

OMI Verification using SmartDV

SmartDV Technologies

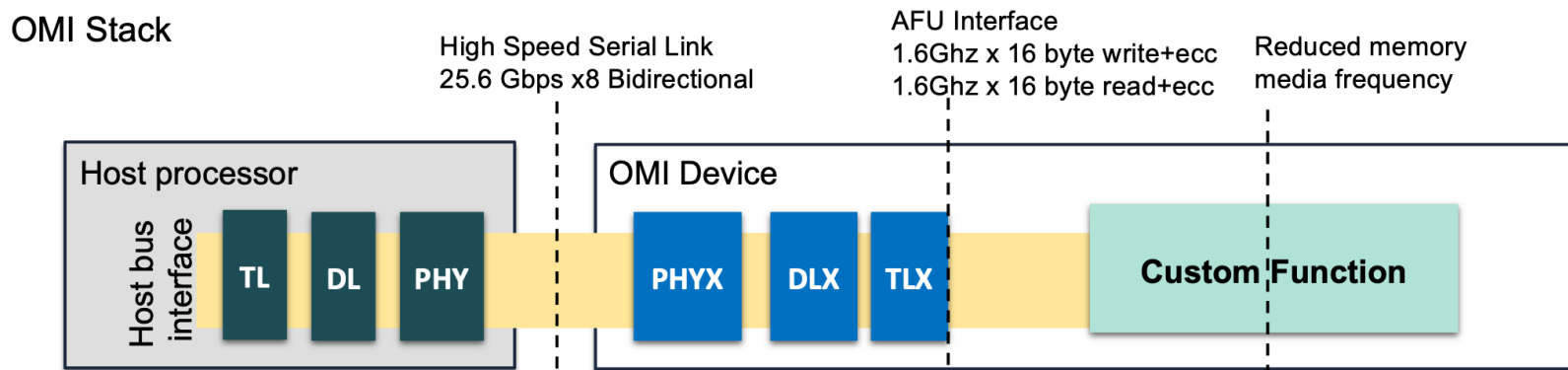
Bipul Talukdar,
Director, Applications Engineering

Topics

- ❖ **OMI Verification IP Features**
- ❖ **SmartDV OMI VIP Development, Architecture**
- ❖ **Key Benefits**
- ❖ **Configurations**
- ❖ **Callbacks**
- ❖ **Error Injection**
- ❖ **OVF/UVM Sequences**

OMI Verification IP Features

- Compliant with versions 3.0, 3.1 and 4.0.
- Complete OMI Tx/Rx with complete AFU I/F functionality.



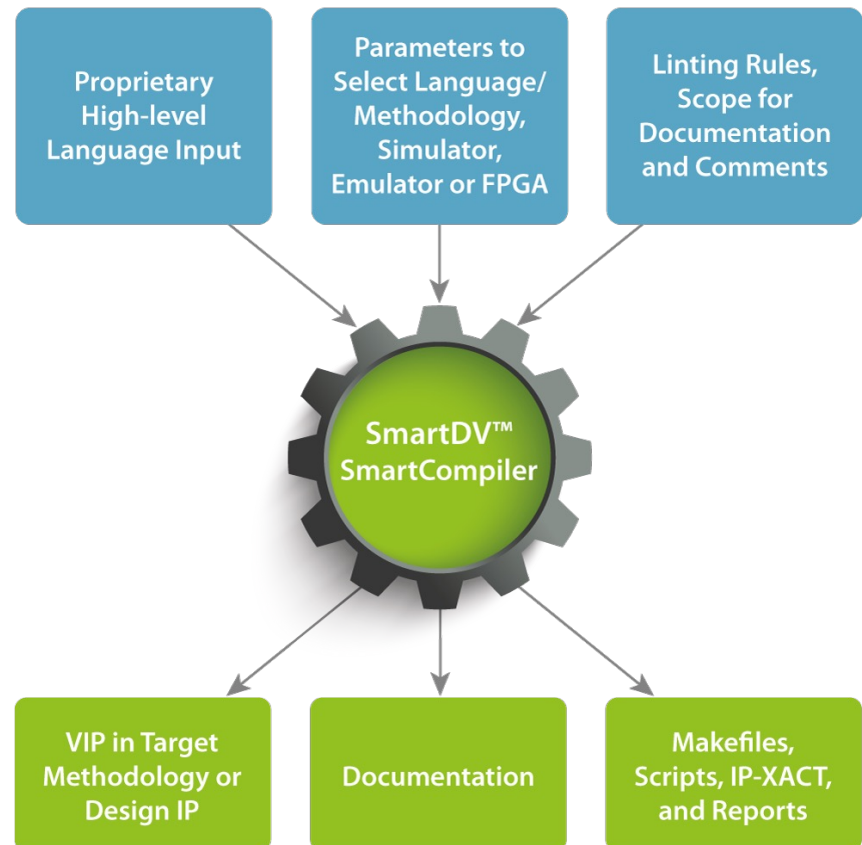
- Can set up verification environment with PCIe PHY
- Compliant with PHY Training and PHY Initialization
- Supports dot-ow and dot-xw formats of commands and responses
- Supports TL and TLX template specifications x'04' through x'08'
- Supports DL training sets.

