Overcoming System Verilog Assertions limitations through temporal decoupling and automation

Mattia De Pascalis, Xia Wu, Matteo Vottero, Jacob Sander
Background

• SVA is a de facto standard for Formal Verification
  • Completeness of logical relations
  • Timing relations between several events or sequences
  • Glue logic used to abstract class of problems

• Identify and automate a class of assertions
  • Common abstraction/glue logic
  • Decouple Assertion and glue logic
  • Automatic direct and inverted assertion generation
Class of problem(s) considered in the paper

- Temporal relationship between two "generic" sequences SeqA, SeqB
  - d is dynamic (reconfigurable)
  - Internal uncertainties DL, DH] (DL, DH constants and d – DL > 0)
  - IF SeqA THEN ##[ d – DL : d + DH ] SeqB
Problems discussed in the paper

• **SVA language does not support dynamic delays**
  • Need to abstract the problem
  • Need to develop an architecture that implement the solution

• Decouple Sequences, helper logic and assertions
  • Sequences shall be developed as “standalone” library
  • Shall be possible to try different type of helper logic models
  • It should be easy to debug

• Automatic generation of inverse assertion
Abstract and overcome SVA limitation

\[ SeqA.\text{triggered} \rightarrow \#\#[d - DL : d + DH] \text{SeqB.\text{triggered}} \]

- \( d = \) dynamic delay, DL and DH constants
- Not supported in SVA as it is

• Assumptions:
  - Introduce SeqAD = SeqA delayed of K clock cycles
  - Substitute SeqA with SeqAD in [1]
  - Express new timing relationship between SeqAD and SeqB
Abstract and overcome SVA limitation

\(\text{SeqAD.triggered} \rightarrow \#\#[\text{d} - \text{K} - \text{DL} : \text{d} - \text{K} + \text{DH}] \ \text{SeqB.triggered} \quad [2]\)

- Make use of auxiliary logic such that \(\text{K} = \text{d} - \text{DL}\)

\(\text{SeqAD.triggered} \rightarrow \#\#[\text{d}-(\text{d} - \text{DL}) - \text{DL} : \text{d}-(\text{d} - \text{DL}) + \text{DH}] \ \text{SeqB.triggered} \quad [3]\)

\(\text{SeqAD.triggered} \rightarrow \#\#[0 : \text{DL} + \text{DH}] \ \text{SeqB.triggered} \quad [4]\)

- [4] does not contain dynamic delay anymore
- SeqAD can be modelled with auxiliary code
- [4] Can be expressed in SVA
Problems discussed in the paper

• SVA language does not support dynamic delays
  • Need to abstract the problem
  • Need to develop an architecture that implement the solution

• Decouple Sequences, helper logic and assertions
  • Sequences shall be developed as “standalone” library
  • Shall be possible to try different type of helper logic models
  • It should be easy to debug

• Automatic generation of inverse assertion
SVAGen architecture

• Python framework
  • Templates to describe a specific auxiliary logic architecture
  • Describes timing relationship between SVA sequences
  • Configure each assertion with any of the available templates
  • Configure each assertion with higher/lower logic (help debug)
  • Enable/Disable automatic inverse assertion generation

• Available as opensource tool
SVAGen architecture

Choose Template
Describe/config Assertions
SVAGen
Auxiliary code
Direct Assertions
Inverse Assertions
SVA sequence and property library
dut.sv
top_sva.sv
Formal Tool(s)

Enable visibility of internal signals
Change/customize template

Formal proof found
Cannot Converge
Failure
Validation and test

• **DUT definition**
  - $d$ : Reconfigurable delay
  - $L$ : rnd in range $[-1 : +2]$ (LSFR)
  - $d - DL \geq 0 \rightarrow d \geq 1$

• **SVA Sequence definition**
  - Develop generic sequences
  - No timing relationship in sequences

```vhdl
sequence s_a;
  $rose(a);
endsequence: s_a;

sequence s_b;
  $fell(b);
endsequence: s_b;
```
Example

- SVAGen configuration
  - Select template for dynamic delay
  - Configure assertion
  - Run SVAGen tool
  - Integrate output file into Formal TB

```python
assertionModule = AssertionContainer(
    Parser(
        "/sva_gen/templates/fifo/template.yaml", <- Template YAML
        "/sva_gen/templates/fifo/template.svh", <- Template SVH
        "out_macro.svh",
        "out_checkers.svh")
    )

assertionModule.add_assertion(
    AssertionRangeDelay(
        "rose_a_then_fell_b",
        "s_a",
        "s_b",
        "d",
        "f",
        "g",
        1000)
    )
```
Example

• Shift Reg
  • Register s_a trigger in position [d – DL]
  • Advance every clk cycle 1 position

• Validity Window
  • First “event” activate direct assertion
  • Inv assertion looks at full Validity Window

