A Complete Approach to Unreachable State Diagnosability via Property Directed Reachability

Ryan Berryhill
Andreas Veneris
• Motivation
• Background
• Unreachability Debugging
• Incremental Application of PDR
• Experimental Results
• Future Work
• Conclusion
Outline

• Motivation
• Background
• Unreachability Debugging
• Incremental Application of PDR
• Experimental Results
• Future Work
• Conclusion
Motivation

• Functional verification can take up to 70% of the design effort, 60% of which is debugging

• Many errors manifest themselves with an error trace

• Traditional SAT-based debugging can be applied to accelerate the debugging process
Motivation

• Safety properties
  • The design can never reach a bad state
  • On failure, an error trace is returned

• Liveness properties
  • The design is capable of reaching a state
  • On failure, no error trace can possibly exist

• Without an error trace, traditional automated debugging cannot be applied
Motivation

- Techniques for debugging this type of liveness property exist
  - Approximation-based and incomplete

- We propose a PDR-based debugging technique for this type of liveness property
  - Complete and exact: returns every solution

- Possible to efficiently answer the question:
  How do I make the design reach this state?
• Motivation
• **Background**
  • Unreachability Debugging
  • Incremental Application of PDR
  • Experimental Results
• Future Work
• Conclusion
Background

• SAT-based debugging [Smith et. al TCAD ‘05]
  • Using an error-trace, finds all locations that can be changed to correct the circuit’s behavior
  • Unroll the circuit as an ILA, insert MUXes at the output of each gate
• E.g. for the following circuit and two-cycle counter-example
Background

- SAT-based debugging [Smith et. al TCAD ’05]
  - Using an error-trace, finds all locations that can be changed to correct the circuit’s behavior
  - Unroll the circuit as an ILA, insert MUXes at the output of each gate
- E.g. for the following circuit and two-cycle counter-example

![Circuit Diagram](image)
Background

- **Property-Directed Reachability [Bradley, VMCAII’11]**
  - Determines if a state is reachable or not
  - Compute an over-approximation of the set of reachable states in each clock cycle
  - Terminate when the over-approximation converges to an *inductive invariant*
  - Inductive invariant: a set of states closed under the circuit’s transition relation

- We propose a technique leveraging PDR to debug unreachability without an error trace
• Approximation-based approach [Berryhill and Veneris, DATE’15]
  • User sets a parameter $K$
  • Use PDR to over-approximate the set of states reachable in $K$ or fewer cycles
  • Debug a state transition from the over-approximation to the target state
• Solution set is not necessarily complete
• Finds all solutions that reach the target state:
  • within \((K+1)\) cycles; and
  • one cycle after an already-reachable state
• We propose a technique based on PDR that is complete by nature
  • No parameters to set
  • Higher runtime (roughly 4-5x)
Unreachability Debugging

• **Key idea:** create a new transition relation in which the target is reachable if solutions exist
  • Use a novel error model construction to perform the role of the debugging MUX
  • Use PDR to determine if the target is reachable
  • The counter-example returned by PDR indicates a solution

• **Result:** All solutions are found, but runtime is typically higher than the approximation-based approach
Unreachability Debugging

• Construct the enhanced transition relation
  • Insert the error model at each suspect location
  • If $e_i = 0$ the circuit’s behavior is unaffected
  • If $e_i = 1$ the location is replaced by free variable $w_i$
• The register output feeds back to its input
  • The register assumes a value as part of the initial state assignment and maintains it forever

![Diagram of a DQ flip-flop with inputs $e_i$, $l_i$, $w_i$, and output $z_i$.]
Unreachability Debugging

- Construct initial state constraints
  - Valid initial state of the circuit
  - Exactly $N$ active error-select registers
    - Find solutions where $N$ locations are simultaneously changed to correct the error
    - $N$ is the error cardinality
    - For simplicity, we only consider $N = 1$
Unreachability Debugging

• Execute PDR, using the target state as the unsafe state set

• If reachable, the active error-select register in the counter-example is a solution
  • Update the initial state constraint to block the solution, run PDR again

• If unreachable, no solutions exist
  • Terminate
• From the initial state \((s_1 = 0)\) it is impossible to reach the target state \((s_1 = 1)\)
Unreachability Debugging

- From the initial state \(s_1 = 0\) it is impossible to reach the target state \(s_1 = 1\).
- Setting \(e_2 = 1\) makes it reachable.
  - The AND gate is a solution.
  - Changing the AND to an OR indeed makes the target state reachable.

