1. Cover only the 1s by drawing maximally sized rectangles whose dimensions (in cells) are powers of 2.
2. For each box, make a product of the inputs (or input complements) that are 1 for all cells in the box. (minterms)
3. Take the sum of these products.

Find sum of products visually

Arithmetic Logic Unit (ALU)

Let's implement some MIPS instructions!

1-bit adder

How do we implement this with the tools we have so far?
1-bit ALU

1-bit ALU implementing: AND, OR, addition.

How can we build a 32-bit ALU?

The ripple carry adder

Adding subtraction

\[ a - b = a + (-b) \]

How can 2's complement make this easy?

NOR and NAND

A MIPS ALU also needs a NOR function. We implement this by noting that \( a + b = \overline{a \cdot b} \).

What about NAND?
Comparisons?

Set on less than (\texttt{slt}): 
1 \text{iff } a < b, 
0 \text{ otherwise.}

In other words: 
1 \text{ iff } a - b < 0 
0 \text{ otherwise.} 
Use sign bit of \([a - b]!\)

Is that it? (PS7...)

MSB 1-bit ALU

Potentially nonzero only on LSB. 
Outputs used only on MSB.

32-bit ALU

[Illustrates what we do with that blasted Set out and how the Less values get set.]

Supporting \texttt{beq/bne}

How do we test if 
\(A \rightarrow B? A \rightarrow B?\)
Black box representing ALU

<table>
<thead>
<tr>
<th>ALU control lines</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>AND</td>
</tr>
<tr>
<td>0001</td>
<td>OR</td>
</tr>
<tr>
<td>0010</td>
<td>add</td>
</tr>
<tr>
<td>0110</td>
<td>subtract</td>
</tr>
<tr>
<td>0111</td>
<td>set on less than</td>
</tr>
<tr>
<td>1100</td>
<td>NOR</td>
</tr>
</tbody>
</table>

Ripple-carry addition is expensive.

Compute CarryIn₁₁

Carry lookahead

Carry generate  \( g_i = a_i \cdot b_i \)
Carry propagate  \( p_i = a_i + b_i \)
Using these define  \( c_{i+1} = g_i + p_i (c_i) \)

Unwind the recursion to a sequence of equations only four gates deep.