Multilevel Combinational Circuits

Motivation: There are many functions that are too "expensive" to implement in two-level form

Try 16-bit adder $\Rightarrow$ 32 input lines and 2^{16} product terms!

2-level: control logic design

multi-level: datapath logic or random logic

Two-Level versus Multilevel

2-Level:

$$f_1 = AB + AC + AD$$
$$f_2 = \overline{A}B + \overline{A}C + \overline{A}E$$

6 product terms which cannot be shared.

24 transistors in static CMOS

Multi-level:

Note that $B + C$ is a common term in f_1 and f_2

$$K = B + C$$
$$f_1 = AK + AD$$
$$f_2 = \overline{A}K + \overline{A}E$$

3 Levels

20 transistors in static CMOS not counting inverters

Technologies

"Closed book": gate-array, standard-cell

"Open book": CMOS Domino, complex gate static CMOS

LOGIC EQUATIONS

↓

TECHNOLOGY-INDEPENDENT OPTIMIZATION — Factoring, Commonality Extraction

↓

TECH-DEPENDENT OPTIMIZATION (MAPPING, TIMING) ← LIBRARY

↓

OPTIMIZED LOGIC NETWORK

Tech.-Independent Optimization

Involves:

- **Minimizing two-level logic functions.**
- **Finding common subexpressions.**
- **Substituting one expression into another.**
- **Factoring single functions.**

Factored versus Disjunctive forms

$$f = ac + ad + bc + bd + a\bar{e}$$

sum-of-products or disjunctive form

$$f = (a + b)(c + d) + a\bar{e}$$

factored form

multi-level or complex gate

Optimizations

$$\mathcal{F} = \begin{cases} f_1 = \mathcal{AB} + \mathcal{AC} + \mathcal{AD} + \mathcal{AE} + \overline{\mathcal{A}}\,\overline{\mathcal{B}}\,\overline{\mathcal{C}}\,\overline{\mathcal{D}}\,\overline{\mathcal{E}} \\ f_2 = \overline{\mathcal{A}}\mathcal{B} + \overline{\mathcal{A}}\mathcal{C} + \overline{\mathcal{A}}\mathcal{D} + \overline{\mathcal{A}}\mathcal{F} + \overline{\mathcal{A}}\,\overline{\mathcal{B}}\,\overline{\mathcal{C}}\,\overline{\mathcal{D}}\,\overline{\mathcal{F}} \end{cases}$$

Factor $\mathcal{F}$

$$\mathcal{F} = \begin{cases} f_1 = \mathcal{A}(\mathcal{B} + \mathcal{C} + \mathcal{D} + \mathcal{E}) + \overline{\mathcal{A}}\,\overline{\mathcal{B}}\,\overline{\mathcal{C}}\,\overline{\mathcal{D}}\,\overline{\mathcal{E}} \\ f_2 = \overline{\mathcal{A}}(\mathcal{B} + \mathcal{C} + \mathcal{D} + \mathcal{F}) + \overline{\mathcal{A}}\,\overline{\mathcal{B}}\,\overline{\mathcal{C}}\,\overline{\mathcal{D}}\,\overline{\mathcal{F}} \end{cases}$$

Extract common expression

$$\mathcal{G} = \begin{cases} g_1 = \mathcal{B} + \mathcal{C} + \mathcal{D} \\ f_1 = \mathcal{A}(g_1 + \mathcal{E}) + \overline{\mathcal{A}}\,\overline{\mathcal{E}}\,\overline{g_1} \\ f_2 = \overline{\mathcal{A}}(g_1 + \mathcal{F}) + \overline{\mathcal{A}}\,\overline{\mathcal{F}}\,\overline{g_1} \end{cases}$$

What Does "Best" Mean?

