The Importance of Complete Signoff Methodology for Formal Verification

Iain Singleton, Mahesh Parmer, Geogy Jacob
Agenda

• Introduction to formal signoff

• Case Study – BPU
  – Block overview
  – Signoff methods
  – Results

• Conclusions
Introduction to Formal Signoff

Formal usage across the industry is at an all time high

Great!

The exhaustive nature of formal means that it is held to a higher standard

Makes sense

Without a thorough signoff methodology you could still miss bugs!

Wait… what?!
Introduction to Formal Signoff

Formal is exhaustive but only with respect to what you write!

• Assertions are verified with respect to constraints

Overconstraints prevent legal inputs from reaching the design & cause missed bugs
Introduction to Formal Signoff

Formal is exhaustive but only with respect to what you write!

- The behavior in the design which is tested is limited to the assertions that are written.

No assertion means no verification!
The ultimate goal of formal signoff is to ensure that “what you write” does in fact cover all the scenarios you expected.
Case Study – Branch Prediction Unit

The BPU sits within the Instruction Fetch Unit of a high performance low power microprocessor design.

The goal of the BPU is to reduce the branch penalty in highly pipelined designs to improve computational performance.
Case Study – Branch Prediction Unit

- Information received from Execution Unit
- Dynamic prediction with global history
- Dual bank memory with up to 4 instructions aligned to 16 bit boundaries
Formal Verification Approach

• BPU chosen target for formal despite extensive simulation testing
  – High level of control complexity and potential for hidden bugs made this a good target

• Initial FV approach did not have a well defined closure criteria
  – Properties developed
  – Designer reviewed properties
  – Human analysis of bounded depths

Human review techniques are valuable but not as extensive as fully defined methodologies
### Initial FV Work Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertions</td>
<td>141</td>
</tr>
<tr>
<td>Covers</td>
<td>19</td>
</tr>
<tr>
<td>Constraints</td>
<td>40</td>
</tr>
<tr>
<td>Assertion Depth Bound</td>
<td>Capped at 15</td>
</tr>
<tr>
<td>%Bounded Proofs</td>
<td>80%</td>
</tr>
<tr>
<td>Other Metrics</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- Large number of bounded proofs
- Restricting bound could cause missed bugs
- No other confidence metrics
Our Signoff Approach

Step 1: Bounded Depth Analysis

1. Generate cover points within COI of property
2. Analyze cover points using shortest path formal engines
3. Maximum depth reached gives a rough idea of the sequential depth of the property – any number less than this shows lack of required exploration

Performing this analysis showed us that a number of cover points were reached at depths beyond 15

Potential missed bugs!
Our Signoff Approach

Steps 2 - 5

Essential
- Overconstraint Reachability Analysis

Strong
- Structural property Cone of Influence analysis

Stronger
- Formally generated core of driving logic on properties

Strongest
- Automated fault injection assertion stress testing
Signoff Approach

- Beyond the bounded analysis there is a four step approach to formal signoff

1. Essential
   - Overconstraint Reachability Analysis

2. Strong
   - Structural property Cone of Influence analysis

3. Stronger
   - Formally generated core of driving logic on properties

4. Strongest
   - Automated fault injection assertion stress testing

Reachability analysis of cover property targets in the presence of constraints

Identifies any constraints which are preventing legal behaviour
Signoff Approach

- Beyond the bounded analysis there is a four step approach to formal signoff

1. Essential
   - Overconstraint Reachability Analysis

2. Strong
   - Structural property Cone of Influence analysis

3. Stronger
   - Formally generated core of driving logic on properties

4. Strongest
   - Automated fault injection assertion stress testing

Analyzing the registers within the structural COI of all the assertions

Finds holes in verification quickly, but due to structural nature can miss holes
Signoff Approach

• Beyond the bounded analysis there is a four step approach to formal signoff

1. Essential
   - Overconstraint Reachability Analysis

2. Strong
   - Structural property Cone of Influence analysis

3. Stronger
   - Formally generated core of driving logic on properties

4. Strongest
   - Automated fault injection assertion stress testing

Formal analysis of logic required to prove or reach a bounded proof for the properties.

Finds holes in verification more accurately than a simple COI analysis.
Formal Core Example

- Formal Core is Stronger than COI
  - Formal Core indicates which signals are involved in the proof of an assertion
  - If something within the COI is not required to prove the property it has not been tested

```p1: assert property ( cnt==4'hf |-> full );
```

- For P1 we can see that only full is inside the formal core
Formal Core Coverage

• Adding P2 means we now have the formal core from two properties
  – push, pop and cnt are involved in the proof of P2 and reported in the Formal Core

P1: assert property ( cnt==4'hf |-> full );
P2: assert property ( ~push&~pop |=> $stable(cnt) );

• Formal Core shows that we are testing something about a register BUT
  – What happens to cnt when we have a push or pop?
Signoff Approach

- Beyond the bounded analysis there is a four step approach to formal signoff

1. Essential
   - Overconstraint Reachability Analysis

2. Strong
   - Structural property Cone of Influence analysis

3. Stronger
   - Formally generated core of driving logic on properties

4. Strongest
   - Automated fault injection assertion stress testing

   Analyze the results of properties in the presence of artificially inserted faults.

   If at least one property fails in presence of a fault then assertion is good, if none fail then indication of verification holes.
FTA Example

• FTA is stronger than Formal Core
  – FTA checks whether assertions can catch injected faults
  – Formal Core checks that *something* about a register is checked
  – FTA checks whether *other features* of that logic are checked

• In our example we’re only checking the value of the counter when push and pop are low
  – We are testing the counter
  – But only one part of it
  – Lets look at what FTA will do...
FTA Example

P1: assert property ( cnt==4'hf |-> full );
P2: assert property ( ~push&~pop |=> $stable(cnt) );

always @(posedge clk or negedge rst_x)
    if (!rst_x) begin
        cnt <= '0;
    end
    else begin
        if (0/*push && !pop && cnt!=4'hf*/) // ConditionFalse
            cnt <= cnt + 1'b0/*1'b1*/; // BitFlip
        else if (!push && pop && cnt!=4'h0)
            cnt <= cnt +/-/*-*/ 1'b1; // Operator
        else
            cnt <= cnt;
    end

assign full = (cnt==4'hf);

We didn’t check for increment and decrement so P1 and P2 pass with these faults!

FTA reports “non-detected faults” as verification holes
BPU Results

Bounded Analysis
Several cover points at depths 16-20
Potential missed bugs beyond depth 15

Property Cone of Influence
Showed 87% register coverage
13% of design registers not tested by basic metric

Automated Fault Injection
70% faults detected
More comprehensive metric shows 30% design untested
Action on Results: Missing Assertions

Analysis had shown that approx. 30% of the design was untested with formal
- Review untested areas with designer
- Add missing checks

8 missing assertions added

Fault injection signoff metric used to confirm
Action on Results: Depth limitation

Analysis had shown the depth of 15 was not sufficient
- Remove depth limitation
- Use abstraction and engine techniques to improve bound

Result: 33 previously bounded @15 assertions now FAIL!

4 MISSED RTL BUGS!
Conclusions

• Formal verification is a very powerful tool
  – On a well simulated design a number of bugs found in initial work

• Quality of formal is only as good as the properties you write
  – Without thorough analysis potential for missed bugs is there
  – 4 RTL bugs were found and fixed in a well simulated + formal design by using signoff techniques

• Signoff techniques are essential to obtain higher confidence in formal environments
  – All signoff work in this paper performed using Synopsys VC Formal
Questions?