eInfochips | The Solutions People

snug
Synopsys Users Group
Austin

Layered Testbench Architecture for Serial Protocol using UVM

Gaurav Brahmbhatt, Pinal Patel
eInfochips Ltd.
Joe McCann
Synopsys Inc.

September 29, 2016
Hilton Austin

Agenda

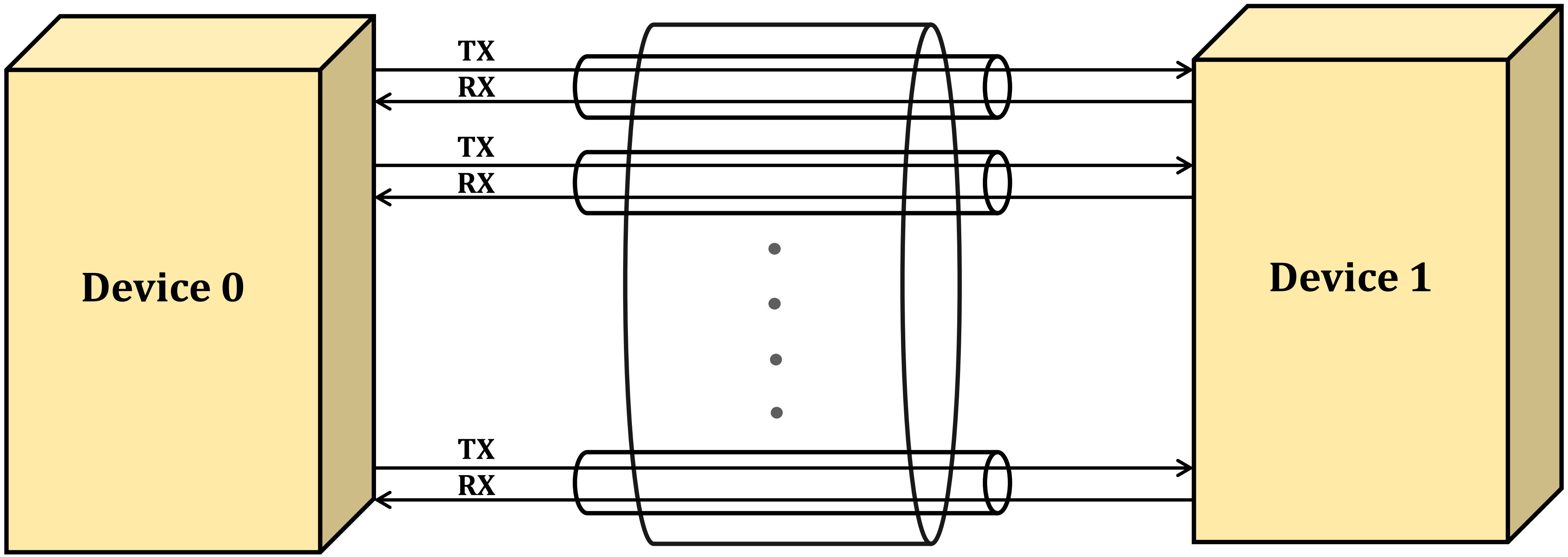
- Introduction
- Layered Testbench Architecture
- Whitebox Layers
- Advantages
- Summary

