Pruning-based Trace Signal Selection Algorithm

Kang Zhao, Jinian Bian

EDA Lab, Dept. Computer Science & Technology
Tsinghua University, Beijing 100084, China
Jan 28, 2011
Contents

• Introduction
• Algorithm
• Experiments
• Conclusion
Contents

• Introduction
• Algorithm
• Experiments
• Conclusion
Background

• Silicon debug
  – Pre-silicon validation: functional and timing errors
  – Post-silicon validation: functional and electronic errors

• Motivation
  – Identify the bugs effectively
  – Know each state for each signal in the circuits
  – Enhance the visibility of the circuits
General Method

• Monitor and trace some internal signals
  – The values of those signals are stored in the trace buffer
  – Estimate the values and states of other signals based on an analysis on the trace signals

• Advantage & Limitation
  – Advantage: need not large buffer to store many signals
  – Advantage: low cost and high efficiency
  – Limitation: the number of trace signals are limited due to the trace buffer

• Focus: how to select a limited set of trace signals
Problem

- **Problem**
  - Given the circuit with \( n \) FFs, find a subset of trace signals not exceeding \( k \), so that the restoration ratio \( r \) is maximum.

- **Focus**
  - How to select limited trace signals, so we can view more states in the circuit.
Related works

- **Two restorations**
  - Forward: F0/F1
  - Backward: B0/B1
  - Restoration ratio: \( \frac{N_{\text{trace}} + N_{\text{restore}}}{N_{\text{restore}}} \)

- **Method**
  - Sum up all restorations for each FF
  - Select the biggest one as the trace signal
  - The restoration ratio is used to measure the quality of the final results

Related works

• Accurate restorations
  – $RV$ is used to replace the $F$ and $B$
  – $RV = P*(F+B-F*B)$
  – $P$ means the probability in the functional mode

• Method
  – Sum up all restorations
  – Select the biggest one as the trace

• Advantage
  – The conditional probability can consider the two directions together
  – $P$ can reduce the additional effect of the initial values

Limitations

• Greedy trace signal selection
  – First select the signal $a$ which can get the biggest restoration ratio
  – Then, fix the signal $a$; select another signal $b$ which can get another biggest restoration ratio
  – ...

• Limitation
  – The signal $a$ may be not the best selection although it is the best choice for the first iteration
  – The previous choice will affect the following iterations
  – The greedy strategy can only get a better result
Contents

• Introduction
• Algorithm
• Experiments
• Conclusion
Strategy

- **Three steps**
  - Compute the restoration range for each signal
  - Generate the constraints based on the candidate trace signals
  - Compare the effects based on different combinations, find the better one

- **Restoration range**
  - \( R(x) = \{ y \mid y \text{ can be restored by } x \} \)
Candidate generation – restoration range

- **Theorem:** Let $R(a)$ and $R(b)$ are two different ranges. If there is no conflict, $R(a, b) \supseteq R(a) \cup R(b)$
  - E.g. if $a=1$ and $c=0$ for the AND gate, the input $b$ will be restored
  - For each gate, if the values of other signals satisfy the constraints in the figure below, the value of last signal will be restored

<table>
<thead>
<tr>
<th>AND</th>
<th>OR</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\begin{array}{c} a = 1 \ b = 0 \end{array}) \quad c = 0 \quad \text{trace: } a=1, \ c=0</td>
<td>(\begin{array}{c} a = 0 \ b = 1 \end{array}) \quad c = 1 \quad \text{trace: } a=0, \ c=1</td>
<td>(\begin{array}{c} a = X, \ b = 0 \end{array}) \quad c = X \quad \text{trace: } a=X, \ c=X</td>
</tr>
<tr>
<td>(\begin{array}{c} a = 1 \ b = 1 \end{array}) \quad c = 1 \quad \text{trace: } a=1, \ b=1</td>
<td>(\begin{array}{c} a = 0 \ b = 0 \end{array}) \quad c = 0 \quad \text{trace: } a=0, \ b=0</td>
<td>(\begin{array}{c} a = X, \ b = X \end{array}) \quad c \quad \text{trace: } a=X, \ b=X</td>
</tr>
</tbody>
</table>

Input=1  
Output=0  
Input=0  
Output=1  
Input=X  
Output=X
Candidate generation – range unit

- **Add function**: is used to compute the restoration range, especially for the unit of ranges
  - `Range(S0+S1)=S0+Add(S0, S1, D)`
  - D denotes the number of invisible ports
  - Once a port is restored, D=D-1 and this process will stop until D=1

```
Add(Set S0, Set S1, Degree & D)
1. A ← ∅; Queue ← S1 − S0;
2. while(Queue is not empty){
3.     x ← the top element of Queue;
4.     Remove the top element of Queue;
5.     for(each gate i connecting with the node x){
6.         if(i is not dead and x is potential for i){
7.             D[i] ← D[i] − 1;
8.         if(D[i] = 1){
9.             A ← A ∪ R(y); // y is the last port of i
10.            Add R(y) to the end of Queue;
11.            Sign the gate i to be dead;
12.        } //end if
13.     } //end if
14. } //end for
15. } //end while
16. return A ∪ S1 − S0;
```
Candidate generation – filter

