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

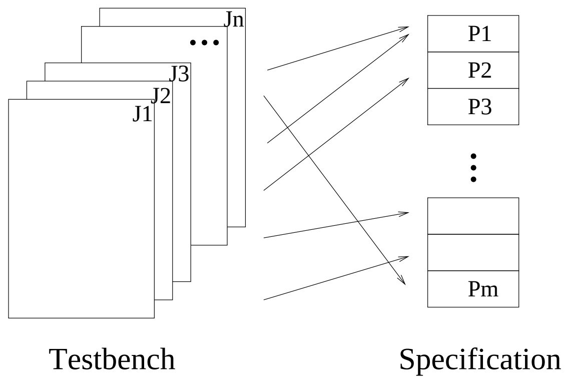
Green Plug Inc.

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

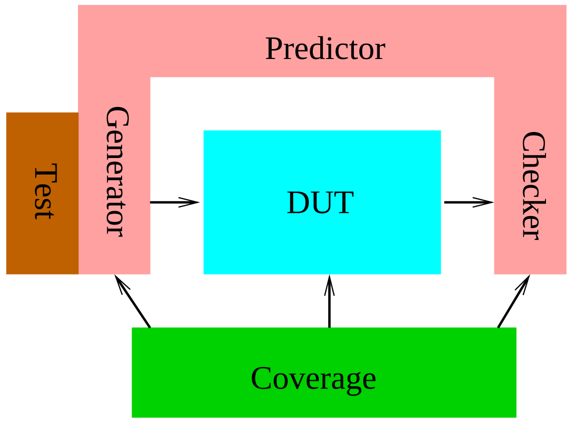
Simulation Based Verification

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



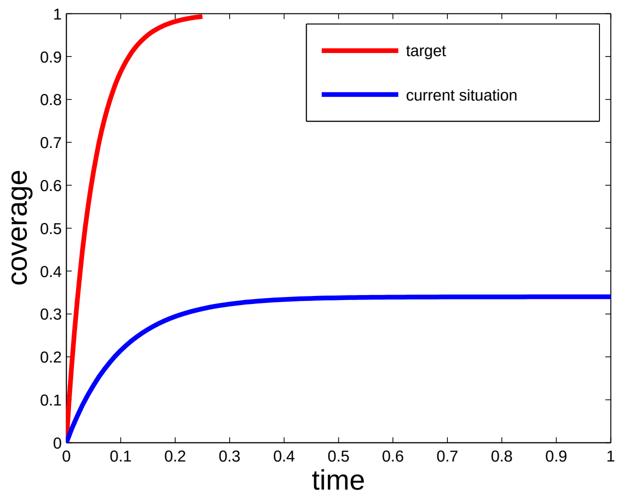
Simulation Based Verification

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



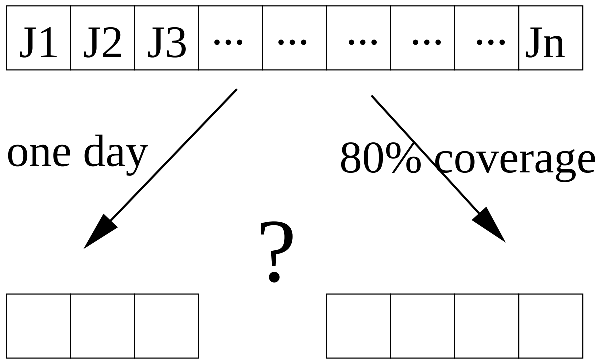
Reordering

- 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



Offline Algorithms

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

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

$$
\begin{array}{ccl}
 & min & \sum_{i=1}^{n} t_i \cdot x_i \\
x_i & \in & [0,1] \quad (i = 1, \cdots, n)
\end{array}
$$

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

$$\begin{array}{rcl} & min & \sum_{i=1}^{n} t_i \cdot x_i \\ x_i & \in & [0,1] \quad (i = 1, \cdots, n) \\ h_j & = & \sum_{i=1}^{n} x_i \cdot M[j,i] \quad (j = 1, \cdots, m) \\ h_j & \leq & 1 \quad (j = 1, \cdots, m) \\ h_j & \geq & 0 \quad (j = 1, \cdots, m) \end{array}$$

