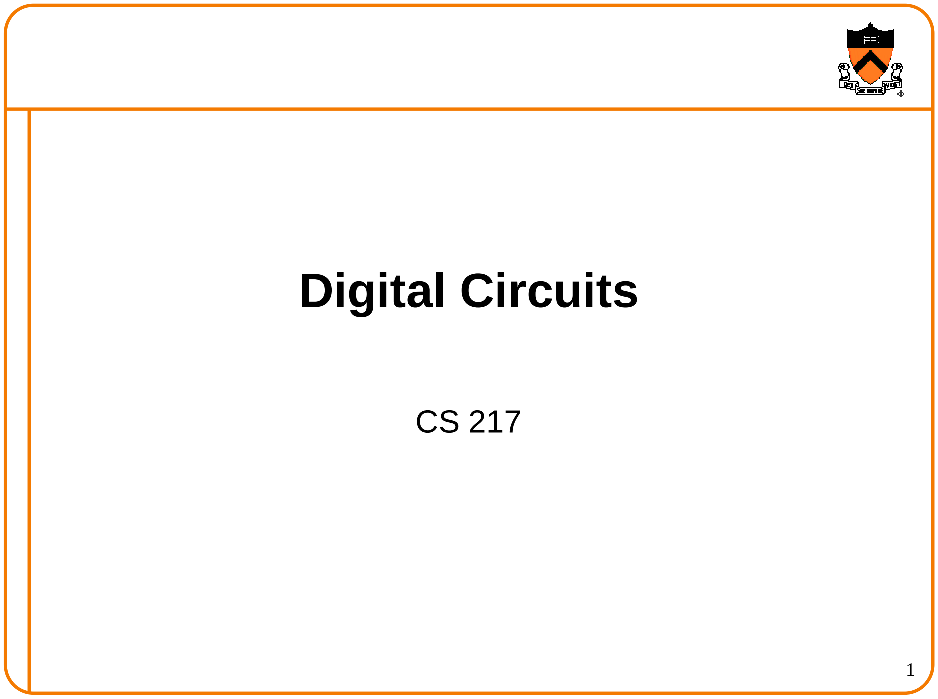
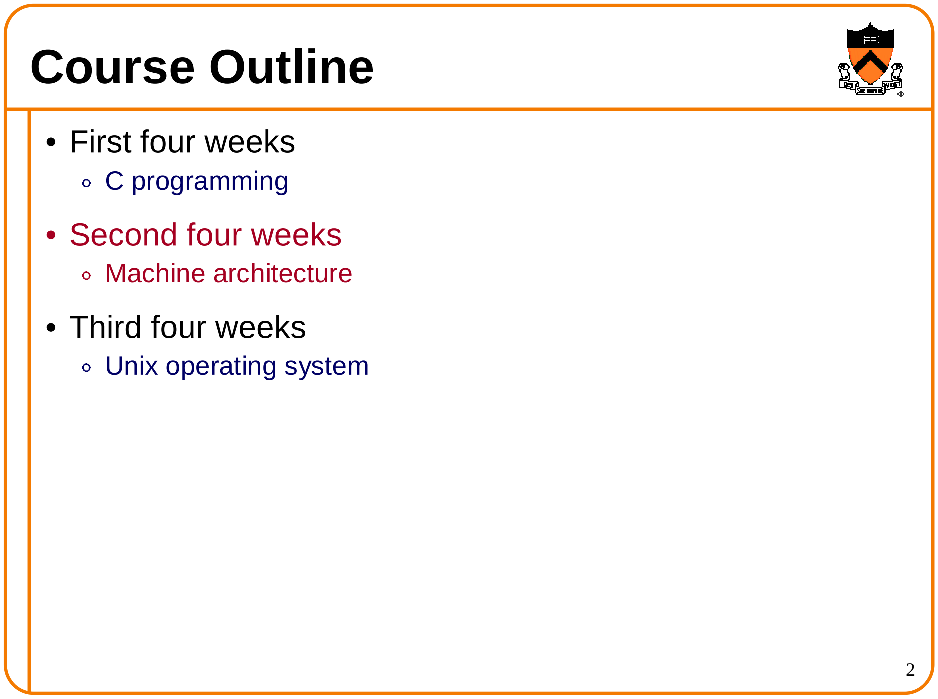


