EC2.101 – Digital Systems and Microcontrollers

Lecture 6 – Boolean algebra

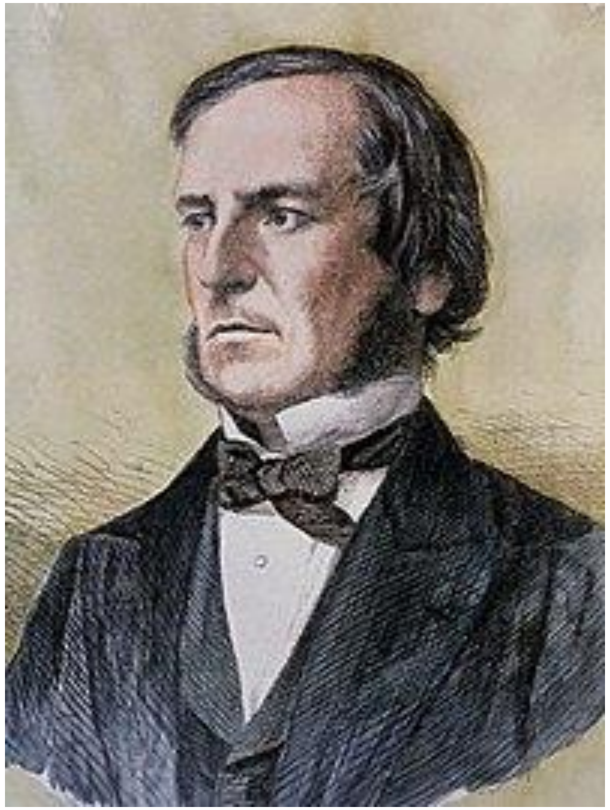
Chapter 2

Binary logic

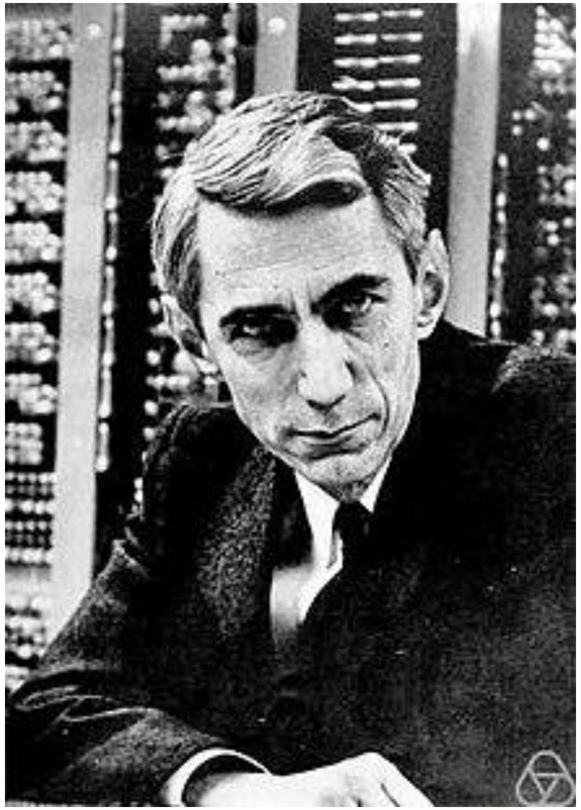
- Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning
- The two values the variables assume may be called by different names (*true* and *false, yes* and *no*, etc.), but for our purpose, it is convenient to think in terms of bits and assign the values 1 and 0
- Binary logic consists of binary variables and a set of logical operations
- The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc., with each variable having two and only two distinct possible values: 1 and 0

Boolean algebra

- The system for formalization of binary logic came much before their applications in electronics/computers
- Boolean algebra was introduced by George Boole in his first book The Mathematical Analysis of Logic (1847)
- In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced switching algebra as a way to analyze and design circuits by algebraic means in terms of logic gates.



