CS 221
Logic Design

Fall 2021

By Wessam El-Behaidy & Salwa Osama

Remember our Rules

BINARY CODES
&
BOOLEAN ALGEBRA

Lecture 2

BINARY CODES

- Digital systems:
  - Use signals that have 2 distinct values and circuit elements that have 2 stable states
  - Represent and manipulate not only binary numbers but also many other discrete elements of information
- Any discrete element of information that is distinct among group of quantities can be represented with a *binary code*.
- However, it must be realized that binary codes merely change the symbols not the meaning of the elements of information that they represent.

BINARY-CODED DECIMAL (BCD)

Decimal Symbol	0	1	2	3	4	5	6	7	8	9
BCD Digit	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Note: 1010, 1011, 1100, 1101, 1110, and 1111 are **INVALID CODE!**

- It is important to realize that BCD numbers are decimal numbers and not binary numbers. The only difference that decimals use symbols 0-9 and BCD use 0000-1001
- A number with k decimal digits will require 4k bits in BCD
- Ex: $(185)_{10}$= $(0001\ 1000\ 0101)_{BCD}$ =$(1011\ 1001)_2$

OTHER DECIMAL CODES

- Binary codes for decimal digits require a minimum of 4 bits per digit
- Many different codes can be formulated by arranging 4 bits into 10 distinct combinations

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi-nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Weighted Codes

- BCD , 2421 code and 8,4,-2,-1 code are examples of *weighted codes.*
- In weighted code, each bit position is assigned a weighted factor
- Ex:

$0110_{BCD} = 6_{10}$

because $8*0 + 4*1 + 2*1 + 1*0 = 6$

$1101_{2421} = 7_{10}$

because $2*1 + 4*1 + 2*0 + 1*1 = 7$

$0110_{8,4,-2,-1} = 2_{10}$

because $8*0 + 4*1 + (-2)*1 + (-1)*0 = 2$

THE GRAY CODE

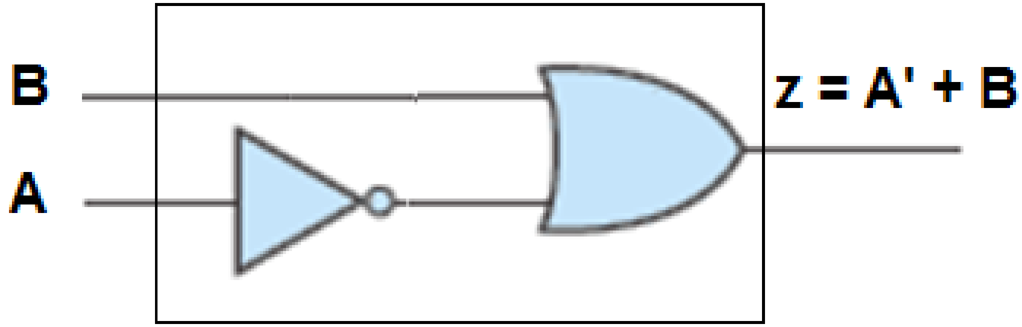
- The Gray code is <u>unweighted</u> and is not an arithmetic code.
  - There are no specific weights assigned to the bit positions.
- Important: the Gray code exhibits only a single bit change from one code word to the next <u>in sequence</u>.
  - This property is important in many applications

Boolean Algebra

So, we can represent the logic circuit with an algebraic equation and use Boolean algebra to simplify this circuit

REMEMBER THIS EXAMPLE

- Draw a logic gate circuit of **A’ +B** and get their truth table



A	B	A’	Z= A’ +B
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Algebraic Manipulation

(a) F1 = $x'y'z + x'yz + xy'$ (b) F2 = $xy' + x'z$

Term
Each term requires a **gate**

3 terms in (a): **$x'y'z$, $x'yz$, xy'**
2 terms in (b): **xy', $x'z$**

Variable
Each variable designates an input to the gate

3 variables in both (a) and (b):
x, y, and z

Literal
Is a single variable within a term, in complemented or uncomplemented form

8 literals in (a): **x', y', z, x', y, z, x, and y'**
4 literals in (b): **x, y', x', and z**

Reducing the number of terms, the number of literals or both in a Boolean function ➔ simpler circuit

BOOLEAN ALGEBRA

- It is equivalent to the binary logic presented heuristically in sec. 1.9
- It is a mathematical methods that simplify circuits.
  - Less components (less gates)
  - Reduce cost
- The cost of the circuits is an important factor addressed by designers.

