Digital Logic Design
CSE3201

Lecture 2

Boolean Algebra and Logic Gates

Boolean Algebra (Axiomatic Definition).

- Boolean algebra is an algebraic structure defined by a set of elements *B*, together with to binary operators + and ., provided that the following postulates are satisfied (Huntington).
- Closure with respect to + and .
- Identity element of + = 0, to . is 1
- Commutative wrt + and •
- Distributive over + and . X •(y+z)=(x • Y)+(x • Z) and over . X+(Y • Z)=(X+Y) •(X+Z)
- For every element in B x, there is x' such that x+x'=1 and x • x'=0
- There are at lease 2 different element in B

Two-Valued Boolean Algebra

- The element 1 and 0 operation are OR and AND
- All postulates are satisfied

Duality

- Duality Principle: In Every algebraic expression deducible from the postulates of Boolean Algebra remains valid if the operators if the operators and identity elements are interchanged. In 2-valued Boolean algebra, exchange AND and OR, and 1 and 0

Basic Theorems

Postualte 1	X+0=x	X . 1=1
Postulate 5	X+X'=1	X . X'=0
Theorem 1	X+X=X	X . X =1
Theorem 2	X + 1 = 1	X * 0 = 0
Theorem 3	(x')'=x	
Commutative	X + Y = Y + X	X . Y = Y . X
Associative	X + (Y + Z)=(X + Y) + Z	X(YZ)=(XY)Z
Distributive	X(Y+Z)=(X.Y)+(X.Z)	X + YZ = (X+Y)(X+Z)
DeMorgan	(x+y)'=x'y'	(xy)'=x'+y'
Absorption	x + xy = x	x(x+y)=x

Boolean Function

- Boolean functions can be represented in a truth table that shows the value of the function for all different combination of the input variables.
- An algebraic expression
- Circuit diagram that implements the algebraic expression
- Show as an example $F=x+y'z$ and $F=x'y'z+xz+yz'$

Algebraic manipulation

- We define a *literal* to be a single variable within x'y+zxy is composed of 2 terms and 5 literals.
- By reducing the number of literals, or terms we can obtain a simpler circuit
- $x(x'+y)=xx'+xy=0+xy=xy$
- $(x+y)(x+y')=x+xy+xy'+yy'=x(1+y+y')=x$

Algebraic manipulation

- You can find the complement of a function by taking their duals, and complementing each literal.
- $F=x'yz'+x'y'z$
- Dual of F is $(x'+y+z')(x'+y'+z')$
- Complemnting literals $(x+y'+z)(x+y+z)$
- $F'=(x'yz')'\ (x'y'z)'$
- $F'=(x+y'+z)(x+y+z')$

Representation

- Truth Table
- Cubes
- Binary Decision Diagram BDD

Truth Tables

- A table that lists all the combinations of input variables and the corresponding value of the function

X_1	X_2	X_3	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Cubes

- A cube of the function $Z(x_1, x_2, ..)$ has the form $v_1, v_2, \ldots | V_z$ such that $V_z = Z(v_1, v_2, ..)$
- An implicant of Z is represented as a cube as follows

  set $v_i = 1(0)$ if x_i ($\bar{x}_i$) appears in Z

  set $v_i = X$ if neither x_i nor $\bar{x}_i$ appears in Z

  set $v_z = 1$
- For example, $00X\,|1$ represents $\bar{x}_1 \bar{x}_2$

Cubes

- If a cube q can be obtained from another cube p by replacing one or more X in p by 0 or 1, then we say that p **covers** q
- For example, 00X|1 covers **000|1** and 001|1
- A cube representing a prime implicant in Z (Z') is called a **prime cube**

Cubes

- To find the value of Z for a given input combination, $(v_1, v_2, \ldots)$ search the primitive cubes of Z until we find one whose input part covers v using intersection operator

0 ∩ 1 is inconsistency

∩	0	1	X
0	0	ϕ	0
1	ϕ	1	1
X	0	1	x

Cubes

- The intersection of 2 cubes is cinsistent if all values are compatible (no $\cap$)

1. Form the cube v1 v2 .. |X
2. Intersect this cube with the prime implicants of Z until a consistent intersection is obtained
3. The value of Z is obtained in the right most position

Cubes --Example

X_1	X_2	X_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$Z = \overline{x}_2 + \overline{x}_1 x_3$$

x_1	x_2	x_3	Z
X	1	0	0
1	1	X	0
X	0	X	1
0	X	1	1

Cubes --Example

X_1	X_2	X_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

1 0 1 X

x_1	x_2	x_3	Z
X	1	0	0
1	1	X	0
X	0	X	1
0	X	1	1

Cubes --Example

X_1	X_2	X_3	Z
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

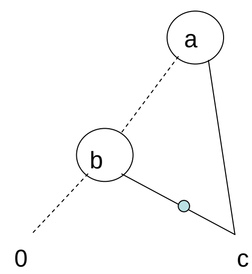
0 1 0 X

x_1	x_2	x_3	Z
X	1	0	0
1	1	X	0
X	0	X	1
0	X	1	1

Binary Decision Diagrams BDD

- A BDD is a graph model of the function of the circuit.
- A simple graph traversal procedure is used to determine the value of the output by sequentially examining values of the input.
- At every node follow left (right) branch if the value of the node is 0(1)
- The value of the function is encountered at exit
- If we encounter an odd number of dots, the exit value is complemented

BDD



Canonical and Standard Forms

- If we have *n* variables, we can have 2^n different combination of these variables either in its normal or complemented form.
- Each of these terms is called a *minterm*
- In a similar matter, n variables added (Ored) can form 2^n *maxterm*
- A boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and taking the OR of all these terms.

