ECE 3060
VLSI and Advanced Digital Design

Lecture 11

Binary Decision Diagrams

Outline

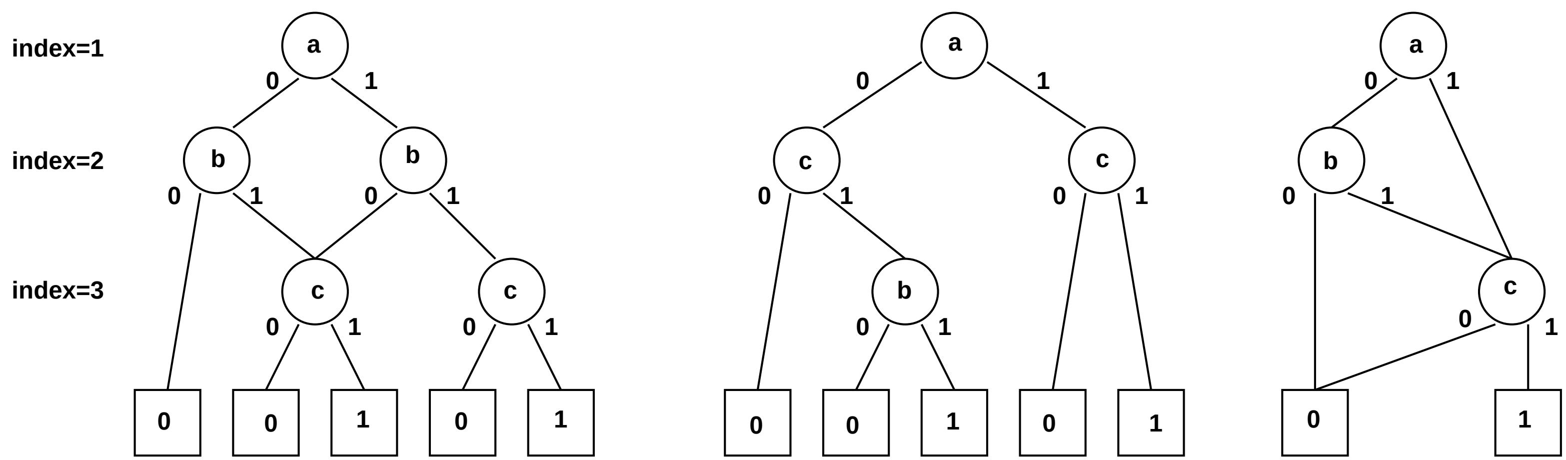
- **Binary Decision Diagrams (BDDs)**
- **Ordered (OBDDs) and Reduced Ordered (ROBDDs)**
- **Tautology check**
- **Containment check**

History

- **Efficient representation of logic functions**
    - **Proposed by Lee and Akers**
    - **Popularized by Bryant (canonical form)**
- **Used for Boolean manipulation**
- **Applicable to other domains**
    - **Set and relation representation**
    - **Formal verification**
    - **Simulation, finite-system analysis, ...**

Definitions

- **Directed Acyclic Graph (DAG)**
  - **vertex set V**
  - **edge set E (each edge has a head and tail => a direction)**
  - **no cycles exist in G(V,E)**
- **Binary Decision Diagram (BDD)**
  - **tree or rooted DAG where each vertex denotes a binary decision**
  - **Example:** $F = (a+b)c$



(a) (b) (c)

Definition of OBDD

- **Ordered Binary Decision Diagram (OBDD)**
  - **the tree (or rooted DAG) can be levelized, so that each level corresponds to a variable**
- **Implementation: each non-leaf vertex v has**
  - **a pointer index(v) to a variable**
  - **two children low(v) and high(v)**
- **Each leaf vertex v has a value (0 or 1)**
- **Ordering:**
  - **index(v) < index(low(v))**
  - **index(v) < index(high(v))**

Properties of an OBDD

- **Each OBDD with root v defines a function f^v:**
  - **if v is a leaf with value(v) = 1, then** $f^v = 1$
  - **if v is a leaf with value(v) = 0, then** $f^v = 0$
  - **if v is not a leaf and** $\mathrm{index}(v) = i$**, then** $f^v = \overline{x_i} \cdot f^{\mathrm{low}(v)} + x_i \cdot f^{\mathrm{high}(v)}$
- **OBDDs are not unique therefore a function may have many OBDDs**
- **The size of an OBDD depends on the variable order**

Cofactor and Boolean expansion

- **Function** $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ **)**
- **Definition: cofactor of** f **with respect to** x_i**:**

$$f_{x_i} = f(x_1, x_2, \ldots, 1, \ldots, x_n)$$

- **Definition: cofactor of** f **with respect to** $\overline{x}_i$**:**

$$f_{\overline{x}_i} = f(x_1, x_2, \ldots, 0, \ldots, x_n)$$

- **Theorem: Let** $f : B^n \to B$**. Then**

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = \overline{x}_i \cdot f_{\overline{x}_i} + x_i \cdot f_{x_i}$$