George Boole



Claude Shannon

Basic operations

- **NOT:** This operation is represented by a prime (sometimes by an overbar). For example, $z = x'$ (or $z = \bar{x}$); meaning that *z* is what *x* is not
- In other words, if *x* = 1, then *z* = 0, but if *x* = 0, then *z* = 1
- The NOT operation is also referred to as the *complement* operation, since it changes a 1 to 0 and a 0 to 1, i.e., the result of complementing 1 is 0, and vice versa

- **AND:** This operation is represented by a dot or by the absence of an operator
- For example, $z = x \cdot y$ or $z = xy$
- The logical operation AND is interpreted to mean that *z* = 1 if and only if *x* = 1 and *y* = 1; otherwise *z* = 0

- **OR:** This operation is represented by a plus sign. For example, $z = x + y$, meaning that *z* = 1 if *x* = 1 or if *y* = 1 or if both *x* = 1 and *y* = 1. If both *x* = 0 and *y* = 0, then *z* = 0

Basic operations

- We make a table of all possible values of the variables and the results of these operations (truth table)

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	x'
0	1
1	0

Binary logic

- *Binary logic* is different from binary numbers although it uses some of the same symbols
- In binary logic, we assume that variables can have ONLY two values – no other values are possible
- In binary numbers variables can have higher values or fraction or negative values, however, that is not the case in binary logic
- For example: in binary numbers, $(1+1 = 10)_2$, however, in binary logic, 1+1 = 1 because two trues make a true
- There are formal rules and proofs for many of the statements we make in binary logic
- In modern circuits, logic gates are used to perform binary logic using a variety of complex architectures

Formalization of Boolean algebra

- Boolean algebra, like any other deductive mathematical system, may be defined with a set of elements, a set of operators, and a number of unproved axioms or postulates
- A *operator* defined on a set S of elements is a rule that assigns, to each pair of elements from S, a unique element from S
- As an example, consider the relation $a * b = c$. We say that * is an operator if it specifies a rule for finding c from the pair (a, b) and also if $a, b, c \in S$
- However, * is not an operator if $a, b \in S$, and if $c \notin S$.

Postulates of Boolean algebra – Closure

- A set S is closed with respect to an operator if, for every pair of elements of S, the operator specifies a rule for obtaining an element of S
- For example, the set of natural numbers $N = \{1, 2, 3, 4, \ldots\}$ is closed with respect to the operator **+** by the rules of arithmetic addition, since, for any $a, b \in N$, there is a unique $c \in N$ such that $a + b = c$
- The set of natural numbers is *not* closed with respect to the operator **–** by the rules of arithmetic subtraction, because $2 - 3 = -1$ and $2, 3 \in N$, but $(-1) \notin N$
- *The Boolean logic structure is closed with respect to* NOT, AND *and* OR *logic operations*

Postulates of Boolean algebra – Associative law

- The operator * on a set S is said to be *associative* whenever $(x * y) * z = x * (y * z)$ for all $x, y, z, \in S$
- In case of real numbers, the multiplication and addition operations are associative while subtraction and division are not
- Similarly in binary logic, the operators AND and OR are associative
- Thus, x AND (y AND z) is the same as (x AND y) AND z
- Also, x OR (y OR z) is the same as (x OR y) OR z

Postulates of Boolean algebra – Commutative law

- The operator * on a set *S* is said to be *commutative* whenever $x * y = y * x$ for all $x, y \in S$
- In case of real numbers, the multiplication and addition operations are commutative, while subtraction and division are not
- Similarly in binary logic, the operators AND and OR are commutative
- Thus, x AND y is the same as y AND x
- Also, x OR y is the same as y OR x

Postulates of Boolean algebra – Identity

- A set S is said to have an *identity element* with respect to an operation $*$ on S if there exists an element $e \in S$ with the property that $e * x = x * e = x$ for every $x \in S$
- *Example:* The element 0 is an identity element with respect to the operator + on the set of integers $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$, since $x + 0 = 0 + x = x$ for any $x \in I$
- The set of natural numbers, N, has no identity element w.r.t the operator +, since 0 is excluded from the set
- In Boolean logic, 0 is the identity element for OR operation and 1 is the identity element for AND operation

Postulates of Boolean algebra – Distributive

- If * and & are two operators on a set S, * is said to be *distributive* over & whenever $x * (y \& z) = (x * y) \& (x * z)$
- In normal algebra, multiplication is distributive over addition
- In Boolean logic, the operator AND (·) is distributive over OR (+); that is,
  $$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
- Also, the operator OR (+) is distributive over AND (·); that is,
  $$x + (y \cdot z) = (x + y) \cdot (x + z)$$
- This is counter intuitive!
- An easy way to prove the distributive law is the make a table of all possible values of the variables and their results

Postulates of Boolean algebra – Distributive

$$x\,.\,(y+\ z) =\ (x.y) +\ (x\,.\,z)$$

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$y+z$	$x\cdot(y+z)$
0	0
1	0
1	0
1	0
0	0
1	1
1	1
1	1

$x\cdot y$	$x\cdot z$	$(x\cdot y)+(x\cdot z)$
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
1	0	1
1	1	1