

# Advanced Boolean Algebra

– Lecture Series by Intel



© 2009 Intel India Pvt Limited. All rights reserved. The information provided in this presentation is intended for the sole use of the recipient and is for educational purposes only. No part of this presentation may be reproduced or transmitted in any form or by any means, including photocopying and recording, without written permission. Permission must also be obtained before any part of this presentation is stored in a retrieval system in any nature. No responsibility can be accepted by Intel India Pvt Limited, the Editorial Board or contributors for action taken as a result of information contained in this presentation. The views expressed in this presentation by the presenter are not necessarily those of the Editorial Board or Intel India Pvt Limited.



# Lecture Series

Basic Boolean Algebra

Oct 28<sup>th</sup>

First hour

Advanced Boolean Algebra

Oct 28<sup>th</sup>

Second hour



# Advanced Boolean Algebra

What will we learn?

- ❑ Boolean Decomposition
- ❑ Cofactors
- ❑ Boolean Derivatives and Applications
- ❑ Boolean Cubes
- ❑ Boolean Quantification and Applications



# Boolean Decomposition: Motivation

## □ Analogy to calculus...

- A complex function can be represented using simple functions
- Ex:  $\exp(x)$  can be expressed in terms of  $1, x, x^2, x^3 \dots$  as

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots,$$

- Next, we can see that the above can be derived from a general form called Taylor series expansion:

$$f(x) = f(0) + \frac{f'(0)x}{1!} + \frac{f''(0)x^2}{2!} + \frac{f'''(0)x^3}{3!} + \dots$$

- Further, if we look at more math, there are other ways of representing an arbitrary  $f(x)$ 
  - If it is periodic then we can use Fourier series
  - Other polynomials like Legendre polynomials

## □ Anything like this for Boolean functions?



# Boolean Decomposition

## □ Shannon Expansion

### ▪ Preliminaries

- Let us say  $f$  is a Boolean function of  $n$  variables  $x_1, x_2, x_3, \dots, x_n$
- Let  $B = \{0,1\}$

Then we write formally:

$$f : B^n \rightarrow B$$

- Often the variables  $x_1, x_2, x_3, \dots, x_n$  are lumped together and called the support of  $f$ , or  $\text{sup}(f)$
- A new function is obtained by setting one of the  $x_i = \text{constant}$ 
  - Ex:  $f(x_1, x_2, \dots, x_i=1, \dots, x_n)$
  - Ex:  $f(x_1, x_2, \dots, x_i=0, \dots, x_n)$
- Result is a **new function** that no longer depends on that variable (in these examples independent of  $x_i$ )



# Boolean Decomposition

## □ Shannon Expansion

### ▪ Preliminaries

- Let us say  $f$  is a Boolean function of  $n$  variables  $x_1, x_2, x_3, \dots, x_n$
- Let  $B = \{0,1\}$

Then we write formally:

$$f : B^n \rightarrow B$$

- Often the variables  $x_1, x_2, x_3, \dots, x_n$  are lumped together and called the support of  $f$ , or  $\text{sup}(f)$
- A new function is obtained by setting one of the  $x_i = \text{constant}$ 
  - Ex:  $f(x, y, z) = xy + xz' + y(x'z + z')$   
 $f(x=1, y, z) = y + z' + yz' = y + z'$ ;  $f(x, y=0, z) = xz'$
- Result is a **new function** that no longer depends on that variable (in these examples independent of  $x_i$ )



# Boolean Decomposition

## □ Shannon Cofactors

- The **shannon cofactor** with respect to  $x_i$  is

$$f(x_1, x_2, \dots, x_i=1, x_n) = f_{x_i} : \text{positive cofactor}$$

$$f(x_1, x_2, \dots, x_i=0, x_n) = f_{\bar{x}_i} : \text{negative cofactor}$$

- The **cofactor**  $f_a$  of  $f$  by a literal  $a=x_i$  or  $a=\bar{x}_i$  is

$$f_{x_i}(x_1, x_2, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

$$f_{\bar{x}_i}(x_1, x_2, \dots, x_n) = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

- $f(x_1, x_2, \dots, x_i=1, x_n)$  is written as  $f|x_i$   
 $f(x_1, x_2, \dots, x_i=0, x_n)$  is written as  $f|\bar{x}_i$



# Boolean Decomposition

## □ Shannon Expansion Theorem

- Given any Boolean function  $f(x_1, x_2, \dots, x_n)$  and any  $x_i$  in the support of  $f()$ ,  $f()$  can be represented as:

$$f(x_1, x_2, \dots, x_n) = x_i f_{x_i} + x_i' f_{x_i'}$$

- Ex: Let  $f(a, b, c) = ab + c$ ;

$$a=1; f_a = b+c;$$

$$a=0; f_{a'} = c;$$

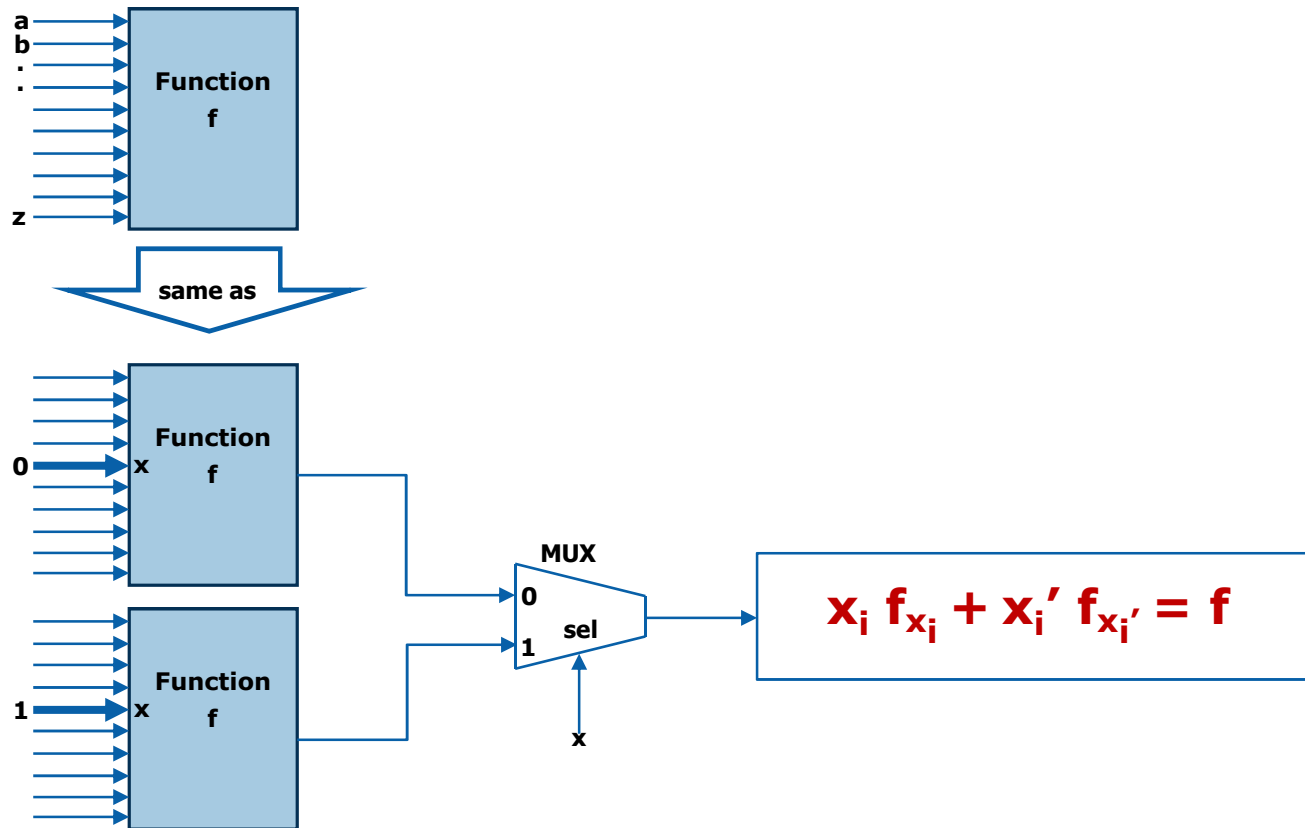
$$af_a + a'f_{a'} = a(b+c) + a'c = ab + c = f(a, b, c)$$



# Boolean Decomposition

## □ Shannon Expansion

$$f(x_1, x_2, \dots, x_n) = x_i f_{x_i} + x_i' f_{x_i'}$$



# Boolean Decomposition

- Shannon Expansion: Multiple Variables
  - Just keep on applying the theorem.

$$f(x, y, z, w) = x f(x=1) + x' f(x=0)$$

$$f(x=1) = yf(x=1,y=1) + y'f(x=1,y=0)$$

$$f(x=0) = yf(x=0,y=1) + y'f(x=0,y=0)$$

$$f(x,y,z,w) = xyf(x=1,y=1) + x'y(x=0,y=1) + xy'f(x=1,y=0) + x'y'(x=0,y=0)$$

Expanded around variables x and y

# Boolean Decomposition

## □ Shannon Expansion: Multiple Variables

### ▪ Notations:

- Write  $f(x_1, x_2, \dots, x_i=1, \dots, x_j=0, \dots, x_n)$  as  $f_{x_i x_j'}$
- Ditto for any number of variables  $x_i, x_j, x_k, \dots$
- The order does not matter:  $(f_x)_y = (f_y)_x = f_{xy}$
- In the previous example

$$f(x,y,z,w) = xyf(x=1,y=1) + x'y(x=0,y=1) + xy'f(x=1,y=0) + x'y'(x=0,y=0)$$

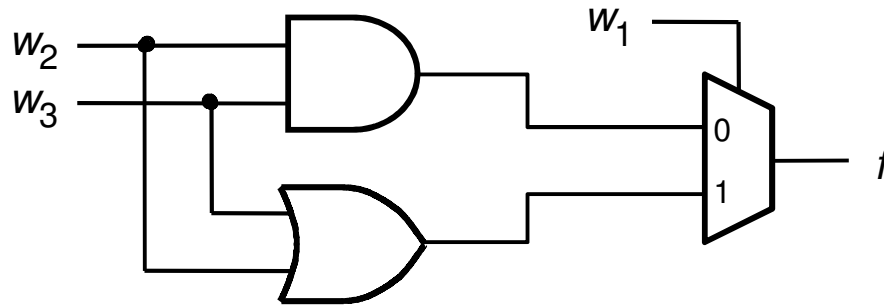
can be written as:

$$f(x,y,z,w) = xyf_{xy} + x'yf_{x'y} + xy'f_{xy'} + x'y'f_{x'y'}$$

# Shannon Expansion Examples

## □ Multiplexers Based on Shannon Expansion

- $f(w_1, w_2, w_3) = w_1' (w_2 w_3) + w_1 (w_2 + w_3)$  is implemented using 2:1 MUX as:



# Shannon Expansion Examples

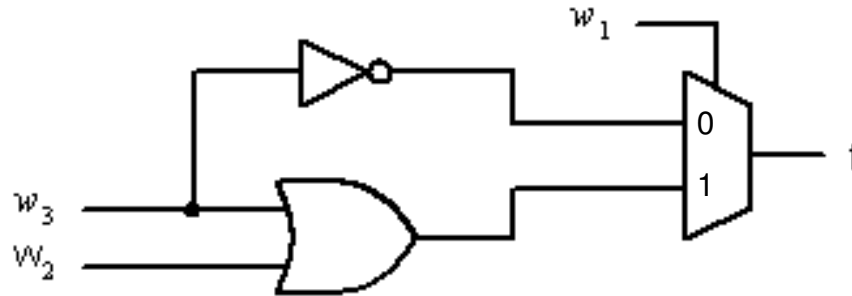
□ Let  $f(w_1, w_2, w_3) = w_1'w_3' + w_1w_2 + w_1w_3$

- Expand on  $w_1$ :

$$f(w_1, w_2, w_3) = w_1' f(0, w_2, w_3) + w_1 f(1, w_2, w_3)$$

$$= w_1' (w_3') + w_1(w_2 + w_3)$$

implemented using 2:1 MUX as:



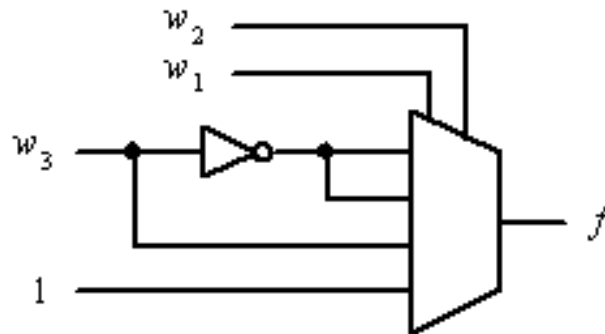
# Shannon Expansion Examples

□ Let  $f(w_1, w_2, w_3) = w_1'w_3' + w_1w_2 + w_1w_3$  (same as previous)

- Expand on both  $w_1$  and  $w_3$ :

$$\begin{aligned} f(w_1, w_2, w_3) &= w_1'w_2'f(0,0,w_3) + w_1'w_2f(0,1,w_3) \\ &\quad + w_1w_2'f(1,0,w_3) + w_1w_2f(1,1,w_3) \\ &= w_1'w_2'(w_3') + w_1'w_2(w_3') + w_1w_2'(w_3) + w_1w_2(1) \end{aligned}$$

implemented using 4:1 MUX as:



# Properties of Cofactors

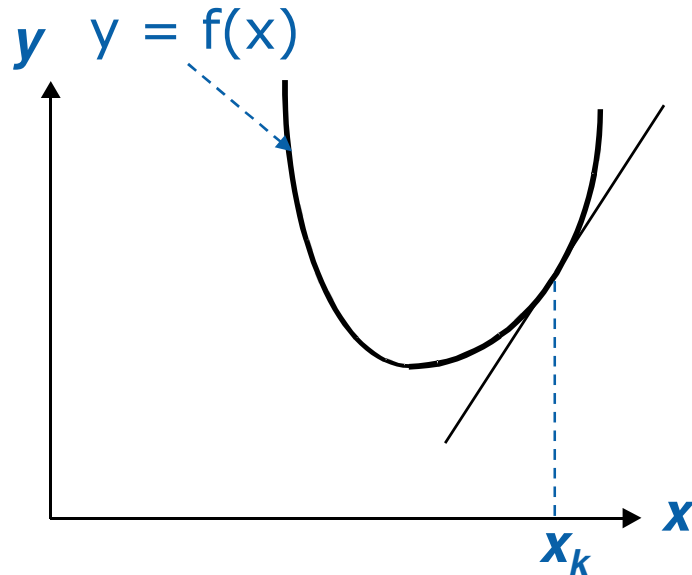
- ❑ Let  $F(x)$  and  $G(x)$  be two Boolean functions, where  $x = (x_1, x_2, \dots, x_n)$
  
- ❑ Complement:
  - $F_{x'} = (F_x)'$  Cofactor of complement is complement of cofactor
  
- ❑ Binary boolean operators
  - $(F \cdot G)_x = F_x \cdot G_x$
  - $(F + G)_x = F_x + G_x$
  - $(F \oplus G)_x = F_x \oplus G_x$
  
- ❑ These are true for ANY binary operator on Boolean functions
  
- ❑ These can often help in getting cofactors for complex formulas

# Properties of Cofactors

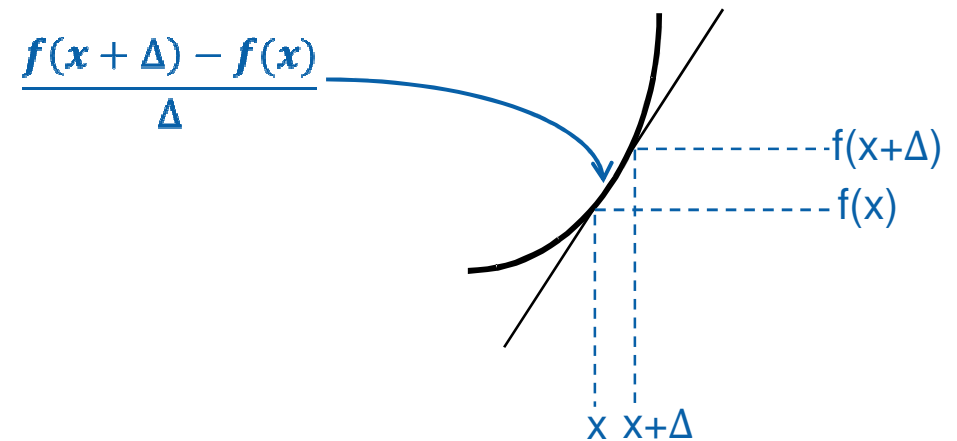
- ❑ Let us consider the combination of cofactors
  
- ❑ Suppose we have  $F(x)$  and we get  $F_x$  and  $F_{x'}$ ,
  - $F_x \oplus F_{x'} = ?$
  - $F_x \cdot F_{x'} = ?$
  - $F_x + F_{x'} = ?$
  
- ❑ These are true for ANY binary operator on Boolean functions
  
- ❑ These can often help in getting cofactors for complex formulas

# Derivative

- Suppose we have  $y = f(x)$

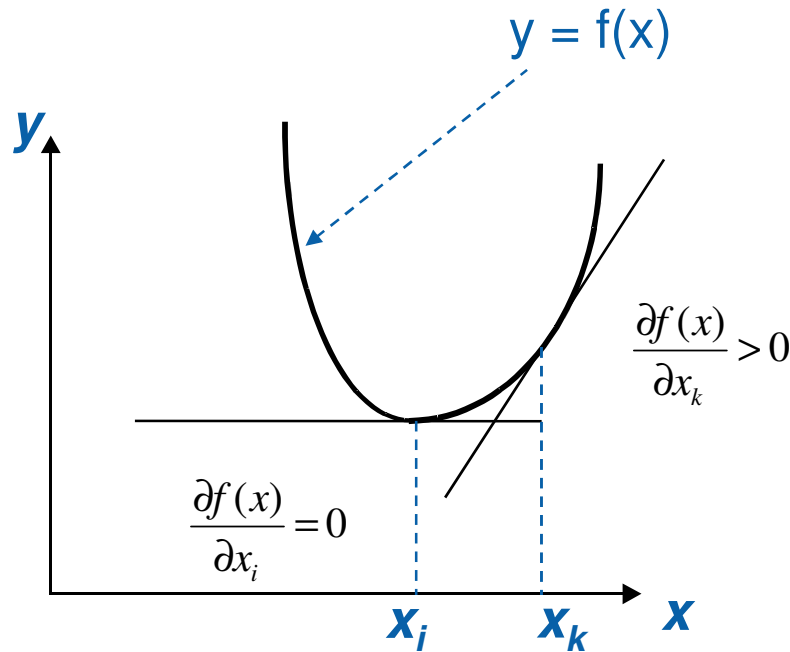


- Defined as slope of curve as a function of point  $x$



- Let  $\Delta$  go to 0 in the limit and we get  $\frac{df(x)}{dx}$

# Boolean Derivatives



- For real-valued  $f(x)$ ,  $\frac{df(x)}{dx}$  tells how  $f$  changes when  $x$  changes
- For 0,1-valued Boolean function,  $x$  cannot be changed by small  $\Delta$
- $x$  can only change  $0 \leftarrow \rightarrow 1$ , but can still check how  $f$  changes with  $x$
- Compares the value of  $f()$  when  $x=0$  against when  $x=1$ ;  
 $\Rightarrow 1$  only if these are different

Real-valued  $f(x)$  :

$$df/dx = \frac{f(x+\Delta) - f(x)}{\Delta}$$

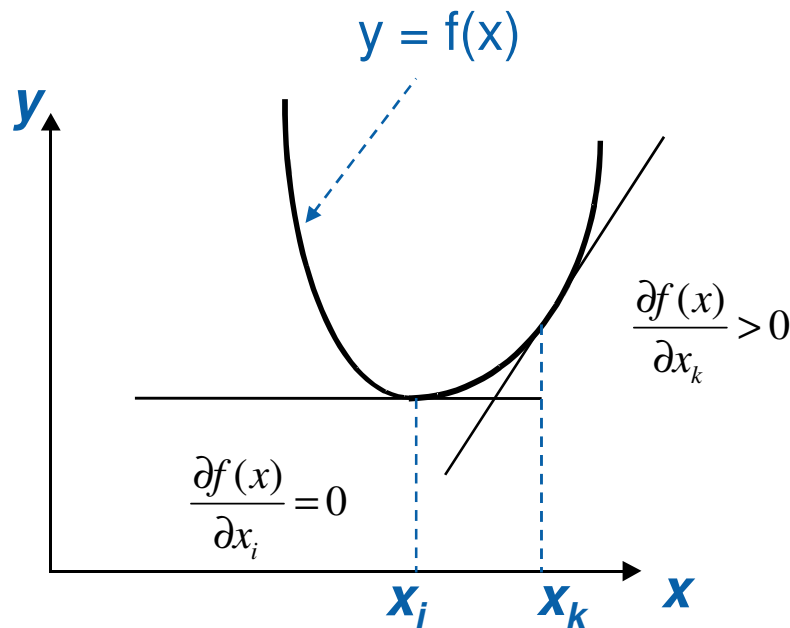
Binary-valued  $f(x)$  :

$$df/dx = f_x \oplus f_{x'}$$

# Boolean Difference

- $df/dx = \text{XOR of the Shannon cofactors with respect to } x$
- For Boolean variables, it is usually written as  $\partial f/\partial x$
- Called the **Boolean Difference** of a function  $f$  w.r.t variable  $x$

Traditional algebra: **speed**



Boolean algebra: **change**

$$x \in \{0,1\}, f(x) \in \{0,1\}$$

$$\frac{\partial f(x)}{\partial x_i} = 1 \quad f(x) \text{ will change if } x_i \text{ changes}$$

$$\frac{\partial f(x)}{\partial x_i} = 0 \quad f(x) \text{ will not change if } x_i \text{ changes}$$

# Boolean Derivatives

□ Boolean function:

$$Y = F(x) = F(x_1, x_2, \dots, x_n)$$

□ Boolean partial derivative:

$$\frac{\partial F(X)}{\partial x_i} = F(x_1, \dots, x_i, \dots, x_n) \oplus F(x_1, \dots, \bar{x}_i, \dots, x_n)$$

$$\frac{\partial F(X)}{\partial x_i} = F(x_1, \dots, x_i = 1, \dots, x_n) \oplus F(x_1, \dots, x_i = 0, \dots, x_n)$$

## Boolean Difference: Properties

- Order of variables does not matter
  - $\partial f / \partial x \partial y = \partial f / \partial y \partial x$
- Derivative of XOR is XOR of derivatives
  - $\partial(f \oplus g) / \partial x = \partial f / \partial x \oplus \partial g / \partial x$
- If function  $f$  is actually constant ( $f=1$  or  $f=0$  for all inputs), then
  - $\partial f / \partial x = 0$  for any  $x$
- If  $f$  is a function of only  $x$  (only changing  $x$  can change  $f$ ) then
  - $\partial f / \partial x = 1$
- If  $f$  is independent of  $x$  (i.e. change  $x$ ,  $f$  never changes), then
  - $\partial f / \partial x = 0$
  - $\partial(f \cdot g) / \partial x = f \cdot \partial g / \partial x$
  - $\partial(f + g) / \partial x = f' \cdot \partial g / \partial x$

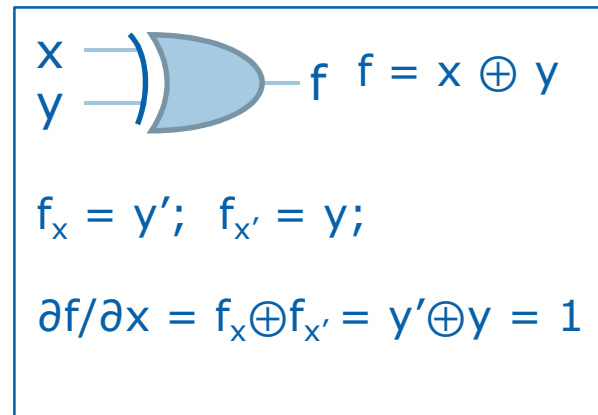
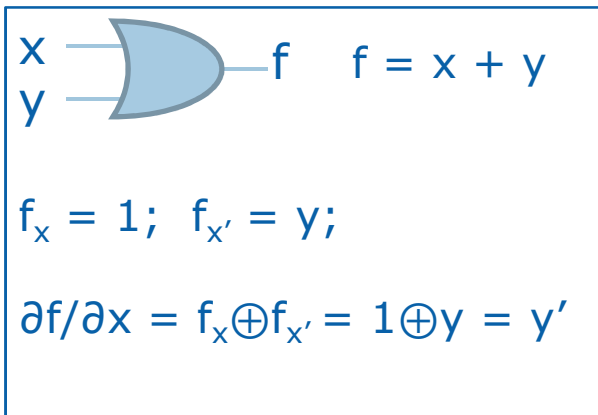
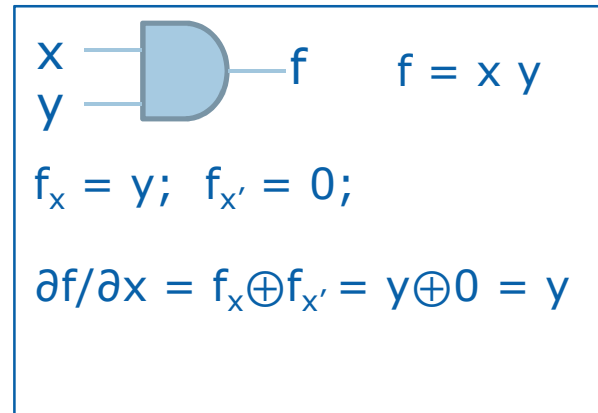
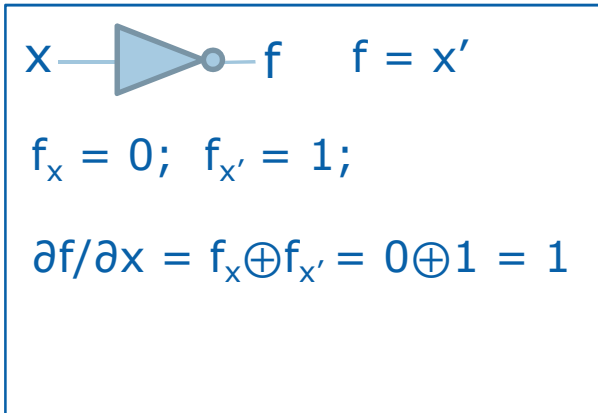
These are "similar" to the regular calculus on real numbers



## Boolean Difference: Properties

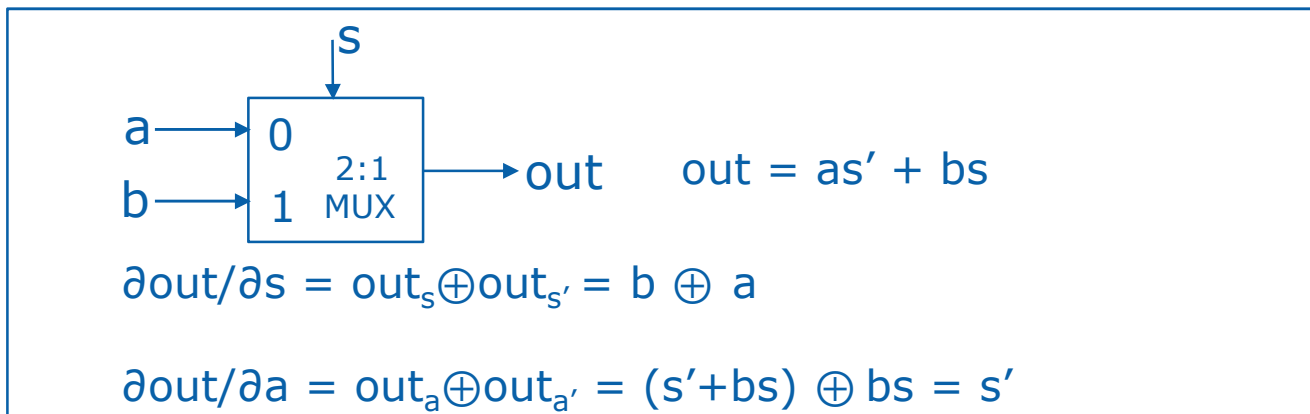
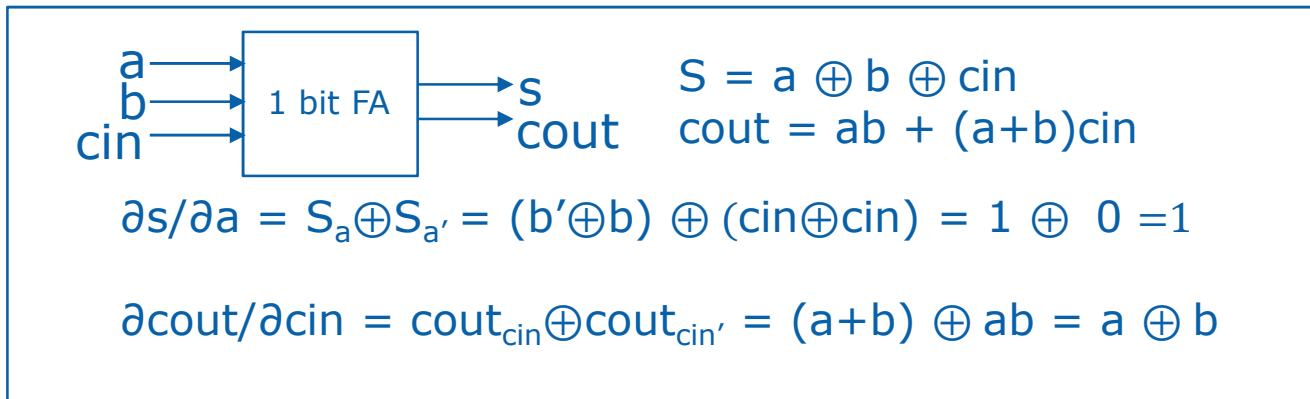
- ❑ Note that AND and OR Boolean operations don't always behave like MULTIPLICATION and ADDITION on real numbers
  
- ❑ Computing  $\partial(f \bullet g)/\partial x$  when  $f$  does depend on variable  $x$ 
  - $\partial(f \bullet g)/\partial x = f \partial g/\partial x \oplus g \partial f/\partial x \oplus (\partial f/\partial x \cdot \partial g/\partial x)$
  
- ❑ Computing  $\partial(f + g)/\partial x$  when  $f$  does depend on variable  $x$ 
  - $\partial(f + g)/\partial x = f' \partial g/\partial x \oplus g' \partial f/\partial x \oplus (\partial f/\partial x \cdot \partial g/\partial x)$

## Boolean Difference: Gate-level View



When  $\partial f / \partial x = 1$ , then  $f$  changes as  $x$  changes

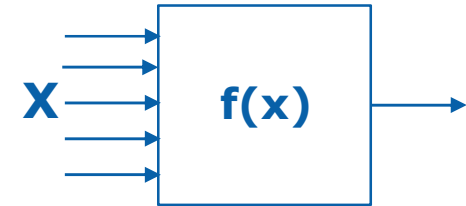
## Boolean Difference: Example



- $\partial f / \partial x$  is a Boolean function, but it does not depend on  $x$

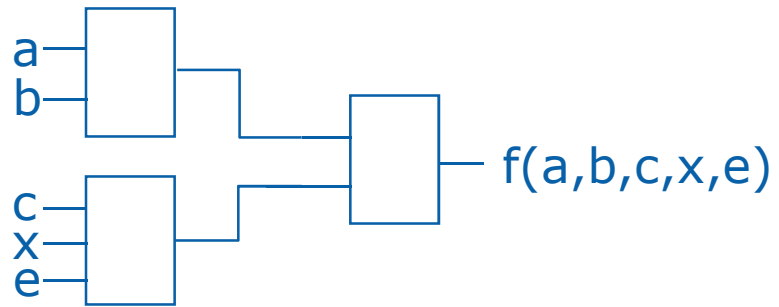
# Boolean Difference Application: Testing

- ❑ Figure out what inputs to apply to figure out the working
- ❑ Can't afford to apply all possible inputs
  - Ex: 50 inputs =  $2^{50}$  = 1000 trillion patterns

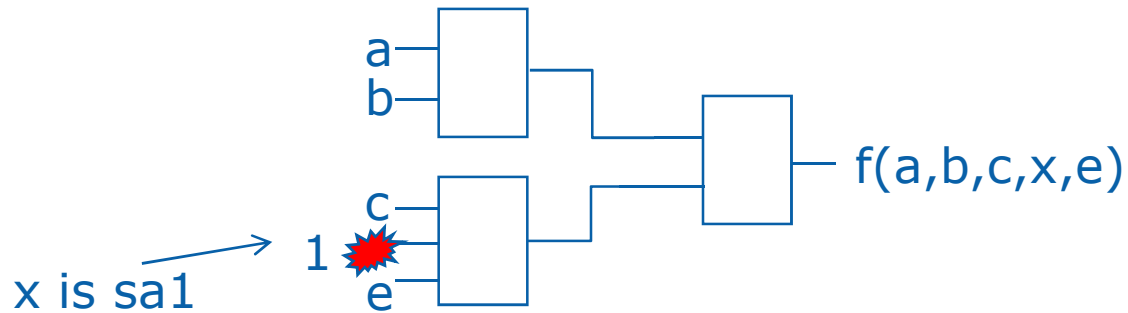


- ❑ We look at the manufacture process and figure out what actually breaks!
  - These are defects
  - Effect of defect is called a fault
  - Modeling fault in the testing procedure is the fault model

# Testing: Stuck-at model



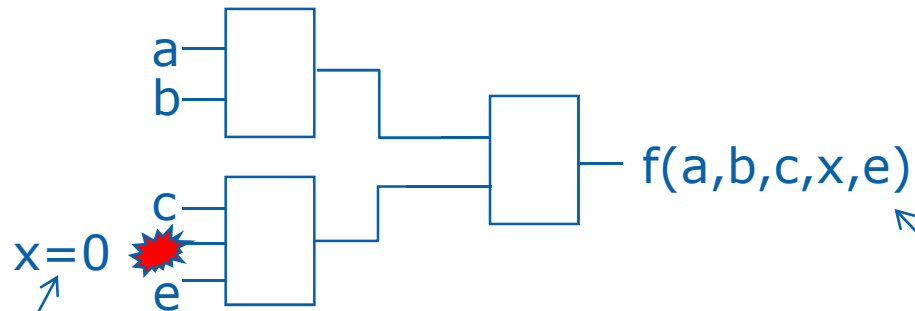
# Testing: Stuck-at model



## □ To test for stuck-at.....

- A pattern of inputs that makes an output that is wrong, i.e., different what it should be if the fault was not present
- Usually try to generate tests to detect specific faults
- If there is big list of possible faults, you generate a test set and ask what fraction of all the faults will get detected, called the fault coverage.

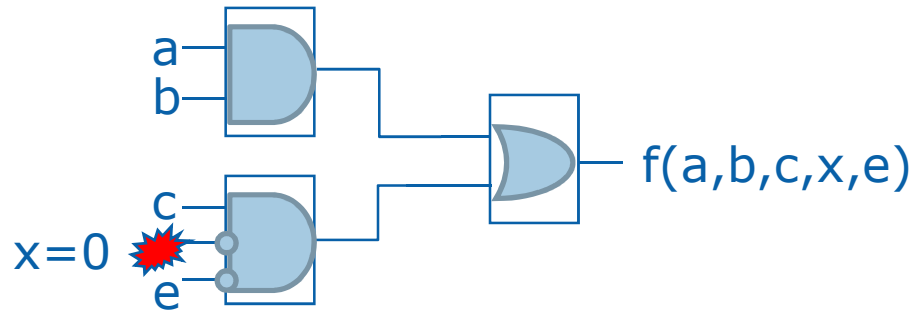
# Testing: Stuck-at model



If this line is to be tested for sa1, apply a 0 to it.

...then see if you get an output  $f$  that was wrong, i.e., it still looks like  $x$  was 1, though we know it was 0

## Testing: Example

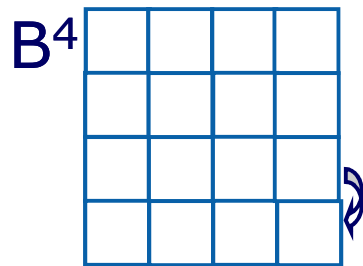
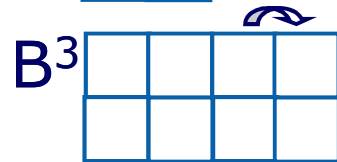
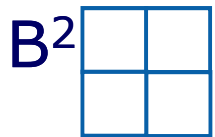


- $f(a,b,c,x,e) = ab + cx'e'$   
 $\partial f / \partial x = f_x \oplus f_{x'} = ab \oplus (ab + ce') = a'ce' + b'ce'$
- Output is independent of  $x$
- By changing values for  $x$  see if the output still depends on  $x$ ; if so then  $x$  is faulty

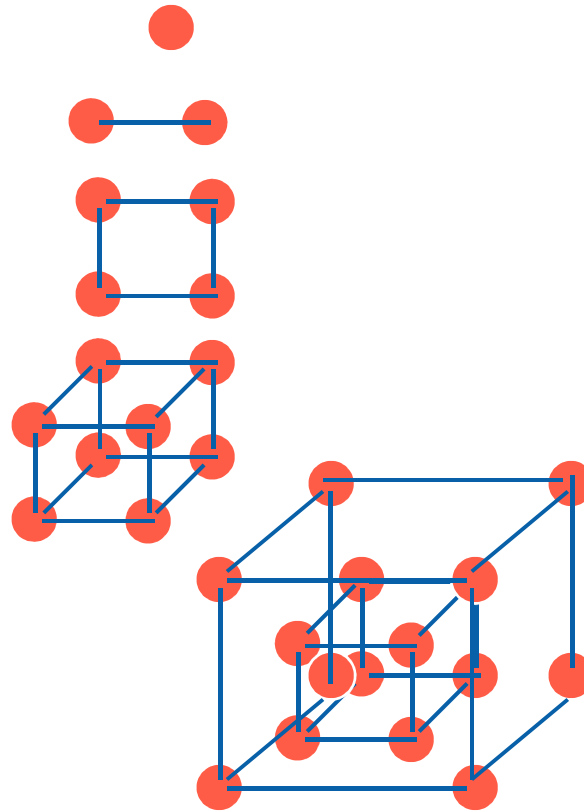
# The Boolean Space $B^n$

$$B = \{0,1\}, \quad B^2 = \{0,1\} \times \{0,1\} = \{00, 01, 10, 11\}$$

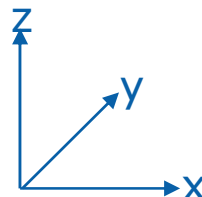
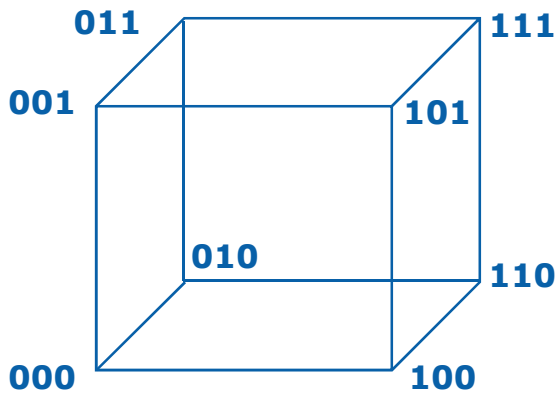
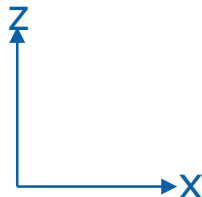
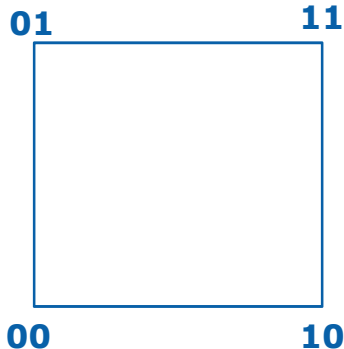
Karnaugh Maps:



Boolean Cubes:

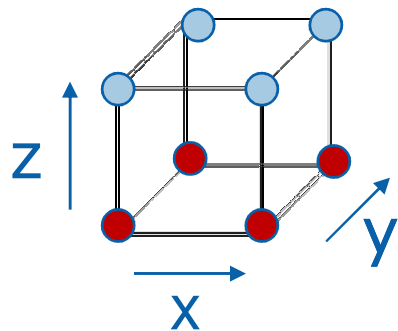


# Boolean Cubes



# Set of Boolean Functions

- **Truth Table or Function Table:**

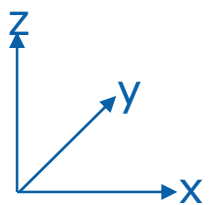
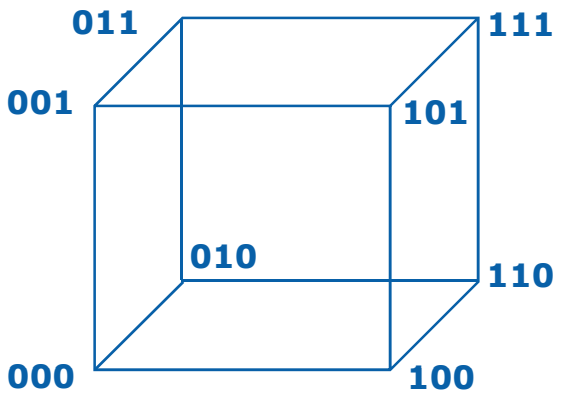


x	y	z	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

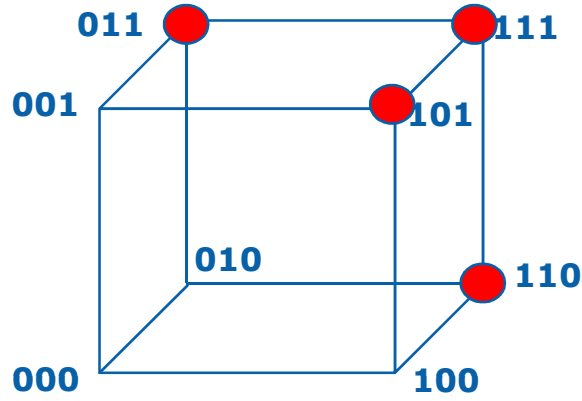
- There are  $2^n$  vertices in input space  $B^n$
- There are  $2^{2^n}$  distinct logic functions.
  - Each subset of vertices is a distinct logic function:  $f \subseteq B^n$

# Boolean Cubes

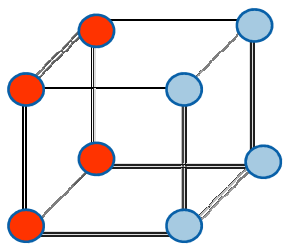
□ Boolean Cubes



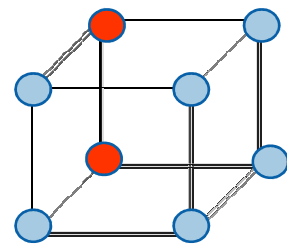
□  $F(x,y,z) = xy + yz + xz$



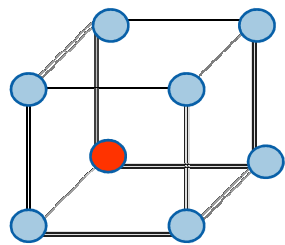
$f = \bar{x}$



$f = \bar{x}y$



$f = \bar{x}\bar{y}z$



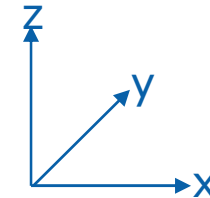
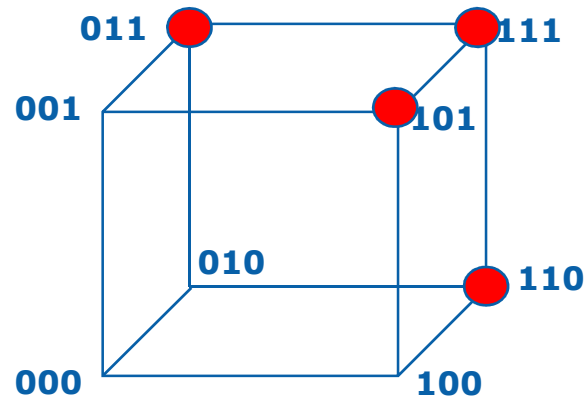
# Boolean Quantification

- Let  $F(x_1, x_2, \dots, x_i, \dots, x_n) = x_i F_{x_i} + x_i' F_{x_i'}$  be the Shannon expansion of  $F$
- The Universal Quantification of  $F$  w.r.t variable  $x_i$  is:
  - $(\forall x_i F) [x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n]$
  - $\forall$  sign is “for all” symbol from logic
  - $(\forall x_i F)$  is a new function
- The Existential Quantification of  $F$  w.r.t variable  $x_i$  is:
  - $(\exists x_i F) [x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_n]$
  - $\exists$  sign is “there exists” symbol from logic
  - $(\exists x_i F)$  is a new function
- $F$  has  $x_i$  in its support, but  $(\forall x_i F)$  and  $(\exists x_i F)$  both omit  $x_i$

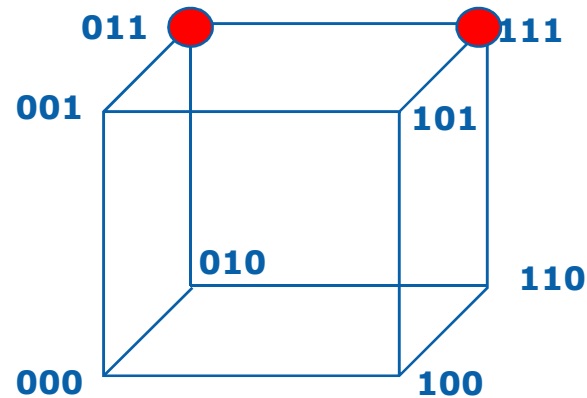


# Universal Quantification or Consensus

$$F(x,y,z) = xy + yz + xz$$

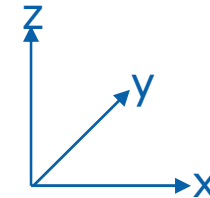
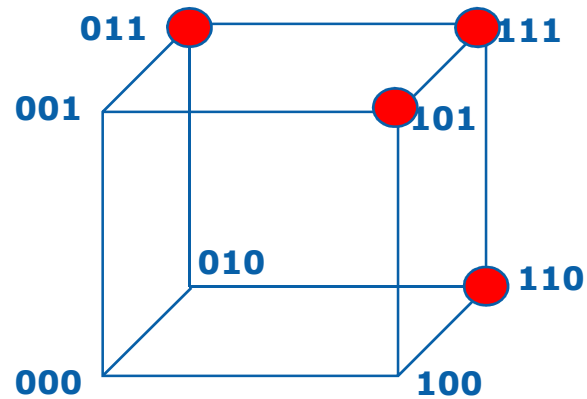


$$(\forall x F)[y,z] = F_x \bullet F_{x'} = (y+z) \bullet yz = yz$$

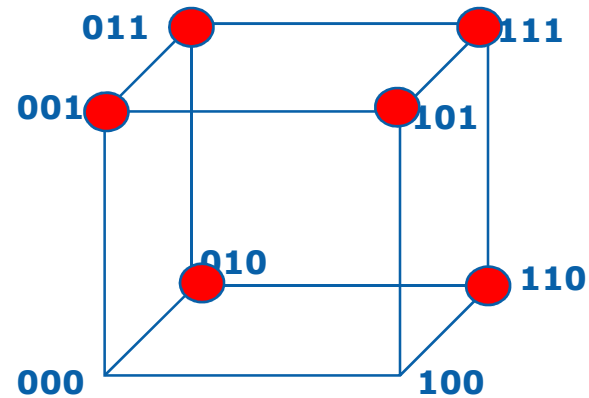


# Existential Quantification - Smoothing

$$F(x,y,z) = xy + yz + xz$$



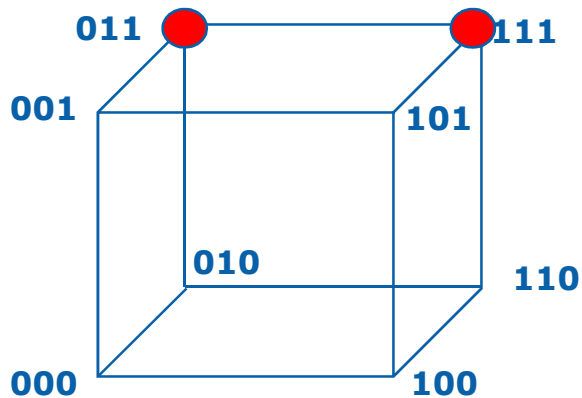
$$(\exists x F)[y,z] = F_x + F_{x'} = (y+z) + yz = y+z$$



