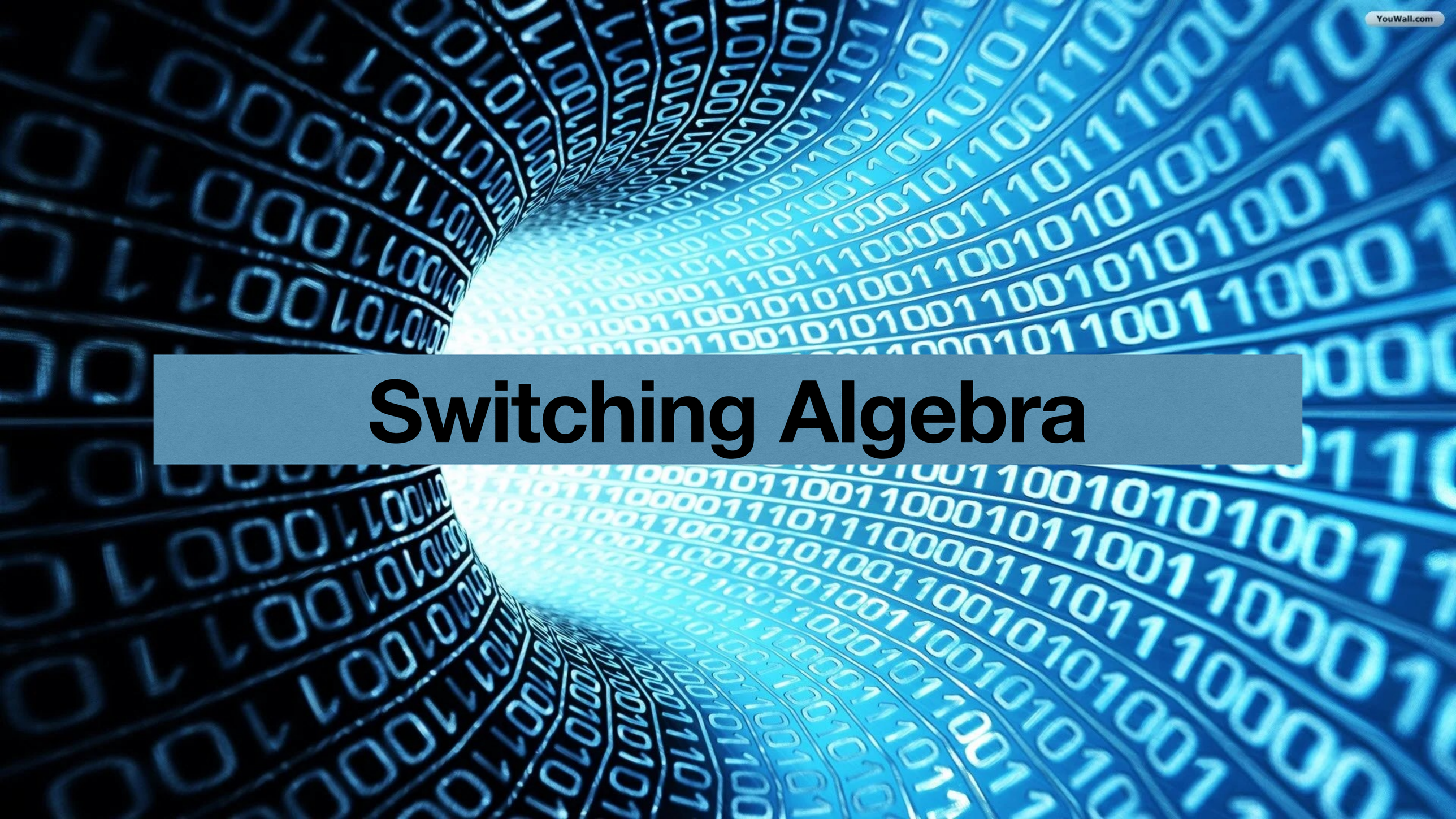
Digital Logic Design + Computer Architecture