Introduction

Introduction

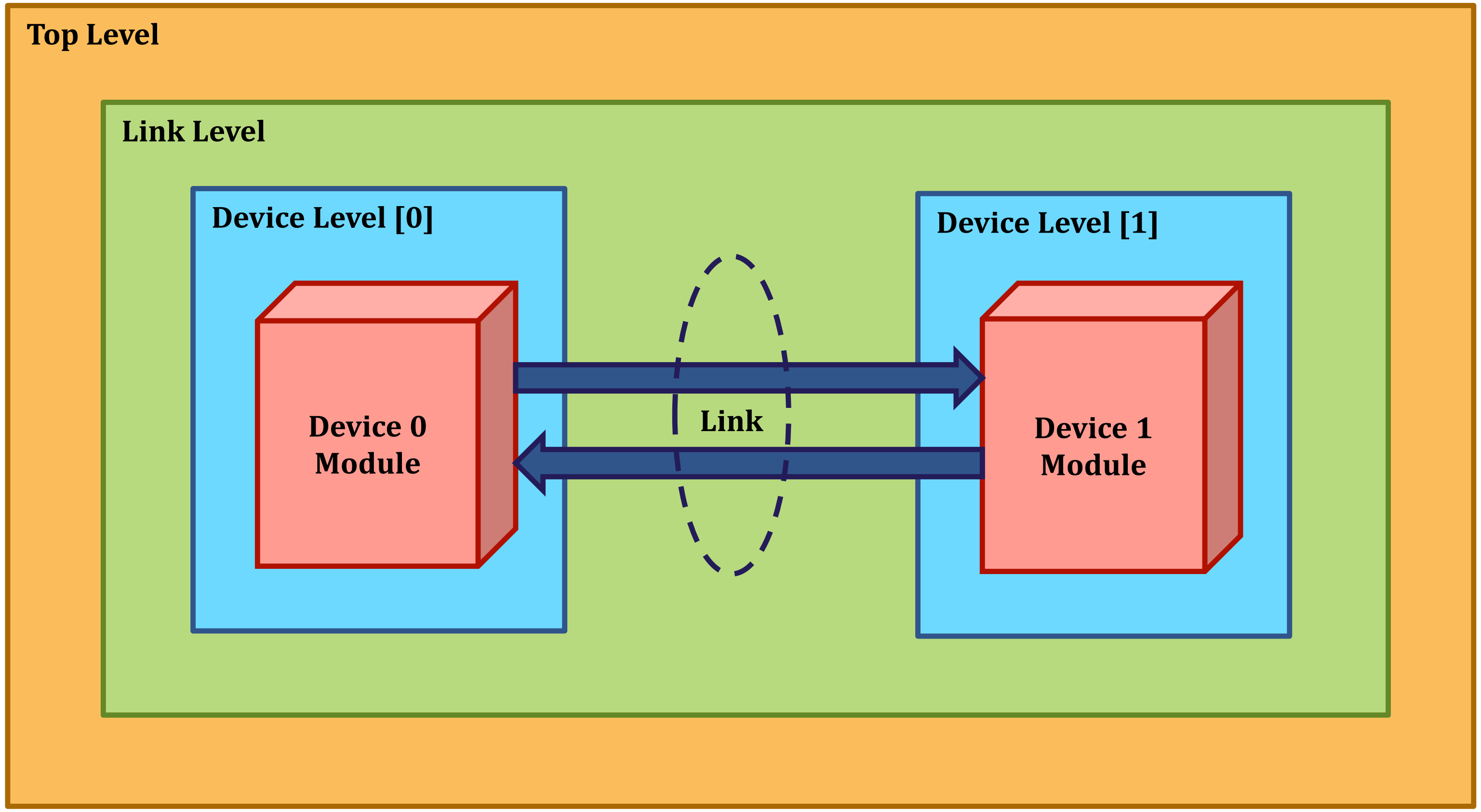
- Serial Protocol having two devices connected through a link

Serial Protocol Interface

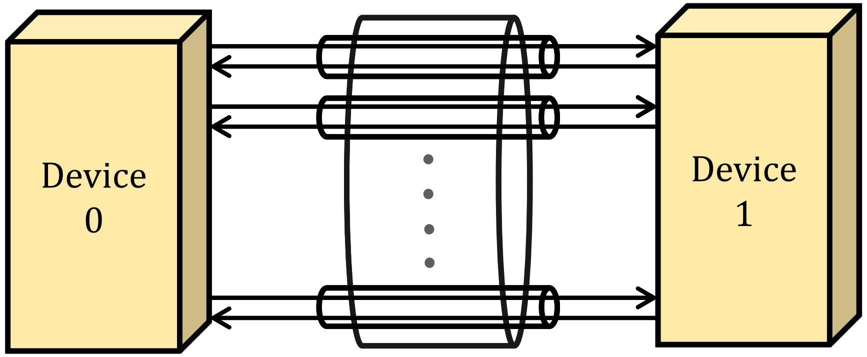


Layered Testbench Architecture

Layered Testbench Architecture

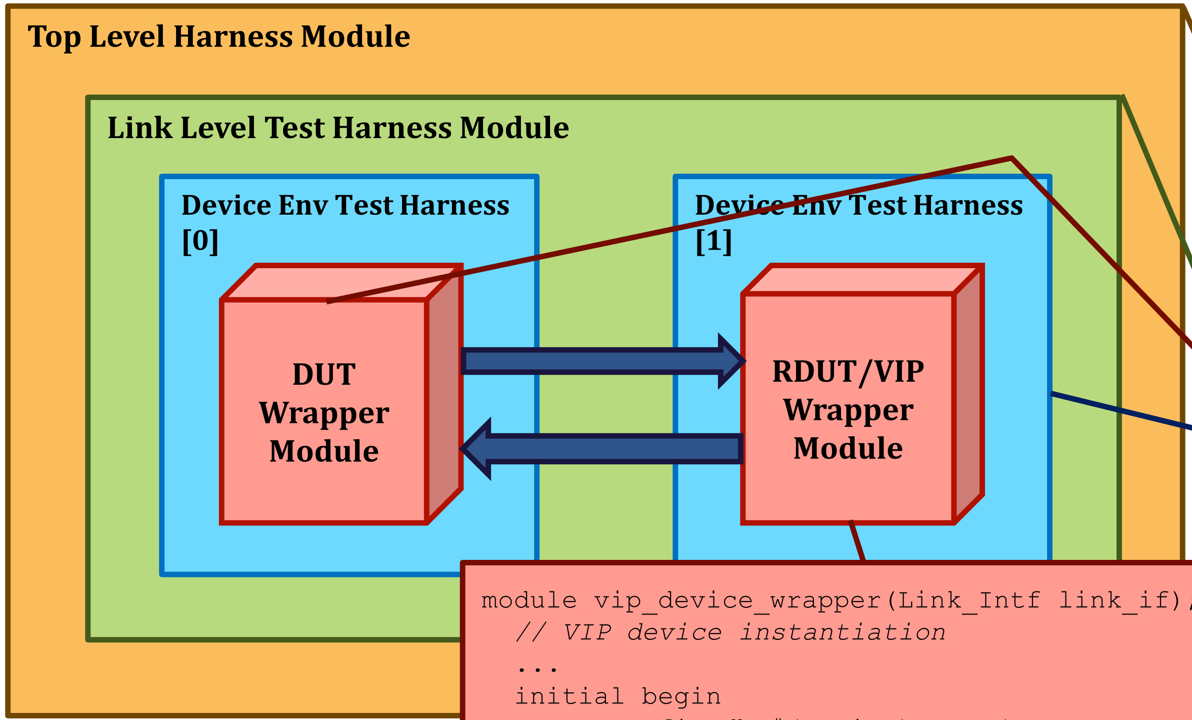


Serial Protocol Interface



Layered Testbench Architecture

Module Hierarchy


Top Level Harness Module
Link Level Test Harness Module
Device Env Test Harness [0]
Device Env Test Harness [1]
DUT Wrapper Module
RDUT/VIP Wrapper Module

```
module vip_device_wrapper(Link_Intf link_if);
  // VIP device instantiation
  ...
  initial begin
    uvm_config_db #(string)::set(
      uvm_root::get(),
      {dev_env_path,"*"},
      "vip_inst_name",
      vip_env_inst_name
    );
  end
endmodule : vip_device_wrapper
```

