EEL3701: Digital Logic & Computer Systems

Menu

- Introduction to Logic Design
- Informal Intro to Boolean Algebra
  - >Propositions
  - >Operators
  - >Truth Tables

EEL3701: Digital Logic & Computer Systems

Introduction to Logic Design

- Let's logically analyze a simple paragraph:

> When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

EEL3701: Digital Logic & Computer Systems

Logically Analyzing a Paragraph

When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

OUTPUT:

- The "output" of this paragraph is **"getting wet"**
- I get wet under either of two specified conditions
  - >It rains and I forget my umbrella
  - >The bathtub is filled and I stick my foot in it

EEL3701: Digital Logic & Computer Systems

Logically Analyzing a Paragraph

When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

INPUT:

- Each condition is made of of two "inputs"
  - >It rains and I forget my umbrella
    - – It rains
    - – Forget umbrella
  - >The bathtub is filled and I stick my foot in it
    - – Bathtub filled
    - – Put foot in it

EEL3701: Digital Logic & Computer Systems

Signal Abbreviations

When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

- Let's abbreviate the inputs and outputs
  - >Output:
    - – Wet
  - >Inputs:
    - – Rain (it rains)
    - – NoUm (forget umbrella)
    - – Fill (bathtub filled)
    - – FootIn (put foot in it)

EEL3701: Digital Logic & Computer Systems

Writing an Equation

When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

- Write an "equation" for the paragraph

$$\text{Wet} = (\text{Rain } \textbf{AND } \text{NoUm}) \textbf{ OR } (\text{Fill } \textbf{AND } \text{FootIn})$$

EEL3701: Digital Logic & Computer Systems

Logical Values

- Every logical variable (e.g., Rain, NoUm, ..., Wet) has two possible values
  - > True
  - > False
- The logical abbreviations are:
  - > True: T or 1
  - > False: F or 0

EEL3701: Digital Logic & Computer Systems

Logical **AND** Function

- The symbol for a logical AND operation is often one of the symbols used for multiplication: $\times$, $*$, $\bullet$, $\wedge$, or no symbol
  - >$Z = A \times B$
  - >$Z = A * B$
  - >$Z = A \bullet B$
  - >$Z = A \wedge B$
  - >$Z = AB$

AND Truth (Logic) Table

A	B	Z=AB
F	F	F
F	T	F
T	F	F
T	T	T

Graphical AND Symbol

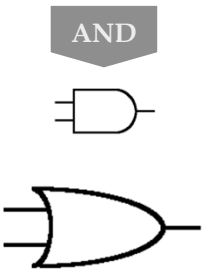
EEL3701: Digital Logic & Computer Systems

Logical **OR** Function

- The symbol for a logical OR operation is often the '+' sign or sometimes '∨',
  - >Z = A + B
  - >Z = A ∨ B

A	B	Z=A+B
F	F	F
F	T	T
T	F	T
T	T	T

OR Truth (Logic) Table

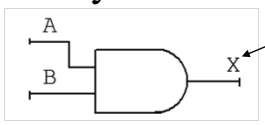


Graphical **OR** Symbol

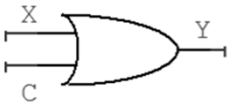
EEL3701: Digital Logic & Computer Systems

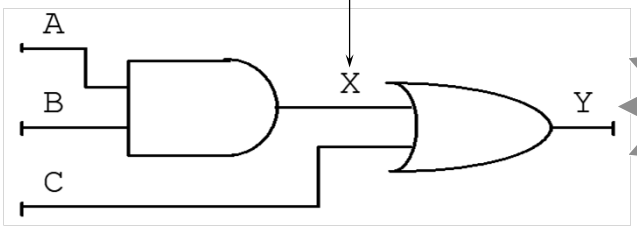
Logic Circuit

- Logic circuit is easily constructed from the logic equation
  - >Y = AB + C
  - >Let X = AB
  - >Y = X + C



X=AB



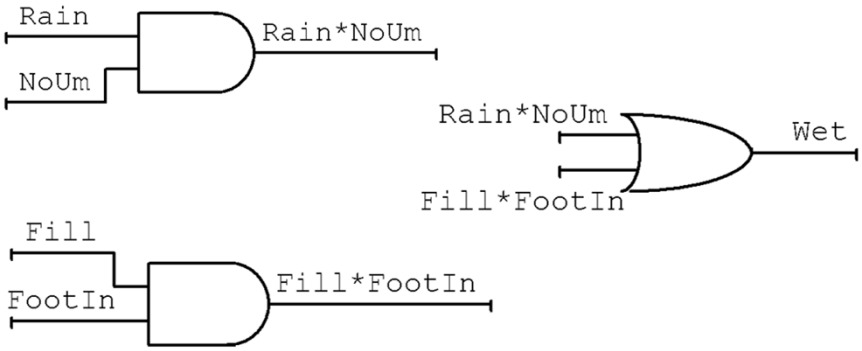


Ready for Wet?

