Announcements

- Let’s use 1 for True and 0 for False
- Homework #1 has been posted
  - Submit on GradeScope
  - Did you get the email?
  - How to scan and submit
- Office Hours are in room…

- Quiz tomorrow

Arguments

Recall: An **argument** is a conjecture that says:

If you make certain assumptions, then a particular statement must follow.

- The assumptions are called **premises**
- The statement that (supposedly) follows is the **conclusion**

Example:

$$\left.\begin{array}{l} p \vee q \\ q \to r \\ \sim p \end{array}\right\} \text{Premises}$$

$$\therefore\ r \qquad \text{Conclusion}$$

Validity

We say an argument is **valid** when:

Every interpretation that makes all of the premises true also makes the conclusion true.

Not all arguments are valid!

Is this argument valid? Let's check.

$$\left.\begin{array}{l} p \vee q \\ q \rightarrow r \\ \sim p \end{array}\right\} \text{Premises}$$

$$\therefore\ r \qquad \text{Conclusion}$$

We will need this today…

Given any statement variables p, q, and r, a tautology t and a contradiction c, the following logical equivalences hold:		
1. Commutative laws:	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. Associative laws:	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Distributive laws:	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Identity laws:	$p \wedge t \equiv p$	$p \vee c \equiv p$
5. Negation laws:	$p \vee \sim p \equiv t$	$p \wedge \sim p \equiv c$
6. Double negative law:	$\sim(\sim p) \equiv p$	
7. Idempotent laws:	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. DeMorgan's laws:	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
9. Universal bounds laws:	$p \vee t \equiv t$	$p \wedge c \equiv c$
10. Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. Negations of t and c:	$\sim t \equiv c$	$\sim c \equiv t$

Rules of Inference

Rules of inference are short arguments that are known to be valid. We will use them to *prove* the validity of more complex arguments.

Modus Ponens

$$\begin{array}{l} p \to q \\ \underline{p \qquad} \\ \therefore q \end{array}$$

Modus Tollens

$$\begin{array}{l} p \to q \\ \underline{\sim q \qquad} \\ \therefore \sim p \end{array}$$

Conjunction

$$\begin{array}{l} p \\ \underline{q \qquad\quad} \\ \therefore p \wedge q \end{array}$$

Transitivity

$$\begin{array}{l} p \to q \\ \underline{q \to r \quad} \\ \therefore p \to r \end{array}$$

Elimination

$$\begin{array}{l} p \vee q \\ \underline{\sim q \quad} \\ \therefore p \end{array} \qquad \begin{array}{l} p \vee q \\ \underline{\sim p \quad} \\ \therefore q \end{array}$$

Generalization

$$\begin{array}{l} \underline{p \qquad\quad} \\ \therefore p \vee q \end{array} \qquad \begin{array}{l} \underline{q \qquad\quad} \\ \therefore p \vee q \end{array}$$

Specialization

$$\begin{array}{l} \underline{p \wedge q} \\ \therefore p \end{array} \qquad \begin{array}{l} \underline{p \wedge q} \\ \therefore q \end{array}$$

Contradiction rule

$$\begin{array}{l} \underline{\sim p \to c} \\ \therefore p \end{array}$$

Proof by division into cases

$$\begin{array}{l} p \vee q \\ p \to r \\ \underline{q \to r} \\ \therefore r \end{array}$$

- You don't need to memorize this
- Posted on class webpage (under "resources")

Proof

Instead of using truth tables, we can try to **prove** the validity of an argument.

For now, a **proof** is a sequence of statements, beginning with the premises. Each subsequent statement must follow from the previous statements according to a valid “rule of inference” (or using one of the known equivalencies). The last statement should be the conclusion.

Practicing Formal Proofs

Let's prove the validity of these arguments:

P1: $p \vee q$

P2: $q \rightarrow r$

P3: $\sim p$

$\therefore$ r

P1: $p \wedge q$

P2: $p \rightarrow s$

P3: $\sim r \rightarrow \sim q$

$\therefore$ $s \wedge r$

- Do these examples represent proofs in the "real world"?
- Are the proofs in the rest of this course going to be this tedious, mechanical, and dull?

Interesting Question

Do you think there could be a **valid** argument (in propositional logic) that is **not provable** using the Equivalence Laws and Rules of Inference that we have on our charts?

Unit 2
Digital Circuits

Number Base Review

What are number bases?

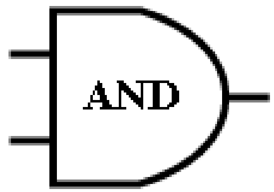
How do we convert a number from an arbitrary base into base 10?
How do we convert a number from base 10 into an arbitrary base?

We are mostly concerned with base 10 (decimal) and base 2 (binary).

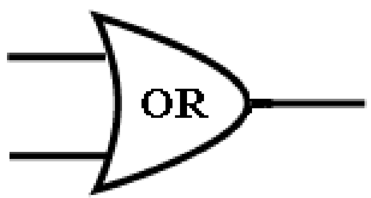
Basic logic gates

Computer circuits are comprised of “logic gates”. These are physical devices which we will consider in abstract. The “inputs” and “outputs” are bits (0’s or 1’s)

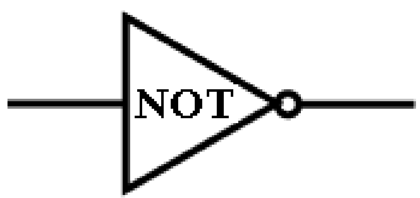
- An **and** gate:



- An **or** gate:

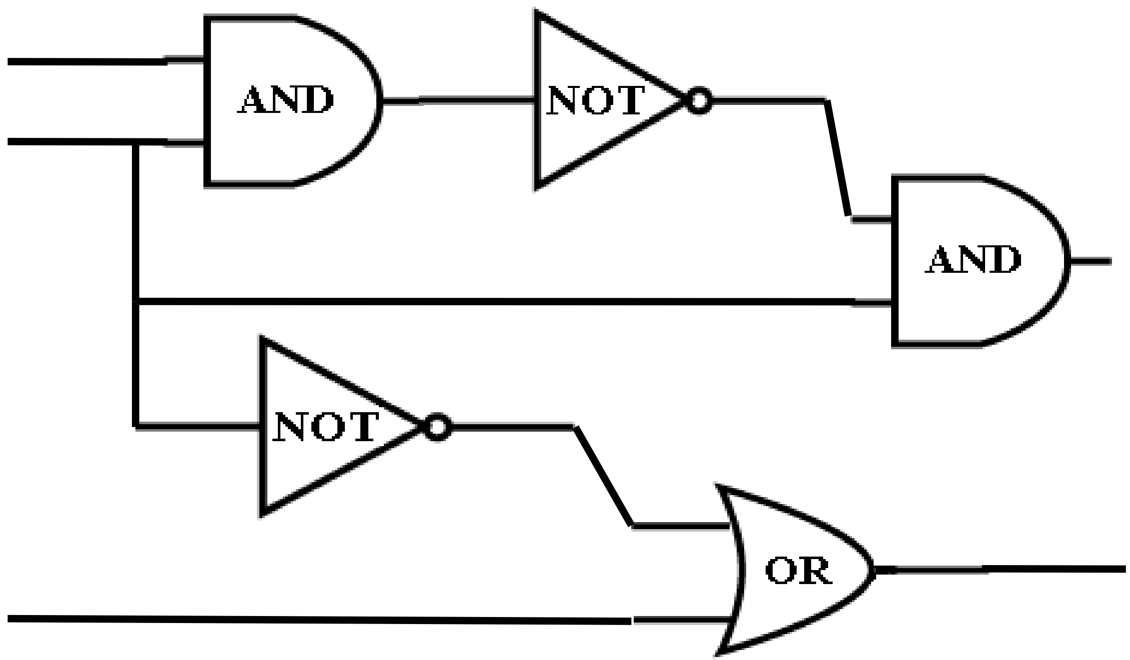


- A **not** gate:



Digital Circuits

Circuits are formed by combining logic gates.



- How many input bits?
- How many output bits?
- What is the output when the input is 110?

Propositional Logic and Circuits

Each statement of propositional logic can be represented by a circuit with one input for each variable, and a single output bit.

Practice making circuits for these:

- $p \vee \sim(q \wedge r)$
- $p \leftrightarrow q$

Hardware representing Truth Tables

- Any column in a truth table can be represented with a statement of propositional logic. How?
- Now any truth table can be built from an actual circuit.

Example:

p	q	r	output
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Circuits that Calculate

Circuits can perform math!

