Computer Organization and Design

Transistors & Logic - I

Henry Fuchs

Slides adapted from Montek Singh, who adapted them
from Leonard McMillan and from Gary Bishop
Back to McMillan & Chris Terman, MIT 6.004 1999

Tuesday, March 17, 2015

Lecture 9

Today's Topics

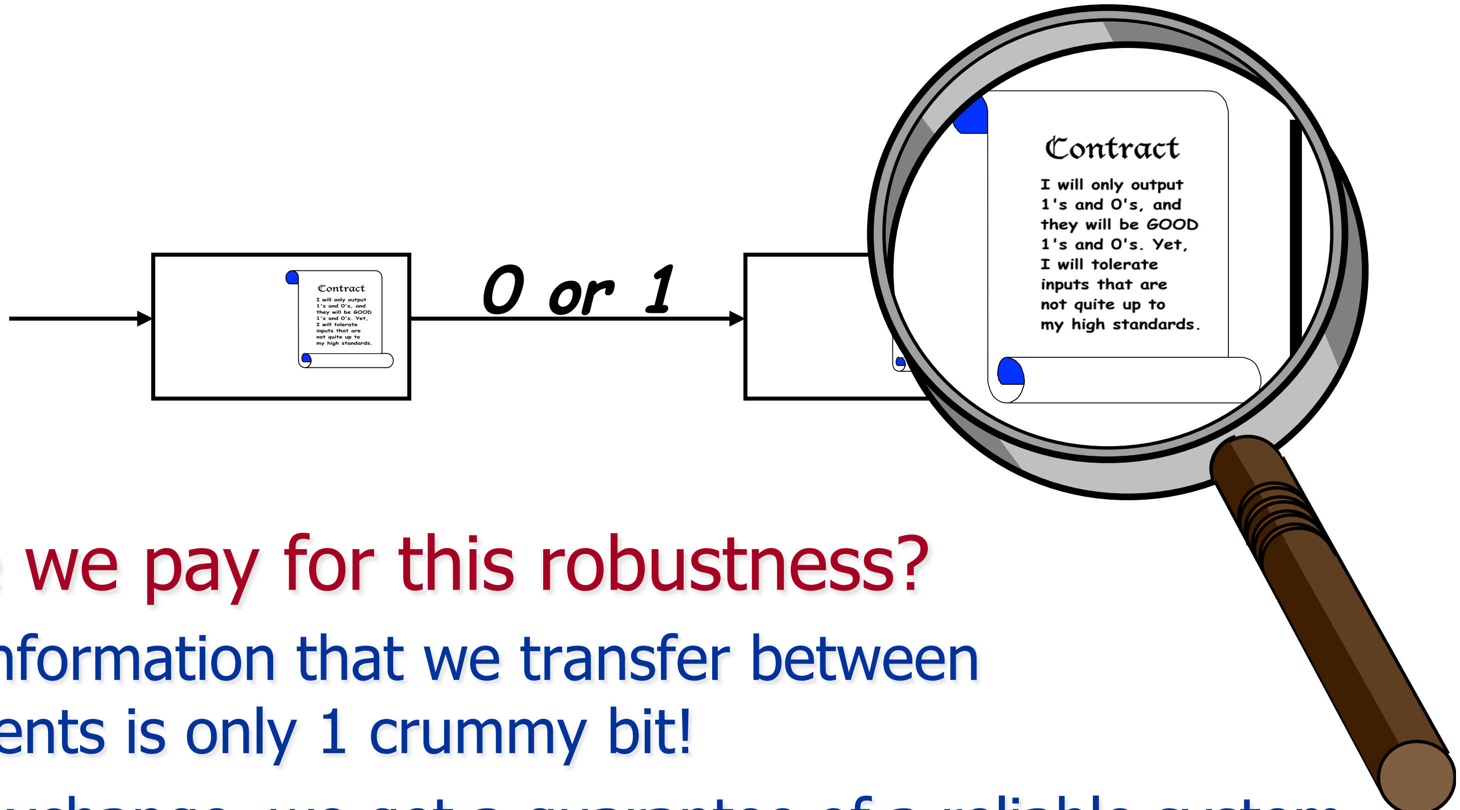
- Where are we in this course?

- Today's topics
  - Why go digital?
  - Encoding bits using voltages
  - Digital design primitives
    - transistors and gates

Let's go digital!

⁕ Why DIGITAL?

