Circuits

Part 11

1

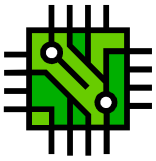
Circuits

Boolean Hardware

2

Circuits

- Boolean algebra gives designers tools to design & analyze complex solutions
- Can we create hardware that performs solutions?
- Can electronic circuits allow Boolean logic to exist in the physical world?



3

Two Bit Multiplier? Can we make it?

4

Designing It

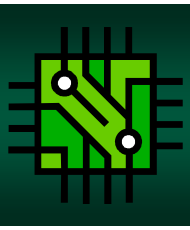
- To design a circuit that multiplies two 2-bit numbers, we can use Boolean algebra
- We need to figure the logic – given that bits of 1 and 0 will map directly to truth values
- The result of the algebra will be the desired output

5

It Takes the Following Skills

1. Design a truth-table to represent the different inputs and the desired output
2. Convert the truth-table into a Boolean function
3. Simplify the Boolean function
4. Finally, convert it into a circuit

6



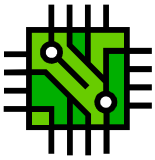
Gates

Boolean Hardware

7

Gates

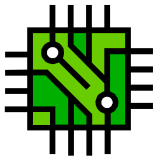
- Electronic devices are made up of *gates*
- Gates take in two inputs and produce a single output
- This is how hardware is used to implement Boolean logic (or any logic)



8

Gates

- Gates can be combined into circuits with *any number of input wires* and a single output wire
- We will chain these together much like subexpressions can be combined into larger expressions



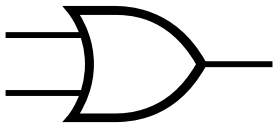
9

Graphical Representation

- Gates are typically represented using graphical shapes – much like flowcharts
- There are two different competing symbol standards
- We will use the standard, distinct, symbols rather than the IEC (European) ones

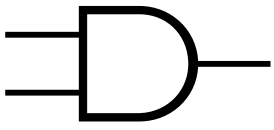
10

Or Gate



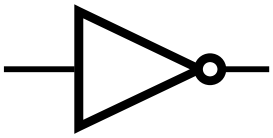
11

And Gate



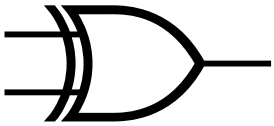
12

Not Gate



13

Exclusive Or Gate (aka XOR)



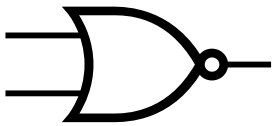
14

Some Other Gate Symbols

- There are also gate symbols for negated operators
- I won't use these much in class, but it's good to be aware of them (since they are quite common in computer engineering
- For each, note the circle on the output line – it means "not"

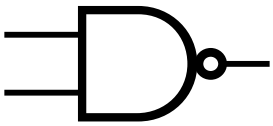
15

Not Or Gate (aka NOR)



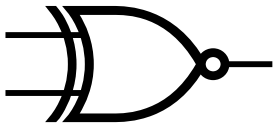
16

Not And Gate (aka NAND)



17

Not Exclusive Or Gate (aka XNOR)



18

Computer Engineering Notation

Same thing, different paint job

19

Computer Engineering Notation

- Gates, used to computer engineering, form a Boolean Algebra
- i.e. they can define the three operations (and, or, not) and share the same axioms

20

Computer Engineering Notation

- The notation used in computer engineering is a *tad different* that what we have covered
- … and certainly different than any programming language you have used

21

Computer Engineering Notation

- But is serves the same purpose
- And, not surprisingly, it works better for writing expressions in this discipline

