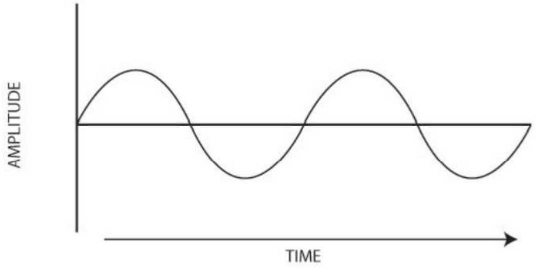
Gates and Combinational Logic Design

LECTURE 12

DR. SK WIJAYASEKARA

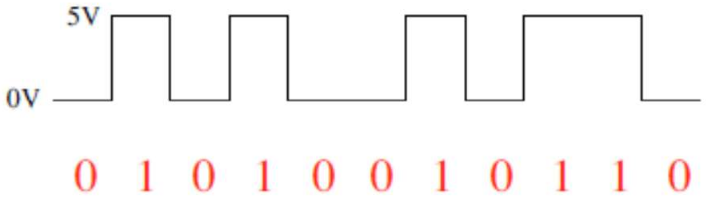
Analog Signals

- Continuously changing signals
- Signal values at different times are useful information



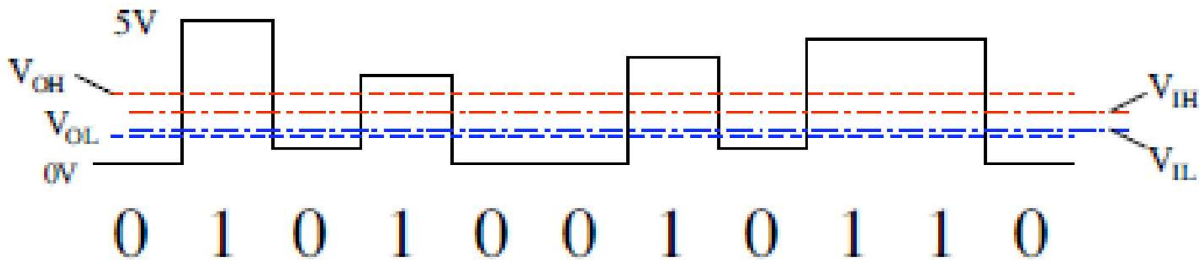
Digital Signals

- Assumes just a few analog values. e.g., 0V and 5V
- Transmit information in the form of a sequence of 0V and 5 segments



Gates and Logic Design

Digital Signals



Generation:

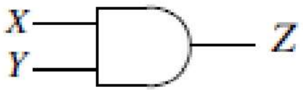
Any voltage higher than VOH is HIGH. (VOH = 2.4V*)
Any voltage lower than VOL is LOW. (VOL = 0.4V)

Detection:

Any voltage higher than VIH is HIGH. (VIH = 2.0V)
Any voltage lower than VIL is LOW. (VIL = 0.8V)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Truth table of AND

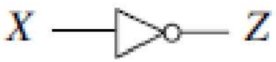


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Truth table of OR

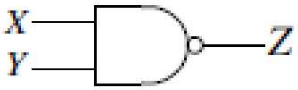
X	Z
0	1
1	0

Truth table of NOT



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Truth table of NAND



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Truth table of NOR

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Truth table of XOR (Exclusive OR)

