Customizing UVM Agent Supporting Multi-Layered & TDM Protocols

Amit Pessach , amitp@veriests.com

Veriest

accellera SYSTEMS INITIATIVE

2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

Methodology Development & Problem statement

Agent – Basic Principles

Multi-Layered Architecture

Channelized Agent

Summary

Methodology development & Problem statement

- Verification complexity is growing along the years.

- Many languages and methodologies were introduced in the past such as E/eRM/URM , Vera/RVM , SV/VMM , SV/AVM/OVM , etc…
  Causing Vendor competitions and non-uniformed methods in the industry.

- SV & UVM were standardized and provide very good infrastructure for any verification environment keeping on the main principles of modularity, reuse and easy maintenance **but**:
  - A lot of “behind the scenes” in standard’s libraries.
  - Methodology & implementation holes still exist.

Problem Statement

For complex usage, UVM API is not always sufficient causing non efficient, time consuming, non reusable and hard for maintenance implementations.

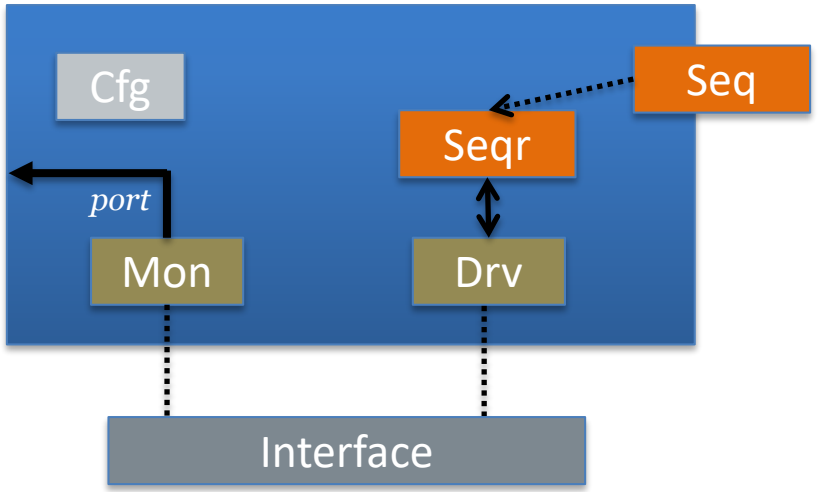
Getting to the “right” solution for a specific problem depends on many factors.

UVM infrastructure awareness and deep understanding is one of these.

During this presentation will focus on couple of usage cases examples for demonstrating the above.

