Multi-Level Logic Optimization
- Node Optimization

Simplification

- Represent each node in two level form
- Use espresso to minimize each node
- Several simplification procedures which vary only in the size of don’t care constructed
  - don’t care set empty
  - subset of don’t care

Simplification

Don’t care

- External DC
- Internal DC (derived from the structure)

  (1) Satisfiability DC (SDC)

  (2) Observability DC (ODC)

Satisfiability Don't Care

- Satisfiability DC (to all nodes)

  Ex: $x = ab$

  $y = a'b'$

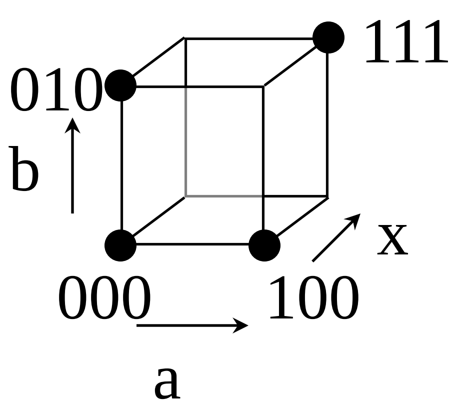
  $f = x'y'+a$

  $\therefore\ x \oplus ab = x(ab)'+x'(ab)$

- Why?

  Expand the Boolean space of PI to include intermediate variables $B^n \rightarrow B^{n+m}$

  n=2 m=1



In general, SDC = $\sum (y_i \bullet f_i' + y_i' \bullet f_i)$

Observability Don't Care

- Observability DC (to certain intermediate node)

$\frac{\delta f}{\delta x}$ f is sensitive to the value of x

$(\frac{\delta f}{\delta x})'$ f is not sensitive to the value of x

x is observable at f if $\frac{\delta f}{\delta x} \neq 0$

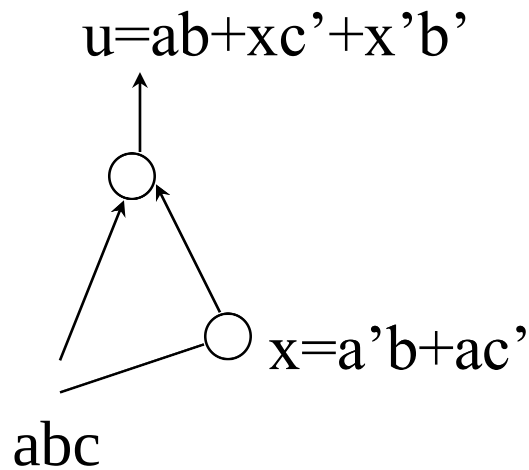
(Note $\frac{\delta f}{\delta x}$ is a function of the inputs)

Example

Ex:

$$u = ab+xc'+x'b'$$

$$x = a'b+ac'$$



$$\left(\frac{\delta u}{\delta x}\right)' = u_x \bullet u_{x'} + u_x' \bullet u_{x'}'$$

$$u_x = ab + c'$$

$$u_{x'} = ab + b'$$

$$\left(\frac{\delta u}{\delta x}\right)' = u_x \bullet u_{x'} + u_x' \bullet u_{x'}'$$

$$= (ab+b'c') + a'bc \quad \text{(the ODC of x)}$$

$$\mathbf{u = ab + xc' + x'b'}$$

$a = 1$ $b = 1$ $\Rightarrow u = 1 \cdot 1 + xc' + x'b' = 1$

$b = 0$ $c = 0$ $\Rightarrow u = a \cdot 0 + x \cdot 1 + x' \cdot 1 = 1$

$a = 0$ $b = 1$ $c = 1$ $\Rightarrow u = 0 \cdot 1 + x \cdot 0 + x' \cdot 0 = 0$

Observability Don't Care

- ODC relative to x is

$$ODC_x = \prod_{f_i \in \text{output}} \left(\frac{\delta f_i}{\delta x}\right)'$$

- Including the external DC

$$\prod_{f_i \in \text{output}} \left(\left(\frac{\delta f_i}{\delta x}\right)' + D_{xi} \right)$$

- In general, the complete DC is too large

  solutions:

  filter the DC

  => subset "support" filter

Standard Script of SIS

sweep	constant propagation, remove buffer
eliminate –l	collapse; node has < 1000 cubes
simplify	espresso for each node
eliminate –l	
sweep	
eliminate 5	value > 5
simplify	
resub –a	algebraic div
gkx –abt 30	-a generate all kernel –b the best kernel inter
	-t value saved
resub –a; sweep	
gcx –bt 30	-b the best cube -t the value saved
resub –a; sweep	
gkx –abt 10	
resub –a; sweep	
gcx –bt 10	
resub –a; sweep	
……..	