22

Computer Engineering Notation

1 ≡ T

0 ≡ F

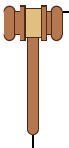
* ≡ ∧

\+ ≡ ∨

' ≡ ¬

23

Commutative Law

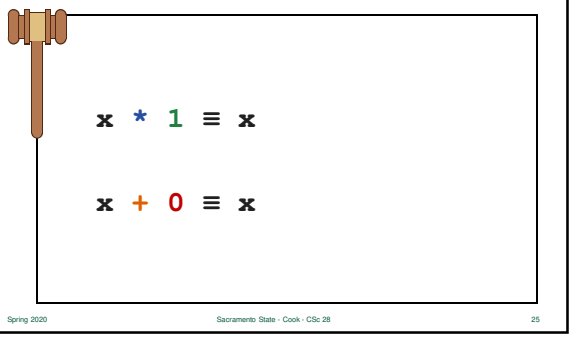


x + y ≡ y + x

x * y ≡ y * x

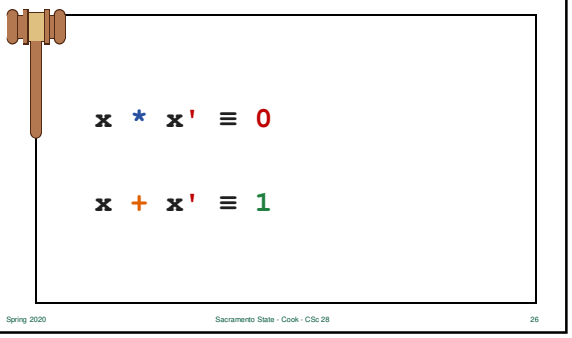
24

Identity Law



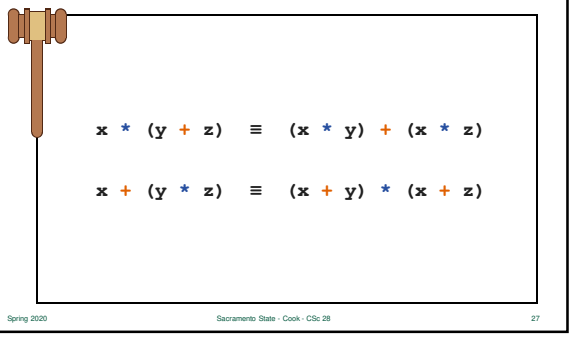
25

Complement Law



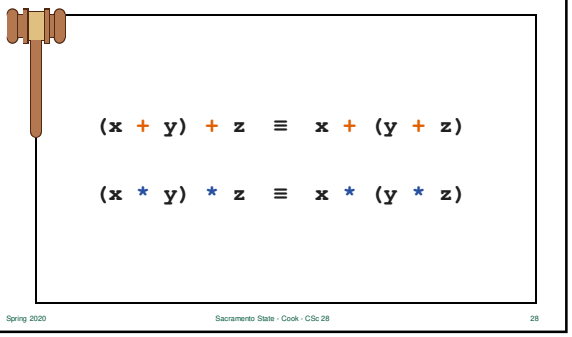
26

Distributive Law



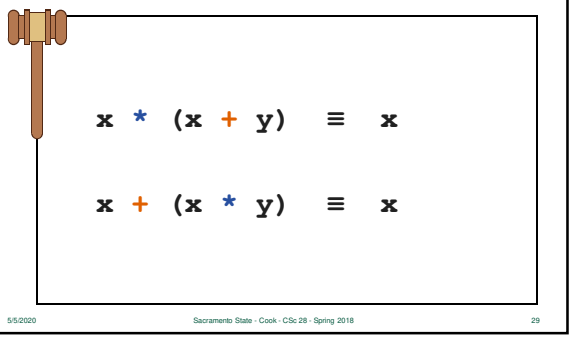
27

Associative Law



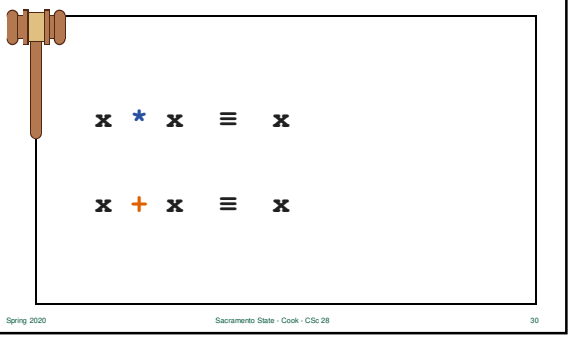
28

Absorption Law



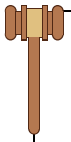
29

Idempotent Law



30

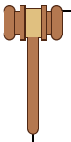
Involution Law



x'' ≡ x

31

Domination Law

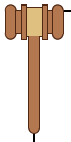


x + 1 ≡ 1

x * 0 ≡ 0

32

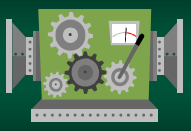
DeMorgan's Law



(x * y)' ≡ x' + y'

(x + y)' ≡ x' * y'

33



Converting Boolean to Circuits

From Logic to Wires

34

Converting Boolean to Circuits

- Converting from Boolean to circuits maintains a *one-to-one* correspondence between gates and operators in the equation
- But, given an *arbitrary* Boolean expression, how do we realize a circuit for it?

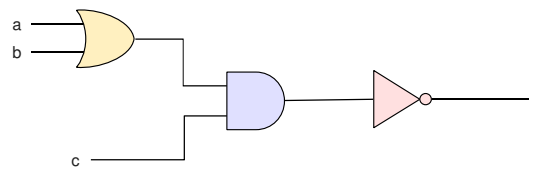
35

Steps

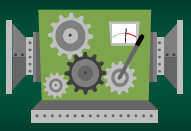
1. Choose the last operation evaluated
2. Draw a gate and hook up its output
3. Goto 1 until all operations have associated gates
4. Attach the expression inputs

36

((a + b) * c)'



37



Converting Circuits to Boolean

From Logic to Wires

38

Converting Circuits to Boolean

- The other direction is easy too
- Any circuit can be realized as a Boolean expression using the same basic algorithm

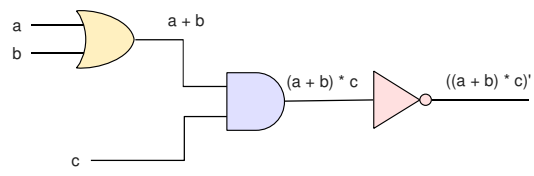
39

Converting Circuits to Boolean

1. Pick a wire that has known Boolean values
2. Write *on the wire* a Boolean expression for its value
3. Goto 1 until all wires are complete
4. Circuit's expression written on the circuit's output wire

40

Example Circuit



41