



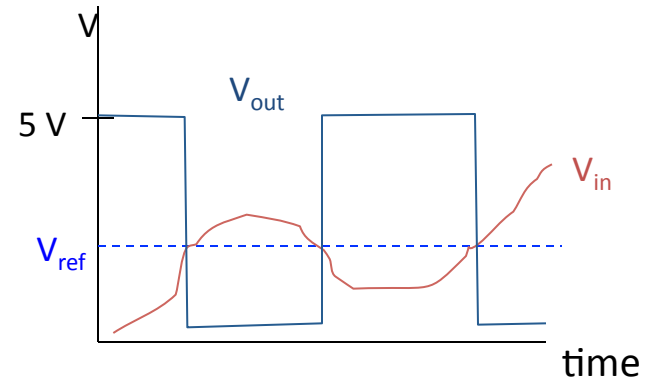
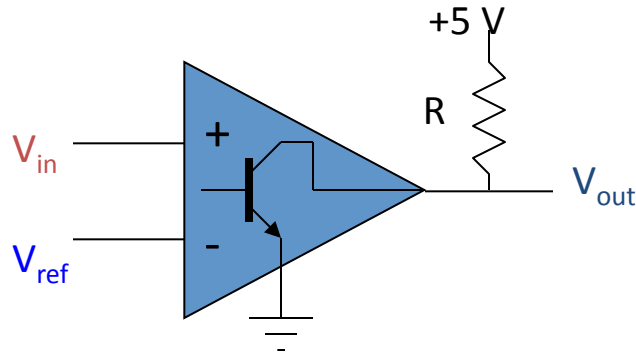
# Comparators, FETS, & Logic

Other Useful Devices

# Comparators

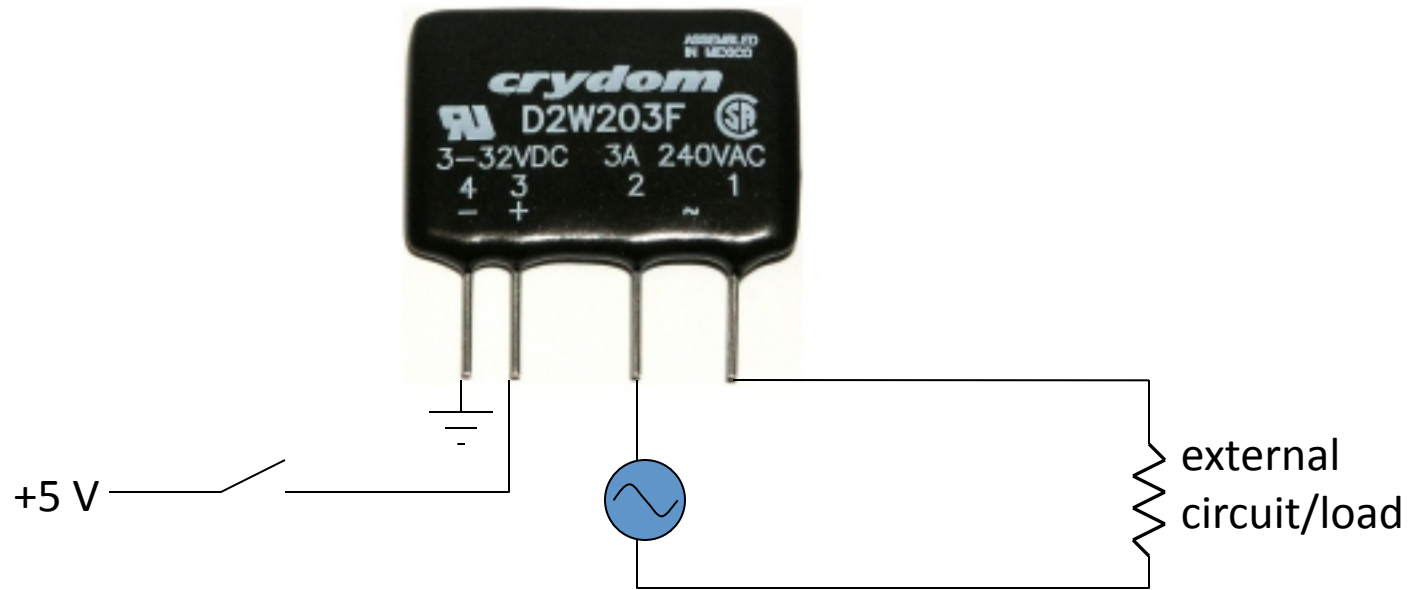
- It is very often useful to generate a strong electrical signal associated with some **event**
- If we frame the “event” in terms of a voltage threshold, then we use a **comparator** to tell us when the **threshold** is exceeded
  - could be at a certain temperature, light level, etc.: **anything that can be turned into a voltage**
- Could use an op-amp without feedback
  - set inverting input at threshold
  - feed test signal into non-inverting output
  - op-amp will rail (negative rail if test < reference; positive rail if test > reference)
- But op-amps have relatively slow “slew rate”
  - 15 V/ $\mu$ s means 2  $\mu$ s to go rail-to-rail if powered  $\pm 15$  V

