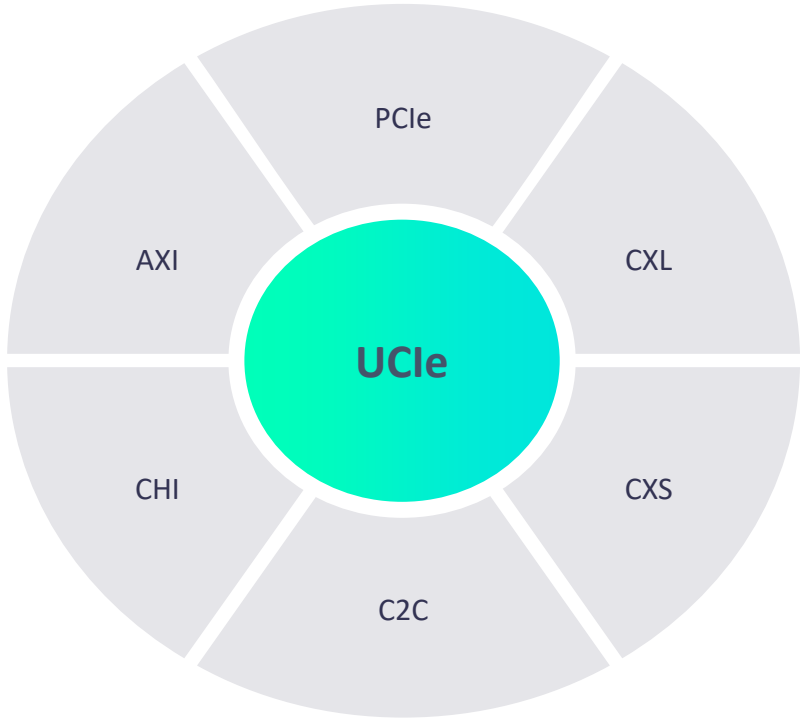
Versatile Verification framework for multi-protocol UCIe Design

Himani Kaushik, Siemens EDA

UCIe: The Protocol Playground

- An agnostic innovation to construct System-in-Package (SiP) with relevant application layer and design parameters making interoperability with externally sourced chiplets conceivable
- FDI and RDI enable flexible on-package integration over UCIe, allowing any protocol layer to drive traffic without restrictions.
- Diverse flit formats, including raw data, enabling broad protocol integration.
- UCIe Consortium introduced further support for protocols like JEDEC Standard No. 204E (JESD204E)
- It is reasonable to expect that UCIe will continue evolving to accommodate wider array of protocols
- Protocol integration verifies parameters for cross die communication that play a significant role in overall verification of the system like
  - synchronization discrepancies,
  - timing variations and clock consumption
  - Bandwidth and latency



What needs fixing?

Protocols like PCIe, CXL, CHI, AXI, JESD204E etc. cannot be broken down into entities that can drive UCIe interfaces directly. Interfaces like C2C and CXS cannot directly driver FDI or RDI interfaces. These protocols and interfaces exhibit significant diversity, sharing minimal or no commonality. There is an absence of an industry standard that lays down the interprotocol communication between Protocol layers and UCIe.

From Chaos to Cohesion : Versatile Adapter

Synchronization between protocol and UCIe state machine

For example, protocol layer initiated low power state is transmitted on UCIe and vice versa

Data abstraction by protocol layer to drive traffic

UCIe does not fill Flit payload. Meaningful data is encapsulated by Protocol layer and sent over UCIe

Deployment of multiple protocol layers, identical or distinct, on each UCIe stack

In case of multiple protocol stacks, multiple independent paths of communication can be setup.

Protocol-to-UCIe Adapter

Error escalation

Error escalated by UCIe reaches protocol layer for further action by protocol layer

Debug handling

Protocol flit tracking over UCIe is simplified with an existing mapping

Reusability of testbench and test suite

Existing VIP testbench can be used with minimal updates to run traffic using existing protocol as well as UCIe test suite

Versatile Adapter: Plan of action

Protocol agnostic adapter that can communicate with both Protocol VIP and UCIe VIP by using

- UVM **callbacks** and
- System Verilog **tasks**.