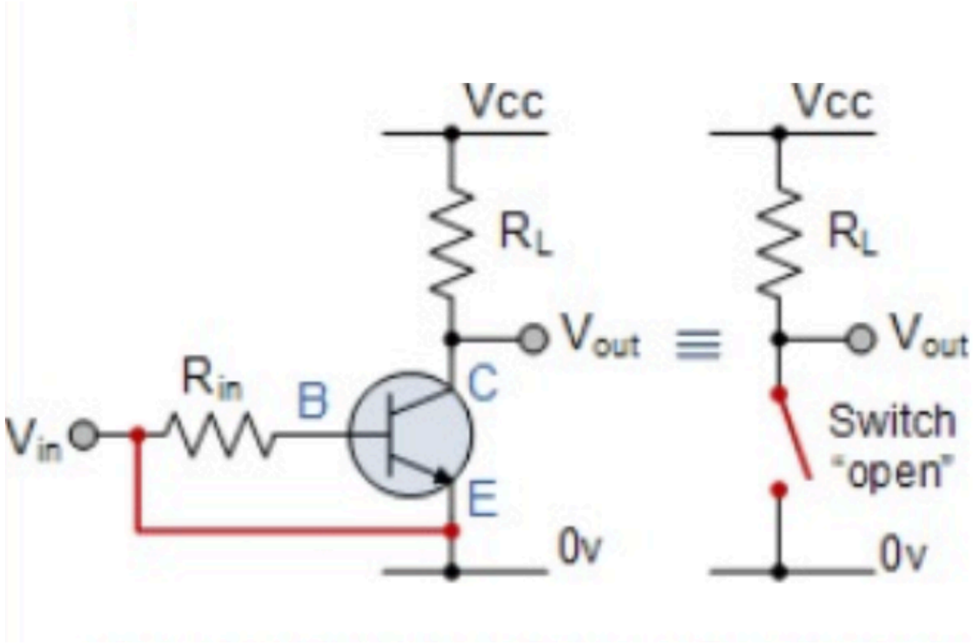
Sayandeep Saha

Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay



Switching Algebra

Switches



Algebra of Switches

- **Warning!!! Too informal**: An algebraic system consists of a set and some well-defined operators on the elements of the set and possibly some relations defined over the elements.
- We will be studying one of the simplest system defined over the set {0, 1} — but ironically this set is going to complicate the rest of your life. :P (just joking)
- **Set**: {0,1}
- **Operators**:
  - AND, OR (Binary)
  - NOT (Unary)

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 1 + 1 = 1.$$

$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 0, \quad 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

$$0' = 1, \quad 1' = 0.$$

Algebra of Switches: Basic Properties

- **Idempotency**: $x + x = x$

  $x \,.\, x = x$

How do we prove this property?
Perfect induction*: proving a theorem by verifying every combination of values that the variables may assume*

Proof of $x + x = x$: $1 + 1 = 1$ and $0 + 0 = 0$

- If x is a switching variable, then: $x + 1 = 1$

  $x \,.\, 0 = 0$

  $x + 0 = x$

  $x \,.\, 1 = x$

- **Commutativity**: $x + y = y + x$

  $x \,.\, y = y \,.\, x$

Algebra of Switches: Basic Properties

- **Associativity**: $(x + y) + z = x + (y + z)$
  $(x \,.\, y) \,.\, z = x \,.\, (y \,.\, z)$

- **Complementation**: $x + x' = 1$
  $x \,.\, x' = 0$

- **Distributivity**: $x \,.\, (y + z) = x \,.\, y + x \,.\, z$
  $x + y \,.\, z = (x + y) \,.\, (x + z)$

Proofs are easy, you can write down the entire truth table

x	y	z	xy	xz	$y+z$	$x(y+z)$	$xy+xz$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

Algebra of Switches: Principle of Duality

- **Observe**: Preceding properties grouped in pairs
- One statement can be obtained from the other by interchanging operations OR and AND and constants 0 and 1
- The two statements are said to be dual of each other
- **Implication**: necessary to prove only one of each pair of statements

Algebra of Switches: Some Basic Laws

- **Switching expression**: combination of finite number of switching variables and constants via switching operations (AND, OR, NOT)
  - Any constant or switching variable is a switching expression
  - If T_1 and T_2 are switching expressions, so are T_1', T_2', T_1+T_2 and T_1T_2
  - No other combination of constants and variables is a switching expression

