Lecture 23: Logic Synthesis (2)

Slides courtesy of Deming Chen

Some slides Courtesy of Prof. J. Cong of UCLA
Outline

- Reading
  - Synthesis and optimization of digital circuits, G. De Micheli, 1994, Section 2.5.2

- Overview
  - BDD (binary decision diagram)
  - OBDD (ordered BDD)
  - ROBDD (reduced OBDD)
  - ITE (if-then-else operator)
  - Introduction of mapping
Binary Decision Diagrams

- Efficient representation of logic functions
  - Proposed by Lee and Akers.
  - Popularized by Bryant (canonical form)
- Used for Boolean manipulations
- Applicable to other domains:
  - Set and relation representation
  - Simulations, finite-system analysis

Definition:
- A BDD is a tree or a rooted DAG with a decision at each vertex.
Ordered BDD

- Each decision is the evaluation of a boolean variable
- The tree (or dag) can be levelized so that each level corresponds to a Boolean variable
- Definition
  - Rooted DAG
  - Each non-leaf vertex \( \mathbf{v} \) has
    - A pointer index to a variable
    - Two children: \emph{low}(\mathbf{v}) and \emph{high}(\mathbf{v})
  - Each leaf vertex \( \mathbf{v} \) has a value (1 or 0)
- Ordering
  - \emph{index}(\mathbf{v}) < \emph{index}(\emph{low}(\mathbf{v}))
  - \emph{index}(\mathbf{v}) < \emph{index}(\emph{high}(\mathbf{v}))
More OBDDs

- Each OBDD with root $v$ defines a function $f^v$:
  - If $v$ is a leaf (1) then $f^v = 1$
  - If $v$ is a leaf (0) then $f^v = 0$
  - If $v$ is not a leaf and $\text{index}(v) = i$ then
    $$f^v = x'_i \cdot f^{\text{low}(v)} + x_i \cdot f^{\text{high}(v)}$$

- A function may have different OBDDs

- The size of an OBDD depends on the variable ordering
Ordered BDD Examples

Example: $f = (a+b) \cdot c$
Reduced OBDDs

- No redundancy
  - no vertex with \( \text{low}(v) = \text{high}(v) \)
  - no pair \( \{u, v\} \) with isomorphic subgraphs rooted at \( u, v \)
- Reduction can be achieved in polynomial time
- ROBDDs can be such by construction
- ROBDDs (canonical) allow us to
  - Verify logic equivalence in constant time
  - perform logic operations in time proportional to the graph size (vertex cardinality)
  - Although in some cases this graph approach is of exponential complexity, many cases result in very compact representations
- Drawback: ROBDD size depends on variable ordering
OBDD Reduction Algorithm

Outline
1) Visit OBDD bottom up
2) Label each vertex \( v \) with an identifier \( id(v) \)
3) For each \( v \) in the subset of vertices with index \( i \) (level \( i \))
   a) If \( id(low(v)) = id(high(v)) \), we set \( id(v) = id(low(v)) \) (\( v \) is redundant)
   b) If there are \( u \) & \( v \) such that \( id(low(v)) = id(low(u)) \) and \( id(high(v)) = id(high(u)) \) then we set \( id(v) = id(u) \) (isomorphic)
4) In the remaining cases, different \( id \) given to each vertex at level \( i \)
5) Terminate at the root.
Example
The *ite* Operator

- Apply operators to ROBDDs
  - Three boolean functions: $f, g, h$ with top variable $x$
    - $ite(f, g, h)$
      - If $f$ then $g$ else $h$
    - $fg + f' h$
  - Recursive Property:
    $$ite(f, g, h) = ite(x, ite(f_x, g_x, h_x), ite(f'_x, g'_x, h'_x))$$
Examples

- Apply and to two ROBDDs: \( f, g \)
  \[ fg = \text{ite}(f, g, 0) \]

- Apply or to two ROBDDs: \( f, g \)
  \[ f + g = \text{ite}(f, 1, g) \]

- Similar for other Boolean operators
ITE - Boolean operators

0 : ite(0,0,0)  
1 : ite(1,1,1)  
f : ite(f,1,0)  
g : ite(g,1,0)  
f' : ite(f,0,1)  
g' : ite(g,0,1)  
f·g : ite(f,g,0)  
(f·g)' : ite(f,g',1)  
f·g' : ite(f,g',0)  
(f+g)' : ite(f,0,g')  
f+g : ite(f,1,g)  
f+g' : ite(f,1,g')  
f ⊕ g : ite(f,g',g)  
f xnor g : ite(f,g,g')
The *ITE* algorithm

- Used to construct the ROBDD of a function.