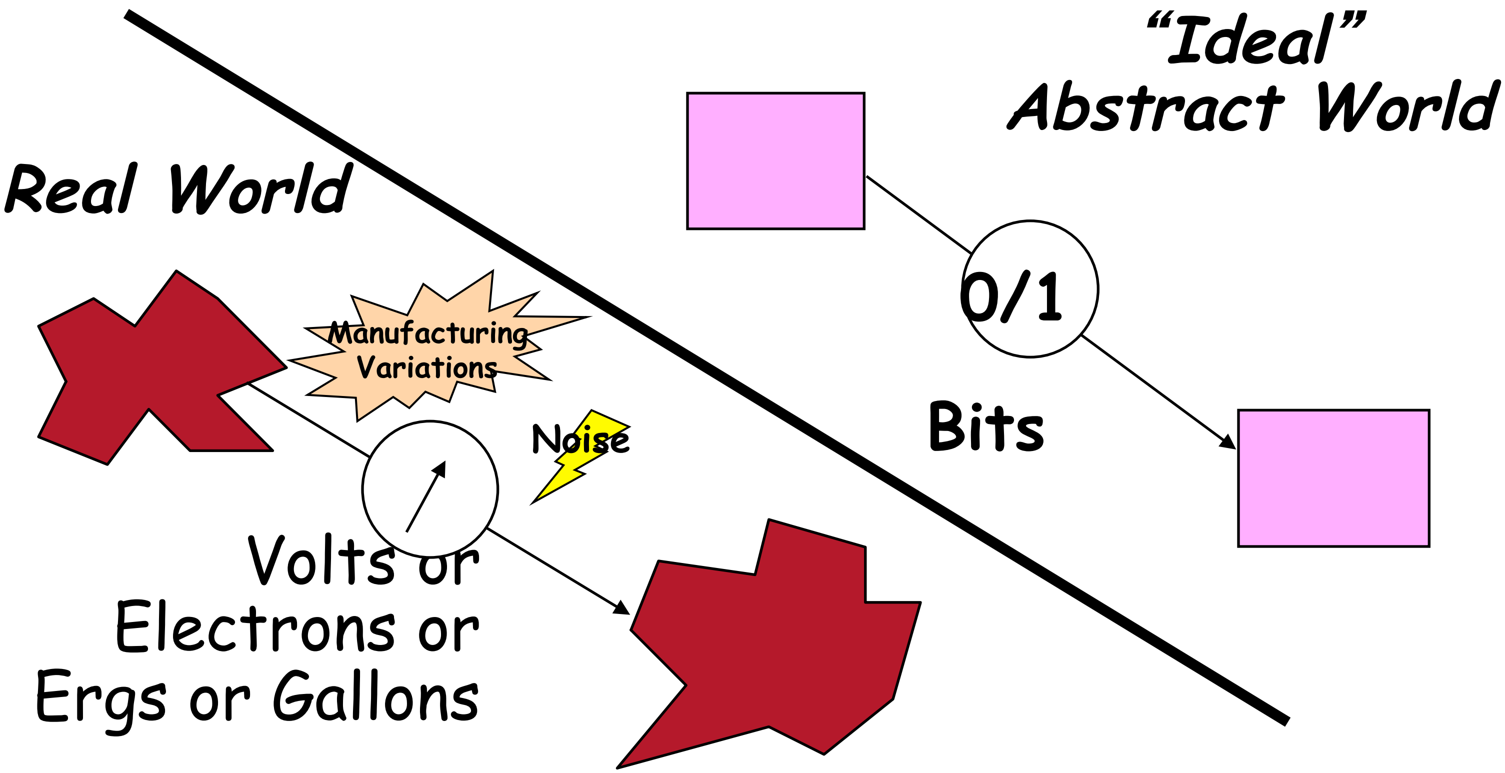
- … because it helps guarantee a reliable system



⁕ The price we pay for this robustness?

- All the information that we transfer between components is only 1 crummy bit!
- But, in exchange, we get a guarantee of a reliable system.

The Digital Abstraction



Keep in mind, the world is not digital, we engineer it to behave so.
We must use real physical phenomena to implement digital designs!

Types of Digital Components

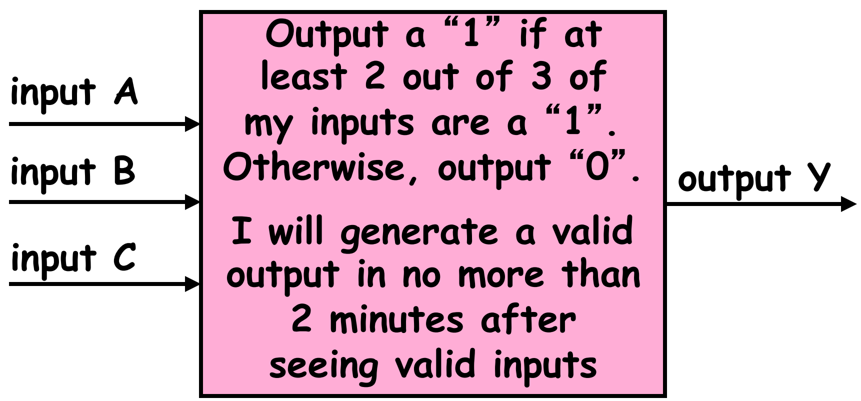
- Two categories of components
  - those whose output only depends on their current inputs
    - called COMBINATIONAL
    - they are "memory-less", don't remember the past
  - those who output depends also on their past state
    - called SEQUENTIAL
    - they are "state-holding", remember their past
    - key to building memories

