University of British Columbia
Electrical and Computer Engineering
Digital Systems and Microcomputers CPEN312

L04: Boolean Algebra

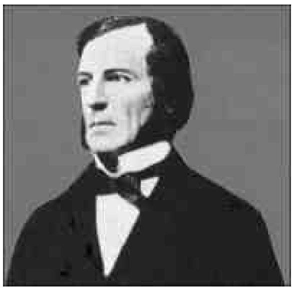
Dr. Jesús Calviño-Fraga. P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

January 11, 2019

Objectives

- Definitions of Boolean Algebra
- Axioms and Theorems of two valued Boolean Algebra
- Boolean Function Simplification
- Canonical forms: Minterms and Maxterms

Boolean Algebra



George Boole (Nov 2, 1815 – Dec 8, 1864)

Boolean Algebra

- Algebra: the part of mathematics in which letters and other symbols are used to represent numbers and quantities in functions.
- Boolean Algebra: Algebra for binary variables.
- Extremely useful to design/test/build complex systems, for example:
  - Functions of a computer.
  - ICs and ASICs device.
  - Programmable logic.
  - Transitions in state machines.
  - Control systems.
  - Programming.

Laws of Boolean Algebra

- **Identity Law:**
  - A+0=A
  - A.1=A
- **Commutative Law**
  - A+B=B+A
  - A.B=B.A
- **Associate Law**
  - *(A + B) + C = A + (B + C)*
  - *(A .B). C = A .(B .C)*
- **Distributive Law**
  - *A. (B + C) = A .B + A .C*
  - *A + (B. C) = (A + B) (A + C)*
- **Redundancy Law**
  - *A + A. B = A*
  - *A .(A + B) = A*

Axioms of Boolean Algebra

- Axiom: *a statement or proposition that is regarded as being established, accepted, or self-evidently true.*

• A+A'=1	A.A'=0
• AB+AB'=A	(A+B).(A+B')=A
• 1+A=1	0.A=0
• A+A'.B=A+B	A.(A'+B)=A.B
• (A+B)'=A'.B'	(A.B)'=A'+B'
• A+A=A	A.A=A

Example 1

- Prove that A+A'.B=A+B using both a truth table and algebra.

A	B	A'.B	A+ A'.B	A+B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

Example 1

A+A'.B=

= A.1+A'.B	*Since A=A.1*
= A.(1+B)+A'.B	*Since 1=(1+B)*
= A+A.B+A'B	*Distributive law*
= A+(A+A').B	*Distributive law*
= A+1.B	*Since A+A'=1*
= A+B	*Since 1.B=B*

Example 2

- Prove (A+B).(A'+C)=A.C+A'.B algebraically.

(A+B).(A'+C)=

= A.A'+A.C+B.A'+B.C = A.C+B.A'+B.C

= A.C.(B+B')+B.A'.(C+C')+B.C.(A+A')

= A.B.C+A.B'.C+A'.B.C+A'.B.C'+A.B.C+A'.B.C

= $\boxed{\text{A.B.C+A.B'.C}}$+$\boxed{\text{A'.B.C+A'.B.C'}}$

=A.C.(B+B')+A'.B.(C+C')

=A.C+A'.B

Example 3

- Use the truth table below to prove DeMorgan's Theorem:
  (A+B)'=A'.B' (A.B)'=A'+B'

A	B	A'	B'	A+B	(A+B)'	A'.B'	A.B	(A.B)'	A'+B'
0	0								
0	1								
1	0								
1	1								

Example 3

- Use the truth table below to prove DeMorgan's Theorem:
  (A+B)'=A'.B'   (A.B)'=A'+B'

A	B	A'	B'	A+B	(A+B)'	A'.B'	A.B	(A.B)'	A'+B'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

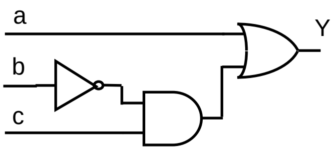
Boolean Functions

- Boolean functions have the form F=f(a,b,c…) where a, b, c, etc. are binary variables.
- Use logic gates to implement the function.
- Use Boolean algebra to simplify the function so it makes it more convenient to implement using logic gates.

