

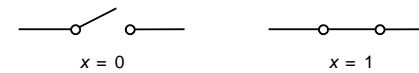
Digital Logic and Boolean Algebra

CIT 595
Spring 2008

Digital Computer



- Modern day computers are made out of electronic device called transistor
 - We are not going to study the physics behind this device
- Transistor act as *switch* (abstraction)
 - Applying a certain input voltage, the transistor acts as a switch i.e. ON(OPEN) or OFF(CLOSE)
 - Voltage is discrete i.e. HIGH (1) or LOW(0)

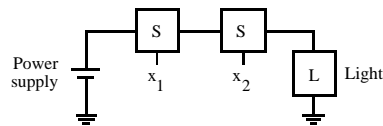


CIT 595

two states of a switch

2/65

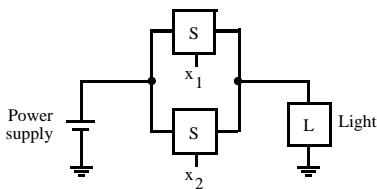
Making Logical Functions from Switches



S- switch (0 – OPEN, 1 – CLOSED)

x_1	x_2	L
0	0	0
0	1	0
1	0	0
1	1	1

AND



OR Function

x_1	x_2	L
0	0	0
0	1	1
1	0	1
1	1	1

CIT 595

3/65

Transistor: Building Block of Computers

- Logically, each transistor acts as a switch
- Combined to implement logic functions
 - AND, OR, NOT
- Combined to build higher-level structures
 - Adder, register, multiplexer, decoder, ...
- Combined to build processor
 - LC-3, Intel Pentium 4, Sun SPARC

CIT 595

4/65

Integrated Circuits

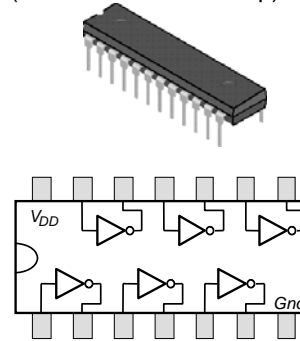
- Integrated Circuits (IC) or contain dozens transistors on single silicon chip
- Very Large Scale Integrated circuits (VLSI) have more than 10,000 components per chip
- Enabled the creation of microprocessors
 - The first was the 4-bit Intel 4004
 - Intel 80 series had around 29,000 transistors

CIT 595

5/65

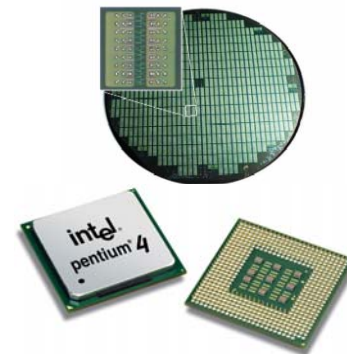
Integrated Circuits (IC) Examples

7404 Chip
(6 Inverters on one chip)



CIT 595

Millions of Transistors on silicon Wafer (VLSI)



Moore's Law (Rule of Thumb)

- Gordon Moore, Intel founder in 1965 said "The density of transistors in an integrated circuit will double every year."
- Contemporary version: "The density of silicon chips doubles every 18 months."
 - Faster computation (also, prices decline)
 - experts predict this trend might continue until ~2020
 - limited when size reaches molecular level

CIT 595

7/65

Moore's Law example



CIT 595

DEC LSI-11, Early 1980's



DEC PDP-11, mid 1970's

These computers were functionally equivalent

8/65

Transistor Density Increase

- **Intel Pentium II processor(1997)**
 - 7 million transistors
- **Intel Pentium III processor(1999)**
 - 28 million transistors
- **Intel Pentium 4 processor(2000)**
 - 42 million transistors
- **Intel Pentium 4 Duo Core(2006)**
 - 291 million transistors