- **Absorption law**: $x + xy = x$

  $x(x + y) = x$

Proof: $x + xy = x1 + xy$ [basic property]

$= x(1 + y)$ [distributivity]

$= x1$ [commutativity and basic property]

(Note we write in the law y + 1 = y, so we apply commutativity)

$= x$ [basic property]

Algebra of Switches: Some Basic Laws

- **Switching expression**: combination of finite number of switching variables and constants via switching operations (AND, OR, NOT)
  - Any constant or switching variable is a switching expression
  - If T_1 and T_2 are switching expressions, so are T_1', T_2', T_1+T_2 and T_1T_2
  - No other combination of constants and variables is a switching expression

- **Absorption law**: $x + xy = x$

  $x(x + y) = x$

Proof: $x + xy = x1 + xy$ [basic property]

$= x(1 + y)$ [distributivity]

$= x1$ [commutativity and basic property]

(Note we write in the law y + 1 = y, so we apply commutativity)

$= x$ [basic property]

Simplification of Switching Expressions

- **Law 2:** $x + x'y = x + y$, $x(x' + y) = xy$
- **Law 3**: $xy + x'z + yz = xy + x'z$, $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$. **(Consensus Theorem)**
- **Law 4**: $(x')' = x$. **(Involution)**
- **Law 5**: $(x+y)' = x'y'$, $(xy)' = x' + y'$ **(De-Morgan's Theorem)**

Food of thought: Can we extend it for n variables???

x	y	x'	y'	$x+y$	$(x+y)'$	$x'y'$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Try the proofs!!!

Simplification of Switching Expressions

Example: Simplify $T(x,y,z) = (x + y)[x'(y' + z')]' + x'y' + x'z'$

Simplification of Switching Expressions

Example: Simplify $T(x,y,z) = (x + y)[x'(y' + z')]' + x'y' + x'z'$

$$
\begin{aligned}
(x + y)[x'(y' + z')]' + x'y' + x'z' &= (x + y)(x + yz) + x'y' + x'z' \\
&= (x + xyz + yx + yz) + x'y' + x'z' \\
&= x + yz + x'y' + x'z' \\
&= x + yz + y' + z' \\
&= x + z + y' + z' \\
&= x + y' + 1 \\
&= 1
\end{aligned}
$$

Thus, $T(x,y,z) = 1$, independently of the values of the variables

Simplification of Switching Expressions

Example: Prove $xy + x'y' + yz = xy + x'y' + x'z$

Simplification of Switching Expressions

Example: Prove $xy + x'y' + yz = xy + x'y' + x'z$

- From consensus theorem, $x'z$ can be added to LHS
- Consensus theorem can be applied again to first, third and fourth terms in $xy + x'y' + yz + x'z$ to eliminate yz and reduce it to RHS

Simplification of Switching Expressions

Example: Prove $xy + x'y' + yz = xy + x'y' + x'z$

- From consensus theorem, $x'z$ can be added to LHS
- Consensus theorem can be applied again to first, third and fourth terms in $xy + x'y' + yz + x'z$ to eliminate yz and reduce it to RHS

- LHS : $xy + x'y' + yz = xy + (x'y' + yz) = xy + (y'x' + (y')'z + x'z) = xy + y'x' + yz + x'z = (xy + x'z + yz) + x'y' = xy + x'z + x'y' = RHS$

Switching Functions

- **Switching function $f(x_1, x_2, \ldots, x_n)$**: values assumed by an expression for all combinations of variables $x_1, x_2, \ldots, x_n$
- **Complement function**: $f'(x_1, x_2, \ldots, x_n)$ assumes value 0 (1) whenever $f(x_1, x_2, \ldots, x_n)$ assumes value 1 (0)
- **Logical sum of two functions**: $f(x_1, x_2, \ldots, x_n) + g(x_1, x_2, \ldots, x_n) = 1$ for every combination in which either f or g or both equal 1
- **Logical product of two functions**: $f(x_1, x_2, \ldots, x_n) \cdot g(x_1, x_2, \ldots, x_n) = 1$ for every combination for which both f and g equal 1

x	y	z	f	g	f'	$f+g$	fg
0	0	0	1	0	0	1	0
0	0	1	0	1	1	1	0
0	1	0	1	0	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	1	1	0
1	0	1	0	0	1	0	0
1	1	0	1	1	0	1	1
1	1	1	1	0	0	1	0

