Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Department of Electrical & Computer Engineering

Digital Logic And Computing Systems

Chapter 02 –Switching Algebra

Dr. Christophe Bobda

EEL3701C Fall 2022

Agenda

- Boolean Algebra
- Switching Algebra
- NAND- and NOR-Functions

Boolean Algebra

- George Boole (1815-1864)
  - Application of algebraic methods on logic assertions
  - ‘0’ and ‘1’ as logic values for “false” and “true”
  - Variables to model the value of assertion
- Ernst Schröder (1841-1902), Edward V. Huntington (1874-1952)
  - “Axiomatisation” of Boolean Algebra
  - Application to telephone equipment
- Claude Shannon (1916-2001)
  - Application of Boolean algebra to analyze the design of digital electromechanical circuits

Boolean Algebra

- A Boolean Algebra (B, $\vee$, $\wedge$, $\neg$) consist of
  - A support B (set of all variables)
  - Two 2-input operators

    $\vee, \wedge : B^2 \rightarrow B$

    ($\vee$ known as disjunction, union, OR, ...)
    ($\wedge$ known as conjunction, intersection AND, ...)
  - One 1-input operator $\neg : B \rightarrow B$

    ($\neg$ known as negation, complement, NOT, ...)

Boolean Algebra Laws

- Huntington's Postulate: For the following axioms

(1b) $\forall\ a,b \in B\text{:}\ \ a \vee b \in B$ (Closure)

(1b) $\forall\ a,b \in B\text{:}\ \ a \wedge b \in B$

Boolean Algebra

Laws

- Huntingtons Postulate (cont)

(2a) $\exists\ 0 \in B$: $a \vee 0 = a$ (Existence of a Null-Element)
(2b) $\exists\ 1 \in B$: $a \wedge 1 = a$ (Existence of a One-Element)

(3a) $\forall\ a,b \in B$: $a \vee b = b \vee a$ (commutativity)
(3b) $\forall\ a,b \in B$: $a \wedge b = b \wedge a$

(4a) $\forall\ a,b,c \in B$: $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ (Distributivity)
(4b) $\forall\ a,b,c \in B$: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$

(5a) $\forall\ a \in B$: $\exists\ \neg a$: $a \vee \neg a = 1$ (Existence of an inverse)
(5b) $\forall\ a \in B$: $\exists\ \neg a$: $a \wedge \neg a = 0$

(6) $\exists\ a,b \in B$: $a \neq b$ (B has min. 2 elements)

Switching Algebra

- A Switching algebra is a Boolean algebra with the smallest support B = {0,1}
- A switching algebra knows variables and constants
- There are only two constants: 0 and 1

Interrupted wire (connection) 0

Non-Interrupted wire (connection) 1

- A variable can have two possible values: 0 und 1

Switch open A = 0

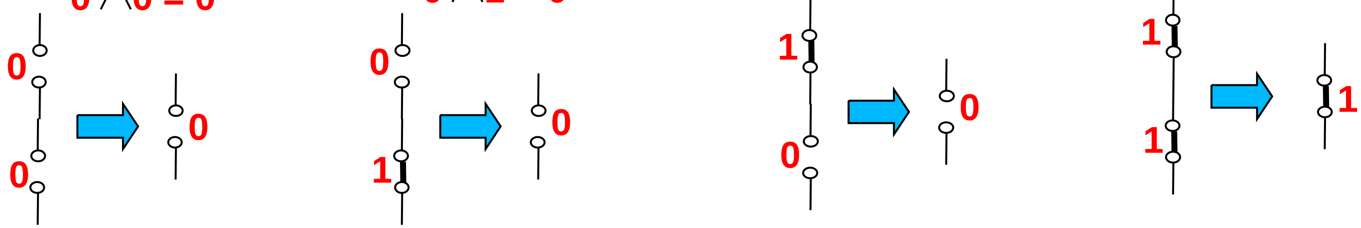
Switch closed A = 1

Switching Algebra
Basic Rules

- Basic rules of switching algebra define the relation between variables and constants