Transistor count → AREA

Number of circuits → POWER

Number of levels → DELAY (Speed)

Need quick estimators of area, delay and power which are also accurate

Algebraic vs. Boolean Methods

Algebraic techniques view equations as polynomials and attempt to factor equations or "divide" them

Do not exploit Boolean identities e.g., $a\,\overline{a} = 0$

In algebraic substitution (or division) if a function $f = f(a, b, c)$ is divided by $g = g(a, b)$, a and b will not appear in f/g

Algebraic division: O(n log n) time

Boolean division: 2-level minimization required

Comparison

$$f = a\bar{b} + a\bar{c} + b\bar{a} + b\bar{c} + c\bar{a} + c\bar{b}$$

Algebraic factorization procedures

$$f = a(\bar{b} + \bar{c}) + \bar{a}(b + c) + b\bar{c} + c\bar{b}$$

Boolean factorization produces

$$f = (a + b + c)(\bar{a} + \bar{b} + \bar{c})$$

$$l = (b\bar{f} + \bar{b}f)(a + e) + \bar{a}\bar{e}(\bar{b}\bar{f} + bf)$$

$$r = (b\bar{f} + \bar{b}f)(\bar{a} + \bar{e}) + ae(\bar{b}\bar{f} + bf)$$

Algebraic substitution of l into r fails

Boolean substitution

$$r = a(\bar{e}\bar{l} + el) + \bar{a}(\bar{e}l + e\bar{l})$$

$$l = a(er + \bar{e}\bar{r}) + \bar{a}(\bar{e}r + e\bar{r})$$

Strong (or Boolean) Division

Given a function f to be strong divided by g

Add an extra input to f corresponding to g, namely $\mathcal{G}$ and obtain function h as follows

$$h_{\mathcal{DC}} = \mathcal{G}\,\overline{g} + \overline{\mathcal{G}}g$$

$$h_{\mathcal{ON}} = f_{\mathcal{ON}} - h_{\mathcal{DC}}$$

$$h_{\mathcal{OFF}} = \overline{f_{\mathcal{ON}} + h_{\mathcal{DC}}}$$

Minimize h using two-level minimizer

Strong Division Example

$$f = \bar{a}\,\bar{b}\,c + \bar{a}\,b\,\bar{c} + a\,\bar{b}\,\bar{c} + a\,b\,c$$

$$g = a\,\bar{b} + \bar{a}\,b$$

$$h_{DC} = G\,(a\,b + \bar{a}\,\bar{b}) + \bar{G}\,(a\,\bar{b} + \bar{a}\,b)$$

$$h_{ON} = f_{ON} - h_{DC}$$

bc \ Ga	00	01	11	10
00		x	1	x
01	1	x		x
11	x	1	x	
10	x		x	1

Function h

Minimization gives $h = \bar{G}\,c + G\,\bar{c}$

Weak (or Algebraic) Division

Definition: support of f as $sup(f)$ = { set of all variables v that occur in f as v or $\overline{v}$ }

Example: $f = \mathcal{A}\overline{\mathcal{B}} + \mathcal{C}$

$sup(f) = \{ \mathcal{A}, \mathcal{B}, \mathcal{C} \}$

Definition: we say that f is orthogonal to g, $f \perp g$, if $sup(f) \cap sup(g) = \phi$

Example: $f = \mathcal{A} + \mathcal{B}$ $\quad$ $g = \mathcal{C} + \mathcal{D}$

$\therefore \quad f \perp g$ since $\{ \mathcal{A}, \mathcal{B} \} \cap \{ \mathcal{C}, \mathcal{D} \} = \phi$

Weak Division - 2

We say that g divides f weakly if there exist h, r such that $f = gh + r$ where $h \neq \phi$ and $g \perp h$

Example: $f = ab + ac + d$

$g = b + c$

$f = a(b + c) + d \quad h = a \quad r = d$

We say that g divides f evenly if $r = \phi$

The quotient f/g is the largest h such that $f = gh + r$ i.e., $f = (f/g)g + r$

Computing f/g

Given $f = \{ c_i \}$, $g = \{ a_i \}$ **i.e., lists of cubes**

$$h_i = \{ b_j \mid a_i b_j \in f \} \; \forall i$$

i.e., all the multipliers of the cube a_i **in** g **that produce elements of** f **are in** h_i

Theorem: $$f/g = \bigcap_{i=1}^{|g|} h_i = h_1 \cap h_2 \; \ldots \; h_{|g|}$$

Weak Division Example

$$f = abc + abde + abh + bcd$$
$$g = c + de + h$$

Theorem says $f/g = f/c \cap f/de \cap f/h$

$$f/c = ab + bd$$
$$f/de = ab$$
$$f/h = ab$$

$$f/g = (ab + bd) \cap ab \cap ab = ab$$

$$f = ab(c + de + h) + bcd$$

Time complexity: $O(|f|\,|g|)$

Computing f/g

Given $f = \{ c_i \}$, $g = \{ a_i \}$

1) Encode cubes $a_i \in g$ with unique integer codes, by assigning a unique bit position for every literal in $sup(g)$

e.g., $g = ab + e$

110 001

2) Encode cubes $c_j \in f$ similarly

e.g., $f = abc + abd + de$

110 110 001

Time complexity?