Canonical Forms

- Truth tables are one of the most standard way of representing switching functions.
- But a functions with n variables require 2^n rows each of size $(n+1)$ bits. — quite some overhead.
- **Let's consider the problem of deriving a compact expression of a function from a given truth table**

Decimal code	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

$$f = x'y'z' + x'yz' + x'yz + xyz' + xyz$$

- Find the sum (OR) of all terms for which the function evaluates to 1.
  - In this case, for (0, 2, 3, 6, 7).
- Each term is a product of the variables on which the function depends
- Variable x_i appears in uncomplemented (complemented) form in the product if has value 1 (0) in the combination

Canonical Forms: Sum of Products

- **Minterm**: a product term that contains each of the *n* variables as factors in either complemented or uncomplemented form
  - It assumes value 1 for exactly one combination of variables
  - In the following table 000, 010, 011, 110, 111 are minterms, which are usually represented with their decimal equivalent (0,2,3,6,7). In terms of variables *x'y'z', x'yz', x'yz, xyz', xyz*

- **Canonical sum-of-products**: sum of all minterms derived from combinations for which function is 1
  - Also called **disjunctive normal expression**

- **Compact representation**: $\sum (0,2,3,6,7)$

Decimal code	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Canonical Forms: Product of Sums

- **Maxterm**: a sum term that contains each of the *n* variables in either complemented or uncomplemented form
  - It assumes value 0 for exactly one combination of variables
  - Variable x_i appears in uncomplemented (complemented) form in the sum if it has value 0 (1) in the combination
- **Canonical product-of-sums**: product of all maxterms derived from combinations for which function is 0
  - Also called **conjunctive normal expression**

- **Compact representation**: $\prod(1,4,5)$

$$f = (x + y + z')(x' + y + z)(x' + y + z')$$

Decimal code	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Derivation of SOP Forms

1. Examine each term: if it is a minterm, retain it; continue to next term

2. In each product which is not a minterm: check the variables that do not occur; for each x_i that does not occur, multiply the product by $(x_i + x_i')$

3. Multiply out all products and eliminate redundant terms

Example: $T(x,y,z) = x'y + z' + xyz$

$$= x'y(z + z') + (x + x')(y + y')z' + xyz$$
$$= x'yz + x'yz' + xyz' + xy'z' + x'yz' + x'y'z' + xyz$$
$$= x'yz + x'yz' + xyz' + xy'z' + x'y'z' + xyz$$

Canonical product-of-sums obtained in a dual manner

Example:

$T = x'(y' + z)$

$= (x' + yy' + zz')(y' + z + xx')$

$= (x' + y + z)(x' + y + z')(x' + y' + z)(x' + y' + z')(x + y' + z)(x' + y' + z)$

$= (x' + y + z)(x' + y + z')(x' + y' + z)(x' + y' + z')(x + y' + z)$

Transforming Canonical Forms

- **Example**: Find the canonical product-of-sums for

$$T(x,y,z) = x'y'z' + x'y'z + x'yz + xyz + xy'z + xy'z'$$

$$T = (T')' = [(x'y'z' + x'y'z + x'yz + xyz + xy'z + xy'z')']'$$

Complement T' consists of minterms not contained in T. Thus,

$$T = [x'yz' + xyz']' = (x + y' + z)(x' + y' + z)$$

- Canonical forms are unique

- **Two switching functions are equivalent if and only if their corresponding canonical forms are identical**

Introducing Exclusive-OR (XOR)

Exclusive-OR: modulo-2 addition, i.e., $A \oplus B = 1$ if either A or B is 1, but not both.

