271: Introduction to Digital Circuits and Systems

- Professor Scott Hauck, EEB-307Q (hauck@uw.edu)
  - Office Hours: stop by or email w/schedule for a slot
- Recommended Book: Harris & Harris, *Digital Design and Computer Architecture* (Arm Edition, 2016)
- TAs (EEB-361):
  - Brian Hsu (brianhsu@uw.edu)
  - Nick Sycamore (ys264@uw.edu)
  - Pengyu Yang (yangp8@uw.edu)
- TA Office Hours: most times most weekdays (check website)

Grading

- 25% - Homeworks
- 40% - Labs
- 15% - Midterm Exam
- 20% - Final Exam
- Homework is due at the end of class on the specified date.
- Late penalties:
  - <24 hours: -10%
  - <48 hours: -30%
  - <72 hours: -60%
  - >72 hours: not accepted

Joint Work Policy

- Labs will be done alone, homeworks in groups of 1-2.
  - Students may not collaborate on labs/projects, nor between groups on the specifics of homeworks.
  - All submitted student work must be from their own efforts, and not any other source.
- OK:
  - Studying together for exams
  - Discussing lectures or readings
  - Talking about general approaches
  - Help in debugging, tools peculiarities, etc.
- Not OK:
  - Developing a lab together
  - Checking homework answers between groups
- Violation of these rules is <u>at minimum:</u>
  - Loss of twice the points of that assignment.
  - Report of Academic Misconduct to Dean's Level.
  - Potentially fail class, be expelled from UW.

Class & Lab Meetings

- Labs:
  - Each student assigned a lab kit, can work where-ever.
  - ***There are no specific assigned lab times.***
  - TAs have large blocks of office hours to help with labs, homeworks, class material, etc.
  - Signups for lab demos will be posted shortly.

- Labs are an integral portion of the class learning. Failure to make a good-faith effort at the labs is grounds for failing the class.

Motivation

- Readings: 1.1-1.3, 1.5-1.6.2, 2.1-2.2.2

- Electronics an increasing part of our lives
  - Computers & the Internet
  - Car electronics
  - Robots
  - Electrical Appliances
  - Cellphones
  - Portable Electronics
- Class covers digital logic design & implementation

Example: Car Electronics

- Door Ajar (DriverDoorOpen, PassDoorOpen):

- High-beam indicator (lights, high beam selected):

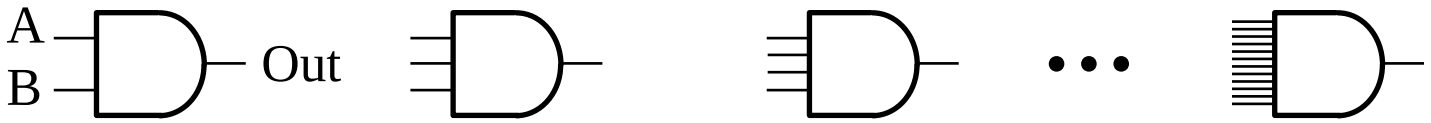
Example: Car Electronics (cont.)

- Seat Belt Light (driver belt in):

- Seat Belt Light (driver belt in, passenger belt in, passenger present):