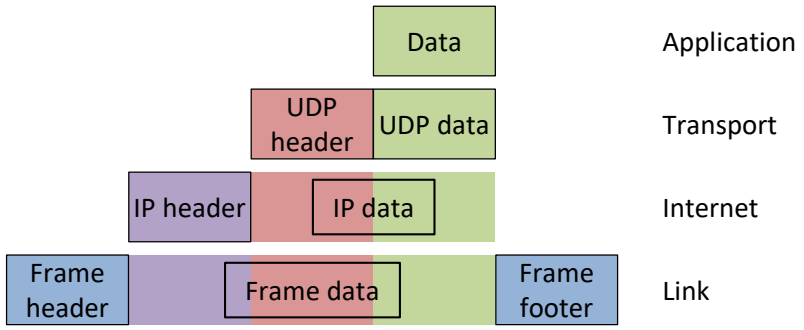
UVM Agent - Basic Principles

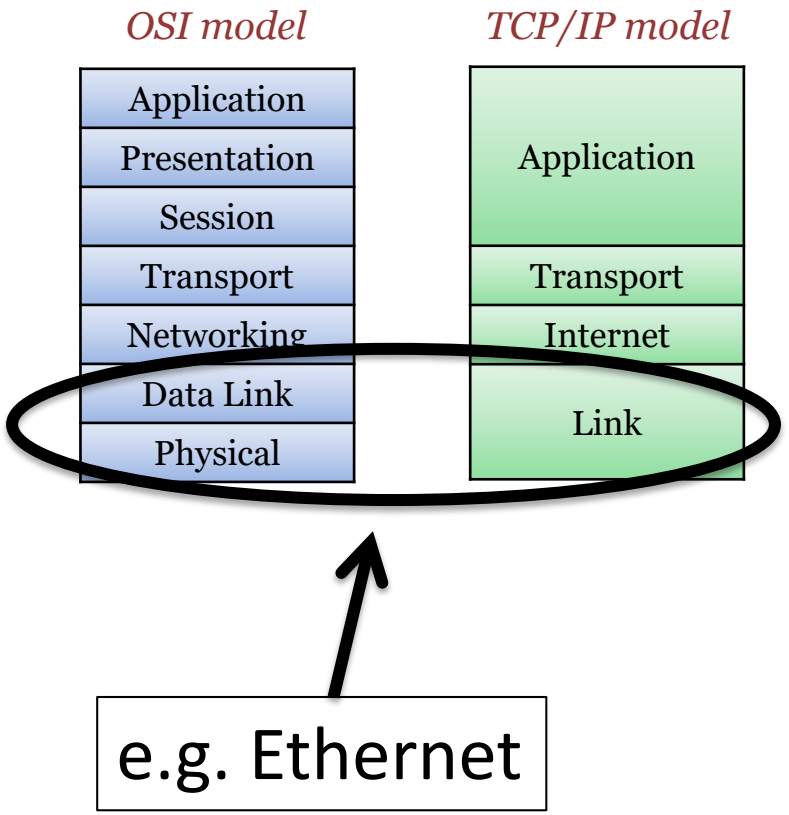
- Handling interaction with a **single** DUT interface
- Supports both initiator/master and responder/slave sides
- **Must not be DUT-biased**
- **Simple** but still supporting **full** interface view
- Provides **protocol** related checks **only** (including **relevant** data checks)
- Easy to integrate in all levels (vertical & horizontal reuse)



Multi-Layered Architecture – what is it all about ?

- Most common used network protocols enable layering
- Allows diverse systems to communicate with each other
- Each layer handles encapsulated data relevant to its functionality only





Multi-Layered Architecture – "Common Approach"

- Single agent handling single data structure with control fields for all layer's behavior.