Examples:

- Addition of integers
- Multiplication of integers
- Compute $3x^4 + 2x^2 + 7$, where x is an integer
- Approximations of real-valued functions

Our goal today will be to build a circuit that can add numbers together:

Inputs: 77 and 49 (in binary)

Output: 126 (in binary)

Brute force: Addition by Truth Table

Adding 2-bit numbers:

	X		Y		Answer			
3 + 3	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**6**
3 + 2	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**5**
3 + 1	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**4**
3 + 0	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**3**
2 + 3	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**5**
2 + 2	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**4**
2 + 1	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**3**
2 + 0	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**2**
1 + 3	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**4**
1 + 2	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**3**
1 + 1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**2**
1 + 0	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
0 + 3	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**3**
0 + 2	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**2**
0 + 1	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
0 + 0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

- Now we can build a circuit with 4 input bits and three output bits.
- How big would this table be with 64-bit operands?
- Is there a more elegant approach?

Addition of binary numbers

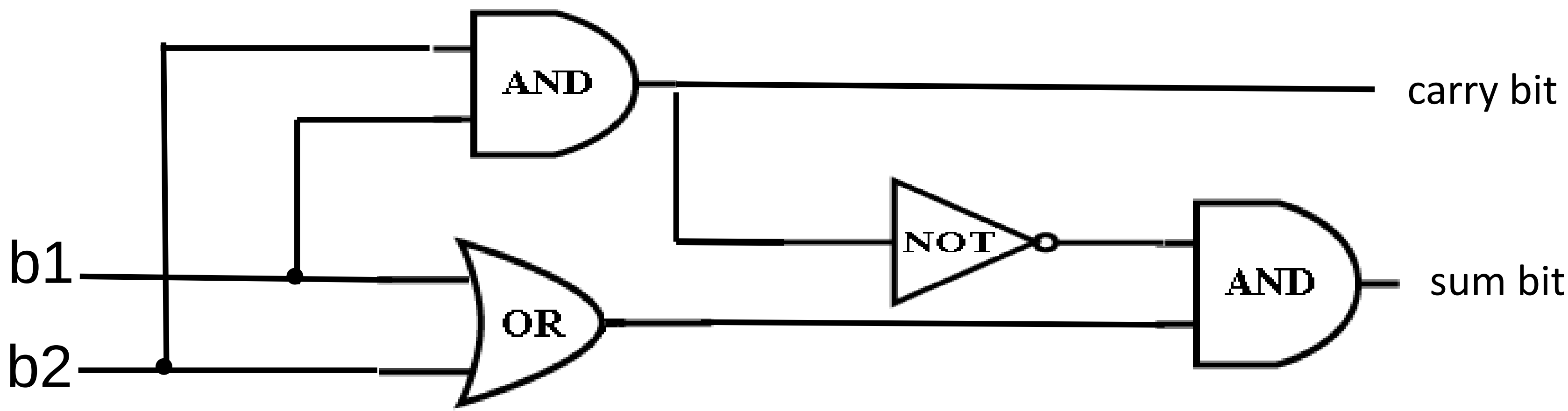
Practice:

$$\begin{array}{r} 1001 \\ +\ 0010 \\ \hline \end{array} \qquad \begin{array}{r} 1001 \\ +\ 0011 \\ \hline \end{array} \qquad \begin{array}{r} 1011 \\ +\ 0010 \\ \hline \end{array} \qquad \begin{array}{r} 1101 \\ +\ 0111 \\ \hline \end{array}$$

Can we create a circuit that models this process?