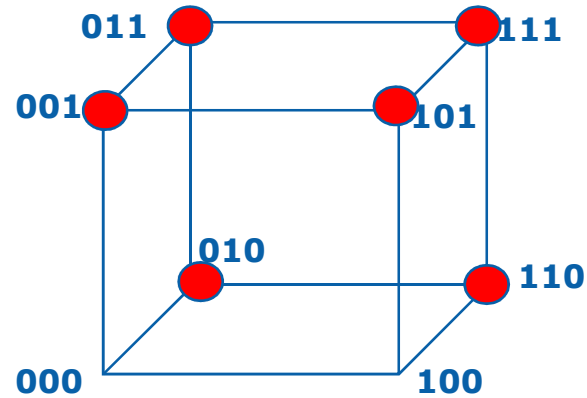
# Understanding Quantification

$$F(x,y,z) = xy + yz + xz$$

$$(\forall x F)[y,z] = F_x \cdot F_{x'} = (y+z) \cdot yz = yz$$



$$(\exists x F)[y,z] = F_x + F_{x'} = (y+z) + yz = y+z$$

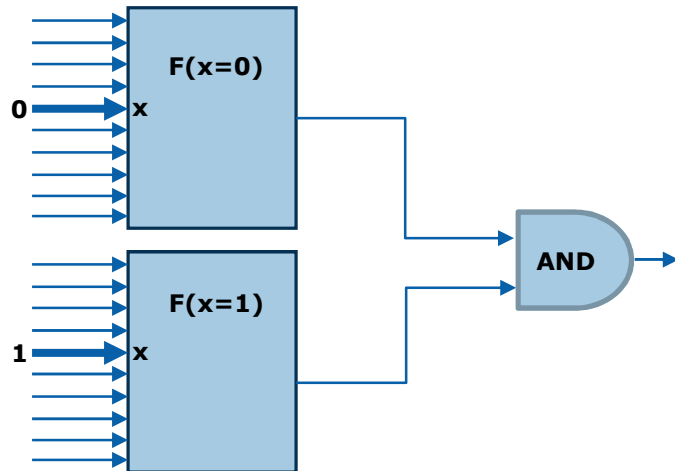


## □ Properties:

▪ Suppose we have  $F(x,y,z,w)$ :

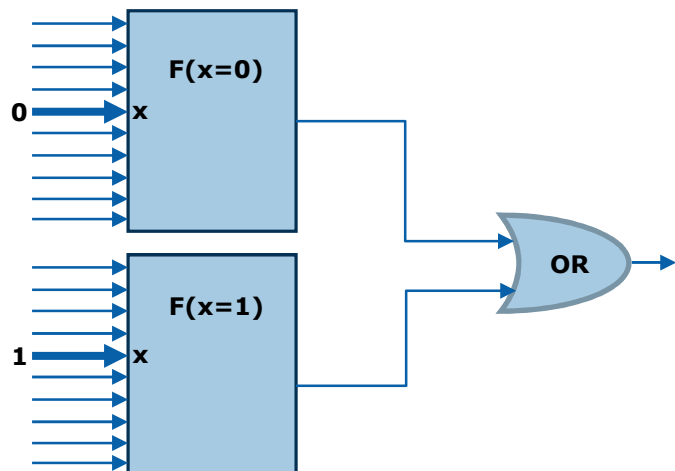
- $(\forall xy F)[z,w] = (\forall x (\forall y F)) = F_{xy} \cdot F_{x'y} \cdot F_{xy'} \cdot F_{x'y'}$
- $(\exists xy F)[z,w] = (\exists x (\exists y F)) = F_{xy} + F_{x'y} + F_{xy'} + F_{x'y'}$

# Meaning of Quantification Notation



$(\forall x F)[\text{all original vars but } x] == 1$

$$F_{x'} \cdot F_x$$

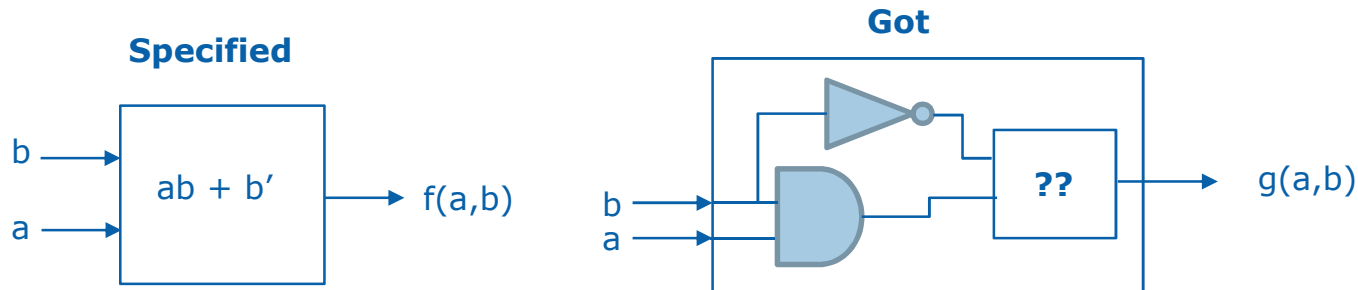


$(\exists x F)[\text{all original vars but } x] == 1$

$$F_{x'} + F_x$$

# Quantification Application: Network Repair

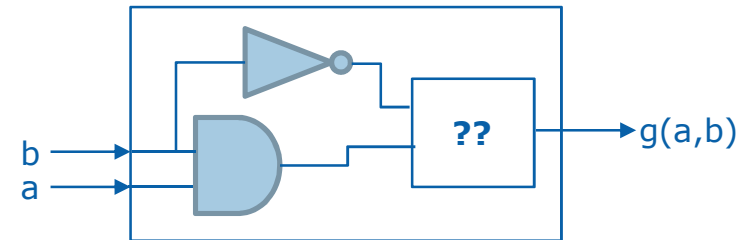
- Suppose we have to implement  $f(a,b) = ab + b'$ 
  - But we implemented it wrong...in particular we got ONE gate wrong



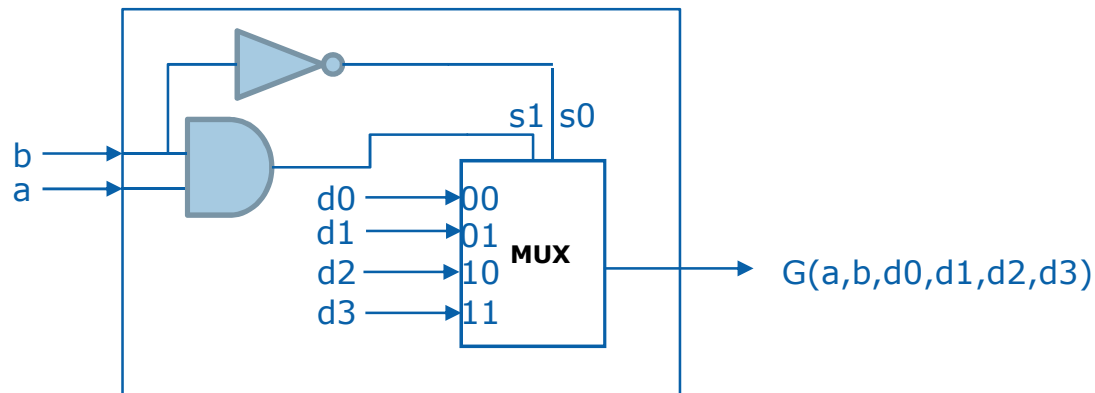
- Can we deduce how precisely to change this gate to restore the correct functionality?

# Quantification Application: Network Repair

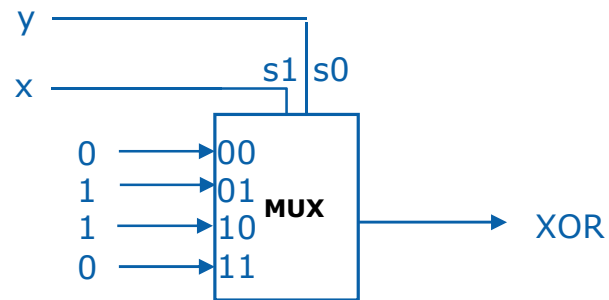
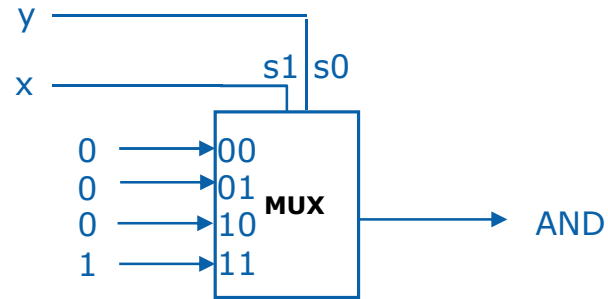
- ❑ Clever trick is to replace the suspect by a 4:1 MUX with 4 arbitrary new variables
- ❑ By cleverly assigning values to  $d_0, d_1, d_2, d_3$ , we can fake any gate
- ❑ What are the right values for  $d$  so that  $g$  is repaired



Replace with

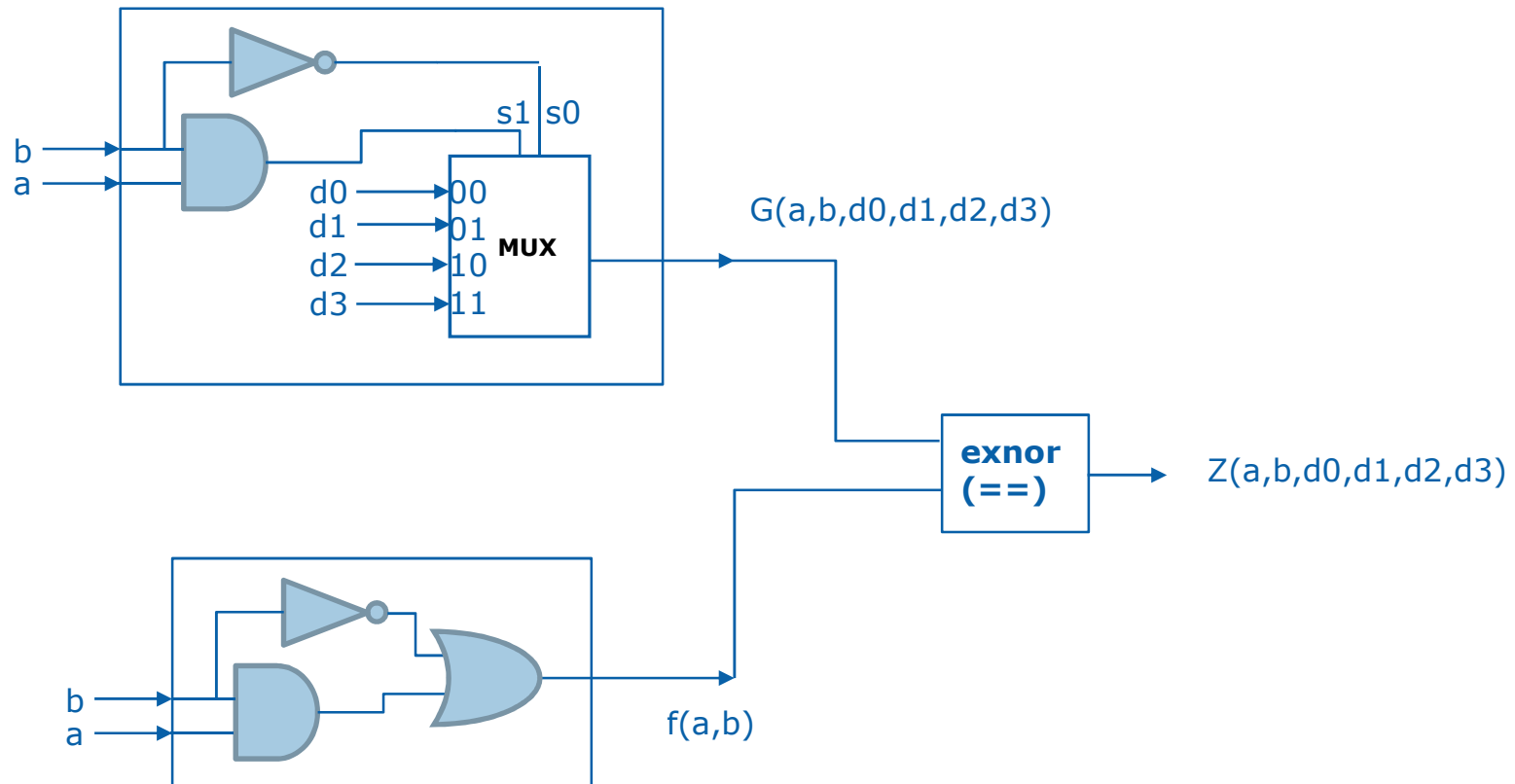


# Implementing gate with a MUX



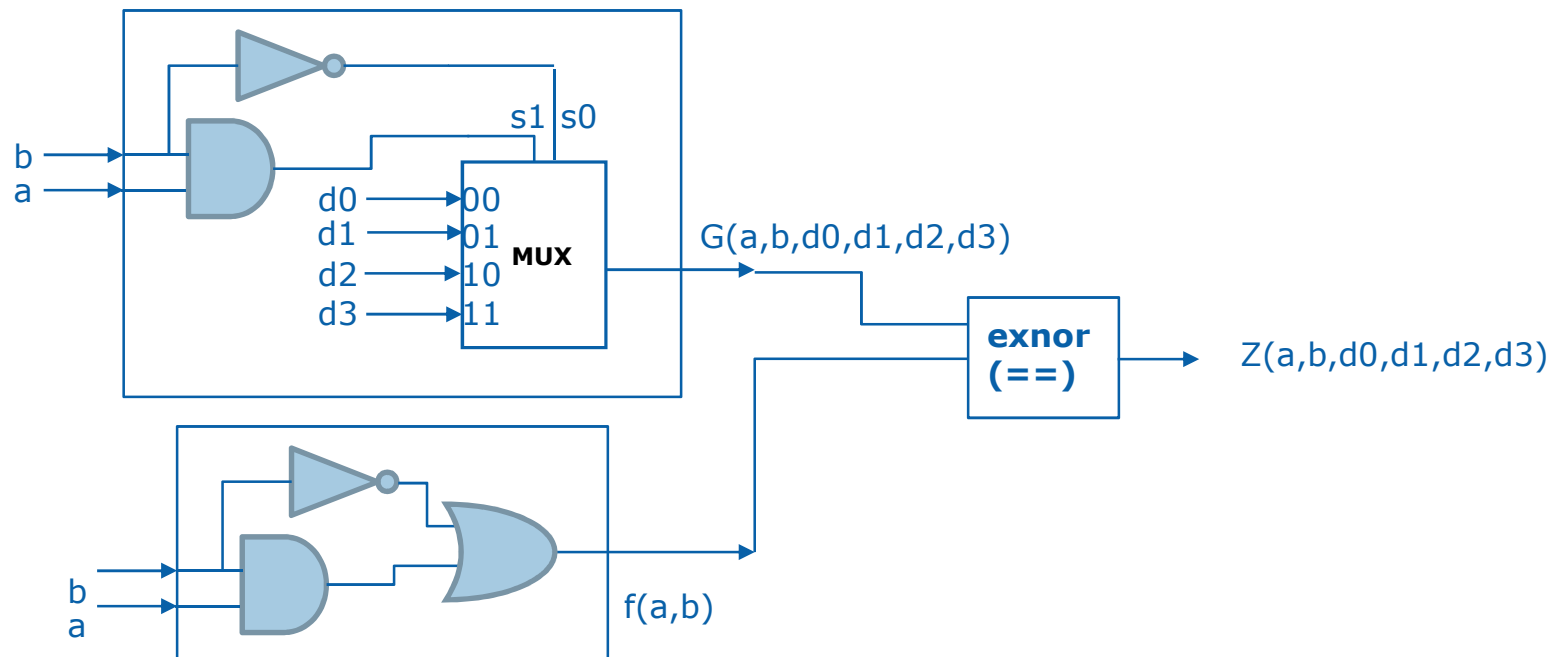
# Network Repair Using Quantization

- Make a new function  $Z(a,b,d_0,d_1,d_2,d_3)$  that is equal to 1 when  $G==f$



# Network Repair Using Quantization

- ❑ Nothing but find values for  $d_0, d_1, d_2, d_3$  such  $Z(a, b, d_0, d_1, d_2, d_3) = 1$  (for all possible values of  $a, b$ )
- ❑ This is nothing but any pattern of  $d_0, d_1, d_2, d_3$  that makes  $(\forall a, b) Z[a, b, d_0, d_1, d_2, d_3] == 1$
- ❑ Universal quantification (consensus) of  $Z$  w.r.t variables  $a, b$ !



# Advanced Boolean Algebra

What **did** we learn?

- ❑ Boolean Decomposition
- ❑ Cofactors
- ❑ Boolean Derivatives and Applications
- ❑ Boolean Cubes
- ❑ Boolean Quantification and Applications