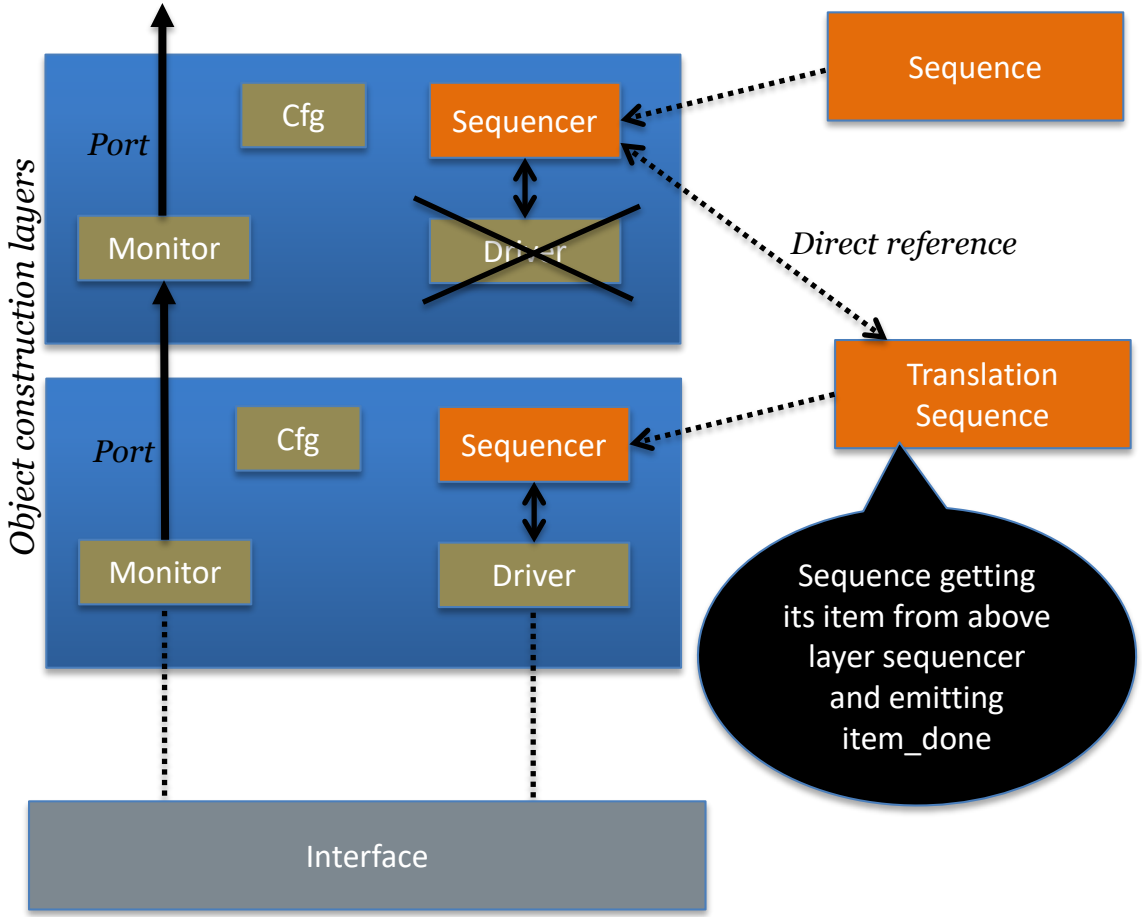
  Might not be good enough …

- Difficult to control lower layer behavior from high layer data structures.
- Difficult to focus on single layer testing in case required.
- Poor controllability on relations between 'objects' of different layers.