Commutativity: $A \oplus B = B \oplus A$

Associativity: $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

Distributivity: $(AB) \oplus (AC) = A(B \oplus C)$

If $A \oplus B = C$, then

$A \oplus C = B$

$B \oplus C = A$

$A \oplus B \oplus C = 0$

Exclusive-OR of an even (odd) number of elements, whose value is 1, is 0 (1)

Functional Completeness

- A set of operations is **functionally complete** (or **universal**) if and only if every switching function can be expressed by operations from this set

- Every switching function can be expressed in canonical form consisting of a finite number of switching variables, constants and operations OR, AND, NOT

- **Example**: Set {+, ., ’}
- **Example**: Set {+, ’} since using De Morgan’s theorem, $x \cdot y = (x' + y')'$. Thus, + and ’ can replace the . in any switching function
- **Example**: Set {., ’} for similar reasons
- **Example**: NAND since NAND(x,x) = x' and NAND[NAND(x,y),NAND(x,y)] = xy
- **Example**: NOR since NOR(x,x) = x' and NOR[NOR(x,y),NOR(x,y)] = $x + y$

What Can be done with Switching Algebra?

Isomorphism: Two algebraic systems are isomorphic if

- For every operation in one system, there exists a corresponding operation in the second system
- To each element x_i in one system, there corresponds a unique element y_i in the other system, and vice versa
- If each operation and element in every postulate of one system is replaced by the corresponding operation and element in the other system, then the resulting postulate is valid in the second system

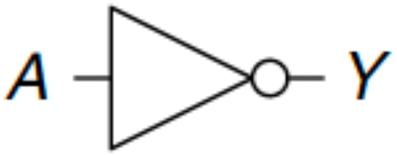
Thus, two algebraic systems are isomorphic if and only if they are identical except the labels and symbols used to represent the operations and elements.

- Switching algebra is isomorphic to proposition calculus — something which encodes declarative statements with values “True” or “False”, but never both.
  - “If it rains, then we play”

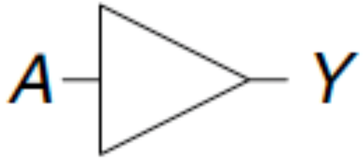
- **Most important**!!!Switching algebra is isomorphic to the **network of logic gates**, something which we build with transistors.