CIT 595

9/65

Logic Gates

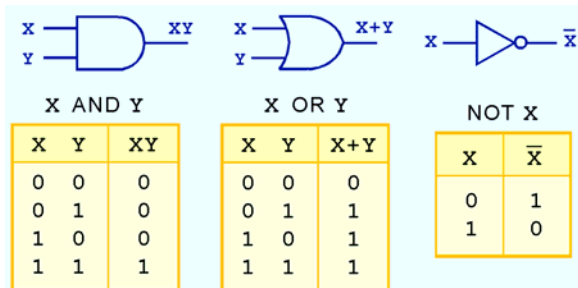
- A gate is an electronic device that produces a result based on one or more input values
 - In reality, gates consist of transistor(s), but digital designers think of them as a single unit
- Integrated circuits contain collections of gates suited to a particular purpose

CIT 595

10/65

Basic Gates

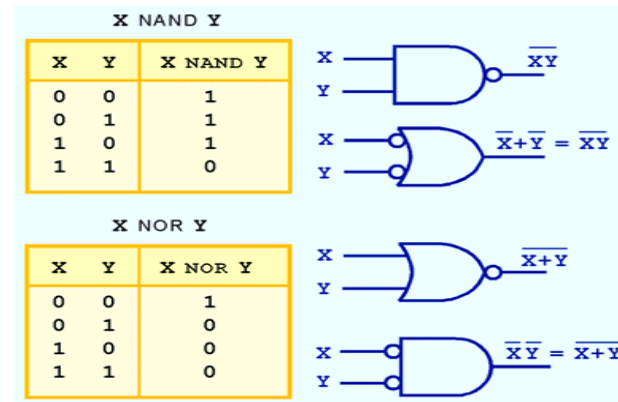
- The three simplest gates are the AND, OR, and NOT gates



CIT 595

11/65

NAND and NOR gates

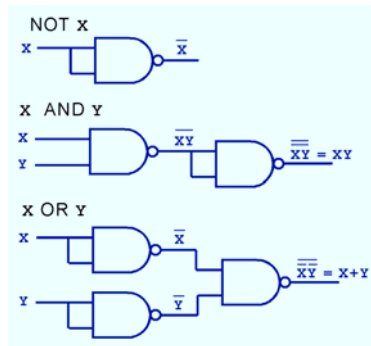


CIT 595

12/65

Universal Gates

- NAND and NOR are known as *universal gates* because they are inexpensive to manufacture and any Boolean function can be constructed using only NAND or only NOR gates

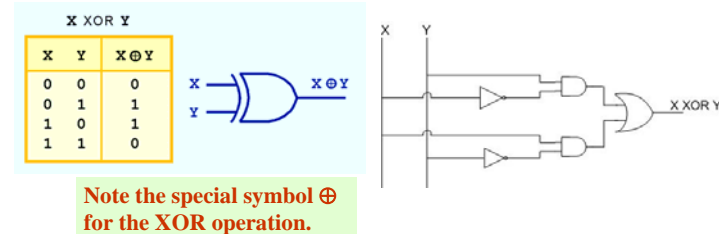


CIT 595

13/65

XOR Gate

- Another very useful gate is the exclusive OR (XOR) gate
- The output of the XOR operation is true only when the values of the inputs differ



CIT 595

14/65

Boolean Algebra

- Computer components
 - So how do I know what combination of basic gates will give the desired output??
- Boolean algebra is a mathematical system for the manipulation of variables that can take on only two values
 - "true" and "false", "1" and "0", or "high" and "low"
 - Inventor George Boole (1815 – 1864)
- In the late 1930s Claude Shannon showed that Boolean algebra provides an effective means of describing circuits built with switches

CIT 595

15/65

Boolean Expression & Function

- Boolean expressions are created by performing "operations on Boolean variables"
 - Common Boolean operators include AND, OR, and NOT
 - E.g. NOT(X AND Y) -> X and Y are Boolean variables
- A Boolean function has:
 - At least one Boolean variable
 - At least one Boolean operator, and
 - At least one input from the set {0,1}
 - It produces an output that is also a member of the set {0,1}
 - E.g. $F(x,y) = \text{NOT}(X \text{ AND } Y)$

