

L 2.1 Primitives Fri. February 8, 2002

AND:

$x$	$y$	$x * y$
0	0	0
0	1	0
1	0	0
1	1	1

OR:

$x$	$y$	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT:

$x$	$\bar{x}$
0	1
1	0

L 2.2 DeMorgan's Theorem Fri. February 8, 2002

Proof of DeMorgan's Theorem:  
 (for 2 variables)

$x$	$y$	$x + y$	$\overline{(x + y)}$	$\bar{x}$	$\bar{y}$	$\bar{x} * \bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

$x$	$y$	$x * y$	$\overline{(x * y)}$	$\bar{x}$	$\bar{y}$	$\bar{x} + \bar{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

L 2.3 Identities Fri. February 8, 2002

Elementary:

$A * 0 = 0$	$A + 1 = 1$
$A * 1 = A$	$A + 0 = A$
$A * A = A$	$A + A = A$
$A * \bar{A} = 0$	$A + \bar{A} = 1$

Commutative:

$A * B = B * A$	$A + B = B + A$
-----------------	-----------------

Distributive:

$A * (B + C) = A * B + A * C$	$A + (B * C) = (A + B) * (A + C)$
-------------------------------	-----------------------------------

Absorption:

$A * (A + B) = A$	$A + (A * B) = A$
-------------------	-------------------

Nameless:

$A * (\bar{A} + B) = A * B$	$A + (\bar{A} * B) = A + B$
-----------------------------	-----------------------------

Consensus:

$(A + B) * (\bar{A} + C) * (B + C)$	$A * B + \bar{A} * C + B * C$
$= (A + B) * (\bar{A} + C)$	$= A * B + \bar{A} * C$

L 2.4 Duality and DeMorgan Fri. February 8, 2002

Equals is by DeMorgan's Theorem



L 2.5 Mass. Stoplight Check Fri. February 8, 2002

Massachusetts Stoplight Check Function  
 (F = 1 implies stoplight consistency: may be working!)

Truth Table: Allowed Combinations of lights lit are red, yellow, green, and red + yellow.	r	y	g	F
	0	0	0	0
	0	0	1	1
	0	1	0	1
	0	1	1	0
	1	0	0	1
	1	0	1	0
	1	1	0	1
	1	1	1	0

L 26 Standard Sum of Products Fri. February 8, 2002  
 Standard Sum of Products (Or of And's)

$$\begin{aligned}
 F &= r * y * g + r * y * \bar{g} + r * \bar{y} * g + r * \bar{y} * \bar{g} \\
 &= r * y * g + \bar{g} * (r * y + r * \bar{y} + r * y) \\
 &= r * y * g + \bar{g} * (r * y + r * (y + \bar{y})) \\
 &= r * y * g + \bar{g} * (r * y + r) \\
 &= r * y * g + \bar{g} * (r + y) \\
 &= r * y * g + r * \bar{g} + y * \bar{g}
 \end{aligned}$$

Or, using DeMorgan:

$$\begin{aligned}
 F &= (\overline{r * y * g}) * (\overline{\bar{g} * (r + y)}) \\
 &= (r + y + g) * (g + r * y)
 \end{aligned}$$

L 2.7 Standard Product of Sums Fri. February 8, 2002

Standard Product of Sums (And of Ors)

$$\begin{aligned}
 F &= (r + y + g) * (r + \bar{y} + \bar{g}) * (\bar{r} + y + \bar{g}) * (\bar{r} + \bar{y} + \bar{g}) \\
 &= (r + y + g) * (\bar{g} + (r + \bar{y})) * (\bar{r} + y) * (\bar{r} + \bar{y}) \\
 &= (r + y + g) * (\bar{g} + (r + \bar{y})) * \bar{r} \\
 &= (r + y + g) * (\bar{g} + r * \bar{y}) \\
 &= (r + y + g) * (\bar{g} + r) * (\bar{g} + \bar{y})
 \end{aligned}$$