# Enter the comparator



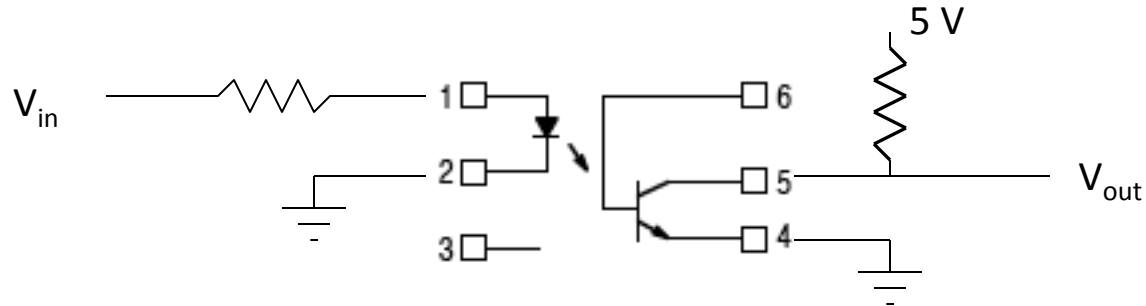
- When  $V_{in} < V_{ref}$ ,  $V_{out}$  is pulled high (through the pull-up resistor— usually  $1\text{ k}\Omega$  or more)
  - this arrangement is called “open collector” output: the output is basically the collector of an npn transistor: in saturation it will be pulled toward the emitter (ground), but if the transistor is not driven (no base current), the collector will float up to the pull-up voltage
- The output is a “digital” version of the signal
  - with **settable** low and high values (here ground and 5V)
- Comparators also good at turning a slow edge into a fast one
  - for better timing precision

# Relays



- Relays provide a way to switch on/off an AC line with a logic signal
- Simple: 5 volts in  $\rightarrow$  AC switch flipped on
- Often will phase to AC line so it turns on at zero-crossing, so-as not to jar electronics

# Opto-isolators



PIN 1. LED ANODE  
2. LED CATHODE  
3. N.C.  
4. EMITTER  
5. COLLECTOR  
6. BASE

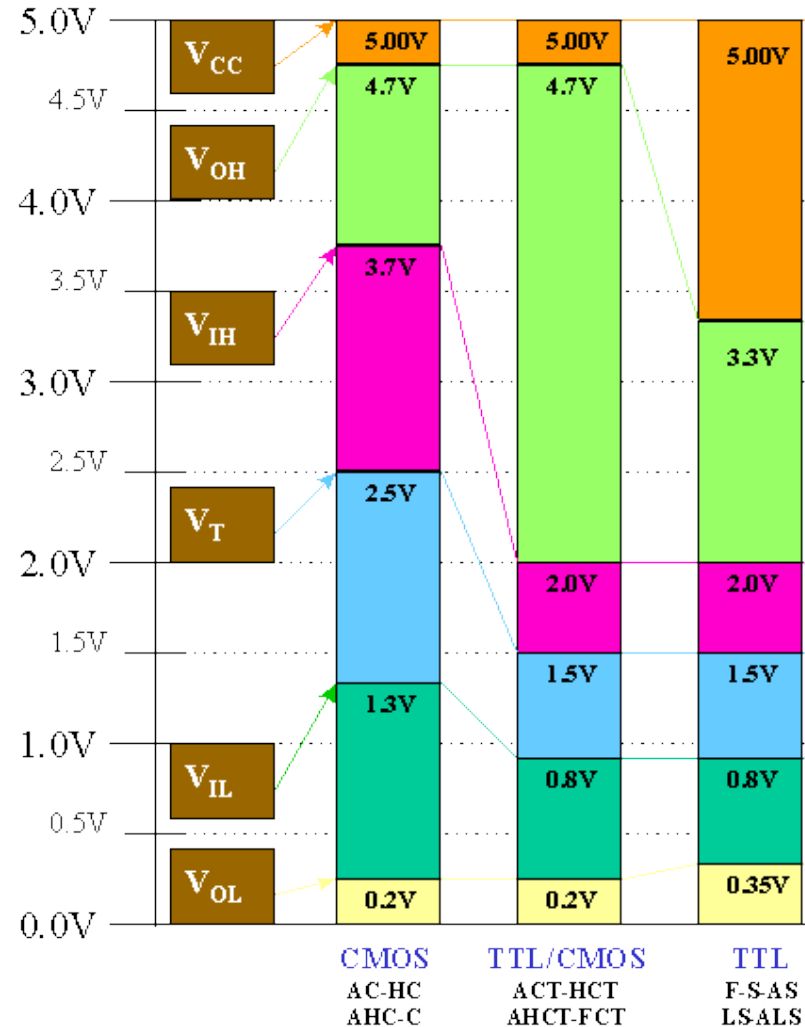
- Optoisolators provide a means of connecting signals without copper (so can isolate grounds, noise, etc.)
  - LED shines light on a phototransistor, bringing it into saturation
  - in the above circuit, the output is pulled up to 5 V when the input is *inactive*, and drops near ground when the input sees a voltage

# Logic Families

- **TTL**: transistor-transistor logic: BJT based
  - chips have L, LS, F, AS, ALS, or H designation
  - output: logic high has  $V_{OH} > 3.3 \text{ V}$ ; logic low has  $V_{OL} < 0.35 \text{ V}$
  - input: logic high has  $V_{IH} > 2.0 \text{ V}$ ; logic low has  $V_{IL} < 0.8 \text{ V}$
  - dead zone between 0.8V and 2.0 V
    - nominal threshold:  $V_T = 1.5 \text{ V}$
- **CMOS**: complimentary MOSFET
  - chips have HC or AC designation
  - output: logic high has  $V_{OH} > 4.7 \text{ V}$ ; logic low has  $V_{OL} < 0.2 \text{ V}$
  - input: logic high has  $V_{IH} > 3.7 \text{ V}$ ; logic low has  $V_{IL} < 1.3 \text{ V}$
  - dead zone between 1.3V and 3.7 V
    - nominal threshold:  $V_T = 2.5 \text{ V}$
  - chips with HCT are CMOS with TTL-compatible thresholds