Adapter Class

Adapter class uvm_application_bfm that extends from protocol_device_bfm where protocol_device_bfm refers to a uvm_component that acts as the Bus Function Model (BFM) for protocol integrated over UCIe. This is done so that the adapter behaves as the application layer that can send and receive transactions to lower layer

```
class uvm_application_bfm extends protocol_device_bfm ;
        ...
        ...
endclass
```

UCIe device

Create handle to ucie_device_bfm where ucie_device_bfm refers to the BFM for UCIe device as shown in Figure 2. This is useful for connecting Protocol VIP with UCIe VIP.

```
ucie_device_bfm ucie_bfm;
```

Versatile Adapter

Transmission

Invoke a callback from protocol_device_bfm at the point where packet exits from the protocol layer. Add logic to create a UCIe compatible packet from the received Protocol compatible packet. Use UCIe VIP callback to inject the converted packet to ucie_device_bfm.

```
task send_traffic();
    protocol_packet packet;
    this.wait_protocol_callback("tx_packet_exit_point", packet);
    fork
        protocol_to_ucie_packet_conversion(packet);
    join_none
endtask
```

```
    this.inject_ucie_packet_callback("tx_packet_enter_point", ucie_flit);
endtask // protocol_to_ucie_packet_conversion
```

Reception

Invoke a callback from ucie_device_bfm at the point where packet exits from UCIe VIP. Add logic to create Protocol compatible packet from the received UCIe compatible packet. Use Protocol VIP callback to inject the converted packet to protocol_device_bfm.

```
task receive_traffic();
    ucie_flit flit;
        ucie_bfm.wait_ucie_callback("rx_flit_exit_point", flit);
    fork
        ucie_to_protocol_packet_conversion(flit);
    join_none
endtask
```

```
    this.inject_protocol_packet_callback("rx_packet_enter_point", packet);
endtask // ucie_to_protocol_packet_conversion
```

Versatile Adapter: Plan of action

- UVM Environment

Create a consolidated UVM environment that contains ucie_device_bfm and uvm_application_bfm and connect them in connect_phase. In the run_phase start the uvm_application_bfm and ucie_device_bfm.

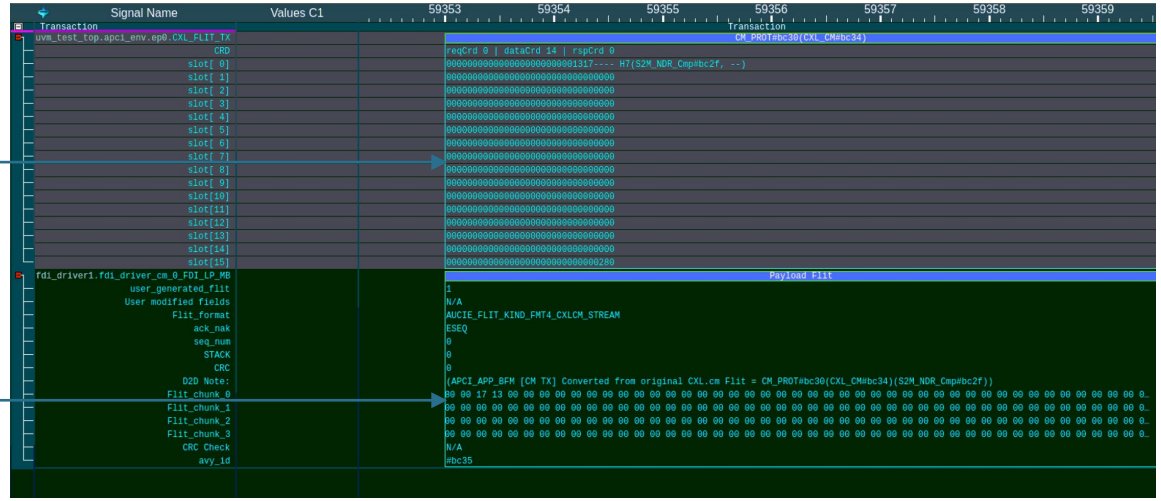
An existing testbench for Protocol VIP/ UCIe VIP can be modified so that both ucie_device_bfm and uvm_application_bfm can co-exist.