Boolean Difference and ODC for Logic Network of Simple Gates

Controlling and Non-controlling Values of Simple Gates

- Given a **simple gate** (i.e. AND, OR, NAND, NOR), a **controlling** value on an input determines the output of the gate independent of the other inputs
- Given a **simple gate** (i.e. AND, OR, NAND, NOR), a **non-controlling** value on an input cannot determine the output of the gate independent of the other inputs
- Example:
  - 0 is a controlling value for AND gate

    1 is non-controlling value for AND gate
  - 0 is a controlling value for NAND gate

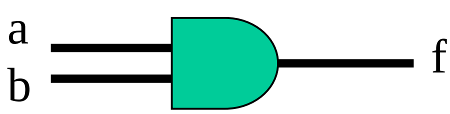
    1 is non-controlling value for NAND gate
  - 1 is a controlling value for OR gate

    0 is non-controlling value for OR gate
  - 1 is a controlling value for NOR gate

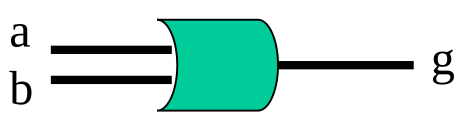
    0 is non-controlling value for NOR gate

Boolean Difference of Simple Gates

- Non-controlling value of a side input is merely a specialization of the Boolean difference of an on-input to the simple gate
- *a* is an on-input and *b* is a side input



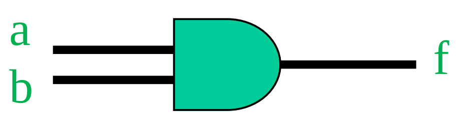
$$\frac{\partial f}{\partial a} = b$$



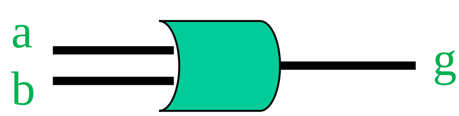
$$\frac{\partial g}{\partial a} = \overline{b}$$

ODC of Simple Gates

- Controlling value of a side input is merely a specialization of ODC of an on-input to the simple gate
- *a* is an on-input and *b* is a side input



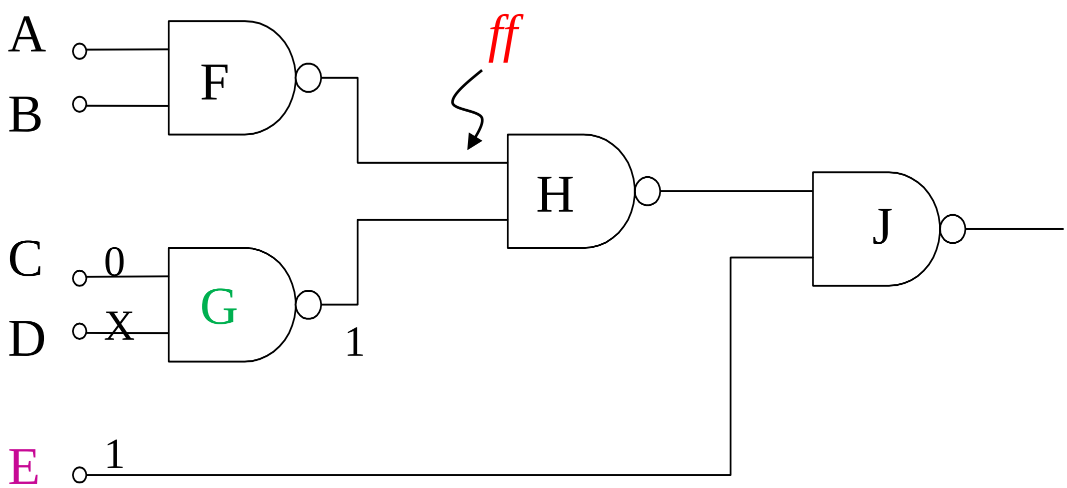
ODC (a) = b’



ODC (a) = b

- ODC = complement of Boolean difference

Example of Computing Boolean Difference of *ff*



- (C = 0 and D=x) or (C = x and D=0)