# Logic Family Levels

- CMOS is closer to the “ideal” that logic low is zero volts and logic high is 5 volts
  - and has a bigger dead zone
- The 3.3V line accommodates both the TTL/CMOS levels
- Example: A TTL device must:
  - interpret any input below 0.8 V as logic low
  - interpret any input above 2.0 V as logic high
  - put out at least 3.3 V for logic high
  - put out less than 0.35 V for logic low
- The differing input/output thresholds lead to noise immunity



# Field-Effect Transistors

- The “standard” npn and pnp transistors use **base-current** to control the transistor current
- FETs use a field (**voltage**) to control current
- Result is **no current flows** into the control “gate”
- FETs are used almost exclusively as switches
  - pop a few volts on the control gate, and the effective resistance is nearly zero

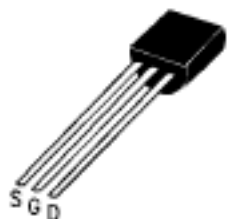
2N7000 FET

ON CHARACTERISTICS*						
$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}, I_D = 1 \text{ mA}$	0.8	2.1	3	V
$r_{DS(on)}$	Static Drain-Source On-Resistance	$V_{GS} = 10\text{V}, I_D = 0.5\text{A}$ $T_C = 125^\circ\text{C}$		1.2	5	$\Omega$
				1.9	9	$\Omega$
$V_{DS(on)}$	Drain-Source On-Voltage	$V_{GS} = 10\text{V}, I_D = 0.5\text{A}$		0.6	2.5	V
		$V_{GS} = 4.5\text{V}, I_D = 75 \text{ mA}$		0.14	0.4	V
$I_{D(on)}$	On-State Drain Current	$V_{GS} = 4.5\text{V}, V_{DS} = 10\text{V}$	75	600		mA
$g_{FS}$	Forward Transconductance	$V_{DS} = 10\text{V}, I_D = 200 \text{ mA}$	100	320		ms

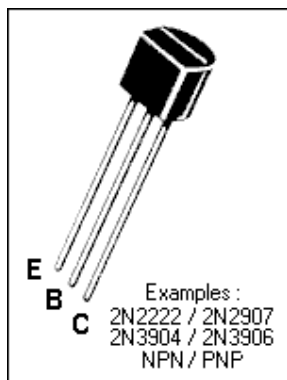


# FET Generalities

FET



BJT

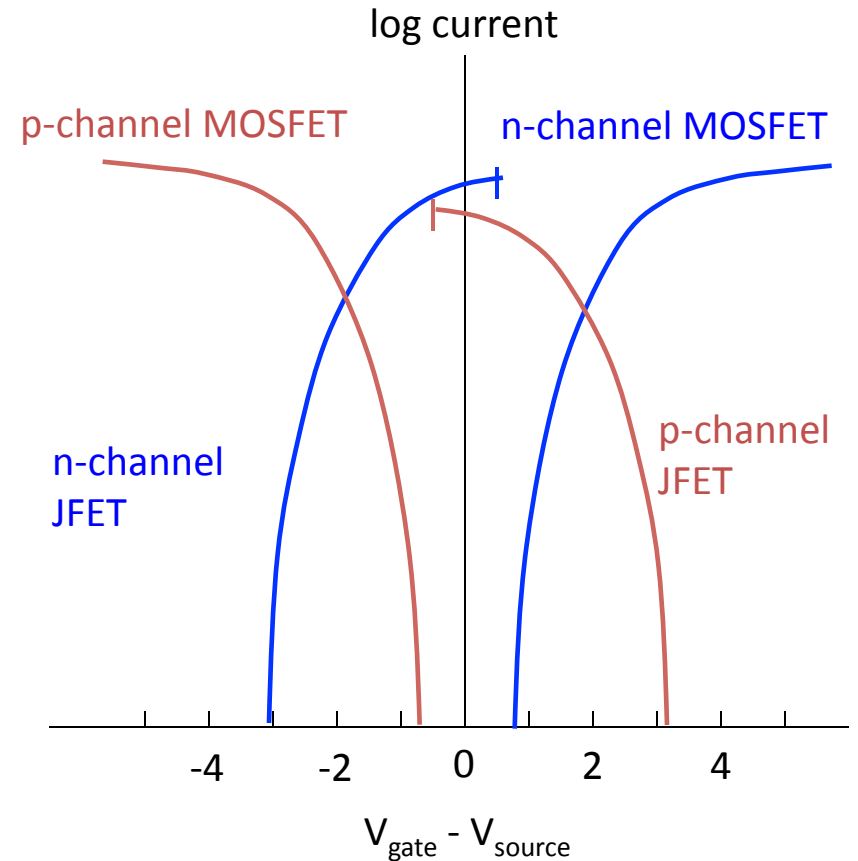


note pinout  
correspondence

- Every FET has at least three connections:
  - source (S)
    - akin to emitter (E) on BJT
  - drain (D)
    - akin to collector (C) on BJT
  - gate (G)
    - akin to base (B) on BJT
- Some have a body connection too
  - though often tied to source

