Retrascope: Open-Source Model Checker for HDL Descriptions

Alexander Kamkin, Mikhail Lebedev, Sergey Smolov

Ivannikov Institute for System Programming of RAS
Formal Verification in HDL

**Goal:** check user-defined properties on HDL modules quickly, correctly and without user’s additional efforts

- **Interest & strength growth**
  - More automation, more tools (SMT solvers, abstraction)
  - Attempts to apply FV tools to industrial cases

- **Open Source invades HW world**
  - RISC-V, MIPS Open

- **Model checking**
  - Formal techniques for determining whether a given **model** satisfies given **specifications**
  - Counterexample generation (+)
  - “State explosion” problem (-)
HDL Model Checkers: History

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SMV</td>
<td>NuSMV</td>
<td>NuSMV 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesis tools</td>
<td>ABC</td>
<td>Yosys</td>
<td>SymbiYosys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External model checkers</td>
<td></td>
<td></td>
<td>Verilog2SMV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software verification</td>
<td>CBMC</td>
<td>VCEGAR</td>
<td>EBMC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Accellera Systems Initiative
Retrascope toolkit

- **Extendible framework** for functional verification and analysis of HDL modules
- Provides **engines** for code parsing, model extraction and visualization
- Several kinds of **models** are supported: CFG, EFSM, GADD

![Diagram](attachment:image.png)
Model checking flow in Retrascope

1) HDL code & specification parsing
2) Control Flow Graph (CFG) building
3) CFG → GADD transformation
4) SMV model generation
5) SMV model checking
   a) Standalone checker (NuSMV, nuXmv)
6) Model checker trace parsing
   a) HDL testbench generation
Guarded actions in CFG model

- **guard** – branch condition
- **action** – assignments
- **guarded action** (GA) is a pair $\gamma \rightarrow \delta$, where $\gamma$ is a guard and $\delta$ is an action
- **GADD** – Guarded Actions Decision Diagram
  - transformed CFG
  - GA are **atomic**
Guarded Actions Decision Diagram

- Phase ($\varphi$) – integer variable that keeps its value within a GA
- New value of $\varphi$ is assigned at the action of every GA
- GADD is translated into the SMV format

### Phase ($\varphi$) – integer variable that keeps its value within a GA

- $y = (x == 0) ? 1 : 0$
- $\varphi = 5$
- $x = 0$
- $\varphi = 2$
- $x = 1$
- $\varphi = 10$
- $x = 3$
- $\varphi = 7$

### New value of $\varphi$ is assigned at the action of every GA

- $y = y + x$
- $\varphi = 7$

### GADD is translated into the SMV format

- $z = y$
- $\varphi = 4$
- $arr = \text{STORE}(arr, index, x)$
- $\varphi = 9$
- $\varphi = 3$
- $\varphi = 5$
- $\varphi = 7$
- $\varphi = 0$
Properties in Retrascope

• Automatically generated for typical errors
  – Potential conflicts
    • Write-write: at least 2 processes write the same variable on the same tick
    • Write-read-write: no reads between two writes
    • Undefined: variable is read before initialization
  – Dead code detection
    • EFSM model extraction, enabling conditions for transitions are checked by nuXmv

• User-defined specifications
  – PSL (supported by the SMV-based checkers)
  – SVA subset → automatically translated to PSL
Evaluation: HDL Model Checkers

• **EBMC**
  – The Enhanced Bounded Model Checker
  – University of Oxford, England (Daniel Kroening)

• **SymbiYosys**
  – Front-end driver program for Yosys Open SYnthesis Suite
  – Clifford Wolf, Austria

• **Verilog2SMV**
  – Yosys-based Verilog-to-SMV translation tool
  – Foundazione Bruno Kessler (FBK), Italy
## Model checker overview

<table>
<thead>
<tr>
<th>Tool</th>
<th>EBMC</th>
<th>SymbiYosys</th>
<th>Verilog2SMV</th>
<th>Retrascope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Verilog, netlist, SystemVerilog, SMV</td>
<td>Verilog, SystemVerilog</td>
<td>Verilog, SystemVerilog</td>
<td>Verilog, SystemVerilog, VHDL</td>
</tr>
<tr>
<td>Properties</td>
<td>LTL, SVA subset</td>
<td>SVA subset</td>
<td>SVA subset</td>
<td>PSL, SVA subset</td>
</tr>
<tr>
<td>Distribution</td>
<td>binary</td>
<td>source code</td>
<td>source code</td>
<td>source code</td>
</tr>
<tr>
<td>Multi-file module processing</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Separate property checking</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>VCD counterexample generation</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HDL counterexample generation</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>GUI</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>License</td>
<td>ISC</td>
<td>ISC</td>
<td>GNU GPL v3</td>
<td>Apache License 2.0</td>
</tr>
</tbody>
</table>
HDL benchmarks

