Additional Circuits (Sub-circuits, Comparators and Adders)

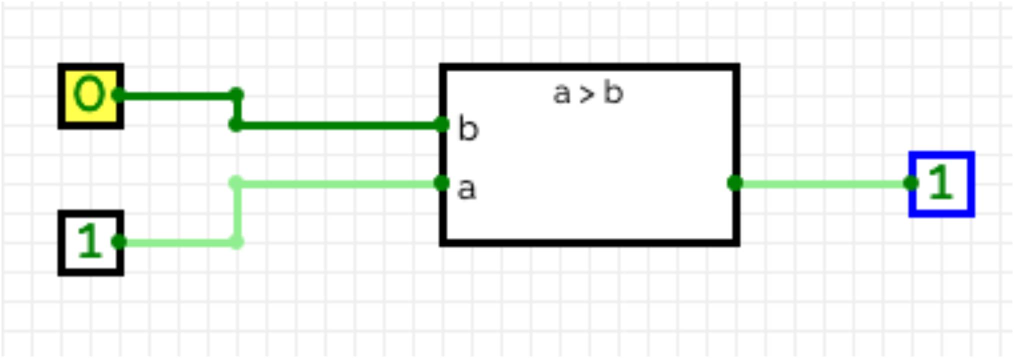
Garrett Dancik, PhD

Fall 2021

Course Notes: https://gdancik.github.io

Sub-circuits

- Sub-circuits are self-contained circuits that can be reused
- A sub-circuit is denoted with a box having one or more inputs and outputs, and behaves like a "black box" that carries out an operation.



- The above circuit implements $a > b$, where a and b are each 1 bit.
- Can you design a circuit that outputs $a > b$ when a and b are each 2 bits, denoted as a_1a_2 and b_1b
  - Note that $a > b$ if either of the following are TRUE
    - $a_1 > b_1$
    - NOT $a_1 > b_1$ AND $a_2 > b_2$

Constructing a circuit using sub-circuits

- A circuit for the 1 bit comparison of *a* > *b* is available here:
  - https://circuitverse.org/users/89029/projects/a-gt-b-1-bit

- In CircuitVerse, use this circuit to implement the following Boolean expression, which outputs 1 if *a* > *b* and outputs 0 otherwise, where *a* and *b* are each 2 bits.

    - $(a_1 > b_1)$ OR (NOT $a_1 > b_1$ AND $a_2 > b_2$)

- To do this, fork the above circuit, then select Circuit → New Circuit. Give the new circuit an appropriate name (like "a > b (2 bits)"). Then select Circuit → Insert SubCircuit, and select the "a > b" circuit. This sub-circuit is a "black box" that has 2 inputs (a and b) and outputs the value of a > b. Use as many sub-circuits as necessary to implement the above expression.

