An Intelligent SOC Verification Platform

Deyong.Yang(ydy12345678@163.com)
Fabo.Deng(fabo.deng@unisoc.com)
Background Introduction

• The features of SOC are more and more complex, but time cost and manpower cost needed to verify these features are more and more.
• To rapidly meet changing market demand and further reduce cost, the project team wishes the faster SOC verification and less manpower.
• SOC features have a strong regularity, So we can build a universal SOC model.
SOC Universal Model

• SOC Model can be abstracted into relationships of master, slave and region.
• Build excel format to express this relationships.
• Design universal verification model to this relationships.
Design Universal SOC Verification Model

- Systba (System Test-bench automaton) based on SOC model
  - Common sequences and tests
  - Golden behavior model for check
  - Common coverage database for common features

- Systba test features
  - BusMatrix Connections
  - BusMatrix address decoder and remap
  - Channel Performance
  - SOC Low-power
  - SOC Anti-hang and so on
Contrast Between Systba and Others

<table>
<thead>
<tr>
<th>Function</th>
<th>Cadence Interconnect TB</th>
<th>Systba</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto hook VIP (APB/AHB/AXI/ACE...)</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Modify VIP configure/constraint on fly</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Auto generate common features sequence</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Auto common features check Mechanism</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Auto generate common features coverage</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Easy bottom-to-up integration multi-subsys</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Compatible C code</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Easy Debug</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Easy review</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Isolation testbench with VIP vendor</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Performance measurement</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
Systba Architecture

• To layer implementation of relationships intelligent inference, configuration, stimulus and checker.
• To isolate test-bench usage with vendor’s IP specification.
Systba’s Common Pattern (1/2)

- Auto create Common Pattern
  - Special constraints based on known information;
  - Special transaction;
  - Special sequence;
  - Special test;
Systba’s Common Pattern (2/2)

- Users can quickly and easily create their own sequence;
- Support for using regular expressions to represent model features;
- Easy add features to special master, special slave, or a special path;

```
Class no_send_burst_length16 extends systba_axi_trans;
  function new();
    super.new();
  endfunction
  constraint no_length16{
    if(tr.direct==WRITE) {tr.burst_length != 16;}
  }
endclass
```

```
no_send_burst_length16  m_no_burst_length16;
m_no_burst_length16 = new("No_burst_length16");
m_resource_man.add_constraint(m_no_burst_length16, "master1->slave1->*");
```

.. raw:: html

    <a href="https://example.com">This is a link</a>
Systba Golden Behavior Model (1/2)

- The master knows that it can access the corresponding slave and region, similarly, also know it cannot access the slave and region.
- So can build access check mechanism based-on relationship of master to slave
Systba Golden Behavior Model (2/2)

- Behavior model
  - Address remap feature;
  - Address decode feature;
  - Access switch feature;
  - Control mechanism, including clock, reset, low-power and firewall;
Bottom to Up Reuse Subsystem

- Support bottom to up reuse subsystem
- The structure of each subsystem is the same, so can easy cascade subsystem by Systba attribute configure
- The platform architecture of N subsystem is the same as a subsystem
Low-Power Verification Demo by Systba

- SOC low-power include clock gating, power-gating and DVFS;
- SYSTBA intelligent generate low-power stimulus;
- SYSTBA low-power manager control clock/power SVA check enable and check low-power behavior;
Security Verification Demo by Systba

- SOC security scenario include security/non-security master access security/non-security slave;
- Check data-path behavior to meet security scenario;
- Check security information transform correctly;
Summary

• SOC common features and possibility of abstraction into model;
• Characteristics of SOC model, including architecture, test pattern and behavior model;
• Bottom to Up Reuse Subsystem;
• Introduction verification demo for Low-power and security scenario;
Thanks you