Creating the combinational circuit for add: the full adder

Example of adding two binary numbers:

```
    0 0 0 1 1 1 1 0 0 1      carry in
    1 0 0 0 1 1 0 1 1 0 1    A
+   1 1 0 0 0 1 1 1 0 0 1    B
  1 0 1 0 1 0 1 0 0 1 1 0
```

Eight cases:

```
  0      1      0      1      0      1      0      1
  0      0      0      0      1      1      1      1
+ 0    + 0    + 1    + 1    + 0    + 0    + 1    + 1
 00     01     01     10     01     10     10     11
```

Perform the add in two stages:

Add A and B digits first:

```
  0      0      0      0      1      1      1      1
+ 0    + 0    + 1    + 1    + 0    + 0    + 1    + 1
 00     00     01     01     01     01     10     10
```

Use only the "sum" digit (the one on the right) from stage 1 (the top row, below) and add the carry in (the bottom row, below):

```
  0      0      1      1      1      1      0      0
+ 0    + 1    + 0    + 1    + 0    + 1    + 0    + 1
 00     01     01     10     01     10     00     01
```

Notice that the "sum" digit (the one on the right) is the correct final "sum" digit. But the "carry" digit is not always the correct final "carry out" digit. Specifically, the last two cases produce the wrong "carry out". But these were the two cases that had a carry of one in the first stage, which we ignored in the second stage. The correct final "carry out" digit can be obtained by OR-ing the carry digits from the two stages.