CIT 595

16/65

Truth Tables

- Useful depicting information involving logic functions
- Enumerate all possible values for a function
- Function of n variables has
 - at least $n + 1$ columns (n input cols and 1 output col)
 - 2^n rows
 - $> 2^n$ possible input combinations and their respective output
- We will always list the rows of the truth table in numerical order (base 2) of the variables, so the values in the function column completely specify the function

CIT 595

17/65

Boolean Operator

- A Boolean operator can be completely described using a truth table
- The AND operator is also known as a "Boolean product".
 - It is also represented with dot symbol. E.g. $x \cdot y$
- The OR operator is the "Boolean sum".
 - It is also represented with '+' symbol. E.g. $x + y$
- The NOT operation is most often designated by an overbar.
 - It is sometimes indicated by a prime mark (') or an "elbow" (˘) or tilda (~)

X AND Y		
X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y		
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X	
X	\bar{X}
0	1
1	0

18/65

Boolean Operator Precedence

- Boolean operations have rules of precedence
- The NOT operator has highest priority, followed by AND and then OR

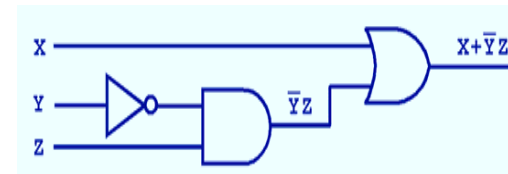
$F(x, y, z) = x\bar{z} + y$					
x	y	z	\bar{z}	$x\bar{z}$	$x\bar{z} + y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

CIT 595

19/65

Gate representation of Boolean Function

$$F(X, Y, Z) = X + \bar{Y}Z$$



CIT 595

20/65

Boolean Function Standardization

- A technique to express unknown function
- Boolean expression is derived using inputs and outputs from a truth table
- The function is said to be expressed in *standardized* or *canonical* form

CIT 595

21/65

Standard or Canonical Form

- There are two canonical forms for Boolean expressions: *sum-of-products* and *product-of-sums*
 - Recall the Boolean product is the AND operation and the Boolean sum is the OR operation.
- In the sum-of-products form, ANDed variables are ORed together.
 - For example: $F(x, y, z) = xy + xz + yz$
- In the product-of-sums form, ORed variables are ANDed together:
 - For example: $F(x, y, z) = (x+y)(x+z)(y+z)$

CIT 595

22/65

Conversion to Sum-of-Products Form

- We are interested in the values of the variables that make the function "true" (i.e. output 1)
- Using the truth table, we list the values of the variables that result in a true value
 - The variables corresponding to row with output 1 are "ANDed"
 - If the variable's input value is 1 then it is written as it is else the complement of that variable is written
- Each group of variables is then "Ored" together

$$F(x, y, z) = x\bar{z} + y$$

x	y	z	$x\bar{z} + y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

CIT 595

23/65

Conversion to Sum-of-Products form (contd..)

- The sum-of-products form for function is:

$$F(x, y, z) = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}\bar{z} + xy\bar{z} + xyz$$

One ANDed Group is known as Minterm

Note: This function is not in simplest terms. It was just show how the function can be rewritten in canonical sum-of-products form.

$$F(x, y, z) = x\bar{z} + y$$

x	y	z	$x\bar{z} + y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

CIT 595

24/65

Conversion to Product-of-Sums

- We are interested in the values of the variables that make the function "false" (i.e. output 0)
- Using the truth table, we list the values of the variables that result in a false value
 - The variables corresponding to row with output 0 are "ORed"
 - If the variable's input value is 0 then it is written as it is else the complement of that variable is written
- Each group of variables is then "ANDed" together

$$F(x, y, z) = x\bar{z} + y$$

x	y	z	$x\bar{z} + y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

CIT 595

25/65

Conversion to Product-of-Sums (contd..)