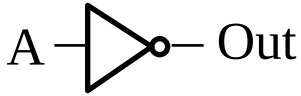
Basic Logic Gates

- AND: If all inputs are True (A and B), then Out is True

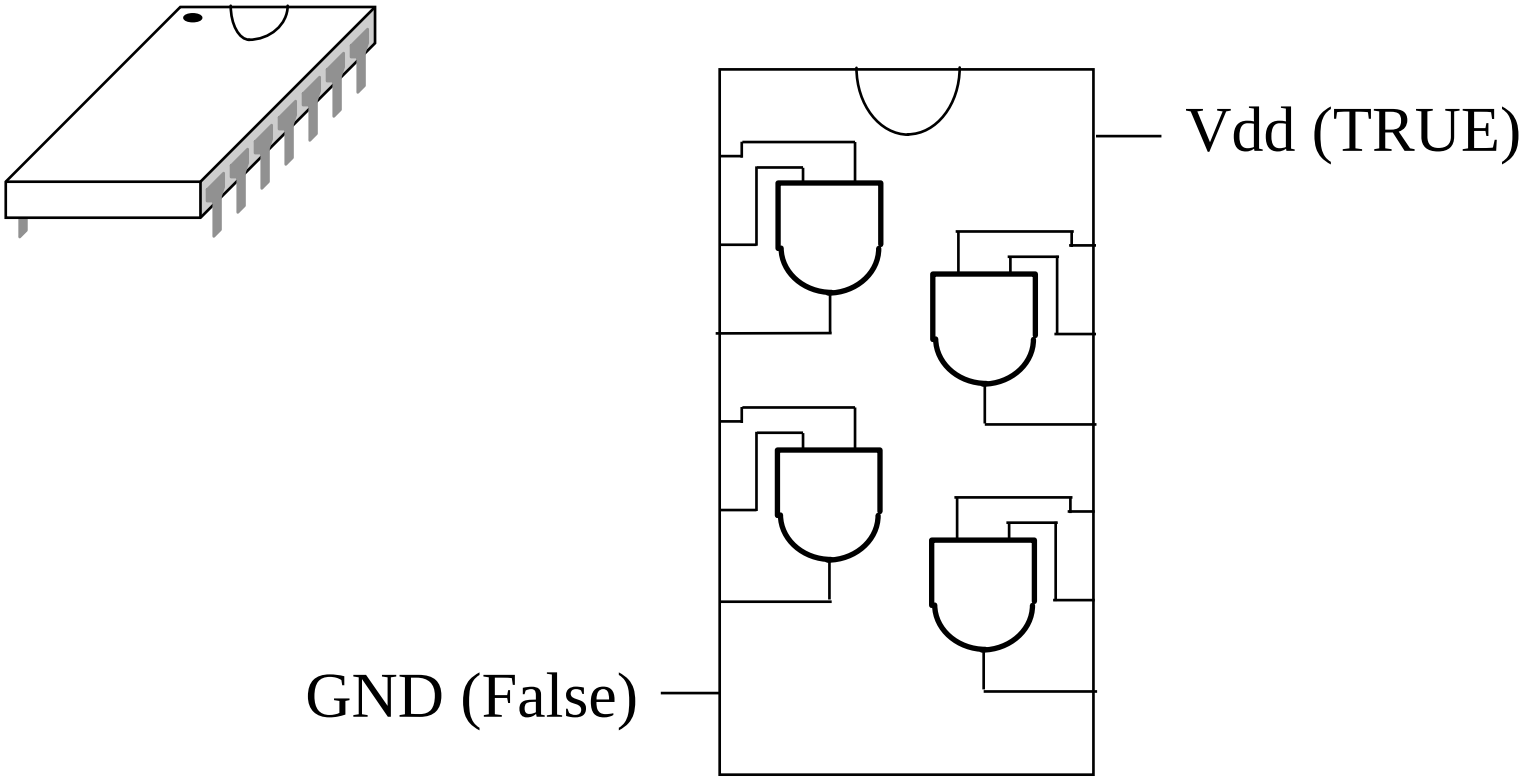


- OR: If any input is True (A or B), then Out is True

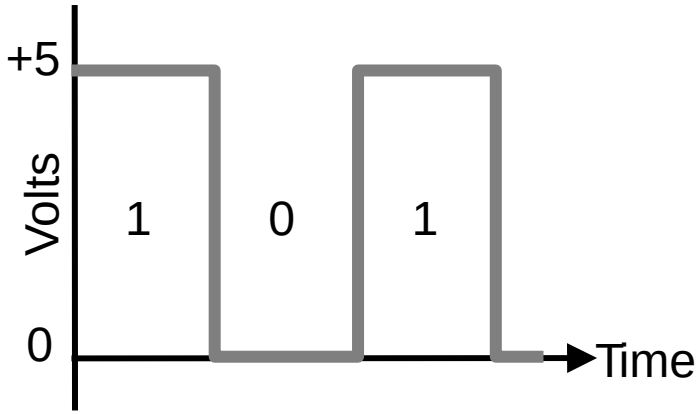
- Inverter (NOT): If A is False, then Out is True

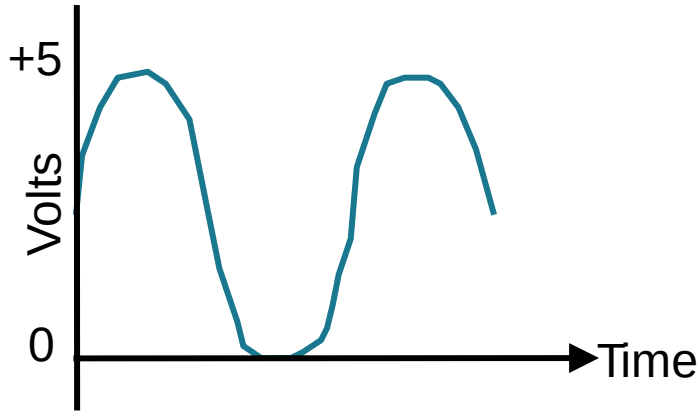


TTL Logic



Digital vs. Analog





Digital:
only assumes discrete values

Binary/Boolean (2 values)
yes, on, 5 volts, high, TRUE, "1"
no, off, 0 volts, low, FALSE, "0"

