Multilevel logic optimization

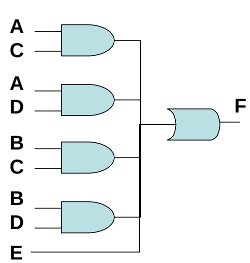
Hà Thủy Tú – 10070502
Trần Thế Anh - 10070471

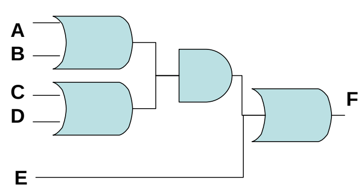
Multilevel logic optimization

- Goals
- Methodology
- Algebraic optimization method

Goals

- F = AC + AD + BC + BD + E





Goals

- Representation of the Boolean function
- Optimal
  - Area
  - Speed
  - Testability
  - Power dissipation

Methodology

- How to optimize

Re-write logic functions using logic operations

1. **Decomposition**
2. **Extraction** (decompose multiple functions)
   - Find optimal intermediate functions (area, delay,..)
3. **Factoring**
   - Find a factored form with min number of literals
4. **Substitution**
5. **Elimination**

Methodology

- Decomposition

  A single function => a collection of new functions

  F = ABC + ABD + A'C'D' + B'C'D' (12 literals)

⇒ F = XY + X'Y' (4 literals)

X = AB

Y = C + D

Methodology

- Extraction

  Identify common expressions

  F = (A + B)CD + E (11 literals)

  G = (A + B)E'

  H = CDE

⇒ F = XY + E (7 literals)

G = XE'

H = YE

X = A + B

Y = CD

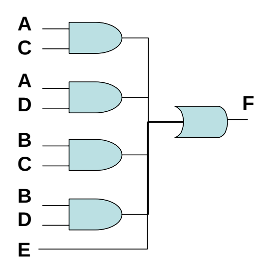
Methodology

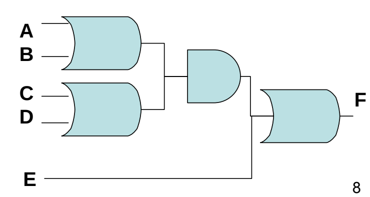
- Factoring

  2-level function => multi-level function

  F = AC + AD + BC + BD + E (9 literals)

⇒F = (A+B)(C+D) + E (5 literals)





Methodology

- Substitution
  rewrite F in terms of G and its original inputs
  F = A + BC (5 literals)
  G = A + B

$\Rightarrow$ F = G(A + C) (4 literals)
G = A + B

Methodology

- Elimination
  Inverse operation of substitution
  F = GA + G'B (4 literals)
  G = C + D

$\Rightarrow$ F = AC + AD + BC'D' (7 literals)
G = C + D

"division" plays a key role in all of these (except elimination)

Example

Restructuring Problem: **Given initial network, find best network.**

Initial network: $f_1 = abcd+abce+ab'cd'+ab'c'd'+a'c+cdf+abc'd'e'+ab'c'df'$
$f_2 = bdg+b'dfg+b'd'g+bd'eg$

minimizing,
$f_1 = bcd+bcf+b'cd'+a'c+cd'e+abc'd'e'+ab'c'df'$
$f_2 = bdg+dfg+b'd'g+d'eg$

factoring,
$f_1 = c(d(b+f)+d'(b'+e)+a')+ac'(bd'e'+b'df')$
$f_2 = g(d(b+f)+d'(b'+e))$

decompose,
$f_1 = c(x+a')+ac'x'$
$f_2 = gx$
$x = d(b+f)+d'(b'+e)$

Two problems:
- **find good common subfunctions**
- **how to do division**

Algebraic optimization method

- Division
  - Logic division
  - Algebraic division
  - Kernels and algebraic divisors
  - Computing kernels
- Algebraic method for logic operations
  - The basic of algebraic optimization method
  - Factoring
  - Extraction and resubstitution
  - Resubstitution with complement

Division

- Logic division

Given f, p
Find q, r : **f = p·q + r**
"Division of **f** by **p** generating ***quotient* q** and ***remainder* r**"

If r is null , **p** is called a **factor** (Boolean factor)
If r is not null, **p** is called a **divisor** (Boolean divisor)

q, r unique ???

Division

- Logic division **f = p·q + r**

X1	X2	X3	f	p	q	r
0	0	0	1			1
0	0	1				
0	1	0	1	1	1	
0	1	1	1	1	1	
1	0	0		1		
1	0	1			1	
1	1	0				
1	1	1				

Division

So many factors, how to optimize ?

- Logic division **f = p·q + r** **q, r not unique**

X1	X2	X3	f	p	q	r
0	0	0	1			1
0	0	1				
0	1	0	1	1	1	
0	1	1	1	1	1	
1	0	0		1		
1	0	1			1	
1	1	0			1	
1	1	1			1	

Division

- Logic division **f = p·q + r**

X1	X2	X3	f	**p=X1'X2**	**q=X3**	r
0	0	0	1			1
0	0	1			1	
0	1	0	1	1		1
0	1	1	1	1	1	
1	0	0				
1	0	1			1	
1	1	0				
1	1	1			1	

Division

- Logic division **f = p·q + r** **q, r unique**

X1	X2	X3	f	p=X1'X2	q=X3	r
0	0	0	1			1
0	0	1			1	
0	1	0	1	1		1
0	1	1	1	1	1	
1	0	0				
1	0	1			1	
1	1	0				
1	1	1			1	

$p \perp q$

Division

- Algebraic division

p is orthogonal to q, $\mathbf{p \perp q}$, if $sup(p) \cap sup(q) = \emptyset$

(p = ab'+ c, then sup(p)={a,b,c})

ex. p= a+b q=c+d, then $p \perp q$

$\exists$ h, r : f = g·h + r where $h \neq 0$ and $g \perp h$

=> **g** is an algebraic divisor of **f**

Quotient h = f/g is unique

Division

- Algebraic division
  - Computing quotient

Given f, g are lists of cubes

$f = \{b_1, b_2, \ldots, b_{|f|}\}$

$g = \{a_1, a_2, \ldots, a_{|g|}\}$

Define

$h_i \equiv \{c_j \mid a_i \cdot c_j \in f\}, \forall i=1,2,\ldots,|g|$

h_i is all multipliers of the cube a_i producing b_x

Then

$h = f/g = h_1 \cap h_2 \cap \ldots \cap h_{|g|}$

Division

- Algebraic division
  - Computing quotient
  - example

f = abc + abd + de

g = ab + e

$h_1=\{c, d\}$, $h_2 =\{d\}$

$h = f/g = h_1 \cap h_2 = \{d\}$

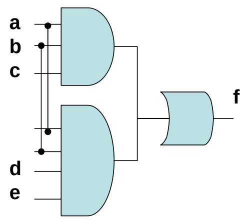
f = (ab + e)d + abc

Division

- Kernels and algebraic divisors

Situation

f = abc + abde => optimal multiple levels ?

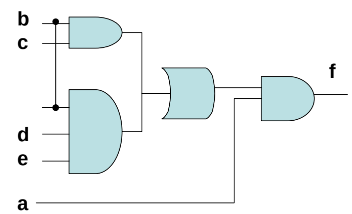


Division

- Kernels and algebraic divisors

minimizing

f = a(bc + bde)



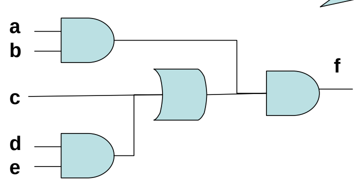
Division

- Kernels and algebraic divisors

Solution

f = ab(c + de)

Standard cells



Division

- Kernels and algebraic divisors

Solution

f = ab(c + de)

h, kernel

g, cokernel (cubes)