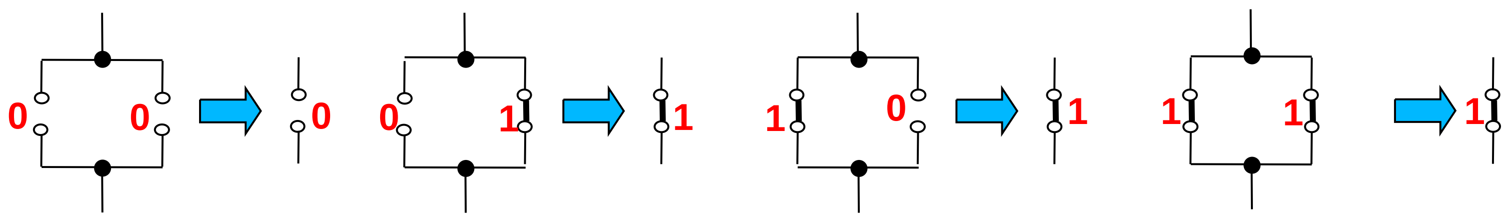
- AND

$0 \wedge 0 = 0$ $\quad$ $0 \wedge 1 = 0$ $\quad$ $1 \wedge 0 = 0$ $\quad$ $1 \wedge 1 = 1$



- OR

$0 \vee 0 = 0$ $\quad$ $0 \vee 1 = 1$ $\quad$ $1 \vee 0 = 1$ $\quad$ $1 \vee 1 = 1$



- NOT

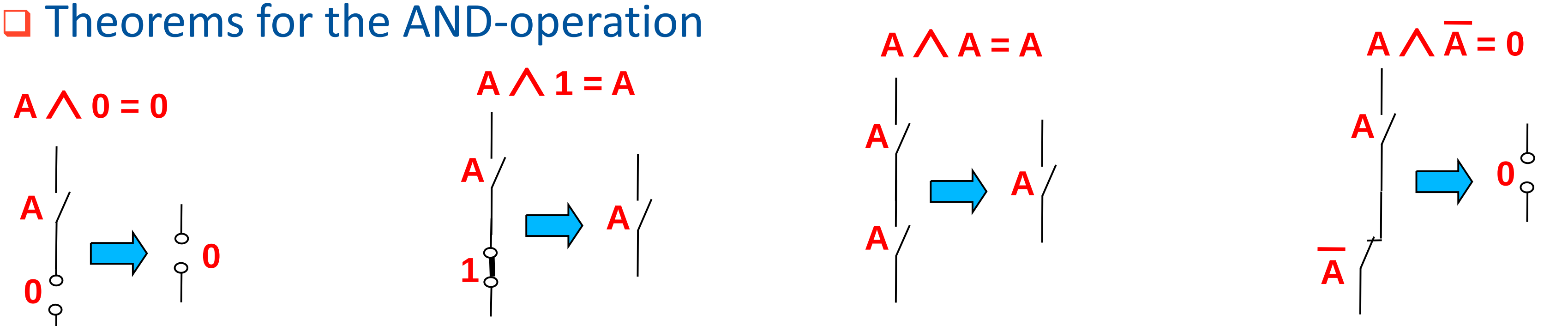
$\overline{0} = 1$ $\quad$ $\overline{1} = 0$

Switching Algebra Rules

- Theorems define rules for operation between
  - A variable and a constant
  - A variable and a variable
  - A variable and it‘s negation

- For variable, the following notations A, B, C, will be used. What is true for A is also true for B, C, …

