

CS137: Electronic Design Automation

Day 5: January 21, 2004
Multi-level Synthesis



CALTECH CS137 Winter2004 -- DeHon

Today

- Multilevel Synthesis/Optimization
 - Why
 - Transforms -- defined
 - Division/extraction
 - How we support transforms
 - Boolean Division
 - Higher quality division
 - Speedup sketch

CALTECH CS137 Winter2004 -- DeHon

Multi-level

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

CALTECH CS137 Winter2004 -- DeHon

Why multi-level?

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

CALTECH CS137 Winter2004 -- DeHon

Why multi-level?

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

CALTECH CS137 Winter2004 -- DeHon

Why Multilevel

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

CALTECH CS137 Winter2004 -- DeHon

Why Multilevel

- $a \text{ xor } b$
 - $x1=a/b+/ab$
- $a \text{ xor } b \text{ xor } c$
 - $x2=x1/c+/x1*c$
- $a \text{ xor } b \text{ xor } c \text{ xor } d$
 - $x3=x2/d+x2*d$
- Multi-level
 - exploit common sub-expressions
 - linear complexity
- Two-level
 - exponential complexity

CALTECH CS137 Winter2004 -- DeHon

Goal

- Find the structure
- Exploit to minimize gates
 - Total (area)
 - In path (delay)

CALTECH CS137 Winter2004 -- DeHon

Multi-level Transformations

- Decomposition
- Extraction
- Factoring
- Substitution
- Collapsing
- [copy these to board so stay up as we move forward]

CALTECH CS137 Winter2004 -- DeHon

Decomposition

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

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Extraction

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

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Factoring

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

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

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 $H=a+c?$ ($F=GH$)

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Moves

- These transforms define the “moves” we can make to modify our network.
- Goal is to apply, usually repeatedly, to minimize gates
 - ...then apply as necessary to accelerate design
- MIS/SIS
 - Applies to canonical 2-input gates
 - Then covers with target gate library
 - (back to day2)

CALTECH CS137 Winter2004 -- DeHon

Division

CALTECH CS137 Winter2004 -- DeHon

Division

- **Given:** function (f) and divisor (p)
- **Find:** quotient and remainder

$$f=pq+r$$

E.g.

$$f=abc+abd+ef, p=ab$$

$$q=c+d, r=ef$$

CALTECH CS137 Winter2004 -- DeHon

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:** $q=(a+c)$, $r=0$

CALTECH CS137 Winter2004 -- DeHon

Algebraic Division

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

CALTECH CS137 Winter2004 -- DeHon

Algebraic Division Example (stay on alg.; work ex on board)

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

CALTECH CS137 Winter2004 -- DeHon

Algebraic Division Example

- $f=abc+abd+de$, $p=ab+e$
- $p=\{ab, e\}$
- $h1=\{c, d\}$
- $h2=\{d\}$
- $h1 \cap h2=\{d\}$
- $f/g=d$
- $r=f - p * (f/g)$
- $r=abc+abd+de-(ab+e)d$
- $r=abc$

CALTECH CS137 Winter2004 -- DeHon

Algebraic Division Time

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

CALTECH CS137 Winter2004 – DeHon

Primary Divisor

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

CALTECH CS137 Winter2004 – DeHon

Cube Free

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

CALTECH CS137 Winter2004 – DeHon

Kernel

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

CALTECH CS137 Winter2004 – DeHon

Kernel Extraction

- Kernel1(j, g)
 - $R = g$
 - N max index in g
 - for($i = j + 1$ to N)
 - if (l_i in 2 or more cubes)
 - c_i = largest cube divide g/l_i
 - if (forall $k \leq i, l_k \notin c_i$)
 - » $R = R \cup$
 - » $\text{KERNEL1}(i, g/(l_i \cap c_i))$
 - return(R)
- Find c_i = largest cube factor of f
- $K = \text{Kernel1}(0, f/c_i)$
- if (f is cube-free)
 - return($f \cup K$)
- else
 - return(K)

Must be to Generate Non-trivial kernel

Consider each literal for cofactor once (largest kernels will already have been found)

CALTECH CS137 Winter2004 – DeHon

Kernel Extract Example (stay on prev. slide, ex. on board)

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

CALTECH CS137 Winter2004 – DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Factoring

- Trick is picking divisor
 - pick from kernels
 - goal minimize literals **after** resubstitution
 - Re-express design using new intermediate variables
 - Variable and complement

CALTECH CS137 Winter2004 -- DeHon

Extraction

- Identify cube-free expressions in many functions (common sub expressions)
- Generate kernels for each function
- select pair such that $k_1 \cap k_2$ is not a cube
- new variable from intersection
 - $v = k_1 \cap k_2$
- update functions (resubstitute)
 - $f_i = v * (f_i / v) + r_i$
 - (similar for common cubes)

CALTECH CS137 Winter2004 -- DeHon

Extraction Example

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

CALTECH CS137 Winter2004 -- DeHon

Extraction Example

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

CALTECH CS137 Winter2004 -- DeHon

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: $M = cL+f$
- $X = ab(M+g)+h$
- $Y = ai(M+f)+k$

CALTECH CS137 Winter2004 -- DeHon

Extraction Example

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

CALTECH CS137 Winter2004 -- DeHon

Extraction Example

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

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Good Divisors?