Digital Circuits

CS 217

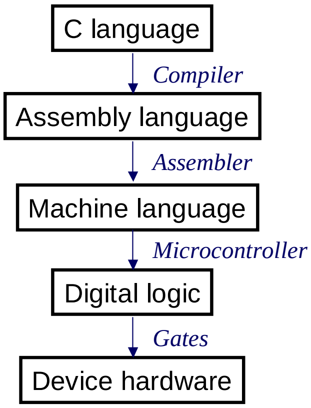


Course Outline

- First four weeks
  - C programming
- Second four weeks
  - Machine architecture
- Third four weeks
  - Unix operating system

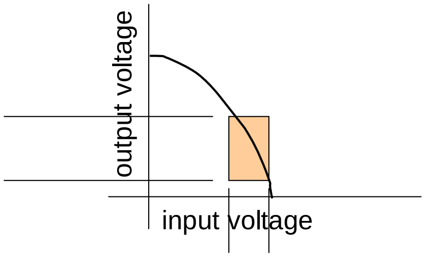
Levels of Abstraction

- We have been looking at programming at high level
  - Abstractions provided by C
- Let's now look under the hood
  - How does C program actually execute?
  - Start from bottom (device hardware) and work our way up



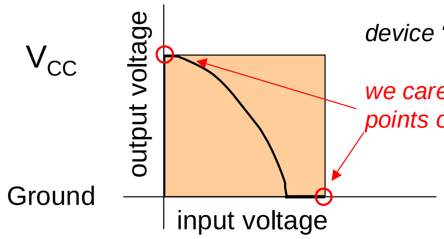
Analog circuits

- Components: resistors, inductors, capacitors, transistors ...
- Voltage, current are continuous functions of time
  - and of d/dt of current, voltage...
- Build: amplifiers, radios ...
- Typical device characteristic:



Digital circuits

- Components: transistors, transistors, transistors ...
  - (and the occasional capacitor)
- Pick two voltages of interest: “V_{CC}” and “Ground”
- Build: clocks, adders, computers, computers, computers...
  - “computers” includes: cell phone, Nintendo, cash register, ...
- Typical device characteristic:

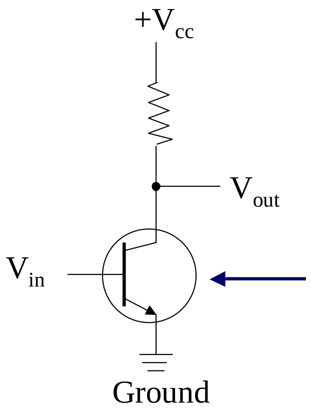


device “saturated” ---

we care only about these two points on the graph

Digital Circuits

- Wires, voltage, resistors, ground, etc.



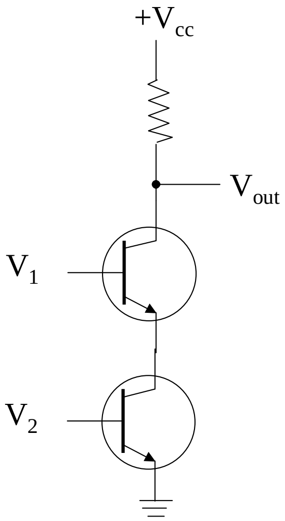
V_{in}	V_{out}
0	V_{cc}
V_{cc}	0

This is an NOT gate! (an inverter)

Transistor (electrical switch)

Electricity flows through iff V_{in} exceeds critical value (V_C)

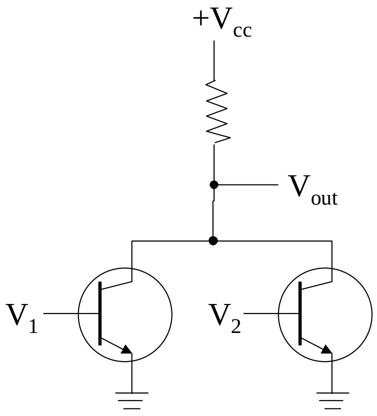
Digital Circuits



V_1	V_2	V_{out}
0	0	V_{cc}
0	V_{cc}	V_{cc}
V_{cc}	0	V_{cc}
V_{cc}	V_{cc}	0

This is a NAND gate!