- The sum-of-products form for function is:

$$(x + y + z) \cdot (x + y + \bar{z}) \cdot (\bar{x} + y + \bar{z})$$

One ORed Group is known as Maxterm

$$F(x, y, z) = x\bar{z} + y$$

x	y	z	$x\bar{z} + y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

CIT 595

26/65

Boolean Function Reduction – Part I

- Digital computers contain circuits that implement Boolean functions
- The simpler we can make a Boolean function, the smaller the circuit that will result
 - Simpler circuits are cheaper to build, consume less power, and run faster than complex circuits
- With this in mind, we always want to reduce our Boolean functions to their simplest form
- One way to do Boolean function reduction is by using Boolean identities

CIT 595

27/65

Boolean Identity Group I

- Most Boolean identities have an AND (product) form as well as an OR (sum) form.

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\bar{x} = 0$	$x + \bar{x} = 1$

CIT 595

28/65

Boolean Identity Group II

- The second group of Boolean identities should be familiar to you from your study of algebra:

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

CIT 595

29/65

Boolean Identity Group III

- If you have studied set theory or formal logic, these laws are also familiar to you

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

CIT 595

30/65

Example using Boolean Identity

- We can use Boolean identities to simplify the function:

$$\begin{aligned}
 &(x + y)(x' + y) \\
 &= xx' + xy + yx' + yy \quad \text{Distributive Law} \\
 &= 0 + xy + yx' + y \quad \text{Inverse \& Idempotent Law} \\
 &= xy + yx' + y \quad \text{Identity Law} \\
 &= y(x + x') + y \quad \text{Distributive Law} \\
 &= y(1) + y \quad \text{Inverse Law} \\
 &= y + y \quad \text{Identity Law} \\
 &= y \quad \text{Idempotent Law}
 \end{aligned}$$

CIT 595

31/65

De Morgan's Law

- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly
- DeMorgan's law provides an easy way of finding the complement of a Boolean function

$$\overline{(xy)} = \bar{x} + \bar{y} \quad \text{and} \quad \overline{(x+y)} = \bar{x}\bar{y}$$

CIT 595

32/65

De Morgan's Law (contd..)

- DeMorgan's law

$$\overline{(xy)} = \bar{x} + \bar{y} \quad \text{and} \quad \overline{(x+y)} = \bar{x}\bar{y}$$

- Can be extended to any number of variables
- Replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs
- Thus, we find the the complement of:

$$\begin{aligned} F(X, Y, Z) &= (XY) + (\bar{X}Z) + (Y\bar{Z}) \\ \bar{F}(X, Y, Z) &= \overline{(XY) + (\bar{X}Z) + (Y\bar{Z})} \\ &= \overline{(XY)} \overline{(\bar{X}Z)} \overline{(Y\bar{Z})} \\ &= (\bar{X} + \bar{Y})(X + \bar{Z})(\bar{Y} + Z) \end{aligned}$$

CIT 595

33/65

Boolean Function Simplification – Part II

- Simplification of Boolean functions leads to simpler (and usually faster) digital circuits
- Simplifying Boolean functions using identities is time-consuming and/or error-prone
- This special section presents an easy, systematic method for reducing Boolean expressions

CIT 595

34/65

Karnaugh Map

- In 1953, Maurice Karnaugh was a telecommunications engineer at Bell Labs
- He invented a graphical way of visualizing and then simplifying Boolean expressions
- This graphical representation, now known as a Karnaugh map, or Kmap, is named in his honor

CIT 595

35/65

Description of Kmap & Terminology

- A Kmap is a matrix consisting of rows and columns that represent the output values of a Boolean function
- Kmap can be of two forms
 - Sum-of-Product (SOP) Form
 - Product-of-Sum (POS) Form