Boolean Algebra

- In 1854,
  - **George Boole** developed an algebraic system called *Boolean algebra*
- In 1904,
  - **E.V. Huntington** formulated the *formal* definition of *Boolean algebra* → "Huntington postulates"
- In 1934,
  - **C.E. Shanon** introduced two-valued Boolean algebra called *switching algebra*

ANY DEDUCTIVE MATHEMATICAL SYSTEM

It may be defined with :

1) A set of elements S

- Ex: A={1,2,3,4}

2) A set of operators and

- Ex: Binary operator defined on set S , if

  a * b = c a, b, c Є S

➔ * is not a binary operator, if a, b Є S and c not belong to S

3) A number of unproved axioms or postulates

- The most common postulates:
  - Closure
  - Associative law
  - Inverse
  - Identity element
  - Commutative law
  - Distributive law

TWO-VALUED BOOLEAN ALGEBRA

- Also called "Switching algebra"
- Is defined by:
  - A set of two elements, B= {0, 1}
  - Two binary operators:

    + (OR) and . (AND) and

    complement operator (NOT) with rules in table:

x	y	$x \bullet y$
0	0	0
0	1	0
1	0	0
1	1	1

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

x	x'
0	1
1	0

HUNTINGTON POSTULATES FOR TWO-VALUED BOOLEAN ALGEBRA

We will show that Huntington postulates are valid for set B and two binary operators (+ and .)

1. ***Closure*** with respect to the two operators

   It is obvious from the tables, since the result of each operation is either 1 or 0 and 1,0 Є B

N={1,2,3,..}
* is closed to N,
a*b=c, a,b,c ЄN
- is not closed to
N,

2. **Identity:** **(a) x + 0 = 0 + x = x** **(b) x. 1 = 1 . x = x**

x=0	0 **+ 0** = 0	0 **.1** = **1 .** 0 = 0
x=1	1 **+ 0** = **0 +** 1 = 1	1**. 1** = 1

3. **Commutative** **(a) x + y = y + x** **(b) x. y = y . x**

   It is obvious from the symmetry of the binary operator tables

HUNTINGTON POSTULATES FOR TWO-VALUED BOOLEAN ALGEBRA

4. Distributive **(a) x . (y + z) = (x .y) + (x . z)** **(b) x + (y z) = (x + y) . (x + z)**

(a)

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

(b) The *distributive* law of + over . can be shown to hold by means of a *truth* table similar to the one in part (a).

HUNTINGTON POSTULATES FOR TWO-VALUED BOOLEAN ALGEBRA

5. Complement **(a)** $x + x' = 1$ **(b)** $x\,x' = 0$

x=0	$0 + 1 = 1$	$0 .1 = 0$
x=1	$1 + 0 = 1$	$1 .0 = 0$

6. At least two elements $x, y \in B$ such that $x \neq y$.

It is satisfied because the two-valued Boolean algebra has two elements, 1 and 0, with $1 \neq 0$

NOTE:

Huntington postulates do not include the ***associative law***. However, this law holds for two-valued Boolean algebra

POSTULATES AND THEOREMS OF BOOLEAN ALGEBRA

1) Postulates need no proof
2) Theorems must be proven

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

DUALITY PRINCIPLE

- It states that:
  - Every algebraic expression deducible from the postulates of Boolean algebra remains valid if,
    - the operators and identity elements are interchanged.
- If the *dual* is desired,
  - Interchange OR and AND operators
  - Replace 1's by 0's and 0's by 1's.
  - Ex:
    - Postulate 2(a): $x + 0 = 0 + x = x$