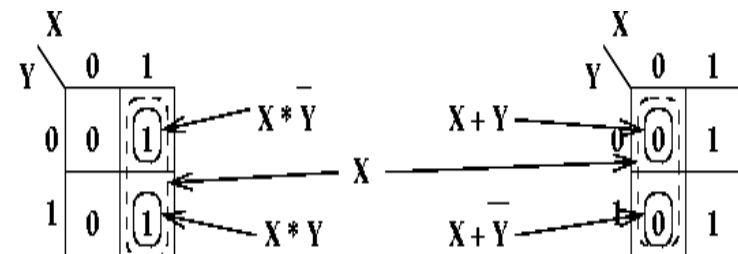
By DeMorgan:

$$F = r * \bar{y} * \bar{g} + g * (r + y)$$

L 2.8 Karnaugh Maps Fri. February 8, 2002

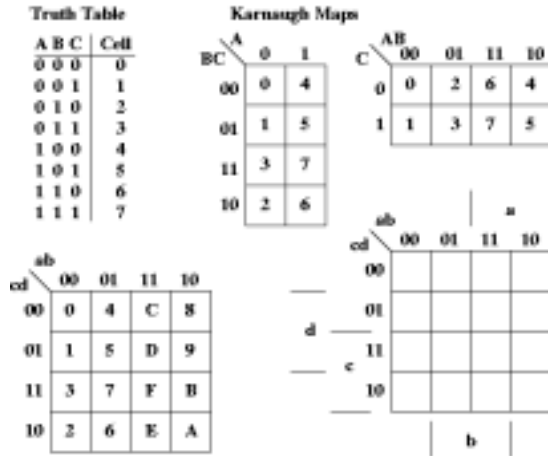
Karnaugh Maps, are:

a simple remapping of truth tables and  
 a graphical means of reducing logic equations.

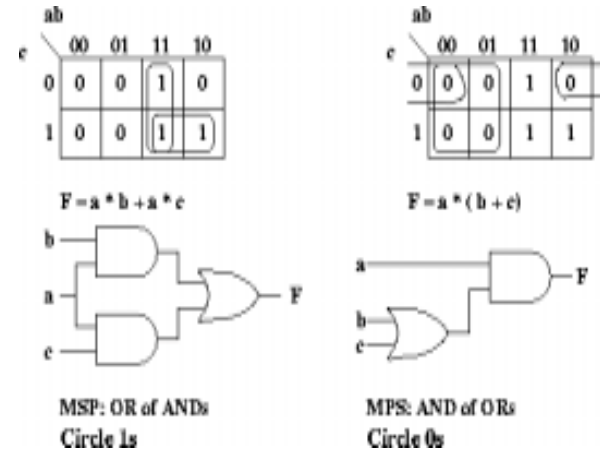


L 2.9 Use of Kmaps Fri. February 8, 2002

Karnaugh Maps are useful for 3-6 variables.  
 (Though HARD for > 4 variables.)  
 Adjacent cells have a one bit change, like a Gray Code.

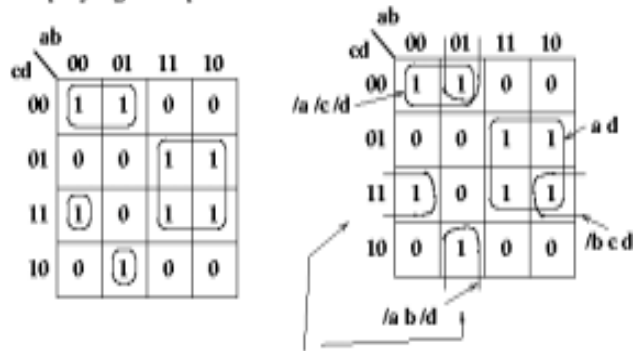


L 2.10 Circling Groups Fri. February 8, 2002



L 2.11 Simple Circles Fri. February 8, 2002

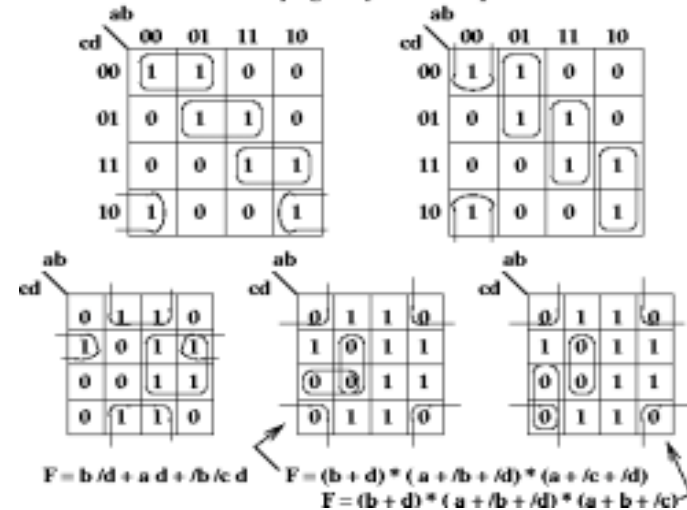
The simplest groups are the largest: this is how we can use K-maps to simplify logical expressions.



