ECE 697B (667)

Spring 2003

Synthesis and Verification of Digital Systems

BDD-based Bi-decomposition

BDS system

Outline

- Review of current decomposition methods
  - Algebraic
  - Boolean
- Theory of BDD decomposition [C. Yang 1999]
  - *Bi-decomposition* $F = D \Theta Q$
  - Boolean *AND/OR* Decomposition
  - Boolean *XOR* Decomposition
  - *MUX* Decomposition
- Logic optimization based on BDD decomposition

Functional Decomposition – previous work

- Ashenhurst [1959], Curtis [1962]
  - Tabular method based on cut: bound/free variables
  - BDD implementation:
    - Lai *et al*. [1993, 1996], Chang *et al*. [1996]
    - Stanion *et al*. [1995]
- Roth, Karp [1962]
  - Similar to Ashenhurst, but using cubes, covers
  - Also used by SIS
- Factorization based
  - SIS, algebraic factorization using cube notation
  - Bertacco *et al*. [1997], BDD-based recursive bidecomp.

Drawbacks of Traditional Synthesis Methods

- Weak *Boolean* factorization capability.
- Difficult to identify *XOR* and *MUX* decomposition.
- Separate platforms for Boolean operations and factorization.

- Our goal: use a common platform to carry out both Boolean operations and factorization: BDD's

What is wrong with Algebraic Division?

- Divisor and quotient are orthogonal!!
- Better factored form might be:

  $(q_1+ q_2+ \ldots+q_n) (d_1+d_2+\ldots+d_m)$

  – g_i and d_j may share same or opposite literals

- Example:

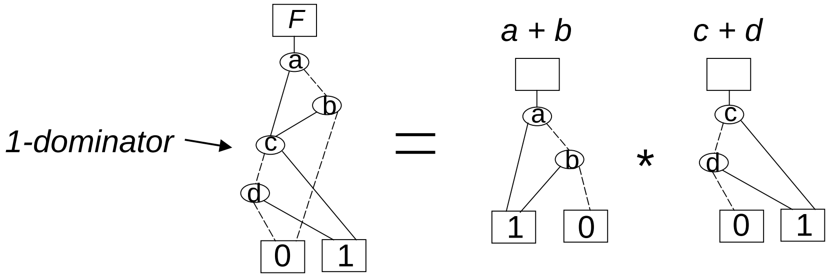
SOP form: $F = abg + acg + adf + aef + afg + bd + ce + be + cd$**.** (23 lits)

Algebraic: $F = (b + c)(d + e + ag) + (d + e +g)af$**.** (11 lits)

Boolean: $F = (af + b + c)(ag + d + e).$ (8 lits)

First work, Karplus [1988]: 1-dominator

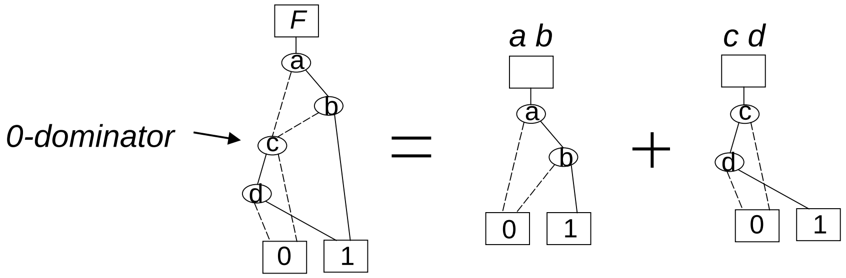
- Definition: *1-dominator* is a node that belongs to every path from root to terminal *1*.



- *1-dominator* defines <u>*algebraic*</u> *conjunctive (AND)* decomposition: $F = (a+b)(c+d)$.

Karplus: 0-dominator

- Definition: *0-dominator* is a node that belongs to every path from root to terminal *0*.



- *0-dominator* defines <u>*algebraic*</u> *disjunctive (OR)* decomposition: $F = ab + cd$.