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

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

Optimization Problem

- P1 $\Rightarrow$ 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

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

Optimization Problem

- P1 $\Rightarrow$ Integer Linear Program (ILP)
- P2 $\Rightarrow$ Integer Linear Program

$$
\begin{array}{rcl}
 & max & \sum_{j=1}^{m} h_j \cdot w_j \\
x_i & \in & [0,1] \quad (i = 1, \cdots, n) \\
h_j & = & \sum_{i=1}^{n} x_i \cdot M[j,i] \quad (j = 1, \cdots, m) \\
h_j & \leq & 1 \quad (j = 1, \cdots, m) \\
h_j & \geq & 0 \quad (j = 1, \cdots, m) \\
T & \geq & \sum_{i=1}^{n} x_i \cdot t_i
\end{array}
$$

Approximation Algorithm

- ILP is NP-hard
- Greedy Algorithm
  - Select the most *important* test case at each step
  - Importance of test case j_m is

$$r = \frac{C_{\{j_1,\cdots,j_k,j_m\}} - C_{\{j_1,\cdots,j_k\}}}{t_m}$$

- 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

$$
\begin{array}{rcl}
min & & \sum_{i=1}^{n} t_i \cdot x_i \\
h_j & \leq & \sum_{i=1}^{n} x_i \cdot M[j,i] \quad (j = 1, \cdots, m) \\
C & \leq & \sum_{j=1}^{m} h_j \cdot w_j \\
0 & \leq & h_j, x_i \quad (j = 1, \cdots, m; i = 1, \cdots, n) \\
1 & \geq & h_j, x_i \quad (j = 1, \cdots, m; i = 1, \cdots, n)
\end{array}
$$

- 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

$$\begin{array}{rcl} max & & \sum_{j=1}^{m} h_j \cdot w_j \\ h_j & \leq & \sum_{i=1}^{n} x_i \cdot M[j,i] \quad (j = 1, \cdots, m) \\ T & \geq & \sum_{i=1}^{n} x_i \cdot t_i \\ 0 & \leq & h_j, x_i \quad (j = 1, \cdots, m; i = 1, \cdots, n) \\ 1 & \geq & h_j, x_i \quad (j = 1, \cdots, m; i = 1, \cdots, n)) \end{array}$$

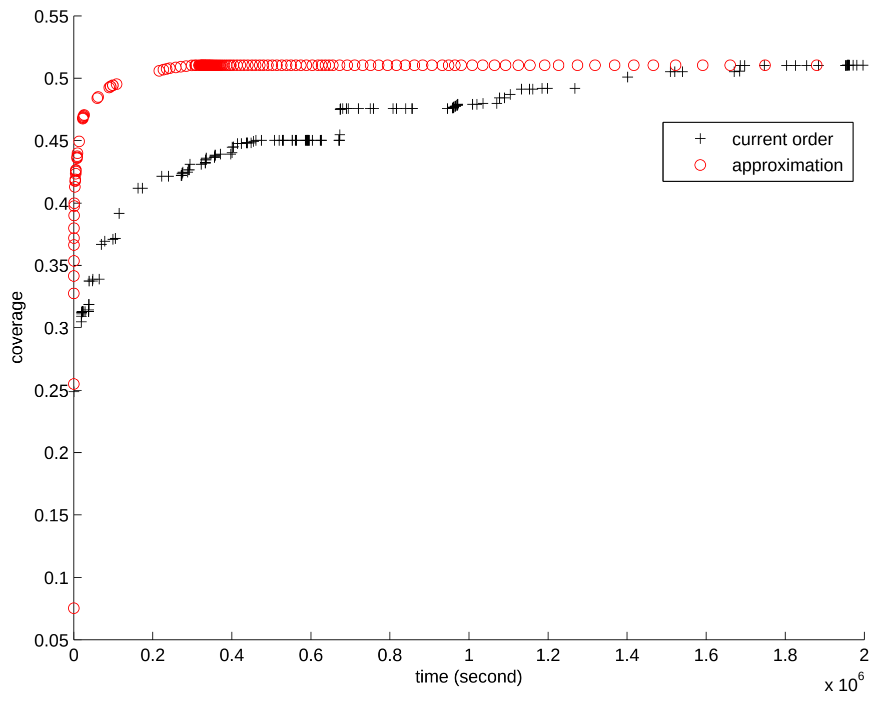
- The relaxations are LPs which can be solved efficiently

