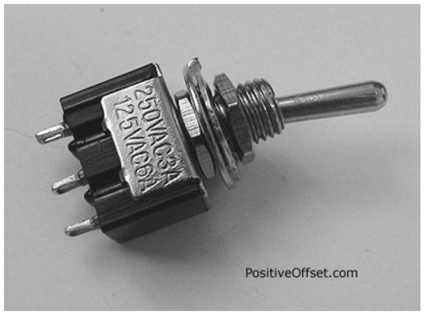
Lec 2: Gates and Logic

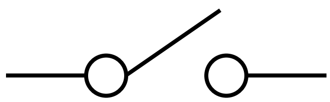
Kavita Bala
CS 3410, Fall 2008
Computer Science
Cornell University

Announcements

- Class newsgroup created
  - Posted on web-page
- Use it for partner finding
- First assignment is to find partners
  - Due this Friday
- Sections start this week

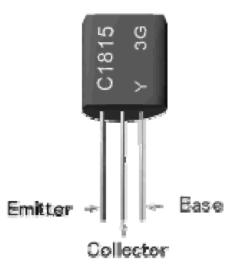
A switch



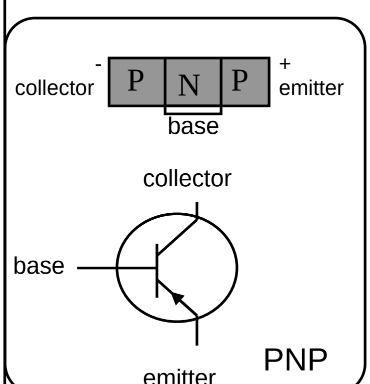


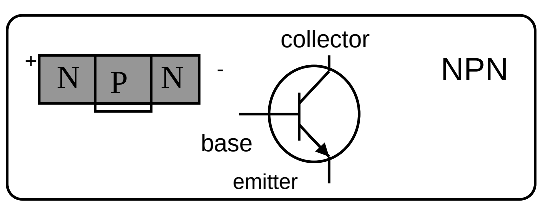
- A switch is a simple device that can act as a conductor or isolator
- Can be used for amazing things…

Transistors



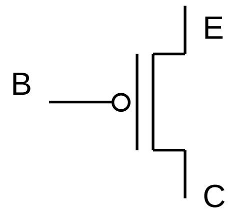
- Solid-state switch
  - The most amazing invention of the 1900s
- PNP and NPN





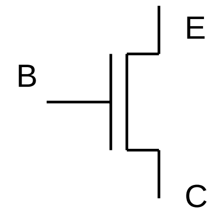
P and N Transistors

- PNP Transistor



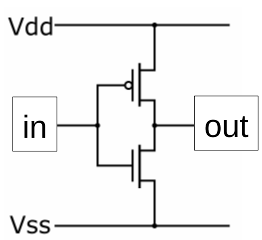
- Connect E to C when base = 0

- NPN Transistor

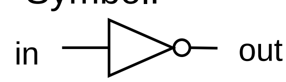


- Connect E to C when base = 1

Inverter



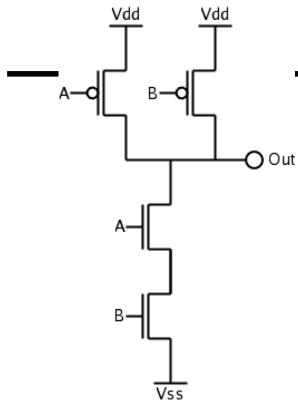
- Function: NOT
- Called an inverter
- Symbol:



In	Out
0	1
1	0

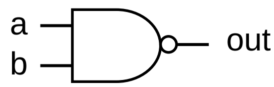
Truth table

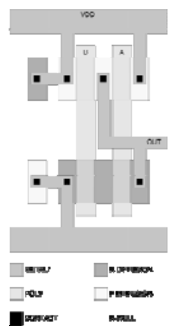
- Useful for taking the inverse of an input
- CMOS: complementary-symmetry metal**–oxide–semiconductor**