There is a temptation to NOT wraparound these cells and, thus, to have a more complicated answer than the correct one.

L 2.12 Uniqueness? Fri. February 8, 2002

Groupings may not be unique!



L 2.13 Don't Cares Fri. February 8, 2002

Don't Cares can simplify things: (impossible inputs, for example).

a b		c d			
		00	01	11	10
c d	00	1	0	0	1
	01	0	X	1	X
	11	1	1	X	0
	10	1	0	0	1

a b		c d			
		00	01	11	10
c d	00	1	0	0	1
	01	0	X	1	X
	11	1	1	X	0
	10	1	0	0	1

$MSP = /b /d + b d + /a c d$      $MPS = (/b + d) * (/a + /c + /d) * (a + c + /d)$

Here abcd = 0101, 1111, and 1001 are don't cares.

Note that  $MSP \neq MPS$ .

L 2.14 XOR Fri. February 8, 2002

Now, there are some functions you can't do very much with:

Like this one: a parity function     $F = \bar{a}b\bar{c} + a\bar{b}\bar{c} + \bar{a}\bar{b}c + abc$

a b		c			
		00	01	11	10
c	0	0	1	0	1
	1	1	0	1	0

$= (\bar{a}b + ab) * \bar{c} + (\bar{a}\bar{b} + ab) * c$   
 $= (a \oplus b) * \bar{c} + (\overline{a \oplus b}) * c$   
 $= (a \oplus b) \oplus c$

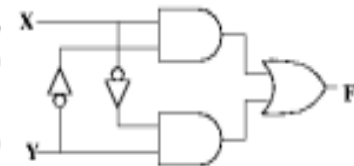
It can be implemented with this (new) function, the exclusive OR.



Exclusive OR

$F = X \oplus Y$

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0



L 2.15 Gate Symbols Fri. February 8, 2002

<p><b>AND</b></p>	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	x	y	f	0	0	0	0	1	0	1	0	0	1	1	1	<p><b>OR</b></p>	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	x	y	f	0	0	0	0	1	1	1	0	1	1	1	1
x	y	f																															
0	0	0																															
0	1	0																															
1	0	0																															
1	1	1																															
x	y	f																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	1																															
<p><b>NAND</b> (Not AND)</p>			<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	x	y	f	0	0	1	0	1	1	1	0	1	1	1	0															
x	y	f																															
0	0	1																															
0	1	1																															
1	0	1																															
1	1	0																															
<p><b>NOR</b> (Not OR)</p>			<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>f</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	x	y	f	0	0	1	0	1	0	1	0	0	1	1	0															
x	y	f																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	0																															

L 2.16 Stoplight Check Function Fri. February 8, 2002

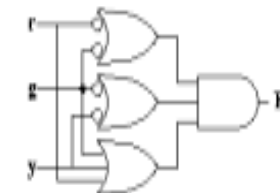
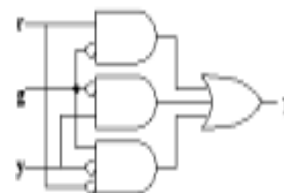
Massachusetts Stoplight Check Function

r y		g			
		00	01	11	10
g	0	0	1	1	1
	1	1	0	0	0

$MSP = r * /g + y * /g + /r * /y * g$

r y		g			
		00	01	11	10
g	0	0	1	1	1
	1	1	0	0	0

$MPS = (/r + /g) * (/y + /g) * (r + y + g)$

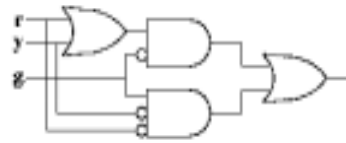
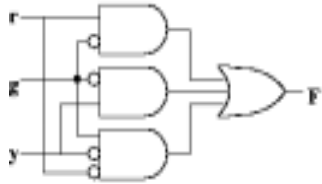


L 2.17 Mass. Stoplight with NANDs Fri. February 8, 2002

Massachusetts Stoplight Check Function

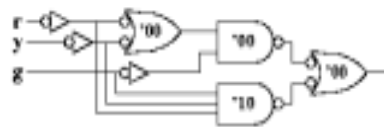
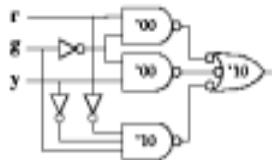
$MSP = r * /g + y * /g + /r * /y * g$