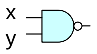
Digital Circuits



V_1	V_2	V_{out}
0	0	V_{cc}
0	V_{cc}	0
V_{cc}	0	0
V_{cc}	V_{cc}	0

This is a NOR gate!

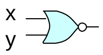
Gates



NAND gate

x	y	x & y
0	0	1
0	1	1
1	0	1
1	1	0

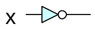
also written: $\overline{xy}$



NOR gate

x	y	x | y
0	0	1
0	1	0
1	0	0
1	1	0

also written: $\overline{x+y}$

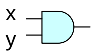


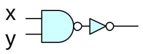
NOT gate

x	~x
0	1
1	0

also written: $\overline{x}$, ¬x

Gates

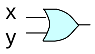


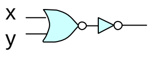


AND gate

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

also written: xy





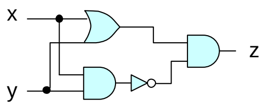
OR gate

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

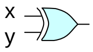
also written: x+y

Circuits

- Build higher-level boolean logic out of gates

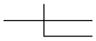


x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

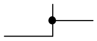


XOR gate

x XOR y = (x+y) & ¬(x&y)



wires crossing
(not connected)

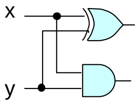


wire junction
(connection)

Adder (for 1-bit binary numbers)

x	y	add(x,y)
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

carry sum



sum = x XOR y

carry = x&y

N-bit binary adder

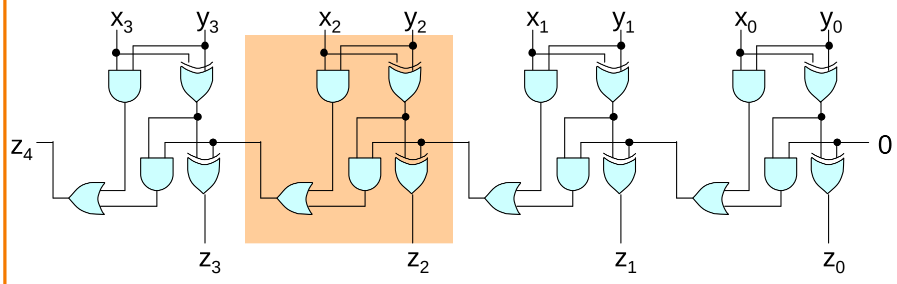
$x_3\ x_2\ x_1\ x_0$

$+\ \ y_3\ y_2\ y_1\ y_0$

$z_4\ z_3\ z_2\ z_1\ z_0$

$\text{sum}_i = x_i \text{ XOR } y_i \text{ XOR carry}_{i-1}$

$\text{carry}_i = (x_i \ \&\ y_i) + ((x_i \text{ XOR } y_i) \ \&\ \text{carry}_{i-1})$