CIT 595

36/65

Kmap for Sum-Of-Product Form

- The output values placed in each cell of the matrix are derived from the “minterms” of a Boolean function
- A *minterm* is a **product** term that contains all of the function’s variables exactly once, either complemented or not complemented

CIT 595

37/65

Minterm Example with Two variables

- For example, the minterms for a function having the inputs x and y are: $\bar{x}\bar{y}$, $\bar{x}y$, $x\bar{y}$, and xy

Minterm	X	Y
$\bar{x}\bar{y}$	0	0
$\bar{x}y$	0	1
$x\bar{y}$	1	0
xy	1	1

Note: If variable input is 1, then it is written as it is else the complement of that variable is written

CIT 595

38/65

Minterm Example with Three variables

- Similarly, a function having three inputs, has the minterms that are shown in this diagram

Minterm	X	Y	Z
$\bar{x}\bar{y}\bar{z}$	0	0	0
$\bar{x}\bar{y}z$	0	0	1
$\bar{x}y\bar{z}$	0	1	0
$\bar{x}yz$	0	1	1
$x\bar{y}\bar{z}$	1	0	0
$x\bar{y}z$	1	0	1
$xy\bar{z}$	1	1	0
xyz	1	1	1

CIT 595

39/65

Kmap Cell using SOP form: Example 1

- A Kmap has a cell for each minterm
- This means that it has a cell for each line for the truth table of a function
- The truth table for the function $F(x,y) = xy$ is shown at the right along with its corresponding Kmap

$F(x, y) = xy$

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X \ Y	0	1
0	0	0
1	0	1

CIT 595

40/65

Kmap Cell using SOP Form: Example 2

- Truth table and Kmap for the function, $F(x,y) = x + y$

$F(x, y) = x + y$		
x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

- This function is equivalent to the OR of all of the minterms that have a value of 1 (Sum of Product Form). Thus:

$$F(x, y) = x + y = \bar{x}y + x\bar{y} + xy$$

x \ y	0	1
0	0	1
1	1	1

CIT 595

41/65

Kmap Simplification for Two Variables using SOP Form

- Of course, the minterm function that we derived from our Kmap was not in simplest terms
 - That's what we started with in this example
- We can, however, reduce our complicated expression to its simplest terms by finding adjacent 1s in the Kmap that can be collected into groups that are "powers of two"

- In our example, we have two such groups
 - Can you find them?

x \ y	0	1
0	0	1
1	1	1

CIT 595

Example 2 42/65

Reduced Expression for Example 2

- In the "green" group (vertical), it does not matter what value x has, hence the group is only dependent on variable y
- Similarly in the "pink" group (horizontal), it does not matter what value y has, the group is only dependent on variable x
- Hence the Boolean function reduces to $x + y$

x \ y	0	1
0	0	1
1	1	1

CIT 595

43/65

Rules for Kmap Simplification using Sum of Products Form (SOP)

The rules of Kmap simplification are:

- Groupings can contain only 1s; no 0s
- Groups can be formed only at right angles; diagonal groups are not allowed
- The number of 1s in a group must be a power of 2 – even if it contains a single 1
- The groups must be made as large as possible
- Groups can overlap and wrap around the sides of the Kmap

CIT 595

44/65

Kmap with Three Variables (SOP form)

- A Kmap for three variables is constructed as shown in the diagram below
- We have placed each minterm in the cell that will hold its value
 - Notice that the values for the yz combination at the top of the matrix form a pattern that is not a normal binary sequence
- A Kmap must be ordered so that each minterm differs only in one variable from each neighboring cell hence 11 appears before 10 – Rule!! (will help simplification)

X \ YZ	00	01	11	10
0	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
1	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

CIT 595

45/65

Kmap with Three Variables (SOP Form)

Note:

- Thus, the first row of the Kmap contains all minterms where x has a value of zero
- The first column contains all minterms where y and z both have a value of zero

X \ YZ	00	01	11	10
0	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
1	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

CIT 595

46/65

Kmap - Three Variable (SOP Form): Example 1

- Consider the function:

$$F(x, y, z) = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz$$

- Its Kmap is given below:

- What is the largest group of 1s that is a power of 2?