      2 (b): $x . 1 = 1 . x = x$
    - $A+B'C$

      Its dual is: $A. (B'+C)$

PROOF USING POSTULATES:
PROOF THEOREM 1 (A) AND (B)

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$

- Theorem 1(a): x + x = x

x + x	= (x + x) .1	Post. 2(b)
	= (x + x) (x + x’)	5(a)
	= x + x x’	4(b)
	= x + 0	5(b)
	= x	2(a)

- Theorem 1(b): x . x = x

x . x	= x x +0	Post. 2(a)
	= x x + x x’	5(b)
	= x (x + x’)	4(a)
	= x . 1	5(a)
	= x	2(b)

By duality

Section 2.4

PROOF USING POSTULATES:

PROOF THEOREM 2 (A) AND 6(A)

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$

- Theorem 2(a): x + 1 = 1

x + 1	= 1. (x + 1)	Post.2(b)
	= (x + x') (x + 1)	5(a)
	= x + x' . 1	4(b)
	= x + x'	2(b)
	= 1	5(a)

- Theorem 2(b): x . 0 = 0
  by duality

- **Absorption theorem:**

Theorem 6(a): x + xy = x

x + xy	= x .1 + xy	Post. 2(b)
	= x (1 + y)	4(a)
	= x (y + 1)	3(a)
	= x . 1	T2(a)
	= x	2(b)

- Theorem 6(b): x (x + y) = x
  by duality

Section 2.4

Proof using Postulates: Proof Theorem 3

- **Theorem 3: (x')' = x**
  - From postulate 5: x + x' = 1 and x . x' = 0
  - So, complement of x' = x = (x')'

PROOF USING TRUTH TABLE:
PROOF THEOREM 5 (A)

- The theorems of Boolean algebra can be shown to be hold true by means of ***truth tables***.
- **DeMorgan's theorem (5a): (x + y)' = x' y'**

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

x'	y'	$x'y'$
1	1	1
1	0	0
0	1	0
0	0	0

Operator precedence

- Operator precedence of Boolean expressions:
  - Parentheses ( )
  - NOT
  - AND
  - OR
- Ex: DeMorgan's law: $(x + y)' = x' y'$
  - Left side $(x + y)'$:
    - Inside the parentheses is evaluated first then
    - The result is complemented
  - Right side $x' y'$:
    - The complement of x and y are evaluated first then
    - The result is ANDed

POSTULATES AND THEOREMS OF BOOLEAN ALGEBRA

Table 2.1
Postulates and Theorems of Boolean Algebra

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

- x (x'+ y)

= xx' +xy dist.

= 0 + xy p5b

= xy p2a

By duality

- x + x'y

= (x + x') (x + y) dist.

= 1 (x + y) p5a

= x + y p2b

- (x + y)(x + y')

= x + xy + xy' + yy' dist.

= x (1 + y + y')+ yy'

= x

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

- xy + x’z + yz = xy + x’z + yz(x + x’) p2b,p5a
  = xy + x’z + xyz + x’yz dist.
  = xy (1 + z) + x’z (1 + y) T2a, p2b

xy + x’z + yz = xy + x’z

Consensus theorem

- By duality,

(x + y)(x’ + z)(y + z) = (x + y)(x’ + z)

COMPLEMENT OF A FUNCTION

- Complement of function F is F ’
- Obtained by one of two methods:
  1) Applying DeMorgan’s law as many times as necessary
  2) Take the dual of the function and complement each literal

EXAMPLES 2.2 & 2.3 (P.64)

Find the complement of the functions:

$F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$

- Using DeMorgan's:

  $F'_1 = (x'yz' + x'y'z)'$
  $= (x'yz')' (x'y'z)'$
  $= (x + y' + z) (x + y + z')$

- Using dual + literal complement

  $F'_1 = x'yz' + x'y'z$
  $= (x+y'+z)(x+y+z')$

→ $F'2 =?$

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

Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$

Thanks

Next week: Boolean Function forms