University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

The Ripple Carry Adder

Build an Addition Device Based on Human Addition

Weeks ago, we talked about a "hardware device" to perform addition.

Now, you're ready to design it.

Let's start by reviewing the human approach.

Basing a design on the human approach

- is usually the **easiest way**, and
- **often** leads to **a good design**, too.
- (Humans are pretty smart.)

Example: Addition of Unsigned Bit Patterns

Let's do an example with **5-bit unsigned**

```
   11
   01110  (14)
 + 00100  (4)
 -------
   10010  (18)
```

Good, we got the right answer!

Name Signals (Bits) for Our Hardware Design

Let's do an example with **5-bit unsigned**

```
carry C      11000
      A      01110
      B    + 00100
           -------
  sum S      10010
```

There is no "blank" bit.

Each 1-bit sum needs a C input.

Good, we got the right answer!

For least significant bit, `C←0`.

For other bits, `C` comes from next bit to right.

Inputs and Outputs for a Full (One-Bit) Adder

Think about adding a single bit (a column).

A **full adder*** has **three inputs**

- **A** (one bit of the number **A**)
- **B** (one bit of the number **B**)
- C_{in} (a carry input from the next least significant bit, or 0 for bit 0)

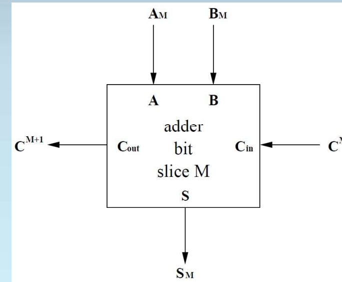
And a full adder produces **two outputs**

- C_{out} (a carry output for the next most significant bit)
- **S** (one bit of the sum **S**)

*A one-bit adder is called a "full adder" for historical reasons. A "half adder" adds two bits instead of three.

Connecting the Full Adder to the N-Bit Problem

Consider bit **M** of the addition (bit 0 is on the right, bit 1 to the left of bit 0, and so forth).



We need to add A_M, B_M, and C^M to produce bit S_M, of the sum and bit C^{M+1}, the carry into bit **M+1** of the addition.

Write a Truth Table for Full Adder Outputs

Let's calculate the outputs for a full adder.

You may remember solving this truth table a few weeks ago.

But let's do it again…

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fill a K-map for C_{out} from the Truth Table

Now fill in the truth table for C_{out}.

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

C_{out}

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Solve the K-map to Find C_{out}

And find the loops.

So we can write $C_{out} = AB + AC_{in} + BC_{in}$

(called a majority function, by the way).

C_{out}

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

The Sum is Best Written as an XOR

What about S? We can (of course) use another K-map.

But a K-map doesn't give us the best answer in this case (a rare case!).

S is 1 when an odd number of inputs are 1.

So $S = A \oplus B \oplus C$.

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

* *

XOR Shows up as a Checkerboard Pattern

Here's the K-map for S. Notice the checkerboard pattern of the XOR.

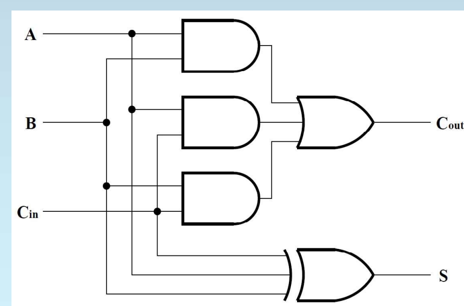
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Circuit for a Full Adder Using AND, OR, and XOR

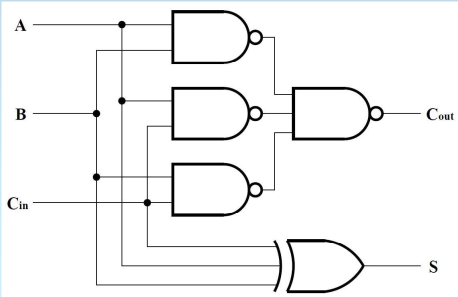
We can draw our full adder using AND, OR, and XOR.



CMOS Implementation Using NAND and XOR

In CMOS, we replace AND/OR with NAND/NAND.

The XOR remains as an XOR gate.



How Do We Add N Bits?

Use a full adder for each of the **N** columns.

Feed a **0** into $\mathbf{C_{in}}$ for the least significant bit.

$\mathbf{C_{out}}$ of the most significant bit is the adder's carry out.

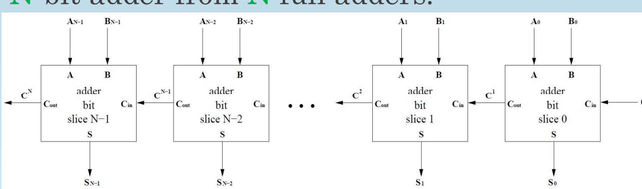
For the other carry signals, connect $\mathbf{C_{out}}$ of each bit to $\mathbf{C_{in}}$ of the next most significant bit.

Divide the bits of **A** and **B** amongst the full adders.

Collect the bits of **S** from the full adders.

Use N One-Bit Adders to Build an N-Bit Adder

The figure below illustrates construction of an **N**-bit adder from **N** full adders.



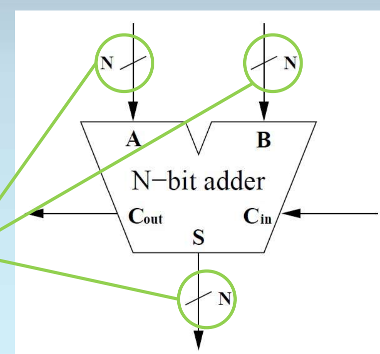
This approach is called a **ripple carry adder** because the carry ripples slowly from low to high. We also call it a bit-sliced adder.

Symbol for an N-Bit Adder

We draw an **N-bit adder** as shown here.

Note the shape.

Note also the crosshatch and bit width ("N") for multi-bit signals.



To Build a Bigger Adder, Just Connect C_{out} to C_{in}

We can build bigger adders by connecting adders together physically (as shown below) or virtually (by saving the carry out bit and using it as the carry in to the next adder).