EEL3701: Digital Logic & Computer Systems

Design Example: Get Wet!

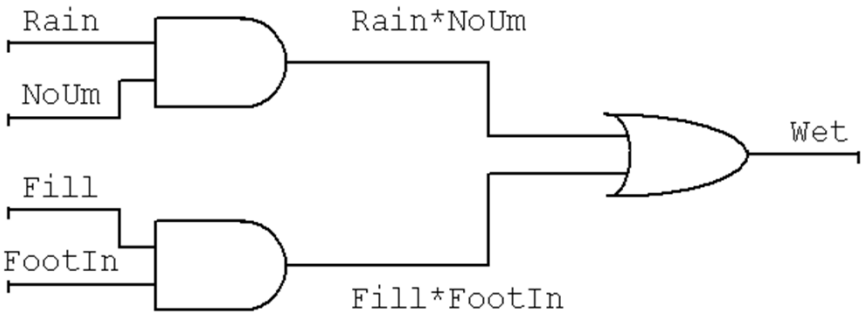
- Wet = (Rain ∗ NoUm) + (Fill ∗ FootIn)



EEL3701: Digital Logic & Computer Systems

Design Example: Get Wet!

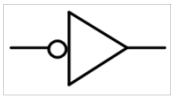
- Wet = (Rain ∗ NoUm) + (Fill ∗ FootIn)



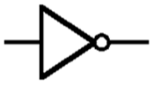
EEL3701: Digital Logic & Computer Systems

Logical **Level Shift** Function

- The symbol for an **INVERTER** operation (also called a **NOT** operator) is one of the following:
  - > Y = /A
  - > Y = $\overline{A}$
  - > Y= A'
  - > Y= ~A



or



NOT Truth (Logic) Table

A	Y=/A
F	T
T	F

Graphical NOT Symbol

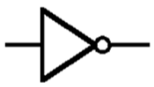
EEL3701: Digital Logic & Computer Systems

Design Example: Stay Dry!

- We can write a whole new paragraph based on the equation **Dry =/Wet**

OLD: When I forget my umbrella and it rains, I get wet. I also get wet when I stick my foot in the bathtub when it has already been filled.

NEW: I stay dry if I do not get wet!

Wet(H)  Dry(H)

EEL3701: Digital Logic & Computer Systems

Last Logical Example

- What is the equation for the below circuit?

Hint /TwoB

EEL3701: Digital Logic & Computer Systems

Getting Wet with I/O

- See live simulation with LogicWorks using with inputs and outputs (switches and LEDs)

`Get_Wet_action.cct`

EEL3701: Digital Logic & Computer Systems

Intro to Logic Design Summary

- Learned to convert a problem statement into a logic equation
- Learned the basic components of a digital circuit: AND, OR, NOT
- Learned to construct a digital circuit from a logic equation

EEL3701: Digital Logic & Computer Systems

Informal Intro to Boolean Algebra

- Motivation: **Boolean Algebra** is the ***calculus of logic***. It is also called ***Propositional Logic*** or the ***calculus of simple assertions***.
- Assertion 1: **Tacos are good.** This is T or F (Not both)
  - > We can express the assertion using symbols, e.g., X, Y, A, B
  - > Let X represent the assertion, then X=T or X=F
- Assertion 2: **Tacos are fatty.**
  - > Let Y represent assertion 2, then Y=T or Y=F
  - > We can make it more interesting by allowing combinations of assertions and negative sentences (assertions).

Ex: Let X=T and Y=F, then

- > "Tacos are **not** good" is false. (We write this as X' = F)
- > "Tacos are fatty" is false. (Y = F)
- > "Tacos are **not** fatty" is true. (Y'=F'=T)

❖**X: Tacos are good**
❖**Y: Tacos are fatty**

EEL3701: Digital Logic & Computer Systems

Informal Intro to Boolean Algebra

- Assertion 3: Z: Tacos are cheap. (let X=T, Y=F, Z=T)

  Now what does it mean to say:

  X and Z (also written as X * Z) T or F ?
  X or Z (also written as X + Z) T or F ?
  X and Y (also written as X * Y) T or F ?
  X or Y (also written as X +Y) T or F ?

❖**X: Tacos are good**
❖**Y: Tacos are fatty**
❖**Z: Tacos are cheap**

The combining of assertions using ***AND*** is called a **conjunction**, written as XY, X*Y, X∧Y or X•Y, and it is T iff both assertions are T, else it is F.

The combining of assertions using ***OR*** is called a **disjunction,** written as X∨Y or X+Y, and it is F iff both assertions are F, else it is T.

The combining of assertions using ***EOR* (XOR)** is called an **exclusive disjunction (exclusive or)**, written as X⊕Y or X**:+:**Y, and it is false iff both assertions are either both F or both T, else it is true.

The combining of assertions using ***EQUIV*** is called an **exclusive conjunction**, **(equivalence)**, written as X⊗Y or X**:*:**Y, and it is true iff both assertions are either both F or both are T, else it is false. (This is the complement of the exclusive-OR.)

