Efficient Simulation Based Verification by Re-ordering

(Previous work in Rambus Inc.)

Chao Yan
chaoyan@cs.ubc.ca
The University of British Columbia

Kevin Jones
drkdjones@gmail.com
Green Plug Inc.
Outline

- Problem and Motivation
- Offline Algorithms
  - ILP based algorithm: optimal solution
  - Greedy algorithm: sub-optimal solution
  - LP based algorithm: super-optimal solution
  - Results
- Online Algorithm
  - Algorithm
  - Result
- Conclusion and Future Work
Simulation Based Verification

- Verification in Industry
  - Simulation and Verification
  - Difficulties of formal methods in AMS verification
  - Specification and testbench
- Verification Platform
Simulation Based Verification

- Verification in Industry
- Verification Platform
  - Specman
  - Transaction generation
  - Output checker
  - Coverage generation

Efficient Simulation Based Verification by Re-ordering — 25th Feb. 2010 — DVCon10 — p.3/16
Problem of (AMS) simulations
- Expensive: several days or even more than a month
- Redundancy: test cases from several engineers, not optimized
- Coverage: low

Goal
- Reduce running time
- Increase coverage
- Efficiency = coverage/time!

Order does matter!
- Run *important* test cases first
- Remove redundant test cases
- Offline and Online
Collect Data

- Specification: \( F = \{f_1, \cdots, f_m\} \)
- Testbench: \( J = \{j_1, \cdots, j_n\} \)
- Running time: \( T = [t_1; t_2; \cdots; t_n] \)
- Coverage vector and coverage matrix: \( M[m, n] = 1(0), \text{if } j_n \text{ (not) checks } f_m \)
- Set coverage: \( C_{\{j_1, \cdots, j_k\}} = \left\langle W \cdot \min(1, \sum_{i=1}^{k} M[i, i]) \right\rangle. \)

Problems

- P1: What is the best order that uses minimum time to achieve given coverage \( C \)?
- P2: what is the best order that achieves maximum coverage within time \( T \)?
Optimization Problem

P1: \( T \), given lower bound of coverage \( C \)

- Let \( x_i \) be the variable that indicates whether test case \( j_i \) should be tested or not
- Let \( h_j \) be the total coverage of function \( f_j \)
- Total coverage should be greater than \( C \)
Optimization Problem

- P1: ↓T, given lower bound of coverage $C$
  - Let $x_i$ be the variable that indicates whether test case $j_i$ should be tested or not
  - Let $h_j$ be the total coverage of function $f_j$
  - Total coverage should be greater than $C$

$$\min \sum_{i=1}^{n} t_i \cdot x_i$$

$$x_i \in [0, 1] \quad (i = 1, \ldots, n)$$
Optimization Problem

- P1: \( \downarrow T \), given lower bound of coverage \( C \)
  - Let \( x_i \) be the variable that indicates whether test case \( j_i \) should be tested or not
  - Let \( h_j \) be the total coverage of function \( f_j \)
  - Total coverage should be greater than \( C \)

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} t_i \cdot x_i \\
x_i & \in [0, 1] \quad (i = 1, \cdots, n) \\
h_j & = \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \cdots, m) \\
h_j & \leq 1 \quad (j = 1, \cdots, m) \\
h_j & \geq 0 \quad (j = 1, \cdots, m)
\end{align*}
\]
Optimization Problem

- **P1: \( \downarrow T \), given lower bound of coverage \( C \)**
  - Let \( x_i \) be the variable that indicates whether test case \( j_i \) should be tested or not.
  - Let \( h_j \) be the total coverage of function \( f_j \).
  - Total coverage should be greater than \( C \).