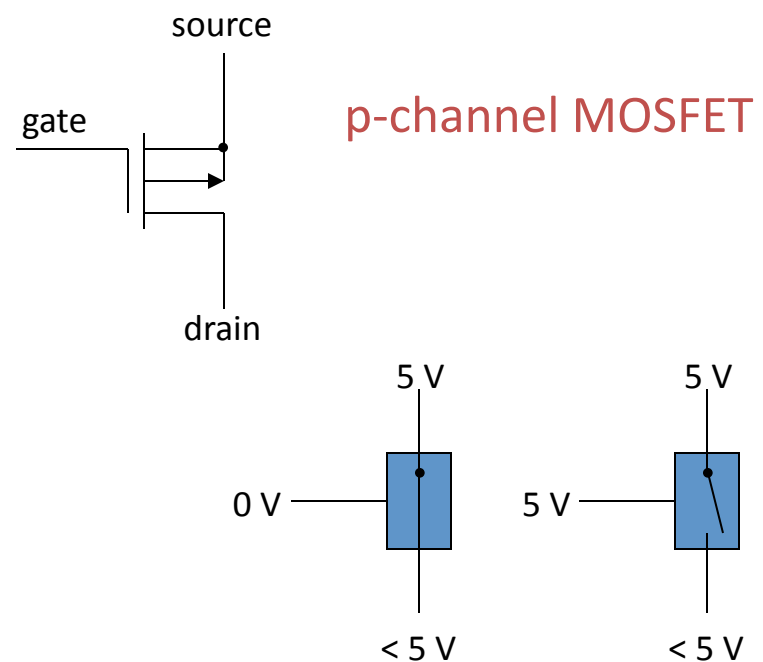
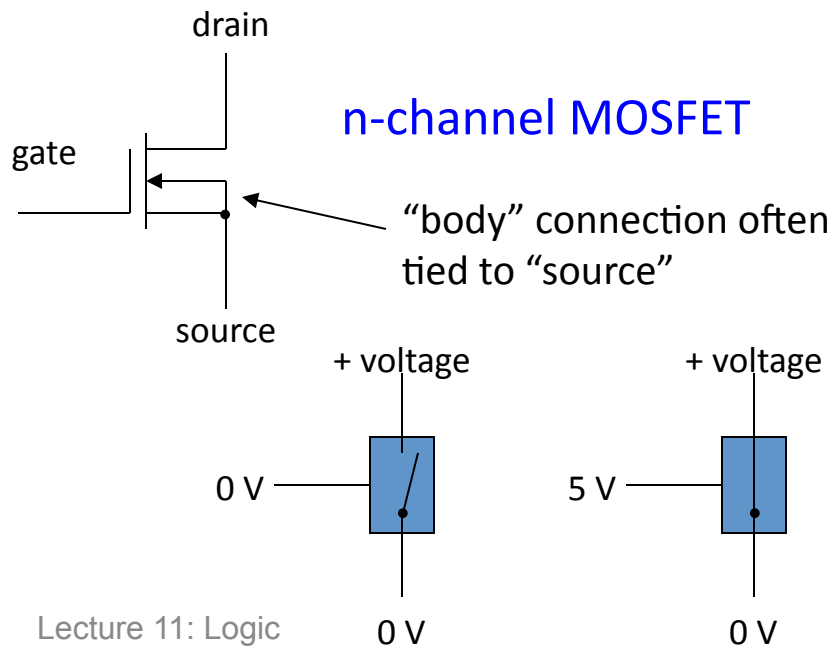
# FET Types

- Two flavors: **n** and **p**
- Two types: JFET, MOSFET
- MOSFETs more common
- JFETs conduct “by default”
  - when  $V_{\text{gate}} = V_{\text{source}}$
- MOSFETs are “open” by default
  - must turn on deliberately
- JFETs have a p-n junction at the gate, so must not forward bias more than 0.6 V
- MOSFETs have total isolation: do what you want



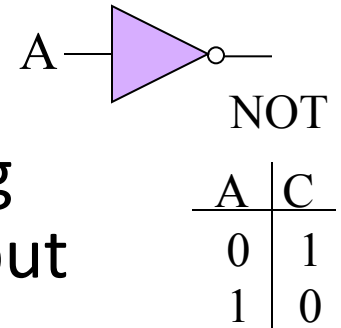
# MOSFET Switches

- MOSFETs, as applied to logic designs, act as **voltage-controlled switches**
  - **n-channel** MOSFET is closed (conducts) when positive voltage (+5 V) is applied, open when zero voltage
  - **p-channel** MOSFET is open when positive voltage (+5 V) is applied, closed (conducts) when zero voltage
    - (MOSFET means metal-oxide semiconductor field effect transistor)



# Data manipulation

- All data manipulation is based on *logic*
- Logic follows well defined rules, producing predictable digital output from certain input
- Examples:



AND

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

OR

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

XOR

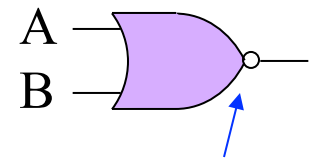
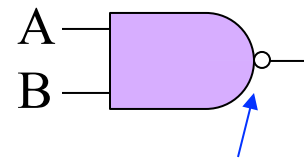
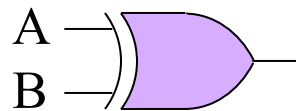
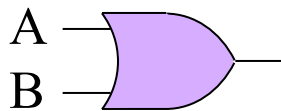
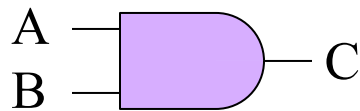
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

NAND

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

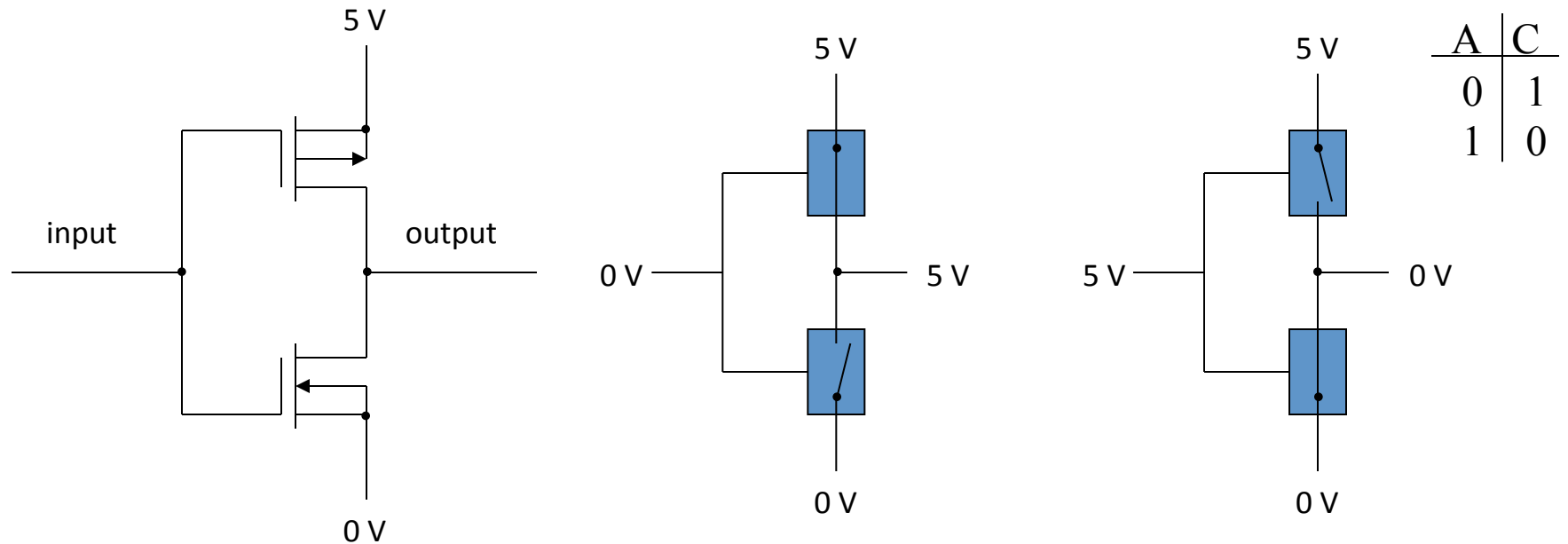
NOR

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



bubbles mean inverted (e.g., NOT AND → NAND)

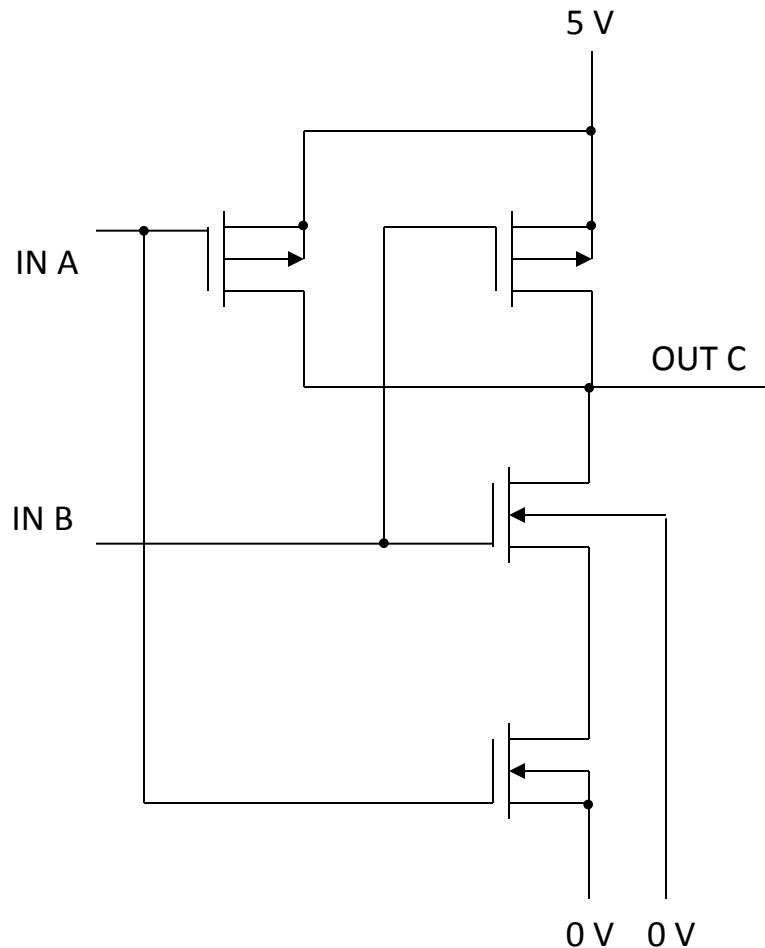
# An inverter (NOT) from MOSFETs: NOT



A	C
0	1
1	0

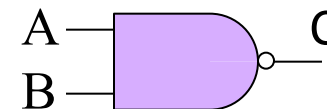
- 0 V input turns **OFF** lower (n-channel) FET, turns **ON** upper (p-channel), so output is connected to +5 V
- 5 V input turns **ON** lower (n-channel) FET, turns **OFF** upper (p-channel), so output is connected to 0 V
  - Net effect is logic inversion:  $0 \rightarrow 5; 5 \rightarrow 0$
- Complementary MOSFET pairs  $\rightarrow$  **CMOS**