Bi-decomposition based on Dominators

- We can generalize the concept of algebraic decomposition and dominators to:
  - Generalized dominators
  - <u>*Boolean*</u> bi-decompositions (AND, OR, XOR)
- Bi-decomposition: $F = D \, \Theta \, Q$
- First, let's review fundamental theorems for Boolean division and factoring.

Boolean Division

Definitions

- G is a *Boolean divisor* of F if there exist H and R such that

  $F = G H + R$, and $G H \neq 0$.

- G is said to be a *factor* of F if, in addition, $R=0$, that is:

  $F = G H$.

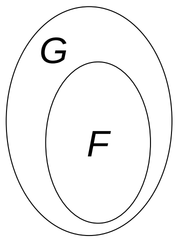
  where H is the quotient, R is the remainder.

  Note: H and R may not be unique.

Boolean Factor - Theorem

Theorem:

Boolean function G is a *Boolean factor* of Boolean function F iff $F \subseteq G$, (i.e. $FG' = 0$, or $G' \subseteq F'$).



Proof:

$\Rightarrow$: G is a Boolean factor of F. Then $\exists H$ s.t. $F = GH$; Hence, $F \subseteq G$ (as well as $F \subseteq H$).

$\Leftarrow$: $F \subseteq G \Rightarrow F = GF = G(F + R) = GH$. (Here R is any function $R \subseteq G'$.)

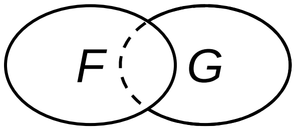
Notes:

- Given F and G, H is not unique.
- To get a small H is the same as getting a small $F + R$. Since $RG = 0$, this is the same as minimizing (simplifying) f with $DC = G'$.

Boolean Division - Theorem

Theorem:

G is a *Boolean divisor* of F if and only if $F G \neq 0$.



Proof:

$\Rightarrow$: $F = GH + R$, $GH \neq 0 \Rightarrow FG = GH + GR$. Since $GH \neq 0$, $FG \neq 0$.

$\Leftarrow$: Assume that $FG \neq 0$. $F = FG + FG' = G(F + K) + FG'$. (Here $K \subseteq G'$.)

Then $F = GH + R$, with $H = F + K$, $R = FG'$. Since $GH = FG \neq 0$, then $GH \neq 0$.

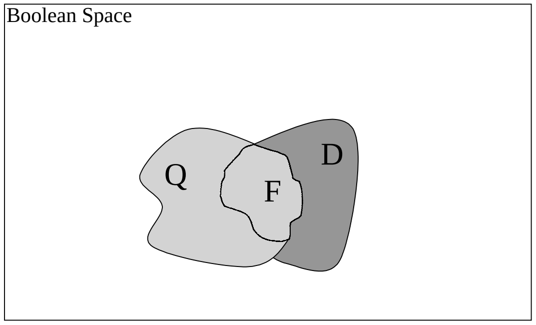
Note:

f has many divisors. We are looking for a g such that f = gh + r, where g, h, r are simple functions. (simplify f with DC = g')

Boolean Division

Goal : for a given F, find D and Q such that

$F = Q \cdot D$.

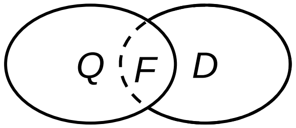


$F = e + bd$, $D = e + d$, $Q = e + b$

Conjunctive (AND) Decomposition

- Conjunctive (*AND*) decomposition: $F = D\,Q$.
- Theorem:

  Boolean function F has conjunctive decomposition iff $F \subseteq D$. For a given choice of *D*, the quotient *Q* must satisfy: $F \subseteq Q \subseteq F + D'$.

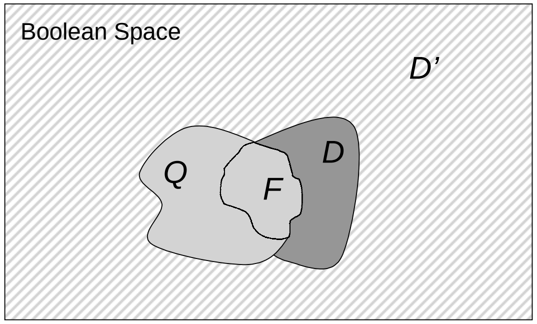


- For a given pair (*F,D*), this provides a recipe for *Q* .

