

EDA (CS286.5b)

Day 16
Logic Synthesis:
Multi-Level

Today

- Multilevel Synthesis/Optimization
 - Why
 - Transforms
 - Division/extraction
 - Boolean Division
 - Don't care

Multi-level

- General circuit netlist
- May have
 - sums within products
 - products within sum
 - arbitrarily deep
- $y = ((a(b+c)+e)fg+h)i$

Why multi-level?

- $ab(c+d+e)(f+g)$

Why multi-level?

- $ab(c+d+e)(f+g)$
- $abcf+abdf+abef+abcf+abdg+abeg$
- 6 product terms
- 3 gates: and4,or3,or2
- Cannot share sub-expressions

Why Multilevel

- $a\bar{b}$
 - $a/b+ab$
- $a\bar{b}\bar{c}$
 - $a/bc+abc+a/b/c+ab/c$
- $a\bar{b}\bar{c}\bar{d}$
 - $a/bcd+abcd+a/b/cd+ab/cd+ab/c/d+a/b/c/d+abc/d+a/bc/d$

Why Multilevel

- $a \cdot b$
 - $x_1 = a/b + ab$
- $a \cdot b \cdot c$
 - $x_2 = x_1/c + x_1 \cdot c$
- $a \cdot b \cdot c \cdot d$
 - $x_3 = x_2/d + x_2 \cdot d$
- Multi-level
 - exploit common sub-expressions
 - linear complexity
- Two-level
 - exponential complexity

Multi-level Transformations

- Decomposition
- Extraction
- Factoring
- Substitution
- Collapsing

Decomposition

- $F = abc + abd + a/c/d + b/c/d$
- $F = XY + X/Y$
- $X = ab$
- $Y = c + d$

Decomposition

- $F = abc + abd + a/c/d + b/c/d$
 - 4 3-input + 1 4-input
- $F = XY + X/Y$
- $X = ab$
- $Y = c + d$
 - 5 2-input gates
- Note: use X and /X, use at multiple places

Extraction

- $F = (a+b)cd + e$
- $G = (a+b)/e$
- $H = cde$
- $F = XY + e$
- $G = x/e$
- $H = Ye$
- $X = a + b$
- $Y = cd$

Extraction

- $F = (a+b)cd + e$
- $G = (a+b)/e$
- $H = cde$
- 2-input: 4
- 3-input: 2
- $F = XY + e$
- $G = x/e$
- $H = Ye$
- $X = a + b$
- $Y = cd$
- 2-input: 6

Common sub-expressions over multiple output

Factoring

- $F=ac+ad+bc+bd+e$
- $F=(a+b)(c+d)+e$

Factoring

- $F=ac+ad+bc+bd+e$
 - 4 2-input, 1 5-input
 - 9 literals
- $F=(a+b)(c+d)+e$
 - 4 2-input
 - 5 literals

Substitution

- $G=a+b$
- $F=a+bc$
- Substitute G into F
- $F=G(a+c)$
 - (verify) $F=(a+b)(a+c)=aa+ab+ac+bc=a+bc$
- useful if also have $H=a+c$? ($F=GH$)

Collapsing

- $F=Ga+Gb$
- $G=c+d$
- $F=ac+ad+b/c/d$
- opposite of substitution
 - sometimes want to collapse and refactor
 - especially for delay optimization

Division

- **Given:** function (f) and divisor (p)
 - **Find:** quotient and remainder
- $$f=pq+r$$

E.g.

$$f=abc+abd+ef, p=ab$$
$$q=c+d, r=ef$$

Algebraic Division

- Use basic rules of algebra, rather than full boolean properties
- Computationally simple
- Weaker than boolean division
- $f=a+bc$ $p=(a+b)$
- Algebra: not divisible
- Boolean: $q=(a+c)$, $r=0$

Algebraic Division

- f and p are expressions (lists of cubes)
- $p = \{a_1, a_2, \dots\}$
- $h_i = \{c_j \mid a_i * c_j \hat{=} f\}$
- $f/p = h_1 \hat{=} h_2 \hat{=} h_3, \dots$

Algebraic Division Example

- $f = abc + abd + de$
- $g = ab + e$

Algebraic Division Example

- $f = abc + abd + de$
- $p = ab + e$
- $p = \{ab, e\}$
- $h_1 = \{c, d\}$ $h_2 = \{d\}$
- $h_1 \hat{=} h_2 = \{d\}$
- $f/g = d$
- $r = f - f/g = abc + abd + de - (ab + e)d = abc$

Algebraic Division Time

- $O(|f||p|)$ as described
 - compare every cube pair
- Sort cubes first
 - $O((|f|+|p|)\log(|f|+|p|))$

Primary Divisor

- f/c such that c is a cube
- $f = abc + abde$
- $f/a = bc + bde$ is a primary divisor

Cube Free

- The only cube that divides p is 1
- $c + de$ is cube free
- $bc + bde$ is not cube free

Kernel

- Kernels of f are
 - cube free primary divisors of f
- $f=abc+abde$
- $f/ab = c+de$ is a kernel
- ab is cokernel of f to $(c+de)$
 - cokernels always cubes

