HQ U.S. Air Force Academy

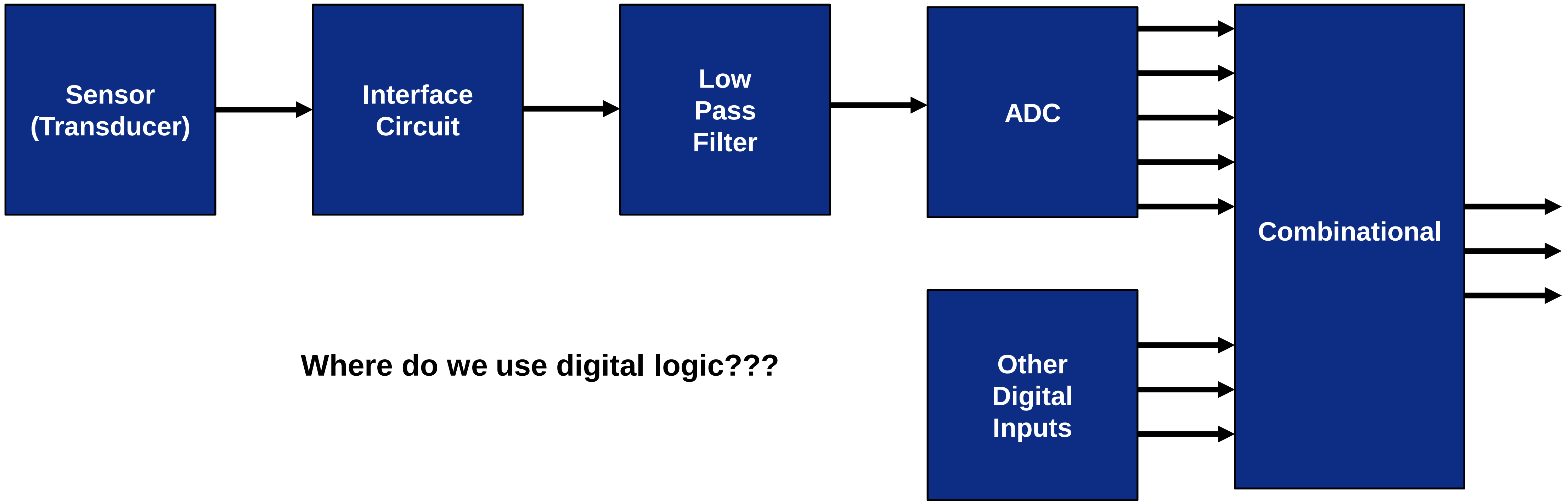
UNITED STATES
AIR FORCE
ACADEMY

ECE215
Principles of Air Force Electronic Systems
Objective 2.7: Digital Logic

Objective 2.7

I can relate digital inputs to digital outputs using tools such as fundamental digital logic gates (AND, OR, NOT), truth tables, and Sum of Products (SOP) and Product of Sums (POS) Boolean expressions.

Digital Logic

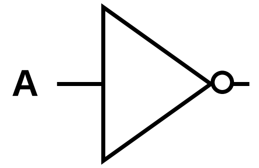


Where do we use digital logic???

Using digital logic, we can relate to through logic, where the output is either TRUE (1) or FALSE (0)