Boolean Division ⇒ AND decomposition

Given function *F* and divisor $D \supseteq F$, find Q such that:

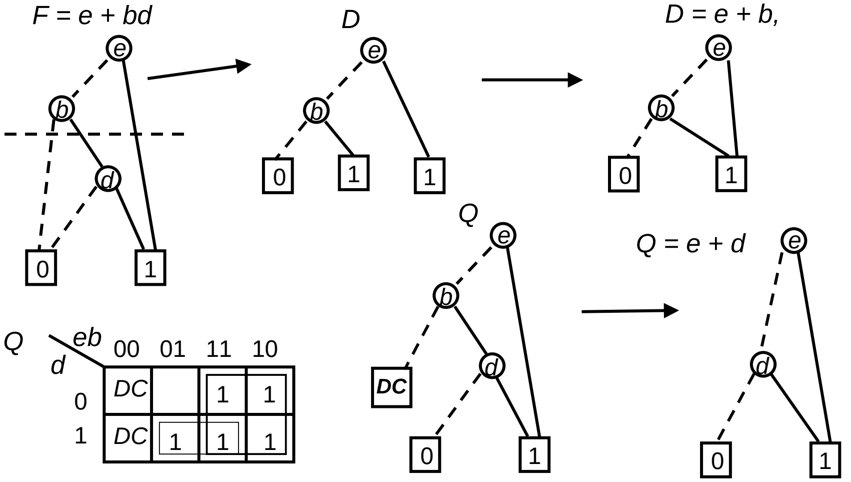
$$F \subseteq Q \subseteq F + D'.$$



$F = e + bd$, $D = e + d$, $Q = e + b \subseteq F + D'$

AND Decomposition ($F = D\,Q$): Example

- Recall: given (*F,D*), quotient *Q* must satisfy: $F \subseteq Q \subseteq F + D'$.



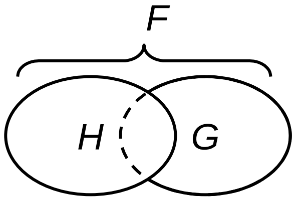
Q d \ eb	00	01	11	10
0	DC		1	1
1	DC	1	1	1

Disjunctive (OR) Decomposition

- Disjunctive (*OR*) decomposition: $F = G + H$.
- Theorem:

  Boolean function F has disjunctive decomposition iff $F \supseteq G$. For a given choice of *G*, the term *H* must satisfy: $F' \subseteq H' \subseteq F' + G$.

Dual to conjunctive decomposition.



- For a given (*F,G*), this provides a recipe for *H* .

Boolean AND/OR Bi-decompositions

- Conjunctive (AND) decomposition

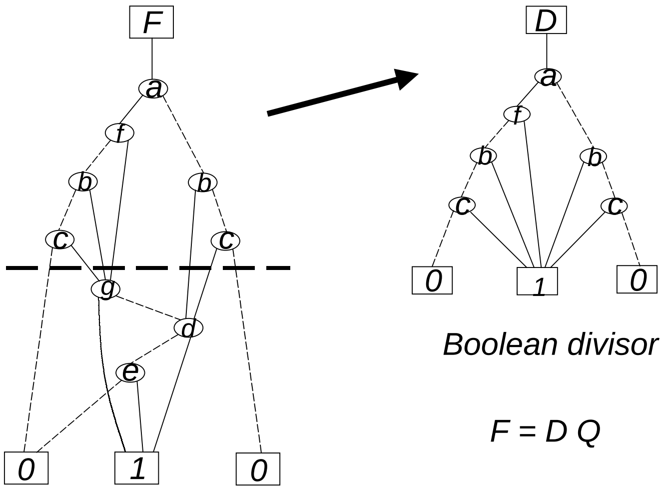
  If $D \supseteq F$, $F = F\,D = Q\,D$.

- Disjunctive (OR) decomposition

  If $G \subseteq F$, $F = F + G = H + G$.