OMI Verification IP Features (contd.)

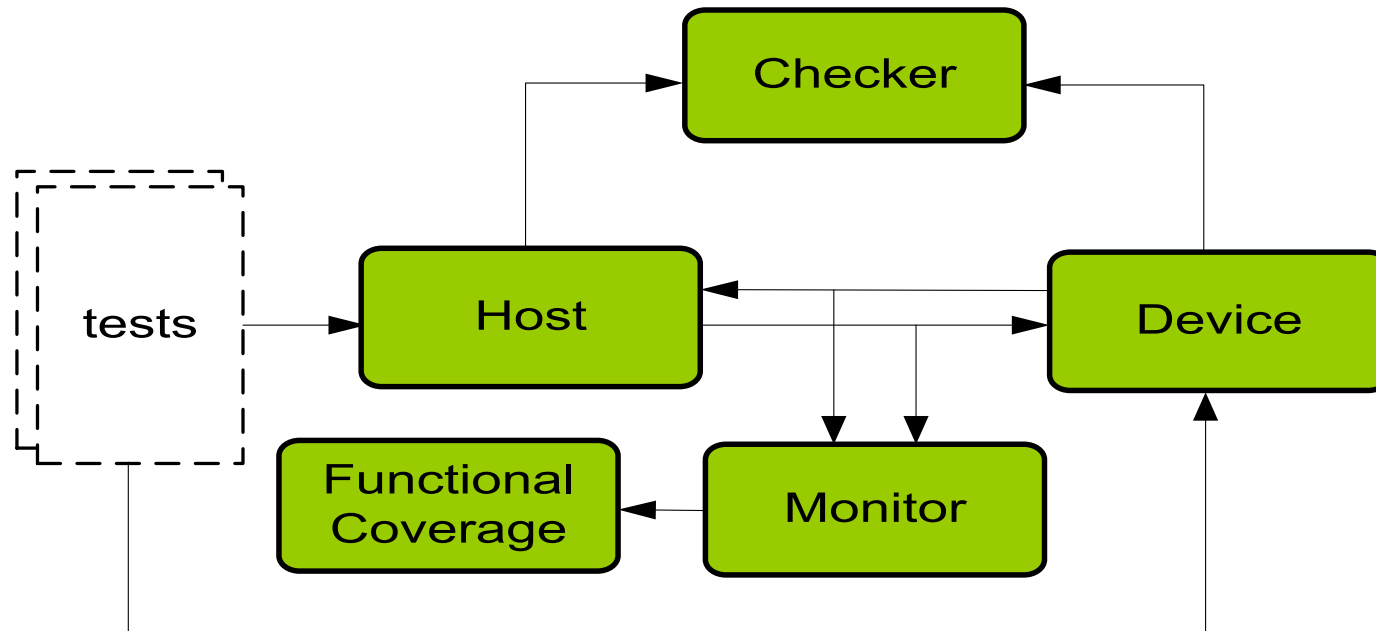
- Supports various data-rates/speed from 19.2 Gbps to 31.875 Gbps
- Supports for Lane Reversals, Degraded Mode and Deskew Markers
- Virtual Channel (VC) and Data Credit Pool (DCP) supports
- Optional enablement of scrambler/descrambler
- 64B/66B line encoding and decoding
- Constraint Random user chosen Methodology with:
 - Error injections: scrambler , CRC, framing, training pattern, protocol errors
 - Callbacks in Host, Device and Monitor for user processing of data
 - Complete test suite
 - Functional Coverage Model
 - Scoreboard