Example 4

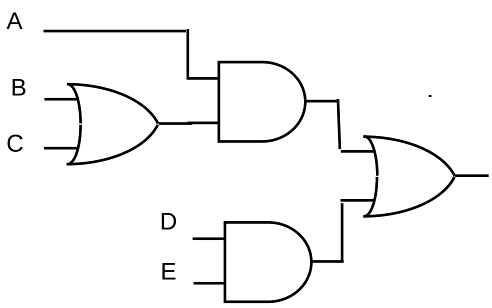
- Get the truth table and circuit for Y=a+b’.c

a	b	c	b’.c	a+b’.c
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1



Boolean Algebra Benefits

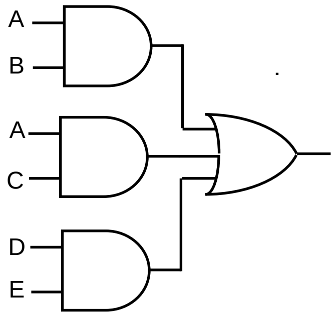
- Y=A.(B+C)+D.E



3 level of gates

Boolean Algebra Benefits

- Y=A.(B+C)+D.E=A.B+A.C+D.E



2 level of gates=less delay

Canonical Forms

- Canonical means the standard form of representing an equation.
- Boolean functions can appear in two different canonical forms:
  – Sum of Minterms (or sum of products SOP): f=m0+m1+m2+m3+…
  – Product of Maxterms (or product of sums POS): f=M0.M1.M2.M3….
- For n variables there are 2^n Minterms or Maxterms.

Canonical Forms

Inputs			Minterms		Maxterms	
a	b	c	Term	Designation	Term	Designation
0	0	0	a'.b'.c'	m0	a+b+c	M0
0	0	1	a'.b'.c	m1	a+b+c'	M1
0	1	0	a'.b.c'	m2	a+b'+c	M2
0	1	1	a'.b.c	m3	a+b'+c'	M3
1	0	0	a.b'.c'	m4	a'+b+c	M4
1	0	1	a.b'.c	m5	a'+b+c'	M5
1	1	0	a.b.c'	m6	a'+b'+c	M6
1	1	1	a.b.c	m7	a'+b'+c'	M7

Fixed!

Canonical Forms

- The Boolean function can be obtained by inspection from the truth table using:
  - The sum of Minterms or SOP:

$$f(a,b,c,\ldots)=\sum m_i$$

  - The product of Maxterms or POS:

$$f(a,b,c,\ldots)=\prod M_i$$

Example 5

- Get the Boolean functions for the two outputs in the truth table below. Use the sum of Minterms (SOP).

Inputs			Outputs	
c	**b**	**a**	**X**	**Y**
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

Minterms for X

Minterms for Y

X=c'.b'.a+c.b'.a'+c.b.a

Y=c'.b.a+c.b'.a+c.b.a'+c.b.a

Example 6

- Get the Boolean functions for the two outputs in the truth table below. Use the product of Maxterms (POS).

Inputs			Outputs	
c	**b**	**a**	**X**	**Y**
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

Maxterms for X

Maxterms for Y

X=(c+b+a).(c+b'+a).(c+b'+a').
(c'+b+a').(c'.b'.a)

Y=(c+b+a).(c+b+a').(c+b'+a)
.(c'+b+a).

Fixed!

Example 7

- Obtain a digital circuit that implements the truth table below.

Inputs			
C	**B**	**A**	**Y**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Tip: check the output before you decide to use either minterms or maxterms. Pick the less numerous. In this case three outputs are 1 while five outputs are 0. So it looks easier to work with minterms.

Example 7

C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$Y = \overline{C} \cdot B \cdot \overline{A} + C \cdot \overline{B} \cdot A + C \cdot B \cdot \overline{A}$$

The function above requires a whole lot of gates! Use Boolean algebra to reduce it.