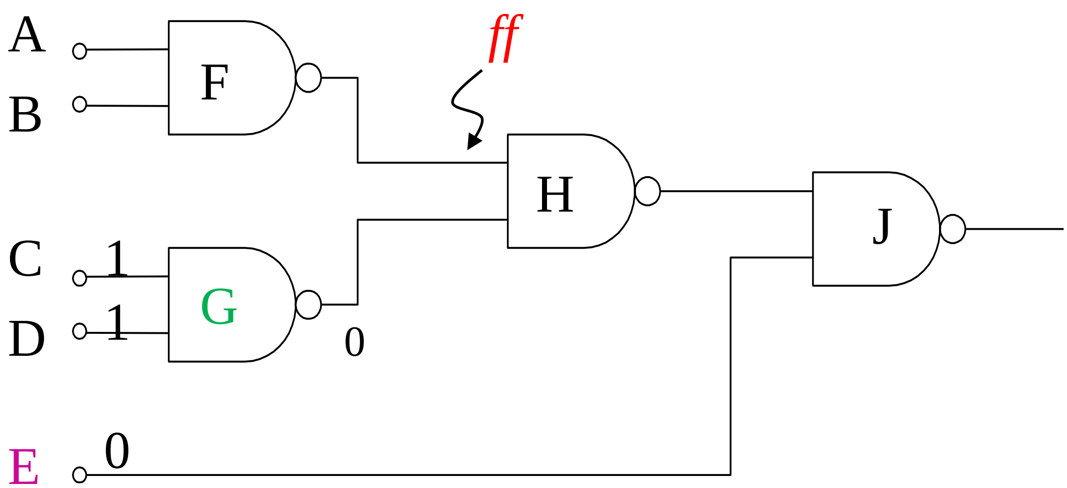
  => C' + D'

- E = 1

  => E

- Boolean Difference (*ff*) = (C' + D') E (conditions of side inputs are *anding*)

Example of Computing ODC of *ff*



- (C = 1 and D=1)