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



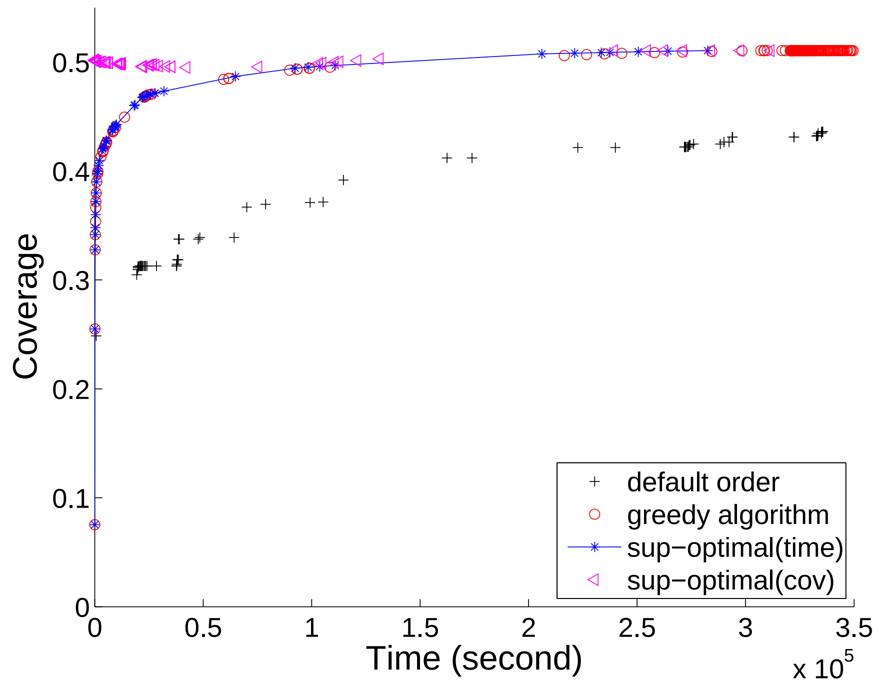
Results

- Toshiba XIO device
- Flow
- Results
  - The result of Greedy algorithm is much better than the default one
  - 53 of 222 test cases

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

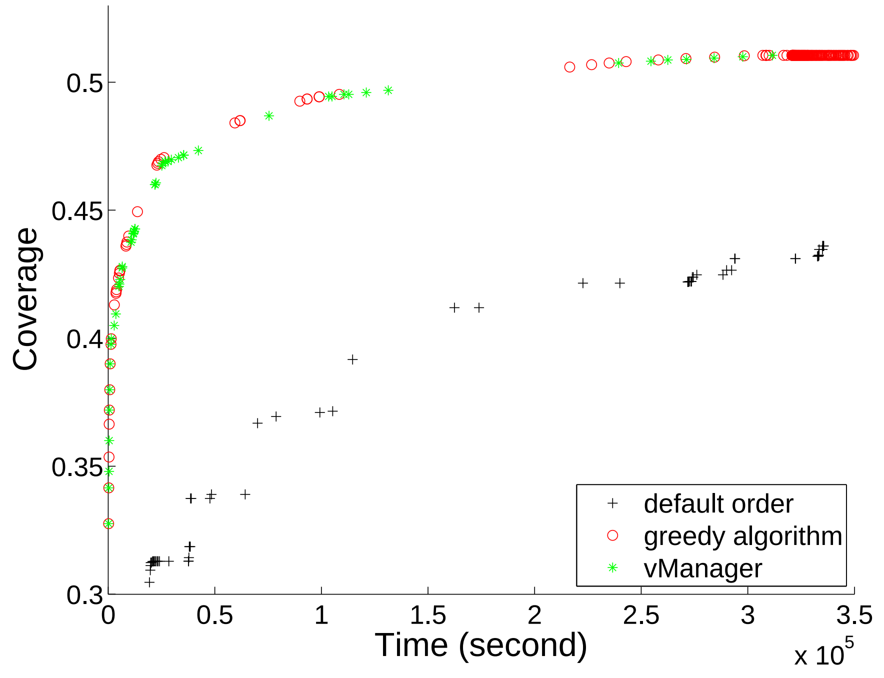
Results

- Toshiba XIO device
- Flow
- Results
  - The result of Greedy algorithm is close to the super-optimal one



