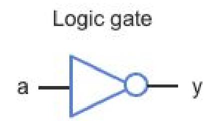
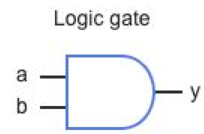
1.2 Gates

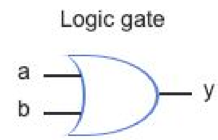
- **Inverter -** Not Gate: output1 if input is 0, vice versa



- AND Gate: output 1 if both gates' inputs are 1



- NOR Gate: output 1 if either or both of the gate's inputs is 1



- pMOS conducts when input is 0
- nMOPS does NOT conduct when input is 0

Inputs	0	1
pMOS	1	Not conduct
nMOS	Not conduct	0

1.3 Boolean Algebra and Equations

- Consider an event as a variable, and conduct logic operators to see what the result is
- Example
    - Input a: …
    - Input b: …
    - Output c: …
    - Indicate whether some equations match the human logic

- Boolean Equation
    - Boolean variable on the **left** = a boolean expression on the **right**
    - **Shorthand**
        -

a AND b	**ab**	**abutment**
a OR b	**a + b**	**addition**
NOT(a)	**a'**	**complement**

- Boolean functions

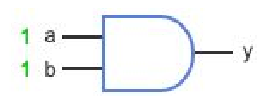
- A relation of inputs' values to an output's values
- Y = ab (y = a AND b)
- Note
    - output y is 1 if both inputs are 0's; otherwise, y is 0
    - Also if both inputs are 1's, y is 1
    - **This is not a function**
    - **A function defines exactly one output value for unique input values**

    - Output y is 1 if both inputs are 0's
    - **This is not a function**
    - **A function must include all input possibilities (do not know what y should be for other input values like 0, 1)**

1.4 Digital Circuit simulator

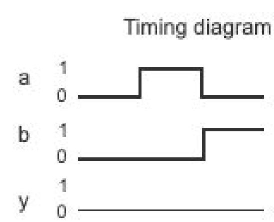
- Instantiating: adding an item to a circuit, such item is called an **instance**
- Exporting transfers the current circuit diagram into text
- Importing a circuit diagram using the text exported before

1.5 Timing Diagrams

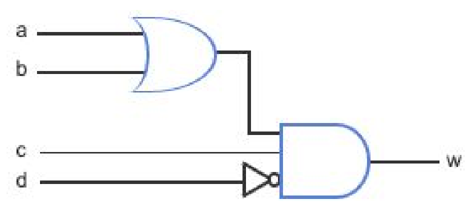


- Def: graphically shows a circuit's output values for given input values that change over time

- Each signal name is listed on the left
- Time proceeds to the right



1.6 Equations to/from circuits



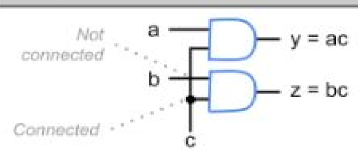
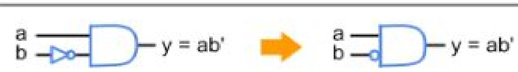
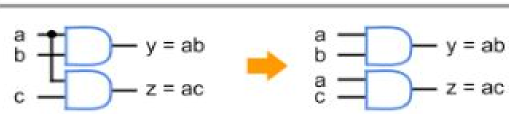
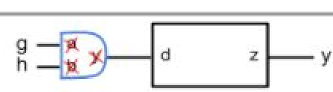
- Design: converting a behavior to a circuit
- W = (a + b)cd'
- Example
    - Airbags deployed
    - Airbags can harm kids, and aren't needed for non-human objects. Thus, cars have sensors to help detect whether an airbag should be enabled by a large enough human being seated.

    - In one car, a seat back sensor detects a heartbeat (h = 1). A seat bottom sensor indicates if over 60 pounds is detected (w = 1). A switch can be used to manually disable the airbags (d = 1). An output e indicates that the airbag is enabled (e = 1).

