# Boolean Logic

COMP370

Introduction to Computer Architecture

## Difficult Textbook Topics

- How to use the Boolean Identities to simplify logic equations
- I don't fully understand the implementation of the brute force method
- Determining which identity laws were used in each step
- Better examples of sum-of-products and products-of-sum

## Boolean Expression Forms

- You can express a Boolean equation in many ways.
- The Sum of Products form ORs together sub-expressions that are ANDed

F = ABC + A'B'C + AB'

- The Product of Sums form ANDs together sub-expressions that are ORed

F = (A+B+C) * (B' + C')

## Standard Forms

- Using the standard forms can make it easier to use a Boolean equation
- It may be possible to simplify an equation if you do not use the standard forms.

F = A * (B + C)

F = AB + AC

## Converting Truth Table to Expressions

- To create a Sum of Products equation
- For each row where the output is "1", AND together the input variables. NOT the input variables that are zero.
- OR together all of the AND statements.

## Truth Table to Equation

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

F = A'B'C' + A'BC + AB'C + ABC

## Create a SoP Expression

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

## Create a SoP Expression

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

F = A'B'C + AB'C' + ABC

## Converting Truth Table to Gates

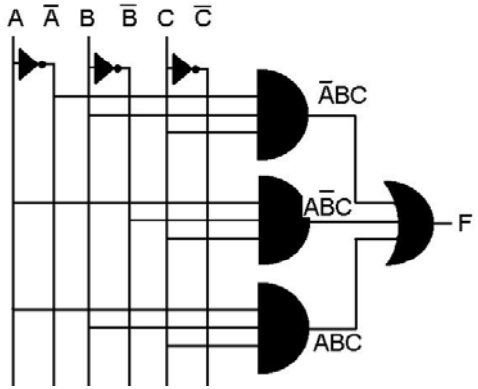
- It is straight forward to convert a truth table or a Boolean expression to a logic gate circuit.
- For the Sum of Products form, use an AND gate for each row. The inputs are the values of the variables in the row.
- Connect the output of the AND gates into an OR gate.

## Be Neat

- It helps to put the input values at the top with vertical lines holding each value.
- Create another set of vertical lines connect from the input lines by a NOT gate.
- Put the gates on the right.
- Draw lines to connect the values or inverse values to the AND gates.

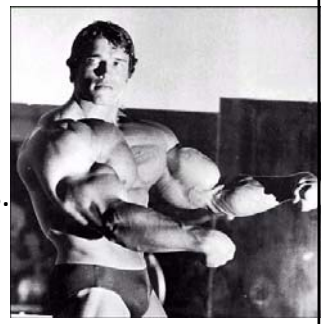
## Creating Logic Gate Circuits

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |




## Why Brute Force?

- The textbook describes this method of creating circuit diagrams as the brute force method.
- It is “brute force” because it does not simplify the circuits.
- We will learn that many Boolean expressions and circuits can be simplified. This makes the implementation easier.



## Inverse Implementation

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

- Sometimes the equation has a lot of terms.
- It may be simpler to solve the inverse and then NOT it.

F = A'B'C' + A'BC' + A'BC + AB'C' + AB'C + ABC' + ABC

F = (A'B'C)'

## Basic Truth Tables

| X | Y | X AND Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| X | Y | X OR Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Boolean Identities

| Name | AND version | OR version |
|---|---|---|
| Identity | X * 1 = X | X + 0 = X |
| Complement | X * X' = 0 | X + X' = 1 |
| Commutative | X * Y = Y * X | X + Y = Y + X |
| Distribution | X * (Y + Z) = X*Y + X*Z | X + (Y*Z) = (X+Y) * (X+Z) |
| Idempotent | X * X = X | X + X = X |
| Null | X * 0 = 0 | X + 1 = 1 |
| Involution | (X')' = X | -- |
| Absorption | X * (X + Y) = X | X + (X * Y) = X |
| Associative | X * (Y * Z) = (X * Y) * Z | X + (Y + Z) = (X + Y) + Z |
| de Morgan | (X * Y)' = X' + Y' | (X + Y)' = X' * Y' |

## Boolean Simplification

- Just like regular algebra, you can use Boolean algebra to simplify an equation.
- The identities can be used to remove terms and variables that do not make a difference.

| | |
|---|---|
| A * ( A' + B ) | **rule** |
| A*A' + A*B | Distribution |
| 0 + A*B | Complement |
| A*B | Identity |

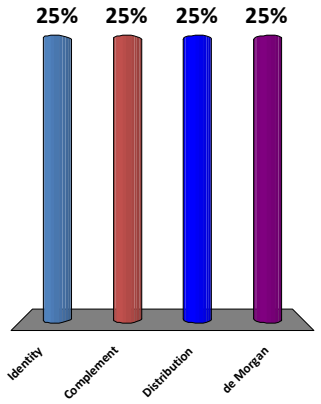
## Simplification Example

| | |
|---|---|
| (A + B) * (A + B') * (A' + B) | |
| [A + (B * B')] * (A' + B) | OR Distribution |
| (A + 0) * (A' + B) | AND Complement |
| A * (A' + B) | OR Identity |
| A * A' + A * B | AND Distribution |
| 0 + A * B | AND Complement |
| A * B | OR Identity |