Results

- Toshiba XIO device
- Flow
- Results
  - The result of Greedy algorithm is almost the same with the result from *vManager's rank* function



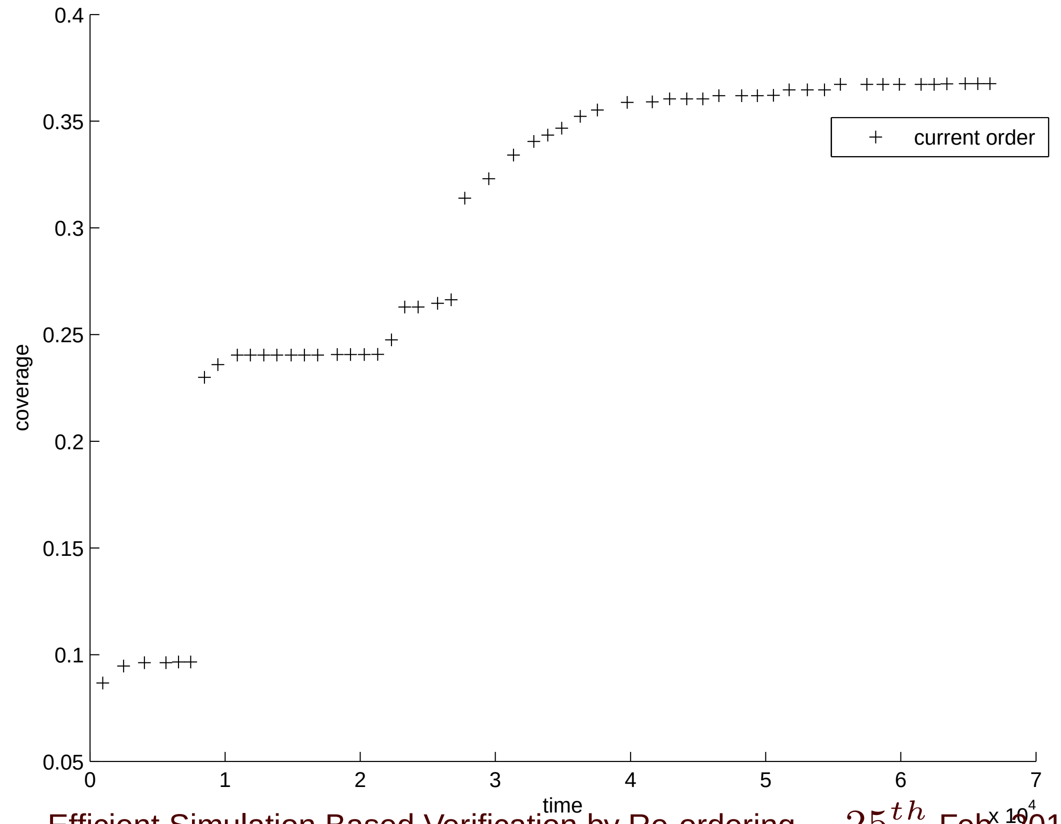
Online Algorithm

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

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

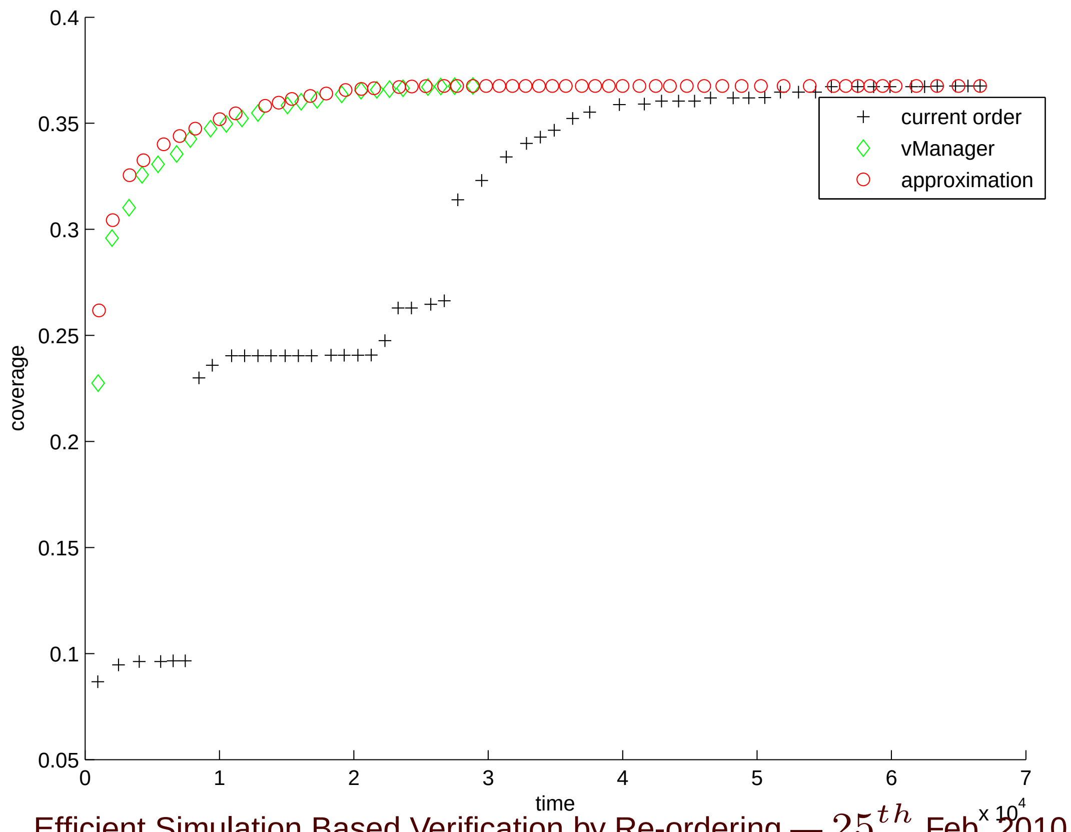
Different Running Orders

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