- Primary divisors of f = P(f) ={f/c| c is a cube}
- A function g is termed cube-free if the only cube that divides g **evenly** is 1
- Kernel of f = K(f)={k| k ∊ P(f), k is cube-free}
- Ex. **f= abc + abde**

  f/a = bc + bde is a prime divisor but not cube-free

  f/(ab)=c + de is a kernel

Division

- **Computing kernels**

```
KERNELS(f) {
  c_f = largest cube (max number of literals) factor of f;
  K=KERNEL1(0,f/c_f);
  if (f is cube-free) return (f ∪K);
  return(K);
}

KERNEL1(j,g) {
  R=g;
  N=Max index of variables in g;
  For (i=j+1;i≤N;i=i+1){
      if (l_i in 1 or no cubes of g) continue;
      c = largest cube dividing g/l_i evenly;
      if (for all k≤i, l_k ∉c) R=R ∪KERNEL1(i,g/(l_i∩c));
  }
  Return(R);
}
```

l_1=a, l_2=b,…

Division

- Computing kernels
  - Example 1

f = abcd + abce + abef   K(f) = R = ?

1. c_f = ab
2. f/c_f = g = cd + ce + ef , R ={cd + ce + ef}
3. Lexical ordering: a, b, c, d, e, f
4. a,b ∉ {cd, ce, ef} , repeat step 3
5. g/c = d + e ∈ P(f) & cube-free
   R = {cd + ce + ef, d + e}
6. repeat step 3

Result R = {cd + ce + ef, d + e, c + f}

Division

- Computing kernels
  - Example 2

f = adf + aef + bdf + bef + cdf + cef + g

K(f) = R = ?

f = (a+b+c)(d+e)f + g

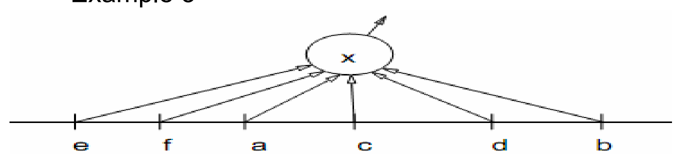
•A kernel may have many co-kernels.
•Co-kernel can be the trivial cube 1.

Table 1 Kernels and Co-Kernels of $(a + b + c)(d + e)f + g$

Kernel	Co-Kernel	Level
$a + b + c$	df, ef	0
$d + e$	af, bf, cf	0
$(a + b + c)(d + e)$	f	1
$(a + b + c)(d + e)f + g$	1	2

Division

- Computing kernels
  - Example 3



(a)

$x = b'c'd' + acd' + ace' + de'f + ce'f + bc'$

co−kernel	kernel
e' f	c + d
c e'	a + f
a c	e' + d'
c	a d' + a e' + e' f
c'	b' d' + b
d'	b' c' + a c
e'	a c + d f + c f

Algebraic optimization method

- Division
  - Logic division
  - Algebraic division
  - Kernels and algebraic divisors
  - Computing kernels
- Algebraic method for logic operations
  - The basic of algebraic optimization method
  - Factoring
  - Extraction
  - Extraction and resubstitution
  - Resubstitution with complement

Algebraic method - logic operations

- The basic of algebraic optimization method

Theorem

f and g have a nontrivial common divisor d (ie. d is not a cube) iff there exist kernels $k_f \in K(f)$ and $k_g \in K(g)$ such that $k_f \cap k_g$ is nontrivial with 2 or more terms (not a cube).

f = ae+be+cde+ab, g = ad+ae+bd+be+bc

k_f ={a+b+cd} , k_g ={a+b}

$k_f \cap k_g$ ={a+b+cd} ∩{a+b} = {a+b}

=> (a+b) is a nontrivial common algebraic divisor for f and g

Algebraic method - logic operations

- The basic of algebraic optimization method

Meaning

- The theorem is used to detect if two or more expressions have any common algebraic divisor than just the single cubes.

If $k_f \cap k_g$ is nontrivial => found a **good** common algebraic divisor to consider during logic optimization.

If $k_f \cap k_g$ is ∅ or **trivial** => need only look for divisors consisting of **single cubes**.

Algebraic method - logic operations

- Factoring

```
GFACTOR(f)
{
  if (number of terms in f is 1)  return(f);
  g = CHOOSE_DIVISOR(f);
  (h,r)=DIVIDE(f,g);
  f= GFACTOR(g) · GFACTOR(h) + GFACTOR(r);
  return(f);
}
```

•CHOOSE_DIVISOR chooses a factor
•Critical to obtaining a **good** factorization.