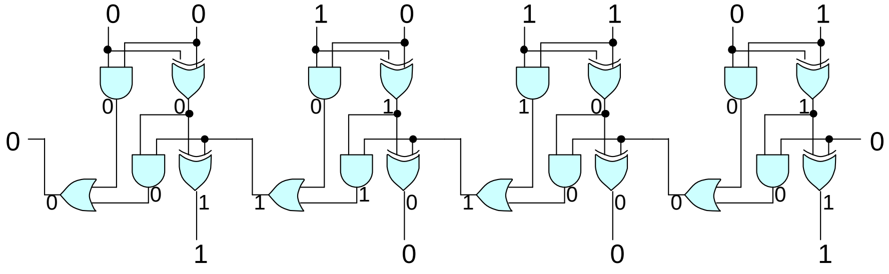
N-bit binary adder

$x_3\ x_2\ x_1\ x_0$

$+\ \ y_3\ y_2\ y_1\ y_0$

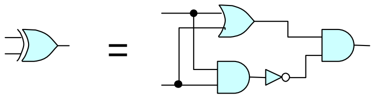
$z_4\ z_3\ z_2\ z_1\ z_0$

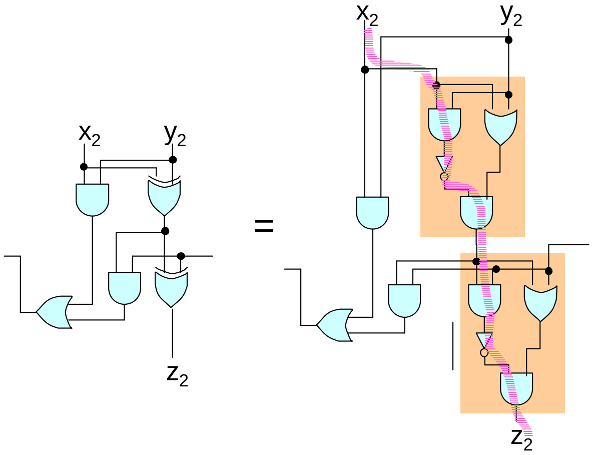
0 1 1 0	6
+ 0 0 1 1	+ 3
0 1 0 0 1	9



“Seat of the pants” design

- You just saw it!
- Can be inefficient:

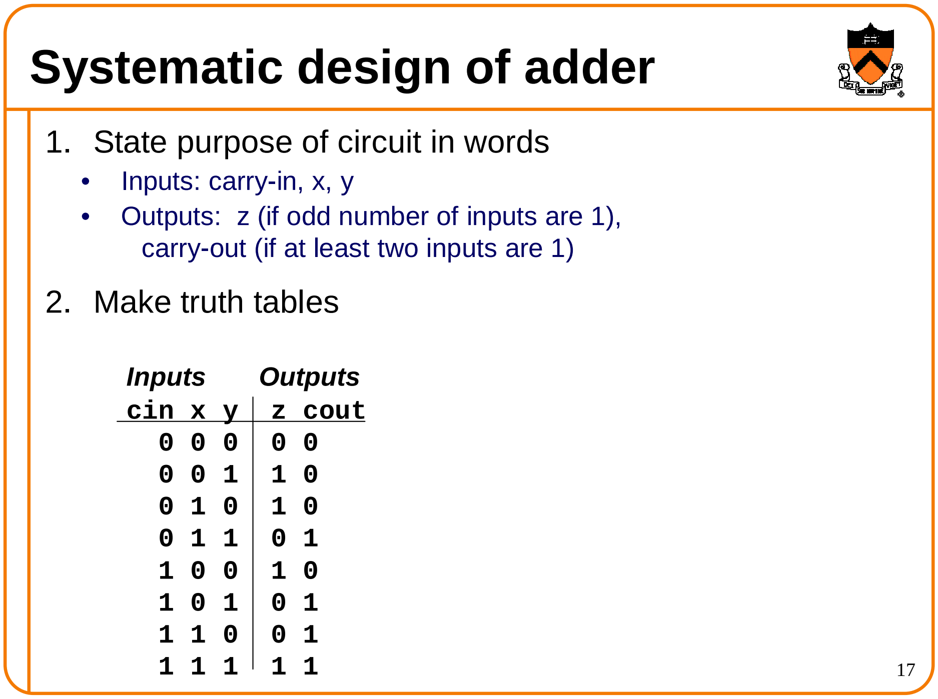




longest path goes through 6 gates; that’s slow

Systematic design

1. State purpose of circuit in words
2. Make truth tables
3. Identify “true” rows
4. Construct sum-of-products expression
5. Construct circuit