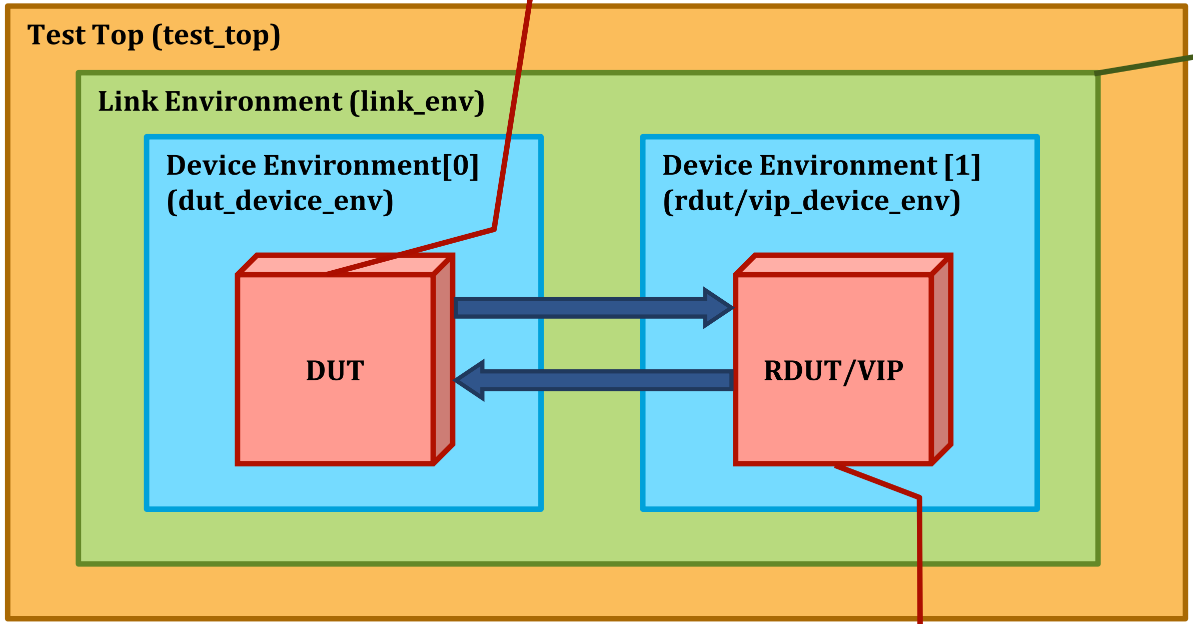
```
`d
mo
ay
EX,i))
en
–
de
ar
–
ar
-line
```

```
module device_test_harness ( Link_Intf link_if );
// generate device wrapper
  generate
    case(DEVICE_HARNESS_TYPE)
      // dut device
      DUT:
          if (RDUT_ENABLE==0) begin: g
             dut_device_wrapper #(
              .LINK_INDEX   (LINK_INDEX  ),
              .DEVICE_INDEX (DEVICE_INDEX),
              .IS_DUT       (1)
              ) u_wrap (.link_if  (link_if));
          end
`ifdef DUT_TO_RDUT
          else begin: g
             dut_device_wrapper #(
              .LINK_INDEX   (LINK_INDEX  ),
              .DEVICE_INDEX (DEVICE_INDEX),
              .IS_DUT       (1)
              ) u_wrap (.link_if  (link_if));
          end
`endif
      // vip device
      VIP:
          begin: g
             vip_device_wrapper #(
              .LINK_INDEX   (LINK_INDEX  ),
              .DEVICE_INDEX (DEVICE_INDEX),
              .IS_DUT       (0)
              ) u_wrap (.link_if  (link_if));
      end
    endcase
  endgenerate
  ...
endmodule : device_test_harness
```

Layered Testbench Architecture

Environment Hierarchy

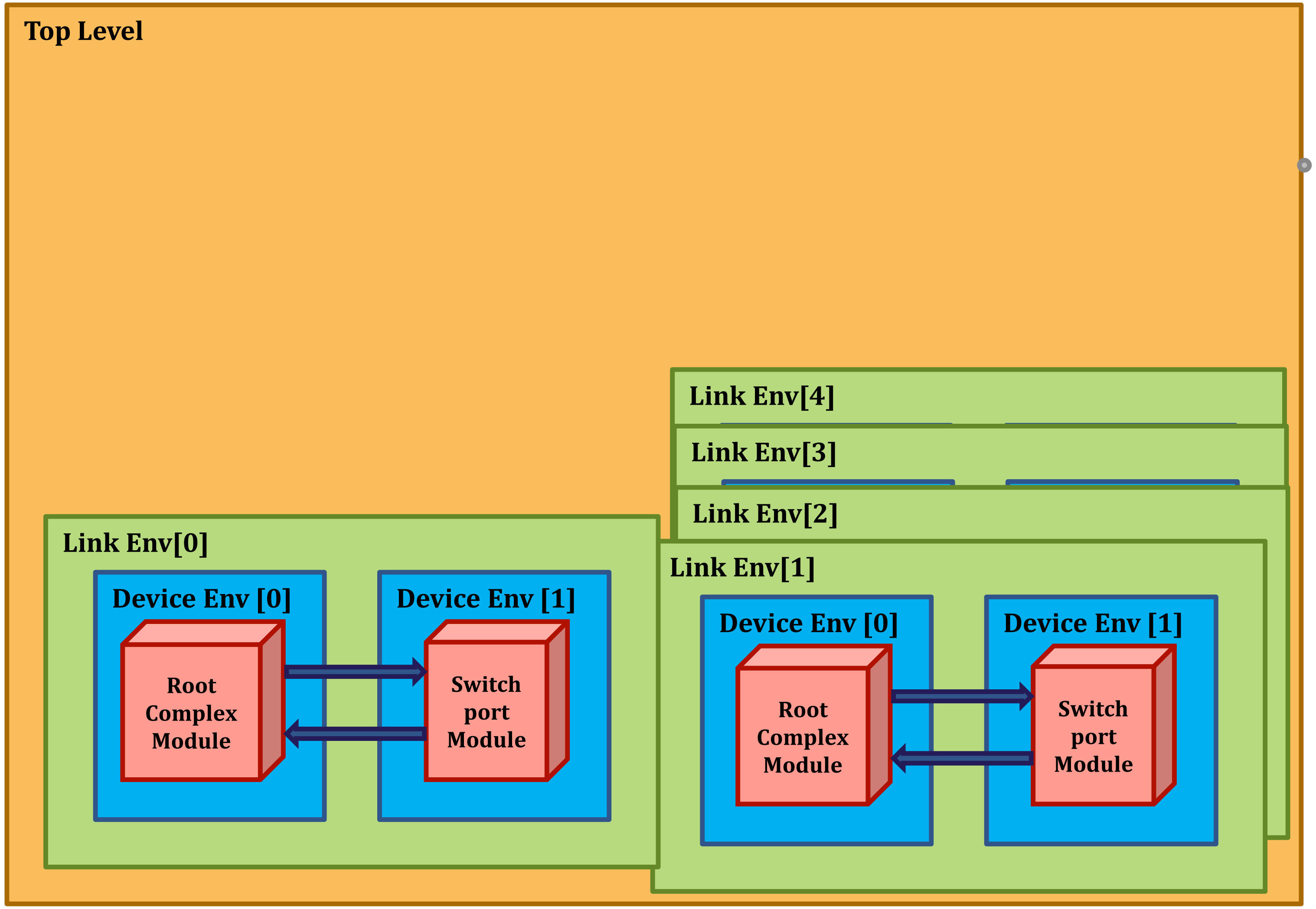
```
dut_device:
          set_inst_override_by_type(
            $sformatf("m_device_env[%0d]",i),
                       device_base_env::get_type(),
                       dut_env::get_type()
                      );
```