- Key to `CHOOSE_DIVISOR` in `GFACTOR`
- Variations to improve
 - e.g. rectangle covering (Devadas 7.4)

CALTECH CS137 Winter2004 -- DeHon

Boolean Division

CALTECH CS137 Winter2004 -- DeHon

Don't Care

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

CALTECH CS137 Winter2004 -- DeHon

Don't Care Example

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

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

CALTECH CS137 Winter2004 -- DeHon

Satisfiability DC

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

CALTECH CS137 Winter2004 -- DeHon

Observability DC

- If output F not differentiated by input y ,
 - then don't care value of y
 - i.e. input conditions where $(F_y = F_{/y})$

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

CALTECH CS137 Winter2004 -- DeHon

Don't Care Optimization

- Select a node in Boolean network
- Compute ODC for all outputs wrt node
- Compute SDC in terms of local inputs
- Minimize wrt $ODC \cup SDC$

CALTECH CS137 Winter2004 -- DeHon

Boolean Division

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

CALTECH CS137 Winter2004 -- DeHon

Boolean Division (check)

- F ON-set $f \cap D$
- $D = P/p + /Pp$
- $/D = Pp + /P/p$
- $F = f \cap D = F(Pp + /P/p)$
- $f = F(Pp + /P/p)p = Fp$
 - Goal since want $F = f//p$

CALTECH CS137 Winter2004 -- DeHon

Boolean Division Example

- $f = abc + a/b/c + ab/c + a/bc$
- $p = ab + a/b$
- $F\text{-}DC = ab/P + a/b/P + abP + a/bP$
- $F\text{-}On = abcP + a/bcP + ab/cP + a/b/cP$
- After minimization get:
 - $F = cP + c/P$
 - $P = ab + a/b$

CALTECH CS137 Winter2004 -- DeHon

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

CALTECH CS137 Winter2004 -- DeHon

Speed Up

(sketch flavor)

CALTECH CS137 Winter2004 -- DeHon

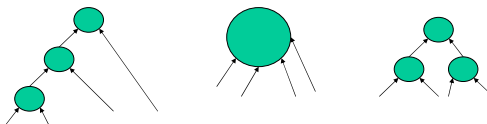
Speed Up

- Start with area optimized network
- Know target arrival times
 - Or know critical path
- Want to reduce delay of node

CALTECH CS137 Winter2004 -- DeHon

Basic Idea

- Improve speed by:
 - Collapsing node(s)
 - Refactoring collapsed subgraph to reduce height



CALTECH CS137 Winter2004 -- DeHon

Speed Up

- While (delay decreasing, timing not met)
 - Compute delay (slack)
 - Generate network close to critical path
 - Weight nodes in network
 - Compute **mincut** of nodes on weighted network
 - Partial collapse and timing redecompose on cut nodes

CALTECH CS137 Winter2004 -- DeHon

Weighted Cut

- Want to minimize area expansion
- Want to maximize likely benefit
 - Prefer nodes with varying input times
 - Prefer nodes with critical path on longer paths

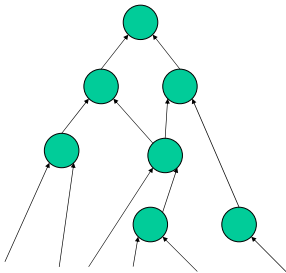
CALTECH CS137 Winter2004 -- DeHon

Timing Decomposition

- Extract area saving kernels that do not include critical inputs to node
- When decompose (e.g. into nand2's) similarly balance with critical inputs closest to output

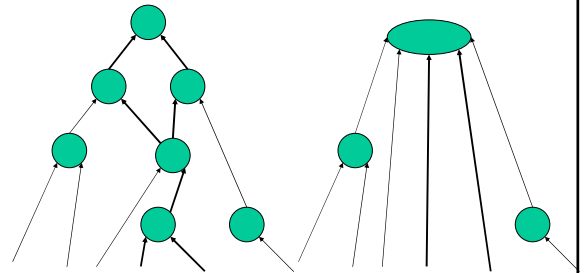
CALTECH CS137 Winter2004 -- DeHon

Example



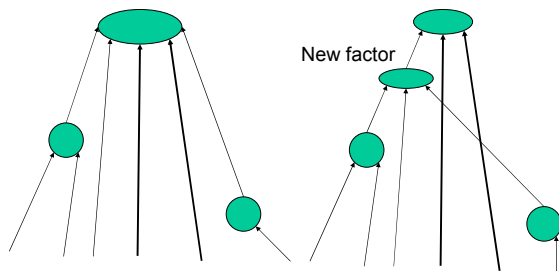
CALTECH CS137 Winter2004 -- DeHon

Example



CALTECH CS137 Winter2004 -- DeHon

Example



CALTECH CS137 Winter2004 -- DeHon

Summary

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

CALTECH CS137 Winter2004 -- DeHon

Admin

- Reading for next lecture on web
 - (exact state assignment TRCAD91)

Big Ideas

- Exploit freedom
 - form
 - don't care
- Exploit structure/sharing
 - common sub expressions
- Techniques
 - Iterative Improvement
 - Refinement/relaxation