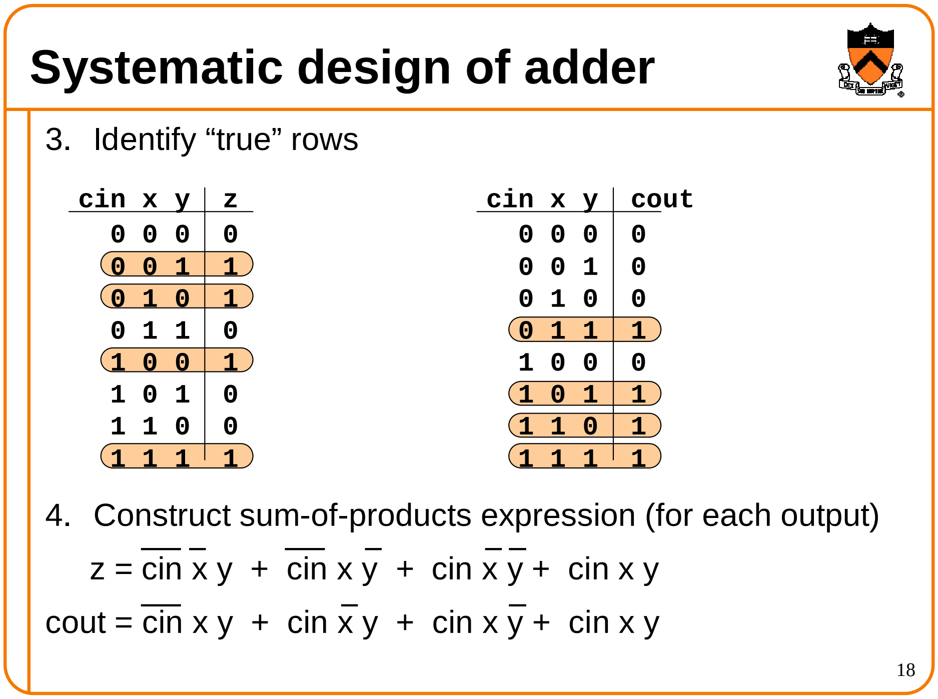


Systematic design of adder

1. State purpose of circuit in words
   - Inputs: carry-in, x, y
   - Outputs: z (if odd number of inputs are 1), carry-out (if at least two inputs are 1)
2. Make truth tables

Inputs			*Outputs*	
cin	**x**	**y**	**z**	**cout**
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

17



Systematic design of adder

3. Identify "true" rows

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

cin	x	y	cout
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

4. Construct sum-of-products expression (for each output)

$$z = \overline{cin}\ \overline{x}\ y + \overline{cin}\ x\ \overline{y} + cin\ \overline{x}\ \overline{y} + cin\ x\ y$$

$$cout = \overline{cin}\ x\ y + cin\ \overline{x}\ y + cin\ x\ \overline{y} + cin\ x\ y$$

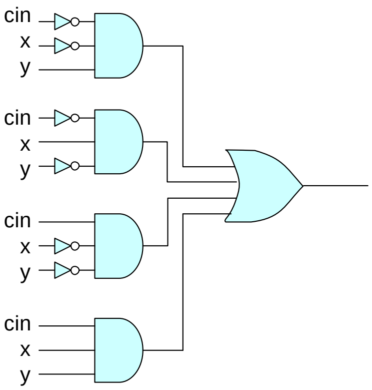
18

Systematic design of adder

5. Construct circuit

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

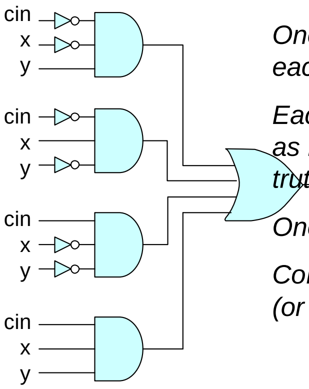
$$z = \overline{cin}\,\overline{x}\,y + \overline{cin}\,x\,\overline{y} + cin\,\overline{x}\,\overline{y} + cin\,x\,y$$



Sum-of-products circuit

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$z = \overline{cin}\,\overline{x}\,y + \overline{cin}\,x\,\overline{y} + cin\,\overline{x}\,\overline{y} + cin\,x\,y$$



One AND-gate for each 1-output in table

Each AND-gate has as many inputs as truth table

One OR-gate

Constant-depth: 2 (or 3, counting NOTs)

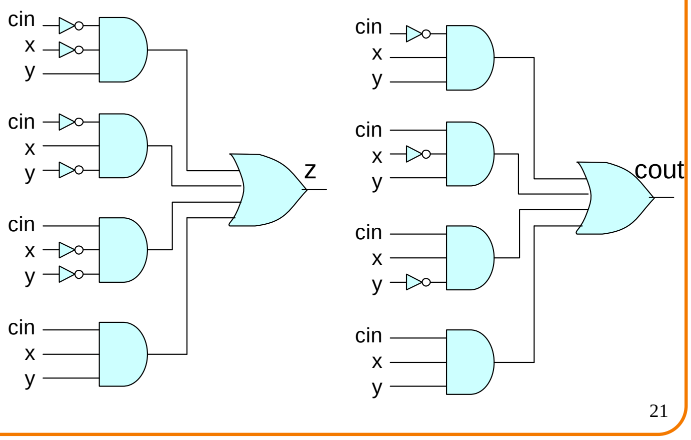
Finishing the adder

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

cin	x	y	cout
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

$$z = \overline{cin}\ \overline{x}\ y\ +\ \overline{cin}\ x\ \overline{y}\ +\ cin\ \overline{x}\ \overline{y} +\ cin\ x\ y$$

$$cout = \overline{cin}\ x\ y\ +\ cin\ \overline{x}\ y\ +\ cin\ x\ \overline{y} +\ cin\ x\ y$$



