Sequential Equivalence Checking - II

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab.
Dept. of Electrical Engineering
Indian Institute of Technology Bombay
viren@ee.iitb.ac.in

EE 709: Testing & Verification of VLSI Circuits
Lecture – 17 (Feb 08, 2012)
Reachability-Based Equivalence Checking

Approach 3: Symbolic Traversal Based Reachability Analysis

- Build product machine of $M_1$ and $M_2$
- Traverse state-space of product machine starting from reset states $S_0$, $S_1$
- Test equivalence of outputs in each state
- Can use any state-space traversal technique
Sequential Verification

• Symbolic FSM traversal of the product machine

• Given two FSMs: \( M_1(X,S_1, \delta_1, \lambda_1,O_1) \), \( M_2(X,S_2, \delta_2, \lambda_2,O_2) \)

• Create a product FSM: \( M = M_1 \times M_2 \)
  - traverse the states of \( M \) and check its output for each transition
  - the output \( O(M) = 1 \), if outputs \( O_1 = O_2 \)
  - if all outputs of \( M \) are 1, \( M_1 \) and \( M_2 \) are equivalent
  - otherwise, an error state is reached
  - error trace is produced to show: \( M_1 \neq M_2 \)
Product Machine - Construction

- Define the product machine $M(X, S, S^0, \delta, \lambda, O)$
  
  - states, $S = S_1 \times S_2$
  - next state function, $\delta(s, x) : (S_1 \times S_2) \times X \rightarrow (S_1 \times S_2)$
  - output function, $\lambda(s, x) : (S_1 \times S_2) \times X \rightarrow \{0,1\}$

\[ O = \begin{cases} 1 & \text{if } O_1 = O_2 \\ 0 & \text{otherwise} \end{cases} \]

\[ \lambda(s, x) = \lambda_1(s_1, x) \oplus \lambda_2(s_2, x) \]

- Error trace (distinguishing sequence) that leads to an error state
  - sequence of inputs which produces 1 at the output of $M$
  - produces a state in $M$ for which $M_1$ and $M_2$ give different outputs
FSM Traversal - Algorithm

- Traverse the product machine $M(X,S,\delta, \lambda, O)$
  - start at an initial state $S_0$
  - iteratively compute symbolic image $Img(S_0, R)$

(set of next states):

$$Img(S_0, R) = \exists_x \exists_s S_0(s) \cdot R(x,s,t)$$

$$R = \prod_i R_i = \prod_i (t_i \equiv \delta_i(s,x))$$

until an error state is reached

- transition relation $R_i$ for each next state variable $t_i$

  can be computed as $t_i = (t \otimes \delta(s,x))$

  (this is an alternative way to compute transition relation, when design is specified at gate level)
Construction of the Product FSM

• For each pair of states, $s_1 \in M_1$, $s_2 \in M_2$
  ➢ create a combined state $s = (s_1, s_2)$ of $M$
  ➢ create transitions out of this state to other states of $M$
  ➢ label the transitions (input/output) accordingly

Output = \{ 1 \text{ OK}, 0 \text{ error} \}
FSM Traversal in Action

Initial states: $s_1=0$, $s_2=0$, $s=(0.0)$

<table>
<thead>
<tr>
<th>State reached</th>
<th>Out($M$)</th>
<th>$x=0$</th>
<th>$x=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$New^0$</td>
<td>(0.0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$New^1$</td>
<td>(1.1)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$New^2$</td>
<td>(0.2)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$New^3$</td>
<td>(1.0)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **STOP** - backtrack to initial state to get error trace: $x=\{1,1,1,0\}$
FSM Traversal in Action

Initial states: $s_1 = 1$, $s_2 = 4$, $s = (1.4)$

<table>
<thead>
<tr>
<th>State reached</th>
<th>$Out(M)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = 0$</td>
<td>1 1</td>
</tr>
<tr>
<td>$x = 1$</td>
<td>1 1</td>
</tr>
</tbody>
</table>

- $New^0 = (1.4)\ 1\ 1$
- $New^1 = (2.5)\ 1\ 1$
- $New^2 = (3.6)\ 1\ 1$

$M = M_1 \times M_2$

- **STOP:** No new reachable state
FSM Traversal in Action

Initial states: $s_1 = 1$, $s_2 = 4$, $s = (1.4)$

<table>
<thead>
<tr>
<th>State reached</th>
<th>$Out(M)$</th>
<th>$x = 0$</th>
<th>$x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{New}^0$</td>
<td>$(1.4)$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\text{New}^1$</td>
<td>$(2.5)$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\text{New}^2$</td>
<td>$(3.6)$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Erroneous states are not reachable
How to Represent States?

- Not practical to represent individual states
- Represent a set of states symbolically
- OBDD encodes boolean functions
  - Code elements S
  - Represent a subset T as boolean function $f_T$
- $f_T = \{0,1\}^n \rightarrow \{0.1\}$
How to Represent States?

- Computation uses symbolic BFS approach to all reachable states by shortest path
- Key step is image computation
  - $\text{Img}(\delta(s,x), C(s))$
- BFS allows to deal multiple states simultaneously
- BDD is used to represent TF
- Let $t_i = \delta_i(s,x)$ $i = 1,2,...,n$
- $C(s)$ is a symbolic state set
How to Represent TF?

- Given a deterministic transition function \((s,x)\) the corresponding transition relation is defined by:
  \[ T(s,x,t) = \prod (t_i = \delta_i(s,x)) \]

- \(T(s,x,t) = 1\) denotes a set of encoded tripples \((s,x,t)\), each representing a transition in the FST of a given FSM.

- Straight forward to compute image

- Need new boolean operation
  - Existential Abstraction
  - \(\exists x_i f = f_{x_i} + f_{x_i'}\)
  - \(f_{x_i}\)-smallest (fewest minterm) function that contains all minterms of \(f\) and independent of \(x_i\)
Thank you