- *D, G* = generalized dominators

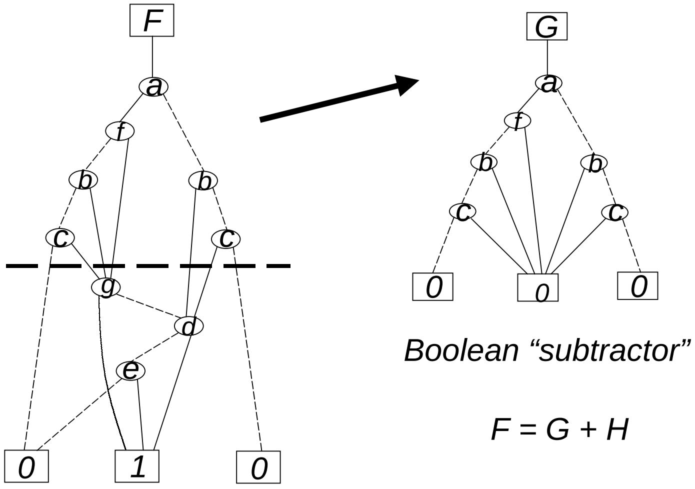
Generalized Dominator D



Boolean divisor

$F = D\,Q$

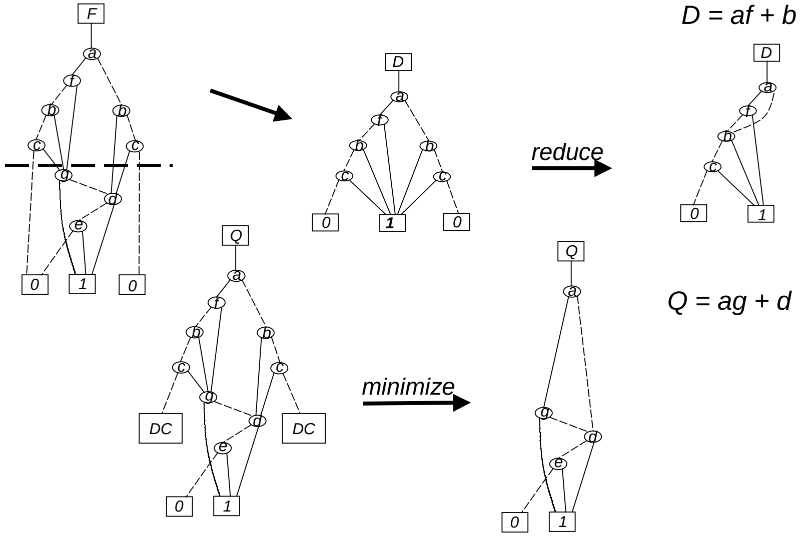
Generalized Dominator G



Boolean "subtractor"

$F = G + H$

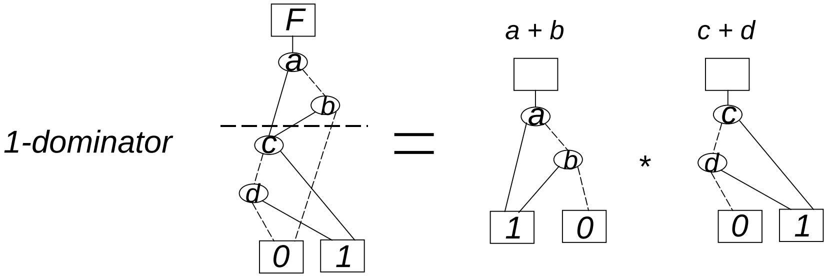
Boolean Division Based on *Generalized Dominator*