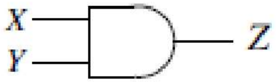
Truth Tables

Boolean Algebra

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Truth table of AND

$$Z = X.Y$$

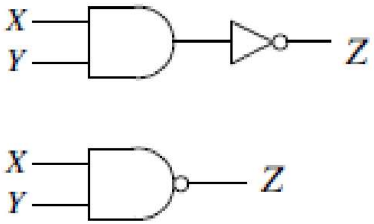


AND gate

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Truth table of NAND

$$Z = \overline{X.Y}$$



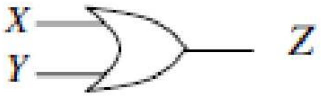
NAND gate

Boolean Algebra

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Truth table of OR

$$Z = X + Y$$

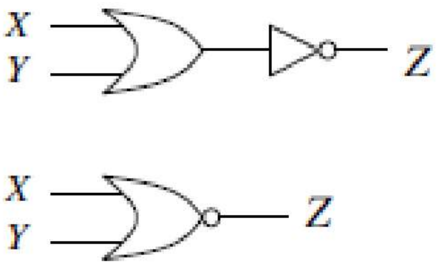


OR gate

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Truth table of NOR

$$Z = \overline{X + Y}$$



NOR gate

Boolean Algebra Simplification

<u>Basic Laws</u>

Commutative: $X+Y = Y+X$

$X.Y = Y.X$

Associative: $X+Y+Z = (X+Y)+Z = X+(Y+Z)$

$X.Y.Z = (X.Y).Z = X.(Y.Z)$

Distributive: $X.(Y+Z) = X.Y + X.Z$

De Morgan's: $\overline{X+Y} = \overline{X} . \overline{Y}$

$\overline{X.Y} = \overline{X} + \overline{Y}$

Expression from Truth Table

- ❑ Min-term corresponds to the product term that give a 1 in the output
- ❑ Output expression is obtained by adding up all min-terms

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

The expression for Z is

$$Z = \overline{X}.Y + X.\overline{Y} + X.Y$$

$$\begin{aligned}
Z &= \overline{X}.\overline{Y} + \overline{X}.Y + X.\overline{Y} \\
&= \overline{X}.(\overline{Y} + Y) + X.\overline{Y} \\
&= \overline{X}.1 + X.\overline{Y} && \leftarrow Y + \overline{Y} = 1 \\
&= \overline{X} + X.\overline{Y} && \leftarrow \overline{X}.1 = \overline{X} \\
&= \overline{(X.(\overline{X} + Y))} && \leftarrow \text{De Morgan' law} \\
&= \overline{(X.\overline{X} + X.Y)} && \leftarrow \text{Distributive} \\
&= \overline{X.Y} && \leftarrow \overline{X}.X = 0
\end{aligned}$$

- Can be simplified down to $Z = X + Y$

Expression from Truth Table

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

The expression for Z is

$$Z = \overline{X}.\overline{Y} + \overline{X}.Y + X.\overline{Y}$$

$$\begin{aligned} Z &= \overline{X}.\overline{Y} + \overline{X}.Y + X.\overline{Y} \\ &= \overline{X}.(\overline{Y} + Y) + X.\overline{Y} \\ &= \overline{X}.1 + X.\overline{Y} && \leftarrow Y + \overline{Y} = 1 \\ &= \overline{X} + X.\overline{Y} && \leftarrow \overline{X}.1 = \overline{X} \\ &= \overline{(X.(\overline{X} + Y))} && \leftarrow \text{De Morgan' law} \\ &= \overline{(X.\overline{X} + X.Y)} && \leftarrow \text{Distributive} \\ &= \overline{X.Y} && \leftarrow \overline{X}.X = 0 \end{aligned}$$

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Truth table of NOR

The expression for Z is

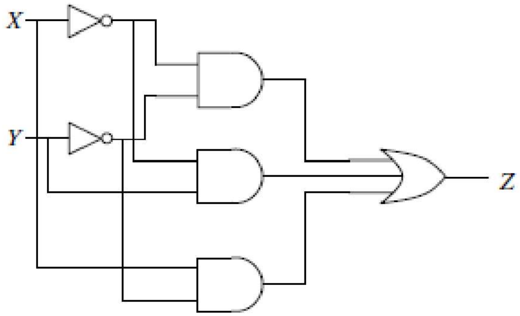
$$Z = \overline{X}.\overline{Y}$$

10

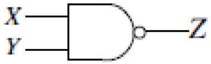
Circuit Realization

Example 1.

$$Z = \overline{X}.\overline{Y} + \overline{X}.Y + X.\overline{Y}$$



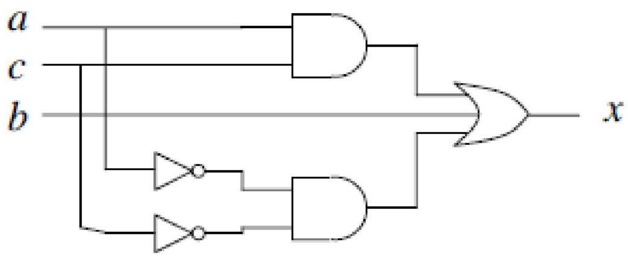
Note: This is same as a NAND gate



Example 2.

