### Bitwise Operations

**Boolean-like operations**

- Bitwise operators (left and right hand operands are integers)
- & (bitwise and) \(1100 \& 1001 = 1000\)
- | (bitwise or) \(1100 \mid 1001 = 1101\)
- ^ (exclusive or) \(1100 \^ 1001 = 0101\)
- ~ (complement) \(\sim 1100 = 0011\)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a &amp; b</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a ^ b</th>
<th>a</th>
<th>~a</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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</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>1</td>
<td>1</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>
<td>1</td>
</tr>
</tbody>
</table>

### Bitwise AND

**Used to extract or unpack information**

- Bitwise AND with a bit-mask is used to mask out or switch bits off
- Mask is usually given as a symbolic constant or in hexadecimal (or occasionally octal)

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
\]

(data)

\[
\begin{array}{cccccc}
\& \\
\end{array}
\]

(mask = 00011111)

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

(result)

### Bitwise OR

**Used to encode or pack information**

- Bitwise OR with a bit-mask is used to switch bits on
- Mask is usually given as a symbolic constant or in hexadecimal (or occasionally octal)

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
\]

(data)

\[
\begin{array}{cccccc}
\text{mask} = 11100000 \\
\end{array}
\]

(result)

### Bit-shift Operators

**Encode or decode information, hashing, CRC’s**

- << (left shift) \(11001100 \ll 2 = 00110000\)
- Discards bit shifted out on the left
- Pads with 0’s on the right
- >> (right shift)
- Discards bits shifted out on the right
- Pads with 0’s on the left if the number is unsigned (typical) or if the hardware doesn’t perform sign extend
- Copies the left-most bit the number is signed and the hardware performs sign extend

\[
\begin{array}{cccccc}
11001100 >> 2 = 00110011 \\
\end{array}
\]

unsigned or w/o sign extend

\[
\begin{array}{cccccc}
11001100 >> 2 = 11110011 \\
\end{array}
\]

signed and sign extend

\[
\begin{array}{cccccc}
00110011 >> 2 = 00001100 \\
\end{array}
\]

signed, unsigned, or sign extend