First, an aside: Binary Addition

- Since numbers can be converted between bases, addition in one base gives the same answer as addition in another base,
- Adding in binary is simpler than adding in decimal,
- So we build computers to add in binary rather than in decimal!

Binary Addition Table

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

Zero, with a Carry

Example

\[
\begin{array}{cccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \text{Carries} \\
0110010101000101 + 0110011100010110 & \Rightarrow & 1100110001011011
\end{array}
\]

Now back to Logic...

When is the sentence True?

- “I am going to the store and going to the beach”
- Clause A: “going to the store”
- Clause B: “going to the beach”
- Either may independently be True or False.
- When is the overall sentence True?
- Only when both are True.
When does Water flow?

• When both valves are open

When does the light come on?

• When both switches are closed

Commonalities

• In all three cases the underlying logic is the same:
  – Both clause A and clause B must be True for the sentence to be True.
  – Both Valve A and Valve B must to open for water to flow.
  – Both Switch A and Switch B must be closed for the light bulb to come on.
• The only thing in common is AND.

Truth Tables

• A Truth Table is a tabular way of describing all possible behaviors for a binary system.
• We could use T and F to indicate True and False, but instead we will define 1 for True and 0 for False.
• The number of rows in a Truth Table is $2^n$ raised to the power of the number of inputs.

Truth Table for AND (two inputs)

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

Can we have more than Two inputs?

• YES.
  • “I am going to the store and going to the beach and going to the movies.”
  • The pipe contains three valves in series, all must be open for water to flow.
  • The circuit contain three switches in series, all must be closed for the light bulb to come on.
What About...

• If we can describe the behavior of AND...
• ...can we also describe OR?
• Yes, but there’s a catch:
  • The OR we use in English is not the same as the OR we use in computer logic!

The OR in English

• “I am going to the store OR going to the beach.”
• If I don’t do either, I’ve lied,
• if I do one or the other, I’ve told the truth,
• but if I do both I’ve lied!
• In English the inference is that if I say I’ll do one thing or the other, I won’t do both!
• This is called an Exclusive OR (aka XOR).

The OR we use in Logic

• If one input or the other is true, the output is true,
• If both inputs are true, the output is still true,
• This is called an Inclusive OR.

When does Water flow?

• When either or both valves are open

When does the light come on?

• When either or both switches are closed
Truth Table(s) for OR

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

General Rules for AND and OR

- **AND:** The output is 1 if all inputs are 1, the output is 0 if any input is 0.
- **OR:** The output is 1 if any input is 1, the output is 0 if all inputs are 0.
- **XOR:** The output is 1 if the inputs differ, the output is 0 if the inputs are the same.

NOT

- “I am NOT going to the store.”

<table>
<thead>
<tr>
<th>A</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Gates

- A Gate is:
  - A mathematical abstraction for...
  - ...a physical device that...
  - ...implements the function of a Truth Table.

AND, OR, and XOR gates

NOT-OR = NOR
NOT-AND = NAND

What Does it Do?

Inputs 0,0

Inputs 1,0

Inputs 0,1

Inputs 1,1
What is its Function?

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

A Binary Adder!

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