At The Gates

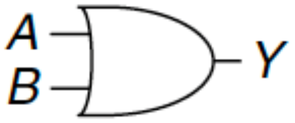
NOT



BUF



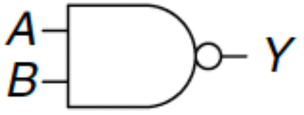
OR



AND

XOR

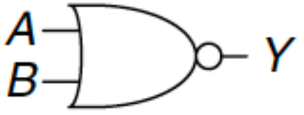
NAND



$Y = \overline{AB}$

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

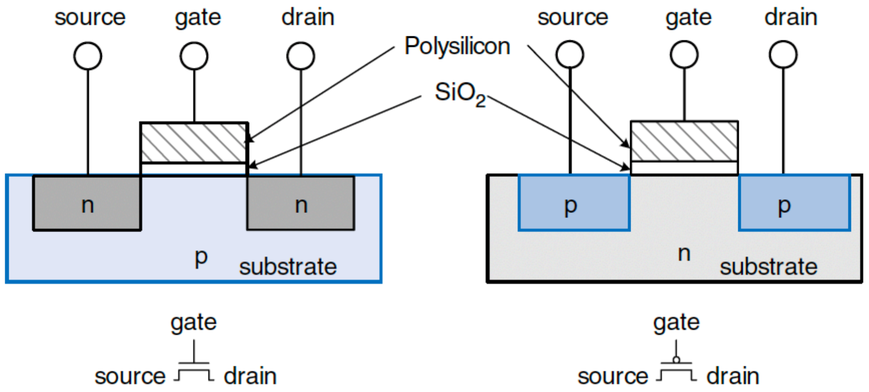
NOR

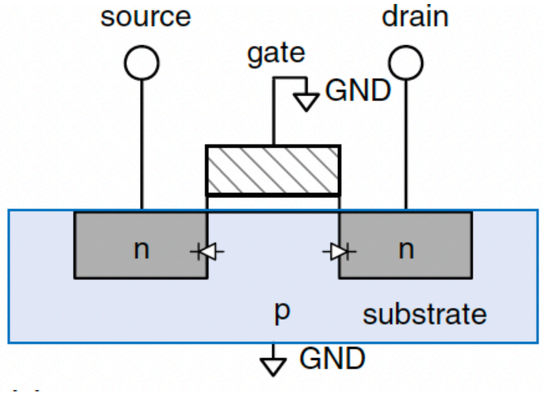


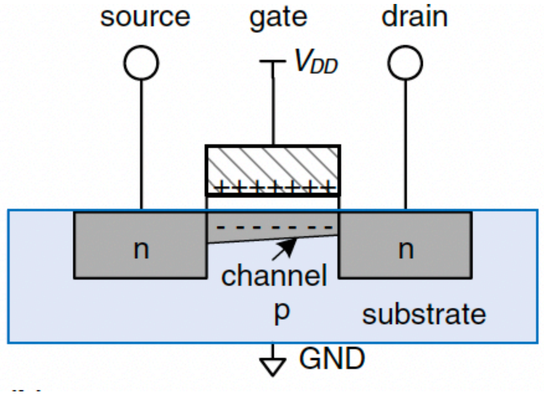
$Y = \overline{A+B}$

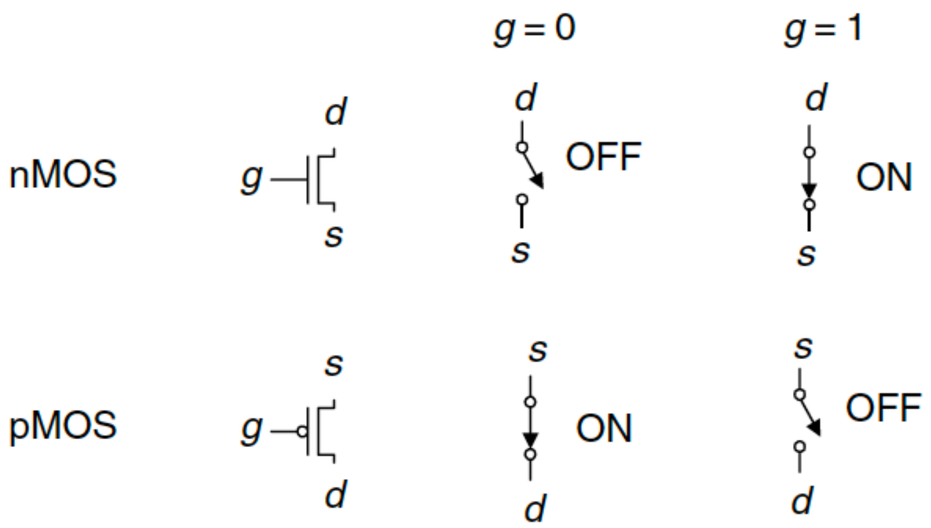
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

