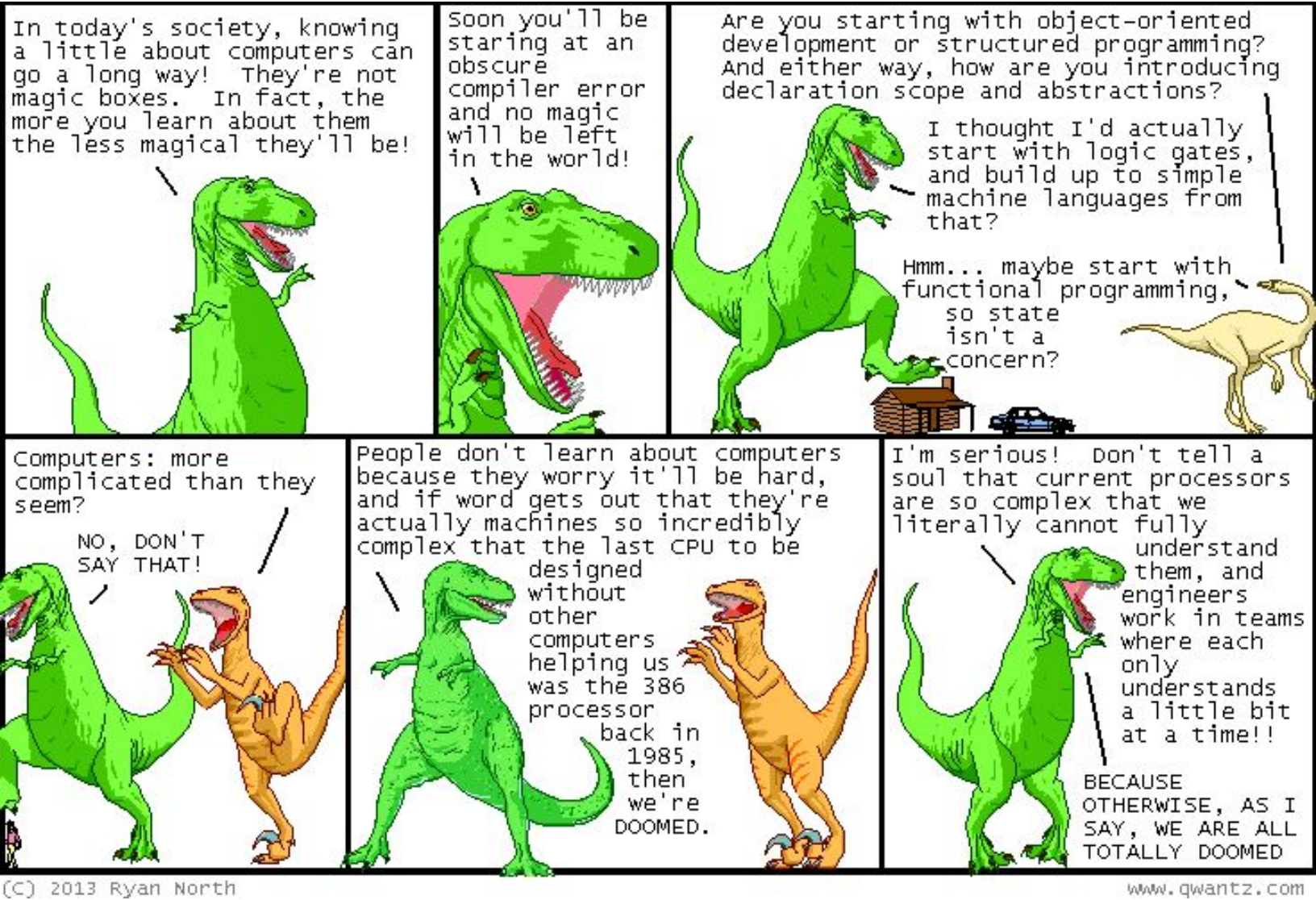


0x00 - Introduction

ENGR 3410: Computer Architecture

Jon Tse

Fall 2020

Who am I?