Developing the VIP(s): SmartCompiler

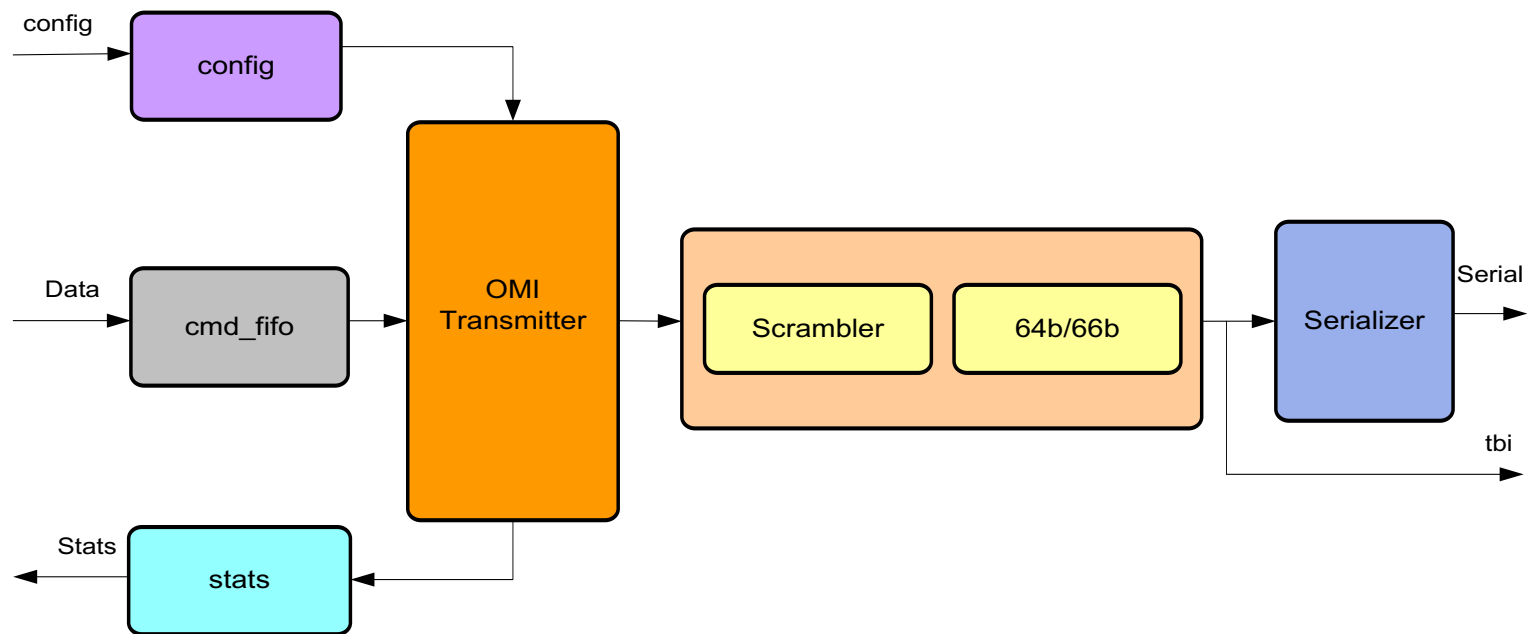
- **High quality products delivered by SmartCompiler automated flow**
- SmartCompiler produces high quality code
 - Proprietary high-level language with parameter-controlled output
 - Uniform code quality instead of varying quality from manual development
 - Automation lowers cost and turnaround time
 - Fast delivery of release updates
- SmartCompiler is proven with a 14+ year track record of delivering a vast, high-quality product line