21

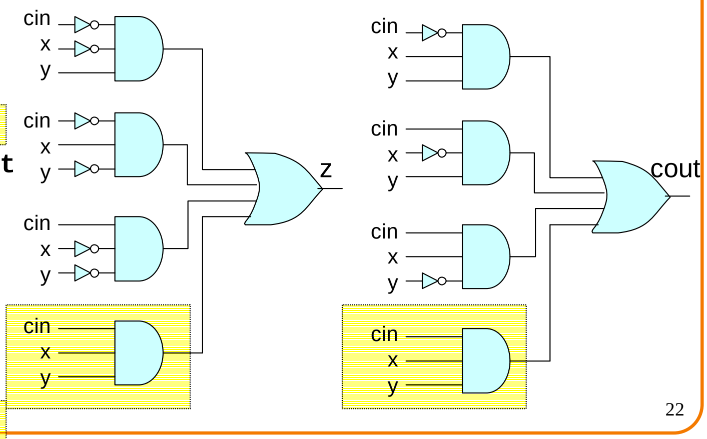
Duplicate terms, duplicate gates

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

cin	x	y	cout
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

$$z = \overline{cin}\ \overline{x}\ y\ +\ \overline{cin}\ x\ \overline{y}\ +\ cin\ \overline{x}\ \overline{y} +\ cin\ x\ y$$

$$cout = \overline{cin}\ x\ y\ +\ cin\ \overline{x}\ y\ +\ cin\ x\ \overline{y} +\ cin\ x\ y$$



22

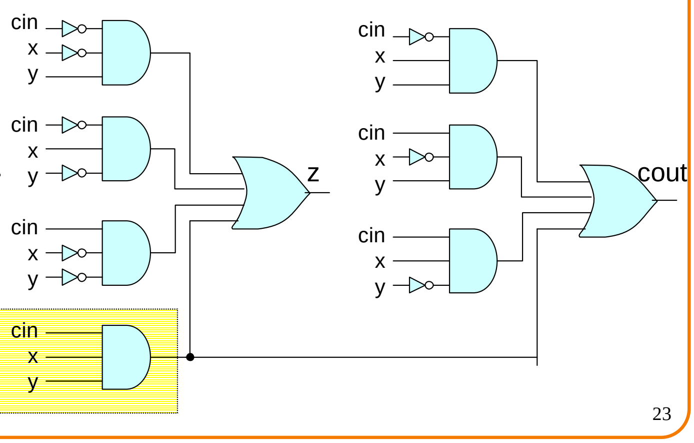
Duplicate terms, duplicate gates

cin	x	y	z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

cin	x	y	cout
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

$$z = \overline{cin}\;\overline{x}\;y \;+\; \overline{cin}\;x\;\overline{y} \;+\; cin\;\overline{x}\;\overline{y} + cin\;x\;y$$

$$cout = \overline{cin}\;x\;y \;+\; cin\;\overline{x}\;y \;+\; cin\;x\;\overline{y} + cin\;x\;y$$



Metrics

- With *N* inputs, *M* outputs in truth table (and in circuit)

  2^N rows in table

  Each AND gate has *N* inputs

  At most 2^N AND gates total

  M OR gates

  Each OR gate has at most 2^N inputs

***N* Inputs**			***M* Outputs**	
cin	**x**	**y**	**z**	**cout**
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

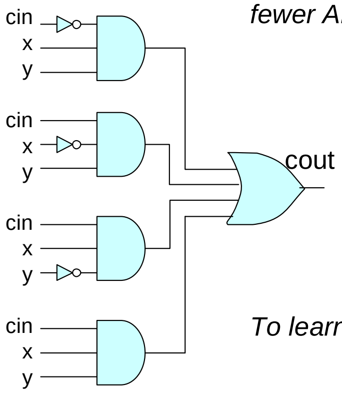
2^N rows

Advanced stuff

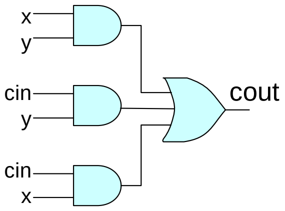
$$cout = \overline{cin}\ x\ y\ +\ cin\ \bar{x}\ y\ +\ cin\ x\ \bar{y} +\ cin\ x\ y$$

