ECE 510 OCE
BDDs and Their Applications

Lecture 16.
Functional Decomposition (2)

May 18, 2000
Alan Mishchenko

Overview

- Other approaches to functional decomposition
  - Recursive decomposition (Bertacco/Damiani,ICCAD'97)
  - Bi-decomposition using 1-, 0-, and x-dominators (Yang/Ciesielski,ICDD'99)
  - Implicit, multi-output decomposition (Wurth/Eckl/Legl,DAC'95)
  - Decomposition based on Information Measures (Chojnacki/Jozwiak,1997)
    presented by Artur Chojnacki

Types of Decomposition

- Simple (Ashenhurst) - intermediary signal is binary / "Complex" (Curtis) - intermediary signal is multi-valued
- Completely specified / Incompletely specified functions
- Single output / Multi-output functions
- Binary / Multi-valued functions
- Multi-valued functions / relations
- Specialized types
  - Bi-decomposition
  - Decomposition with feedback (combinational loops)

Recursive Decomposition

- Bertacco/Damiani method (ICCD'96, ICCAD'97) is the only recursive algorithm of this kind
- The main feature of recursive decomposition is that decomposition of the functions is derived from the decomposition of its cofactors
- This algorithm visits each node of the BDD for function F once and derives the decomposition tree (or netlist composed of two-input gates), which is canonical (!) for given variable order.

Lemma

Lemma (Ashenhurst). Decomposition of a completely specified function F over n variables using k (k < n) non-constant, disjoint-support, single-output functions $A_i(X_i)$:

$F(X) = L(A_1(X_1),A_1(X_2),\ldots,A_k(X_k))$,

such that $\cup X_i = X$, is unique up to permutation / complementation of inputs of $L(A_1, A_1,\ldots,A_k)$

In this formulation, X is the set of input variables of F, while X_i are sets of input variables of A_i

Definitions

Definition. Function L is the decomposition base. The set of inputs $\{A_1, A_1,\ldots,A_k\}$ of L is the decomposition list of F w.r.t. L (denoted F/L). Decomposition base and decomposition list together constitute decomposition of F, (L, F/L). Function F is prime if it cannot be decomposed using any $L(A_1, A_1,\ldots,A_k)$, k < n.

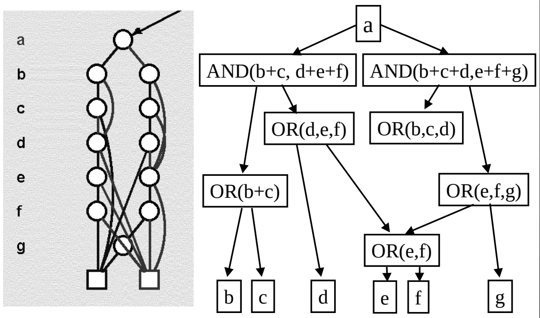
Property. If L has support size two ($|S_L| = 2$), then F is decomposable in *exactly one* of the following ways:

- as an OR of disjoint-support functions
- as an AND of disjoint-support functions
- as an EXOR of disjoint-support functions

Recursive Decomposition Procedure

```
typedef decomposition DEC; // the data type for decomposition (F,F/L)
DEC Dec_recursive( bdd F )
{   if ( F = 0 || F = 1 )     return EMPTY_DECOMPOSITION;
    // check cache for results
    Var is the topmost variable of F; DEC Res;
    DEC D0 = Dec_recursive( F0 );
    DEC D1 = Dec_recursive( F1 );
    if ( F0 = const || F1 = const )  Res = Dec_or_and( Var, D0, D1 );
    else  if ( F0 = !F1)              Res = Dec_exor( Var, D0, D1 );
    else if ( Var belongs to a subfunction of D0 or D1 )
                                      Res = Dec_special_case( Var, D0, D1 );
    else                              Res = Dec_prime( Var, D0, D1 );
    // insert into cache
    return Res; }
```


Example $F = a'(b+c)(d+e+f) + a(b+c+d)(e+f+g)$



Bi-Decomposition

Definition. Bi-decomposition of function F(X) is representation

$F(X) = Q(X_1)\ \Theta\ D(X_2)$,

where $Q(X_1)$ and $D(X_2)$, $X_1 \cup X_2 = X$, are boolean functions and Θ is a boolean operator (AND, OR, EXOR, etc).

If the sets of variables X_1 and X_2 are disjoint, then bi-decomposition is called disjoint bi-decomposition. This approach presented here gives non-disjoint bi-decompositions.

Bi-Decomposition Using Dominators

Definition. BDD of function F is composed of the set of nodes V (including terminal nodes) and the set of edges E.
Definition. A cut (D,V-D) of the BDD is a partition of V into disjoint subsets D and V-D such that the root of F belongs to D and terminal nodes {0,1} belong to V-D. Additionally, it is required that the cut did not cross any path from the root to terminals more than once. A horizontal cut is the cut, in which the supports of D and V-D are disjoint.

Definition. A node that has one of its edges connected to the terminal 0(1), is called 0(1)-dominator. A node that is contained on every path of the BDD is called x-dominator.

Deriving AND-Bi-Decompositions

To derive AND-bi-decomposition of F(X), do the following:

1) Reorder variables to get a small BDD for F(X)

2) Find a cut crossing at least one edge to 0-terminal

3) Create a BDD for $D(X_2)$ from nodes in the BDD for F(X) above the cut in a such a way that the edges going to 0-terminal from above the cut remain intact, while all edges crossing the cut (dangling edges) are redirected to 1-terminal. (X_2 consists of the variables above the cut.)

4) Get the BDD for $Q(X_1)$ by minimizing BDD for F(X) w.r.t. the OFF-set of $D(X_2)$ as the DC set. (Notice that, in general, X_1 consists of all variables X. In practice, after BDD minimization, X_1 often depends on a subset of variables X.)

Deriving OR-Bi-Decompositions

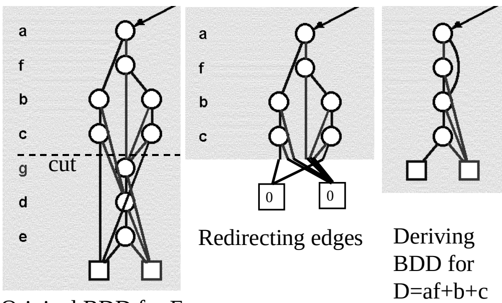
Deriving OR-bi-decompositions is similar, except:

during step 2) a cut is found crossing edges to 1-terminal (instead of 0-terminal)

during step 3) the dangling edges are redirected to 0-terminal (instead of 1-terminal)

during step 4) $Q(X_1)$ is received by minimizing the BDD for F(X) w.r.t. the ON-set of $D(X_2)$ as a DC set (instead of OFF-set)

Example $F = (af+b+c) \& (ag+d+e) = D \& Q$

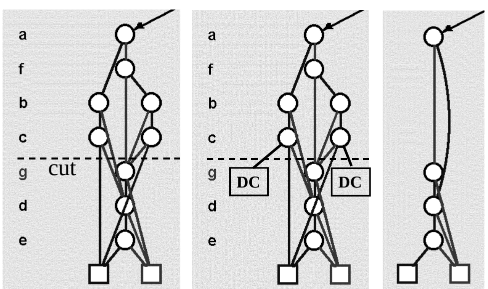


Original BDD for F

Redirecting edges

Deriving BDD for D=af+b+c

Example $F = (af+b+c) \& (ag+d+e)$



Original BDD for F

Finding DC set

Q=ag+d+e