Example 7

$$Y = \overline{C} \cdot B \cdot \overline{A} + C \cdot \overline{B} \cdot A + C \cdot B \cdot \overline{A}$$

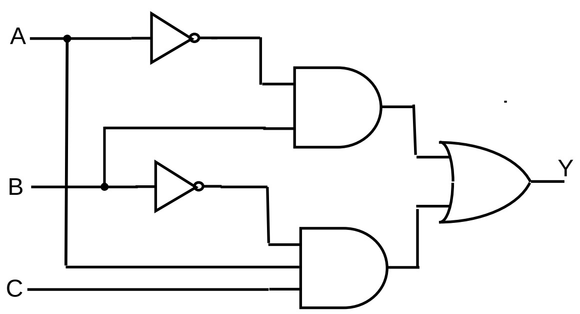
$$Y = \left(\overline{C} + C\right) \cdot B \cdot \overline{A} + C \cdot \overline{B} \cdot A$$

$$Y = B \cdot \overline{A} + C \cdot \overline{B} \cdot A$$

Saved one AND as well as one OR!

Example 7

$$Y = B \cdot \overline{A} + C \cdot \overline{B} \cdot A$$



Simulating Digital Circuits with Multisim

- Multisim is a circuit simulator from National Instruments capable of simulating both analog and digital circuits.
- Download and install Multisim into your personal computer.
- First lab requires the use of Multisim.
- Multisim is also installed in the labs.

Simulating Digital Circuits with Multisim

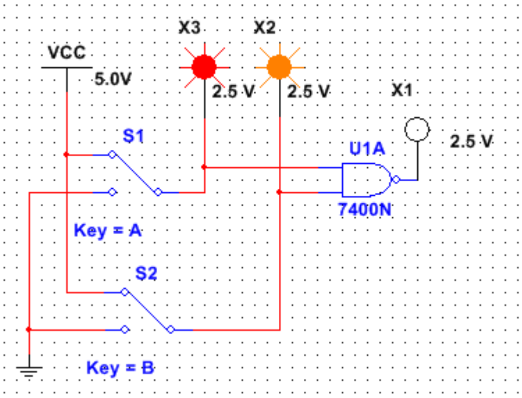
- A massive collection of logic gates and functions are available. Right click, “Place Component”, in the “Group” drop-box select “TTL”. Pick your gate! Some standard gates:
  - 74LS00: 2-input NAND gate
  - 74LS04: 2-input NOT gate
  - 74LS08: 2-input AND gate
  - 74LS32: 2-input OR gate
  - 74LS02: 2-input NOR gate
  - 74LS86: 2-input XOR gate
  - 74LS30: 8-input AND gate

Simulating Digital Circuits with Multisim

- Our inputs will be Single Pole Double Toggle switches. They are in the group “Basic”->“SWITCH”.
- Our outputs will be lights that turn on with logic 1. They are in the group “Indicators”-> “PROBE”

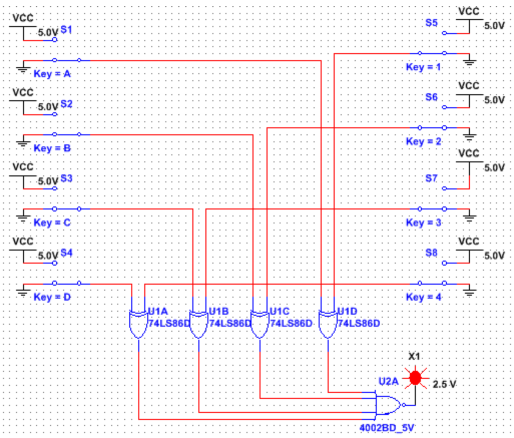
Simulating Digital Circuits with Multisim

- First try, a simple two input NAND gate.



Example 8

- Design a circuit to compare if two 4-bit numbers are equal. Use XOR gates. Simulate with Multisim.

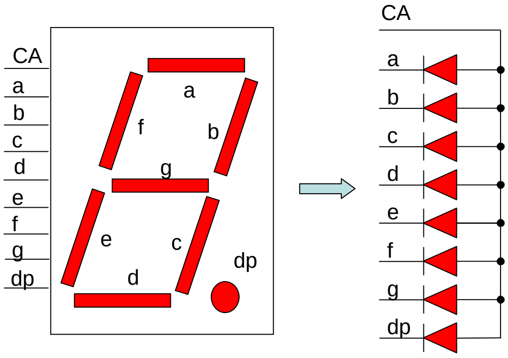


Example 9

- Design a 2-bit to 7-segment decoder. Test the circuit with Multisim.