- ❑ Derive the circuit for this combinational logic.

$$x = \overline{a}\overline{c} + b + ac$$



Example 3.

- binary code to Gray code conversion

Binary *a*	*b*	*c*	Gray code *x*	*y*	*z*
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

$$x = a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c} + abc$$

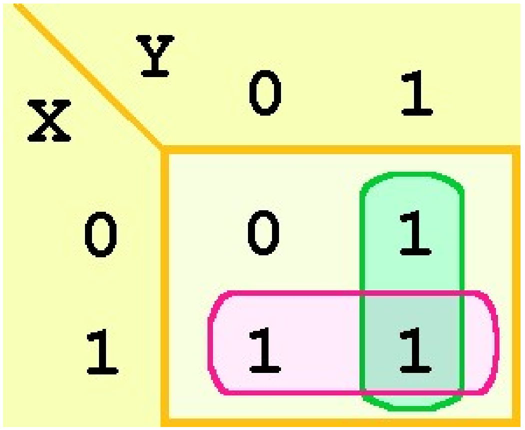
$$y = \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c$$

$$z = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}c + ab\bar{c}$$

- For each of x, y and z, we need a number of inverters, plus 4 AND gates and a multi-input OR gates

Karnaugh Maps - Simplification for Two Variables

- The best way of selecting two groups of 1s form our simple Kmap is shown below.
- We see that both groups are powers of two and that the groups overlap.
- The next slide gives guidance for selecting Kmap groups.



Kmap Simplification for Two Variables

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.

Kmap Simplification for Three Variables

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.

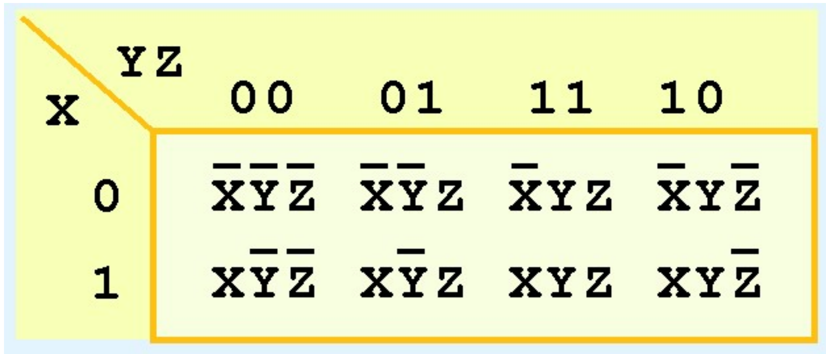
- Notice that the values for the *yz* combination at the top of the matrix form a pattern that is not a normal binary sequence.

X \ YZ	00	01	11	10
0	$\overline{X}\,\overline{Y}\,\overline{Z}$	$\overline{X}\,\overline{Y}Z$	$\overline{X}YZ$	$\overline{X}Y\overline{Z}$
1	$X\overline{Y}\,\overline{Z}$	$X\overline{Y}Z$	XYZ	$XY\overline{Z}$

Kmap Simplification for Three Variables

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.



Kmap Simplification for Three Variables

Consider the function:

$$F(X,Y) = \overline{X}\,\overline{Y}Z + \overline{X}YZ + X\overline{Y}Z + XYZ$$

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

Kmap Simplification for Three Variables

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,

$$F(X,Y) = \overline{X}\,\overline{Y}Z + \overline{X}YZ + X\overline{Y}Z + XYZ$$

reduces to *F(x) = 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

Kmap Simplification for Three Variables

Now for a more complicated Kmap. Consider the function:

$$F(X,Y,Z)=\overline{X}\,\overline{Y}\,\overline{Z}+\overline{X}\,\overline{Y}Z+\overline{X}YZ+\overline{X}Y\overline{Z}+X\overline{Y}\,\overline{Z}+XY\overline{Z}$$

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

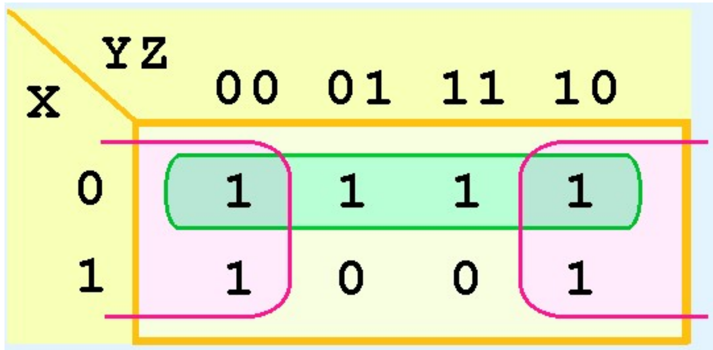
Kmap Simplification for Three Variables

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?

