An Elegant scoreboard eco-system deploying UVM Callbacks, Parameterization for Multimedia designs from Imaging perspective

Chakravarthi Devakinanda, Sahana Ranganathan, Devendra Satish Bilaye, Vivek Kumar, Karthik Majeti

Samsung Semiconductor India Research
Agenda

• Overview of CMOS Image Sensors
• Challenges in Multimedia SoC Verification
• Requirements from a Multimedia Testbench
• Addressing the challenges & requirements
• Results and Key Takeaways
• Q&A
Overview of CMOS Image Sensors

Analog data flows into an ADC, which converts it into digital data. Control signals are sent to the TIMING GENERATOR, which generates timing for the ISP chain. The ISP chain processes the noisy image, resulting in a processed image. Finally, the processed image is output as IF.
Emerging market trends

- High resolution
- More Sensors
- New features
Challenges in Product development

- Shrinking product lifecycles - Spec to silicon < 6 months
  - < 3 months to close all verification metrics

- Higher resolutions (200MP) -> Simulation time ↑

- Increasing Design complexity -> Requires exhaustive testing

- Meet stringent security, safety & quality standards
Basic Multimedia Test Bench
Bottleneck in TB - Scoreboard

• Bigger Designs -> 40+ blocks

• Higher resolutions -> Huge Amount of Data

• More Development time

• Limited Emulator licenses
Basic Scoreboard Structure

• Golden Reference Models:
  • C, C++, python etc
  • end-to-end
  • Unit models

• Unit level score boarding
  • End-end data integrity

Test Bench

M = Monitor class
SB = Scoreboard class

Reference model
Comparison
Reference model
Comparison
Reference model
Comparison

DUT

BLOCK-1

BLOCK-2

BLOCK-3
Requirements from a Multimedia Scoreboard

Scalable
- New Algorithms -> New IPs
- 40+ blocks
- Higher resolutions

Adaptable
- Mobile Image Sensors
- Automotive and Other Image Sensors

Reusable
- Support for Emulation
- Verification at SOC, Sub-System and IP level
- Support for Gate Level Sims, Power Aware

Automation
- Automation friendly at all steps – setup, regression, closure

Resource Management
- Optimal utilization of -> LSF, disk, man-power
- Avoiding Simulation Crashes
- Gain in run-times
Addressing the challenges & Requirements

1. PARAMETERIZATION
2. UVM CALLBACKS
3. FILE BASED DATA HANDLING
4. AUTOMATION
Parameterization

- **UVC Parameters** – monitor and driver handling
- **Scoreboard Parameters** – scoreboard handling

```c
// UVC PARAMS
parameter uvc_params_t uvc_default_params = '{
  // Default values using defines
  `NUM_CHANNELS,
  `DATA_WIDTH,
  `HADDR_WIDTH,
  `VADDR_WIDTH,
  `FLAG_WIDTH
};// SB PARAMS
typedef struct{
  int NUM_OF_IN_CHANNEL;// Reference model data channels
  int NUM_OF_OUT_CHANNEL;// Expected data channels for comparison
  string BLOCK_NAME; // Directory name.
  int ADDR_CHECK_EN; // To enable address checks
  int FLAG_CHECK_EN; // To enable flag checks
} sb_params_t;

// TRANSACTION PACKET
class uvc_data_packet_c #(uvc_params_t params = uvc_default_params)
extends uvm_sequence_item;
{
  rand bit [(params.DATA_WIDTH-1):0] data[(params.NUM_CHANNELS-1:0)];
  rand bit [(params.HADDR_WIDTH-1):0] haddr;
  rand bit [(params.VADDR_WIDTH-1):0] vaddr;
  rand bit [(params.FLAG_WIDTH-1):0] flag;
endclass: uvc_data_packet_c
```
Parameterization – IP

Reusable SB and Monitor

DUT 1

**UVC_DUT1_INPUT_PARAMS**
- NUM_CHANNELS = 4
- DATA_WIDTH = 12
- FLAG_WIDTH = 0

**UVC_DUT1_OUTPUT_PARAMS**
- NUM_CHANNELS = 4
- DATA_WIDTH = 10
- FLAG_WIDTH = 0

**SB_DUT1_PARAMS**
- BLOCK_NAME = "DUT1"
- FLAG_CHECK_EN = 0

DUT 2

**UVC_DUT2_INPUT_PARAMS**
- NUM_CHANNELS = 8
- DATA_WIDTH = 10
- FLAG_WIDTH = 0

**UVC_DUT2_OUTPUT_PARAMS**
- NUM_CHANNELS = 4
- DATA_WIDTH = 12
- FLAG_WIDTH = 4

**SB_DUT2_PARAMS**
- BLOCK_NAME = "DUT2"
- FLAG_CHECK_EN = 1
Parameterization - SOC

Scalable

Test Bench

DUT

BLOCK-1
12 bit
4-ch

BLOCK-2
10 bit
4-bit flag
4-ch

BLOCK-3
11 bit
16-ch

M

SB

Reference model

Comparison

M

SB

Reference model

Comparison

M

SB

Reference model

Comparison

Reference model
UVM Callbacks

- Scoreboard callback class
  - Implementation logic
- Generic scoreboard class
  - Trigger logic

```cpp
//Callback connections
uvm_callbacks#(generic_sb_c#(sb_params_block1,uvc_block1_input_params,uvc_block1_output_params),sb_callback_base)::add(sb_block1,callback_block1);
```

![Diagram showing callback connections and logic blocks](image_url)
UVM Callbacks – scoreboard callback class

uvm_callback

- sb_callback_base
  + pack_input_data()
  + pack_output_data()
  + compare_data()

  sb_callback_block1

  sb_callback_blockN
  + pack_output_data()
  + compare_flag()

Reusable

Resource Management

Scalable
UVM Callbacks – Generic Scoreboard class
File-based monitor

- Queue based monitor – previous method
  - Huge data → Simulation crash ❌
  - Big_mem LSF

- File based monitor – new method
  - Fewer Big_mem LSF
  - Leverage the benefits of both Emulation and Simulation

- Queue based + file based hybrid approach.
File-based approach Advantages

- RTL in HW + SB in software

- Emulation + simulation de-coupled Run with file-based approach.
Automation

- XLS
  - Parameters
  - Connections

- XLS -> Script -> DUMP
Results

Simulation Run-times with SB

- Queue based
- File Based

Chart Title

- Simulation Bringup: Old Method - 14 Days, SB eco-system - 21 Days
- Emulation Bringup: Old Method - 12 Days, SB eco-system - 7 Days
- IP to SOC porting: Old Method - 2 Days, SB eco-system - 1 Day

Regression - 1000 tests

- Old Method: 3 Days
- SB eco-system: 1.5 Days
### Key takeaways

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Reusability</th>
<th>Resource Management</th>
<th>Adaptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily scalable to large designs -&gt; members of the team can work on different blocks independently.</td>
<td>Allows reuse of scoreboard files in IP, Sub-system &amp; SoC.</td>
<td>Reduced big_mem LSF use, optimize emulator use, gain in run-times, fewer simulator crashes.</td>
<td>Improved code readability – easy rampup.</td>
</tr>
<tr>
<td></td>
<td>Rapid scoreboard deployment - quickly sanitize in acceleration platform and regressions can be run in simulation.</td>
<td></td>
<td>Easily adaptable for other Multimedia Designs</td>
</tr>
</tbody>
</table>
Thank You

Q & A