X \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

CIT 595

47/65

Kmap - Three Variable (SOP form): Example1

- This grouping tells us that changes in the variables x and y have no influence upon the value of the function: They are irrelevant
- This means that the function,

$$F(x, y, z) = \bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz$$

reduces to $F(x, y, z) = z$

You could verify this reduction with identities or a truth table.

X \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

CIT 595

48/65

Kmap - Three Variable (SOP Form): Example 2

- Now for a more complicated Kmap. Consider the function:

$$F(x, y, z) = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}yz + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xy\bar{z}$$

- Its Kmap is shown below. There are (only) two groupings of 1s.
 - Can you find them?

X \ YZ	00	01	11	10
0	1	1	1	1
1	1	0	0	1

CIT 595

49/65

Kmap - Three Variable (SOP form): Example 2

- In this Kmap, we see an example of a “group that wraps around the sides” of a Kmap.
- This group tells us that the values of x and y are not relevant to the term of the function that is encompassed by the group
 - What does this tell us about this term of the function?
 - It is dependent on \bar{z}

What about the green group in the top row?

X \ YZ	00	01	11	10
0	1	1	1	1
1	1	0	0	1

CIT 595

50/65

Kmap - Three Variable (SOP): Example 2

- The “green group” in the top row tells us that only the value of x is significant in that group.
- We see input value of x is 0 i.e. minterm is complemented in that row, so the other term of the reduced function is \bar{x}
- Our reduced function is: $F(x,y,z) = \bar{x} + \bar{z}$

Recall that we had six minterms in our original function !!
The function is considerably minimized

X \ YZ	00	01	11	10
0	1	1	1	1
1	1	0	0	1

CIT 595

51/65

Kmap Simplification for Four Variables (SOP Form)

- The model can be extended to accommodate the 16 minterms that are produced by a four-input function
- This is the format for a 16-minterm Kmap

WX \ YZ	00	01	11	10
00	$\bar{w}\bar{x}\bar{y}\bar{z}$	$\bar{w}\bar{x}\bar{y}z$	$\bar{w}\bar{x}y\bar{z}$	$\bar{w}\bar{x}yz$
01	$\bar{w}x\bar{y}\bar{z}$	$\bar{w}x\bar{y}z$	$\bar{w}xy\bar{z}$	$\bar{w}xyz$
11	$wx\bar{y}\bar{z}$	$wx\bar{y}z$	$wxy\bar{z}$	$wxyz$
10	$\bar{w}\bar{x}y\bar{z}$	$\bar{w}\bar{x}yz$	$w\bar{x}y\bar{z}$	$w\bar{x}yz$

CIT 595

52/65

Kmap Four Variables (SOP Form) Example

- We have populated the Kmap shown below with the nonzero minterms from the function:

$$F(W, X, Y, Z) = \bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{W}\bar{X}\bar{Y}Z + \bar{W}\bar{X}Y\bar{Z} + \bar{W}\bar{X}YZ + \bar{W}XY\bar{Z} + \bar{W}XYZ + W\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}Z$$

- Can you identify (only) three groups in this Kmap?