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} t_i \cdot x_i \\
x_i & \in [0, 1] \quad (i = 1, \cdots, n) \\
h_j & = \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \cdots, m) \\
h_j & \leq 1 \quad (j = 1, \cdots, m) \\
h_j & \geq 0 \quad (j = 1, \cdots, m) \\
C & \leq \sum_{j=1}^{m} h_j \cdot w_j
\end{align*}
\]
Optimization Problem

- P1 ⇒ Integer Linear Program (ILP)
  - Let $x_i$ be the variable that indicates whether test case $j_i$ should be tested or not
  - Let $h_j$ be the total coverage of function $f_j$
  - Total coverage should be greater than $C$

\[
\begin{align*}
\text{min} & \quad \sum_{i=1}^{n} t_i \cdot x_i \\
 x_i & \in [0, 1] \quad (i = 1, \ldots, n) \\
 h_j & = \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \ldots, m) \\
 h_j & \leq 1 \quad (j = 1, \ldots, m) \\
 h_j & \geq 0 \quad (j = 1, \ldots, m) \\
 C & \leq \sum_{j=1}^{m} h_j \cdot w_j 
\end{align*}
\]
Optimization Problem

- **P1** ⇒ Integer Linear Program (ILP)

- **P2** ⇒ Integer Linear Program

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{m} h_j \cdot w_j \\
x_i & \in [0, 1] \quad (i = 1, \ldots, n) \\
h_j & = \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \ldots, m) \\
h_j & \leq 1 \quad (j = 1, \ldots, m) \\
h_j & \geq 0 \quad (j = 1, \ldots, m) \\
T & \geq \sum_{i=1}^{n} x_i \cdot t_i
\end{align*}
\]
Approximation Algorithm

- ILP is NP-hard
- Greedy Algorithm
  - Select the most important test case at each step
  - Importance of test case $j_m$ is
    \[ r = \frac{C_{\{j_1, \ldots, j_k, j_m\}} - C_{\{j_1, \ldots, j_k\}}}{t_m} \]

- Simple but Efficient
- How Big is the Gap?
Super-optimal solution

- Do not need to solve the ILP problems to find the gap
- Find the super-optimal solution by relaxing its constraints
- The relaxations are LPs which can be solved efficiently
Super-optimal solution

- Do not need to solve the ILP problems to find the gap
- Find the super-optimal solution by relaxing its constraints

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} t_i \cdot x_i \\
\text{subject to} & \quad h_j \leq \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \ldots, m) \\
\text{subject to} & \quad C \leq \sum_{j=1}^{m} h_j \cdot w_j \\
\text{subject to} & \quad 0 \leq h_j, x_i \quad (j = 1, \ldots, m; i = 1, \ldots, n) \\
\text{subject to} & \quad 1 \geq h_j, x_i \quad (j = 1, \ldots, m; i = 1, \ldots, n)
\end{align*}
\]

- The relaxations are LPs which can be solved efficiently
Super-optimal solution

- Do not need to solve the ILP problems to find the gap
- Find the super-optimal solution by relaxing its constraints

\[
\begin{align*}
\max & \quad \sum_{j=1}^{m} h_j \cdot w_j \\
\text{s.t.} & \quad h_j \leq \sum_{i=1}^{n} x_i \cdot M[j, i] \quad (j = 1, \ldots, m) \\
& \quad T \geq \sum_{i=1}^{n} x_i \cdot t_i \\
& \quad 0 \leq h_j, x_i \quad (j = 1, \ldots, m; i = 1, \ldots, n) \\
& \quad 1 \geq h_j, x_i \quad (j = 1, \ldots, m; i = 1, \ldots, n)
\end{align*}
\]

- The relaxations are LPs which can be solved efficiently
Results

- Toshiba XIO device
- Flow
  - Greedy algorithm
  - LP based algorithm
  - (if gap is large) ILP based algorithm

- Results
Results

- Toshiba XIO device
- Flow

Results
- The result of Greedy algorithm is much better than the default one
- Save 85% time without lost of coverage

Efficient Simulation Based Verification by Re-ordering — 25th Feb. 2010 — DVCon10 — p.9/16
## Results