  - A designer specifies the system as: e = hwd' (heartbeat detected, and enabled if weight over 60, and not manually disabled).

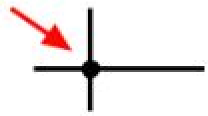
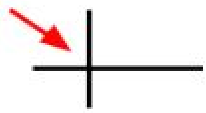
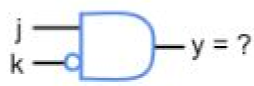
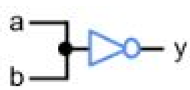
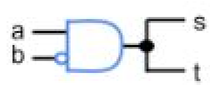
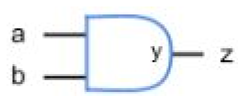
- Analysis: converting a circuit to behavior
- **Combinational circuit (logic)**: a circuit whose output value is determined solely by the present combination of input values
  - Example
    - If one or more people are in the room, turn on the light
    - If the weight in an elevator exceeds 1500 pounds, sound an alarm
    - Turn the faucet on while motion is detected
- **Sequential circuit**: a circuit whose output value may depend on the **past** sequence of input values
  - Example
    - If the sleep mode button is pressed and released, turn the phone ringer off until the sleep mode button is pressed again
    - If the garage door button is pressed, turn motor on until garage door is fully opened

1.7 Basic circuit drawing conventions

Convention	Illustration
Crossing wires only connect at filled circle.	
Inverter often drawn as circle at gate input.	
One output wire splitting is OK. Multiple input wires connecting is not.	
Input labels are sometimes duplicated to simplify a drawing.	
Gate inputs are not labeled (other components may be).	

-

-

Wires connected		True
Wires connected		False
Gate output		jk’
Wiring is OK		False
Wiring is OK		True
Label y is normally used		False

-

1.8 Basic properties of Boolean algebra

Property	Name	Description
a(b + c) = ab + ac	***Distributive (for AND)***	Same as multiplication in regular algebra
a + a' = 1	***Complement***	Clearly one of a, a' must be 1 1 + 0 = 1 0 + 1 = 1
a · 1 = a	***Identity***	Result of a · 1 is always a's value *0* · 1 = *0* *1* · 1 = *1*

-
- Example
    - Sound alarm if person up and button pressed, or person up and button not pressed

- Inputs
    - u: person up from bed
    - n: nurse call button pressed
- Outputs
    - s: sound alarm
- Simplification
    - s = un + un’
    - s = u(n + n’) distributive
    - s = u(1) complement
    - s = u identity

Property	Name	Description
ab = ba	***Commutative (for AND)***	Same as multiplication for regular algebra
a + b = b + a	***Commutative (for OR)***	Same as addition for regular algebra
a + 1 = 1	***Null elements***	OR only needs one 1 to evaluate to 1 a = 0 0 + 1 = 1 a = 1 1 + 1 = 1
a + a = a aa = a	***Idempotent***	0 + 0 = 0 1 + 1 = 1 0 · 0 = 0 1 · 1 = 1

-

(e + 1)(e'f + fe' + d')	Original expression Start with the original expression
(1)(e'f + fe' + d')	Null elements (a + 1) = (1)
e'f + fe' + d'	Identity (1) (...) = ...
e'f + e'f + d'	Commutative (for AND) ab = ba (so fe' = e'f)
e'f + d'	Idempotent a + a = a

-
- Summary

-

Property	Name	Description
a(b + c) = ab + ac a + (bc) = (a + b)(a + c)	Distributive (AND) Distributive (OR)	(AND) Same as multiplication in regular algebra (OR) Not at all like regular algebra
ab = ba a + b = b + a	Commutative	Variable order does not matter. Good practice is to sort variables alphabetically.
(ab)c = a(bc) (a + b) + c = a + (b + c)	Associative	Same as regular algebra
aa' = 0 a + a' = 1	Complement (AND) Complement (OR)	(AND) Clearly one of a, a' must be 0 1 · 0 = 0 · 1 = 0 (OR) Clearly one of a, a' must be 1 1 + 0 = 0 + 1 = 1
a · 1 = a a + 0 = a	Identity (AND) Identity (OR)	(AND) Result of a · 1 is always a's value *0 · 1 = 0 1 · 1 = 1* (OR) Result of a + 0 is always a's value *0 + 0 = 0 1 + 0 = 1*
a · 0 = 0 a + 1 = 1	Null elements	Result doesn't depend on the value of a.
a · a = a a + a = a	Idempotent	Duplicate values can be removed.
(a')' = a	Involution	(0')' = (1)' = 0 (1')' = (0)' = 1
(ab)' = a' + b' (a + b)' = a'b'	DeMorgan's Law	*Discussed in another section*