# A NAND gate from scratch:



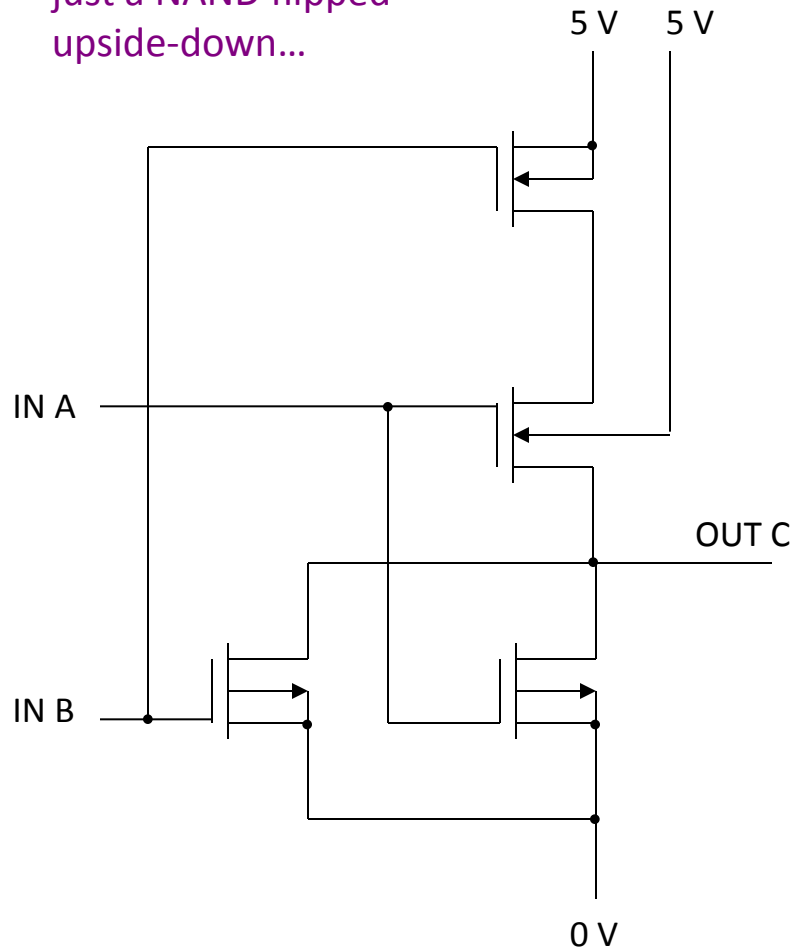
- Both inputs at zero:
  - lower two FETs **OFF**, upper two **ON**
  - result is output HI
- Both inputs at 5 V:
  - lower two FETs **ON**, upper two **OFF**
  - result is output LOW
- IN A at 5V, IN B at 0 V:
  - upper left **OFF**, lowest **ON**
  - upper right **ON**, middle **OFF**
  - result is output HI
- IN A at 0 V, IN B at 5 V:
  - opposite of previous entry
  - result is output HI

NAND		
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0



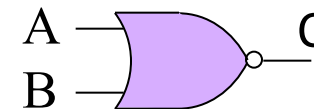
# A NOR gate from scratch:

just a NAND flipped upside-down...



- Both inputs at zero:
  - lower two FETs OFF, upper two ON
  - result is output HI
- Both inputs at 5 V:
  - lower two FETs ON, upper two OFF
  - result is output LOW
- IN A at 5V, IN B at 0 V:
  - lower left OFF, lower right ON
  - upper ON, middle OFF
  - result is output LOW
- IN A at 0 V, IN B at 5 V:
  - opposite of previous entry
  - result is output LOW

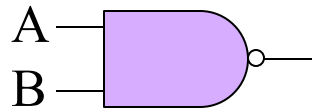
NOR		
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0



# All Logic from NANDs Alone

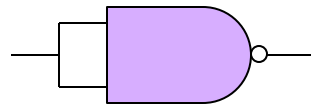
NAND

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



NOT

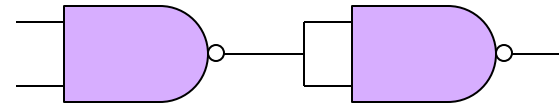
A	C
0	1
1	0



AND

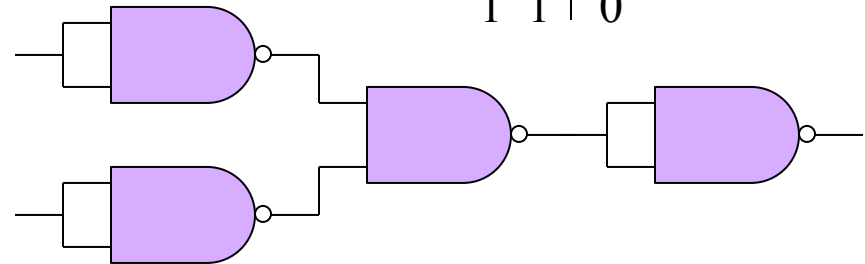
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

invert output (invert NAND)