Ah, the eternal struggle…

Olin College, ECE, Class of ‘08
Cornell, ECE, 2016
Intel Labs

Infrastructure

Work on GitHub

Content/Assessment on Canvas

Discussion on Slack

Tools on Docker

Assessment

Weekly Homework

Multi-week Labs

Midterm

Final *Project* not Exam

Due Date: Mondays at Midnight Eastern

Recitations

Optional “mini lectures”

Offered by NINJAs on Wed 8-9 PM Eastern

Cover adjacent topics for enrichment

Expectations

This is 2-3 classes at other schools.

Staggered incoming knowledge base.

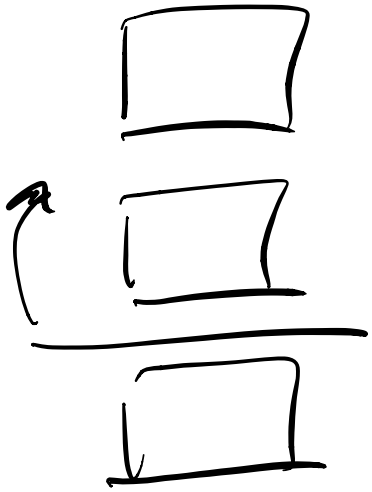
Bidirectional Feedback

“The only thing worse than bad news is bad news late!”

Heavy Lift

Optional Work

Intended to Challenge



Teams Allowed

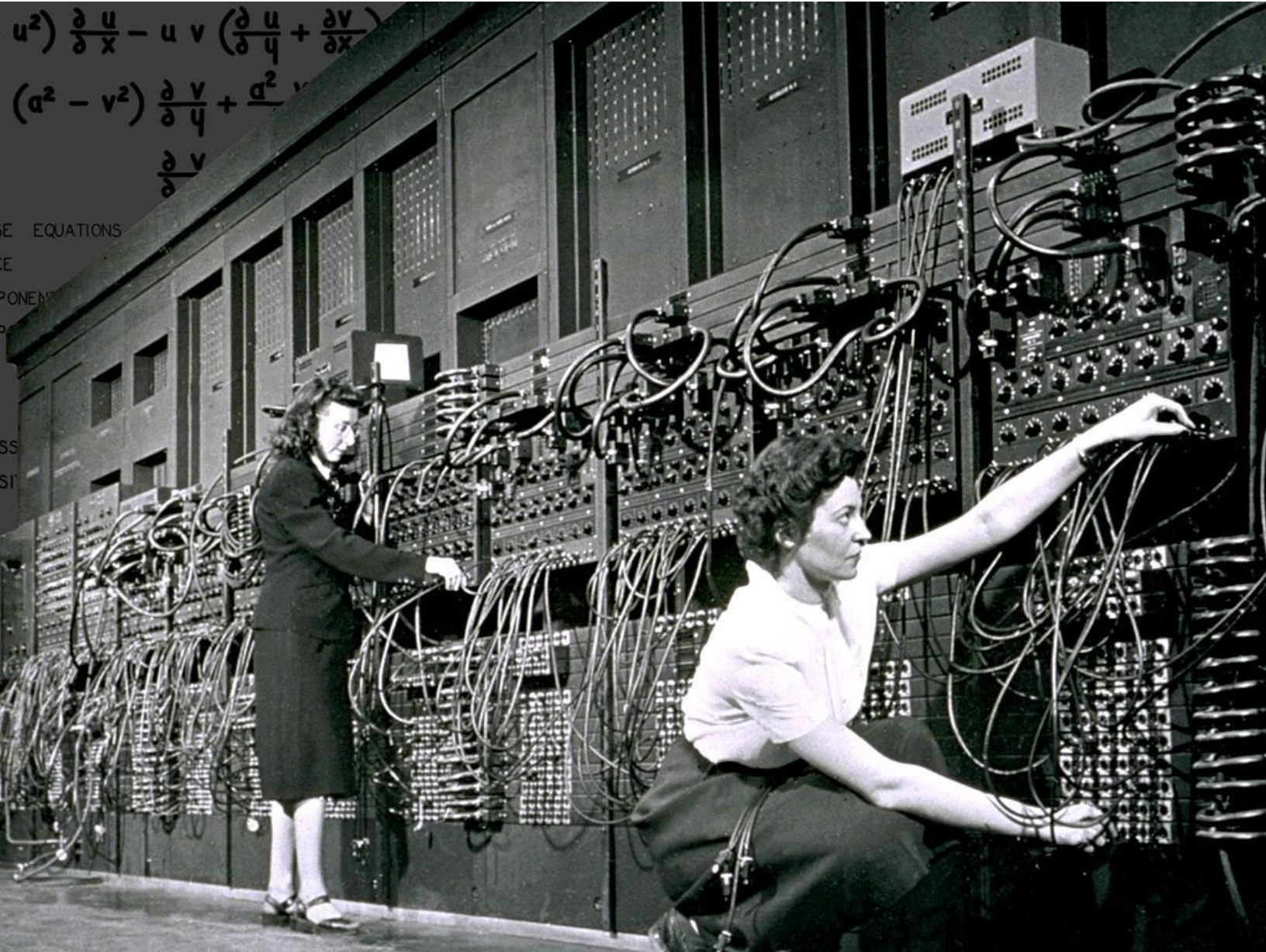
https://www.creativesafetysupply.com/caution-heavy-team-lift-3-x-5-label-roll/

Housekeeping

HW1 and Lab 1 are live now

Labs are over two weeks, 1 deliverable/week!

HW 1, Lab 1A due 9/14 at Midnight Eastern


$u^2)\frac{\partial u}{\partial x} - u\,v\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right.$
$(a^2 - v^2)\frac{\partial v}{\partial y} + \frac{a^2}{}$
$\frac{\partial v}{\partial}$
SE EQUATIONS
CE
PONEN
SS
SI

Computer Architecture

Information Age

Leverage and Transform Data

Data Pervasiveness

Huge Space and Opportunity

Responsibility

Amdahl's Law

Boring Version

Definition [edit]

Amdahl's law can be formulated in the following way:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

where

- S_{latency} is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

Furthermore,

$$\begin{cases} S_{\text{latency}}(s) \leq \frac{1}{1-p} \\ \lim\limits_{s \to \infty} S_{\text{latency}}(s) = \frac{1}{1-p}. \end{cases}$$

shows that the theoretical speedup of the execution of the whole task increases with the improvement of the resources of the system and that regardless of the magnitude of the improvement, the theoretical speedup is always limited by the part of the task that cannot benefit from the improvement.

Amdahl's law applies only to the cases where the problem size is fixed. In practice, as more computing resources become available, they tend to get used on larger problems (larger datasets), and the time spent in the parallelizable part often grows much faster than the inherently serial work. In this case, Gustafson's law gives a less pessimistic and more realistic assessment of the parallel performance.[2]

Intuitive Takeaway

Two independent parts **A** **B**

Original process

Make **B** 5x faster

Make **A** 2x faster

https://en.wikipedia.org/wiki/Amdahl's_law

Digital System Abstraction

6 orders of magnitude!

focus designer attention here

Transistors	10^9	Billions
Gates	10^8	100s of Millions
Microarchitectural Blocks	10^2	100s
Chips	10^1	10s
Boards	10^1	~10
System	10^0	1

Key Concepts

Cover and Assess

- Digital Logic
- Computer Organization
- Finite State Machines
- Design Evaluation

Introduce

- CMOS Implementation
- Computer Architecture
- Memory Hierarchy

Succeeding in CompArch

Meatspace, on paper, is your friend.

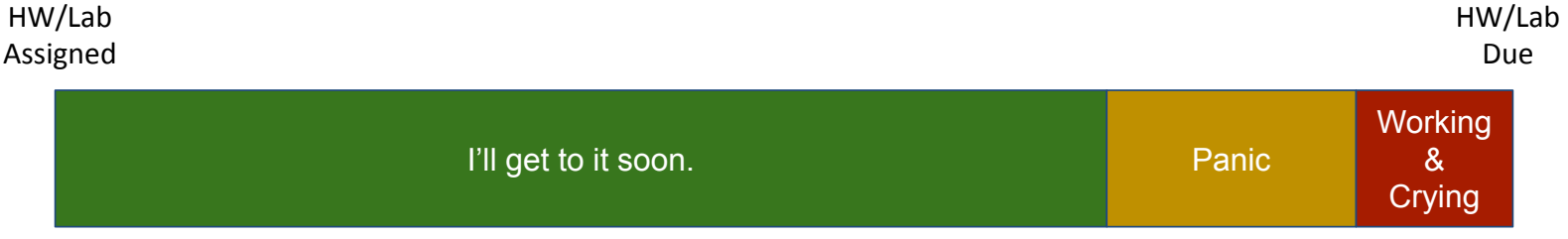
Divide and Conquer -- Abstraction

Hardware Mindset -- HW != SW

Careful Final Project Scope

Feedback, feedback, feedback.

Failing in CompArch



Acknowledgements

Ben Hill (Olin ‘07) course content

Eric VanWyk (Olin ‘07) course content

Mark L. Chang lecture notes for Computer Architecture (Olin ENGR3410)

Patterson & Hennessy: Book & Lecture Notes

Patterson’s 1997 course notes (U.C. Berkeley CS 152, 1997)

Tom Fountain 2000 course notes (Stanford EE182)

Michael Wahl 2000 lecture notes (U. of Siegen CS 3339)

Ben Dugan 2001 lecture notes (UW-CSE 378)

Professor Scott Hauck lecture notes (UW EE 471)

Mark L. Chang lecture notes for Digital Logic (NWU B01)

Today

- Explain how basic logic gates work
- Show the translations between
  - Boolean Algebra
  - Gates
  - Truth Tables
- By the end of the class this stuff should be comfortable if not intuitive.
- We'll do this all again in HW1.

Digital Logic

Binary - Two States

True = 1 and False = 0

Many Names and Representations

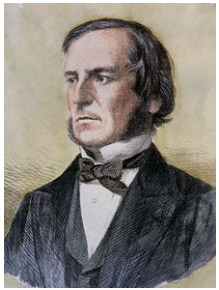
Boolean {Logic, Algebra}

Truth Table

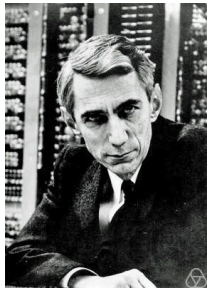
Karnaugh Maps

CMOS Logic, Logic Gates

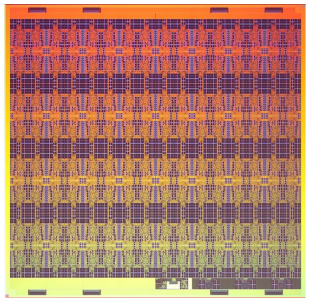
Brief History of Boolean Algebra



George Boole
1s and 0s
1840s



Claude Shannon
Digital Circuits
(and Information Theory)
1930s



Today
Intel Loihi
2018

Technique for manipulating representation of a given circuit or logic to improve execution speed or resources consumed

https://en.wikichip.org/wiki/intel/loihi

Example: Car Electronics (cont.)

Seat Belt Light

light = NOT belt_fastened

Passenger Seatbelt Warning

light = (NOT belt_fastened) AND seat_occupied

Example: Car Electronics

Door Ajar Light

light = driver_open OR passenger_open OR trunk_open

High-Beam Indicator

light = headlamps_on AND high_beam_on

The Digital Abstraction

https://e2e.ti.com/support/microcontrollers/c2000/f/171/t/767263?TMS320F28069M-SPI-is-inserting-an-extra-bit-in-front-of-read-transfer

Logic Representations (NOT)

not X

¬X

~X

$\overline{X}$

X	Out
0	1
1	0

Logic Representations (AND)

X and Y
X&Y
XY

X	Y	Out
0	0	0
0	1	0
1	0	0
1	1	1

Logic Representations (OR)

X or Y

X | Y

X+Y

X	Y	Out
0	0	0
0	1	1
1	0	1
1	1	1

Logic Representations (XOR)

X xor Y

X^Y

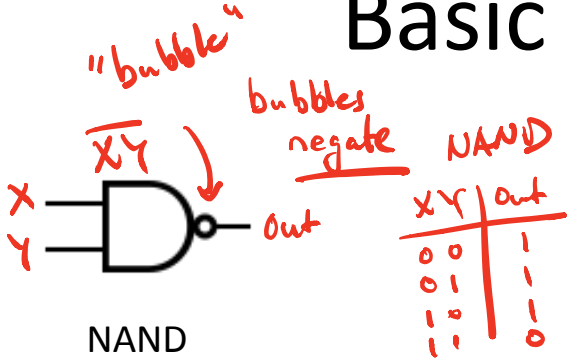
$X \oplus Y$

$X\overline{Y}+\overline{X}Y$

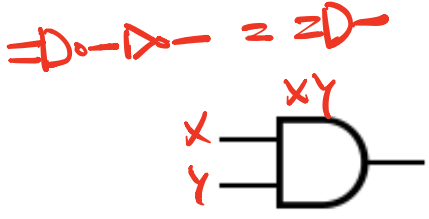
X	Y	Out
0	0	0
0	1	1
1	0	1
1	1	0

XOR is a test for odd #'s of 1's

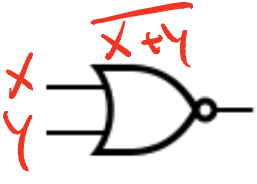
Basic Logic Gates



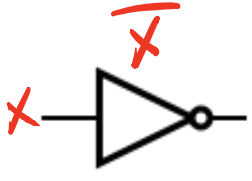
NAND



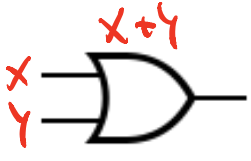
AND



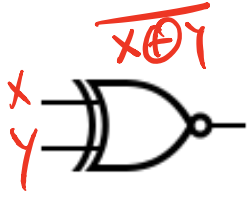
NOR



NOT

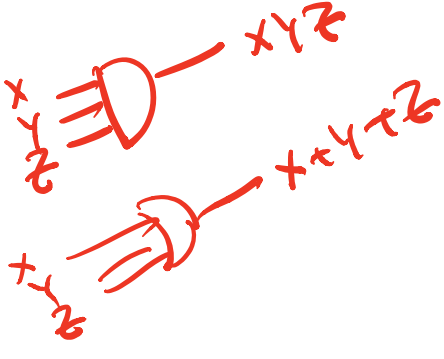


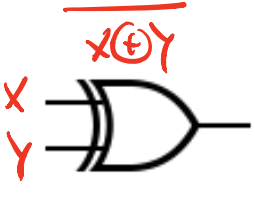
OR



XNOR

n-Input Gates possible





XOR

Boolean Algebra

Identity	$X + 0 =$ X	$X \cdot 1 =$ X
Annulment	$X + 1 =$ 1 (weird but true: 1 + 1 = 1)	$X \cdot 0 =$ 0
Idempotent	$X + X =$ X	$X \cdot X =$ X
Complement	$X + \overline{X} =$ 1	$X \cdot \overline{X} =$ 0
Involution	$\overline{\overline{X}} =$ X	

Boolean Algebra (cont.)

Commutative Law

$X + Y = Y + X$ $XY = YX$

Associative Law

$X+(Y+Z) = (X+Y)+Z$ $X(YZ)=(XY)Z$

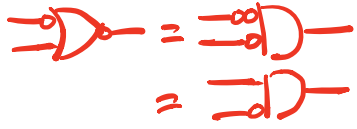
Distributive Law

$X(Y+Z) = XY + XZ$ $X+YZ = (X+Y)(X+Z)$

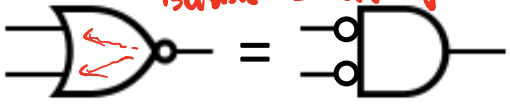
De Morgan's Law

Convert AND ↔ OR

$$\overline{X+Y} = \overline{X}\,\overline{Y}$$

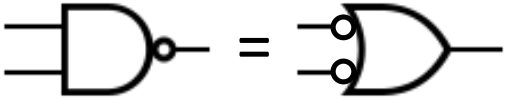


"Bubble Pushing"
"Bubble Shuffling"



change the gate, negate in/out as appropriate

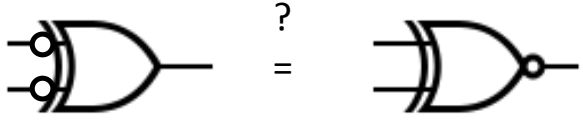
$$\overline{X\,Y} = \overline{X}+\overline{Y}$$



X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

X	Y	$\overline{X}$	$\overline{Y}$	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

XOR Bubble Shuffling?



XOR XNOR

≠

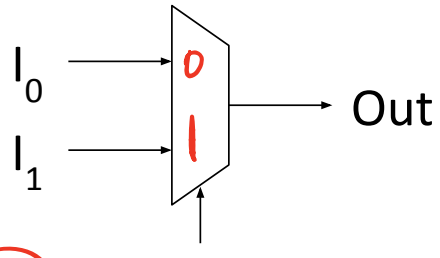
$\overline{X}$	$\overline{Y}$	Out
1	1	0
1	0	1
0	1	1
0	0	0

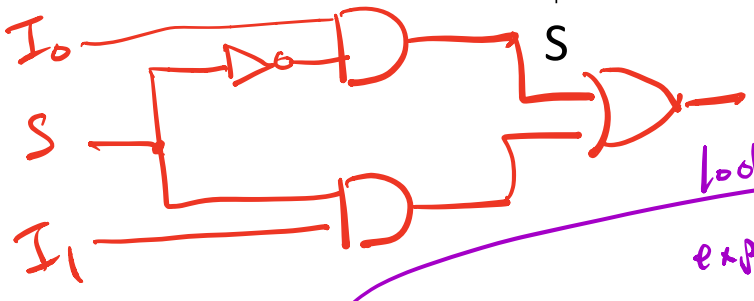
X	Y	Out
0	0	1
0	1	0
1	0	0
1	1	1

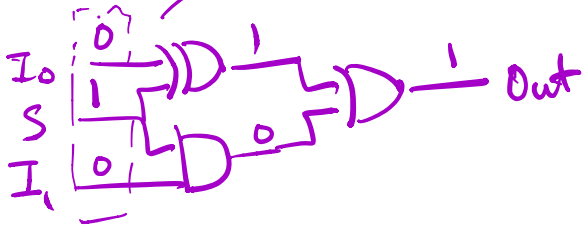
XOR doesn't obey De Morgan's

Implementing Choice

Multiplexer (MUX)







why not XOR?

look at this case

expected 0 on output, got 1

be careful of over-optimization

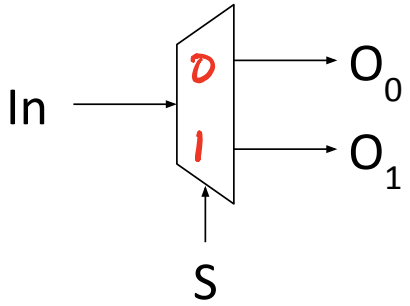
S	I_0	I_1	Out
0	0	0	$I_0 = 0$
0	0	1	$I_0 = 0$
0	1	0	$I_0 = 1$
0	1	1	$I_0 = 1$
1	0	0	$I_1 = 0$
1	0	1	$I_1 = 1$
1	1	0	$I_1 = 0$
1	1	1	$I_1 = 1$

looks like

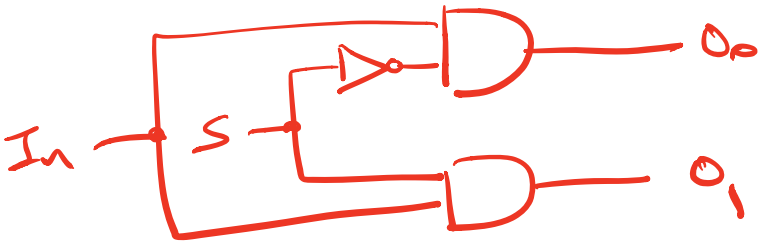
-plex from Latin: “plicare” or to fold, interweave

Implementing Choice

Demultiplexer (DEMUX)



S	In	O_0	O_1
0	0	$I_n=0$	0
0	1	$I_n=1$	0
1	0	0	$I_n=0$
1	1	0	$I_n=1$



-plex from Latin: “plicare” or to fold, interweave

Width

Width of a **Bus** = # of bits in the signal

Wire = Bus of width 1