A circuit can have multiple outputs

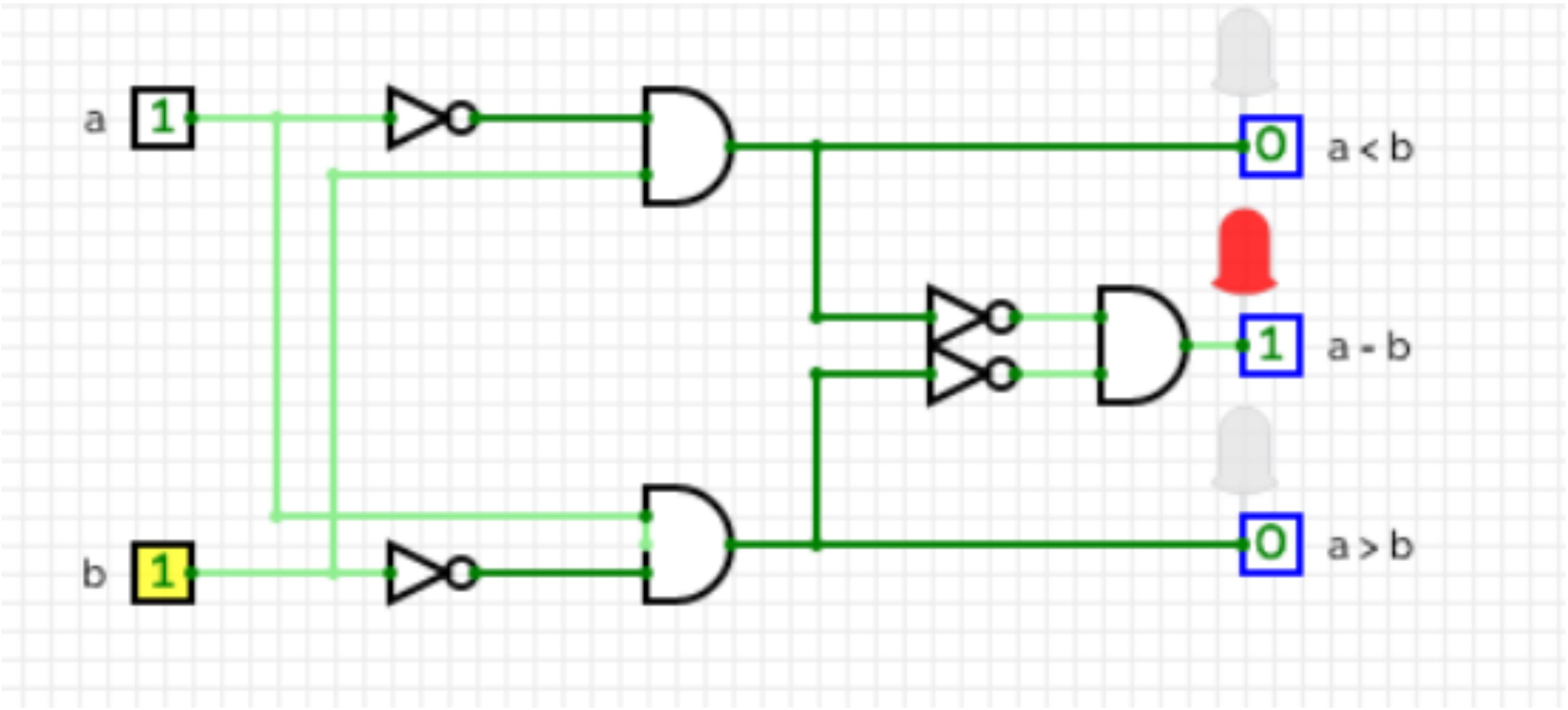
- Example: Single bit magnitude comparator
- Outputs:
  - L: $a < b$ (a is less than b)
  - E: $a = b$ (a is equal to b)
  - G: $a > b$ (a is greater than b)

Inputs		Outputs		
a	b	L: a < b	E: a = b	G: a > b
1	1	0	1	0
1	0	0	0	1
0	1	1	0	0
0	0	0	1	0

Boolean expressions

L = NOT *a* AND *b*
E = NOT *L* AND NOT *G* [we could also use (*a* AND *b*) OR (NOT *a* AND NOT *b*)
G = *a and NOT b*

Circuit for single bit magnitude comparator



https://circuitverse.org/users/89029/projects/single-bit-magnitude-comparator-9d0c3438-7d7a-4862-841f-038685045510

Circuits for adding numbers: Truth table for a half-adder

Half-adder truth table for adding two binary digits

Inputs		Outputs	
a	b	S (Sum)	Cout (Carry output)
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0

Binary addition:

```
  11   ← carry
1011
  11
1110
```

Boolean expressions:

S = *a* XOR *b*

Cout = *a* AND *b*

The half-adder has two inputs (sometimes called the augend and addend) and two outputs, one for the sum (S) and one for the carry (Cout).

Unlike the full adder (next page), the half adder does not have an input for any previous carry.