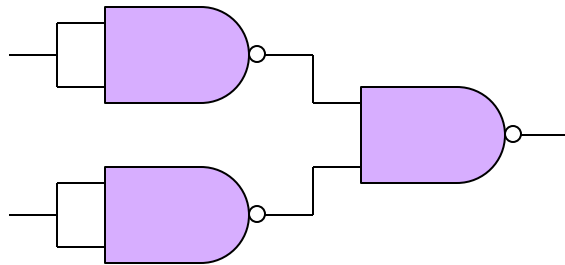
NOR

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



OR

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

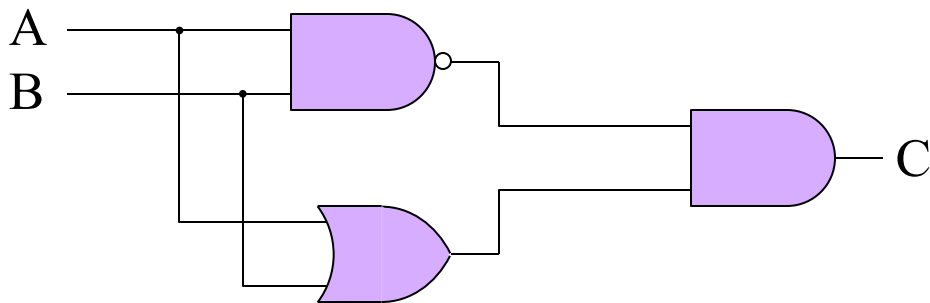


invert both inputs

invert inputs *and* output (invert OR)



# One last type: XOR



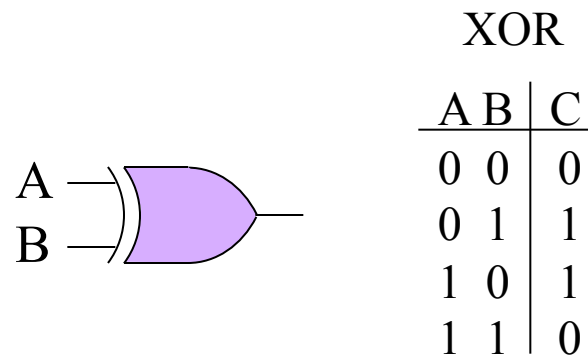
- $XOR = (A \text{ NAND } B) \text{ AND } (A \text{ OR } B)$
- And this you already know you can make from composite NAND gates (though requiring 6 total)
- Then, obviously, XNOR is the inverse of XOR
  - so just stick an inverter on the output of XOR

# Rule the World


- Now you know how to build **ALL** logic gates out of **n-channel** and **p-channel** MOSFETs
  - because you can build a NAND from 4 MOSFETs
  - and all gates from NANDs
- That means you can build computers
  
- So now you can rule the world!

# Arithmetic Example

- Let's add two binary numbers:  
    00101110 = 46  
    + 01001101 = 77  
    01111011 = 123
- How did we do this? We have rules:  
     $0 + 0 = 0$ ;  $0 + 1 = 1 + 0 = 1$ ;  $1 + 1 = 10$  (2): (0, carry 1);  
     $1 + 1 + (\text{carried } 1) = 11$  (3): (1, carry 1)
- Rules can be represented by gates
  - If two input digits are A & B, output digit looks like XOR operation (but need to account for carry operation)

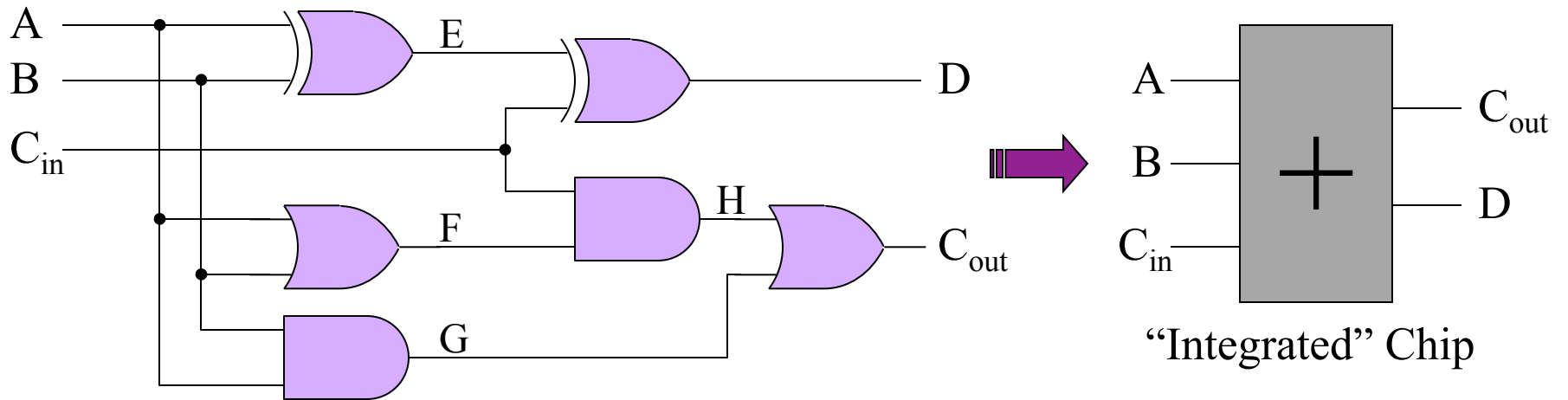


# Can make rule table:

$C_{in}$	A	B		D	$C_{out}$
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1

- Digits A & B are added, possibly accompanied by carry instruction from previous stage
- Output is new digit, D, along with carry value
  - D looks like XOR of A & B when  $C_{in}$  is 0
  - D looks like XNOR of A & B when  $C_{in}$  is 1
  - $C_{out}$  is 1 if two or more of A, B,  $C_{in}$  are 1