- Toshiba XIO device
- Flow

### Results

- The result of Greedy algorithm is much better than the default one
- 53 of 222 test cases

<table>
<thead>
<tr>
<th>Order</th>
<th>Feature</th>
<th>Config</th>
<th>Time</th>
<th>Cov</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>testRegRW</td>
<td>xiox8_gate_yve_2</td>
<td>11</td>
<td>0.07</td>
<td>0.18</td>
</tr>
<tr>
<td>2</td>
<td>ptcal_reg_act</td>
<td>xiox16_gate_yve_2</td>
<td>80</td>
<td>0.25</td>
<td>0.51</td>
</tr>
<tr>
<td>3</td>
<td>hy_tcal</td>
<td>xiox16_gate_yve_2</td>
<td>146</td>
<td>0.32</td>
<td>0.64</td>
</tr>
<tr>
<td>4</td>
<td>testRegInit</td>
<td>xiox16_gate_yve_4</td>
<td>87</td>
<td>0.25</td>
<td>0.66</td>
</tr>
<tr>
<td>5</td>
<td>pllcfg_chk</td>
<td>xiox16_gate_clkoff_yve_2</td>
<td>95</td>
<td>0.23</td>
<td>0.68</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>52</td>
<td>ptcal_avgen_32_ov_31</td>
<td>xiox16_gate_yve_2</td>
<td>13481</td>
<td>0.30</td>
<td>0.99</td>
</tr>
<tr>
<td>53</td>
<td>ptcal_avgen_4_ov_3</td>
<td>xiox16_gate_yve_2</td>
<td>13892</td>
<td>0.30</td>
<td>1.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Results

- Toshiba XIO device
- Flow
- Results

The result of Greedy algorithm is close to the super-optimal one

![Graph showing coverage over time for default order, greedy algorithm, super-optimal (time), and super-optimal (cov).]
Results

- Toshiba XIO device
- Flow

Results

The result of Greedy algorithm is almost the same with the result from vManager’s rank function
Online Algorithm

- Greedy algorithm? has to run all test cases to collect data.
- What order to use for the first time?
- Combinations of configurations and features

<table>
<thead>
<tr>
<th></th>
<th>sa_ecc</th>
<th>sa</th>
<th>sa_x8</th>
<th>sa_ecc_x8</th>
<th>drm_rtl_x8</th>
<th>drm_rtl</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortcut_test</td>
<td>1</td>
<td>10</td>
<td>19</td>
<td>28</td>
<td>37</td>
<td>46</td>
</tr>
<tr>
<td>direct0</td>
<td>2</td>
<td>11</td>
<td>20</td>
<td>29</td>
<td>38</td>
<td>47</td>
</tr>
<tr>
<td>direct1</td>
<td>3</td>
<td>12</td>
<td>21</td>
<td>30</td>
<td>39</td>
<td>48</td>
</tr>
<tr>
<td>dr_wr0</td>
<td>4</td>
<td>13</td>
<td>22</td>
<td>31</td>
<td>40</td>
<td>49</td>
</tr>
<tr>
<td>dr_wr1</td>
<td>5</td>
<td>14</td>
<td>23</td>
<td>32</td>
<td>41</td>
<td>50</td>
</tr>
<tr>
<td>testReg02</td>
<td>6</td>
<td>15</td>
<td>24</td>
<td>33</td>
<td>42</td>
<td>51</td>
</tr>
<tr>
<td>testReg04</td>
<td>7</td>
<td>16</td>
<td>25</td>
<td>34</td>
<td>43</td>
<td>52</td>
</tr>
<tr>
<td>testRegRW</td>
<td>8</td>
<td>17</td>
<td>26</td>
<td>35</td>
<td>44</td>
<td>53</td>
</tr>
<tr>
<td>testRegRw2</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>36</td>
<td>45</td>
<td>54</td>
</tr>
</tbody>
</table>
Different Running Orders

- Current Order
- Greedy Algorithm
- Row First? Column First?
- Random?
Different Running Orders

- Current Order
- Greedy Algorithm
- Row First? Column First?
- Random?