• **Theorem:** let $a$ and $b$ are two different signals and there is no NOT gate between them. Then, we can get:
  
  $b \in R(a) \rightarrow a \not\in R(b)$
  
  $b \in R(a) \rightarrow R(b) \subset R(a)$

• **Filter**
  
  – If $b \in R(a)$, $R(a)$ must be larger or equal to $R(b)$, and $b$ can be replaced by $a$
  
  – If there is NOT gate between $a$ and $b$, one signal can be removed

<table>
<thead>
<tr>
<th>Filter()</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $C \leftarrow$ all signals in the circuit;</td>
</tr>
<tr>
<td>2. for (each two signals $i, j$ in the circuit)</td>
</tr>
<tr>
<td>3. if ($i, j$ are connected with a NOT gate)</td>
</tr>
<tr>
<td>4. $C \leftarrow C - {R(j)}$;</td>
</tr>
<tr>
<td>5. else if ($j \in R(i)$)</td>
</tr>
<tr>
<td>6. $C \leftarrow C - {R(j)}$;</td>
</tr>
<tr>
<td>7. end if</td>
</tr>
<tr>
<td>8. end for</td>
</tr>
<tr>
<td>9. return $C$;</td>
</tr>
</tbody>
</table>
Constraints

- Conjunctive normal form (CNF)
  - Each Boolean circuit can be repressed by CNF
  - “z=x AND y” can be repressed by: \((\bar{z} + x)(\bar{z} + y)(z + \bar{x} + \bar{y})\)
- CNF \(\rightarrow\) constraints
  - The constraints are generated via negative operation on each item of CNF
  - E.g. \(<z=1, x=0>, <z=1, y=0>, \) and \(<z=0, x=1, y=1>\)
Selection Algorithm

Idea: enumerate the combination with the lexicographic order. If violate the constraint, the subtree will be omitted

- (1) initialization, line 1-7
- (2) different combinations are enumerated with the lexicographic order, line 8-12
- (3) once the point exceeds the scope, the exploration stop and pop the stacks, line 13-22
- (4) otherwise, next signal will be put into the stacks, line 23-29

Selection(Set $C_{\text{trace}}$, Size $n$, Constraints)
1. $V \leftarrow \text{Schedule}(C_{\text{trace}}, \text{Constraints})$
2. Generate a new array $A$, and its length is $n$;
3. $A.\text{push}\_\text{back}(V, \text{begin})$; $p \leftarrow 1$; // $p$: a pointer to $V$
4. Generate two empty arrays $X$ and $Y$
5. for (each gate $i$ in the circuit)
6. $D[i] \leftarrow $ the total port number of $i$; // max degree
7. for (each $i \in A$) $R.\text{push}(A[i], X, Y, D)$;
8. while ($A$ is not empty And $V$ is not full){
9. if ($A$ is full and its range is larger than ever){
10. Record the combination in result;
11. }
12. if ($A$ is full){$A.\text{pop}\_\text{back}(); R.\text{pop}(X, Y, D);$}
13. else if ($V$ is full){
14. $A.\text{pop}\_\text{back}();$
15. if (no violation during previous iteration){
16. $A.\text{pop}\_\text{back}(); R.\text{pop}(X, Y, D);$}
17. }
18. if ($A$ is not empty){
19. $p \leftarrow $ the next position in $V$ for $A.\text{end}$;
20. $A.\text{pop}\_\text{back}(); R.\text{pop}(X, Y, D);$}
21. }
22. }
23. else{
24. $A.\text{push}\_\text{back}(V[p])$
25. if ($A$ violates the constraints){
26. $A.\text{pop}\_\text{back}(); ++p$; continue;
27. }
28. }else{$R.\text{push}(V[p], X, Y, D); ++p;}$
29. }
30. }
31. return the better result;

Lexicographic order

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a b c</td>
<td>1</td>
<td>1 1 1</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>a b d</td>
<td>1</td>
<td>1 1 0</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>2</td>
<td>a c d</td>
<td>1</td>
<td>0 1 1</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>3</td>
<td>b c d</td>
<td>0</td>
<td>0 1 1</td>
<td>1 0 1 0</td>
</tr>
</tbody>
</table>
Two features

Pruning
- Each level in the binary tree represents the current signal is selected or not
- Since the enumeration uses the lexicographic order, once the combination violates the constraints, there will be no necessary to explore its subtree

Stack structure
- Since the neighboring candidates have the similar segment, the intermediate results can be reused. We used the stack structure to settle this issue
- A: store the candidate signals; X: stores the restoration range of current signal; Y: records the additive length in X for each change
Contents

• Introduction
• Algorithm
• Experiments
• Conclusion
Experimental Setup

- **Setup**
  - Program with C++ on a Linux machine
  - Intel Xeon 3GHz CPU and 4GB memory
  - OS: Red Hat Enterprise Linux AS release 3
  - Gcc 3.4.6 with option -03
- **Front-end**
  - Benchmarks: some circuits from ISCAS’89
  - We have implemented a random 0/1 generator, using C++ function `rand()`
  - Trace buffer: 8*4k, 16*4k
- **Back-end**
  - Execution time: C++ function `clock()` and the macro `CLOCKS_PER_SEC`
  - A simulator computes the restoration ratio $r$
## Results

- **Comparison**
  - Ko’s selection algorithm in DATE’2008, referred as “gradual”; Xiao’s algorithm in DATE’2009, referred as “greedy”; Ko’s similar method in TCAD’09, referred as “cover”.