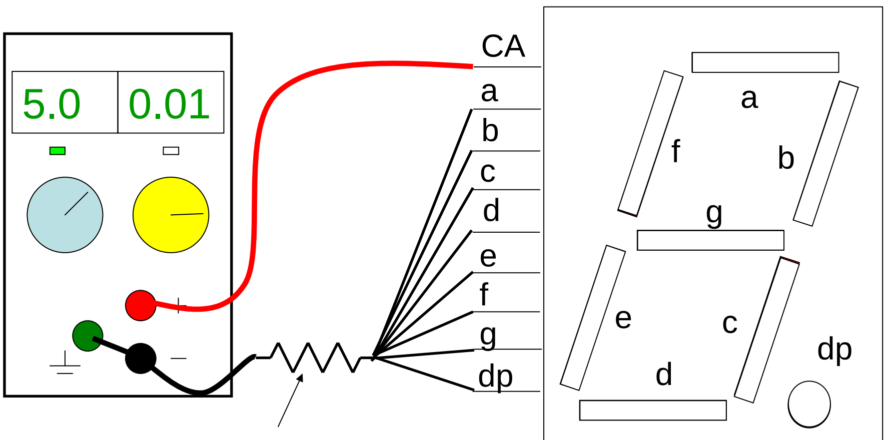
This one can not be solved easily by inspection as the previous example. We need the truth table and the Boolean equations before we can get the circuit going.

7-Segment Display



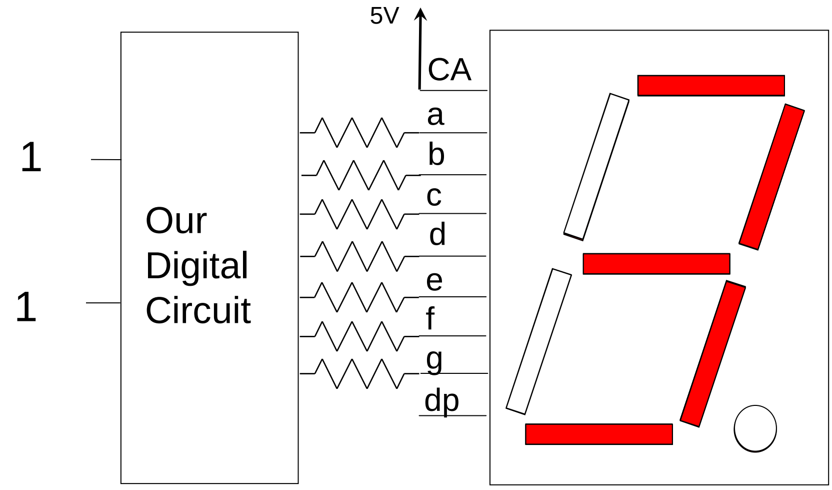
This is the kind of display you'll need to use for LAB 1. It is available in Multisim in Group "Indicators", "HEX_DISPLAY", "SEVEN_SEG_COM_A". There are displays with built in decoders (DCD_HEX), don't use that one for this example!

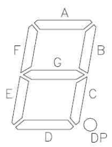
Testing the Display



Select this resistor for 3 mA per segment: 1k

What We Need:





2-bit to decimal

Inputs		Segments On
B	A	
0	0	All but G
0	1	B, C
1	0	All but C, F
1	1	All but E, F

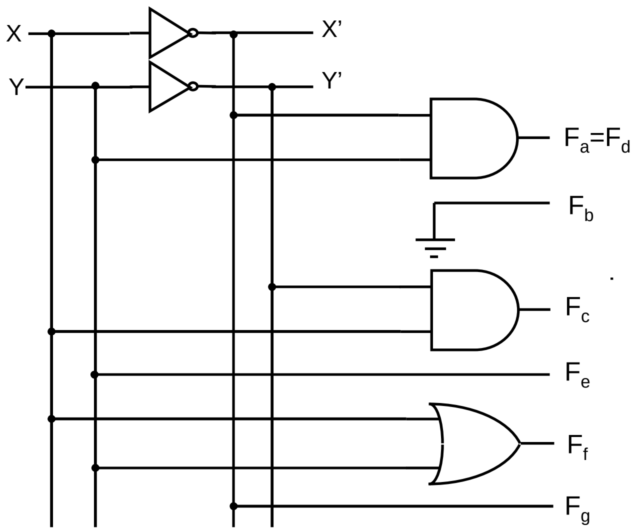
- Start with the truth table and go from there:

IN		OUT						
X	Y	F_a	F_b	F_c	F_d	F_e	F_f	F_g
0	0	0	0	0	0	0	0	1
0	1	1	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0
1	1	0	0	0	0	1	1	0

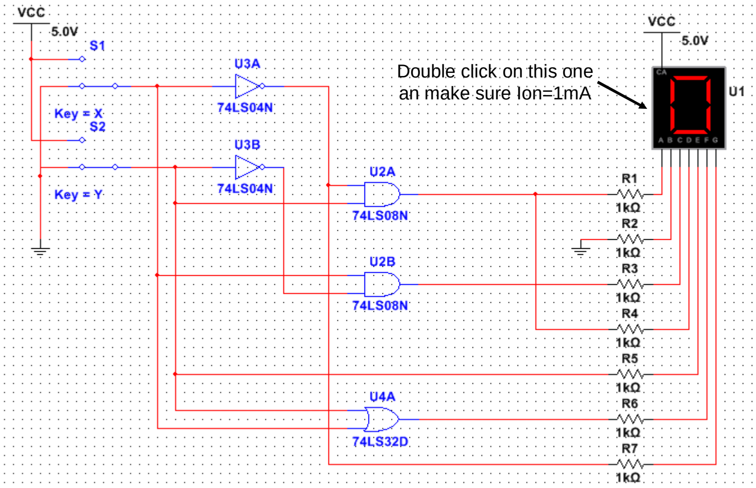
The Equations…

- F_a=F_d=X'.Y
- Fb=0
- Fc=X.Y'
- Fe=X'.Y+X.Y=Y.(X'+X)=Y
- Ff=X+Y (Using Maxterms)
- FG=X'.Y'+X'.Y=X'.(Y'+Y)=X'

The Digital Circuit



The Simulation…

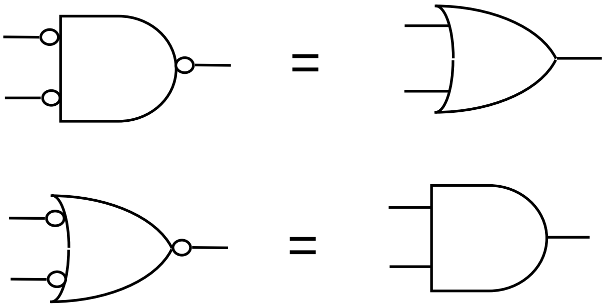


LAB 1

- Posted on Canvas.
- Similar to the exercise above, but display digits based on your student number. Solve it using NAND gates only.
- You'll need to simulate the design (pre-lab) and assemble it with real parts (lab).
- You'll need a breadboard, wire stripper, pliers, and the parts. There are some breadboards and tools available in MCLD410 for you to use. If you have your own, even better!
- Parts in the kit are 2 x 74HC00 (or equivalent) NAND gate integrated circuits, 8 x 1k resistors, 1 x display LTS-4802BJS-H1 (or similar). Also there are parts to assemble one discrete NAND gate using diodes, resistors, and NPN transistor.
- Never used a breadboard? Tutorial on YouTube: http://www.youtube.com/watch?v=Ynxg19lEkvg

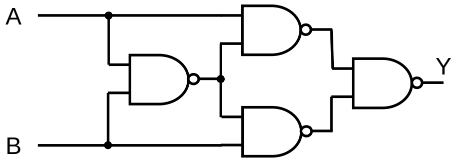
Exercises

- Prove that the following logic gates are equivalent. Tip: Use DeMorgan's Theorem.



Exercises

- Prove both using Boolean algebra and a truth table, that the circuit below implements the XOR function Y=A.B'+A'.B



- Design a 4-bit prime number detector using binary logic.