Approach is not flexible enough to achieve high quality verification abilities…

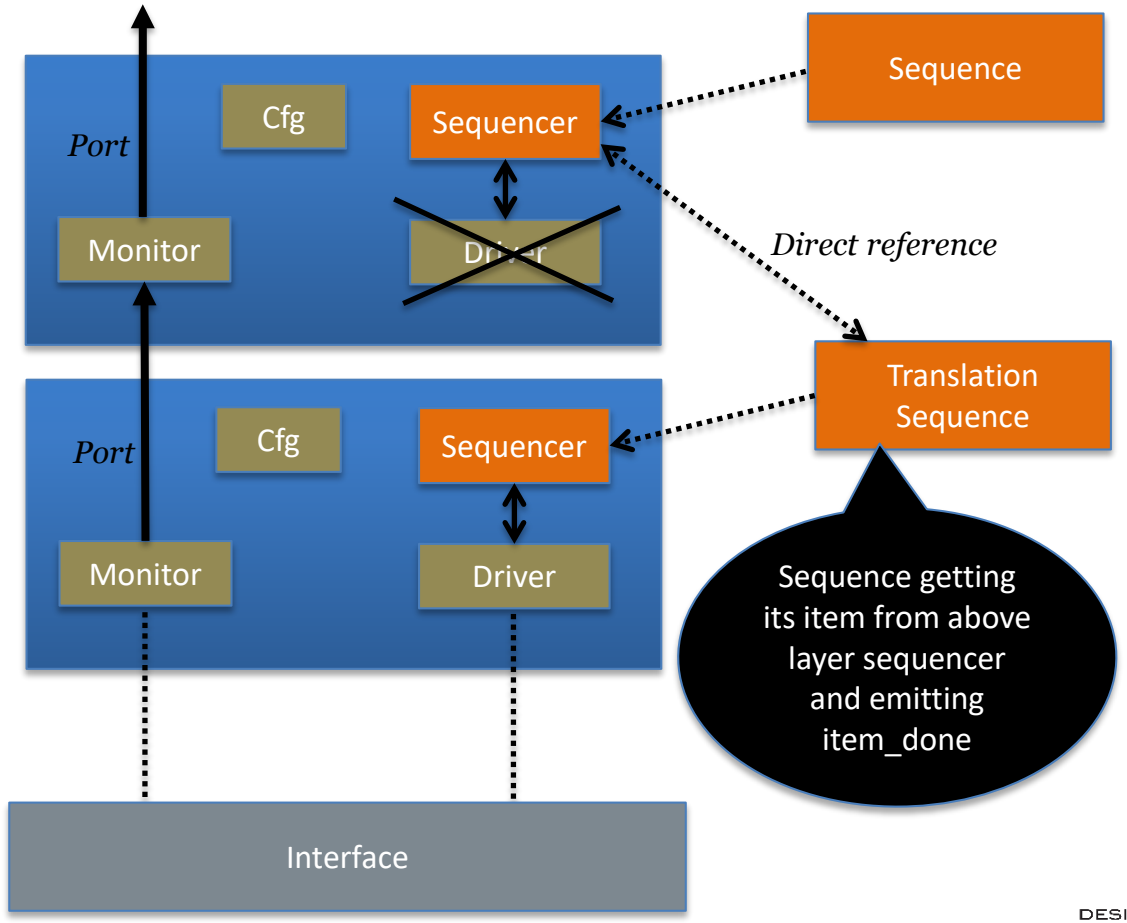
Multi-Layered Architecture

- Layers modeled with layered agents
- Each agent unaware regarding implementation of other agents
- Separate transaction object & translation sequence for each layer
- Higher layers has no driver & physical interface
- Modular object construction
- Full control of each layer