University of Toronto
Unreachability Debugging

- From the initial state \((s_1 = 0)\) it is impossible to reach the target state \((s_1 = 1)\)
- Setting \(e_1 = 1\) does not make the target state reachable
  - The OR gate is not a solution
Outline

• Motivation
• Background
• Unreachability Debugging
• **Incremental Application of PDR**
• Experimental Results
• Future Work
• Conclusion
Incremental Application of PDR

- PDR is called once for each solution found
- Each time, PDR is solving a very similar problem
  - Same transition relation, same property
  - One error-select register forced to 0
- Strictly fewer states are reachable
  - Those in which $e_i = 1$ are all unreachable
  - All others are unaffected
Incremental Application of PDR

• The reachable set after blocking is a subset of the reachable set before blocking
  • The over-approximations remain valid

• Re-use the over-approximations of PDR from the previous run
  • Gives an average 5.1x speedup
• Motivation
• Background
• Unreachability Debugging
• Incremental Application of PDR
• **Experimental Results**
• Future Work
• Conclusion
## Experimental Results

### Approximate Approach (K = 50)

<table>
<thead>
<tr>
<th>Design</th>
<th>#sol</th>
<th>%sol</th>
<th>Time</th>
<th>#sol</th>
<th>Time</th>
<th>#sol</th>
<th>Time</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrisc_core</td>
<td>10</td>
<td>100</td>
<td>15.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.9x</td>
</tr>
<tr>
<td>design1</td>
<td>4</td>
<td>29</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.9x</td>
</tr>
<tr>
<td>divider</td>
<td>1</td>
<td>1.9</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.8x</td>
</tr>
<tr>
<td>spi</td>
<td>40</td>
<td>100</td>
<td>19.0</td>
<td>40</td>
<td>598</td>
<td>40</td>
<td>76.6</td>
<td>7.8x</td>
</tr>
<tr>
<td>wb</td>
<td>247</td>
<td>95</td>
<td>38.7</td>
<td>261</td>
<td>9983</td>
<td>261</td>
<td>211</td>
<td>47.3x</td>
</tr>
<tr>
<td>usb_core</td>
<td>4</td>
<td>100</td>
<td>17.5</td>
<td>4</td>
<td>1065</td>
<td>4</td>
<td>492</td>
<td>2.2x</td>
</tr>
<tr>
<td>ac97_ctrl</td>
<td>10</td>
<td>56</td>
<td>1.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.7x</td>
</tr>
<tr>
<td>rsdecoder</td>
<td>40</td>
<td>100</td>
<td>371</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GEOMEAN</strong></td>
<td><strong>43</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>5.1x</strong></td>
</tr>
</tbody>
</table>

Incremental and initial approaches find the same (complete) set of solutions in every case.

Approximate approach often is able to find all of the solutions.

Approximate approach struggles to find many solutions on highly-pipelined designs.

Approximate approach finds 43% of the complete solution set.
### Experimental Results

#### Solutions Found for Both Approaches

<table>
<thead>
<tr>
<th>Solution Category</th>
<th>Approximate Approach</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>mrisc_core</td>
<td></td>
<td></td>
</tr>
<tr>
<td>design1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>divider</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wb</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>usb_core</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ac97_ctrl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rsdecoder</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- On pipelined designs, the approximate approach may find a small subset of the solutions
  - Finds solutions in the stage closest to error’s observation point
Experimental Results

Solutions Found vs. Time for Both Approaches (mrisc_core)

- Both take the same amount of time to find one solution
- The optimized approach finds subsequent solutions much more quickly
Outline

- Motivation
- Background
- Unreachability Debugging
- Incremental Application of PDR
- Experimental Results
- **Future Work**
- Conclusion
Future Work

• Efficient Suspect Selection [ISAIM’16]

• **Key Idea:** A location being a non-solution may imply other locations are also non-solutions
  
  • Not a register defining the target state (e.g. locations other than $s_1$); and
  
  • Has only one fanout (e.g. $l_3$, $l_2$, $l_1$); and
  
  • Single fanout is a non-solution
  
  • e.g. If $l_2$ is not a solution, $l_1$ is not either
Future Work

• Apply unreachability debugging iteratively
• Start with a suspect set including
  • Registers that define the target state ($s_1$)
  • Locations with multiple fanout
• Debug, “push back” through solutions
  • Add fanins of solutions to the suspect set
Future Work

- Construct Initial Suspect Set
- Debug
- Construct New Suspect Set
- Solutions Found?
  - Yes
  - No
    - Terminate

Solutions

Non-Solution

Solution
Outline

• Motivation
• Background
• Unreachability Debugging
• Incremental Application of PDR
• Experimental Results
• Future Work

• Conclusion
Conclusion

• Debugging unreachable states without an error trace
  • Construct an enhanced model of the circuit
  • Target state reachable if and only if solutions exist
  • Use PDR to find traces that reach the target state thereby indicating solutions

• Complete and exact by nature
  • Returns every solution in the design

• Future Work: Efficient suspect selection