Boolean Algebra and Circuit Design

CSCI 2050U - Computer Architecture

Randy J. Fortier
@randy_fortier

Outline

- Equivalent forms:
    - Truth tables
    - Boolean algebraic expressions
    - Digital circuit diagrams

Truth Tables

CSCI 2050U - Computer Architecture

OntarioTech UNIVERSITY

Truth Table

- A truth table is a table which includes an exhaustive list of all possible input values and the corresponding output values
  - We’ve seen truth tables for basic operations/gates in the previous section
- You can create a truth table for any function that always has the same output for each input combination

Truth Table

- Let’s do an example: A truth table for adding one to an unsigned number

a2	***a1***	***a0***	***b2***	***b1***	***b0***
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Truth Table

- Let's do another example: A truth table for performing twos complement

a2	***a1***	***a0***	***b2***	***b1***	***b0***
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Boolean Algebra

CSCI 2050U - Computer Architecture

Boolean Algebra

- Boolean algebra is just like normal algebra:
  - Variables
  - Operators
  - Axioms - universal truths (e.g. `ab = ba`, `a(b + c) = ab + ac`)
- The key difference is that variables contain Boolean values
  - Their value is either True or False
  - The operators and axioms are also a bit different

Boolean Algebra

- Boolean algebraic operators:
  - NOT
  - AND
  - OR
  - NAND
  - NOR
  - XOR
- We’ve seen all of these (as gates) in the previous section

Boolean Algebraic Expressions

f(x,y,z) = xy + xz + yz

- Both x and y, OR
- Both x and z, OR
- Both y and z

x	**y**	**z**	**xy**	**xz**	**yz**	**f(x, y,**
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Boolean Algebraic Expressions

- Let’s do an example:

f(x,y,z) = xy’ + xz’ + yz

x	**y**	**z**	**xy'**	**xz'**	**yz**	**f(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				

Boolean Algebraic Expressions

- A Boolean algebraic expression is *satisfiable* iff there is at least one combination of variable values that makes the expression true

x	y	xy'
0	0	0
0	1	0
1	0	**1**
1	1	0

Boolean Algebraic Expressions

- A Boolean algebraic expression is *unsatisfiable* iff there is no combination of variable values that makes the expression true

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

Boolean Algebraic Expressions

- A Boolean algebraic expression is *universally valid* iff the expression true for every combination of variable values

x	y	$x'y + x'y' + x$
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**1**

Boolean Algebraic Expressions

- Two Boolean algebraic expressions are *logically equivalent* iff for each combination of variable values, they produce the same output

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

Sum of Products (SOP) Form

- For the output `b` in the truth table given, what input values make it true?

a2	***a1***	***a0***	***b***
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Sum of Products (SOP) Form

- For the output `b` in the truth table given, what input values make it true?
  - Can we come up with a Boolean algebraic expression for this row?

a2	*a1*	*a0*	*b*
0	0	0	0
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	1	1	0
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	1	1	0

Sum of Products (SOP) Form

- For the output `b` in the truth table given, what input values make it true?
  - Can we come up with a Boolean algebraic expression for this row?
    - `a2 = 0, a1 = 0, a0 = 1`
    - `a2’a1’a0`

`a2’a1’a0`

a2	*a1*	*a0*	*b*
0	0	0	0
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	1	1	0
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	1	1	0

Sum of Products (SOP) Form

- For the output `b` in the truth table given, what input values make it true?
  - Extending this to all of the `true` rows...

	a2	**a1**	**a0**	**b**
	0	0	0	0
a2’a1’a0	**0**	**0**	**1**	**1**
a2’a1a0’	**0**	**1**	**0**	**1**
	0	1	1	0
	1	0	0	0
a2a1’a0	**1**	**0**	**1**	**1**
a2a1a0’	**1**	**1**	**0**	**1**
	1	1	1	0

Sum of Products (SOP) Form

- For the output `b` in the truth table given, what input values make it true?
  - `b = a2’a1’a0 + a2’a1a0’ + a2a1’a0 + a2a1a0’`
  - This is called sum of products (SOP) form
    - It is a sum (`+`) of products (`a2a1’a0`)
    - Also called Disjunctive Normal Form (DNF)

a2	***a1***	***a0***	***b***
0	0	0	0
0	**0**	**1**	**1**
0	**1**	**0**	**1**
0	1	1	0
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	1	1	0