Kernel Extraction

- Find c largest cube factor of f
- $K=Kernel1(0,f/c)$
- if (f is cube-free)
 - return($f \setminus K$)
- else
 - return(K)
- Kernel1(j,g)
 - $R=g$
 - N max index in g
 - for($i=j+1$ to N)
 - if (li in 2 or more cubes)
 - c =largest cube divide g/li
 - if (forall $k \in I, k \nmid c$)
 - » $R=R \setminus c$
 - » $Kernel1(I, g/(li \setminus c))$
 - return(R)

Kernel Extract Example

- $f=abcd+abce+abef$
- $c=ab$
- $f/c=cd+ce+ef$
- $R=\{cd+ce+ef\}$
- $N=6$
- a,b not present
- $(cd+ce+ef)/c=e+d$
- largest cube 1
- recurse-> $e+d$
- $R=\{cd+ce+ef,e+d\}$
- only 1 d
- $(d+ce+ef)/e=c+f$
- recurse-> $c+f$
- $R+\{cd+ce+ef,e+d,c+f\}$

Factoring

- Gfactor(f)
 - if (terms==1) return(f)
 - $p=CHOOSE_DIVISOR(f)$
 - $(h,r)=DIVIDE(f,p)$
 - $f=Gfactor(h)*Gfactor(p)+Gfactor(r)$
 - return(f) // factored

Factoring

- Trick is picking divisor
 - pick from kernels
 - goal minimize literals after resubstitution

Extraction

- Identify cube-free expressions in many functions (common sub expressions)
- Generate kernels for each function
- select pair such that $k1 \setminus k2$ is not a cube
- new variable from intersection
- update functions (resubstitute)
- (similar for common cubes)

Extraction Example

- $X=ab(c(d+e)+f+g)+g$
- $Y=ai(c(d+e)+f+j)+k$

Extraction Example

- $X=ab(c(d+e)+f+g)+g$
- $Y=ai(c(d+e)+f+j)+k$
- $d+e$ kernel of both
- $L=d+e$
- $X=ab(cL+f+g)+h$
- $Y=ai(cL+f+j)+k$

Extraction Example

- $L=d+e$
- $X=ab(cL+f+g)+h$
- $Y=ai(cL+f+j)+k$
- kernels: $(cL+f+g)$, $(cL+f+j)$
- extract: $M=cL+f$
- $X=ab(M+g)+h$
- $Y=ai(M+f)+h$

Extraction Example

- | | |
|---------------------|-----------------|
| • $L=d+e$ | • $N=aM$ |
| • $M=cL+f$ | • $M=cL+f$ |
| • $X=ab(M+g)+h$ | • $L=d+e$ |
| • $Y=ai(M+f)+h$ | • $X=b(N+ag)+h$ |
| • no kernels | • $Y=I(N+aj)+k$ |
| • common cube: aM | |

Extraction Example

- | | |
|-----------------|---|
| • $N=aM$ | • Can collapse |
| • $M=cL+f$ | – L into M into N |
| • $L=d+e$ | • Get larger common kernel N |
| • $X=b(N+ag)+h$ | – maybe useful if components becoming too small for efficient gate implementation |
| • $Y=I(N+aj)+k$ | |

Resubstitution

- Also useful to try complement on new factors
- $f=ab+ac+b/cd$
- $X=b+c$
- $f=aX+b/cd$
- $/X=b/c$
- $f=aX+/Xd$
- ...extracting complements not a direct target

Good Divisors?

- Key to CHOOSE_DIVISOR in GFACTOR
- Variations to improve
 - e.g. rectangle covering

Boolean Division

- f/p assuming $\text{sup}(p) \supseteq \text{sup}(f)$
- add input P to f (for p)
- new function F
 - DC-set $D = P/p + Pp$
 - ON-set $f\bar{E}/D$
 - OFF-set $\bar{f}E/D$
- minimize F for minimum literals in sum-of-products

Boolean Division (check)

- Sanity: $(F * Pp + F * P/p)p = Fp$

Boolean Division Example

- $f = abc + a/b/c + ab/c + a/bc$
- $g = ab + a/b$
- $F\text{-DC} = ab/G + a/b/G + abG + a/bG$
- $F\text{-On} = abcG + a/bcG + ab/cG + a/b/c/G$
- minimize:
 - $F = cG + c/G$

Boolean Division

- Trick is finding boolean divisors
- Not as clear how to find
 - (not as much work on)
- Can always use boolean division on algebraic divisors
 - give same or better results

Don't Care

- Mentioned last time
- Structure of logic restricts values intermediates can take on

Don't Care Example

- $e = a/b$
- $g = c/d$
- $f = e/g$

- $e=0, a=0, b=0$ cannot occur
- DC: $e(a+b) + e/a/b$

Satisfiability DC

- In general:
 - $y = F(x,y)$
 - $y/F(x,y) + /yF(x,y)$ is don't care

Observability DC

- If F output not differentiated by input y ,
 - then don't care value of y
- $F_y = F_{/y}$

- $ODC = P(F_y = F_{/y})$
 - (product over all outputs)

Summary

- Want to exploit structure in problems to reduce (contain) size
 - common subexpressions
 - structural don't cares
- Identify component elements
 - decomposition, factoring, extraction
- Division

Today's Big Ideas

- Exploit freedom
 - form
 - don't care
- Exploit structure/sharing
 - common sub expressions