What about the green group in the top row?



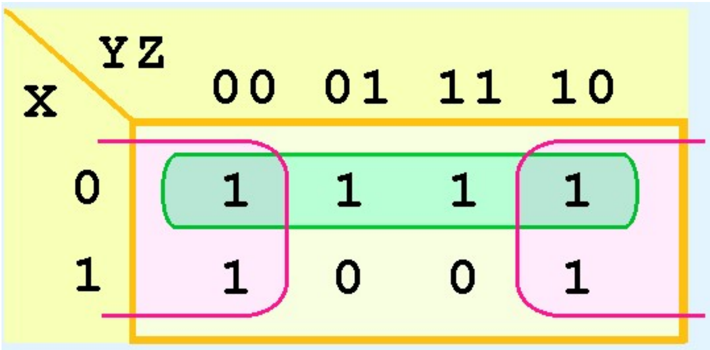
Kmap Simplification for Three Variables

The green group in the top row tells us that only the value of *x* is significant in that group.

We see that it is complemented in that row, so the other term of the reduced function is $\overline{X}$.

Our reduced function is: $F(X,Y,Z) = \overline{X}+\overline{Z}$

Recall that we had six minterms in our original function!



Kmap Simplification for Four Variables

Our 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	$\overline{W}\overline{X}\overline{Y}\overline{Z}$	$\overline{W}\overline{X}\overline{Y}Z$	$\overline{W}\overline{X}YZ$	$\overline{W}\overline{X}Y\overline{Z}$
01	$\overline{W}X\overline{Y}\overline{Z}$	$\overline{W}X\overline{Y}Z$	$\overline{W}XYZ$	$\overline{W}XY\overline{Z}$
11	$WX\overline{Y}\overline{Z}$	$WX\overline{Y}Z$	$WXYZ$	$WXY\overline{Z}$
10	$W\overline{X}\overline{Y}\overline{Z}$	$W\overline{X}\overline{Y}Z$	$W\overline{X}YZ$	$W\overline{X}Y\overline{Z}$

Kmap Simplification for Four Variables

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

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

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

Recall that groups can overlap.

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

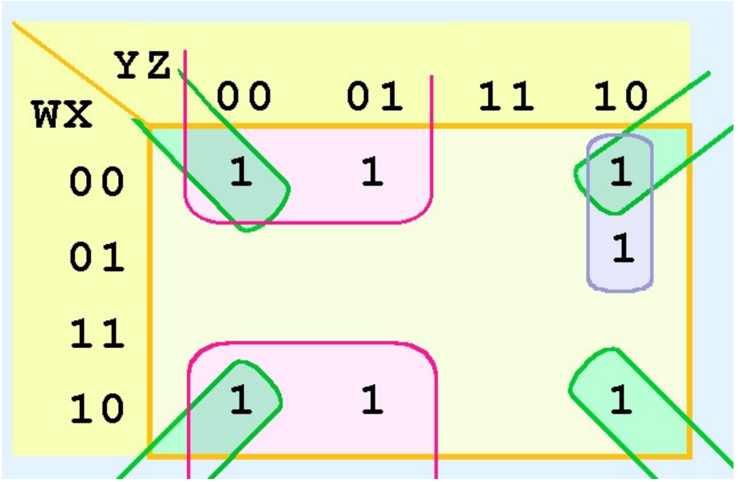
Kmap Simplification for Four Variables

Our 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:

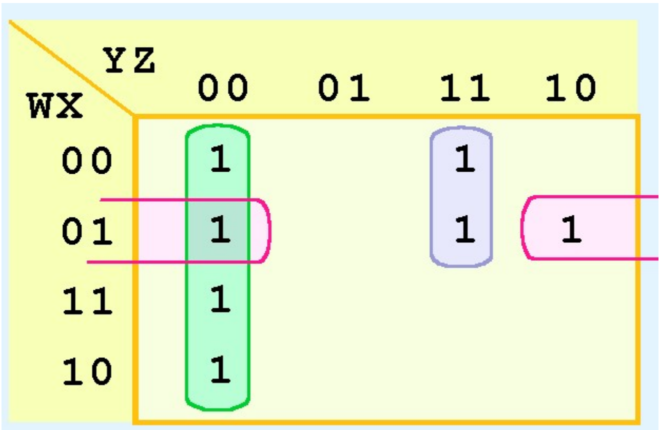
$$F(W,X,Y,Z) = \overline{W}\,\overline{Y} + \overline{X}\,\overline{Z} + \overline{W}Y\overline{Z}$$

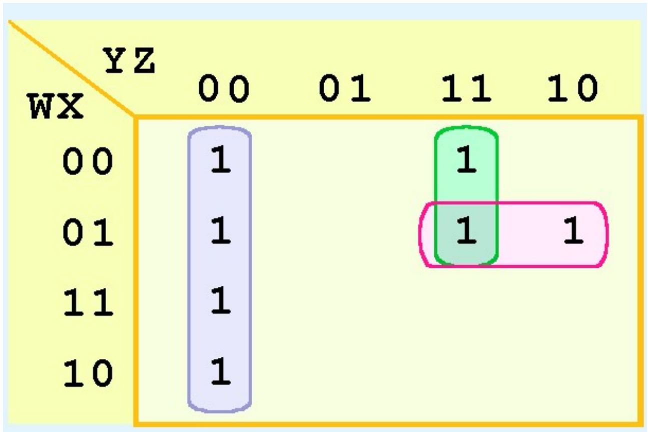


Kmap Simplification for Four Variables

It is possible to have a choice as to how to 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.





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 Kmap circuit simplification.

Don’t Care Conditions

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	

Don’t Care Conditions

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

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

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

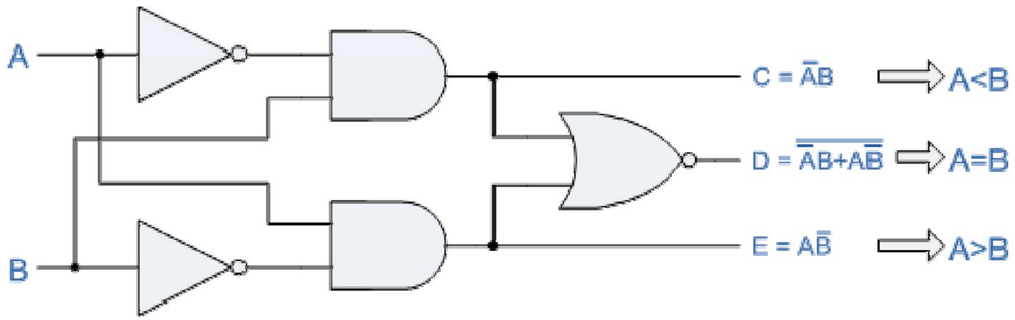
Combinational Logic Design

1. Single Bit Comparator

$A > B, \quad A = B, \quad A < B$

Inputs		Outputs		
B	A	A>B	A=B	A<B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Truth Table



Combinational Circuits

Two Bit Comparator

Inputs				Outputs		
B[1]	B[0]	A[1]	A[0]	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Simplify the minterm expressions to:

$$G = A1 \bullet \overline{B1} + A0 \bullet \overline{B0} \bullet \left(\overline{B1} + A1\right)$$

$$E = \overline{(A1 \oplus B1) + (A0 \oplus B0)}$$

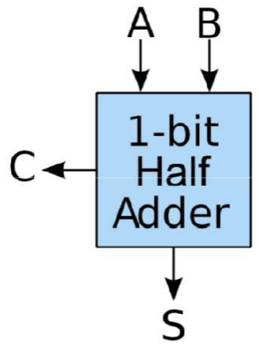
$$L = \overline{A0} \bullet B0 \bullet \left(\overline{A1} + B1\right) + \overline{A1} \bullet B1$$