NAND Gate

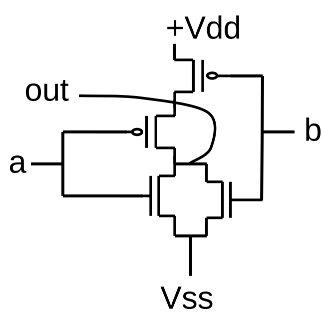
- Function: NAND
- Symbol:





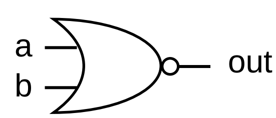
A	B	out
0	0	1
1	0	1
0	1	1
1	1	0

NOR Gate

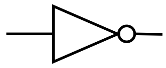
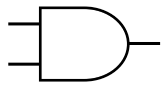
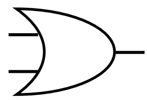


- Function: NOR
- Symbol:

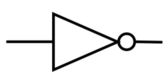
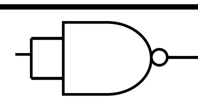
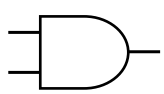
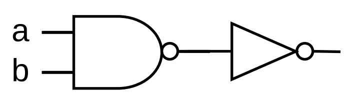
A	B	out
0	0	1
1	0	0
0	1	0
1	1	0

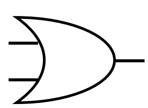
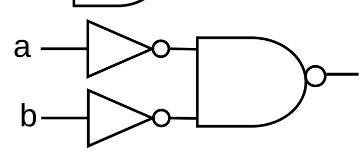


Building Functions

- NOT: 
- AND: 
- OR: 
- NAND and NOR are universal
  - Can implement any function with NAND or just NOR gates
  - useful for manufacturing

Building Functions

- NOT:  
- AND:  

- OR:  

- NAND and NOR are universal
  - Can implement any function with NAND or just NOR gates
  - useful for manufacturing

Logic Equations

- AND
  - out = a b = a&b = a∧b
- OR
  - out = a + b = a|b = a∨b
- NOT
  - out = $\overline{a}$ = !a = ¬a

Identities

- Identities useful for manipulating logic equations
  - For optimization & ease of implementation

  - $a + \overline{a} = 1$
  - $a + 0 = a$
  - $a + 1 = 1$
  - $a\,\overline{a} = 0$
  - $a\,0 = 0$
  - $a\,1 = a$
  - $\overline{a(b+c)} = \overline{a} + \overline{b}\,\overline{c}$
  - $\overline{(a + b)} = \overline{a}\,\overline{b}$
  - $\overline{(a\,b)} = \overline{a} + \overline{b}$
  - $a + a\,b = a$

Logic Manipulation

- Can specify functions by describing gates, truth tables or logic equations
- Can manipulate logic equations algebraically
- Can also use a truth table to prove equivalence
- Example: (a+b)(a+c) = a + bc

(a+b)(a+c)
= aa + ab + ac + bc
= a + a(b+c) + bc
= a(1 + (b+c)) + bc
= a + bc

a	b	c	a+b	a+c	LHS	bc	RHS
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Logic Minimization

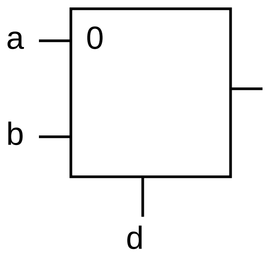
- A common problem is how to implement a desired function most efficiently
- One can derive the equation from the truth table

a	b	c	minterm
0	0	0	$\overline{a}\overline{b}\overline{c}$
0	0	1	$\overline{a}\overline{b}c$
0	1	0	$\overline{a}b\overline{c}$
0	1	1	$\overline{a}bc$
1	0	0	$a\overline{b}\overline{c}$
1	0	1	$a\overline{b}c$
1	1	0	$ab\overline{c}$
1	1	1	abc

for all outputs
that are 1,
take the corresponding
minterm
Obtain the result in
"sum of products" form