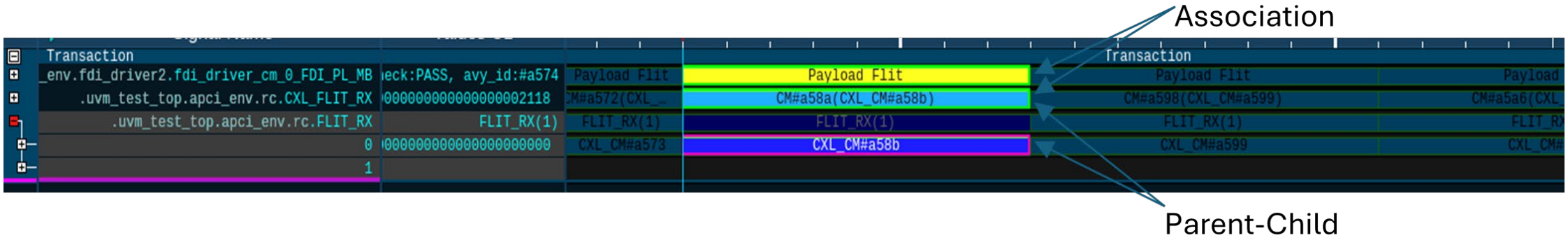
```
class protocol_ucie_uvm_env extends uvm_env;

    uvm_application_bfm  app_bfm;
    ucie_device_bfm ucie_bfm;


    function void build_phase(uvm_phase phase);
        app_bfm = uvm_application_bfm::type_id::create("app_bfm", this);
        ucie_bfm = ucie_device_bfm::type_id::create("ucie_bfm", this);
    endfunction

    function void connect_phase(uvm_phase phase);
        app_bfm.ucie_bfm = ucie_bfm;
    endfunction

    task run_phase(uvm_phase phase);
        ucie_bfm.init_run();
        app_bfm.init_run();
    endtask

endclass
```

Outcome: CXL over UCIe with Questa One illustration

Visual representation of a CXL Flit

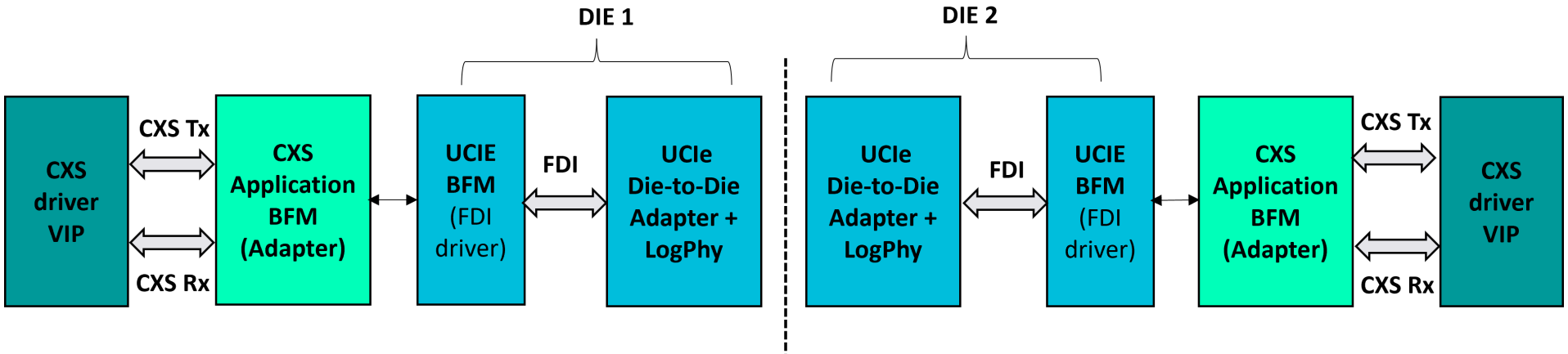
Visual representation of a UCIe Flit mapped to CXL Flit





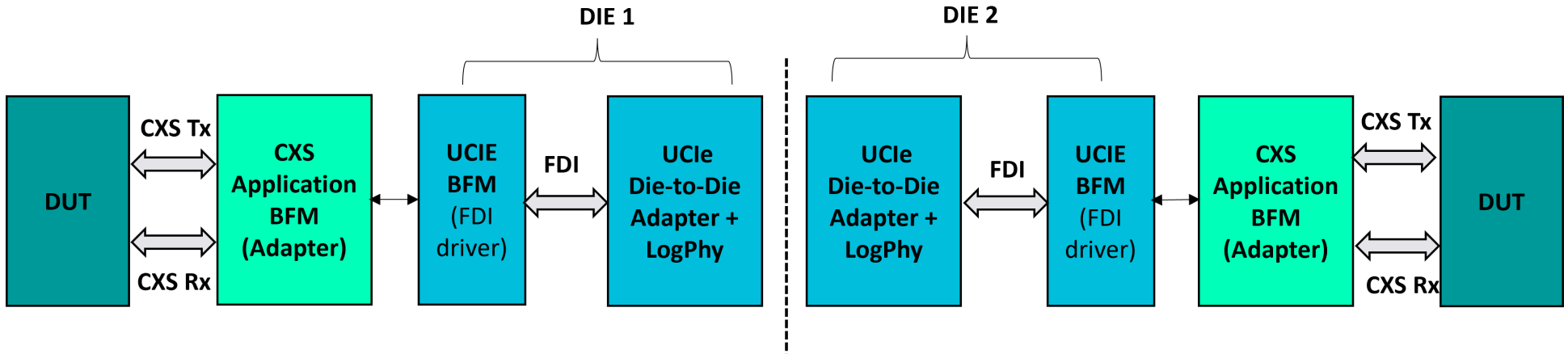
Case study 1 : Integrating Interfaces over UCIe