Terminology

- System
  - a reasonably large assembly of components
  - division of a system into components is typically arbitrary but almost always hierarchical
- Component/Element
  - an individual part of a bigger system
  - clearly-defined function and interface
  - implement it and put a black-box around it
  - larger components created using smaller components
- Circuit
  - a small (often leaf-level) component consisting of a network of gates

Combinational Components

- A circuit is combinational if-and-only-if it has:
  - one or more digital *inputs*
  - one or more digital *outputs*
  - a *functional specification* that details the value of each output for every possible combination of valid input values
    - output depends only on the latest inputs
  - a *timing specification* consisting (at minimum) of an upper bound t_{pd} on the time this circuit will take to produce the output value once stable valid input values are applied

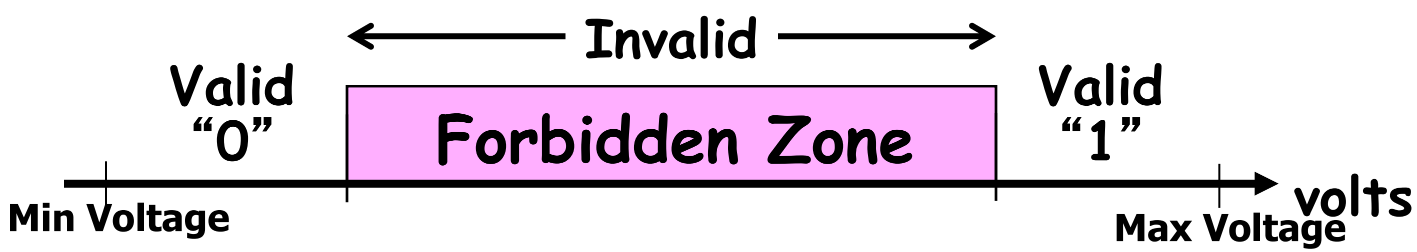


A Combinational Digital System

- Theorem: A system of interconnected elements is combinational if-and-only-if:
  - each primitive circuit element is combinational
  - every input is connected to exactly one output or directly to a source of 0's or 1's
  - the circuit contains no directed cycles
    - no feedback (yet!)
- Proof: By induction
  - Start with the rightmost level of elements
    - their output only depends on their inputs, which in turn are outputs of the level of element just to their left
    - and so on… until you arrive at the leftmost inputs
- But, in order to realize digital processing elements we have one more requirement!