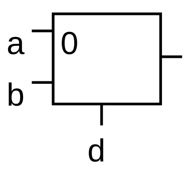
- How does one find the most efficient equation?
  - Manipulate algebraically until satisfied
  - Use Karnaugh maps (or K maps)

Multiplexer



- A multiplexer selects between multiple inputs
  - out = a, if d = 0
  - out = b, if d = 1
- Build truth table
- Minimize diagram
- Derive logic diagram

Multiplexer Implementation



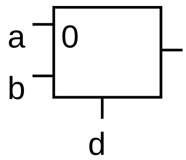
a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Build a truth table

$= abd + ab\bar{d} + \bar{a}bd + a\bar{b}\bar{d}$

$= a\bar{d} + bd$

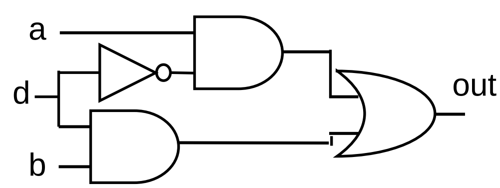
Multiplexer Implementation



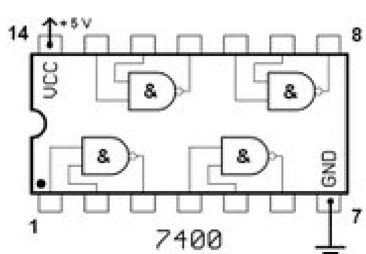
- Draw the circuit

a	b	d	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

– out = $a\bar{d} + bd$

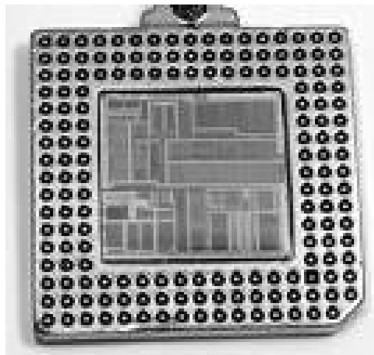


Logic Gates



- One can buy gates separately
  - ex. 74xxx series of integrated circuits
  - cost ~$1 per chip, mostly for packaging and testing
- Cumbersome, but possible to build devices using gates put together manually

Integrated Circuits



- Or one can manufacture a complete design using a custom mask
- Intel Pentium has approximately 125 million transistors

Voting machine

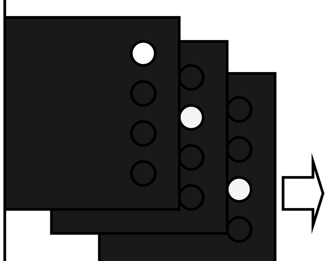
- Build something interesting
- A voting machine
  - Elections are coming up!
- Assume:
  - A vote is recorded on a piece of paper,
  - by punching out a hole,
  - there are at most 7 choices
  - we will not worry about “hanging chads” or “invalids”

Voting machine

- For now, let’s just display the numerical identifier to the ballot supervisor
  - we won’t do counting yet, just decoding
  - we can use four photo-sensitive transistors to find out which hole is punched out

- A photo-sensitive transistor detects the presence of light
- Photo-sensitive material triggers the gate

Ballot Reading

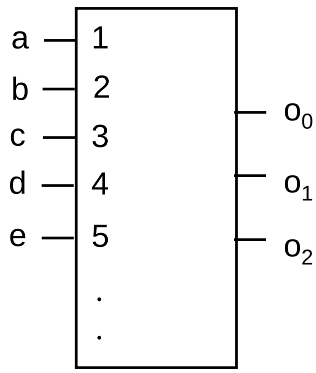


Ballots

The 3410 vote recording machine

- Input: paper with a hole in it
- Out: number the ballot supervisor can record

Encoders



A 3-bit encoder
(7-to-3)
(5 inputs shown)

- N sensors in a row
- Want to distinguish which of the N sensors has fired
- Want to represent the firing sensor number in compact form
  - N might be large
  - Only one wire is on at any time
  - Silly to route N wires everywhere, better to encode in log N wires

Number Representations

37

10^1 10^0

- Decimal numbers are written in base 10
  - $3 \times 10^1 + 7 \times 10^0 = 37$
- Just as easily use other bases
  - Base 2 - “Binary”
  - Base 8 - “Octal”
  - Base 16 – “Hexadecimal”

Number Representations

37
_ _
10^1 10^0

- Base conversion via repetitive division
  - Divide by base,
    write remainder,
    move left with quotient
  - Sanity check with 37 and 10

Binary Representation

- 37 = 32 + 4 + 1

0100101
_ _ _ _ _ _ _
2^6 2^5 2^4 2^3 2^2 2^1 2^0

64 32 16 8 4 2 1

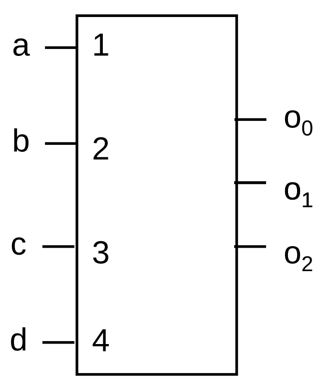
Hexadecimal Representation

$$\underline{2}\,\underline{5}$$

$16^1 \; 16^0$

- 37 decimal = $(25)_{16}$
- Convention
  - Base 16 is written with a leading 0x
  - 37 = 0x25
- Need extra digits!
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Binary to hexadecimal is easy
  - Divide into groups of 4, translate groupwise into hex digits

Encoder Truth Table

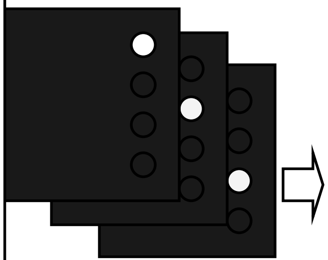


A 3-bit encoder
with 4 inputs for simplicity

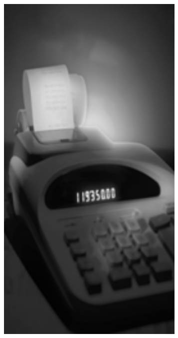
a	b	c	d		o2	o1	o0
0	0	0	0		0	0	0
1	0	0	0		0	0	1
0	1	0	0		0	1	0
0	0	1	0		0	1	1
0	0	0	1		1	0	0

- $o2 = \overline{abc}d$
- $o1 = \bar{a}b\overline{cd} + \overline{ab}c\bar{d}$
- $o0 = a\overline{bcd} + \overline{ab}c\bar{d}$

Ballot Reading



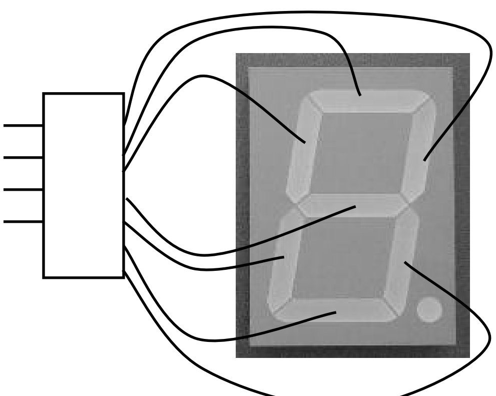
Ballots



The 3410 voting machine

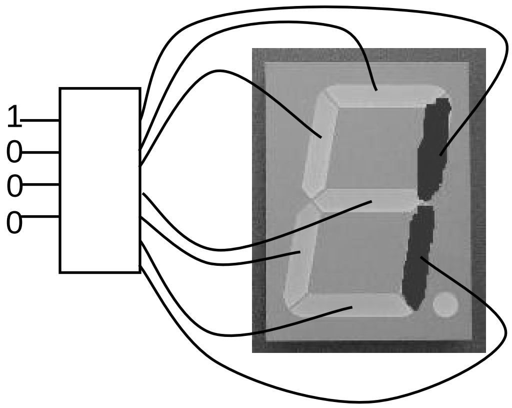
- Ok, we built first half of the machine
- Need to display the result