- Evaluate the $\text{ite}(f,g,h)$ operator recursively – bottom up

- Keep OBDDs in reduced form

- Use two tables (per function):
  - Unique table: represents ROBDD
  - Computed table: stores previous info

- Smart implementations of *ITE* have linear time complexity in the product of the ROBDD sizes.
ITE (if-then-else) Algorithm

ITE(f,g,h) /* one call of ITE */
    if (terminal case)
        return (r=trivial result);
    else {
        if (computed table has entry {(f,g,h),r})
            return (r from computed table)
        else {
            x = top var of f,g,h
            t = ite(f_x, g_x, h_x)
            e = ite(f'_x, g'_x, h'_x)
            if (t==e) /* isomorphic */
                return (t);
            r = find_or_add_unique_table(x,t,e);
            update computed table with {(f,g,h),r}
            return (r);
        }
    }

Intro. VLSI System Design
Example using ITE Algorithm

- \( f = ac + bc \)
Advantages of ROBDDs

- Several algorithms for ROBDD manipulation
  - Polynomial time

- Very often the ROBDDs have small size
  - Various heuristic algorithms available for ROBDD reordering

- Software packages available
Design Flow for Programmable Logic

**RTL Design**
- **RTL design**
  - RTL elaboration and optimization
- **Architecture-independent optimization**
- **Technology mapping & Architecture-specific optimization**

**Logic Synthesis**
- **Clustering & placement**
- **Placement-driven optimization & incremental placement**

**Physical Design**
- **Routing**
- **Bitstream generation**

**Controller**
- **Datapath**

**Net List**
- **RTL Synthesis**

**Bitstream generation**
Technology Mapping

- Technology mapping converts a given Boolean circuit (a netlist) into a functionally equivalent network comprised only of LUTs or PLAs
- Technology mapping is a crucial optimization step in the programmable logic design flow
- Direct impact on
  - Delay (number of levels of logic)
  - area/power (number of LUTs or PLAs)
  - Interconnects (number of edges)
Definitions and Motivation

- **DAG**: Boolean network
- **Cone** $C_v$: sub-network rooted on node $v$
- **$K$-feasible cone**: $|\text{input}(C_v)| \leq K$
- **Fanin Cone** $F_v$: the largest $C_v$
- **$K$-feasible cut**: a $K$-feasible $C_v$
- **Unit delay model**:
  - One LUT contributes one unit delay
  - No edge delay
Cut Enumeration

Combine sub-cuts on the inputs of the gate
Process each gate in topological order from PIs to POs
Cut Selection – Mapping Generation

- From POs to PIs
- Critical paths
  optimal delay + best area available
- Non-critical paths
  relaxed delay + better area

LUT roots in list L
- L: f, g
- L: g, e, d
- L: e, d
- L: b
## Detailed Experimental Results on Industrial Benchmarks

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>CutMap</th>
<th>DAOMap</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LUT No.</td>
<td>Run Time (s)</td>
<td>LUT No.</td>
</tr>
<tr>
<td>big1</td>
<td>9928</td>
<td>301</td>
<td>9169</td>
</tr>
<tr>
<td>big2</td>
<td>-</td>
<td>&gt;10H.</td>
<td>14625</td>
</tr>
<tr>
<td>big3</td>
<td>10005</td>
<td>28926</td>
<td>9031</td>
</tr>
<tr>
<td>big4</td>
<td>11800</td>
<td>583</td>
<td>9364</td>
</tr>
<tr>
<td>big5</td>
<td>-</td>
<td>&gt;10H.</td>
<td>32230</td>
</tr>
<tr>
<td>big6</td>
<td>39000</td>
<td>14437</td>
<td>32028</td>
</tr>
<tr>
<td>Ave.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After mapping into 5-LUTs
Summary

- BDD is a popular way to represent and optimize logic
- ROBDD useful for optimization and verification
- Logic synthesis includes technology independent optimization and technology dependent optimization
- Logic synthesis is in general a mature field

- Next lecture
  - Placement