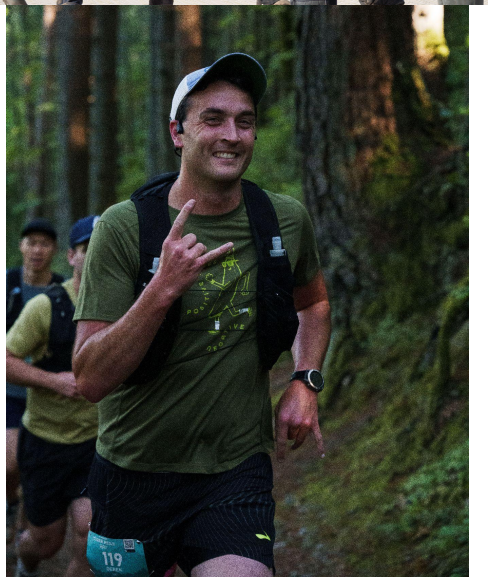
Section 1

Boolean Algebra Review

Meet your TAs!

Derek de Leuw (he/him)

- 4th year Computer Engineering student
- Took 369 in Autumn 2024
- 2nd time TAing this class, have TAed 333 previously as well



Meet your TAs!

Isabel Froelich (she/her)

- 4th year Computer Engineering student
- Took 369 in Autumn 2024
- 2nd time TAing this class, have TAed 390Z and 390T previously
- This is my roommate's cat, Willow, playing a very intense game of bananagrams

Meet each other!

In groups of 3-5, introduce yourselves:

- Name
- Year
- Favorite class so far
- Why you are taking this class?
- How would you say DE1-SoC?

Administrivia

- **Pre-course survey:** Due today (4/3, link).
- **Lab kits:** If you didn't pick up a kit, come to OH ASAP so you can get started on the lab. There are always ***LOTS*** of hardware and software issues to figure out.
- **Lab 1:** Report due next Wednesday (4/8) @ 2:30 pm, demo during your assigned slot of the same week your lab report is due.
  - Lab demo slots will be assigned on Monday (4/6) via Canvas.

What to expect from section

- **When:** Every Friday at 1:30pm in CSE2 271
  - Recordings will be made available afterward on Panopto
  - Different TAs each week!
- **What:** SystemVerilog is a tricky language and we don't have much time to talk about it during lectures (only 80 min/week), so we are introducing optional sections focused on it.
- Please come and share your feedback! Let us know if there is anything you would like to focus on!

369 Workflow

More on this in Lecture 2

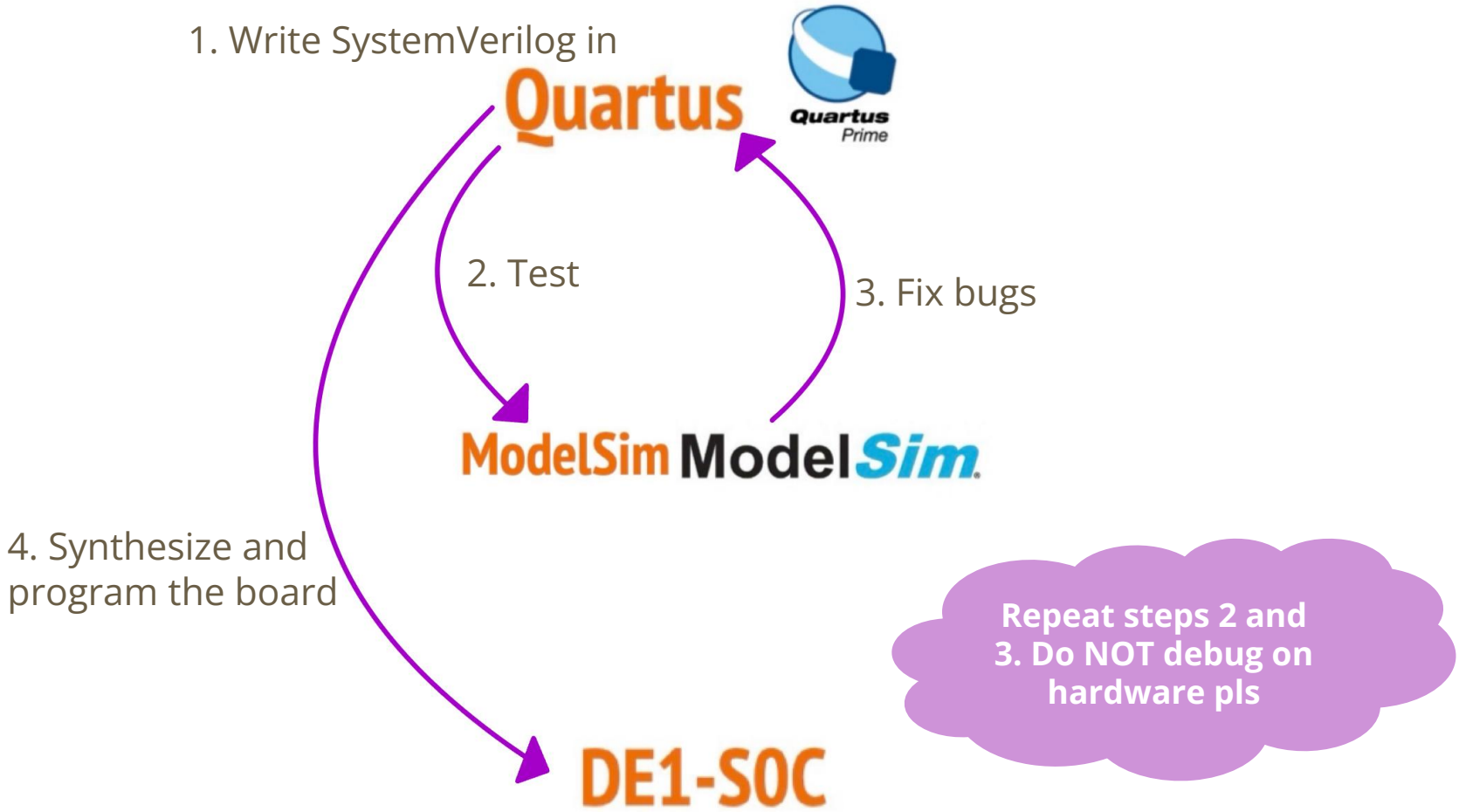
Quartus

- The Intel Quartus Software is a tool for designing, synthesizing, and programming FPGAs.
- With it, you are able to (1) **write SystemVerilog code**, (2) **synthesize it**, and (3) **program your DE1-SoC board**.
- More information: Quartus Tutorial

ModelSim

ModelSim

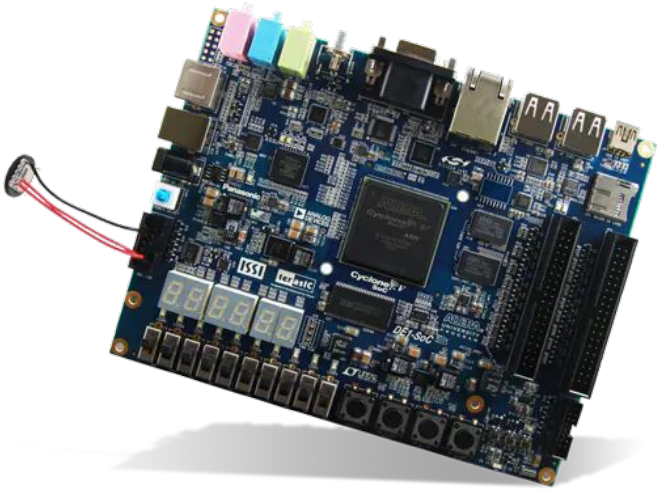
- ModelSim is a simulation and debugging tool for a variety of hardware description languages, including SystemVerilog.
- With it, you are able to run simulations to **verify your code's logical behavior** without endangering your hardware.
  - This is your main tool for **debugging**!
- More information: ModelSim Usage Guide


1. Write SystemVerilog in
Quartus
Quartus Prime
2. Test
3. Fix bugs
ModelSim ModelSim
4. Synthesize and program the board
DE1-SOC
Repeat steps 2 and 3. Do NOT debug on hardware pls

DE1-SoC Overview

Some details on how FPGAs work will be covered Lecture 9

- The DE1-SoC is a development kit built around an FPGA.
  - An FPGA is a large array of logical elements with reprogrammable connections. This means we can use it to build many different kinds of hardware all on the same chip!
  - The kit also contains other functionalities that we will use in our labs.
- More information: FPGA overview



Logic Review

Boolean Algebra

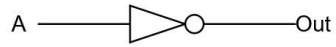
- Represent inputs and outputs as variables
  - Each variable can only take on the value of 0 or 1
- Overbar is NOT: “logical complement”
  - If A is 0, then $\bar{A}$ is 1 and vice-versa
- Plus (+) is 2-input OR: “logical sum”
- Product (·) is 2-input AND: “logical product”
- All other gates and logical expressions can be built from combinations of these
  - *e.g.*, A XOR B = $A \oplus B = \bar{A}B + B\bar{A}$

Truth Tables

- Table relating inputs to a combinational logic (CL) circuit to its output
  - Outputs *only* depend on current inputs
  - Use 0/1 instead of low/high voltage
  - Show output for *every* possible combination of inputs (“black box” approach)
  - For N inputs, 2^N rows required to map all outputs

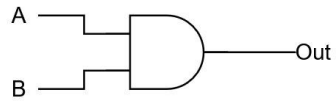
Logic Gates

NOT



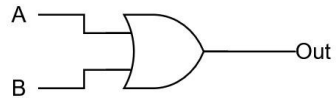
A	Out
0	1
1	0

AND



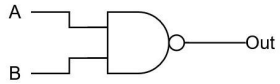
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR



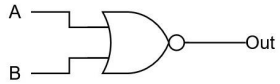
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

NAND



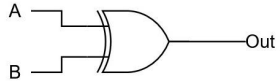
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table to Boolean Expression

- Read off of table
  - For 1, write variable name
  - For 0, write complement of variable
- *Sum of Products (SoP)*
  - Take rows with 1’s in output column, sum products of inputs
- *Product of Sums (PoS)*
  - Take rows with 0’s in output column, product the sum of the complements of the inputs

Basic Boolean Equivalences

- Identity: $X + 0 = X$ $X \cdot 1 = X$
- Domination: $X \cdot 0 = 0$ $X + 1 = 1$
- Idempotent: $X + X = X$ $X \cdot X = X$
- Negation: $\bar{X} + X = 1$ $\bar{X} \cdot X = 0$
- Double Negation: $\bar{\bar{X}} = X$

Basic Boolean Laws

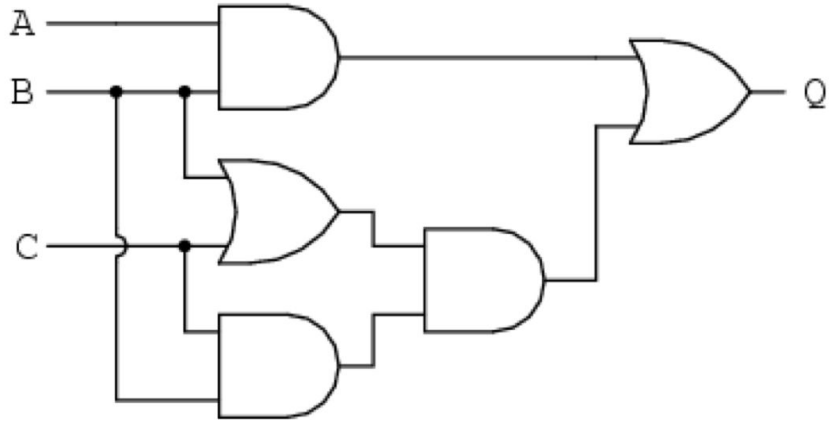
- Commutative: $X + Y = Y + X$ $X \cdot Y = Y \cdot X$
- Associative: $X + (Y + Z) = (X + Y) + Z$ $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$
- Distributive: $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ $X + YZ = (X + Y) \cdot (X + Z)$

Boolean Absorption

- $X + XY = X$
- $XY + X\bar{Y} = X$
- $X + \bar{X}Y = X + Y$
- $X(X + Y) = X$
- $(X + Y)(X + \bar{Y}) = X$
- $X(\bar{X}+Y) = XY$

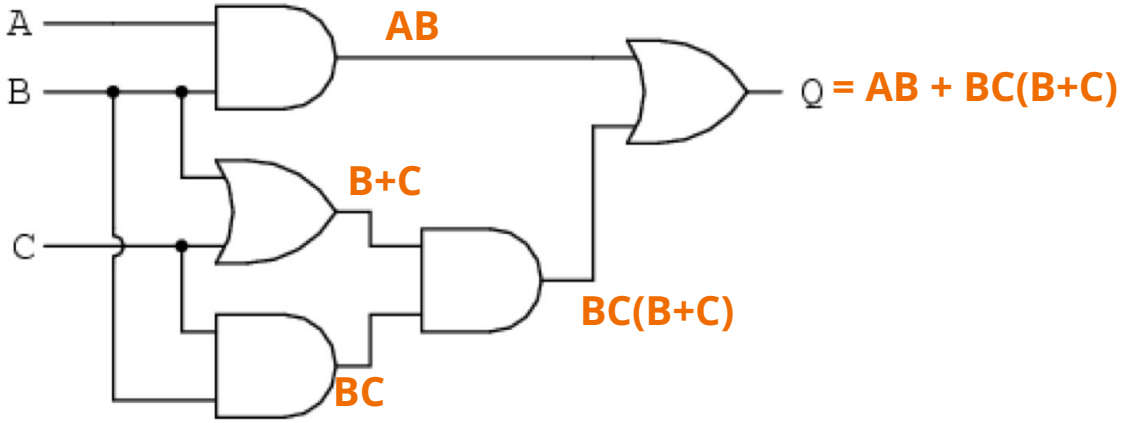
Exercise 1

- a) Write out the Boolean Algebra expression for Q for the following circuit.
- b) Simplify the expression for Q.



Exercise 1 (Solution)

- a) Write out the Boolean Algebra expression for Q for the following circuit.



Exercise 1 (Solution)

- b) Simplify the expression for Q.

AB + BC(B+C)
AB + BC + BC
AB + BC
B(A + C)

Not All Logic Gates Are Created Equally!

- Can recreate all other gates using only NAND or only NOR gates
  - Called “universal” gates
  - *e.g.* A NAND A = $\bar{A}$, B NOR B = $\bar{B}$
  - DeMorgan’s Law helps us here!

2-Input Gate Type	# of CMOS Transistors
NOT	2
AND	6
OR	6
NAND	4
NOR	4
XOR	8
XNOR	8

Logic Minimization

- Reduce complexity at gate level
  - Allows us to build smaller and faster hardware
  - Care about # of gates, # of literals (gate inputs), “depth” of gate levels, and types of logic gates

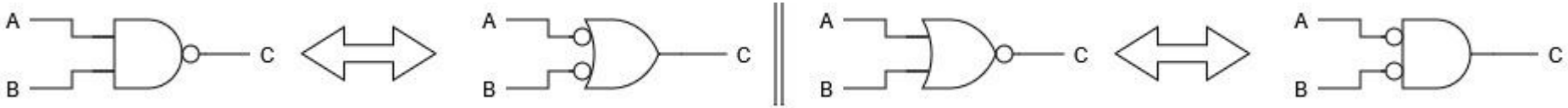
Logic Minimization

- Reduce complexity at gate level
  - Allows us to build smaller and faster hardware
  - Care about # of gates, # of literals (gate inputs), “depth” of gate levels, and types of logic gates
- Faster hardware?
  - Fewer inputs implies faster gates in some technologies
  - Fan-ins (# of gate inputs) are limited in some technologies
  - Fewer levels of gates implies reduced signal propagation delays
  - # of gates influences manufacturing costs
  - Simpler Boolean expressions means smaller transistor networks, which means smaller circuit delays, which gives us faster hardware

DeMorgan's Law

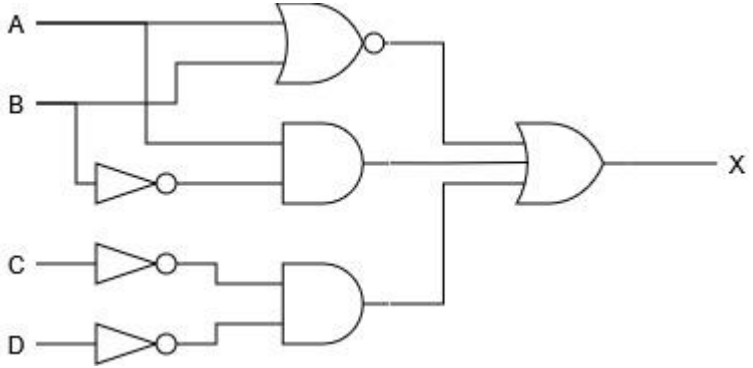
If you remember one thing from section, it should be this pretty pls!

- In Boolean Algebra, converts between AND-OR and OR-AND expressions
  - $\overline{X + Y} = \bar{X} \cdot \bar{Y}$
  - $\overline{X \cdot Y} = \bar{X} + \bar{Y}$
- At gate level, can convert from AND/OR to NAND/NOR gates
  - "Flip" all input/output bubbles and "switch" gate (Bubble Pushing)



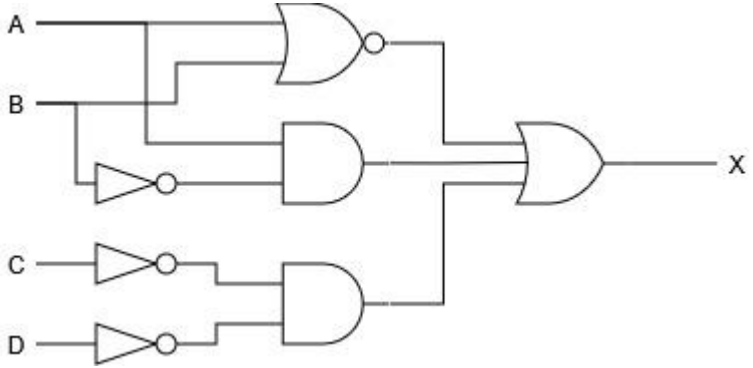
Exercise 2

- Find the simplified Boolean expression for the following diagram:



Exercise 2 (Solution)

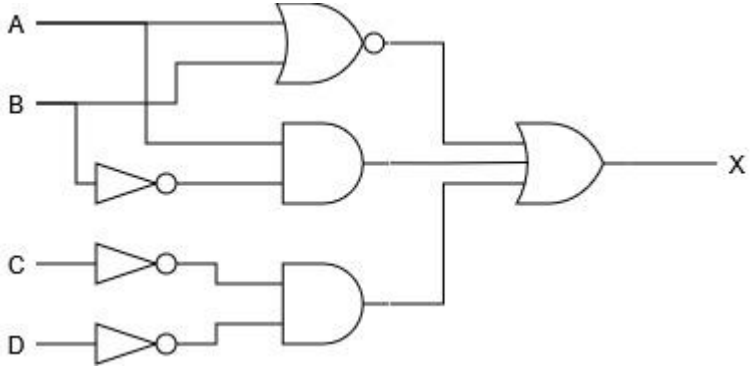
- Find the simplified Boolean expression for the following diagram:



$$X = \overline{A + B} + A\overline{B} + \overline{C}\,\overline{D}$$

Exercise 2 (Solution)

- Find the simplified Boolean expression for the following diagram:

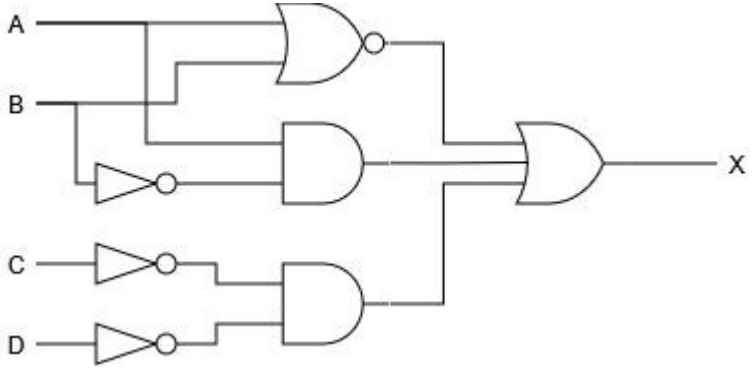


$$X = \overline{\overline{A} + B} + A\overline{B} + \overline{C}\,\overline{D}$$

$$X = \overline{A}B + A\overline{B} + \overline{C}\,\overline{D}$$

Exercise 2 (Solution)

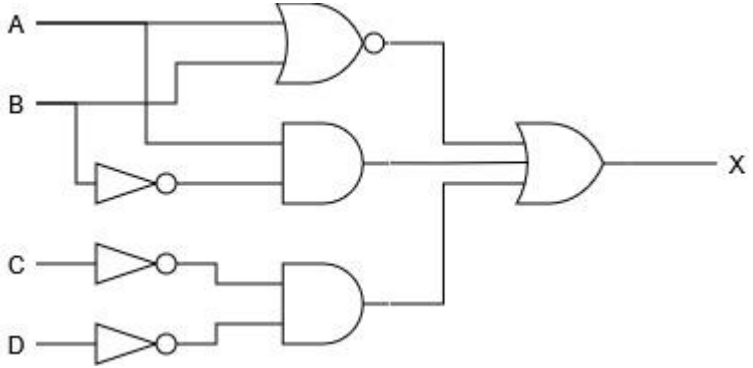
- Find the simplified Boolean expression for the following diagram:



$$X = \overline{\overline{A} + B} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{AB} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{B} + \overline{C}\,\overline{D}$$

Exercise 2 (Solution)

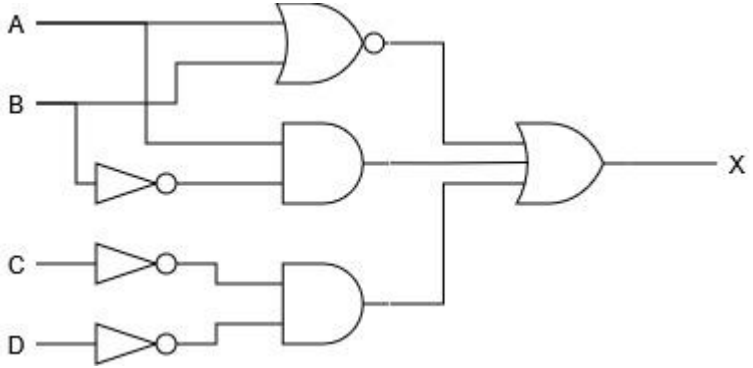
- Find the simplified Boolean expression for the following diagram:



$$X = \overline{\overline{A} + B} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{AB} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{B} + \overline{C + D}$$

Exercise 2 (Solution)

- Find the simplified Boolean expression for the following diagram:



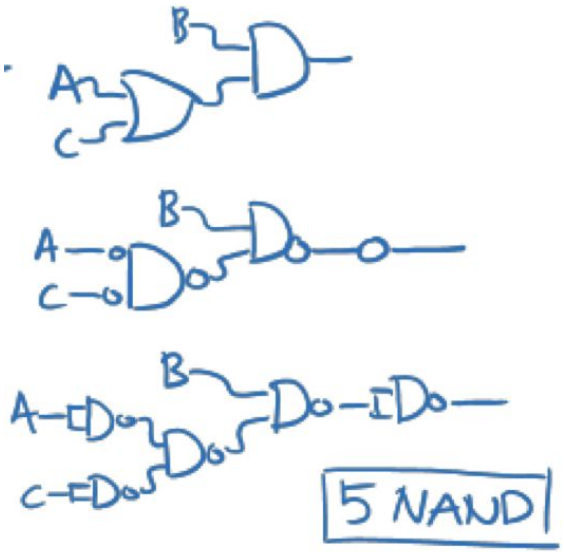
$$X = \overline{A + B} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{A}\,\overline{B} + A\overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{B} + \overline{C}\,\overline{D}$$
$$X = \overline{B} + \overline{C + D}$$
$$\mathbf{X = \overline{B(C + D)}}$$

Exercise 3

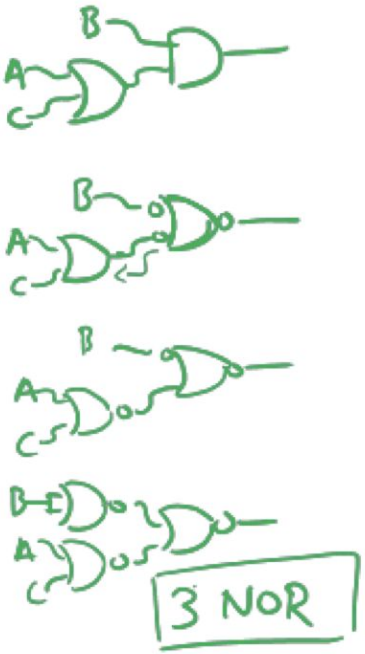
- Implement the Boolean expression B(A + C) with the fewest number of a single universal gate. What does your solution look like?
  - Universal gates are NAND and NOR

Exercise 3 (Solution)

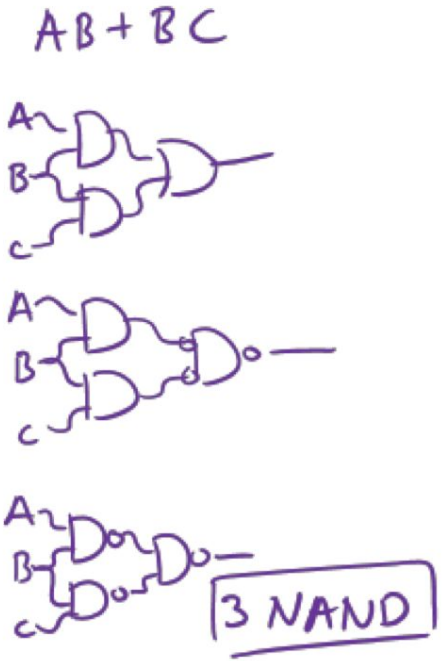
NAND



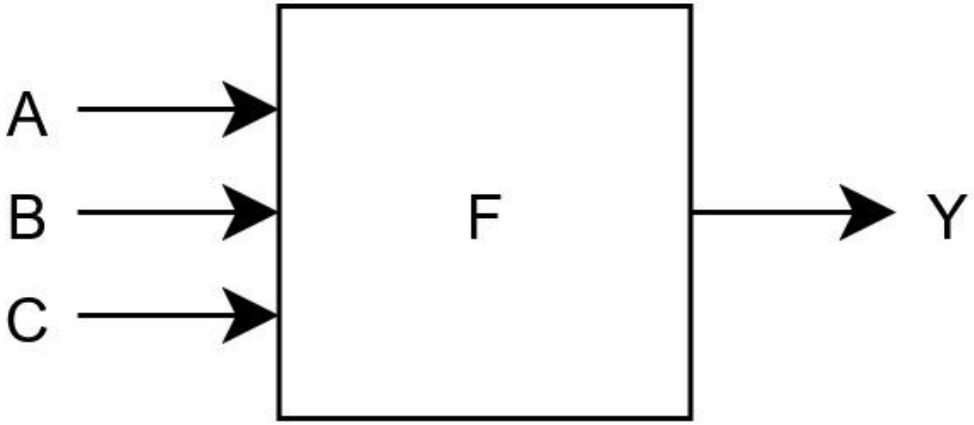
NOR



NAND (alternative)



CL General Form



A	B	C	Y
0	0	0	F(0,0,0)
0	0	1	F(0,0,1)
0	1	0	F(0,1,0)
0	1	1	F(0,1,1)
1	0	0	F(1,0,0)
1	0	1	F(1,0,1)
1	1	0	F(1,1,0)
1	1	1	F(1,1,1)

For *N* inputs, function F maps each row to 0 or 1, so **2^(2^N) possible functions.**

Installation Process

Installing Quartus and ModelSim

- If you wish to skip installation altogether, you can use the lab computers in CSE 003.
- Follow the installation tutorial on the website: link
  - This class only supports the **Windows** version of Quartus and ModelSim.
  - For **MacOS**, you can use a Windows virtual machine (described in the tutorial).
- Installation tips:
  - All 3 downloaded files need to be in the same directory so that ModelSim and Cyclone V options are available during Quartus installation.
  - Try not to miss the USB Blaster II driver installation option at the end.

Workflow Demo

Shortened version of the Quartus Tutorial

File Organization

- Unzip `Lab1_files_Q17.zip` to get started.
  - Download from the Lab 1&2 specs.
  - Can be unzipped again to start any new project or you can copy an existing project directory.
- Create subdirectories for each lab within a class directory (*e.g.*, `CSE369/lab1`).
  - All project SystemVerilog files should be placed in this directory and added to Quartus project.
- Every Verilog module should have a test bench in a *separate* file.
  - Suggested naming scheme: `new_module.sv` and `new_module_tb.sv` for test bench.

Programming Workflow (Quartus)

- Open the Quartus project via the `.qpf` file in the project directory (double-click or find using File → Open Project...).
- In the Project Navigator "Files" tab, need to right-click and select "Set as Top-Level Entity" on the proper file/module.
  - The top-level entity should NOT be a test bench.
- When done coding a module, save the file and then run the Analysis and Synthesis tool: 
  - Quartus' interface for compilation warnings and errors is better than ModelSim's.
  - Use this tool to fix errors with syntax and signal connections before simulation.

Simulation Workflow (ModelSim)

- Double-click `Launch_ModelSim.bat` in the project directory.
- In a text editor, modify `runlab.do` for your project:
  - Add files to compile (modules + test benches).
  - Change which test bench you wish to simulate.
  - Change the waveform script file (`*_wave.do`) – this won't exist at first.
- Execute `do runlab.do` in the *`Transcript`* pane.
  - Use waveforms to verify/debug logical behavior of your module(s).
- Update waveform script file as desired.
  - Click on different modules in the *`sim`* pane to access different signals.
  - Drag signals from the *`Objects`* pane into the *`Wave`* pane.
  - With the *`Wave`* pane selected, `Ctrl+S` to overwrite your waveform script file.

Hardware Workflow (Quartus)

- Make sure the board is off before connecting to the computer's USB, then power the board on (red push button).
- In Quartus, ensure that your top-level module is set as the project's top-level module then use the Compilation tool: 
  - This typically takes a while to run (2–10 minutes).
- Use File → Open... to open the Programmer interface via `ProgramTheDE1_SoC.cdf`.
  - Need to change the file type to "Programming Files":

- Assuming no issues, click "Start" to program your DE1-SoC!