- Theorems for the AND-operation

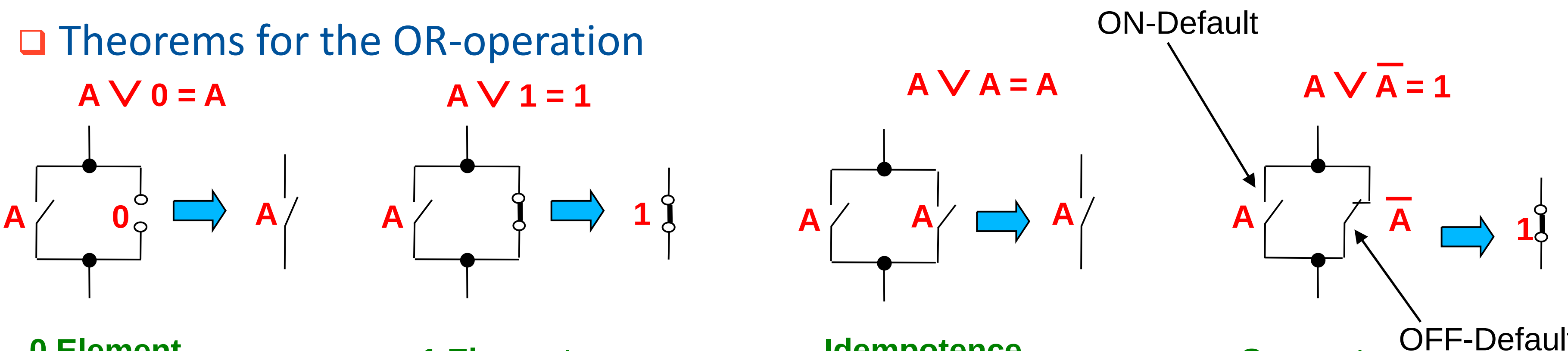


0 Element | 1 Element | Idempotence | Symmetry

Switching Algebra Rules

- Theorems for the OR-operation

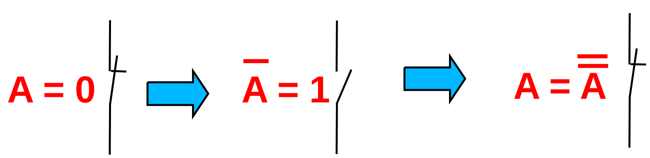
$A \vee 0 = A$ $A \vee 1 = 1$ $A \vee A = A$ $A \vee \overline{A} = 1$