Algebraic method - logic operations

- Factoring

f=ac+ad+ae+ag+bc+bd+be+bf+ce+cf+df+dg

Single-cube factors:

f=a(c+d+e+g) + b(c+d+e+f) + c(e+f) + d(f+g)

16 literals

Multiple-cube factors:

f=(c+d+e)(a+b) + f(b+c+d) + g(a+d) + ce

14 literals

Algebraic method - logic operations

- Extraction and resubstitution (1)

Cách xác định các *cube-tự-do* xuất hiện trong nhiều hàm của tập $\{f_i\}$

- Sinh các kernel của mỗi hàm f_i
- Chọn một cặp các kernel $k_1 \in K(f_i)$ và $k_2 \in K(f_j)$ trong đó $i \neq j$ sao cho $k_1 \cap k_2$ không phải là 1 cube. Nếu không tồn tại cặp (k_1, k_2) → dừng
- Tạo biến mới v cho $k_1 \cap k_2$
- Cập nhật lại hàm liên quan

$$f_i = v.(f_i/(k_1 \cap k_2)) + r_i$$

Algebraic method - logic operations

- Extraction and resubstitution (2)

Các cube chung được phân tách theo cách sau:

- Chọn 1 cặp các cube $c_1 \in f_i$, $c_2 \in f_j$ với $i \neq j$ sao cho $c_1 \cap c_2$ chứa số biến ≥ 2. Nếu không tồn tại cặp (c_1, c_2) → dừng.
- Tạo biến mới x cho $c_1 \cap c_2$
- Cập nhật lại các hàm f_i với biến mới tạo

Mức (bậc) của 1 kernel

- Kernel mức-0: kernel không chứa các kernel khác.
- Kernel mức-N: kernel chứa ít nhất 1 kernel mức-(N-1) và không chứa kernel nào thuộc mức ≥ N

Algebraic method - logic operations

- Extraction and resubstitution (3)

- Cho 2 hàm

$$X = a.b.(c.(d + e) + f + g) + h$$
$$Y = a.i.(c.(d + e) + f + j) + k$$

- Xác định các cube-tự-do trong 2 hàm

$d + e$ là kernel mức-0 cho 2 hàm X và Y

Extraction

$$\begin{cases} L = d + e \\ X = a.b.(c.L + f + g) + h \\ Y = a.i.(c.L + f + j) + k \end{cases}$$

Algebraic method - logic operations

- Extraction and resubstitution (4)

$k1 = (c.L + f + g)$: kernel-mức 0 của hàm X

$k2 = (c.L + f + j)$: kernel-mức 0 của hàm Y

$k1 \cap k2 = c.L + f$

$$\Rightarrow \begin{cases} M = c.L + f \\ L = d + e \\ X = a.b.(M + g) + h \\ Y = a.i.(M + j) + k \end{cases}$$

X và Y không còn các kernel k_1, k_2 mà $k_1 \cap k_2$ không phải là 1 cube

Algebraic method - logic operations

- Extraction and resubstitution (5)
  - Phân tách các cube chung giữa 2 hàm

Cube $a.b.M$ của X và $a.i.M$ của Y có 2 biến chung

$$\Rightarrow \begin{cases} N = a.M \\ M = c.L + f \\ L = d + e \\ X = b.(N + a.g) + h \\ Y = i.(N + a.j) + k \end{cases}$$

Nếu thay thế L vào M, M và N, ta thu được

$$N = a.(c.(d + e) + f)$$
$$X = b.(N + a.g) + h$$
$$Y = i.(N + a.j) + k$$

Algebraic method - logic operations

- Resubstitution with complement

- Phân tích thừa số và tái thay thế đại số có thể được thực hiện với phần bù của ước số cho trước
- VD: $f = a.b + a.c + b'.c'.d$
  - Ta chọn $b + c$ là *kernel mức-0* của f và phân rã f thành

    $f = a.X + b'.c'.d$

    $X = b + c$
  - Ta kiểm tra phần bù của biến mới tạo là một ước số đại số của hàm. Khi đó

    $f = a.X + X'.d$

    $X = b + c$