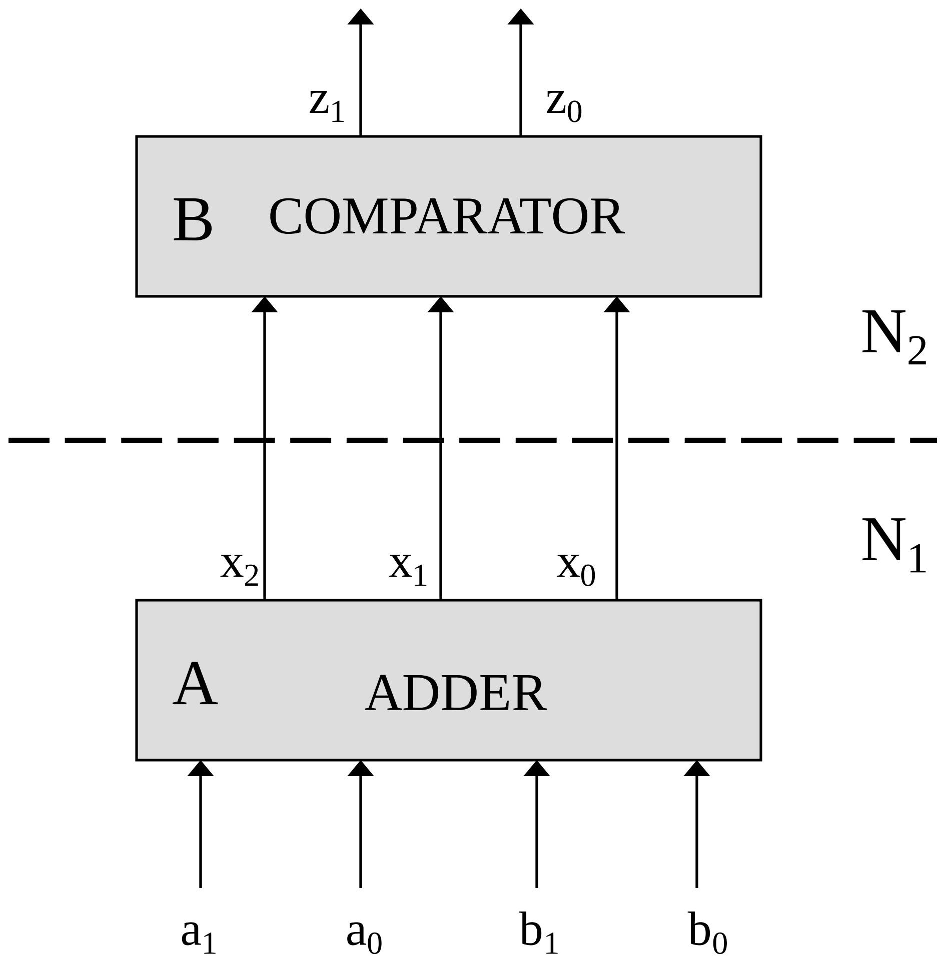
  => C D
- E = 0

  => E'
- ODC (*ff*) = CD + E' (conditions of side inputs are *oring*)
- (ODC (ff))' = Boolean Difference (ff)

  = (C' + D') E

Incompleteness of Don't Cares

- Example



$$Z = 0\,1 \Rightarrow a + b < 3$$
$$Z = 0\,0 \Rightarrow a + b = 3$$
$$Z = 1\,0 \Rightarrow a + b > 3$$

Equivalence

x_2	x_1	x_0	z_1	z_0
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Input values 000, 001, and 010 are equivalent with respect to ***B***.

{000,001,010} forms an equivalence class.

The other equivalent classes are:
{011} and {100,101,110,111}

Boolean Relation Formulation

If x', x" are input values indistinguishable from the outputs of ***B***, they are interchangeable output values for ***A***.

a_1 a_0 b_1 b_0	x_2 x_1 x_0
0 0 0 0	{000,001,010}
0 0 0 1	{000,001,010}
0 0 1 0	{000,001,010}
0 1 0 0	{000,001,010}
1 0 0 0	{000,001,010}
0 0 1 1	{011}
0 1 0 1	{000,001,010}
0 1 1 0	{011}
1 0 0 1	{011}
1 0 1 0	{100,101,110,111}
1 1 0 0	{011}
0 1 1 1	{100,101,110,111}
1 0 1 1	{100,101,110,111}
1 1 0 1	{100,101,110,111}
1 1 1 0	{100,101,110,111}
1 1 1 1	{100,101,110,111}

Example

The optimal implementation of the following relation $R \subseteq B^2 \times B^2$

a	b	x y
0	0	0 0, 0 1
0	1	0 1, 1 0
1	1	1 1
1	0	1 0

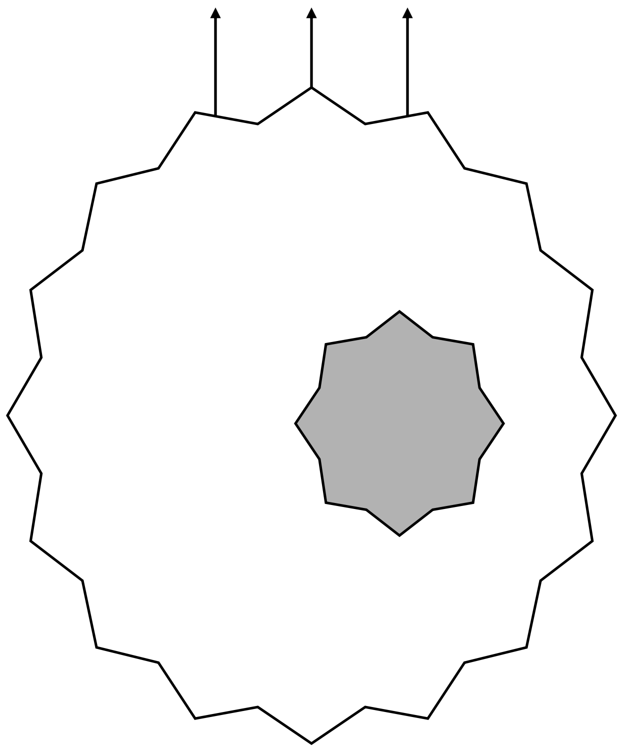
b \ a	0	1
0	00	10
1	01	11

b \ a	0	1
0	00	10
1	10	11

b \ a	0	1
0	01	10
1	01	11

b \ a	0	1
0	01	10
1	10	11

Problems



1. How to find a multiple-output sub-network ?
2. How to utilize Boolean Relations for minimization ?
   – full set of Boolean Relation
   – subset of Boolean Relation