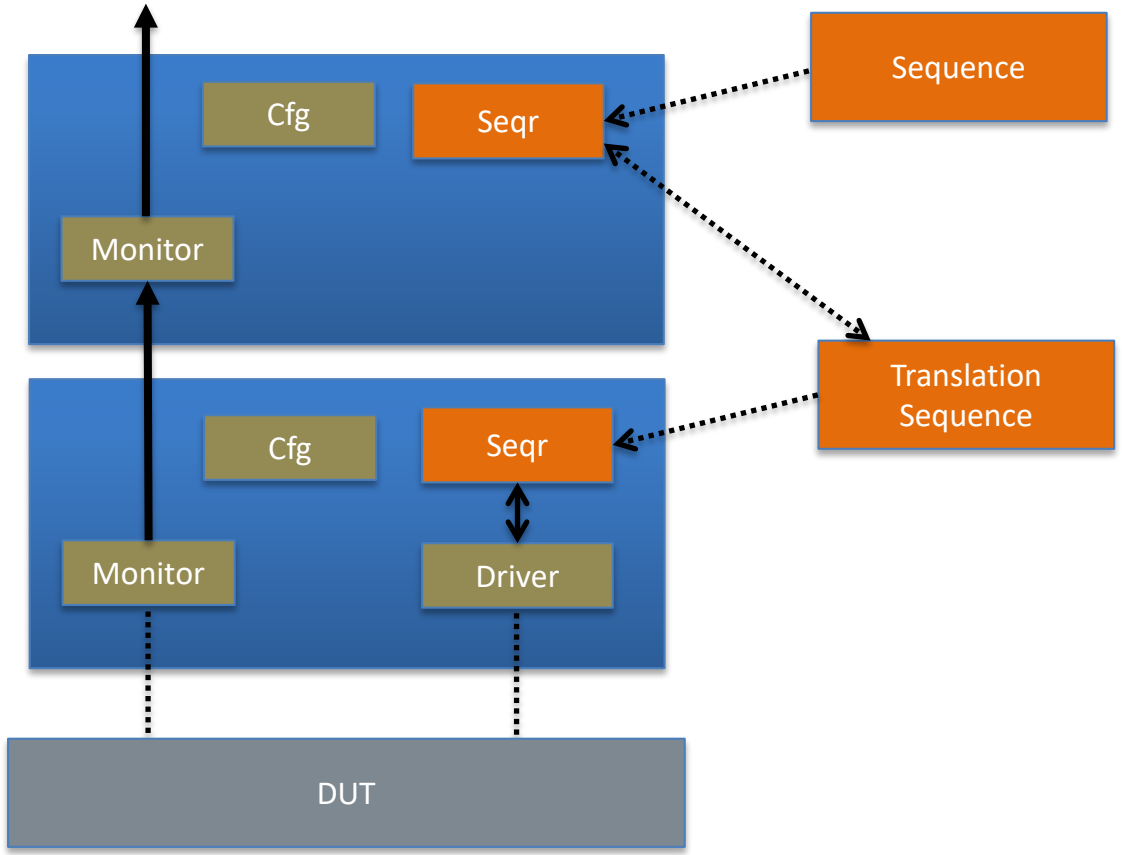
Multi Layered Architecture

- **Methodology holes:**
  - **Driver elimination not supported in ACTIVE agent…**
  - **Monitor concatenation**
  - **Physical interface association**



Multi-Layered Architecture Usage & Debug

- Controlling each layer's data item members separately.
- Tune control members in each layer's configuration object separately.
- Easily enabling/disabling checks of each layer.
- Controlling debug messages verbosity separately for each layer.



Channelized UVM Agent

- Multiple logical traffic flows with unique functionality over same physical interface.
- In more complex cases TDM physical protocol is applied.
- In many cases a simple interface agent exists already and “channelized” functionality needs to be added on top of it.
  Implementation should require minimal effort, clean, friendly & controllable.

Channelized Agent Usage Case #1

On top of existing packet interface, need to add functionality to support multiple traffic flows with user-defined priority for different flows

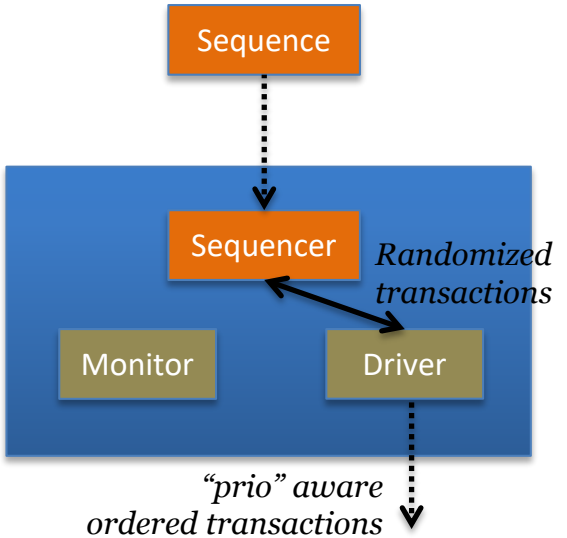
Each traffic flow associated with unique priority which should be reflected on the interface.

Question is how should such adjustment be implemented ?

Again there are many different implementations…
depends on user experience level and UVM infrastructure awareness

Channelizing – “Common” Approach

- Implementation fully controlled in the driver.
- Objects created with embedded priority indication
- Driver handles objects accordingly towards interface.
- Complexity pushed to lower level abstraction layer – generally not the preferred approach.
- Transaction object, driver, sequence and config need to be modified.
- Simple existing agent becomes much more complex, less user friendly and less reusable.