By employing the usage of a versatile adapter, a model can be generated that integrates interfaces such as C2C and CXS over UCIe VIP. This makes integration easier for protocol VIPs compatible with C2C and CXS over UCIe VIP. In the figure, adapter is extended from CXS BFM. It is connected through CXS interface with other CXS driver.



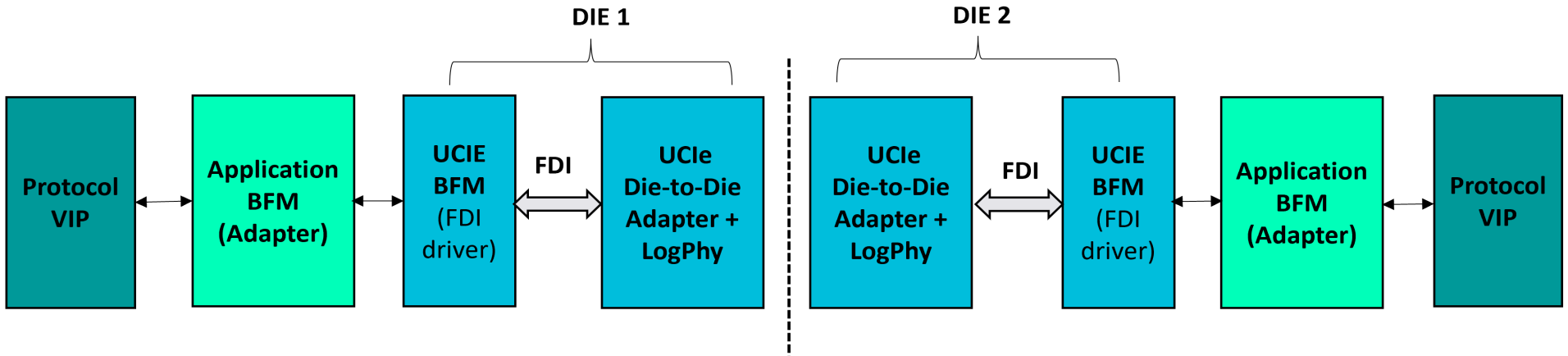
Case study 2 : Integrating Designs over UCIe

By utilizing the strategy to integrate interfaces like C2C and CXS over UCIe, it is feasible for design DUT to drive traffic over UCIe through interfaces C2C and CXS



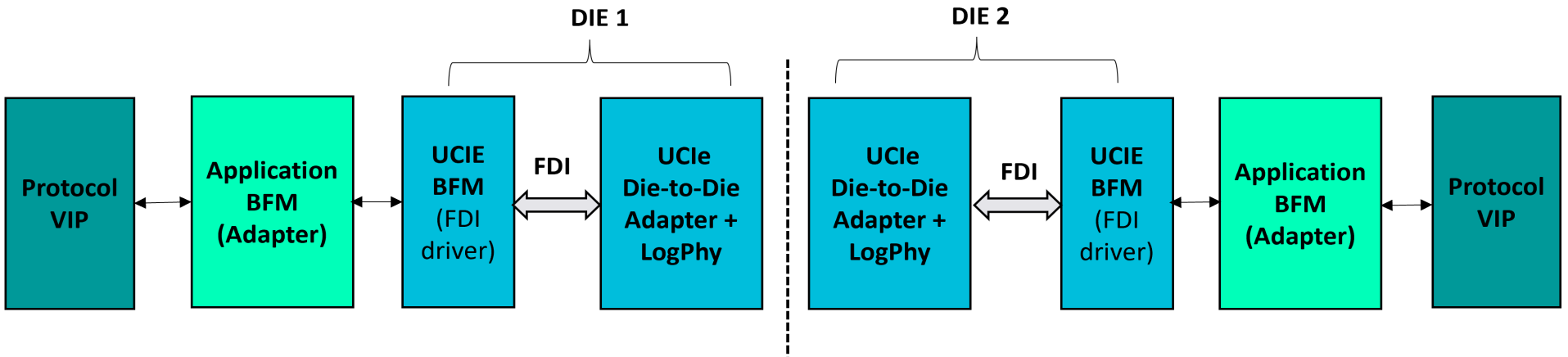
Case study 3 : Integrating VIP Protocol over UCIe

