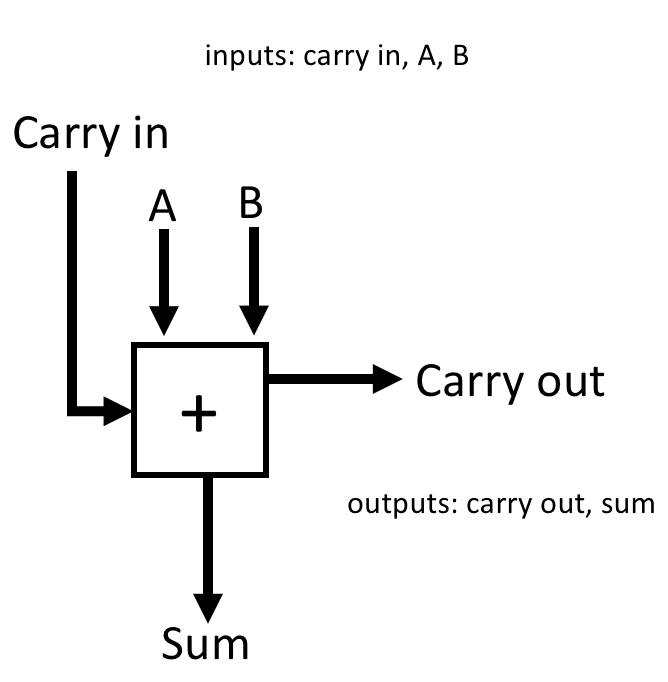
# Why?

You've built some useful combinational logic functions, such as multiplexors (Muxes), out of gates. Now you'll build arithmetic.

# Model 1: 1-bit adder

Consider a 1-bit adder. In computes the sum of two 1-bit numbers A and B. If (addition) CarryIn+A+B is larger than 1, then CarryOut will be 1. So, you can think of Sum as the 1's place and CarryOut as the 2's place.



1. How many rows are there in the truth table for Sum? For CarryOut? Explain.

Sum: 2^3 = 8

CarryOut: 2^3 = 8

**Manager:** assign half your team to answer #2,3 and half your team to answer #4,5.

1. Write the truth table for Sum.

|  |  |  |  |
| --- | --- | --- | --- |
| **Carry In** | **A** | **B** | **Sum** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

1. Write a boolean equation for Sum. You do not need to simplify it.

C + B + A + A\*B\*C

1. Write the truth table for CarryOut

|  |  |  |  |
| --- | --- | --- | --- |
| **Carry In** | **A** | **B** | **Carry Out** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

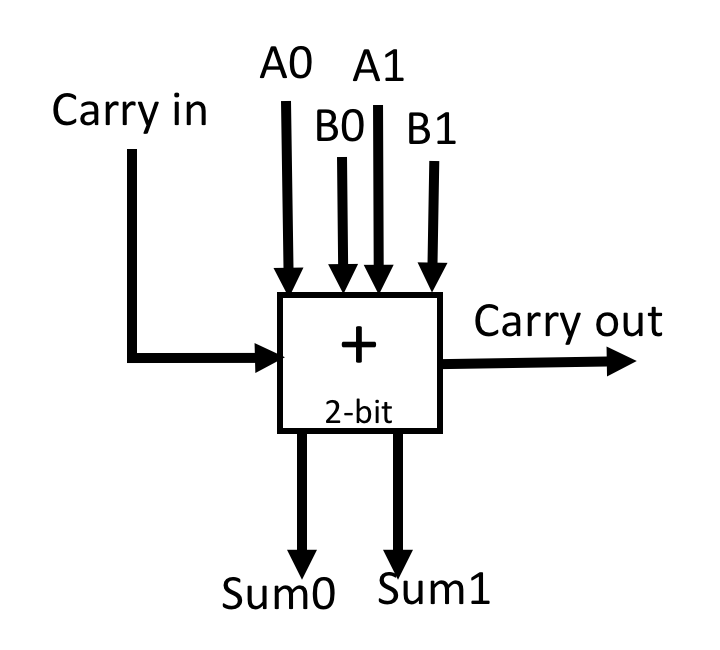
1. Write a boolean equation for CarryOut. You do not need to simplify it.