Factored "MSP" =  
 $(r + y) * /g + /r * /y * g$



Done with real NAND gates:

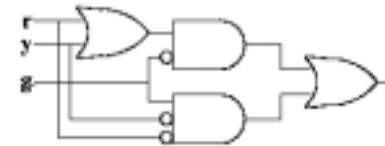
Done with real NAND gates:



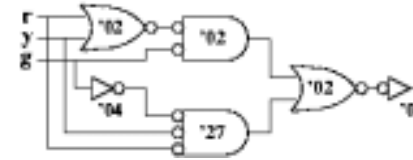
L 2.18 Mass. Stoplight with NORs Fri. February 8, 2002

Massachusetts Stoplight Check Function

Factored "MSP" =  
 $(r + y) * /g + /r * /y * g$

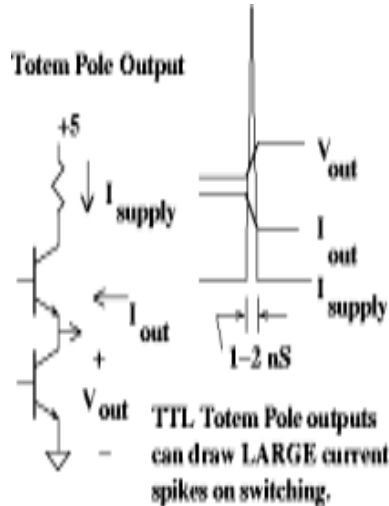


Done with real NOR gates:

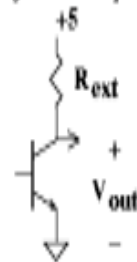


L 2.19 Totem Pole Output Fri. February 8, 2002

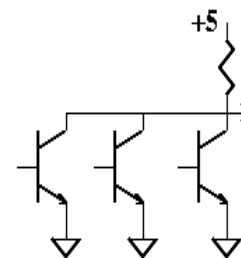
Totem Pole Output



Some outputs are open collector: need a pull-up resistor. Speed is affected by  $R_{ext}$  and by external and junction capacitance.

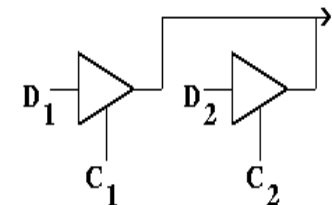


L 2.20 Busses Fri. February 8, 2002



Open collector gates can be wired together like this to make wired ANDs. This is a bus because it can be driven by more than one source. You can't do this with Totem Pole outputs!

By controlling the gates on both transistors of a Totem Pole to be open, a high impedance is created (this is a tri-state). Control inputs  $C_1$  and  $C_2$  are output enables.



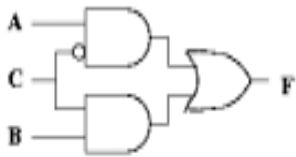
L 2.21

Hazards

Fri. February 8, 2002

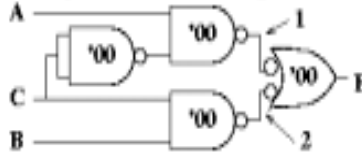
Static Hazards: Consider this function:

$$F = A * \bar{C} + B * C$$

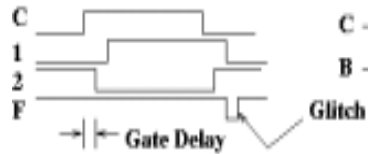


	AB			
C	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Implemented with MSI gates:



A = B = 1

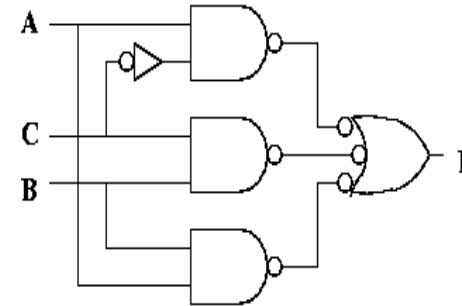


L 2.22

Fixing Hazards

Fri. February 8, 2002

The glitch is the result of timing differences in parallel data paths. It is associated with the function jumping between groupings or product terms on the K-map. To fix it, cover it up with another grouping or product term!



	AB			
C	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$$F = A * \bar{C} + B * C + A * B$$