INTRODUCTION

- Memory subsystem is essential part of any SOC, traditional verification doesn’t guarantee of catching design corners scenarios.
- Usage of formal verification for a complex memory subsystem design is not an easy task because of its huge state space of the design.
- Solving the problems of subsystem verification in structured approach.
- Filling the gap of dynamic simulation using formal simulation.
- Usage of advance approaches to verify a memory subsystem.
- Faster approach for early design bring up.

OBJECTIVES

To do formal verification Memory Subsystem

FORMAL METHODS USED FOR VERIFICATION

- Property Check
- Connectivity Check
- Register Check
- Sequence Equivalence Check
- ABVIP Based Verification
- Split based approach to solve the formal properties

SPLIT BASED APPROACH

FV task can be subdivided for complexity through case splitting. The problem can be defined as

\[ f(x, y, z) = ax + by + c(z) \]

Function is dependent on its variables. Splitting is done in such a way that the range of x is kept random when y and z are constant and vice versa. By doing this the problem complexity is minimized

VERIFICATION FLOW

- Identification of all possible formal vendor APIs.
- Property and register split check of individual IPs.
- Get full proof of all properties.
- Apply the split based approach.
- Apply appropriate formal engine for unsolved properties.
- Do the connectivity check between IPs.
- Automate the regression flow and ensure 100% coverage.

RESULTS

<table>
<thead>
<tr>
<th>IP name</th>
<th>Total number of properties</th>
<th>Number of undetermined properties Before split</th>
<th>After split</th>
<th>Number of undetermined Before split</th>
<th>After split</th>
<th>Undetermined solved using split</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP1</td>
<td>778</td>
<td>33</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>IP2</td>
<td>452</td>
<td>90</td>
<td>72</td>
<td>18</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>IP3</td>
<td>847</td>
<td>351</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>IP4</td>
<td>2827</td>
<td>526</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>IP5</td>
<td>2818</td>
<td>351</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

BUGS FOUND

- Performance: We found a performance, which we would not have found through functional integration verification.
- Connectivity: Four issues were found through connectivity verification early in the project.

CONCLUSIONS

- Though Formal was applied late in the project, but we got significant results.
- 9 issues were found in Architecture, Performance, Register and IP Design.
- Automation helped us to reduce verification time.
- Faster verification for design bring up.
- We got 6x ROI with respect to functional verification

REFERENCES


Thanks Ketki Gosavi who helped on the resolving tool issues from Cadence