1.9 Sum-of-products form

- Product term: an ANDing of one or more variables
- Sum of products form consists only of an ORing of product terms
- Example: ab'c + ab
- Note
    - A single product is still considered a sum of products
    - a(b + c) (*ANDed*) is **not** considered as a sum of product, but ab + ac is

- Converting sum of products to a circuit
    - AND gates followed by OR gates
    - Before creating a circuit, convert to sum of products form

1.10 Binary and counting

- Literally teaching you how to count in binary

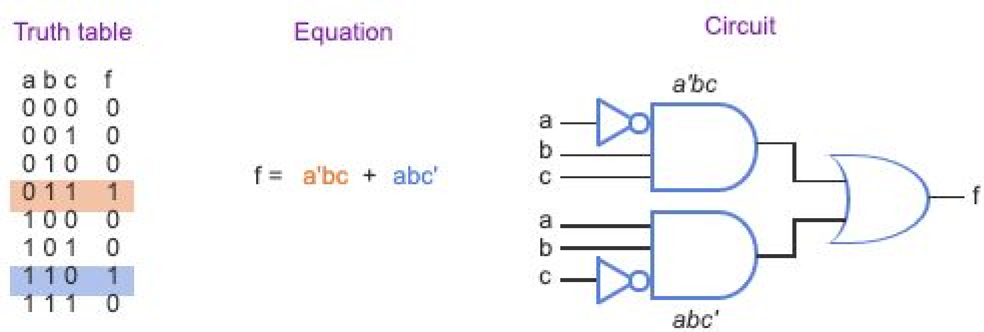
1.11 Sum-of-miniterms form

- Canonical form: standard equation form for a function
- Sum of mini terms: canonical form of a Boolean equation where the right side expression is a sum of products with each product a unique miniterm
- Miniterm: a product term having only **one** literal for **every function variable**
- Literal: variable appearance, in true or complemented form

- Transforming to sum of miniterms
    - For any missing variable v, create miniterm by multiplying (v + v’)
    - Then remove redundant miniterms
    - Example
        - Y = a(b + bc’)
        - Y = ab + abc’
        - Y = ab(c + c’) + abc’
        - Y = abc + abc’ + ~~abc’~~
        - Y = abc + abc’

- Determine if two equations represent the same function by checking if they have the same sum of miniterm form

1.12 Truth tables

- Converting a truth table to an equation
- 

- Only care about the rows with output value of 1
- Such conversion is trivial, need no much thought and tradeoffs

- Capturing behavior as a truth table
    - When inputs are too many, it is better to try to capture through equation
    - The functionality of equation is the same as using a truth table, in terms of capturing the behavoir

1.13 Top-down design + Examples

- Design process:
    - 1. Capture: the task of precisely describing a circuit's desired behavior
    - 2. Convert (implement): the task of translating captured behavior into a circuit, possibly involving simplification
    - More specific
        - Capture as truth table
        - Convert truth table as equation
        - Convert equation to circuit
        - Use the circuit

- Triple modular redundancy - TODO

1.14 Why digital design

- Embedded system
    - a computing system embedded within another device like an automobile, an electronic book reader or music player, a robot, a medical device, a home security system, and much more
    - embedded systems often use various combinations of microprocessors and custom digital circuits.

1.15 Multiple outputs

- Many combinational circuits have multiple outputs for the same inputs