B\*C + A\*C + A\*B + A\*B\*C

**Manager:** have the two groups check each other's answers to 2,3 and 4,5.

# Model 2: 2-bit adder

Now consider the 2-bit adder. It computes the Sum of A and B, where A1 is the 2's bit of A and A0 is the 1's bit of A (and same for B and Sum).



1. How many rows are there in the truth table for Sum1? Explain.

2^5

1. Consider a 3-bit adder. How many rows are there in the truth table for Sum2?

2^7

1. Consider an N-bit adder. How many rows are there in the truth table for SumN-1?

2^(2N+1)

# Read This!

As you can see, it is not feasible to use a single truth table to build combinational logic blocks with a large number of inputs.

# Extension questions

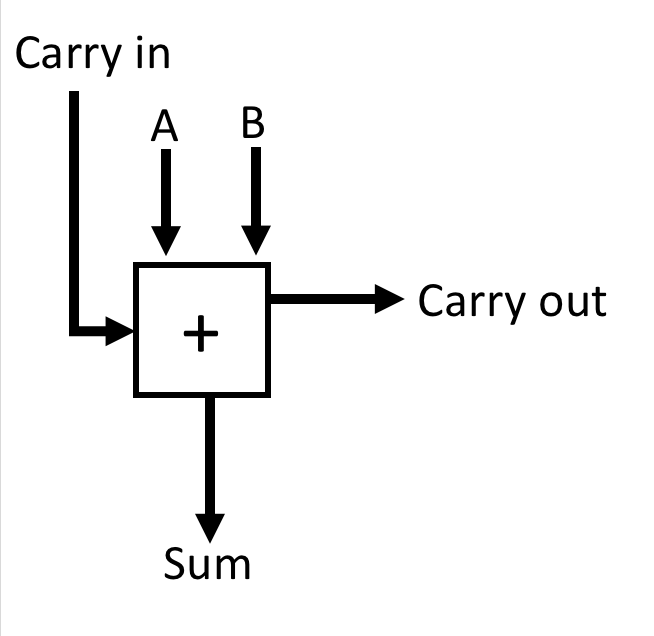
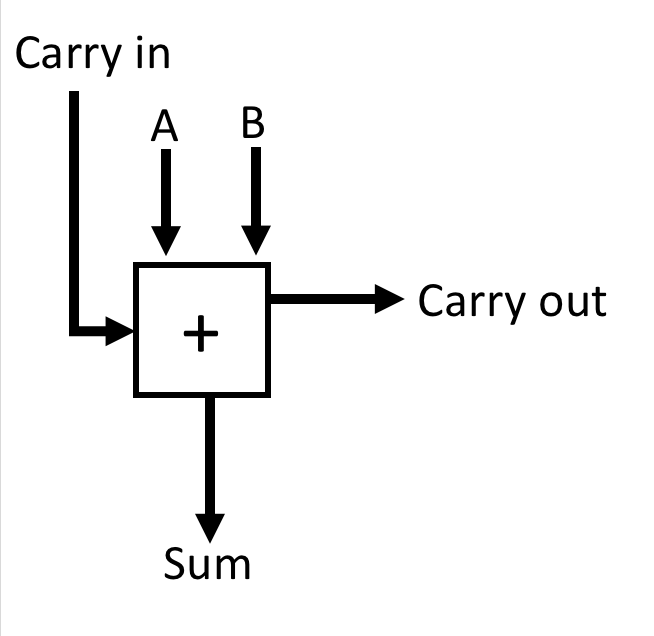
1. Consider a computer with 4GiB of memory, 16 4-byte registers, and 1 4-byte register for the PC. Suppose we wanted to build a combinational logic block that takes the current state (values of all the memory and registers) of the processor as input and produces the new state after one instruction executes. **How many rows are there in the truth table that calculates just the 1st bit of the 1st byte of memory?**

# Model 3: Ripple carry adder

Procedural algorithms to the rescue! Consider the fact that you probably didn't learn in school how to add two 10-digit numbers in a single go. Instead, you learned a procedure to perform addition step-by-step.

