Building an Computer Adder
No matter how complex the circuit, or how complex the task being solved, at the base level, computer circuits are made up of three basic components.

These basics components or gates are:
- **AND**
- **OR**
- **NOT**
Building an Adder

- Examine the following binary addition problem:

\[
\begin{array}{c}
1 & 0 & 1 & 0 \\
+ & 1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\end{array}
\]
Building an Adder

- Examine the following binary addition problem:

\[
\begin{array}{c}
1010 \\
+ 111 \\
\hline
1111
\end{array}
\]
Building an Adder

- Remember that in binary addition there are only five possible numeric combinations.

\[
\begin{array}{cccccc}
0 & 0 & 1 & 1 & 1 & 1 \\
+ 0 & + 1 & + 0 & + 1 & + 1 \\
0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

- 1 \leftarrow \text{Carry In}

\begin{align*}
\text{Carry Out to next digit} & \quad \text{and} \\
& \quad 1 1
\end{align*}
Building an Adder

Let’s put this information in a table.

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

If we rotate this table placing the A, B and Sum at the top we get what looks like a standard truth table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Sum</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>10</td>
</tr>
</tbody>
</table>
Building an Adder

- A computer will treat each column of digits in our binary addition as one operation.

\[
\begin{array}{c}
1010 \\
+ 1111 \\
\hline
1 \\
\end{array}
\]

- Essentially the computer uses the circuit designed to implement the table below to solve each column.

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

Ignore the 1 which carries to next column for the moment!
Building an Adder

- Comparing this truth table to the ones for AND, OR and NOT, it is clear this is none of the above.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>S</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>

- This truth table is close to being a table for OR, but the final set of values doesn’t match.
Building an Adder

- Using basic logic gates computer architects designed the following circuit to match the truth table for adding two binary digits.
Building an Adder

- If A = 0 and B = 0, then Sum = 0
Building an Adder

- If $A = 0$ and $B = 1$, then Sum = 1
Building an Adder

- If $A = 1$ and $B = 0$, then $\text{Sum} = 1$
Building an Adder

- If $A = 1$ and $B = 1$, then $\text{Sum} = 0$. 
Building an Adder

- This circuit has proved so important that it is given its own name.
- This gate is called an eXclusive OR and is given the symbol: $\oplus$ (in a logic statement $\oplus$).
Building an Adder

- Complete the binary addition problem:

  \[
  \begin{array}{c}
  1 \\
  + \ \\
  1 \\
  \hline
  1 \\
  \end{array}
  \]

\[
\begin{array}{cccc}
1 & 0 & 1 & 0 \\
+ & 1 & 1 & 1 \\
\hline
1 & 1 & 1 \\
\end{array}
\]
Building an Adder

Examine the following binary addition problem:

```
  1 0 1 0
+ 1 1 1
-----
  1 1 1 1
```
Building an Adder

Examine the following binary addition problem:

\[
\begin{array}{c}
1 & 0 & 1 & 0 \\
+ & 1 & 1 & 1 \\
\hline
0 & 1 \\
\end{array}
\]
Building an Adder

Examine the following binary addition problem:

```
  1
+ 1 0 1 0
+ 1 1 1
_______
  0 1
```
Building an Adder

- Examine the following binary addition problem:

\[
\begin{array}{c}
1 & 1 \\
1 & 0 & 1 & 0 \\
+ & 1 & 1 & 1 \\
\hline
0 & 0 & 1
\end{array}
\]
Building an Adder

- Examine the following binary addition problem:

```
  1 1
+ 1 0 1 0
   + 1 1 1
   ----
  1 0 0 0 1
```
Building an Adder

- When reviewing the example, the addition is not quite as simple as $A + B = \text{Sum}$.
- In many cases a “carry” impacts our addition.

```
  1 0 1 0
+ 1 1 1
-----
 1 0 1
```
Building an Adder

Looking at our example one column at a time we notice the following:

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

- Sum = $A \oplus B$ (A XOR B)
- Carry Out = $AB$ (A AND B)
Building an Adder

- To accurately reflect single digit binary addition our circuit is:

- Checking this circuit, it matches the truth table.
Building an Adder

- This is still not quite accurate.
- Going back to our binary addition problem, we find that in reality we are not adding two bits, and getting a two bit answer.
- Rather, our Carry Out from one column become the Carry In to the next column.
Building an Adder

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
0 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

carry in: 1

carry out: 1

A \rightarrow B \rightarrow S
Building an Adder

\[
\begin{array}{c}
0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

carry in

\[\Rightarrow \hspace{1cm} \Rightarrow \hspace{1cm} A \]

\[\Rightarrow \hspace{1cm} \Rightarrow \hspace{1cm} B \]

\[\Rightarrow \hspace{1cm} \Rightarrow \hspace{1cm} S \]

carry out
Building an Adder

\[
\begin{array}{cccccc}
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & & & &
\end{array}
\]

- carry in
- A
- B
- S
- carry out

Copyright © 2008 by Helene G. Kershner
Building an Adder

\[
\begin{array}{cccccc}
1 & 1 & 1 & 0 & 1 & \text{carry in} \\
0 & 1 & 0 & 1 & 0 & 1 \\
+ & 0 & 0 & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 0 & \text{S} \\
0 & 1 & 1 & 1 & 0 & 1 & \text{carry out}
\end{array}
\]
Building an Adder

- Adding two 1-bit numbers together turns out to really require that we add 3-bits together (including our Carry In) and producing an answer that is 2-bits (including our carry out).
- A complete or “full” adder 3-bits, Cin, A, B and produces a Sum and a Carry Out.
Building an Adder

The truth table that represents all the possible outcomes of these bits would look like this: This table can be represented by the following logic statements:

- **Sum** = (Cin $\oplus$ (A $\oplus$ B))
- **Cout** = (A $\oplus$ B) (Cin) + AB

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

Copyright © 2008 by Helene G. Kershner
Building an Adder

The circuit that represents this truth table, is called a Full Adder.
Building an Adder

Another way to look at this is:
Building an Adder

- A Full Adder then looks like this:

```
Full Binary Adder

Cin  \arrow{Sum}
A  \arrow{Carry Out/Carry In}
B
```
Building an Adder

- Going back to our binary addition, we find that a Full Adder, adds one column at a time.

```
  1  1  1  0  1  0
    0  1  0  1  0  1
  +  0  0  1  1  0  1
  ________
  1  0  0  0  1  0
    0  1  1  1  0  1
```

Copyright © 2008 by Helene G. Kershner
To perform real binary addition, a series of binary adders are linked together. The number of linked adders is determined by the manufacturer and reflects the finite-length a particular computer can handle. This is usually, 8-bits, 16-bits, 32-bits or more.
Building an Adder

A series of Full Adders would look like this: