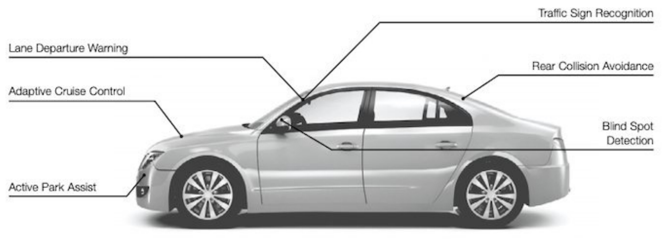
Why the Industry is Demanding FPGAs for Advanced Driver-Assistance Systems (ADAS)

“Advanced driver-assistance systems (ADAS) are quickly being integrated into almost all new automobiles. These systems often introduce automakers and Tier 1s with unique computing needs that the standard CPU or GPU may not be well suited for…

First and foremost, FPGAs offer high customizability and flexibility… FPGA allows the designers the ability to customize their solutions specifically to their needs while differentiating from the competition. FPGAs allow for scalability… engineers can easily build on the FPGA design from previous generations without having to go through the hassle of spinning a new ASIC.

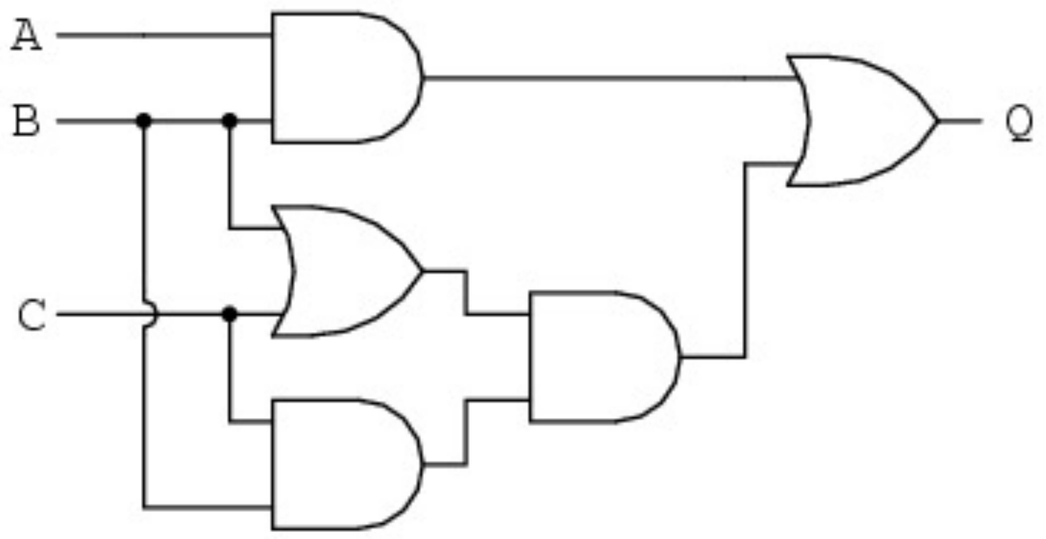
Finally, quick time to market makes FPGAs a desirable solution for car manufacturers…

https://www.allaboutcircuits.com/news/why-the-industry-is-demanding-fpgas-for-adas/



Practice Question:

- What is the MOST simplified Boolean Algebra expression for the following circuit?



Practice Question:

- Implement the Boolean expression $\mathrm{B(A + C)}$ with the fewest number of a single universal gate. What does your solution look like?

Verilog

- Programming language for describing hardware
  - Simulate behavior before (wasting time) implementing
  - Find bugs early
- Similar to C/C++/Java
  - VHDL similar to ADA
- Modern version is “SystemVerilog”
  - Superset of previous; cleaner and more efficient

Brainstorm: Describing Hardware

- How might Verilog be different than software programming languages?

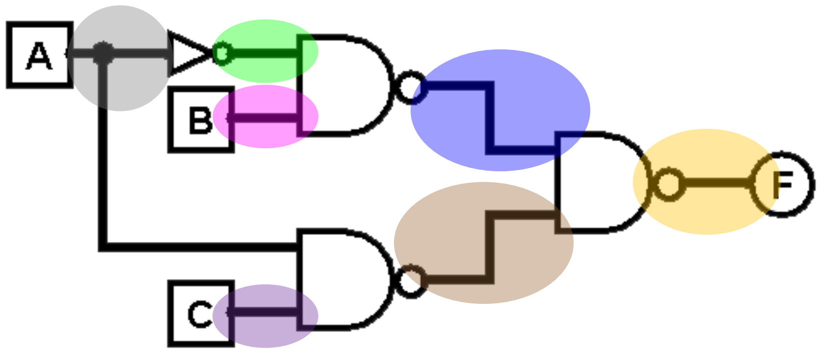
- Starter questions
  - What are the language primitives?
  - What does a “variable” mean in hardware?
  - Initialization?
  - “Program” execution?