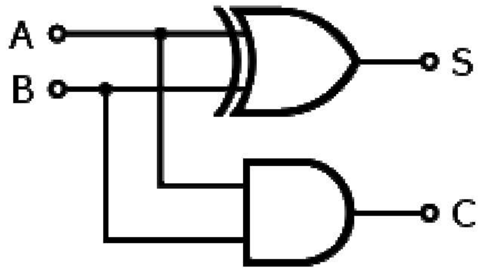
Combinational Circuits – Half Adder

- A half adder adds two one-bit binary numbers *A* and *B*
- It has two outputs, *Sum* and *Carry*

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table





Combinational Circuits – Full Adder

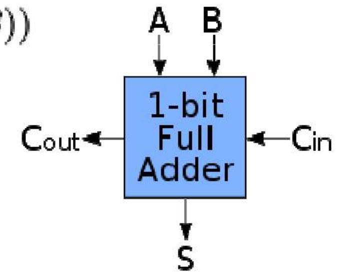
- A full adder adds binary numbers and carried in value
- A one-bit full adder adds three one-bit numbers, *A*, *B*, and C_{in}
- *A* and *B* are the operands, and C_{in} is a bit carried in

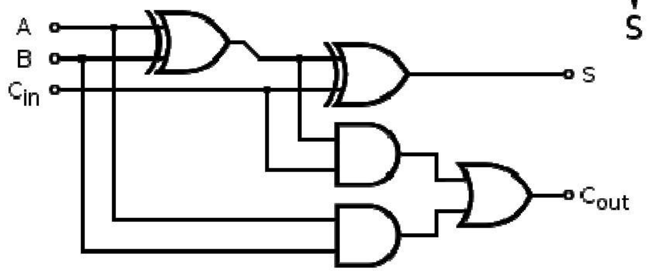
Inputs			Outputs	
A	B	Ci	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth Table

$$C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$$

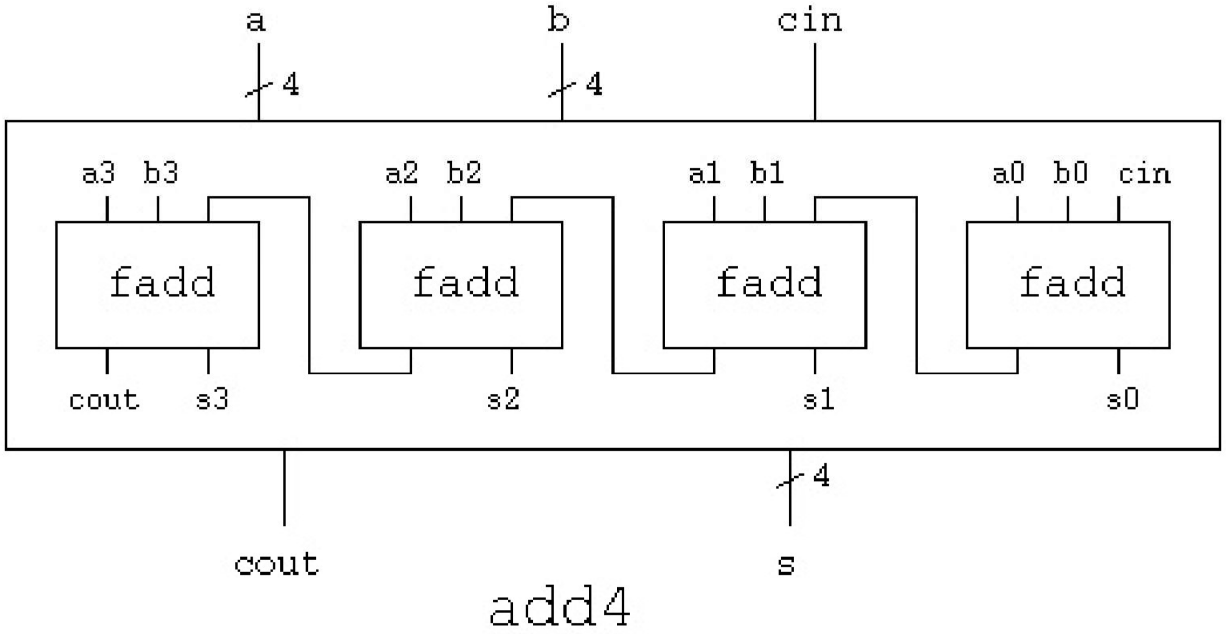
$$S = A \oplus B \oplus C_{in}$$





Combinational Circuits

4bit Full Adder



Thank You