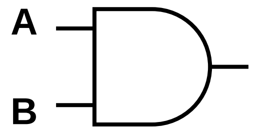
Fundamental Logic Gates

1. NOT gate



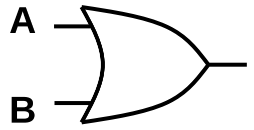
A	Y
0	
1	

2. AND gate



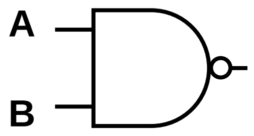
A	B	Y
0	0	
0	1	
1	0	
1	1	

3. OR gate



A	B	Y
0	0	
0	1	
1	0	
1	1	

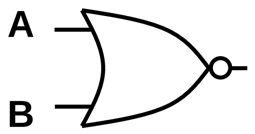
4. NAND gate



A	B	Y
0	0	
0	1	
1	0	
1	1	

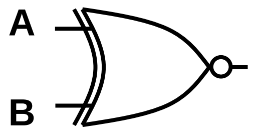
Fundamental Logic Gates

5. NOR gate



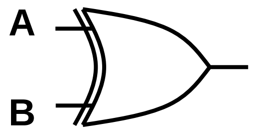
A	B	Y
0	0	
0	1	
1	0	
1	1	

7. XNOR gate



A	B	Y
0	0	
0	1	
1	0	
1	1	

6. XOR gate



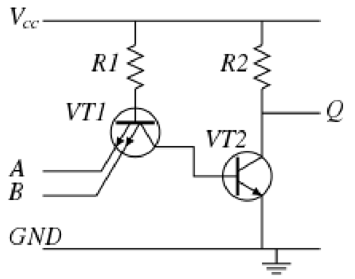
A	B	Y
0	0	
0	1	
1	0	
1	1	

All logic operations can be synthesized by using only , , and gates

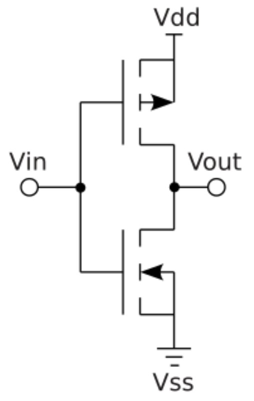
Logic gate fabrication

Digital logic gates are fabricated using two transistor technology types

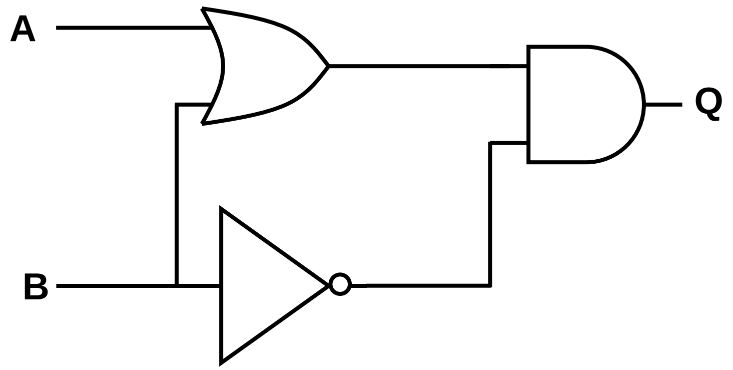
- Transistor-transistor-logic (TTL)
  - Made using bipolar junction transistors (BJTs)
  - Refered to as “7400 series chips”



- Complementary metal oxide semiconductor (CMOS)
  - Made using field effect transistors (FETs)
  - Refered to as “4000 series chips”



Derive a “Truth Table” from a Digital Logic Diagram



A	B	Q
0	0	
0	1	
1	0	
1	1	

We can use the distribute property to simply the expression for Q:

UNITED STATES
AIR FORCE
ACADEMY

Example 1: Toxic Waste Incinerator

Suppose there is a toxic waste incinerator where it is only safe to inject waste if a flame is present. The system has three redundant flame sensors to ensure waste is not inadvertently injected. To minimize susceptibility to sensor failure, we want the “flame present” condition indicated by <u>at least</u> 2 of the 3 sensors. Design the logic circuitry.

Step 1: Construct the required truth table

A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Step 2: Use the Sum of Products to find the circuitry that satisfies the truth table logic