Noise Margins

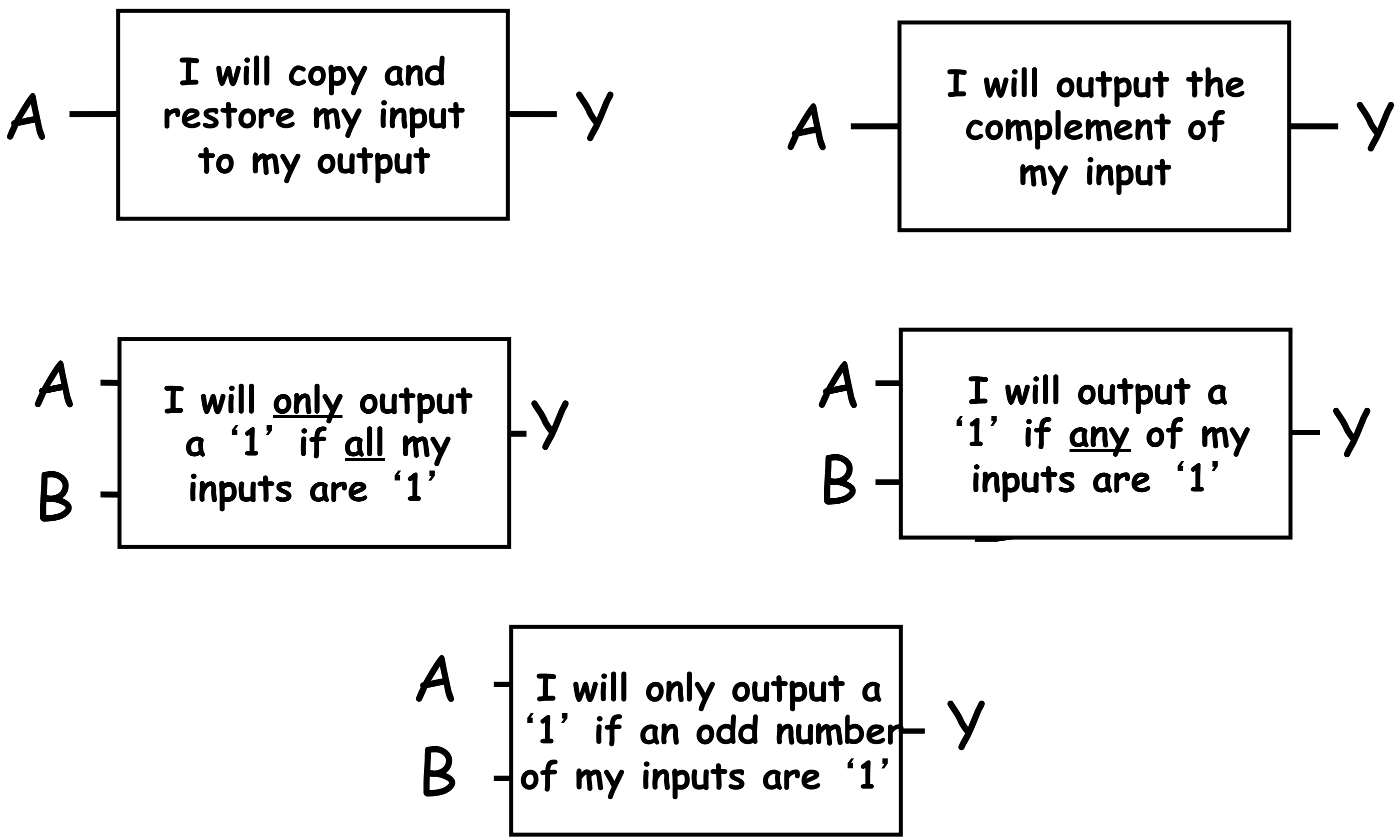
- Key idea: Keep "0"s distinct from the "1"s
  - say, "0" is represented by min voltage (e.g., 0 volts)
  - … "1" is represented by high voltage (e.g., 1.8 volts)
  - use the same voltage representation throughout the entire system!
- For reliability, outlaw "close calls"
  - forbid a range of voltages between "0" and "1"