Truth table for a full-adder

Full-adder truth table for adding two binary digits

Inputs			Outputs	
a	b	Cin	S	Cout
1	1	1	1	1
1	0	1	0	1
0	1	1	0	1
0	0	1	1	0
1	1	0	0	1
1	0	0	1	0
0	1	0	1	0
0	0	0	0	0

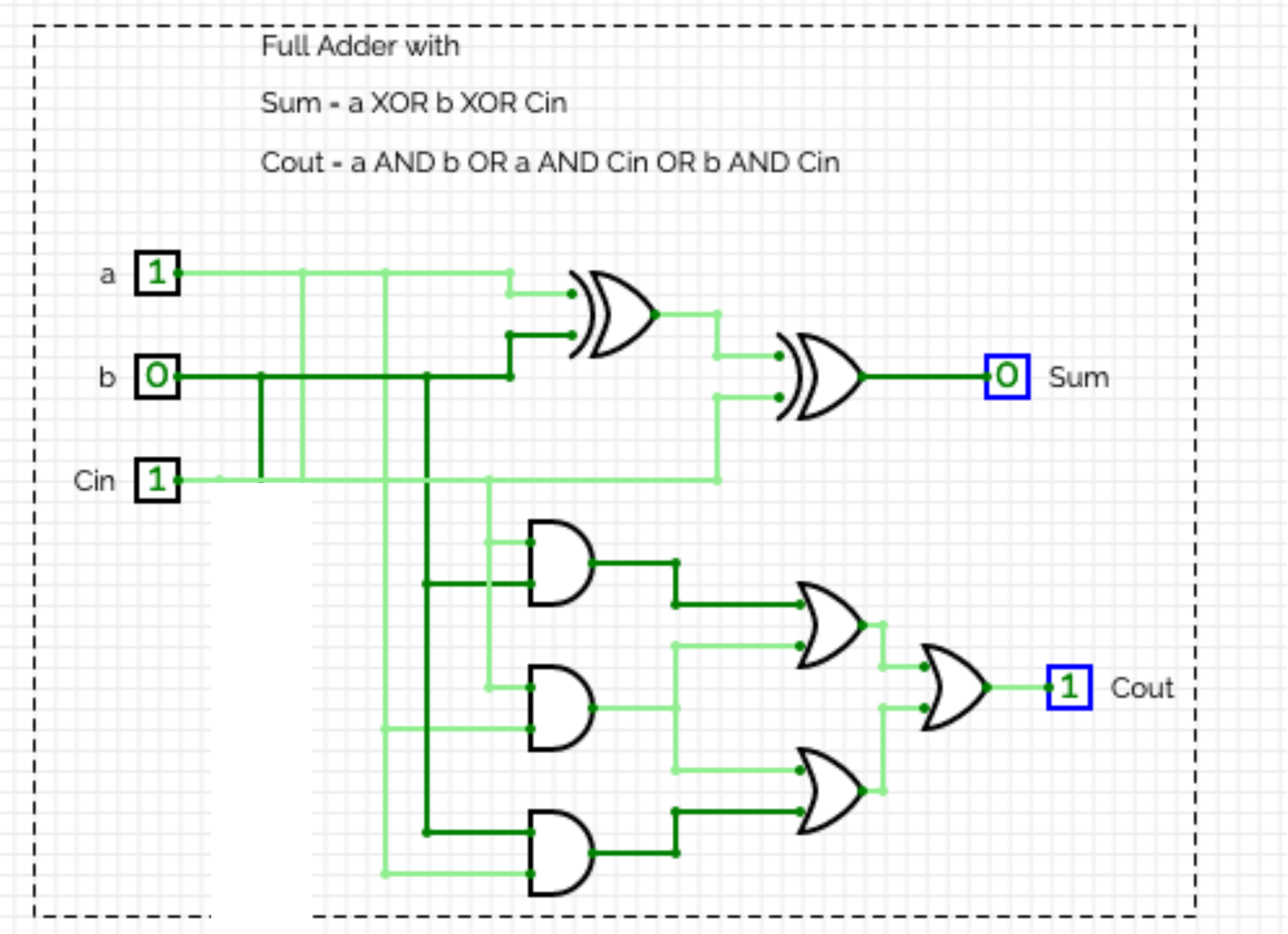
The sum is 1 if only one input is 1 or if all inputs are 1. This can be captured by XOR across all inputs.

S = *a* XOR *b* XOR *Cin*

The carry is 1 if at least 2 inputs are 1. We can therefore use AND to check each pair of inputs.