One Level Below

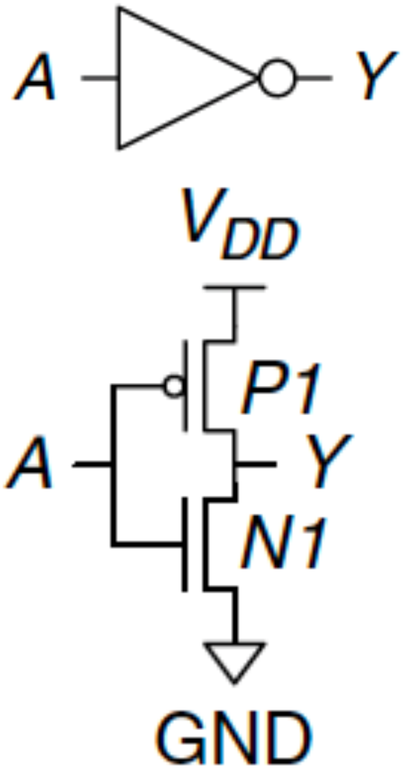


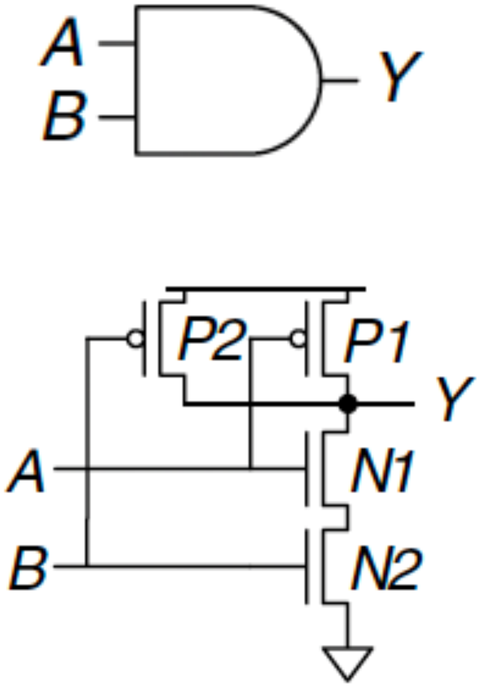


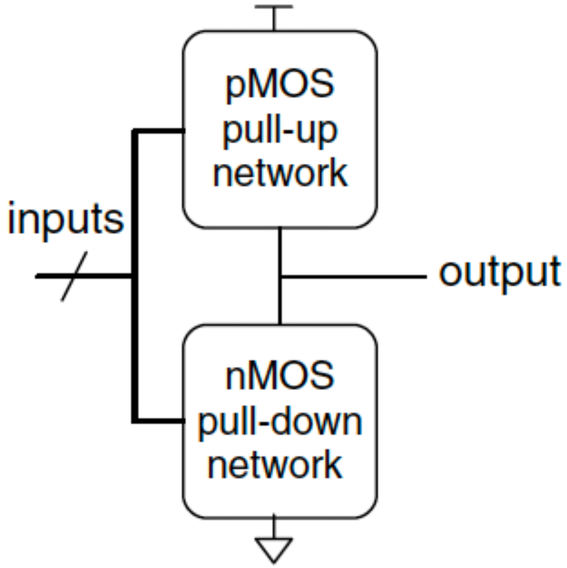




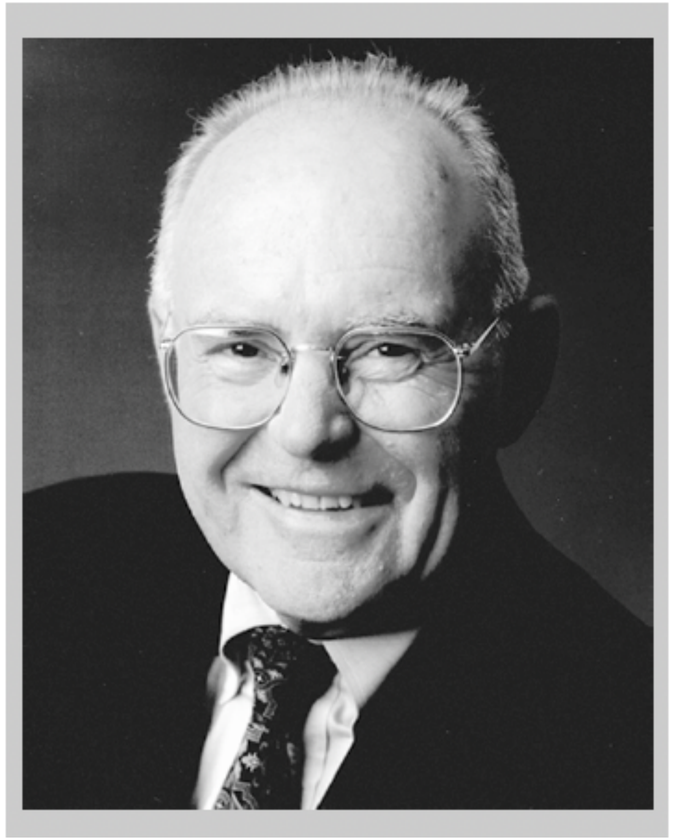
CMOS Level View







A Bit of GK




YouWall.com
Thank you