CONSEQUENCE: Notion of "VALID" and "INVALID" logic levels

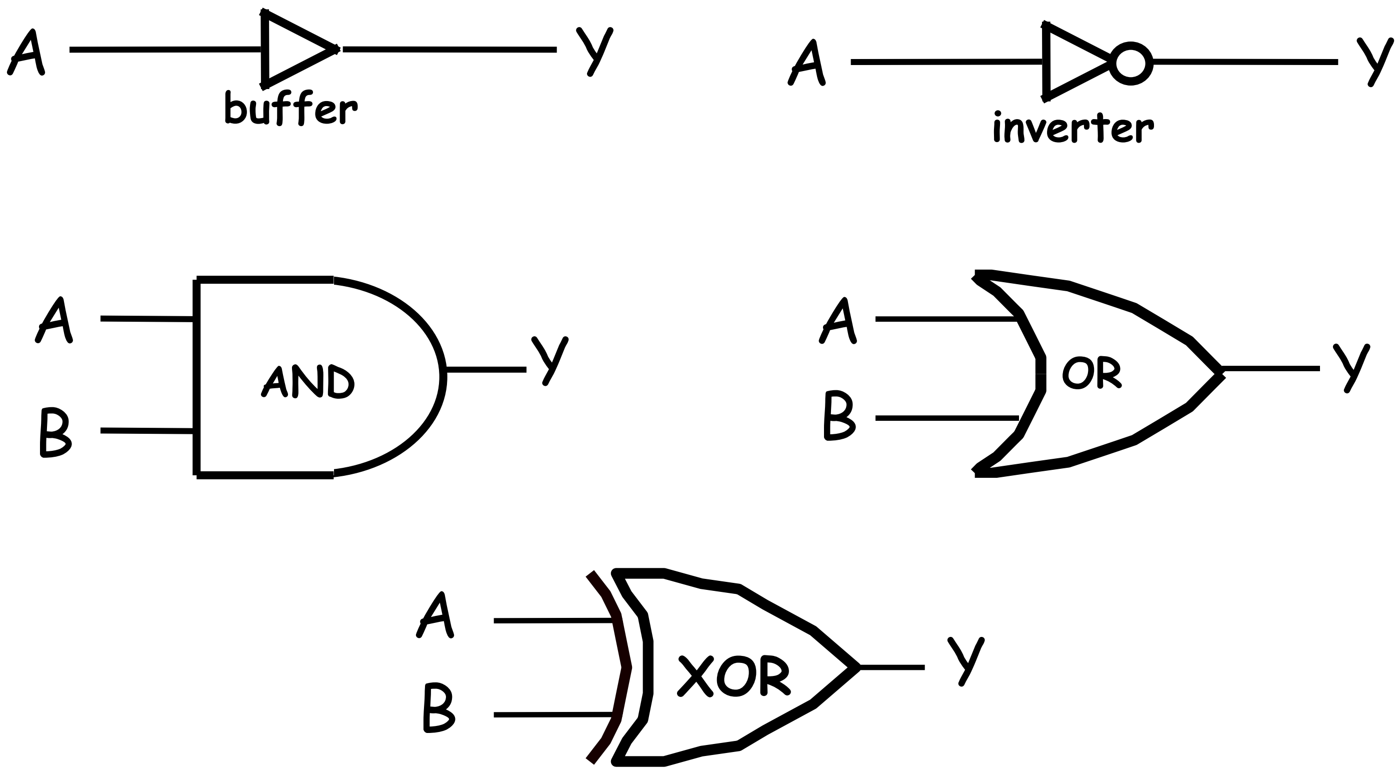
Digital Processing Elements

- Some digital processing elements occur so frequently that we give them special names and symbols



Digital Processing Elements

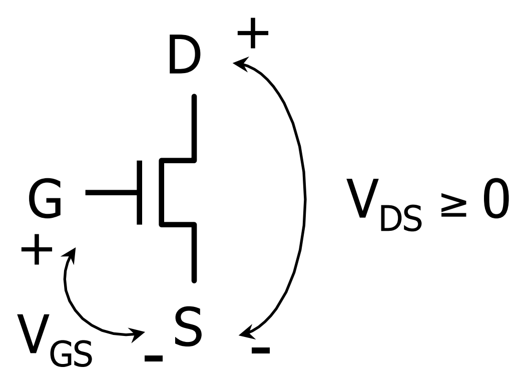
* Some digital processing elements occur so frequently that we give them special names and symbols



Most common technology today

- … is called CMOS
  - everything built using *transistors*
  - a transistor is just a switch
- 2 types of transistors
  - n-type
    - called “NFET”, or “nMOS” or “n channel transistor” or “n transistor”
    - switch is on (i.e., conducts) when its control input is ‘1’
  - p-type
    - called “PFET”, or “pMOS”, or “p channel transistor” or “p transistor”
    - switch is on (i.e., conducts) when its control input is ‘0’
  - need both types to build useful gates

N-Channel Field-Effect Transistors (NFETs)



When the gate voltage is high, the switch connects. Good at pulling things "low".

Operating regions:

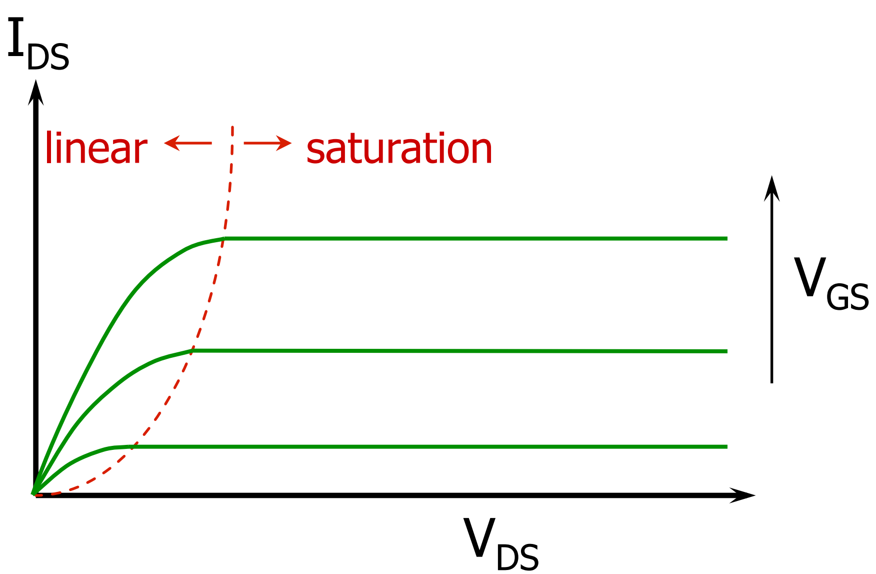
cut-off:
$V_{GS} < V_{TH}$

0.5V

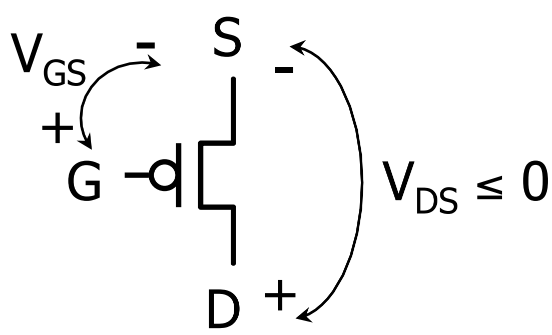
linear:
$V_{GS} \geq V_{TH}$
$V_{DS} < V_{Dsat}$