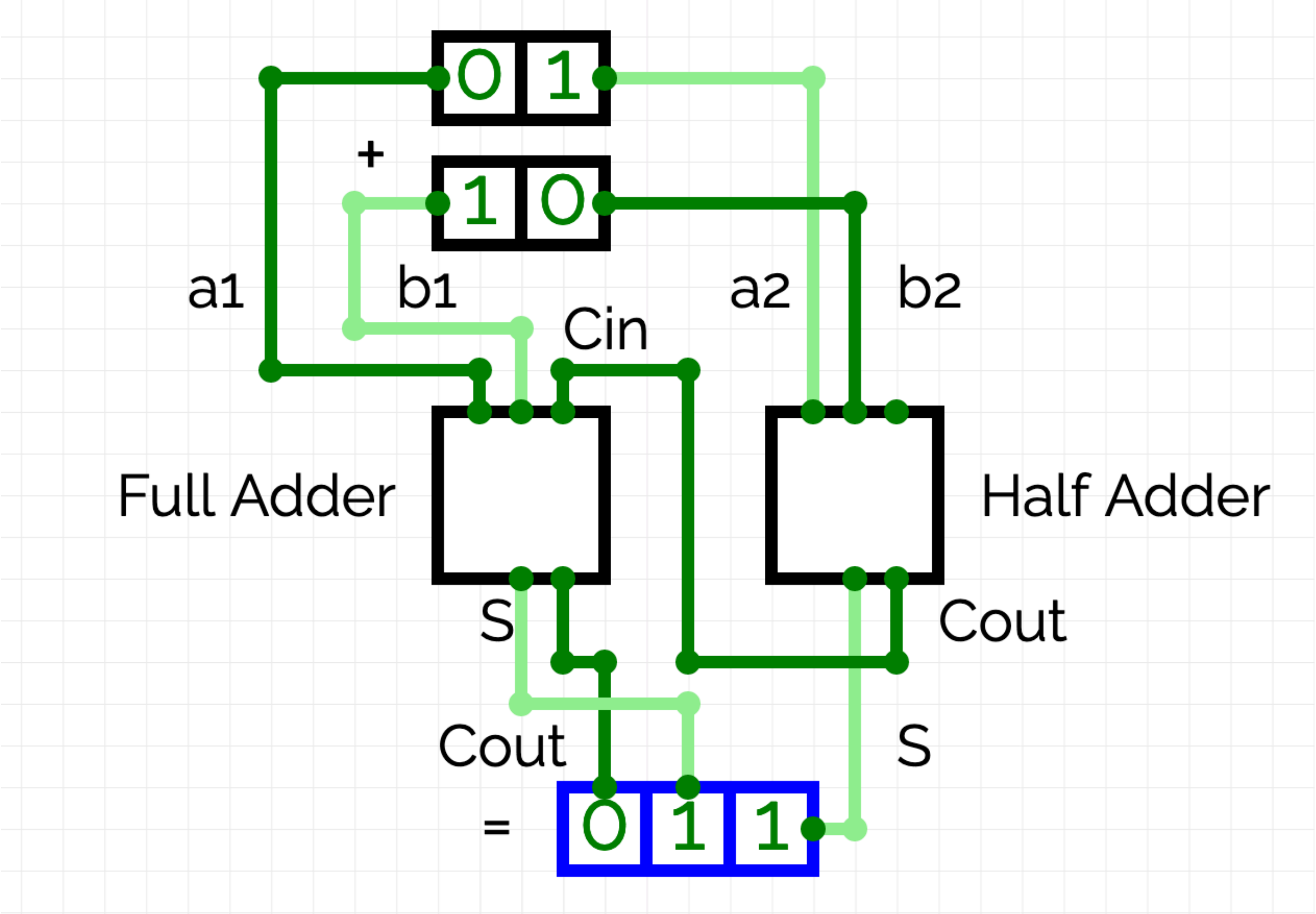
Cout = *a* AND *Cin* OR *b* AND *Cin* OR *a* AND *b*

The full adder has an input for the previous carry bit (Cin)



https://circuitverse.org/users/89029/projects/full-adder-9ea5e811-4d30-41d3-8652-8d5f2e7941d1

Chaining adder circuits for addition of multi-bit numbers



- https://circuitverse.org/users/89029/projects/2-bit-adder-1d2c96f2-834b-48d9-889f-16d7e36e1951