$$cout = x\ y + cin\ y + cin\ x$$

cin	x	y	cout
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



Sometimes you can get by with fewer AND gates

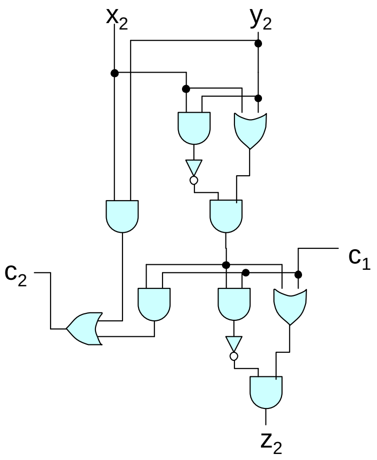


To learn how, take ELE 206!

Circuit analysis

- What does this circuit do?

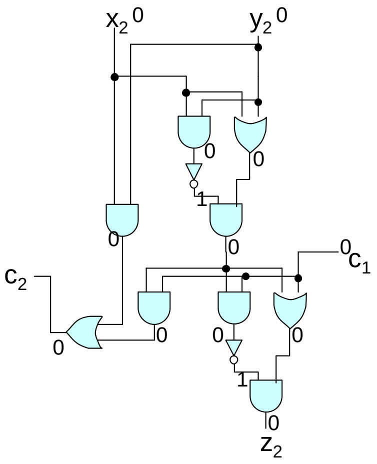
  (pretend you haven't seen it already)