1. Compute 65210 + 36710 manually on paper or the board.
2. Compute 110102 + 010112 manually on paper or the board.
3. Write the general procedure for adding two multi-bit numbers.
4. Considering the procedure, how would you build a 2-bit adder out of two 1-bit adders? Draw only wires to finish the diagram.

*Inputs:* CarryIn A1 A0 B1 B0



*Outputs:* Sum1 Sum0 CarryOut

# Read This!

This design is called a ***ripple-carry adder***, and it works for any size of numbers you want to add.

# Exercises

1. Using your answers from previous models, how many AND, OR, NOT, XOR gates would a *3-bit* ripple-carry adder be made of?

# Extension questions

1. Make a small modification to the ripple-carry adder to change it from computing A+B to computing A-B.

* HINT 1: For a hint, go to <https://www.geocachingtoolbox.com/index.php?lang=en&page=asciiConversion> and enter From: Binary (Base2), To: Text (ASCII)

01110100 01110111 01101111 00100111 01110011 00100000 01100011 01101111 01101101 01110000 01101100 01100101 01101101 01100101 01101110 01110100

* HINT 2: For a hint, go to <https://www.geocachingtoolbox.com/index.php?lang=en&page=asciiConversion> and enter From: Binary (Base2), To: Text (ASCII)

01101110 01100101 01100111 01100001 01110100 01101001 01101111 01101110

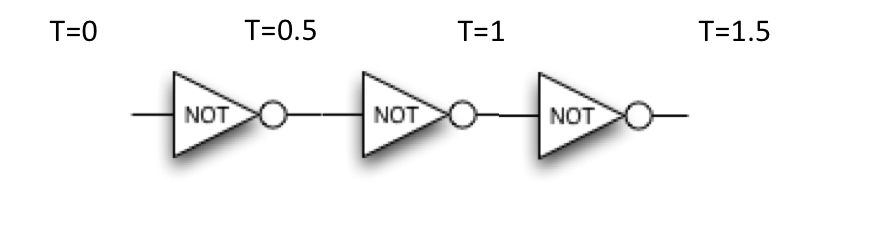
It is a fundamental result in complexity theory that for certain Boolean functions, such as add, the size of the circuit increases exponentially if the depth (number of stages) in the circuit is O(1) with respect to number of inputs.

# Model 4: Delay in a Ripple carry adder

For measuring the delay of a combinational logic circuit, we'll use the following simple model, derived from observations of RC-circuits.

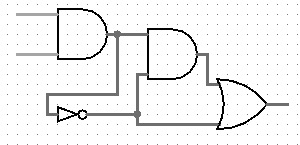
|  |  |
| --- | --- |
| **Rule** | **Justification** |
| propagation delay is higher for more complex gates | higher resistance and capacitance |
| propagation delay increases with the number of inputs to a gate | higher resistance in the pMOS or nMOS network |
| propagation delay of a sequence of gates is additive | pMOS doesn't go from off to on until the input to its gate gets close to 0 (i.e. previous gate is nearly done computing). Similar argument for nMOS. |
| all inputs must be available before logic gate begins computing | a single input could cause the final output to change |

Example: assume an inverter delay of 0.5 nanoseconds (ns).



# Exercises

1. Label the diagram with the time at which each wire has its final value. The two inputs to the first AND are available at time=0. The delays are AND=2, OR=2, NOT=1.





1. If the 32-bit inputs A and B are available at time=0, then at what time is Sum31 available? 1-bit adder delays: sum=2, carry=1.

