + C) + \overline{A}(B \oplus C)
\]
\[
= \overline{A}\overline{B}\overline{C}\overline{B} + \overline{A}(B \oplus C)
\]
\[
= \overline{A}BC + \overline{A}(B \oplus C)
\]
\[
= AB \oplus C + \overline{A}(B \oplus C)
\]
\[
= A \oplus (B \oplus C)
\]
Flip-Flops:

- Basic counting unit in computer:
  - counters
  - shift registers
  - memory
- Circuit whose output depends on the history of its inputs.
- Can make a flip-flop with just 2 transistors (or 2 vacuum tubes 1919!).
- Lots of different types of flip-flops (e.g. RS, JK, T, D).

- Example: RS flip-flop or Reset-Set flip flop
  - Flip-flops, like logic gates are defined by their truth table.
  - Flip-flops are controlled by an external clock pulse.
  - All inputs and outputs are logic levels (e.g. TTL, ECL).
  - Can make an RSFF out of NOR gates:

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>( Q_{n+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( Q_n )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>undefined</td>
</tr>
</tbody>
</table>

- \( Q_n \) is the present state of the FF.
- \( Q_{n+1} \) will be the output after the clock enables the FF to look at its inputs (R and S).
- Many FF change state (\( Q_n \rightarrow Q_{n+1} \)) on the trailing edge of the clock.

*The state with R = S = 1 is undefined. The output is not predictable!*
● Example: D flip-flop (Like RS but only one input)

\[
\begin{array}{c|c}
\text{D} & \text{Q} & \text{Q}_{\text{next}} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

● Example: JK flip-flop
  ◆ JKFF is like the RSFF except that both inputs (J and K) can be high (1).

\[
\begin{array}{c|c|c}
\text{J} & \text{K} & \text{Q}_{\text{n+1}} \\
0 & 0 & \text{Q}_n \\
1 & 0 & 1 \\
0 & 1 & 0 \\
1 & 1 & \overline{\text{Q}_n} \\
\end{array}
\]
  ◆ Most JKFF's have a connection for forcing \( Q = 0 \) (clear) or forcing \( Q = 1 \) (preset).

● Example: T (Toggle) flip-flop
  ◆ T flip-flop is like the JKFF with both inputs (J and K) tied to each other.

\[
\begin{array}{c|c|c}
\text{T} & \text{Q} & \text{Q}_{\text{next}} \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
Flip-Flops are a class of circuits called “multivibrators”
  • Multivibrators are circuits with one or more stable states.
  ▪ Monostable multivibrators (one shot) have one stable state.
    ❑ If the circuit is forced out of its stable state (e.g. by an input pulse)
      ☞ it eventually returns back to the stable state by itself.
  ▪ Bistable multivibrators have two stable states.
    ❑ Transitions between states occur only by an external action (e.g. voltage pulse for flip-flops).
    ❑ Transition voltages can be different for the two states (e.g. Schmitt trigger).
  ▪ Astable multivibrators are two state devices which switch on their own accord.
    ☞ Commonly used as oscillators.
Example: Bistable Multivibrator.

- This circuit has two stable states.
- When either transistor conducts there is 4 mA flowing (by design) in the collector ($I_{C1}$ or $I_{C2}$).
- State 1: transistor 1 off, transistor 2 on
  - $V_{C1} \approx 11\, \text{V}$, $V_{B1} \approx 2\, \text{V}$, $V_{E1} \approx 4\, \text{V}$
  - $V_{C2} \approx 4\, \text{V}$, $V_{B2} \approx 5.5\, \text{V}$, $V_{E2} \approx 4\, \text{V}$
  - Transition from state 1 to state 2:
    - Input pulse forces $T_1$ to conduct, $T_1$ conducting means that $V_{C1}$ drops.
    - $V_{C1}$ causes $V_{B2}$ to drop to the point where $T_2$ is not conducting.
- State 2: transistor 1 on, transistor 2 off
  - $V_{C1} \approx 4\, \text{V}$, $V_{B1} \approx 5.5\, \text{V}$, $V_{E1} \approx 4\, \text{V}$
  - $V_{C2} \approx 11\, \text{V}$, $V_{B2} \approx 2\, \text{V}$, $V_{E2} \approx 4\, \text{V}$

![Circuit Diagram]

$V_{C1}$

input

output

12 V

2 k

14 k

14 k

14 k

T1

1 k

11.2 V

4.1 V

-3 V

time

input

V_{C1}

time

11