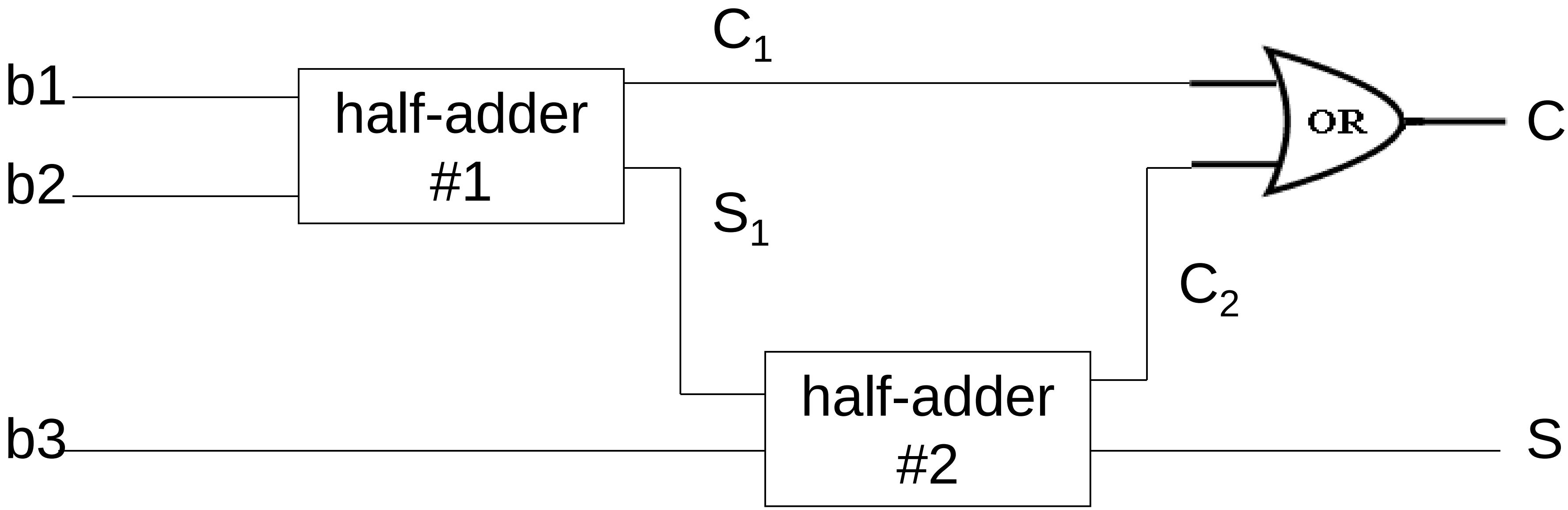
Half-Adder

Circuit that adds two bits together:



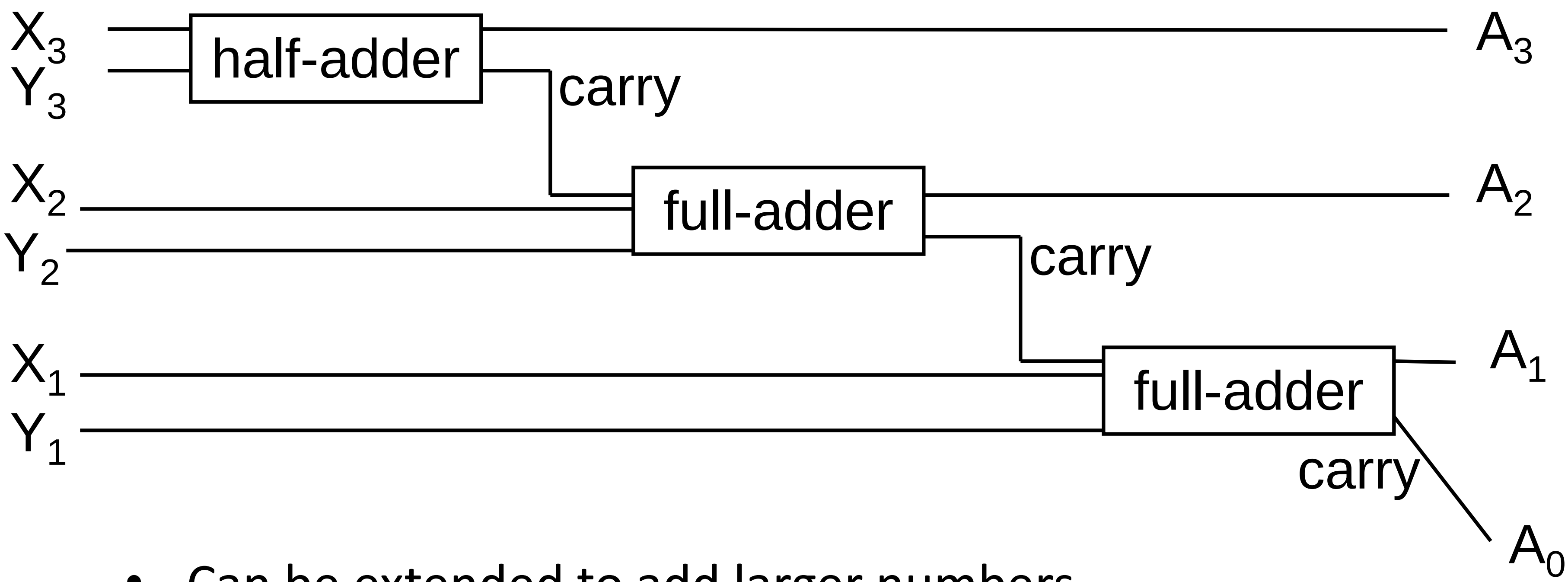
Full adder

Circuit that adds three bits together:



Parallel adder (for three bit operands)

$$\begin{array}{r} X_1X_2X_3 \\ +\ \ Y_1Y_2Y_3 \\ \hline A_0A_1A_2A_3 \end{array}$$



- Can be extended to add larger numbers