V_{GS} - V_{TH}

saturation:
$V_{GS} \geq V_{TH}$
$V_{DS} \geq V_{Dsat}$



P-Channel Field-Effect Transistors (PFETs)



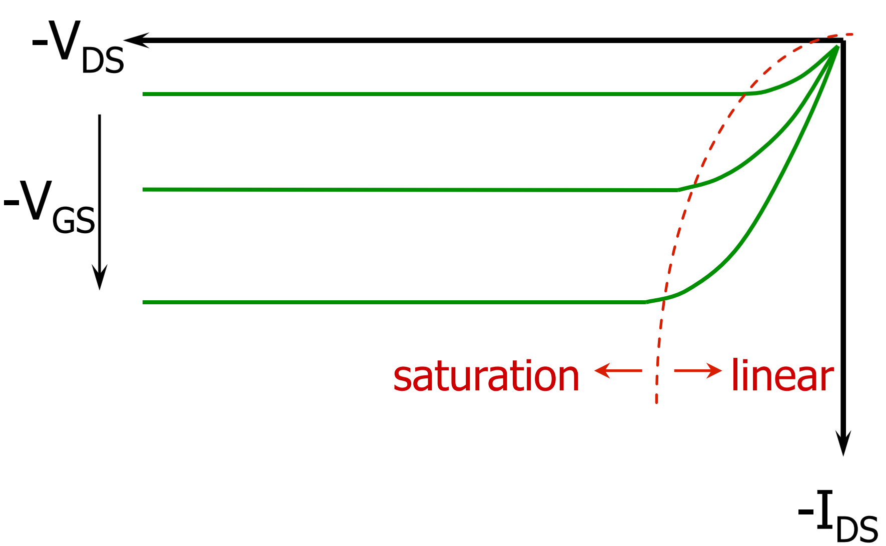
When the gate voltage is low, the switch connects. Good at pulling things "high".

Operating regions:

cut-off:
$V_{GS} > V_{TH}$ **−0.5V**

linear:
$V_{GS} \leq V_{TH}$
$V_{DS} > V_{Dsat}$ **V_{GS} - V_{TH}**

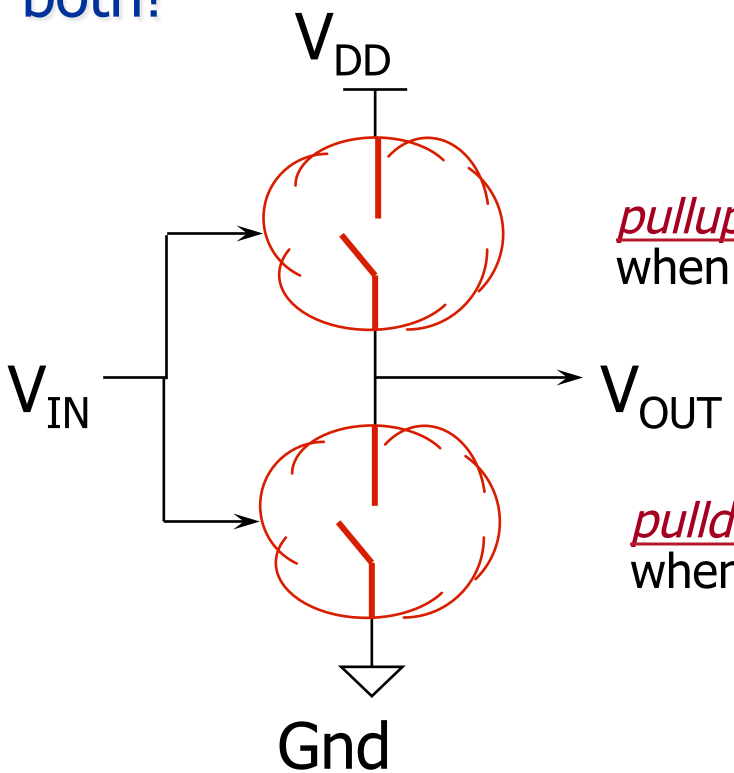
saturation:
$V_{GS} \leq V_{TH}$
$V_{DS} \leq V_{Dsat}$



From Transistors… to Gates!

Logic Gate recipe:

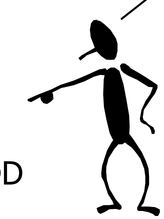
- use <u>complementary</u> arrangements of PFETs and NFETs
  - ➢ called CMOS ("complementary metal-oxide semiconductor")
- at any time: either "pullup" active, or "pulldown", never both!