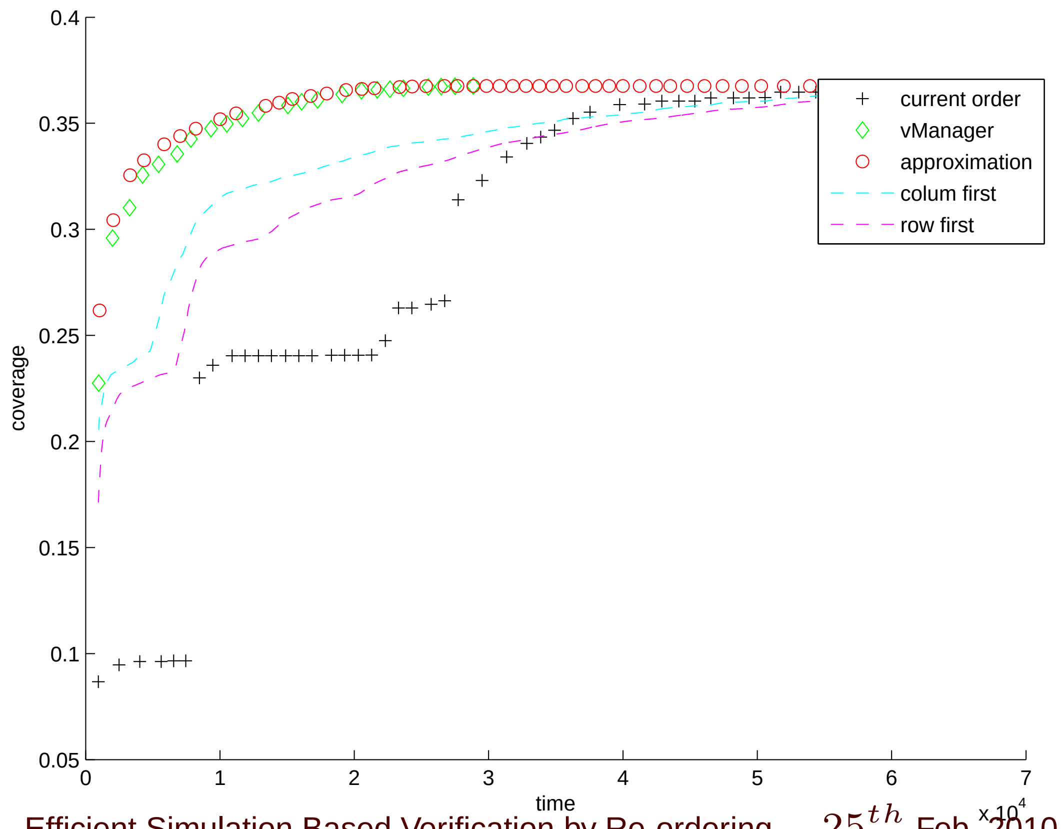
Different Running Orders

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



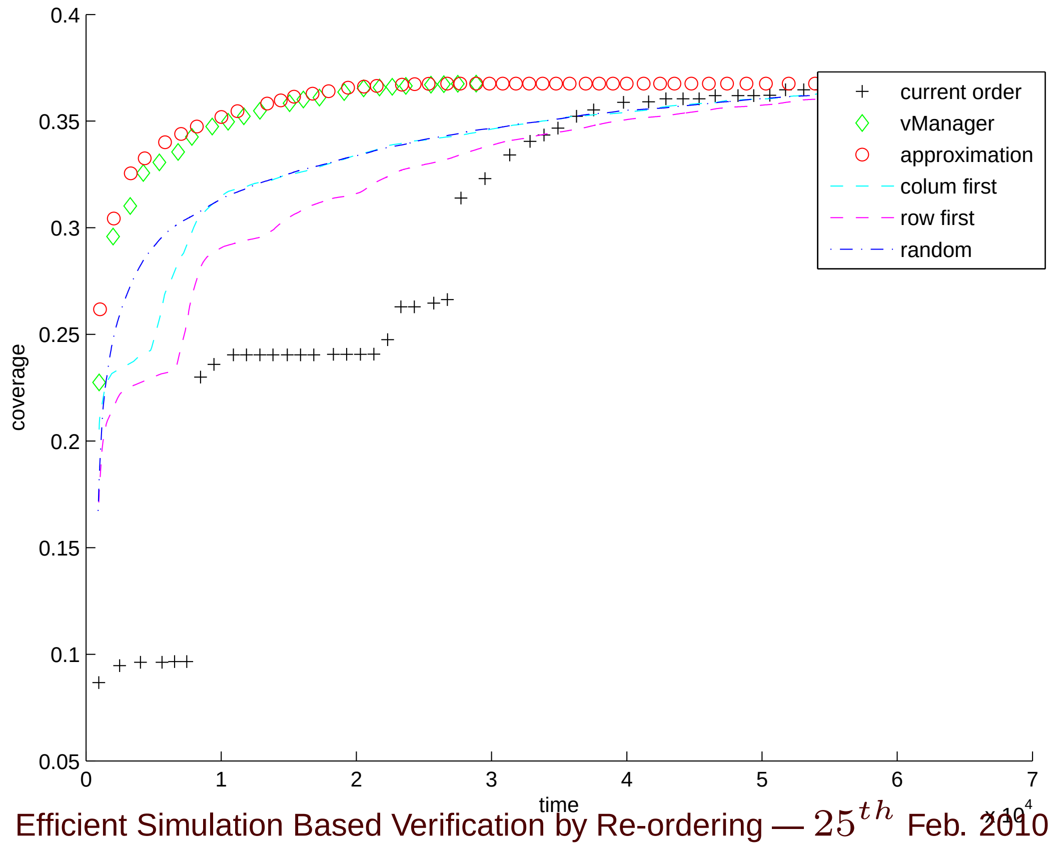
Different Running Orders

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



Different Running Orders

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



Similarity of Coverage Vector

- Similarity of coverage vectors

$$sim(v_i, v_j) \quad = \quad \frac{< v_i \cdot v_j >}{|v_i| \cdot |v_j|}$$

Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features

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

	c1	c2	c3	c4	c5	c6
c1	1.0000	0.9786	0.9782	0.9826	0.9789	0.9777
c2	0.9786	1.0000	0.9832	0.9793	0.9802	0.9805
c3	0.9782	0.9832	1.0000	0.9792	0.9807	0.9831
c4	0.9826	0.9793	0.9792	1.0000	0.9776	0.9797
c5	0.9789	0.9802	0.9807	0.9776	1.0000	0.9814
c6	0.9777	0.9805	0.9831	0.9797	0.9814	1.0000

Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features

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

	f1	f2	f3	f4	f5	f6	f7	f8	f9
f1	1.00	0.78	0.78	0.99	0.99	0.86	0.57	0.63	0.63
f2	0.78	1.00	0.98	0.78	0.78	0.73	0.48	0.53	0.53
f3	0.78	0.98	1.00	0.78	0.78	0.73	0.48	0.53	0.53
f4	0.99	0.78	0.78	1.00	0.99	0.86	0.57	0.63	0.63
f5	0.99	0.78	0.78	0.99	1.00	0.86	0.57	0.63	0.63
f6	0.86	0.73	0.73	0.86	0.86	1.00	0.57	0.62	0.62
f7	0.57	0.48	0.48	0.57	0.57	0.57	1.00	0.88	0.89
f8	0.63	0.53	0.53	0.63	0.63	0.62	0.88	1.00	0.97
f9	0.63	0.53	0.53	0.63	0.63	0.62	0.89	0.97	1.00

Similarity of Coverage Vector

- Similarity of coverage vectors
- Similarity of configurations or features
- Coverage similarity $\approx$ configuration similarity $\times$ feature similarity
    - $sim(21, 43) = 0.4913 \approx csim(3, 5) \cdot fsim(3, 7) = 0.4793$
    - average error is 0.08% and maximum error is 4.25%

	c1	c2	c3	c4	c5	c6
f1						
f2						
f3			21			
f4						
f5						
f6						
f7					43	

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

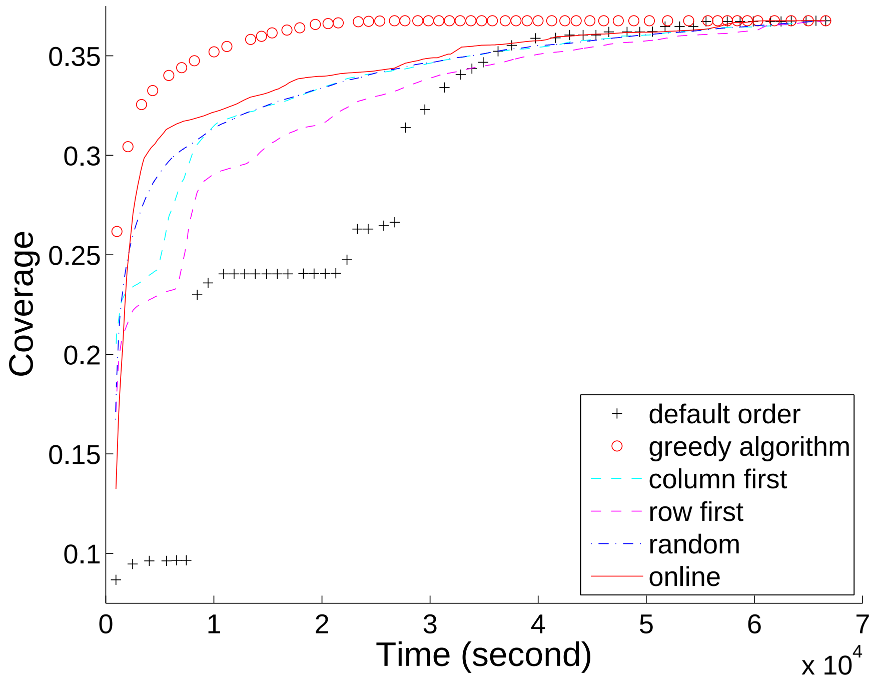
$$max \frac{\Delta C(v)}{|v|} \quad \Leftrightarrow \quad min \sum_{i=1}^{k} \Delta C(v_i) \cdot sim(v_i, v)$$

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

Result

- BE-XIO
- Result
  - 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!