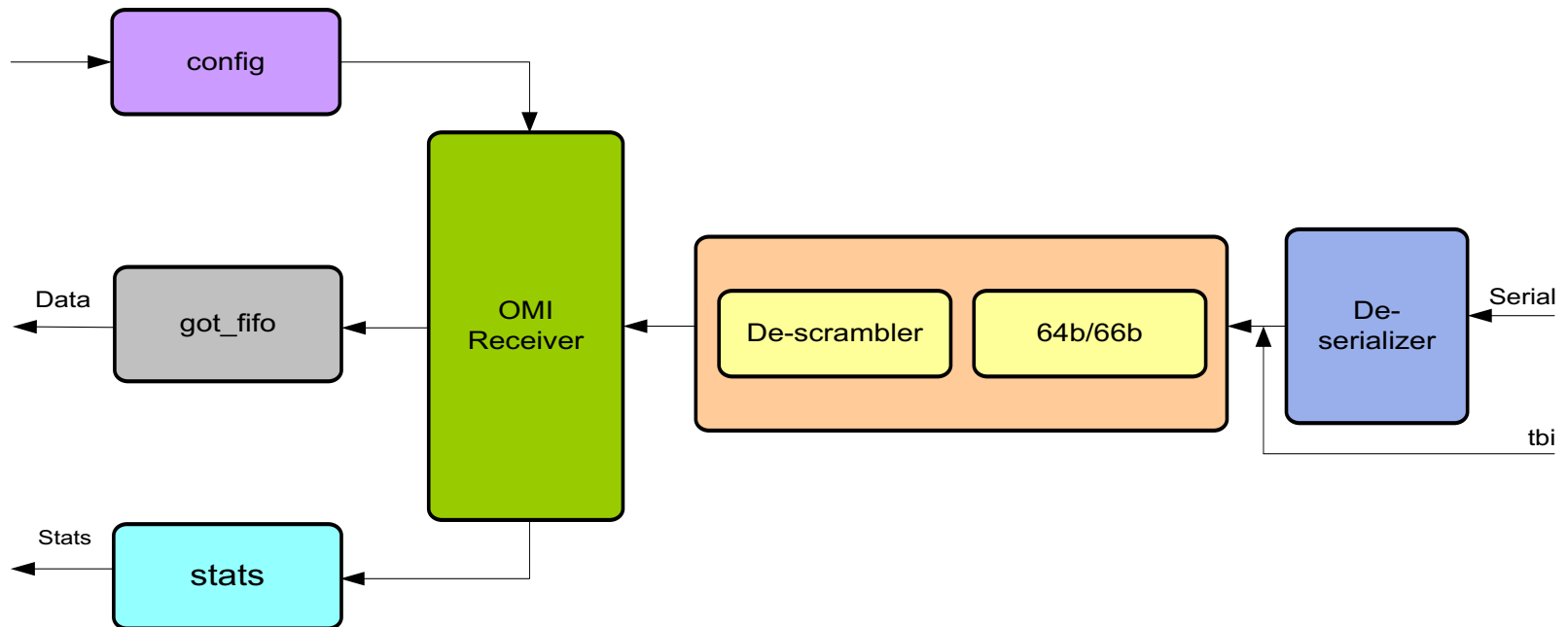
VIP Block Diagram



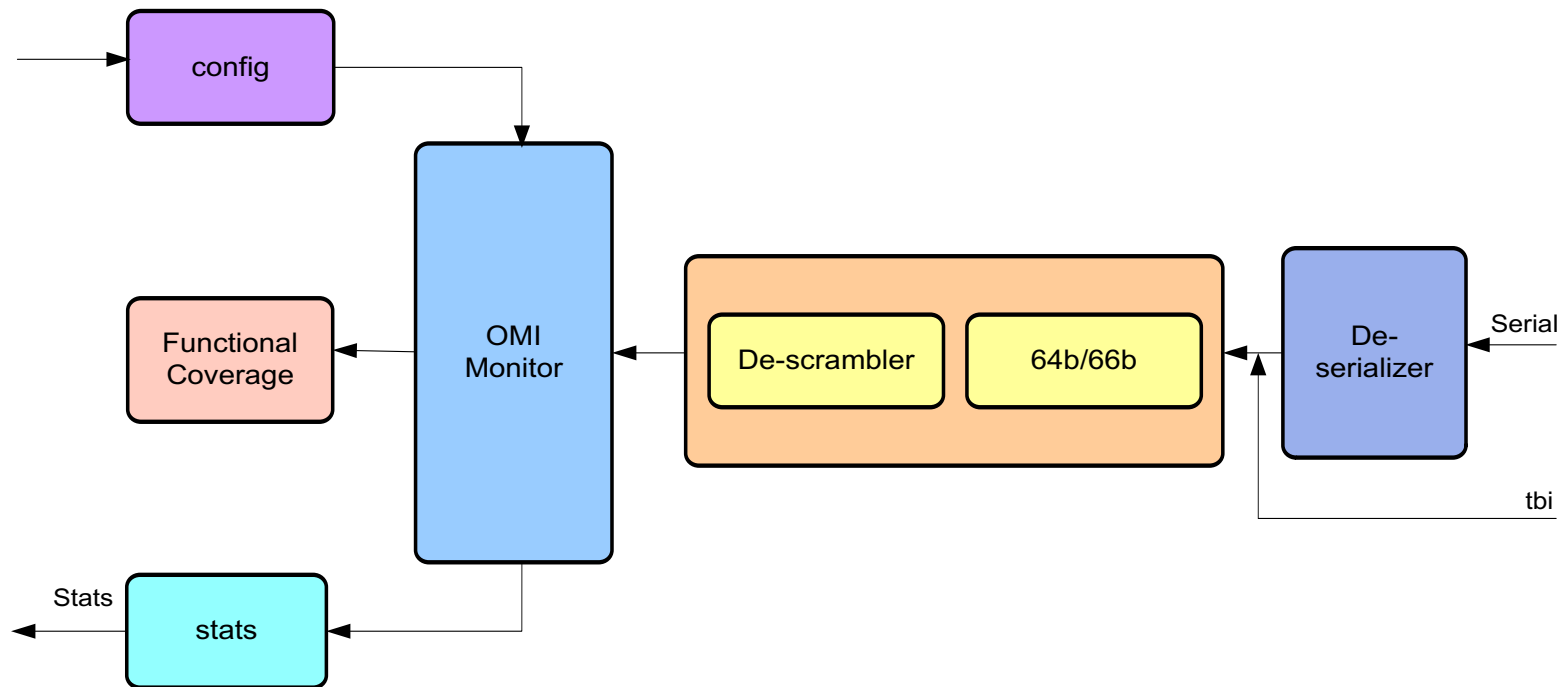
Transmitter Architecture



Receiver Architecture



Monitor Architecture



Key Benefits

- ❖ Standard Proven Methodology: Faster VIP integration.
- ❖ Wrapper-less native target methodology implementation
 - ❖ UVM, SystemVerilog, SystemC and on on - user selectable.
- ❖ VIP comes with complete sequence library of 1000+ test sequences.
- ❖ Reusable examples/sequences
 - ❖ Readymade regression-suite.
- ❖ VIP comes with functional coverage model and scoreboard.
 - ❖ SmartDV R&D helps user directly to integrate and modify for user environment
- ❖ VIP provides various debug ports which can be viewed in waveform for faster debug of RTL/VIP Testcase.
- ❖ Transaction logging feature in VIP can be used for logging all command transfers (Sent and Received) from VIP into separate log in tabular form in nice human readable form.
- ❖ Performance reporting to analyze the efficiency of the protocol.

Configurations

- ❖ Expansive set of configurations for control of VIP operations.
 - ❖ Configurations set required values to control parameters.
- ❖ Modify the VIP BFM's default configs by with two methods,
 - `configure()` :: Global configuration
 - `set_config()` :: Configuration for specific BFM.
- ❖ Example:
 - **SDVT_OMI_CFG_LANES**
 - Minimum value : 1
 - Maximum value : 32
 - Default value : 8

Full lists are documented in VIP user-guide.