## What rule is being used?

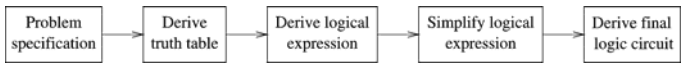
| | |
|---|---|
| (A + B) * (A + B') | |
| A + (B * B') | Distribution |
| A + 0 | ? |
| A | Identity |

1. Identity
2. Complement
3. Distribution
4. de Morgan

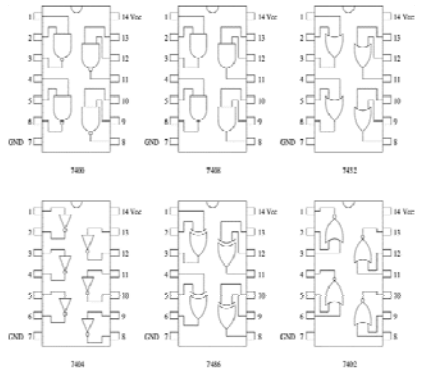



## Logic Circuit Design Process

- A simple logic design process involves
  - Problem specification
  - Truth table derivation
  - Derivation of logical expression
  - Simplification of logical expression
  - Implementation




## TTL Chips



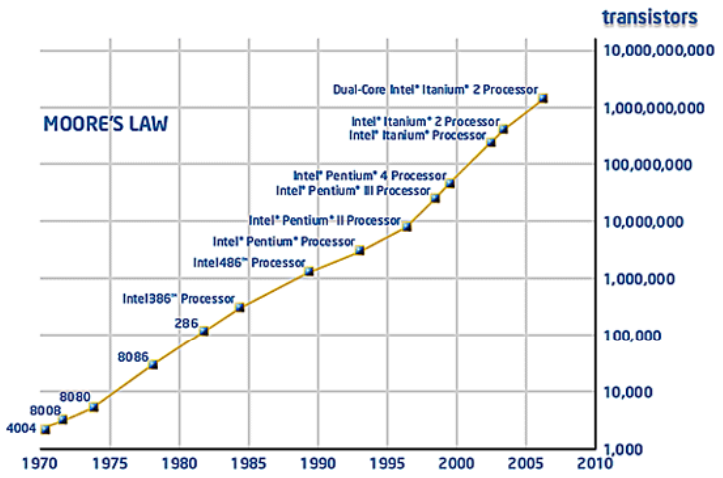
GND is ground
Vcc is power

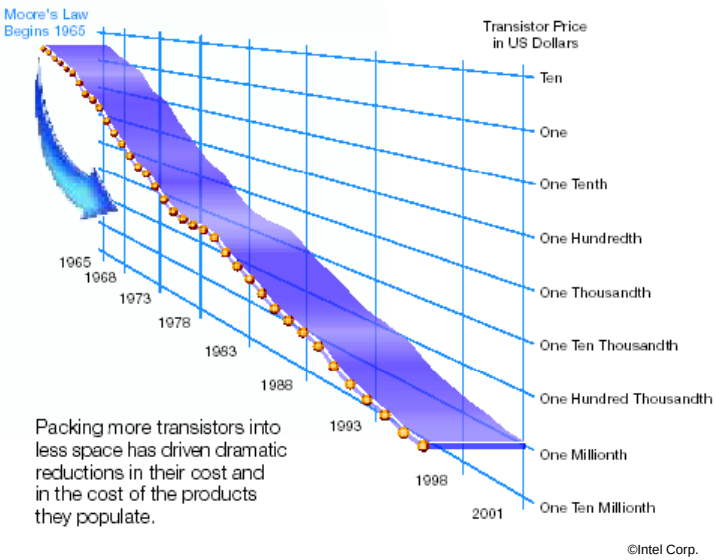
## Integration Levels

- SSI (small scale integration)
  - Introduced in late 1960s
  - 1-10 gates (previous examples)
- MSI (medium scale integration)
  - Introduced in late 1960s
  - 10-100 gates
- LSI (large scale integration)
  - Introduced in early 1970s
  - 100-10,000 gates
- VLSI (very large scale integration)
  - Introduced in late 1970s
  - More than 10,000 gates

## Moore's Law

- Gordon Moore – co-founder of Intel
- Since 1970's development has slowed a little
  - **Number of transistors on a chip doubles every 18 months**
- Cost of a chip has remained almost unchanged
- Higher packing density means shorter electrical paths, giving higher performance
- Reduced power and cooling requirements
- Fewer interconnections increases reliability



## Death of Moore's Law

- Libraries could be filled with articles written over the decades predicting the end of Moore's Law.
- Moore's Law continues to hold
- Obviously Moore's Law must end sometime. Transistors cannot be smaller than an atom.
- Some have written welcoming the end of Moore's law so hardware would stablize.