Circuit analysis

1. Draw the truth table by “simulating” gates

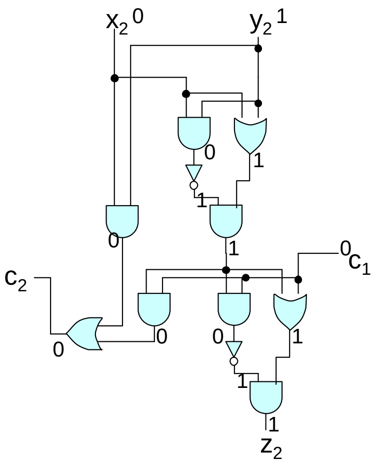
c1	x	y	z	c2
0	0	0	0	0

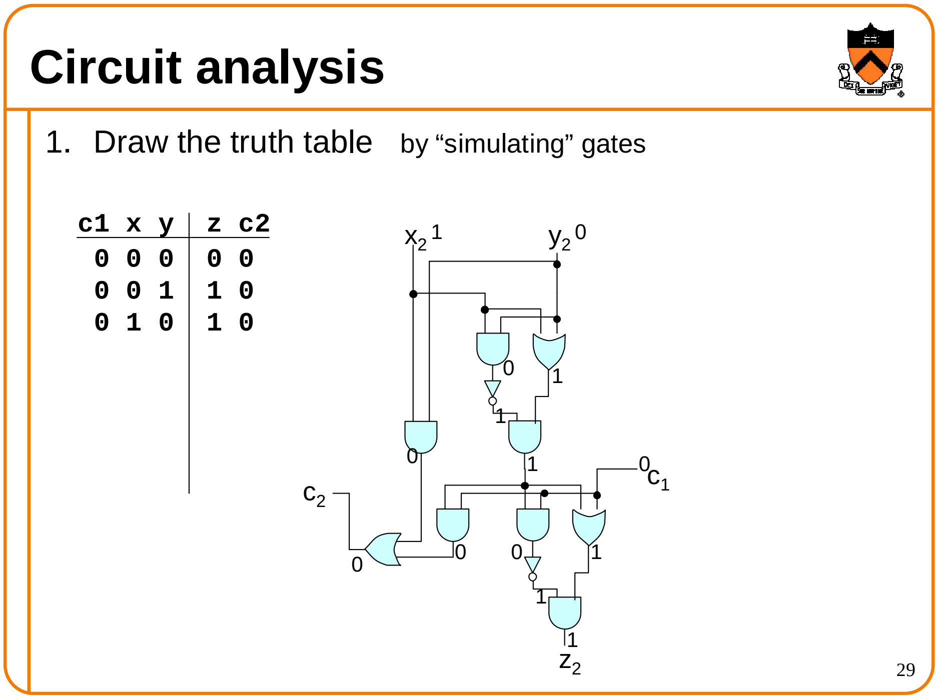


Circuit analysis

1. Draw the truth table by “simulating” gates

c1	x	y	z	c2
0	0	0	0	0
0	0	1	1	0



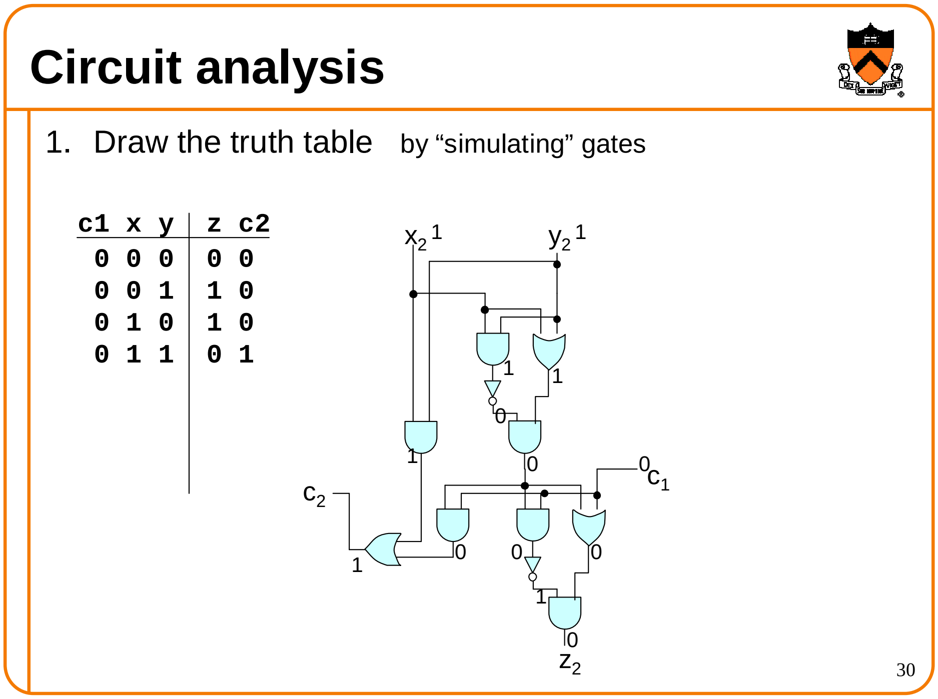


Circuit analysis

1. Draw the truth table by “simulating” gates

c1	x	y	z	c2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

29



Circuit analysis

1. Draw the truth table by “simulating” gates

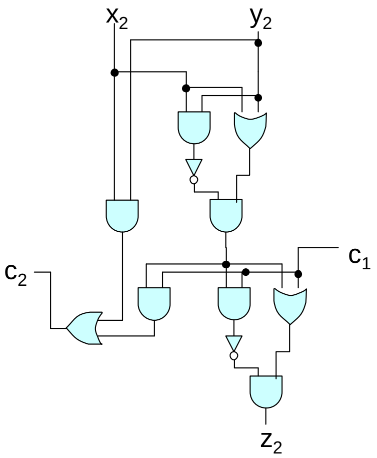
c1	x	y	z	c2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1

30

Circuit analysis

1. Draw the truth table

c1	x	y	z	c2
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



Circuit analysis

2. Say in words what the truth table does

c1	x	y	z	c2
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

z is 1 if an odd number of inputs are 1

c2 is 1 if at least two inputs are 1

Circuit analysis

3. Apply a flash of insight

c1	x	y	z	c2
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

z is 1 if an odd number of inputs are 1

c2 is 1 if at least two inputs are 1

Aha! It's one bit-slice of an adder!

Summary

- Digital Circuits
  - Boolean logic
  - Combinatorial circuits
- Next lectures
  - Sequential circuits
  - Building a computer