- ❖ VIP methodology implementation is not encrypted – this makes it easier for user to understand and use the testbenches better

Callbacks

- ❖ Callbacks are supported in the following VIP components:
 - ✓ Transmitter
 - ✓ Receiver
- ❖ Callbacks get access to the base object (`sdvt_omi_data_t`)
- ❖ Callbacks can be used for:
 - ✓ Dropping a transaction
 - ✓ Error injection
 - ✓ Delay insertion
 - ✓ Modify command/response fields before they are driven onto the bus
- ❖ Important callbacks include:
 - ✓ Callbacks on flits transmission and reception
 - ✓ Callbacks for packet dropping, CRC error injection

Error Injection

- ❖ Error injection is supported in the following VIP components.
 - ✓ Transmitter
 - ✓ Receiver
- ❖ Error can be easily injected per transaction/per beat. Error injection in our VIP can be done in 3 different ways.
 - ✓ Using set_error method
 - ✓ Using set_cmd_field method (directly setting illegal values to fields)
 - ✓ Using callbacks
- ❖ Transmitter/Receiver/Monitor also check for the correctness of the injected error value/index against the error to be injected.
- ❖ VIP supports disable_error_msg method – helps writing negative testcases.

OVM/UVM Sequences

❖ Transmitter sequences

- 1) Sequence For Random data generation.
- 2) Sequence For User data generation.
- 3) Sequence for Loading data from file.
- 4) Configuration, Callbacks, Error injection .

❖ Receiver sequences

- 1) Sequence for idle flit.
- 2) Configuration, Callbacks, Error injection
- 3) Sequence for wait command done

Q/A