Example

- **Function** $f = ab + bc + ac$
- **Cofactors:**

$$f_a = b + c \text{ and } f_{\bar{a}} = bc$$

- **Expansion:**

$$f = \bar{a} \cdot f_{\bar{a}} + a \cdot f_a = \bar{a}bc + a(b + c)$$

ROBDDs

- **Reduced Ordered Binary Decision Diagrams have no redundant subtrees:**
  - **no vertex with low(v) = high(v)**
  - **no pair {u, v} with isomorphic subgraphs rooted in u and v**
- **Reduction can be achieved in time polynomial with respect to the number of vertices**
- **However the number of vertices may be exponential in the number of input variables**
- **ROBDDs can be such by construction**
- **An ROBDD is a canonical form**
- **Example: OBDD (c) on slide 4**

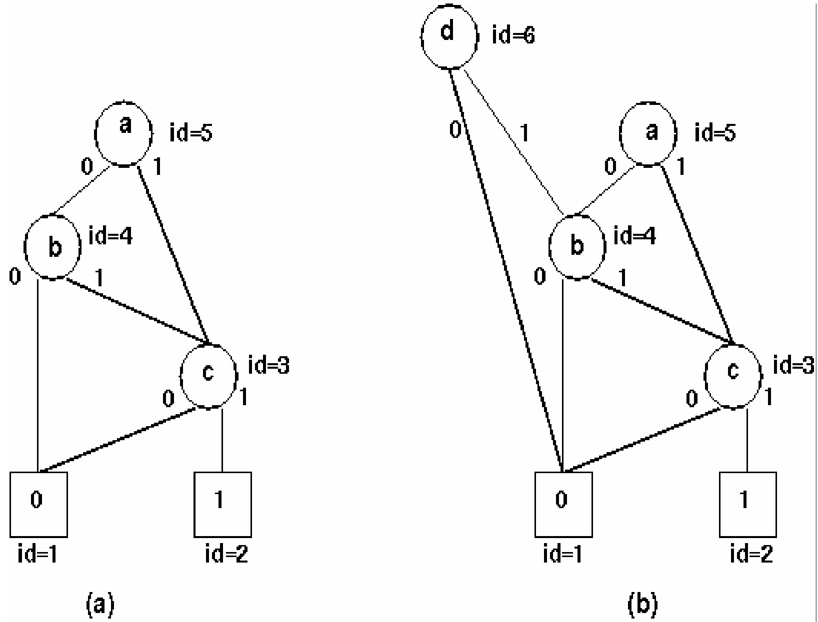
Features

- Canonical form allows us to
  - verify logic equivalence in constant time
  - check for tautology and perform logic operations in time proportional to the graph size
- Drawback:
  - ROBDD graph size depends heavily on variable order
- ROBDD size bounds
- Multiplier:
  - exponential size
- Adders:
  - exponential to linear size
- Sparse logic:
  - good heuristics exist to keep size small

Tabular representation of ROBDDs

- Represent multi-rooted graphs
  - multiple-output functions
  - multiple-level logic forms
- Unique table
  - one row per vertex
  - identifier
  - key: (variable, left child, right child)

Example: Unique Table



Identifier	Key		
	Variable	Left Child	Right Child
6	d	1	4
5	a	4	3
4	b	1	2
3	c	1	2

Tautology Checking

- **Check if a function is always TRUE**
- **Recursive method:**
  - **expand about a variable appearing both complemented (in an implicant) and uncomplemented (in another implicant)**
  - **if all cofactors are TRUE then the function is a tautology**
  - **if any cofactor is not a tautology (i.e., not TRUE), then the function is not a tautology**
- **A function is a tautology iff all of it's cofactors are tautologies**
- **A function is a tautology iff all of the leaves of it's BDD are TRUE**
- **This can be accomplished by traversing the BDD**

Containment Checking

- **Theorem: A cover F contains an implicant α iff F_α is a tautology.**
- **Consequence: containment can be verified by computing the cofactor and checking if it is a tautology.**
- **In general, how do we compute a cofactor?**

Cofactor Computation

- **An arbitrary cofactor of F can be computed from a BDD of F.**
- **Suppose we have an ROBDD for F and we wish to compute F_α, where $\alpha = \prod_{i \in \alpha} x_i$.**
- **First we note that $F_{x_i x_j} = (F_{x_i})_{x_j}$ so we compute the cofactor with respect to a product of literals by considering the literals one at a time.**
- **Consider the cofactor wrt x_i: For each node at index i, trim the BDD by removing the edge associated with $\overline{x_i}$, and move the edge associated with x_i to the parent.**

Example

- **Consider** $F = abc + ab\bar{c} + a\bar{b} + \bar{b}c$
- **Construct an ROBDD for** F
- **Is** a **contained in** F**?**
- **Is** b **contained in** F**?**
- **Is** $\bar{c}$ **contained in** F**?**

Other Uses of BDDs

- **Further uses of BDDs**
- **Can efficiently calculate complement**

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = \overline{x}_i \cdot \overline{f_{\overline{x}_i}} + x_i \cdot \overline{f_{x_i}}$$

- **Can efficiently calculate union, intersection**
- **Equivalence checking**

Summary: BDDs

- **Used mainly in multiple-level logic minimization**
- **Also used in formal verification**
- **Very efficient algorithms:**
  - **most manipulations (tautology check, complementation, etc.) can be done in time polynomial in the size of the BDD**