7-Segment LED Decoder



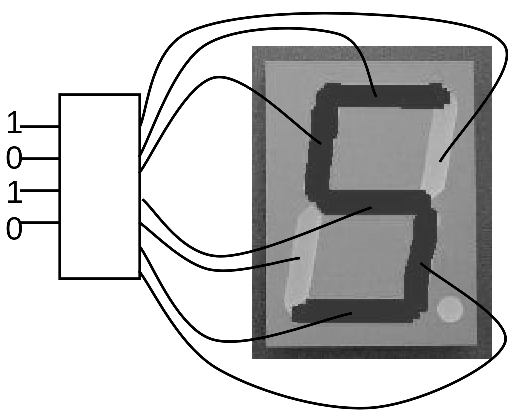
- 4 inputs encoded in binary
- 8 outputs, each driving an independent, rectangular LED
- Can display numbers
- Just a simple logic circuit
- Write the truth table

7-Segment LED Decoder



- 4 inputs encoded in binary
- 8 outputs, each driving an independent, rectangular LED
- Can display numbers

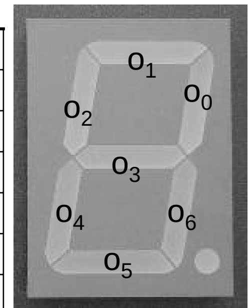
7-Segment LED Decoder



- 4 inputs encoded in binary
- 8 outputs, each driving an independent, rectangular LED
- Can display numbers

7-Segment Decoder Truth Table

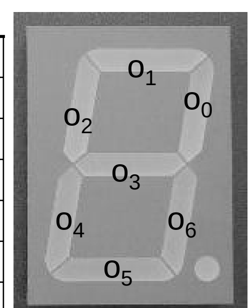
i_3	i_2	i_1	i_0		o_0	o_1	o_2	o_3	o_4	o_5	o_6
0	0	0	0		1	1	1	0	1	1	1
0	0	0	1		1	0	0	0	0	0	1
0	0	1	0		1	1	0	1	1	1	0
0	0	1	1		1	1	0	1	0	1	1
0	1	0	0		1	0	1	1	0	0	1
0	1	0	1		0	1	1	1	0	1	1
0	1	1	0		0	0	1	1	1	1	1
0	1	1	1		1	1	0	0	0	0	0
1	0	0	0		1	1	1	1	1	1	1
1	0	0	1		1	1	1	1	0	1	1



Exercise: find the error(s) in this truth table

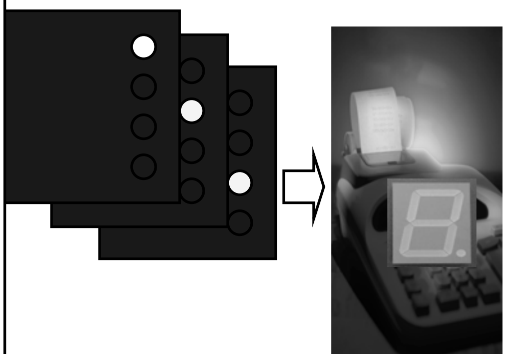
7-Segment Decoder Truth Table

i_3	i_2	i_1	i_0		o_0	o_1	o_2	o_3	o_4	o_5	o_6
0	0	0	0		1	1	1	0	1	1	1
0	0	0	1		1	0	0	0	0	0	1
0	0	1	0		1	1	0	1	1	1	0
0	0	1	1		1	1	0	1	0	1	1
0	1	0	0		1	0	1	1	0	0	1
0	1	0	1		0	1	1	1	0	1	1
0	1	1	0		0	0	1	1	1	1	1
0	1	1	1		1	1	0	0	0	0	1
1	0	0	0		1	1	1	1	1	1	1
1	0	0	1		1	1	1	1	0	1	1



Ballot Reading

- Done!



Ballots

The 3410 voting machine

Summary

- We can now implement any logic circuit
  - Can do it efficiently, using Karnaugh maps to find the minimal terms required
  - Can use either NAND or NOR gates to implement the logic circuit
  - Can use P- and N-transistors to implement NAND or NOR gates