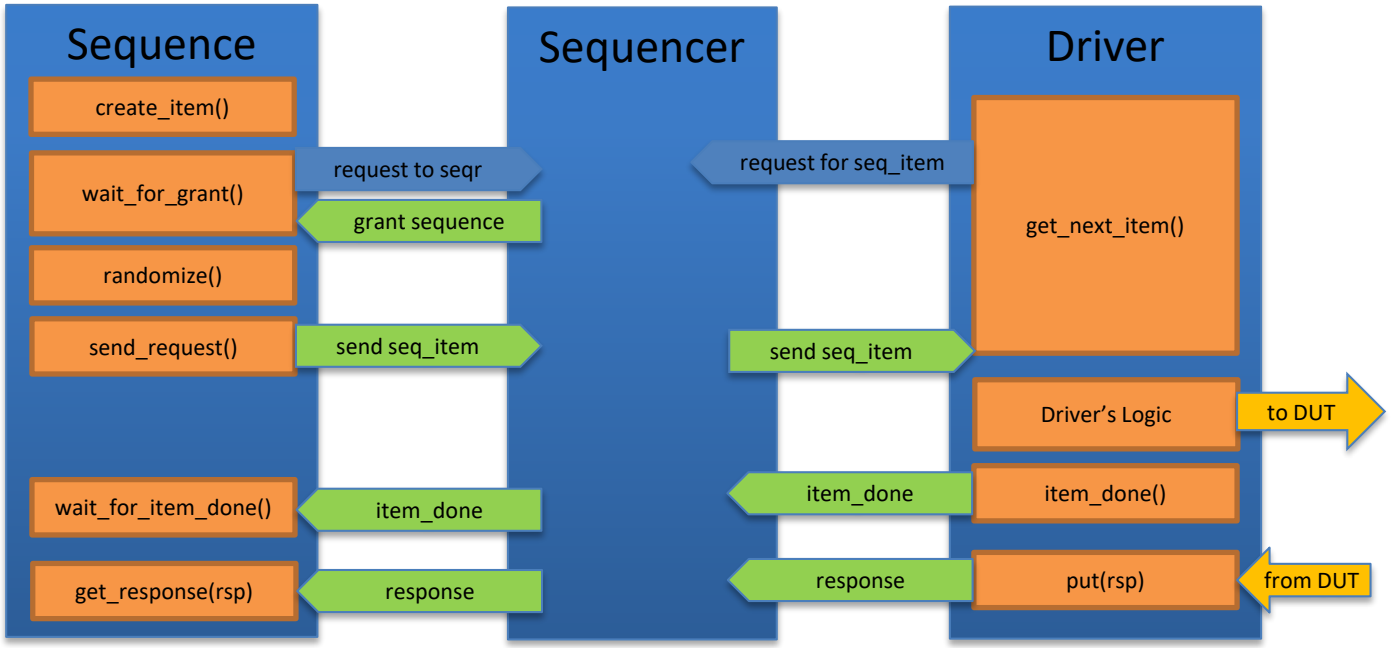
Channelizing – Suggested Approach

Cleaner, controllable and much easier implementation should make use of internal UVM capabilities.

In order to deeply understand suggested solution first need to understand how driver-sequencer-sequence “logical triangle” is executed when handling sequence items.

Driver-Sequencer-Sequence Triangle Theory

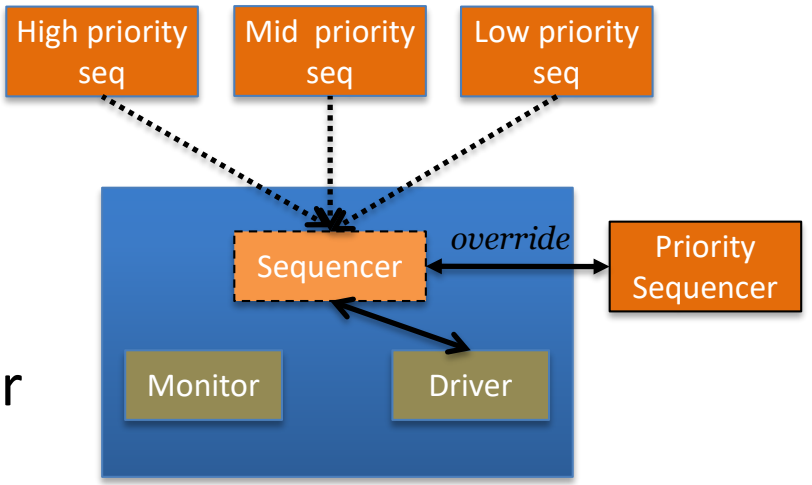
Handshake between agent's sequence, sequencer and driver is demonstrated below:



- Multiple sequences can run in parallel on the same sequencer.
- Arbitration logic within the sequencer selects available sequence to be granted and execute its sequence item.
- UVM supports different Sequence Arbitration mechanisms listed below:
  - `UVM_SEQ_ARB_FIFO`
  - `UVM_SEQ_ARB_WEIGHTED`
  - `UVM_SEQ_ARB_RANDOM`
  - `UVM_SEQ_ARB_STRICT_RANDOM`
  - `UVM_SEQ_ARB_STRICT_FIFO`
  - **`UVM_SEQ_ARB_USER`**

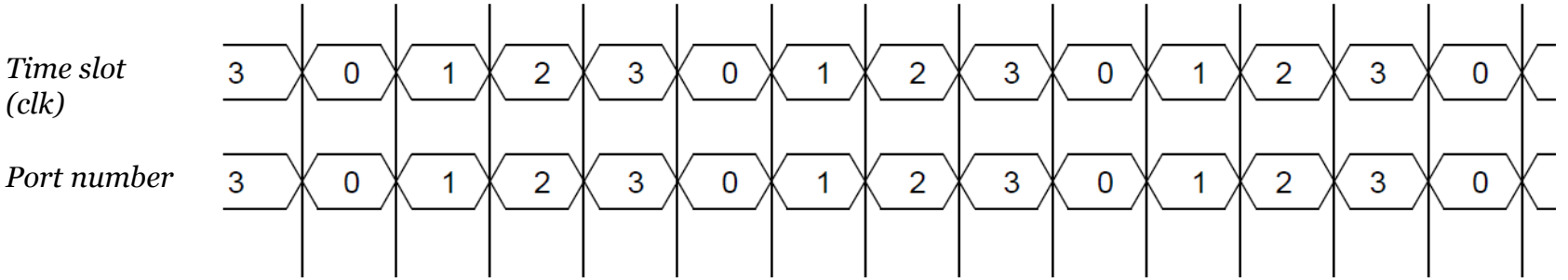
Channelized UVM Agent Suggested Approach

- Separate sequence running in parallel for each traffic priority level
- Required functionality achieved by selecting SEQ_ARB_USER mode and overriding sequencer arbitration behavior
- All modification done in **sequence** level
- Existing agent **untouched** and **unaware** of added priority awareness



TDM Channelized Agent Usage Case #2

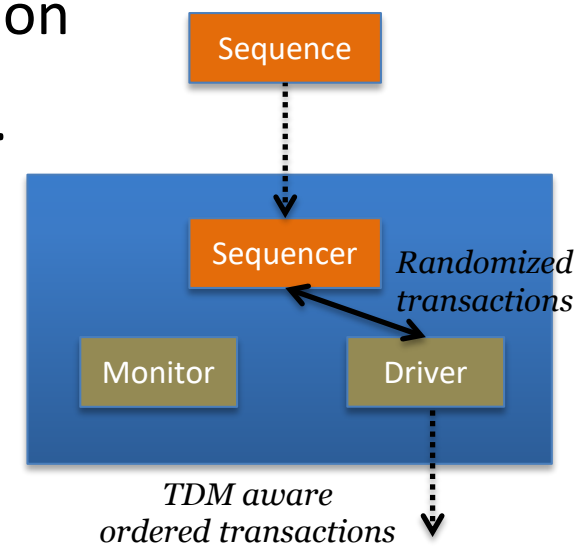
- Multiple traffic flows each physically active only during predefined time slot (TDM)
- Ports are sharing simple packet interface using Time Domain Multiplexing (TDM) technique
- Sharing done in Round Robin fashion as demonstrated below:



- TDM timing **must** preserved whether port has available data to send or not…

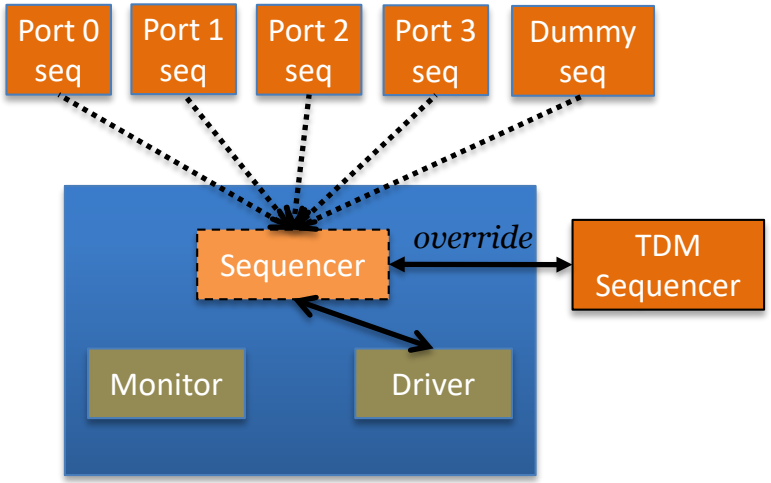
TDM UVM Agent – "Common" Approach

- Implementation fully controlled in the driver.
- Objects created with embedded channel/port number indication
- Driver maintains complex queuing and arbitration mechanism.
- Complexity pushed to lower level abstraction layer – generally not the preferred approach.
- Transaction object, driver, sequence and config need to be modified.
- Simple existing agent becomes much more complex, less user friendly and less reusable.



TDM Channelized Agent Suggested approach

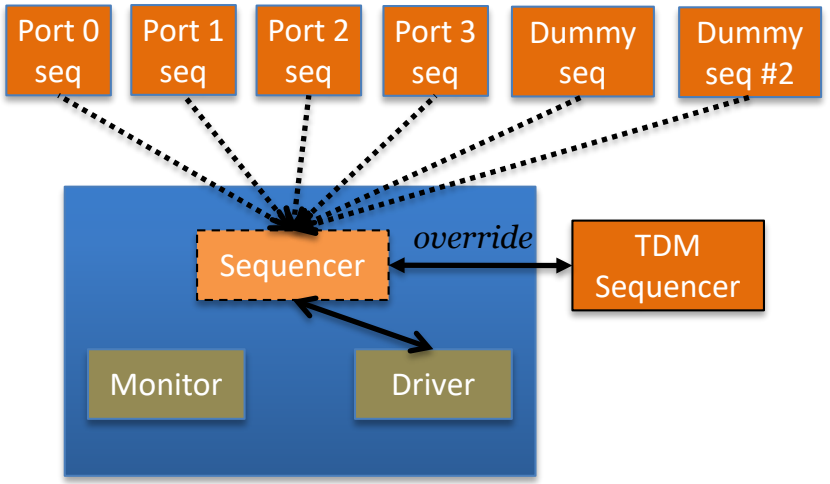
- Separate sequence running in parallel for each port
- Sequencer is overridden to achieve Round Robin arbitration scheme
- What if no packet available on a specific port?
- UVM Arbitration must give grant in case there is any available pending sequence.
  - **There is no option to "skip" arbitration and send notification to the driver of no sequence grant.**
  - **TDM sync will be lost**



- Bypassing this UVM "hole" by adding "dummy" sequence:
  - always available for arbitration
  - creates invalid item that driver will handle by "bubble" insertion

TDM Channelized Agent Suggested approach

- What if no packets available for any of the ports?
- Only “dummy” sequence has item available:
  - **In UVM, arbitration logic is not triggered in case of single available sequence.**
  - **TDM sync is lost**
- Adding second “dummy” sequence
  - ensure arbitration logic is triggered each cycle
  - sequencer always able to keep in sync to its TDM logic



Summary

- Standardization of verification language and associated methodology are crucial for providing common industrial infrastructure keeping on main principles of modularity, reuse and easy maintenance.
- UVM provides very good such infrastructure solution.
- Still UVM has holes and use cases which are not supported by simple and straight forward infrastructure or simple usage guidance.
- Infrastructure awareness will always lead for finding better implementation solutions.

Questions