Recall the Rules of Simplification

	YZ			
WX	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

CIT 595

53/65

Kmap Four Variables (SOP Form) Example

- The three groups consist of:
 - A purple group entirely within the Kmap at the right
 - A pink group that wraps the top and bottom
 - A green group that spans the corners
- Thus we have three terms in our final function:

$$F(W, X, Y, Z) = \bar{X}\bar{Y} + \bar{X}\bar{Z} + \bar{W}Y\bar{Z}$$

	YZ			
WX	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

CIT 595

54/65

Choosing Kmap Groups

- It is possible to have different pick groups within a Kmap, while keeping the groups as large as possible
- The (different) functions that result from the groupings below are logically equivalent

	YZ			
WX	00	01	11	10
00	1		1	
01	1		1	1
11	1			
10	1			

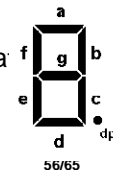
	YZ			
WX	00	01	11	10
00	1		1	
01	1		1	1
11	1			
10	1			

CIT 595

55/65

Don't Care Conditions

- Real circuits don't always need to have an output defined for every possible input
 - For example, some calculator displays consist of 7-segment LEDs. These LEDs can display $2^7 - 1$ patterns, but only ten of them are useful
- If a circuit is designed so that a particular set of inputs can never happen, we call this set of inputs a *don't care* condition
- They are very helpful to us in circuit simplification using Kmap



CIT 595

56/65

Don't Care Example (SOP Form)

- In a Kmap, a don't care condition is identified by an X in the cell of the minterm(s) for the don't care inputs, as shown below
- In performing the simplification, we are free to include or ignore the X's when creating our groups

WX \ YZ	00	01	11	10
00	X	1	1	X
01		X	1	
11	X		1	
10			1	

CIT 595

57/65

Don't Care Example (SOP Form)

- In one grouping in the Kmap below, we have the function:

$$F(W, X, Y, Z) = \bar{W}\bar{X} + YZ$$

WX \ YZ	00	01	11	10
00	X	1	1	X
01		X	1	
11	X		1	
10			1	

CIT 595

58/65

Don't Care Example w/ Different Grouping

- A different grouping gives us the function:

$$F(W, X, Y, Z) = \bar{W}Z + YZ$$

WX \ YZ	00	01	11	10
00	X	1	1	X
01		X	1	
11	X		1	
10			1	

CIT 595

59/65

Don't Care Condition Example

- The truth table of: $F(W, X, Y, Z) = \bar{W}Z + YZ$ is different from the truth table of:

$$F(W, X, Y, Z) = \bar{W}\bar{X} + YZ$$

- However, the values for which they differ, are the inputs for which we have don't care conditions

WX \ YZ	00	01	11	10
00	X	1	1	X
01		X	1	
11	X		1	
10			1	

CIT 595

60/65

Recapping the rules of Kmap Simplification using Sum-Of-Product Form

- Groupings can contain only 1s; no 0s
- Groups can be formed only at right angles; diagonal groups are not allowed
- The number of 1s in a group must be a power of 2 – even if it contains a single 1
- The groups must be made as large as possible
- Groups can overlap and wrap around the sides of the Kmap
- Use don't care conditions when you can

CIT 595

61/65

Kmap using Product-of-Sum (POS) Form

- The output values placed in each cell are derived from the “*maxterm*” of a Boolean function
- A *maxterm* is a sum term that contains all of the function's variables exactly once, either complemented or not complemented

CIT 595

62/65

Maxterm Example

X	Y	Maxterm
0	0	$X + Y$
0	1	$X + Y'$
1	0	$X' + Y$
1	1	$X' + Y'$

Note: If variable input is 0, then it is written as it is else the complement of that variable is written

CIT 595

63/65

Kmap Rules using Product-of-Sum Form

- Groupings can contain only 0s; no 1s
- Groups can be formed only at right angles; diagonal groups are not allowed
- The number of 0s in a group must be a power of 2 – even if it contains a single 0
- The groups must be made as large as possible
- Groups can overlap and wrap around the sides of the Kmap
- Use don't care conditions when you can

CIT 595

64/65

Kmap Conclusion

- Kmaps provide an easy graphical method of simplifying Boolean expressions
- A Kmap is a matrix consisting of the outputs of the minterms of a Boolean function
- In this section, we have discussed 2- 3- and 4- input Kmaps. This method can be extended to any number of inputs through the use of multiple tables