Canonical Form

	Minterms		Maxterms	
X y z	term		term	
0 0 1	x'y'z'	m0	x+y+z	M0
0 0 1	x'y'z	m1	x+y+z'	M1
0 1 0	x'yz;	m2	x+y'+z	M2
0 1 1	x'yx	m3	x+y'+z'	M3
1 0 0	xy'z'	m4	x'+y+z	M4
1 0 1	xy'z	m5	x'+y+z'	M5
1 1 0	xyz'	m6	x'+y'+z	M6
1 1 1	xyz	m7	x'+y'+z'	M7

$m'_i = M_j$

Canonical Form

- A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function, then taking the OR of all these terms.
- It could be also expressed as the product of maxterms, where a maxtrm is formed for each combination of the variables that produces a 0 in the function.

Canonical Form

- Example consider the following table
- F=x'y'z'+x'yz'+xy'z'
- $F=m_0+m_2+m_4$
- F'=x'y'z + x'yz + xy'z +xyz' + xyz
- F=(x+y+z')(x+y'+z') (x'+y+z') (x'+y'+z) (x'+y'+z)
- $F=M_1 . M_3 . M_5 . M_6 . M_7$

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

Canonical Form

- Example

$$F(A,B,C)=\sum(1,4,5,6,7)$$
$$F'(A,B,C)=\sum(0,2,3)=m_0+m_2+m_3$$
$$F=\overline{(m_0+m_2+m_3)}=m_0'm_2'm_3'=M_0+M_2+M_3$$
$$F=\prod(0,2,3)$$

Canonical Form

- Express the function *F=A+B'C* in a sum of minterm
- Method 1 make truth table
- Method 2, note that
    - A=A(B+B')=AB+AB'
    - F=AB+AB'+B'C
    - F=AB(C+C')+AB'(C+C')+(A+A')B'C
    - F=ABC+ABC'+AB'C+AB'C'+AB'C+A'B'C
    - $F=m_7+m_6+m_5+m_4+m_5+m_1=\Sigma(1,4,5,6,7)$

Conversion between Canonical Forms

- The complement of the function expressed as the sum of many terms equals the sum of the minterms missing from the original function.

$$F(A,B,C) = \sum(1,4,5,6)$$
$$\overline{F}(A,B,C) = \sum(0,2,3) = m_0 + m_2 + m_3$$
take the complement using De Morgan's
$$F = \overline{(m_0 + m_2 + m_3)} = M_0 M_2 M_3$$

Other Logic Functions

x	**y**	**F0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	**0**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	**1**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	**0**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	**1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Other Logic Functions

F0=0		Null	Constant 0
F1=xy	x.y	AND	
F2=xy'	x/y	Inhibition	X but not y
F3=x		transfer	
F4=x'y	y/x	Inhibition	Y but not x
F5=y		Transfer	
F6=xy'+x'y	$X \oplus y$	Exclusive OR	X, or y but not both
F7=x+y	X+y	OR	
F8=(x+y)'	$X \downarrow Y$	NOR	Not OR

Other Functions

F9=xy+x'y'	$(x \oplus y)'$	Equivalence	X equals y
F10=y'	Y'	Complement	NOT y
F11=x+y'	$X \subset Y$	Implication	If y, then x
F12=x'	X'	Complement	NOT x
F13=x'+y	$X \supset Y$	Implication	If x, then y
F14=(xy)'	$X \uparrow Y$	NAND	NOT AND
F15=1		Identity	Constant 1

Digital Logic Gates

- Explain AND, OR, NOT, Buffer, NAND, NOR, EX-OR, EX-NOR

Negative Logic

FALL 2007 CSE3201 33

Extension to Multiple Inputs

- The extension of AND, and OR is easy
- Consider NOR
- (X ↓ Y) ↓ Z = ((x+y)’+z)’ = xz’ + yz’
- For simplicity we define
- X ↓ Y ↓ Z = (X+Y+Z)’
- X ↑ Y ↑ Z = (XYZ)’

FALL 2007 CSE3201 34

Positive and Negative Logic

- Hardware digital gates are defined in tyerms of signal values *H* and *L*, it is up to the user to define what is *H* and *L*
- *Consider the following table*
- *If we define H=1, L=0*

It is AND (+ve logic)

- *If we define H=0, L=1*

It is OR (-ve Logic)

X	Y	F
L	L	L
L	H	L
H	L	L
H	H	H

Digital Logic Families

- TTL: standard
- ECL: high speed
- MOS: high component density
- CMOS: Low power, currently the dominant logic family