$D = af + b + c$

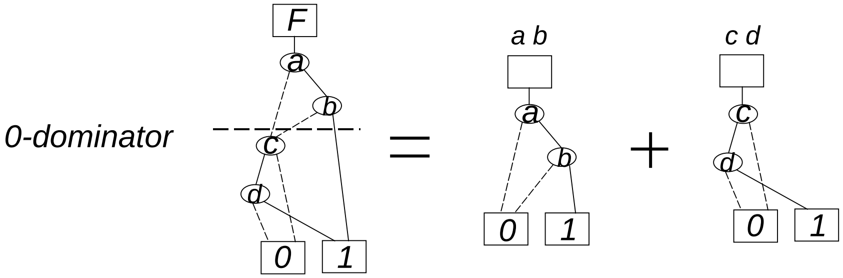
$Q = ag + d + e$

Special Case: *1-dominator*



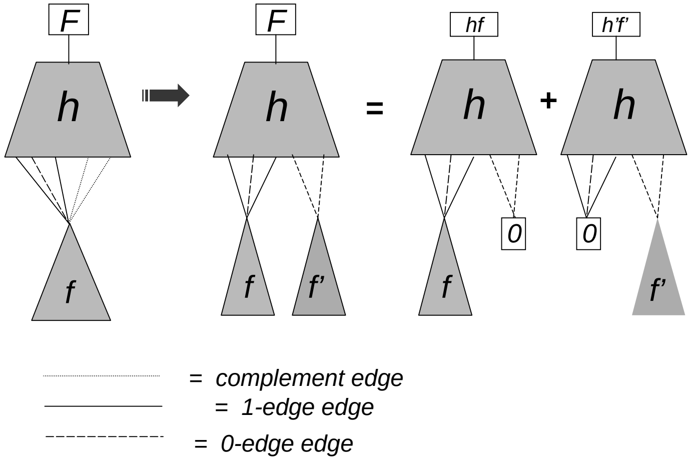
F = (a+b)(c+d)

Special Case: *0-dominators*

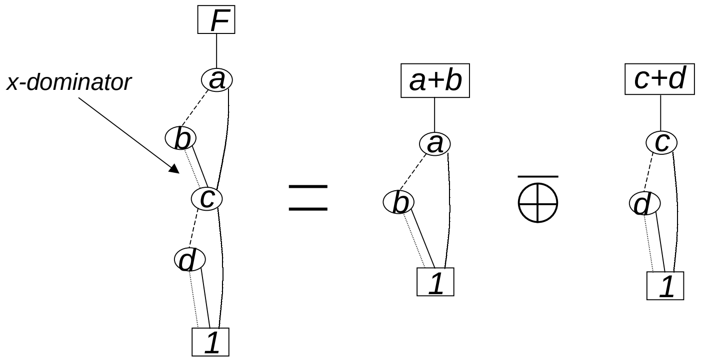


F = ab + cd

Algebraic *XOR* Decomposition

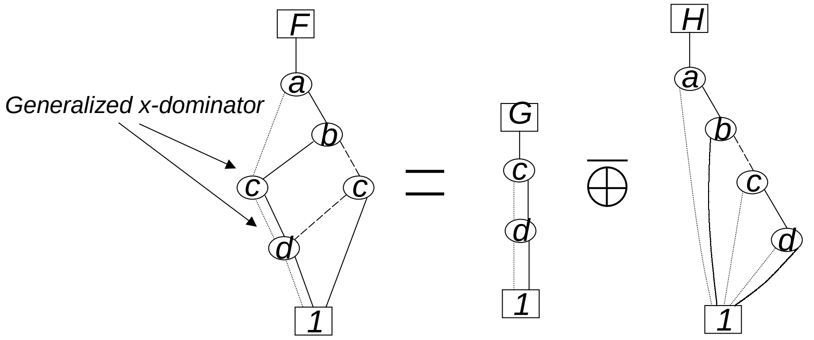


Algebraic *XOR* Decomposition: *x-dominators*



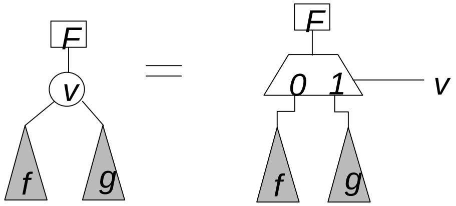
Boolean *XOR* Decomposition: *Generalized x-dominators*

Given *F* and *G*, there exists *H* : *F* = *G* ⊗ *H;* *H* = *F* ⊗ *G*.