• Helper logic generated by SVAGen
  • RangeDelay Template in this example
Example

- Auxiliary logic
  - Handle corner cases
  - Handle fifo/shift reg update
  - Handle Validity window update
  - Handle runtime addr calculation
  - Update all data structures
Example

- Assertions generated from SVAGen
  - Only static delay are used
  - Full proof with formal tool

```verilog
property p_rose_a_then_fell_b;
  validity_window[2] |-> ##[0 : `DL + `DH] rose_a_then_fell_b_triggered;
endproperty: p_rose_a_then_fell_b;

a_rose_a_then_fell_b         : assert property(`CLK_SEQ p_rose_a_then_fell_b);
```
Problems discussed in the paper

- SVA language does not support dynamic delays
  - Need to abstract the problem
  - Need to develop an architecture that implement the solution
- Decouple Sequences, helper logic and assertions
  - Sequences shall be developed as “standalone” library
  - Shall be possible to try different type of helper logic models
  - It should be easy to debug
- **Automatic generation of inverted assertion**
Example

- Assertions generated from SVAGen
  - Inv assertion exploits Validity Window
  - Validity Window stores info about past events
  - Full proof with formal tool

```
property p_rose_a_then_fell_b_inv;
  rose_a_then_fell_b_triggered |-> validity_window != ‘0;
endproperty: p_rose_a_then_fell_b_inv;

a_rose_a_then_fell_b_inv : assert property(`CLK_SEQ  p_rose_a_then_fell_b_inv);
```
Validation and results

- Time to find proof depends on 2 main factors:
  - Interval of delay
  - Range of delay
  - Always used uncertainties of +/- 3

<table>
<thead>
<tr>
<th>Range</th>
<th>A.D</th>
<th>AINV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0,2</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>30</td>
<td>1,6</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>971</td>
<td>980</td>
<td>27,5</td>
</tr>
<tr>
<td>981</td>
<td>990</td>
<td>28,2</td>
</tr>
<tr>
<td>991</td>
<td>1000</td>
<td>11,6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interval</th>
<th>Time (sec)</th>
<th>A.D</th>
<th>AINV</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>Max 10</td>
<td>1</td>
<td>0,2</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>1,3</td>
<td>1,1</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>1,4</td>
<td>1,2</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td>1,4</td>
<td>1,5</td>
</tr>
<tr>
<td>941</td>
<td>960</td>
<td>13,9</td>
<td>24,5</td>
</tr>
<tr>
<td>961</td>
<td>980</td>
<td>12,6</td>
<td>15,7</td>
</tr>
<tr>
<td>981</td>
<td>1000</td>
<td>16,3</td>
<td>16,7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interval</th>
<th>Time (sec)</th>
<th>A.D</th>
<th>AINV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>Max 20</td>
<td>1</td>
<td>0,2</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
<td>1,7</td>
<td>2,3</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>1,4</td>
<td>1,2</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td>1,4</td>
<td>1,5</td>
</tr>
<tr>
<td>951</td>
<td>1000</td>
<td>32,4</td>
<td>27,7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interval</th>
<th>Time (sec)</th>
<th>A.D</th>
<th>AINV</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>Max 50</td>
<td>1</td>
<td>0,2</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>4,4</td>
<td>3,8</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>1,4</td>
<td>1,2</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td>1,4</td>
<td>1,5</td>
</tr>
<tr>
<td>901</td>
<td>950</td>
<td>56,9</td>
<td>47,6</td>
</tr>
<tr>
<td>951</td>
<td>1000</td>
<td>32,4</td>
<td>27,7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interval</th>
<th>Time (sec)</th>
<th>A.D</th>
<th>AINV</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>Max 100</td>
<td>1</td>
<td>0,2</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>4,4</td>
<td>3,8</td>
</tr>
<tr>
<td>21</td>
<td>40</td>
<td>1,4</td>
<td>1,2</td>
</tr>
<tr>
<td>41</td>
<td>60</td>
<td>1,4</td>
<td>1,5</td>
</tr>
<tr>
<td>901</td>
<td>950</td>
<td>56,9</td>
<td>47,6</td>
</tr>
<tr>
<td>951</td>
<td>1000</td>
<td>32,4</td>
<td>27,7</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1352</td>
<td>843</td>
<td>390,8</td>
<td>505,8</td>
</tr>
<tr>
<td>675,67</td>
<td>973,67</td>
<td>981</td>
<td>1313,8</td>
</tr>
</tbody>
</table>
Conclusion

• Automatic handling of dynamic delay
• Automatic handling of inverse assertion
• Easy framework to switch auxiliary logic
• Reusable methodology for formal and simulation
• Tool is open sourced and available for download at this link
Future development

• Investigation for more efficient auxiliary logic

• Refactoring of SVAGen for better reusability
Questions