Analog:
values vary over a broad range continuously

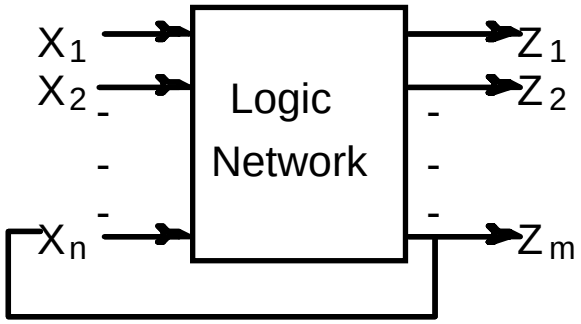
Advantages of Digital Circuits

- Analog systems: slight error in input yields large error in output
- Digital systems more accurate and reliable
  - Readily available as self-contained, easy to cascade building blocks
- Computers use digital circuits internally
- Interface circuits (i.e., sensors & actuators) often analog

This course is about logic design, not system design (processor architecture), not circuit design (transistor level)

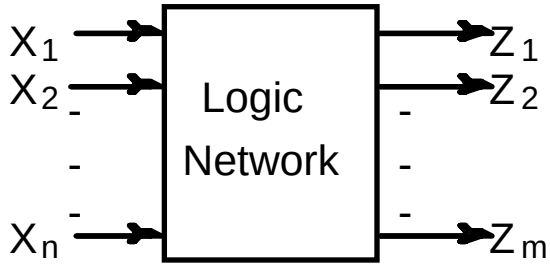
Combinational vs. Sequential Logic

Sequential logic



Network implemented from logic gates. The presence of feedback distinguishes between *sequential* and *combinational* networks.

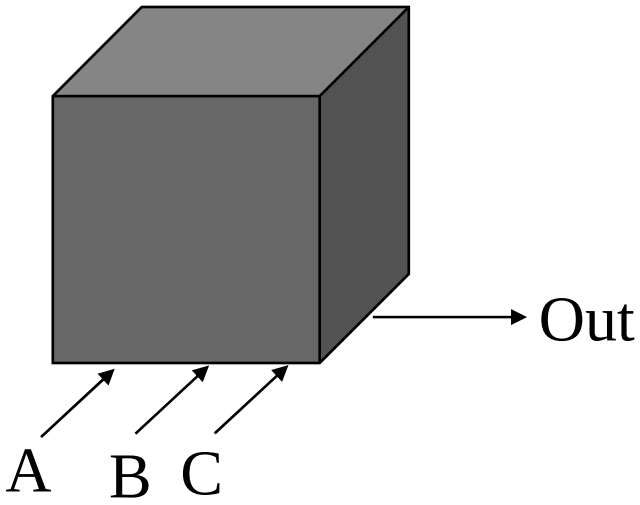
Combinational logic



No feedback among inputs and outputs. Outputs are a function of the inputs only.

Black Box (Majority)

- Given a design problem, first determine the function
- Consider the unknown combination circuit a "black box"



Truth Table

“Black Box” Design & Truth Tables

- Given an idea of a desired circuit, implement it
  - Example: Odd parity - inputs: A, B, C, output: Out

Boolean Elements

Algebra: **variables, values, operations**

In Boolean algebra, the values are the symbols 0 and 1
If a logic statement is false, it has value 0
If a logic statement is true, it has value 1

Operations: AND, OR, NOT

X	Y	X AND Y
0	0	
0	1	
1	0	
1	1	

X	NOT X
0	
1	

X	Y	X OR Y
0	0	
0	1	
1	0	
1	1	

Boolean Equations

Boolean Algebra

values: 0, 1
variables: A, B, C, . . ., X, Y, Z
operations: NOT, AND, OR, . . .

NOT X is written as $\overline{X}$
X AND Y is written as X * Y, or sometimes X Y or X & Y
X OR Y is written as X + Y

Deriving Boolean equations from truth tables:

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Carry =

Sum =

OR'd together *product* terms for each truth table row where the function is 1

if input variable is 0, it appears in complemented form; if 1, it appears uncomplemented

Boolean Algebra

Another example:

A	B	Cin	Cout	Sum
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

Sum =

Cout =

Boolean Algebra (cont.)

Reducing the complexity of Boolean equations

Laws of Boolean algebra can be applied to carry out function to derive the following simplified expression:

Cout =

	A	B	Cin	Cout
0:	0	0	0	0
1:	0	0	1	0
2:	0	1	0	0
3:	0	1	1	1
4:	1	0	0	0
5:	1	0	1	1
6:	1	1	0	1
7:	1	1	1	1

Verify equivalence with the original Carry Out truth table:

place a 1 in each truth table row where the product term is true

each product term in the above equation covers exactly two rows in the truth table; a row can be "covered" by more than one term