EEL3701: Digital Logic & Computer Systems

Activation-Level Notation and Truth Tables

To make the notation easier we assign symbols as follows:

{T,F} ={1,0} (= **{H,L} for positive logic**)

or

{T,F} ={1,0} (= **{L,H} for "n" logic**)

Since X, Y or Z can only be {T,F}, we could represent all possibilities exhaustively in a table, called a **Truth Table**

Ex: Represent X*Y and X+Y and X ⊕ Y in a Truth Table

X	Y	X*Y	X+Y	X⊕Y		X	Y	X*Y	X+Y	X⊕Y
F	F	F	F	F		0	0	0	0	0
F	T	F	T	T		0	1	0	1	1
T	F	F	T	T		1	0	0	1	1
T	T	T	T	F		1	1	1	1	0

EEL3701: Digital Logic & Computer Systems

History of Boolean Algebra

- Boolean Algebra is so named in honor of **George Boole** who developed the notation in his Ph.D. dissertation in 1847.
  - > **Claude Shannon** applied it to switching networks in 1939.
- Boolean Algebra is the basic mathematics required for the study of the design of digital systems — also called switching networks.
- A switching device is (usually) a two state (binary) device.
  - > We represent the two states of these devices by the symbols {0,1} or {F,T} or {L,H}.
  - > It is convenient to use the symbols {0,1} as though they were binary numbers, but they are strictly symbols!

EEL3701: Digital Logic & Computer Systems

Boolean Algebra Notation

- A Boolean variable, usually an uppercase letter, e.g., A, B, X, Y is a variable that can have one and only one state, mainly 0 or 1 (F,T). These variables represent the inputs and outputs of digital devices.
- A Boolean variable may be a function of other Boolean variables, e.g., C = AB+D ⇒ C = f(A,B,D).

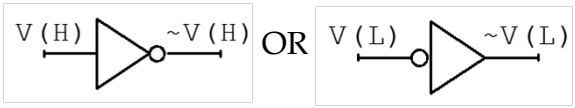
EEL3701: Digital Logic & Computer Systems

Inverter, Not, Level Shifter

- The most basic Boolean operations (the basic operations) include: AND, OR, and NOT (complement) .

Definition of **NOT:**
The complement of a Boolean variable V, written as ~V (or $\overline{V}$ or /V or V'), is defined as:
~V=0 if V=1 or ~V=1 if V=0

> The electronic device that performs the logical complement operation is called LEVEL SHIFTER (or an INVERTER)



V(•)	~V(•)
F	T
T	F

EEL3701: Digital Logic & Computer Systems

Informal Intro to Boolean Algebra

- The second most basic Boolean operation is AND.

Definition of **AND:**
The AND of Boolean variables {A,B}, written as A*B, A•B, A∧B or AB, is defined as:
AB=1 iff A=1 and B=1 else AB=0

> The electronic device that performs the logical AND operation is called an AND gate.

A	B	AB
F	F	F
F	T	F
T	F	F
T	T	T

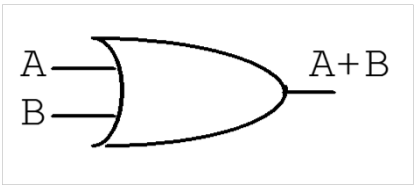
EEL3701: Digital Logic & Computer Systems

Informal Intro to Boolean Algebra

- The third most basic Boolean operation is OR.

Definition of **OR:**
The OR of Boolean variables {A,B}, written as A+B or A∨B, is defined as:
A+B=0 iff A=0 and B=0 else A+B=1

> The electronic device that performs the logical OR operation is called an OR gate.



A	B	A+B
F	F	F
F	T	T
T	F	T
T	T	T

EEL3701: Digital Logic & Computer Systems

The Mathematics of Logic Design - Boolean Algebra

- Boolean expressions are formed with the basic operations of AND, OR, and NOT being applied to one or more constants or variables.
  - >Parentheses are added as needed to specify the order in which operations are performed
  - >In the absence of parentheses we use the following hierarchy

Priority	Operation	Comments
First	NOT	Individual Variables
Second	AND	
Third	OR	
Last	NOT	On entire expressions

EEL3701: Digital Logic & Computer Systems

The Mathematics of Logic Design - Boolean Algebra

Definitions:

- ***TERMS***: The objects of the universe of discourse, e.g., the constants [0,1], variables, and functions.
- ***LITERAL***: A variable or its complement
  >Ex: Let Z = ABC + AB' + A'BC' + B'C'. Then this equation has 4 variables (A, B, C, Z) and 11 literals.
- ***EQUIVALENCE***: Two Boolean expressions are equivalent iff they have the same values for every possible combination of the variables. Since a Truth Table is an exhaustive (complete) tabulation of the input variables, identical columns imply equivalent (equal) expressions.
- ***OBSERVATION***: Each literal in a Boolean expression corresponds to a logic gate input/output.

EEL3701: Digital Logic & Computer Systems

The End!