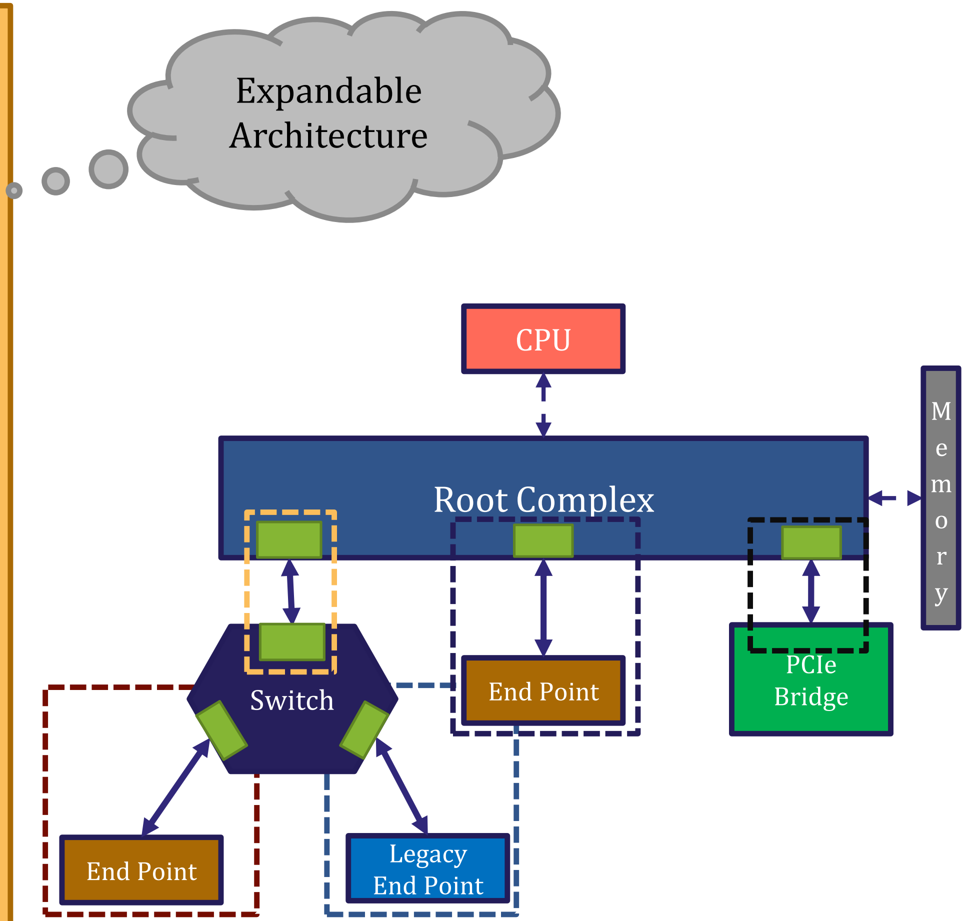


```
`define NO_OF_DEVICES 2
class link_env extends uvm_env;
  device_base_env  device_env[`NO_OF_DEVICES];

foreach (m_device_env[i])
    begin
       // factory override, to support
       // multiple env  types
     end
 end class
```