0 Element **1 Element** **Idempotence** **Symmetry**

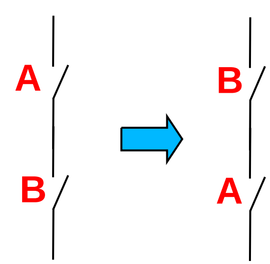
- NOT

$\overline{\overline{A}} = A$

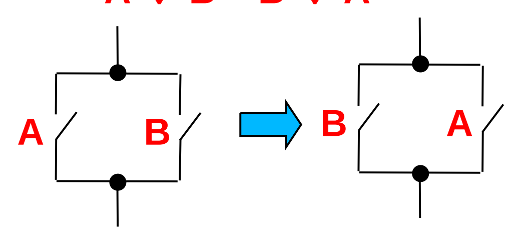


- Commutativity

$A \wedge B = B \wedge A$



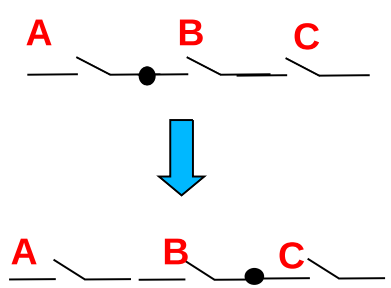
$A \vee B = B \vee A$



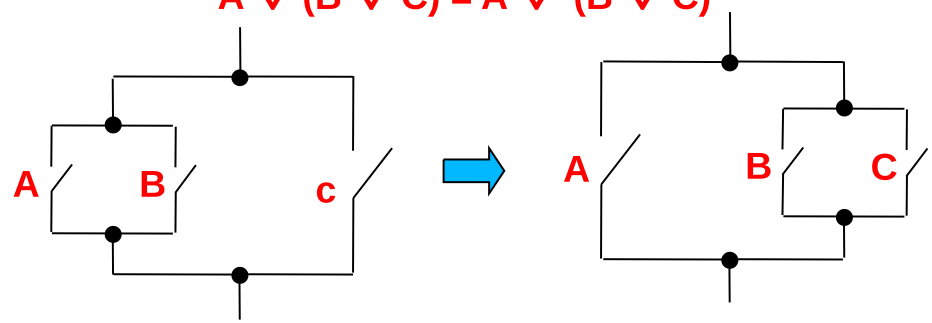
Switching Algebra Rules

- Associativity

$A \wedge (B \wedge C) = (A \wedge B) \wedge C$

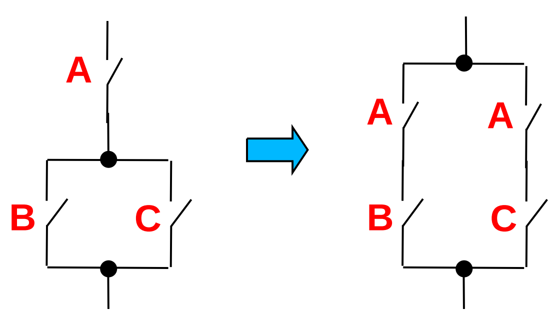


$A \vee (B \vee C) = A \vee (B \vee C)$

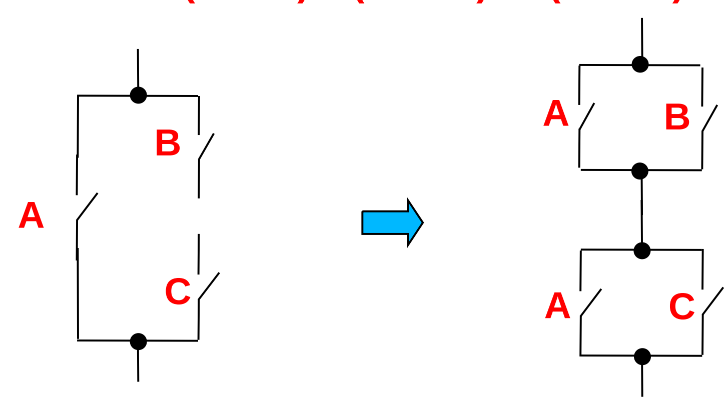


- Distributivity

$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$



$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$



Switching Algebra
De Morgan's Law

- $\overline{A \wedge B} = \bar{A} \vee \bar{B}$
- $\overline{A \vee B} = \bar{A} \wedge \bar{B}$
- Proof
  - In general, the goal of a proof is to establish that two logical expressions or logical networks are functionally equivalent
  - A truth table can be used

A	B	$A \wedge B$	$\overline{A \wedge B}$	$\bar{A}$	$\bar{B}$	$\bar{A} \vee \bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

A	B	$A \vee B$	$\overline{A \vee B}$	$\bar{A}$	$\bar{B}$	$\bar{A} \wedge \bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Switching Algebra Rules

- Precedence Rules: to avoid ambiguity in evaluation of complex statements
  - Which of these two statements is correct?

    $A \wedge B \vee C = (A \wedge B) \vee C$

    $A \wedge B \vee C = A \wedge (B \vee C)$

- Precedence Rules
  - NOT-operations have the highest precedence (stronger than OR and AND)
  - AND-operations have the next highest precedence (Stronger than OR)

Switching Algebra

Rules

- Set priority
  - Operator with the highest precedence is grouped with its operand(s) first
  - Then the next highest operator is grouped with its operands, and so on
  - If there are several logical operators of the same precedence, they will be examined from left to right

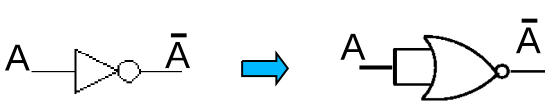
$$A \wedge \bar{B} \vee C = (A \wedge (\bar{B})) \vee C$$

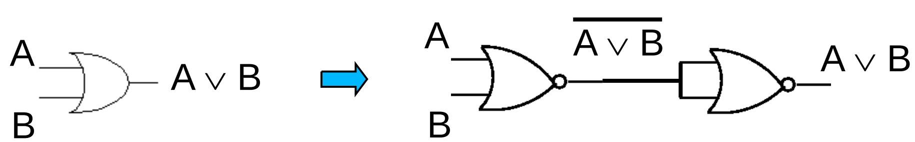
Switching Algebra
Universal Gates (NAND and NOR)

- Switch algebra is built on the three basic operators AND, OR, NOT
  - Any “switching“ function can be implemented with only those three operators

- Furthermore: $A \wedge B = \overline{\overline{A \wedge B}} = \overline{\bar{A} \vee \bar{B}}$
  - AND-gates can be implemented with 1 OR-gate and 3 NOT-gates
    - ➡ AND-gates are not necessary in circuit design
  - OR-gates and NOT-Gates can be implemented only with NOR-Gates

➡ NOR-gates are universal:
- Any circuit can be constructed with only NOR-gates





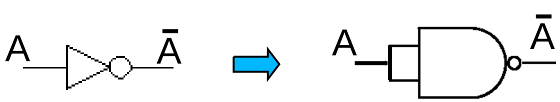
Switching Algebra
Universal Gates (NAND and NOR)

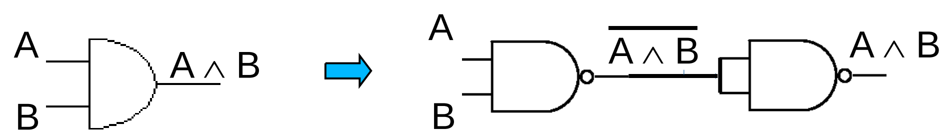
- OR-gates can be implemented with 1 AND-gate and 3 NOT-gates

$$A \vee B = \overline{\overline{A \vee B}} = \overline{\overline{A} \wedge \overline{B}}$$

➡ OR-gates are not necessary in circuit design

- AND-gates and NOT-Gates can be implemented only with NAND-Gates





➡ NAND-gates are universal
  - Any circuit can be constructed with only NAND-gates

UF Herbert Wertheim College of Engineering
UNIVERSITY *of* FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE