Advantages of using UVM/System Verilog IEEE standards to Verify Complex Probabilistic Constellation Shaping Design for a Coherent DSP ASIC.

Nipun Bhatt

Infinera Corporation 140 Caspian Court, Sunnyvale, CA 94089 Phone: 408-543-7485

Email: nbhatt@infinera.com

INTRODUCTION

Superior optical network system performance of the 800G can be achieved using advanced Probabilistic Constellation Shaping(PCS) algorithms and design techniques. Verification of Probabilistic constellation shaping design is very critical to get the desired system performance where verification methodology plays a vital role.

System Verilog[SV] and UVM provide various mechanisms to re-use the verification components - Configuration classes, SV-Interfaces, SV-DPI, UVM Sequences, UVM Agents, UVM TLM Interfaces and many more. The objective is to demonstrate the effective use of these features in creating an environment which mimics the design hierarchy. Furthermore, the reusability of verification environments (from module level to sub2chip level, to sub-chip level, to full chip level testbench) has also been touched upon. The effectiveness of the testing mechanism to verify a complex Probabilistic Constellation Shaping [PCS] Algorithm in a high-performance Coherent DSP ASIC and the key advantages over traditional verification methodologies have also been highlighted.

DESIGN COMPLEXITY

High-performance Coherent DSPs are powered by complex algorithms, one of which is a Probabilistic Constellation Shaping Encoding and Decoding algorithm(Data Distribution Layer). A very high-level optical transceiver data path is showed in *Figure 1*. It shows how the functional layers are divided into different categories.

PCS feature highlights

- Subcarrier level bandwidth distribution.
- Configuration based scheduling of client traffic.
- Symbol level Encoding and Decoding(zero-tolerance Decoding algorithm)
- Parallel processing using several Encoder and Decoder. Throughput and longer codewords drive parallel processing.
- Importance of latency to recreate the subcarrier level traffic.
- Interleave and deinterleave functions and the symbol level communication with Encoder and Decoder blocks to form the boundaries.

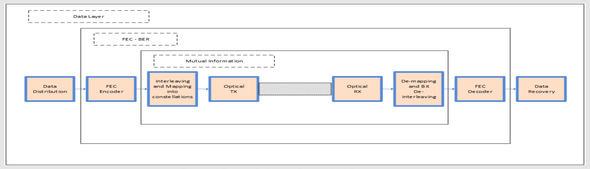

Figure 1: Optical Coherent Transceiver Datapath

TRADITIONAL VERIFICATION METHODOLOGY AND DRAWBACKS

DSP models are written and verified in MATLAB and can be converted to C-models that can be used for verification. Using C-models to verify algorithm-specific RTL design is very popular due to the ease with which C-model can be integrated into the System Verilog testbench using DPI import and deployed for targeted block-level testing, making it an ideal solution. Functions are implemented in C and given import declaration in the System Verilog module (*Figure 2*). However, simplicity comes at the cost of scalability and reusability.

The shortcomings are as follows.

1. Simple System Verilog module with DPI import can verify the module level design but is not very flexible to be integrated at a higher level of testbench hierarchy compared to UVM agent/environment integration.
2. They are not very effective when parallel processing is a requirement. Especially with multiple instances of algorithm-specific blocks with each block working on a different configuration(Encoder and Decoder slice instances *Figure 5* and *Figure 6*). The test is overloaded with managing configurations of multiple SV modules and the instance hierarchy to program the C-models compared to the efficient config_db approach.
3. System Verilog module with DPI import is highly inefficient when communicating with upper or lower-level environment components compared to TLM ports which provide great flexibility modeling layered protocol.

One can make the argument about having an end to end C model instead of using SV and UVM based methodology and fix all the issues highlighted above but, that too has its limitations.

4. Configuration intensive design with a constraint random verification is not possible using only end-to-end C-models.
5. An end-to-end C-model approach is inefficient for self-checking tests. They cannot provide flexible debugging hooks.
6. The effort and work we put in at module level are not scalable and reusable without more toil at a higher level which has timeline or resource implications.

Figure 2:System Verilog Module DPI Testbench for Algorithm Specific Design

DETAILED DESCRIPTION OF VERIFICATION INFRASTRUCTURE

The development effort was focused around four building blocks: pillars of the PCS environment which mimics the design implementation and hierarchy[3].

1. Scheduler Agent (*Algorithm is implemented as a reference model*)
2. Slice level Encoder Agent (*Algorithm is implemented in C-model*)
3. Slice level Decoder Agent (*Algorithm is implemented in C-model*)
4. Interleave/De-interleave Agent (*Algorithm is implemented as a reference model*)

Algorithm specific data integrity for Encoder and Decoder is maintained at slice level UVM agents by integrated system-level C-models using DPI import on the driving and monitoring paths. There are several instances of Encoder and Decoder slice level Agents as part of PCS sub2chip (*Figure 5* and *Figure 6*), receive sub-chip and full-chip environment. The number of PCS instances increases manifold as we go up the testbench hierarchy.

PCS Encoder and Decoder slice level testbench were developed to verify bit rates, as high as 800Gbps for a few supported Giga baud rates. Configuration class accepts the high-level user-programmable ***configurations***

1. Bit rate and
2. Baud rate
3. Datatype

It derives all the other low-level configuration needed to configure the C-model and RTL in PS Encoder and Decoder Slice. *It is very critical to have each slice work independently on its configuration to support the subcarrier level bandwidth distribution requirement.*

The scheduling algorithm is implemented in System Verilog reference model and integrated as a part of Scheduler UVM Agent (*Figure 3*) to create a predictor model. A key feature of the Scheduler Agent is to distribute and recreate the data using subcarrier level bandwidth requirements.

On the transmit path (*Figure 5*), it distributes subcarrier level data to many PCS Encoder slices periodically so that they can work in parallel on the longer codewords. On the receive path (*Figure 6*), it gathers the data from many PCS Decoder slices periodically based on the subcarrier level bandwidth requirements and creates the original client data. Incoming data to the reference model is stored in "queue" data structures, which continuously builds and shrinks in size on both receive and transmit directions.

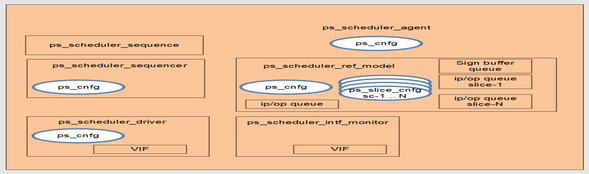

Figure 3: PCS Scheduler Agent

De-interleave(or interleave) UVM Agent (*Figure 4*) uses the design configuration, to achieve the layered testbench structure which mirrors the design hierarchy. The main function is implemented as a reference model working on the design configuration. Interleave/De-interleave Agent communicates to several slice level Agents.

On the transmit path, De-interleave Agent performs interleaving on the Encoded data coming out of several PCS Encoder slices and creates the boundary. Interleaved data is stored in such a way that I (In-phase) and Q (Quadrature) bits remain together when the symbols are recovered. Each constellation point is represented with the number of bits depending on the type of modulation. The bits which belong to different constellation points are put together by PCS interleave block.

On the Receive path, the main function of the De-interleave Agent is to validate the markers and start de-interleaving the bits to distribute through Decoder slice level Agents. With FEC in the data-path, we have considered that the data going through the Decoder slices is error-free and the generated I (in-phase) and Q (Quadrature) bits are recovered together for the decoding algorithm to work. Like in the scheduler Agent, the data is stored in the "queue" structures which periodically builds and shrinks in size.

The order of the bits is very important to recreate the client data. Heavy use of configurations, virtual interfaces, and transaction-level modeling helped achieve effective communication between the UVM Agents and seamless data-path monitoring.

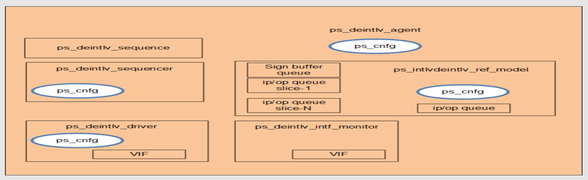

Figure 4: PCS De-interleave Agent

DETAILED DESCRIPTION OF VERIFICATION INFRASTRUCTURE

PCS sub2chip level environment integrated one instance of Scheduler Agent, several Encoder and Decoder slice Agents, and one instance of De-interleave Agent (*Figure 5* and *Figure 6*). A closer look at both test-benches gives clarity on the emphasis of the component reuse. No component is designed for any specific environment or path. UVM TLM interface provides key support in communication between the components in both transmit and receive direction for layered UVM environments.

Our *unique approach* of integrating complete transmit predictor UVM components as part of the receive path Deinterleave sequence would not be possible without UVM/System Verilog based environments. Due to the reference model implementation in both Scheduler and Deinterleave (or interleave) agents. It is very convenient to mimic the complete transmit hierarchy using UVM TLM ports.

In the connect phase, Deinterleave (or interleave) sequencer is connected to the scheduler reference model to distribute the payload data to multiple PCS Encoder slice monitors. Encoded data is transferred from various slices to the Deinterleave (or interleave) reference model where it performs the interleaving function and makes the data available in the Interleave Queue which is part of the Deinterleave sequencer. Data from the sequencer queue is eventually used by Deinterleave (or interleave) driver to pull the data from the sequencer and drive on the interface.

On the monitoring path, Deinterleave (or interleave) agent is used to collect the data from the interface and deinterleave the data using the reference model. Deinterleaved data is distributed to many decoder slice monitors where it goes through the C-models to produce the decoded output, which is sent back scheduler agent to recreate the client data.

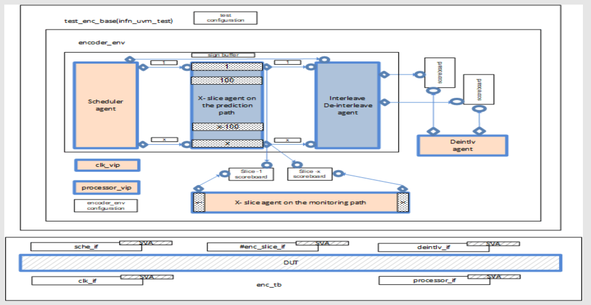

Figure 5: Transmit PCS sub2chip

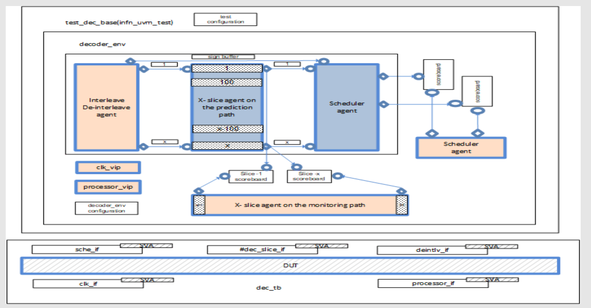

Figure 6: Receive PCS sub2chip

DETAILED DESCRIPTION OF ADVANTAGES

The advantages are as follows.

1. Major Advantage is the *reusability* of the UVM/System Verilog based verification infrastructure. UVM Agents and Environments are reused from module level to sub2chip level, to sub-chip level, to full-chip level verification. Additionally, reuse of all the UVM components on the transmit side and receive side of the testbench using the environment configurations (*Table 1*).
2. PCS Algorithm standpoint the key is the codeword, especially how long the codeword is [4]. The common configuration gives us *flexibility* on both Encoder and Decoder slice level Agents to have a variety of configurations on various slices in use and parallel data processing. Few *higher-level configurations generate all the lower-level configurations parameters* needed for design and C models.
3. Using *UVM set/get config_db()* mechanism all the components were programed correctly with a high level of flexibility which is one of the major advantages over traditional verification methodology. For full-chip level verification, all the configurations are used by *overwriting inline constraints*.

DETAILED DESCRIPTION OF ADVANTAGES

4. To make use of *Transaction Level Modeling*, the interface monitor is decoupled from the protocol monitor which helped us integrate protocol monitors in the layered environments where data is processed through the C-models to generate the expected output. In the module level environment, monitors prepare the expected transaction for scoreboard compare while in sub2chip or chip-level environments it creates the transaction for the higher layer use.
5. *Reuse of all the extended test sequences and tests*. only one unique sequence is needed for Encoder and Decoder slice level data and sub2chip verification.
6. Test structured is very much simplified to have all the *script generated directed tests* based on the baud rate and bit rate requirements and shared between Encoder and Decoder slice level and sub2chip level testing.
7. *Integration of complete transmit environment hierarchy* as part of the receive test sequence. We could leverage *automated data checking* at the boundary of each agent.
8. *Another major advantage during the debugging*, transaction-level Encoder and Decoder output data is always available to get compared by enabling scoreboards tied at the boundaries of Encoder and Decoder slice level agents. In a nutshell, all the debugging hooks available at the component level are used as-is. This *accelerates the whole data-path bring up effort.*
9. *Functional coverage on configuration and scenarios* were also very simplified with UVM/System Verilog based environment.
10. *Environment* developed for PCS sub2chip in both transmit and receive directions are completely *reused on both receive and transmit sub-chip level testbench.*
11. *Latency measurement is simplified* by porting monitors at the interface to measure the time taken for any transaction from a given source to destination path at all the level of testbench hierarchy.
12. *Seamless integration of multiple instances* is only possible due to our selection of UVM/System Verilog based methodology.

Table 1: Reuse of UVM Objects, Components and Sequences

UVM Components	Module Level	Sub2chip level	Sub-chip level	Full-chip level	Receive / Transmit
Configurations	✓	✓	✓	✓	✓
Monitors	✓	✓	✓	✓	✓
Sequences	✓	✓	✓	✓	
Adaptive Sequences		✓	✓	✓	
Scoreboards	✓	✓	✓	✓	✓
Agent on receive and transmit	✓	✓	✓	✓	✓
Environments		✓	✓	✓	

SUMMARY

Traditionally, DSP algorithms are verified using MATLAB models and C-models. Our decision to verify the complex design blocks using the UVM/System Verilog IEEE standards capabilities helped us reuse all the UVM Environments, Agents, Configurations, Virtual Interfaces and Sequences at multiple level of verification hierarchy. We created a common configuration class following constraint random methodologies, which was shared between Encoder and Decoder slice level environment. In like manner, a common configuration was used at the sub2chip level for receive and transmit path. We created only one unique base sequence for each PCS sub2chip on transmit and receive direction and all the extended tests were reused between PCS Encoder and Decoder sub2chip verification.

Having C-models integrated into part of the slice Agent eliminated any symbol level encoding and decoding issues on the monitoring path. Virtual Interfaces helped us achieve the effective integration of monitors at each level of the testbench hierarchy.

Implementing a reusable UVM/System Verilog based testbench environment helped achieve the verification goals on time. Contribution towards the sub2chip level verification efforts can be evaluated by measuring the reuse of test stimulus on both receive and transmit direction. It has immensely helped to verify configuration heavy design block like PCS. Leveraging the environment components, we could exercise the system level scenarios at the full-chip level without changing any sub-chip level environment code. It has aided to close the coverage with almost 100% coverage goals. In a nutshell, reusable environment components validated bit-level integrity in both transmit and receive directions for the PCS sub2chip and at full-chip level. Performance and Error recovery were the other two critical test scenarios effectively verified.

REFERENCES

[1] 1800.2-2017 - IEEE Standard for Universal Verification Methodology Language Reference Manual

[2] 1800-2017 - IEEE Standard for System Verilog--Unified Hardware Design, Specification, and Verification Language

[3] Infinera Proprietary Algorithms – M Torbatian, D Chan, HH Sun, S Thomson, KT Wu - *US Patent App. 16/152,349, 2019*

[4] www.infinera.com/blog/long-codewords-the-secret-to-successful-probabilistic-constellation-shaping

[5] www.infinera.com/white-paper/The-Ultimate-Guide-to-Nyquist-Subcarriers