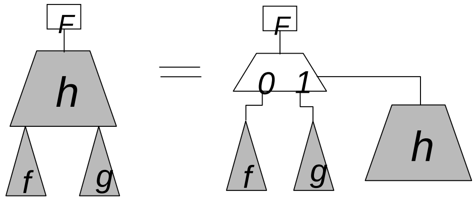


MUX Decomposition

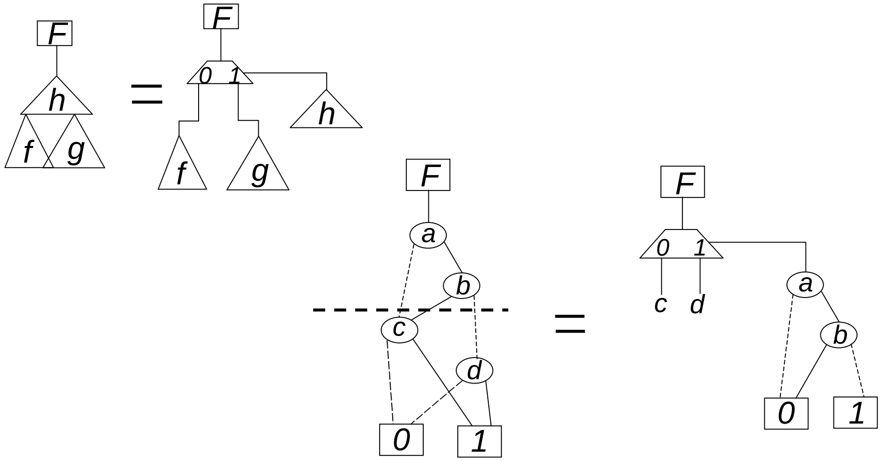
- Simple MUX decomposition



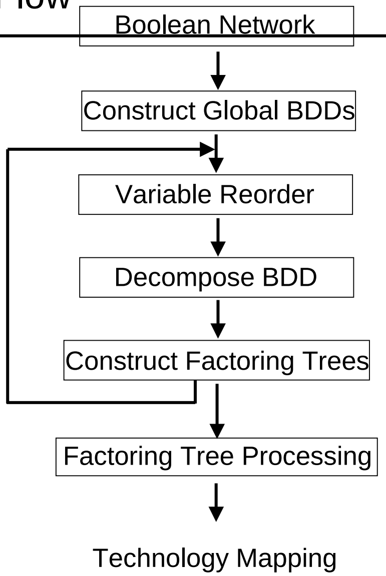
- Complex MUX decomposition



Functional *MUX* Decomposition - example



Synthesis Flow



Boolean Network

Construct Global BDDs

Variable Reorder

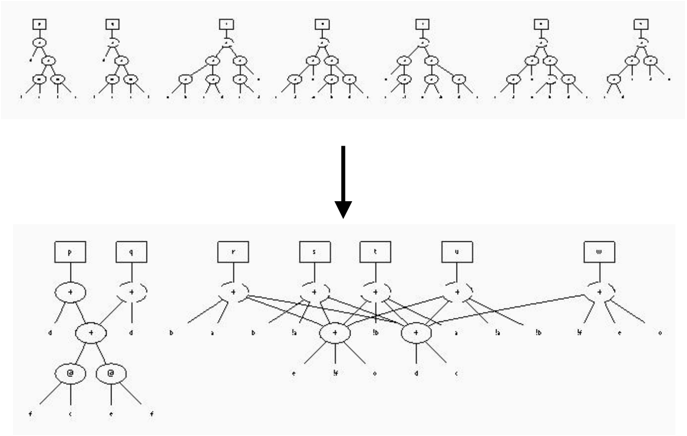
Decompose BDD

Construct Factoring Trees

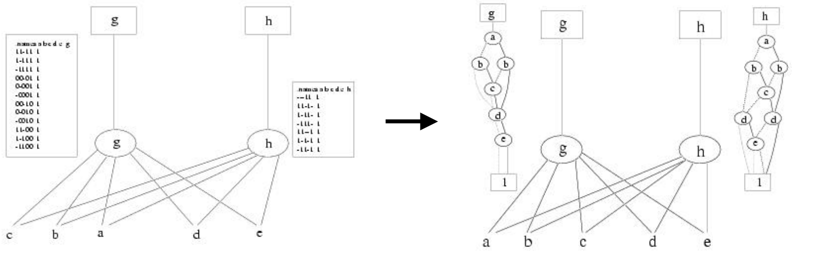
Factoring Tree Processing

Technology Mapping

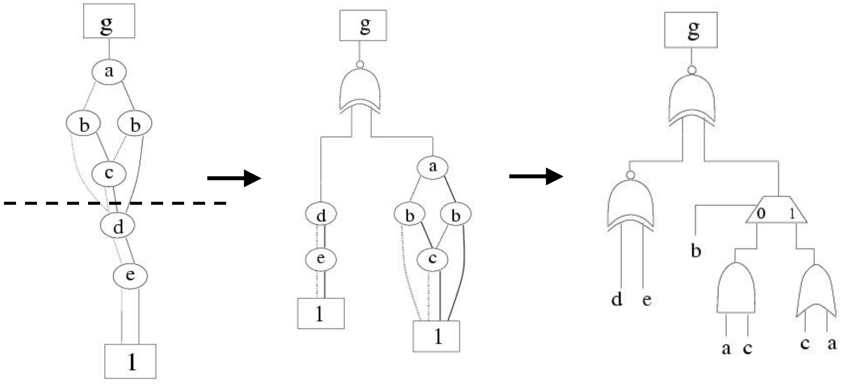
Factoring Tree Processing:



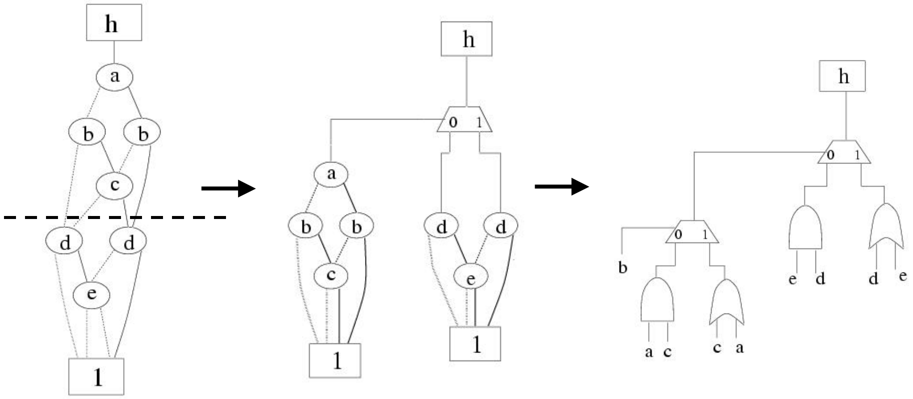