### The results for the buffer 8*4k

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>[6] [4] [7]</td>
</tr>
<tr>
<td>s382</td>
<td>21</td>
<td>51987 2.625 2.90</td>
<td>55977 2.99 2.64</td>
<td>59986 3.49 0.42</td>
<td>40356 11.09 1.51</td>
<td>4.22 3.71 3.18</td>
</tr>
<tr>
<td>s641</td>
<td>19</td>
<td>43997 2.38 12.01</td>
<td>56107 4.51 7.96</td>
<td>50995 7.37 3.73</td>
<td>36321 10.08 1.71</td>
<td>4.24 2.24 1.37</td>
</tr>
<tr>
<td>s1196</td>
<td>18</td>
<td>39664 2.24 6.18</td>
<td>39713 2.24 6.78</td>
<td>52164 5.34 1.24</td>
<td>33692 9.42 8.50</td>
<td>4.21 4.21 1.76</td>
</tr>
<tr>
<td>s1238</td>
<td>18</td>
<td>39664 2.24 5.88</td>
<td>39713 2.24 4.15</td>
<td>52164 5.34 1.10</td>
<td>33692 9.42 9.13</td>
<td>4.21 4.21 1.76</td>
</tr>
<tr>
<td>s1494</td>
<td>6</td>
<td>0 1.00 5.21</td>
<td>0 1.00 7.75</td>
<td>12000 2.00 3.01</td>
<td>10090 3.52 12.66</td>
<td>3.52 3.52 1.76</td>
</tr>
<tr>
<td>s1423</td>
<td>74</td>
<td>262087 9.19 226.22</td>
<td>266530 14.33 152.20</td>
<td>262081 9.19 29.07</td>
<td>144415 37.10 105.32</td>
<td>4.04 2.59 4.04</td>
</tr>
</tbody>
</table>

### The results for the buffer 16*4k

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>#RS r time(s)</td>
<td>[6] [4] [7]</td>
</tr>
<tr>
<td>s382</td>
<td>21</td>
<td>19997 1.31 4.31</td>
<td>55977 2.99 2.37</td>
<td>59986 3.49 0.39</td>
<td>40356 11.09 1.01</td>
<td>8.47 3.71 3.18</td>
</tr>
<tr>
<td>s641</td>
<td>19</td>
<td>11999 1.19 15.99</td>
<td>56107 4.51 7.04</td>
<td>50995 7.37 2.25</td>
<td>36321 10.08 1.13</td>
<td>8.47 2.24 1.37</td>
</tr>
<tr>
<td>s1196</td>
<td>18</td>
<td>7997 1.12 7.22</td>
<td>7999 1.12 8.39</td>
<td>52164 5.35 0.96</td>
<td>33692 9.42 7.08</td>
<td>8.41 8.41 1.76</td>
</tr>
<tr>
<td>s1238</td>
<td>18</td>
<td>7997 1.12 9.72</td>
<td>7996 1.12 7.45</td>
<td>52164 5.34 1.10</td>
<td>33692 9.42 13.73</td>
<td>8.41 8.41 1.76</td>
</tr>
<tr>
<td>s1423</td>
<td>74</td>
<td>230318 4.59 357.53</td>
<td>246605 21.55 90.88</td>
<td>258111 8.17 27.29</td>
<td>144415 37.10 80.39</td>
<td>8.08 1.72 4.54</td>
</tr>
</tbody>
</table>
Results

- Two tables
  - The results when 8 nodes and 16 nodes are selected as trace signals
  - “#TN” means the number of selected trace signals; “#RS” is the number of restoration states
  - “r” is the restoration ratio and “time” is execution time
- Results
  - The proposed algorithm can bring a higher restoration ratio than the previous methods, about 2 - 4 times
  - When the number of the trace nodes increases, our improvement on the ratio may be higher
- Limitation
  - Although pruning technique is used to reduce the huge search space, the executing time will still rise rapidly when the circuit size increases
Contents

- Introduction
- Algorithm
- Experiments
- Conclusion
Conclusion

• Conclusion
  – This paper proposes a pruning-based trace signal selection algorithm to improve the restoration ratio for the data acquisition in the post-silicon validation
  – The algorithm generate the visible restoration range for each FF, and explore different enumerations to find the better one. To accelerate the efficiency, it proposes the CNF-based constraints for the exhaustive pruning

• Future work
  – How to enhance the efficiency during the trace signal selection will be our emphasis