# Binary Arithmetic in Gates



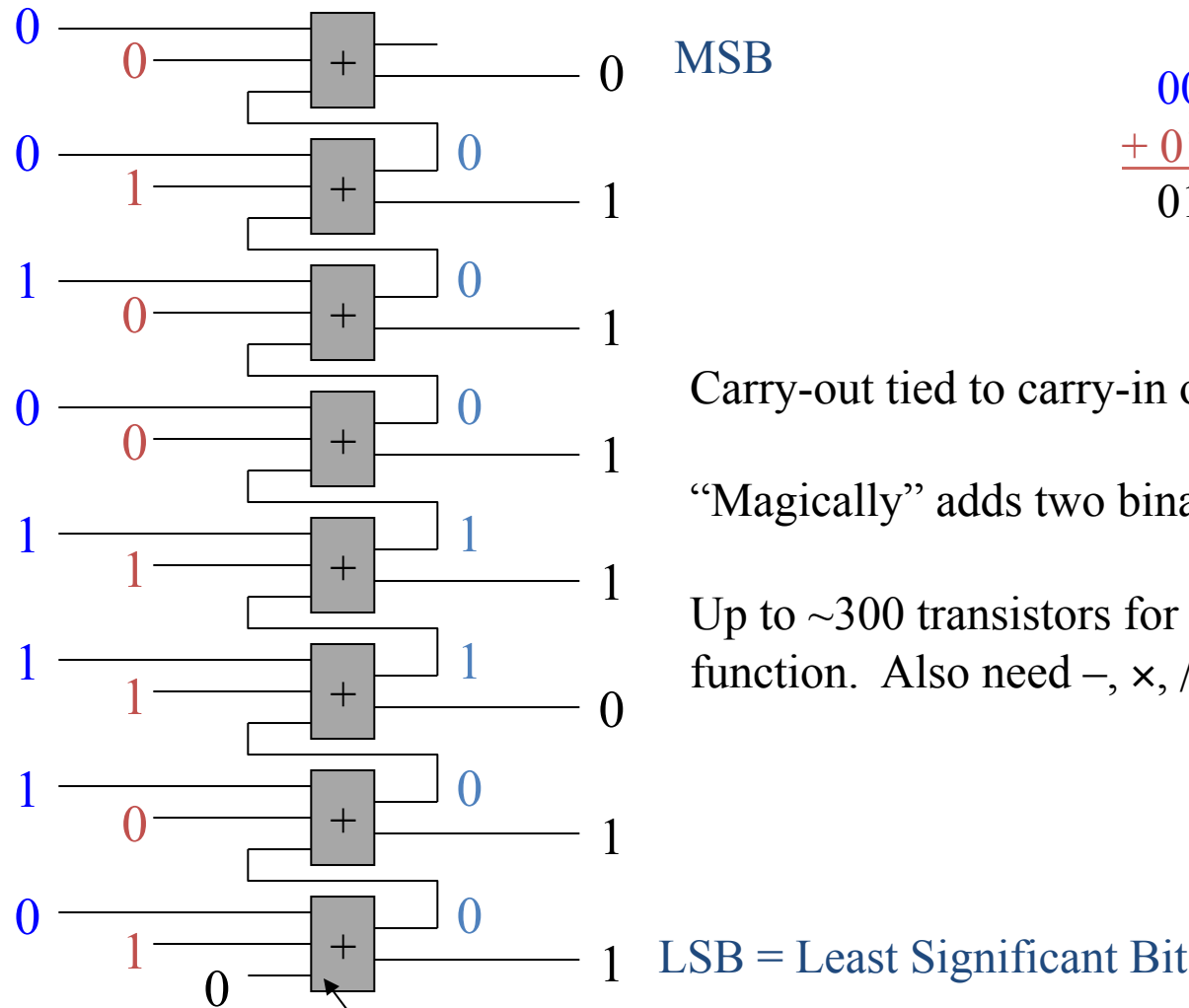
Input			Intermediate				Output	
A	B	$C_{in}$	E	F	H	G	D	$C_{out}$
0	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	1	0
1	0	0	1	1	0	0	1	0
1	1	0	0	1	0	1	0	1
0	0	1	0	0	0	0	1	0
0	1	1	1	1	1	0	0	1
1	0	1	1	1	1	0	0	1
1	1	1	0	1	1	1	1	1

Each digit requires 6 gates

Each gate has ~6 transistors

~36 transistors per digit

# 8-bit binary arithmetic (cascaded)



$$\begin{array}{r}
 \phantom{00}11 \\
 00101110 = 46 \\
 + 01001101 = 77 \\
 \hline
 01111011 = 123
 \end{array}$$

Carry-out tied to carry-in of next digit.

“Magically” adds two binary numbers

Up to ~300 transistors for this basic function. Also need  $-$ ,  $\times$ ,  $/$ , & lots more.

Integrated one-digit binary arithmetic unit (prev. slide)

# Computer technology built up from pieces

- The foregoing example illustrates the way in which computer technology is built
  - start with little pieces (transistors acting as switches)
  - *combine* pieces into functional blocks (gates)
  - *combine* these blocks into higher-level function (e.g., addition)
  - *combine* these new blocks into cascade (e.g., 8-bit addition)
  - blocks get increasingly complex, more capable
- Nobody on earth understands modern chip inside-out
  - Grab previously developed blocks and run
  - Let a computer design the gate arrangements (eyes closed!)

# Reading

- As before, *The Art of Electronics* by Horowitz and Hill, and the *Student Manual* accompaniment by Hayes and Horowitz are valuable resources
- Text reading:
  - p. 432 (p. 461 in 3<sup>rd</sup> ed.) on comparators
  - 6.2.5 on relays (esp. solid state)
  - pp. 461–462 (490–491 in 3<sup>rd</sup>) paragraph on opto-isolators
  - 6.6.10 on logic families
  - p. 410 (p. 449 in 3<sup>rd</sup>) on FETs
  - 6.6.1, 6.6.2, 6.6.3, 6.6.4 on digital logic
  - 6.6.7 on DACs, ADCs