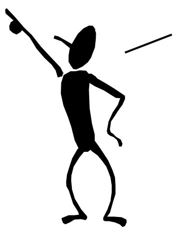


pullup: make this connection when V_{IN} is near 0 so that $V_{OUT} = V_{DD}$

pulldown: make this connection when V_{IN} is near V_{DD} so that $V_{OUT} = 0$

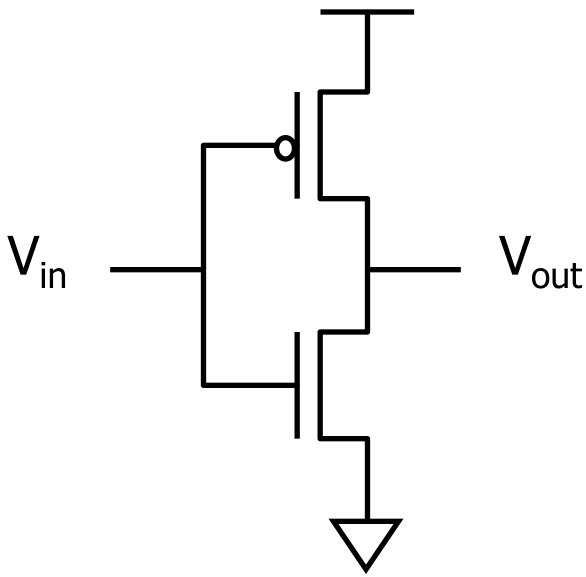
We'll use p-type here

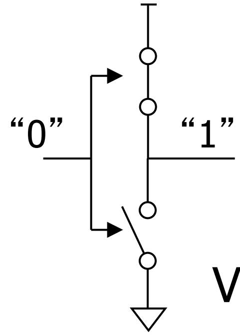


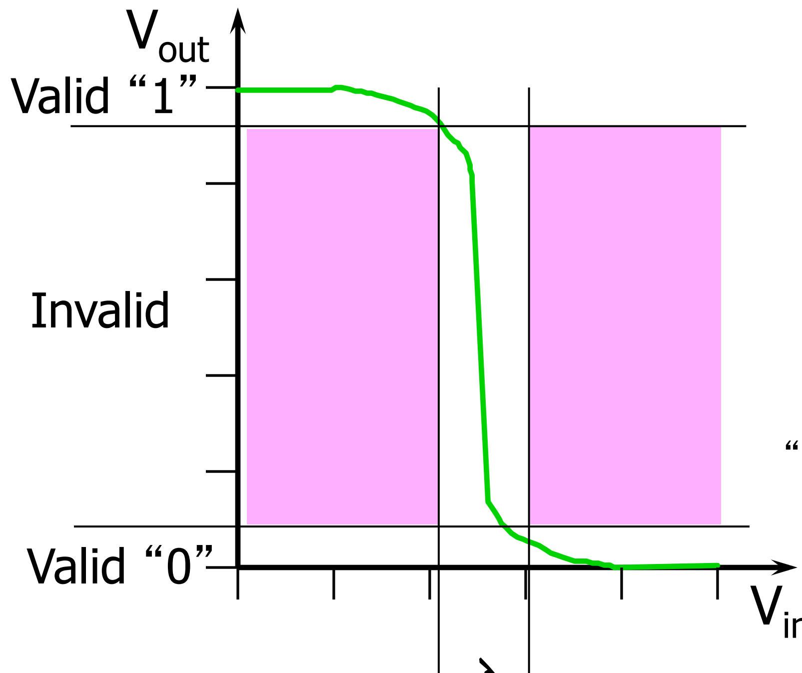


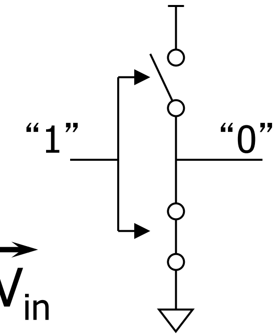
and, n-type here

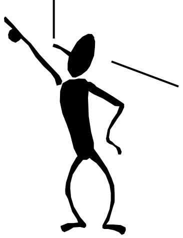
CMOS Inverter



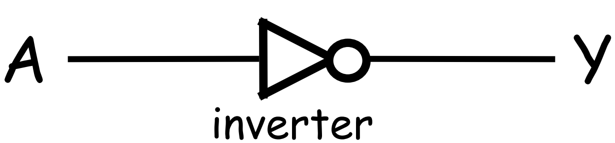




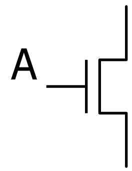




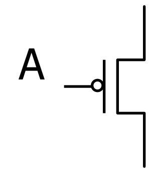
Only a narrow range of input voltages result in “invalid” output values. (This diagram is greatly exaggerated)



