Today

- Multilevel Synthesis/Optimization
  - Why
  - Transforms -- defined
  - Division/extraction
    - How we support transforms

Multi-level Logic

- General circuit netlist
- May have
  - sums within products
  - products within sum
  - arbitrarily deep
- \( y=((a \cdot (b+c))\cdot e)\cdot f\cdot g\cdot h\cdot i \)

Why Multi-level Logic?

- \( ab(c+d+e)(f+g) \)
- \( abcf+abdf+abef+abcg+abd+g+abeg \)
- 6 product terms \( \rightarrow \) 23 2-input gates
- vs. 3 gates: and4, or3, or2 \( \rightarrow \) 6 2-input gates
- Aside from Pterm sharing between outputs,
  - two level cannot share sub-expressions

Why Multilevel

- \( a \oplus b \)
  - \( a/b+/ab \)
- \( a \oplus b \oplus c \)
  - \( a/bc+/abc+/a/b/c+ab/c \)
- \( a \oplus b \oplus c \oplus d \)
  - \( a/bcd+/abcd+/a/b/c+d+/a/b/c+d+/a/bc/d/d+/a/bc/d/ \)

Compare

- \( a \oplus b \)
  - \( x_1=a/b+/ab \)
- \( a \oplus b \oplus c \)
  - \( x_2=x_1/c+/x_1/c \)
- \( a \oplus b \oplus c \oplus d \)
  - \( x_3=x_2/d+/x_2/d \)
### Why Multilevel

- **a xor b**
  - $x_1 = a/b + ab$
- **a xor b xor c**
  - $x_2 = x_1/c + x_1^*c$
- **a xor b xor c xor d**
  - $x_3 = x_2/d + x_2^*d$

- **Multi-level**
  - exploit common sub-expressions
  - linear complexity
- **Two-level**
  - exponential complexity

### Goal

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

### Multi-level Transformations

- **Decomposition**
- **Extraction**
- **Factoring**
- **Substitution**
- **Collapsing**

[copy these to board so stay up as we move forward]

### Decomposition

- $F = abc + abd + /a/c/d + /b/c/d$
  - 4 3-input + 1 4-input $\Rightarrow$ 11 2-input gates
- $F = XY + /X/Y$
- $X = ab$
- $Y = c + d$
  - 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

- \( F = (a+b)cd+e \)
- \( G = (a+b)/e \)
- \( H = cd \)
- 2-input: 4
- 3-input: 2
- \( \Rightarrow 8 \) 2-input gates

Factoring

- \( F = ac+ad+bc+bd+e \)
- \( G = X/e \)
- \( H = Y/e \)
- \( X = a+b \)
- \( Y = cd \)
- 2-input: 6

Common sub-expressions over multiple output

Factoring

- \( F = ac+ad+bc+bd+e \)
  - 4 2-input, 1 5-input \( \Rightarrow 8 \) 2-input gates
  - 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 \( H = a+c \), then \( F = GH \)

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 [saw last time]

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
  - Day 2
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 Example (adv to alg.; work ex on board)

- \( f = abc + abd + de \)
- \( p = ab + e \)
**Algebraic Division Example**

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

**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
- \( f = abc + abd e \)
- \( f/a = bc + bde \) 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 \( f \)
  - "Informally: sums w/ cubes factored out"
- \( f = abc + abd e \)
- \( f/a = c + de \) is a kernel
- \( ab \) is cokernel of \( f \) to \( (c + de) \)
  - cokernels always cubes

**Factoring**

- \( \text{Gfactor}(f) \)
  - if (terms == 1) return(f)
  - \( p = \text{CHOOSE_DIVISOR}(f) \)
  - \( (h, r) = \text{DIVIDE}(f, p) \)
  - \( f = \text{Gfactor}(h) \cdot \text{Gfactor}(p) + \text{Gfactor}(r) \)
  - return(f) // factored
Factoring

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

Kernel Extraction

- Kernel1(j,g)
  - R=g
  - N max index in g
  - for(i=1 to N) if (i in 2 or more cubes)
    - c_i = largest cube divide g_i
    - if (forall k ≤ i, l_k / c_i) » R=R ∪ KERNEL1(i,g/(l_i ∩ c_i))
  - return(R)

Consider each literal for cokernel once (largest cokernels will already have been found)

Kernel Extract Example (ex. on board; adv to return to alg.)

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

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

- f=abcd+abce+abef
  - c_f = ab
  - f/c_f = cd+ce+ef
  - R={cd+ce+ef}
  - N=6
  - a,b not present
  - (cd+ce+ef)/c = e+d

Kernel Extract Example

- Kernel1(j,g)
  - R=g
  - N max index in g
  - for(i=1 to N) if (i in 2 or more cubes)
    - c_i = largest cube divide g_i
    - if (forall k ≤ i, l_k / c_i)
      - R=R ∪ KERNEL1(i,g/(l_i ∩ c_i))
  - return(R)

Must be to Generate Non-trivial kernel

Consider each literal for cokernel once (largest cokernels will already have been found)

Extraction

Identify cube-free expressions in many functions (common sub expressions)

1. Generate kernels for each function
2. select pair such that k1∩k2 is not a cube
   - Note: k1=k2 is simplest case of this
     - but intersection case is more powerful
   - Example to come
3. new variable from intersection
   - v = k1∩k2
4. update functions (resubstitute)
   - f_i = v*(f_i/v) + r_i
   - (similar for common cubes)
Extraction Example

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

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

Extraction Example

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

- \( L = d+e \)
- \( M = cL+f \)
- \( X = ab(M+g)+h \)
- \( Y = ai(M+f)+h \)
- no kernels
- common cube: \( aM \)

- \( N = aM \)
- \( M = cL+f \)
- \( X = ab(M+g)+h \)
- \( Y = ai(M+f)+h \)
- \( L = d+e \)
- \( X = b(N+ag)+h \)
- \( Y = i(N+aj)+k \)

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
Multilevel Optimization

- Unlike Two-level, very heuristic
- Not clear when done
- Goal find common terms to share
- Often start with two-level optimization
  - Identifies product term sharing
- Identify kernels and cubes
- Factor them out
- If can be used many places, get benefit
- Sis included common recipes
- More after timing analysis

Summary

- Want to exploit structure in problems to reduce (contain) size
  - common sub-expressions
- Identify component elements
  - decomposition, factoring, extraction
- Division key to these operations
- Kernels give us divisors

Big Ideas

- Exploit freedom
  - form
- Exploit structure/sharing
  - common sub expressions
- Techniques
  - Iterative Improvement
  - Refinement/relaxation

Announcement

Friday April 5, 3:30pm:

Interesting talk on “Nano-scale VLSI Technologies: Silicon & Beyond”

Dr. Kevin Zhang, Intel Fellow
Room 337 Town Building

Admin

- Everyone should have received Assignment 6 feedback in email
- Reading for Monday on blackboard
- Milestone Mondays…