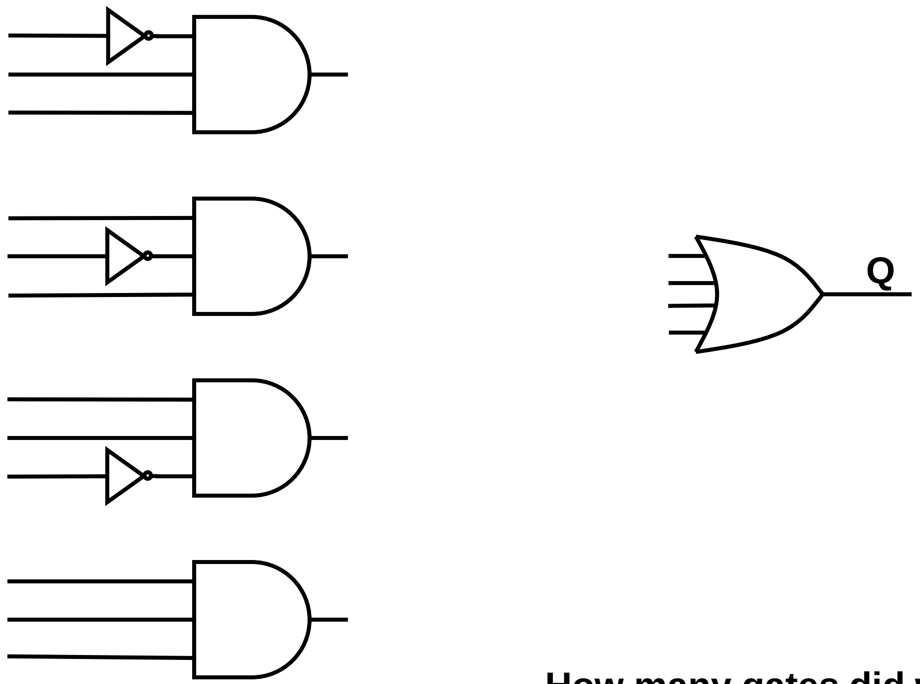
- A sum of products (SOP) is a set of Boolean terms summed together, each term being a product of Boolean variables
- For each truth table row that is TRUE (1), write the corresponding Boolean product term that would equal 1

$$Q =$$

Example 1: Toxic Waste Incinerator

Step 3: Use NOT, AND, and OR gates to synthesize Q

$Q =$



How many gates did we use?

Question: Can this expression be simplified to minimize the required number of gates and signal lines?

- We can use Boolean algebra reduction or Karnaugh-Maps (learn more about this in ECE 281!)

Simplified form:

Example 2: Flame Sensor Fault Detector

Design the logic circuitry needed to detect a sensor disagreement amongst three sensors.

Step 1: Construct the required truth table

A	B	C	Q	Y
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

- We could do a SOP form but it would require 6 terms and algebraic reduction
- Alternatively, we can do a Product of Sums!
- A Product of Sums (POS) is a set of that are together
- For each truth table row that is FALSE (0), write a Boolean sum **that equals 0**

Step 2: Write out the POS form

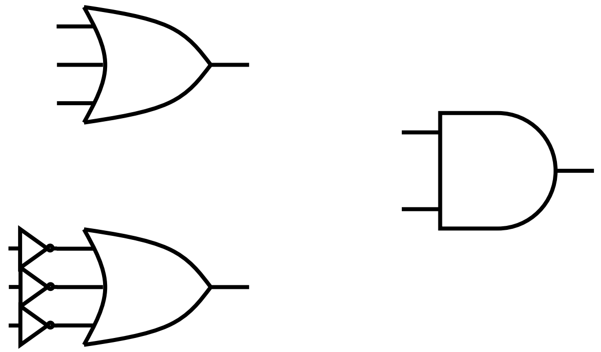
$$Y =$$

Example 2: Flame Sensor Fault Detector

Design the logic circuitry needed to detect a sensor disagreement amongst three sensors.

$$Y =$$

Step 3: Use NOT, AND, and OR gates to synthesize Y



Exercise: Greater than or equal to

Design the logic circuitry needed to detect if an input 3-bit binary value is greater than or equal to 5.

Step 1: Construct the required truth table

Dec	A	B	C	Q
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

Step 2: Write out SOP or POS form

Step 3: Implement with AND, OR and NOT gates

Sequential Logic

- Combinational logic circuits only depend on the ***current input***
- Sequential logic circuits depend on the ***current input*** and the ***previous state*** of the circuit as stored in memory
  - For example, think of a traffic light – it only can go to yellow after being green

- **Flip flop:** clocked electronic circuits used to store binary data (types include RS, JK, D, and T)
  - Good overview here

- When the sequential logic circuit has a finite number of possible and unique states, it is called a **Finite State Machine** (next time!)