Quite a long story...

figure from: https://ddd.fit.cvut.cz/prj/Benchmarks
Selected Verilog benchmarks

• **Texas-97**
  – University of Texas, part of Prof. A. Aziz course on formal verification
  – 74 files, 45655 LOC, 0 properties

• **VCEGAR**
  – Benchmarks for VCEGAR tool (former EBMC)
  – 37 files, 11037 LOC, 2 SVA assertions

• **Verilog2SMV**
  – Benchmarks for Verilog2SMV tool
  – 58 files, 17634 LOC, 92 SVA assertions
Experiment #1: Verilog support

Reference compiler – ModelSim Starter Edition

1) Run tools on original benchmarks
2) Fix errors that were detected by ModelSim
3) Run tools again on fixed benchmarks

Tool behaviors (legend):

OK – no errors
ERR – error is found in erroneous Verilog module
FAIL – error is found in correct Verilog module
CRASH – unexpected crash
# Experiment #1: Results

<table>
<thead>
<tr>
<th>Tool</th>
<th>Texas-97</th>
<th>VCEGAR</th>
<th>Verilog2SMV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ok</td>
<td>Err</td>
<td>Fail</td>
</tr>
<tr>
<td>EBMC</td>
<td>12</td>
<td>0</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>SymbiYosys</td>
<td>8</td>
<td>0</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>47</td>
<td>17</td>
</tr>
<tr>
<td>Verilog2SMV</td>
<td>9</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>47</td>
<td>19</td>
</tr>
<tr>
<td>Retrascope</td>
<td>21</td>
<td>0</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>46</td>
<td>12</td>
</tr>
</tbody>
</table>
Unsupported (but correct!) Verilog

- **EBMC**
  - Assigning inputs to regs or outputs
  - **task** without parameters

- **SymbiYosys, Verilog2SMV**
  - 2 or more complement declarations of the same signal
  - Different types of left / right sides of assignments
  - Uninitialized variables
Experiment #2: Property checking

• In benchmarks – **assert property expr**
  – EBMC doesn’t support them, rewrite with \textit{always} and \textit{assert}(...)

• Time limit – 1 hour per module

• Bounded model checking, bound 100

• Tool behaviours (\textit{legend})
  - **FALSE** – property is \textit{false}, counterexample is generated
  - **TRUE** – property is \textit{true} or bound is reached, no counterexamples
  - **ERR** – error is found in erroneous Verilog module
  - **FAIL** – error is found in \textit{correct} Verilog module
  - **CRASH** – unexpected crash
  - **TIMEOUT** – time limit expiration

© Accellera Systems Initiative 16
Experiment #2: Results

- Yosys-based tools are unable to check properties separately
- Retrascope has found max num of counterexamples (“FALSE”)
- Retrascope is the best bug detector (“ERR”)
- Retrascope provides the largest Verilog support (“FAIL”)
- Most stable tool is EBMC (“CRASH”)
- Performance leader is EBMC; SymbiYosys is close (“TIME”)

<table>
<thead>
<tr>
<th>Tool</th>
<th>FALSE</th>
<th>TRUE</th>
<th>ERR</th>
<th>FAIL</th>
<th>CRASH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBMC</td>
<td>40</td>
<td>105</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>SymbiYosys</td>
<td>29</td>
<td>98</td>
<td>0</td>
<td>11</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>VerilogSMV</td>
<td>27</td>
<td>90</td>
<td>0</td>
<td>13</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>Retrascope</td>
<td><strong>44</strong></td>
<td>81</td>
<td><strong>4</strong></td>
<td><strong>1</strong></td>
<td>3</td>
<td>15</td>
</tr>
</tbody>
</table>

© Accellera Systems Initiative
Case study

• Dispatcher module from Intel Quartus Prime
  – 2200 LOC, 8 sub-modules
  – SystemVerilog properties in a separate top level checker

• Retrascope only is able to check it formally! (small bound)
  – EBMC: doesn’t support `defparam`, local values for params
  – SymbiYosys: incorrect param redefinition
  – Verilog2SMV: incorrect sub-module instantiation, param redefinition
Contacts

Retrascope toolkit page: https://forge.ispras.ru/projects/retrascope

E-mail: retrascope-support@ispras.ru
Thank you!

Questions?