Efficient Simulation Based Verification by Re-ordering — 25th Feb. 2010 — DVCon10 – p.11/16
Different Running Orders

- Current Order
- Greedy Algorithm
- Row First? Column First?
- Random?

Efficient Simulation Based Verification by Re-ordering — 25th Feb. 2010 — DVCon10 – p.11/16
Different Running Orders

- Current Order
- Greedy Algorithm
- Row First? Column First?
- Random?
Similarity of Coverage Vector

- Similarity of coverage vectors

\[ \text{sim}(v_i, v_j) = \frac{\langle v_i \cdot v_j \rangle}{|v_i| \cdot |v_j|} \]
Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features

\[ csim(c_i, c_j) = \prod_{k=1}^{nt} sim([k, i], [k, j])^{1/nt} \]

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>1.0000</td>
<td>0.9786</td>
<td>0.9782</td>
<td>0.9826</td>
<td>0.9789</td>
<td>0.9777</td>
</tr>
<tr>
<td>c2</td>
<td>0.9786</td>
<td>1.0000</td>
<td>0.9832</td>
<td>0.9793</td>
<td>0.9802</td>
<td>0.9805</td>
</tr>
<tr>
<td>c3</td>
<td>0.9782</td>
<td>0.9832</td>
<td>1.0000</td>
<td>0.9792</td>
<td>0.9807</td>
<td>0.9831</td>
</tr>
<tr>
<td>c4</td>
<td>0.9826</td>
<td>0.9793</td>
<td>0.9792</td>
<td>1.0000</td>
<td>0.9776</td>
<td>0.9797</td>
</tr>
<tr>
<td>c5</td>
<td>0.9789</td>
<td>0.9802</td>
<td>0.9807</td>
<td>0.9776</td>
<td>1.0000</td>
<td>0.9814</td>
</tr>
<tr>
<td>c6</td>
<td>0.9777</td>
<td>0.9805</td>
<td>0.9831</td>
<td>0.9797</td>
<td>0.9814</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features

\[ f_{\text{sim}}(c_i, c_j) = \prod_{k=1}^{nc} \sim([i, k], [j, k])^{1/nc} \]

<table>
<thead>
<tr>
<th></th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
<th>f6</th>
<th>f7</th>
<th>f8</th>
<th>f9</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>1.00</td>
<td>0.78</td>
<td>0.78</td>
<td>0.99</td>
<td>0.99</td>
<td>0.86</td>
<td>0.57</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>f2</td>
<td>0.78</td>
<td>1.00</td>
<td>0.98</td>
<td>0.78</td>
<td>0.78</td>
<td>0.73</td>
<td>0.48</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>f3</td>
<td>0.78</td>
<td>0.98</td>
<td>1.00</td>
<td>0.78</td>
<td>0.78</td>
<td>0.73</td>
<td>0.48</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>f4</td>
<td>0.99</td>
<td>0.78</td>
<td>0.78</td>
<td>1.00</td>
<td>0.99</td>
<td>0.86</td>
<td>0.57</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>f5</td>
<td>0.99</td>
<td>0.78</td>
<td>0.78</td>
<td>0.99</td>
<td>1.00</td>
<td>0.86</td>
<td>0.57</td>
<td>0.63</td>
<td>0.63</td>
</tr>
<tr>
<td>f6</td>
<td>0.86</td>
<td>0.73</td>
<td>0.73</td>
<td>0.86</td>
<td>0.86</td>
<td>1.00</td>
<td>0.57</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>f7</td>
<td>0.57</td>
<td>0.48</td>
<td>0.48</td>
<td>0.57</td>
<td>0.57</td>
<td>0.57</td>
<td>1.00</td>
<td>0.88</td>
<td>0.89</td>
</tr>
<tr>
<td>f8</td>
<td>0.63</td>
<td>0.53</td>
<td>0.53</td>
<td>0.63</td>
<td>0.63</td>
<td>0.62</td>
<td>0.88</td>
<td>1.00</td>
<td>0.97</td>
</tr>
<tr>
<td>f9</td>
<td>0.63</td>
<td>0.53</td>
<td>0.53</td>
<td>0.63</td>
<td>0.63</td>
<td>0.62</td>
<td>0.89</td>
<td>0.97</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features
- Coverage similarity $\approx$ configuration similarity $\times$ feature similarity
  