Adapter is constructed as an application BFM extended from Protocol device BFM where protocol can be PCIe, CXL, AXI, CHI, JESD204E, etc.



Case study 4 : Integrating flit packing/unpacking logic over UCIe

Protocols like PCIe and CXL align directly with UCIe flit formats, while others with varied packet or signalling structures use hooks present in versatile adapter for custom or default data packing and unpacking into UCIe flits.

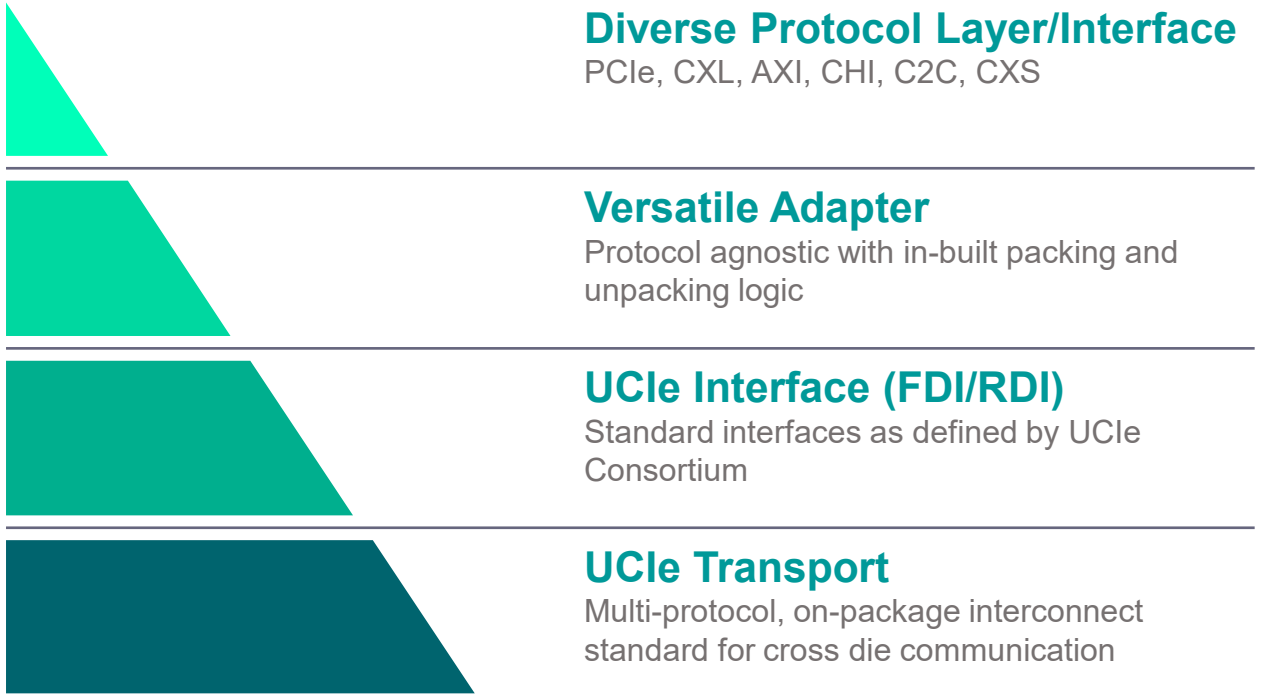


Key Takeaways

It is reasonable to expect that UCIe will continue evolving to accommodate wider array of protocols.

The proposed technique to achieve the adapter can facilitate wider applications as protocols compatible with interfaces like C2C and CXS can also be integrated over UCIe.

Furthermore, the UVM testbench complexities can be reduced multi-fold with this streamline approach to realize communication between Protocols and UCIe.



Thank You

Visit us at Siemens EDA booth!