3) Sort $\{ a_i, c_j \}$ by their codes

e.g., $\underbrace{ab, abc, abd,}_{110} \ \underbrace{e, de}_{001}$

$h_1 = c + d \qquad h_2 = d$

Divisors

Define divisors of f as the set

$$\mathcal{D}(f) = \{\ g \mid f/g \neq \phi\ \}$$

Define primary divisors of f as

$$\mathcal{P}(f) = \{\ f/c \mid c \text{ is a cube } \}$$

Example: $f = abc + abde$

$f/a = bc + bde$ **is a primary divisor**

Every divisor of f is contained in a primary divisor. If g divides f, then $g \subseteq p \in \mathcal{P}(f)$

g **is termed "cube-free" if the only cube dividing g <u>evenly</u> is 1.**

Kernels and Divisors

Define the kernels of f as

$\mathcal{K}(f) = \{\ k \mid k \in \mathcal{P}(f),\ k$ **is cube-free** $\}$

Example: $f = abc + abde$

$f/a = bc + bde$ **is a primary divisor but is not cube-free since** b **is a factor** $f/a = b(c + de)$

$f/ab = c + de$ **<u>is</u> a kernel**

ab **is the <u>co-kernel</u>**

The co-kernel of a kernel is not unique.

Examples

Consider

$$f = acd + bcd + ae + be$$

DIVISOR TYPE

$$f/a = cd + e$$

$$f/c = ad + bd$$

$$f/cd = a + b$$

$$f/e = a + b$$

Common Divisors and Kernels

Goal of multi-level logic optimizer is to find common divisors of two (or more) functions f and g

Theorem: f and g have a non-trivial common divisor d ($d \neq$ cube) if and only if there exist kernels

$k_f \in \mathcal{K}(f)$, $k_g \in \mathcal{K}(g)$ such that

$k_f \cap k_g$ is non-trivial, i.e., not a cube

$\therefore$ can use kernels of f and g to locate common divisors

Algorithm to find All Kernels

Kernels(f)

- Find c_f so f/c_f is cube-free ;
- $\mathcal{K}$ = Kernel1(0, f/c_f) ;
- if (f is cube-free)
  - return($f \cup \mathcal{K}$) ;
- return($\mathcal{K}$) ;

Kernel1(j, g) {

- $\mathcal{R}$ = g ;
- for ($i = j + 1;\ i \leq n;\ i = i + 1$) {
  - if (l_i in one or no terms) continue ;
  - c_e = Max. literal cube evenly dividing g / l_i ;
  - if (l_k not in c_e, for all $k \leq i$)
    - $\mathcal{R}$ = $\mathcal{R} \cup$ Kernel1(i, $(g/l_i)/c_e$)
- }
- return($\mathcal{R}$) ;

}

Kerneling Example

$$f = abcd + abce + adgh + aegh + abde + acdeg + beh$$

co-kernel	kernel
1	$a(bc + gh)(d + e) + ade(b + cg) + beh$
a	$(bc + gh)(d + e) + de(b + cg)$
ab	$c(d + e) + de$
abc	$d + e$
abd	$c + e$
ac	$b(d + e) + deg$
acd	$b + eg$
ace	$b + dg$
ad	$b(c + e) + g(ce + h)$
ade	$b + cg$
adg	$ce + h$
$\vdots$	

Pruning Condition and Example

If the largest cube factor c_e contains an already selected literal, then terminate current branch

- All kernels found by continuing have already been seen

$$f = abc(d+e)(k+l) + agh + m$$

a

$$f/a = bc(d+e)(k+l) + gh$$

b c

$$f/ab = c(d+e)(k+l)$$

$$(d+e)(k+l)$$

. . . .

$$f/ac = b(d+e)(k+l)$$

b **already selected**