  \[ \text{sim}(21, 43) = 0.4913 \approx \text{csim}(3, 5) \cdot \text{fsim}(3, 7) = 0.4793 \]

  average error is 0.08% and maximum error is 4.25%

<table>
<thead>
<tr>
<th></th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f3</td>
<td></td>
<td></td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f7</td>
<td></td>
<td></td>
<td></td>
<td>43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Online Algorithm

- Select the one with maximum \( r = \frac{\Delta C'(v)}{t} \) in greedy algorithm.
- The next test case without its coverage vector and running time?
- How to approximate \( t \)?
- How to approximate \( \Delta C'(v) \)?
Online Algorithm

- Select the one with maximum \( r = \frac{\Delta C(v)}{t} \) in greedy algorithm.
- The next test case without its coverage vector and running time?
- How to approximate \( t \)?
  - Running time similarity
  - The length of coverage vector \( |v| \) is proportion to running time \( t \).
  - Greedy algorithm works well with \( \frac{\Delta C(v)}{|v|} \).
- How to approximate \( \Delta C'(v) \)?
Online Algorithm

- Select the one with maximum \( r = \frac{\Delta C(v)}{t} \) in greedy algorithm.
- The next test case without its coverage vector and running time?
- How to approximate \( t \)?
- How to approximate \( \Delta C(v) \)?
  - Generate random coverage vector \( v \) based on the estimated similarity
  - The result is similar with random algorithm
  - Too much freedom, little constraints
  - \( \frac{\Delta C(v)}{|v|} \) does not converge
Online Algorithm

- Select the one with maximum $r = \frac{\Delta C(v)}{t}$ in greedy algorithm.
- The next test case without its coverage vector and running time?
- How to approximate $t$?
- How to approximate $\Delta C(v)$?
  - Estimate the similarity between new coverage vector and the one in history
  - The one with smallest similarity probably increases coverage most
  - Weight coverage vector visited by their contribution to coverage
  - Transform the problem

$$\max \frac{\Delta C(v)}{|v|} \quad \Leftrightarrow \quad \min \sum_{i=1}^{k} \Delta C(v_i) \cdot \text{sim}(v_i, v)$$
Online Algorithm

- Select the one with maximum $r = \frac{\Delta C(v)}{t}$ in greedy algorithm.
- The next test case without its coverage vector and running time?
- How to approximate $t$?
- How to approximate $\Delta C(v)$?
- The Algorithm
  - Estimate coverage similarity
  - Choose the test case with minimum similarity
  - Update configuration and feature similarity
  - Repeat until all test cases are tested
The online algorithm is better than the random algorithm.

Similarity estimation error is large at the beginning.

Estimation error of coverage vector is large at the end.
Conclusion and Future Work

- Conclusion
  - Improved performance by reordering test cases
  - Developed online algorithm to run efficient test cases first
  - Applied to Rambus XIO device

- Future Work
  - Apply to more simulation problems
  - More general online algorithm
  - Apply the similar idea to smaller granularity
  - Generate new test cases to increase coverage automatically
Acknowledgment

- Mark Greenstreet
- Kathryn Mossawir
- Tom Sheffler
- John Hong
- Victor Konrad

Questions?

Thank You!