Sum of Products (SOP) Form

- Sum of products is very useful, due to:
  - How easily it can be converted into a circuit diagram
  - How easily the SOP form can be deduced from a truth table
    - A truth table for any expression can be straightforwardly created
- It is not very efficient, however
  - It may not be the form that uses the fewest operators/gates
  - We’ll discuss optimization later

Digital Circuits

CSCI 2050U - Computer Architecture

Circuit Diagrams

- Let’s make a circuit that takes a 3-bit twos complement number, and outputs whether or no the number is negative:
  - We can make the truth table by simply looking at the numbers, and determining the correct output

a2	***a1***	***a0***	***Neg***
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Circuit Diagrams

- Recall that the leftmost bit is `1` if the number is negative
  - This makes truth table quite easy to create

a2	***a1***	***a0***	***Neg***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Circuit Diagrams

- Now, let’s determine the SOP for `Neg`:
  - `Neg = a2a1’a0’ + a2a1’a0 + a2a1a0’ + a2a1a0`

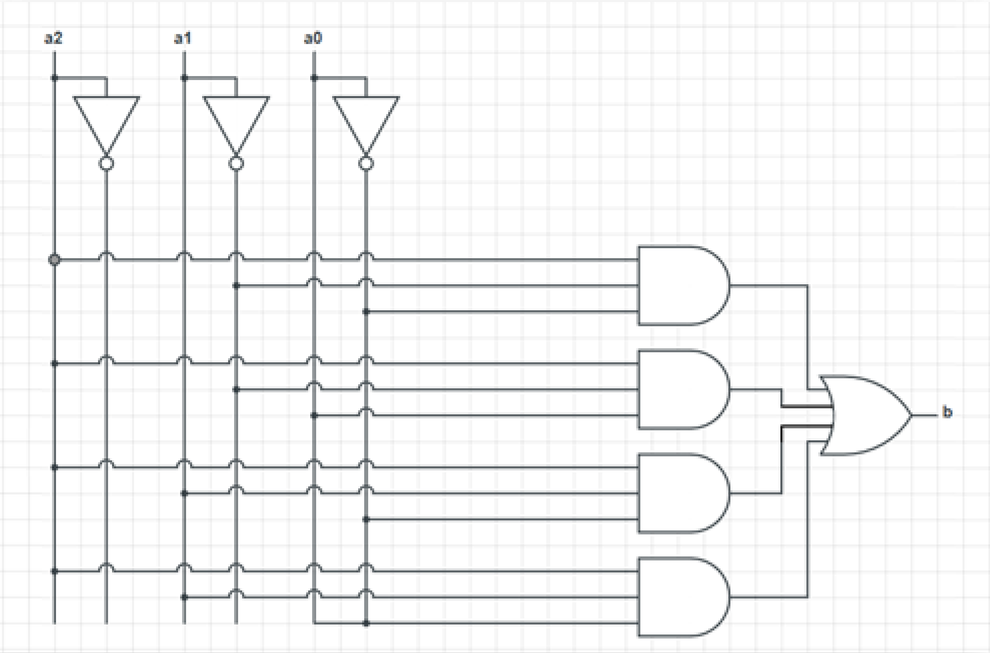
a2	*a1*	*a0*	*Neg*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Circuit Diagrams

- Finally, let’s create the circuit:
  - Neg = a2a1’a0’ + a2a1’a0 + a2a1a0’ + a2a1a0

Circuit Diagrams

- Finally, let’s create the circuit:
  - `Neg = a2a1’a0’ + a2a1’a0 + a2a1a0’ + a2a1a0`



Circuit Diagrams

- Let's make a circuit that takes a 2-bit input and generates the 2-bit output that is the input plus one

a1	***a0***	***b1***	***b0***
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

```
b1 =
b0 =
```

Circuit Diagrams

- Let’s make a circuit that takes a 2-bit input and generates the 2-bit output that is the input plus one

a1	***a0***	***b1***	***b0***
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

```
b1 = a1'a0 + a1a0'
   = a1 XOR a0

b0 =  a1'a0' + a1a0'
```

Arithmetic Exercise

- Create a signed addition overflow recognizer

a7	b7	c7	V
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

$V =$

Wrap-Up

- Equivalent forms:
  - Truth tables
  - Boolean algebraic expressions
  - Digital circuit diagrams

What is next?

- Boolean algebra axioms
- Boolean algebraic simplification