A Complete Synthesis Example



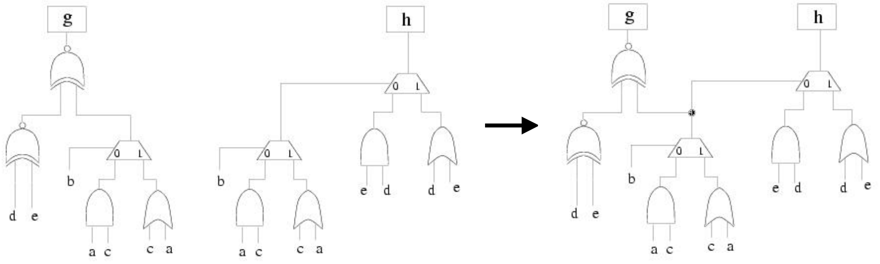
A Complete Synthesis Example (Decompose function *g*)



A Complete Synthesis Example (Decompose function *h*)



A Complete Synthesis Example (Sharing Extraction)



Conclusions

- BDD-based *bi-decomposition* is a good alternative to traditional, algebraic logic optimization
  - Produces *Boolean* decomposition
  - Several types: *AND, OR, XOR, MUX*
- BDD decomposition-based logic optimization is *fast*.
- Stand-alone BDD decomposition scheme is not amenable to large circuits
  - *Global BDD* too large
  - Must partition into network of BDDs (local BDDs)