CMOS Complements

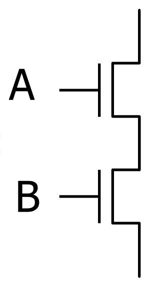


conducts when A is high

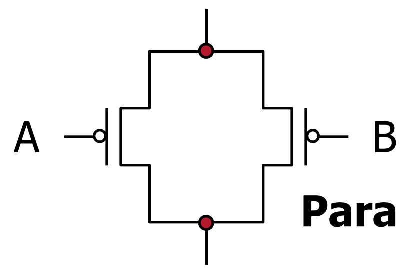


conducts when A is low

Series N connections:



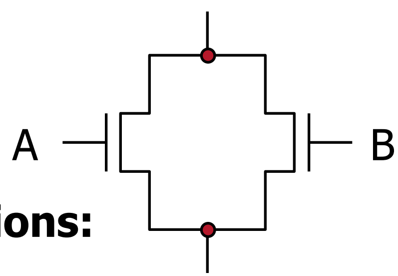
conducts when A is high
and B is high: $A \cdot B$



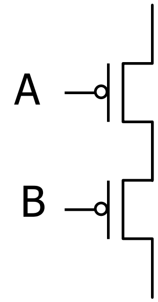
Parallel P connections:

conducts when A is low
or B is low: $\overline{A}+\overline{B} = \overline{A \cdot B}$

Parallel N connections:



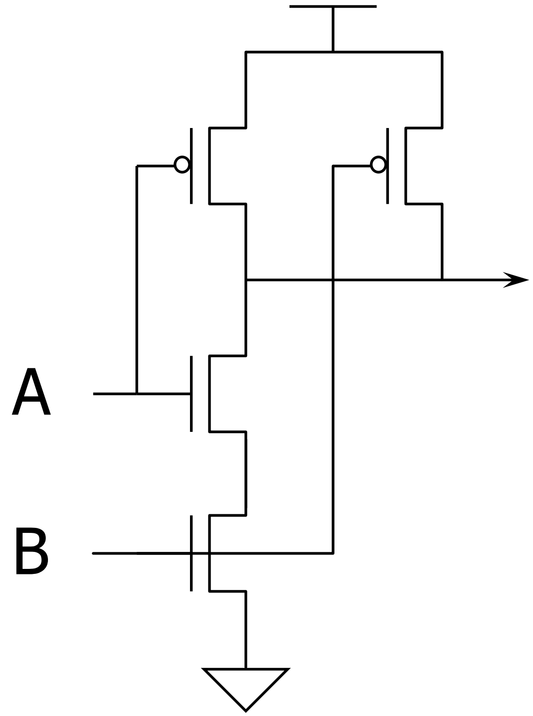
conducts when A is high
or B is high: $A+B$



Series P connections:

conducts when A is low
and B is low: $\overline{A} \cdot \overline{B} = \overline{A+B}$

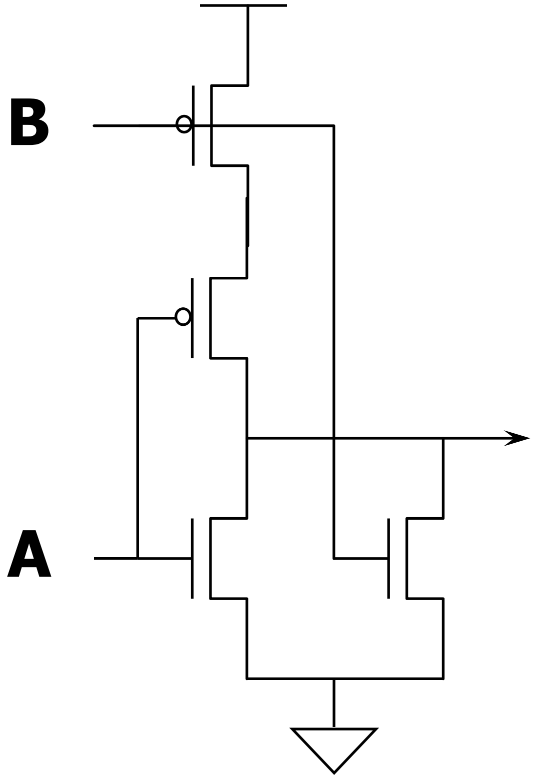
A Two Input Logic Gate



What function does this gate compute?

A	B	C
0	0	
0	1	
1	0	
1	1	

Here's Another…



What function does this gate compute?

A	B	C
0	0	
0	1	
1	0	
1	1	

Next

- We' ll look at one level of abstraction higher than transistors, logic gates
- We' ll then use gates to construct arithmetic circuits