Verilog: Hardware Descriptive Language

- Although it looks like code:

```
module myModule (F, A, B, C);
   output logic F;
   input  logic A, B, C;
   logic AN, AB, AC;

   nand gate1(AB,AN, B);
   nand gate2(AC, A, C);
   nand gate3( F,AB,AC);
   not   not1(AN, A);
endmodule
```

- Keep the hardware in mind:



Verilog Primitives

- Nets (`wire`): transmit value of connected source
  - Problematic if connected to two different voltage sources
  - Can connect to many places (always possible to "split" wire)
- Variables (`reg`): variable voltage sources
  - Can "drive" by assigning arbitrary values at any given time
  - SystemVerilog: variable `logic` can be used as a net, too

- Logic Values
  - 0 = zero, low, FALSE
  - 1 = one, high, TRUE
  - X = unknown, uninitialized, contention (conflict)
  - Z = floating (disconnected), high impedance

Verilog Primitives

- Gates:

Gate	Verilog Syntax
NOT a	`~a`
a AND b	`a & b`
a OR b	`a | b`
a NAND b	`~(a & b)`
a NOR b	`~(a | b)`
a XOR b	`a ^ b`
a XNOR b	`~(a ^ b)`

- Modules: "functions" in Verilog that define *blocks*
  - Input: Signals passed from outside to inside of block
  - Output: Signals passed from inside to outside of block

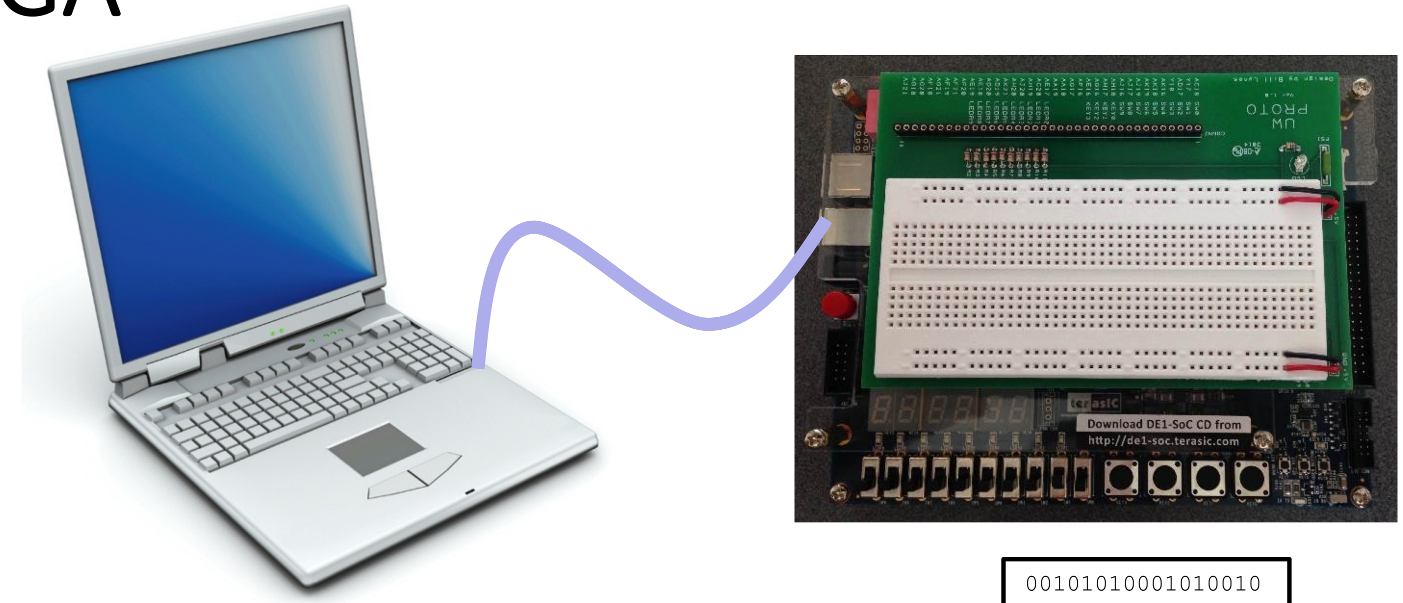
Verilog Execution

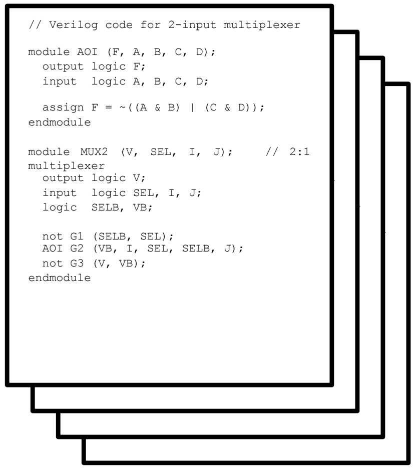
- Physical wires transmit voltages (electrons) near-instantaneously
  - Wires by themselves have no notion of sequential execution

- Gates and modules are constantly performing computations
  - Can be hard to keep track of!

- In pure hardware, there is no notion of initialization
  - A wire that is not driven by a voltage will naturally pick up a voltage from the environment

Using an FPGA





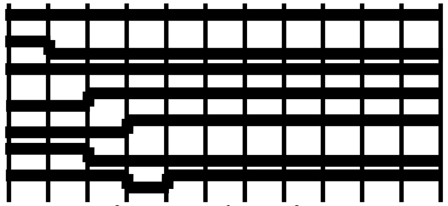
```
// Verilog code for 2-input multiplexer

module AOI (F, A, B, C, D);
  output logic F;
  input  logic A, B, C, D;

  assign F = ~((A & B) | (C & D));
endmodule

module MUX2 (V, SEL, I, J);    // 2:1
multiplexer
  output logic V;
  input  logic SEL, I, J;
  logic  SELB, VB;

  not G1 (SELB, SEL);
  AOI G2 (VB, I, SEL, SELB, J);
  not G3 (V, VB);
endmodule
```

Verilog

FPGA
CAD
Tools

```
00101010001010010
10010010010011000
10101000101011000
10101001010010101
00010110001001010
10101001111001001
01000010101001010
10010010000101010
10100101010010100
01010110101001010
01010010100101001
```

Bitstream



Simulation