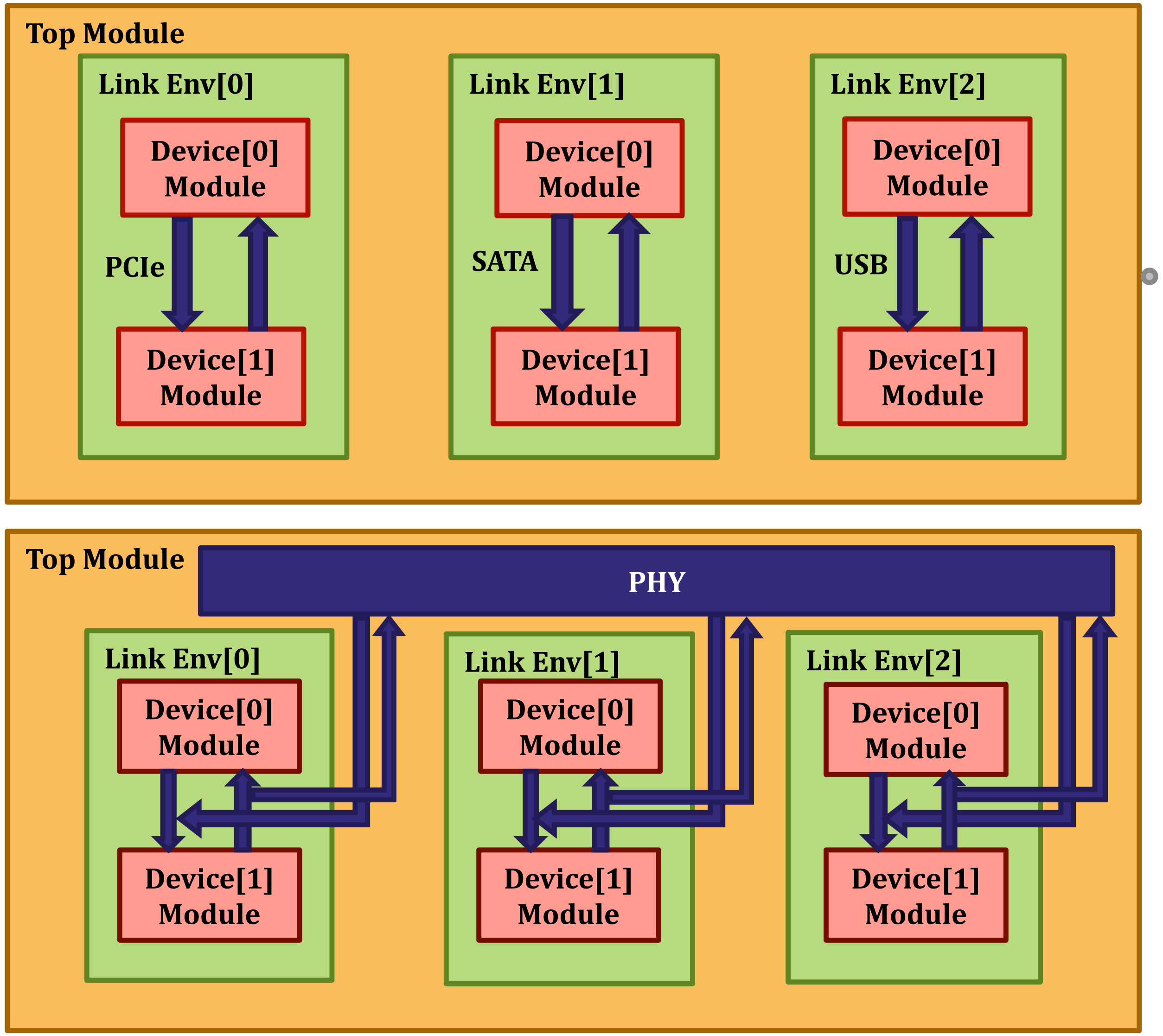
```
vip_device:
          set_inst_override_by_type(
             $sformatf("m_device_env[%0d]",i),
                        device_base_env::get_type(),
                        vip_env::get_type()
          );
```

Layered Testbench Architecture

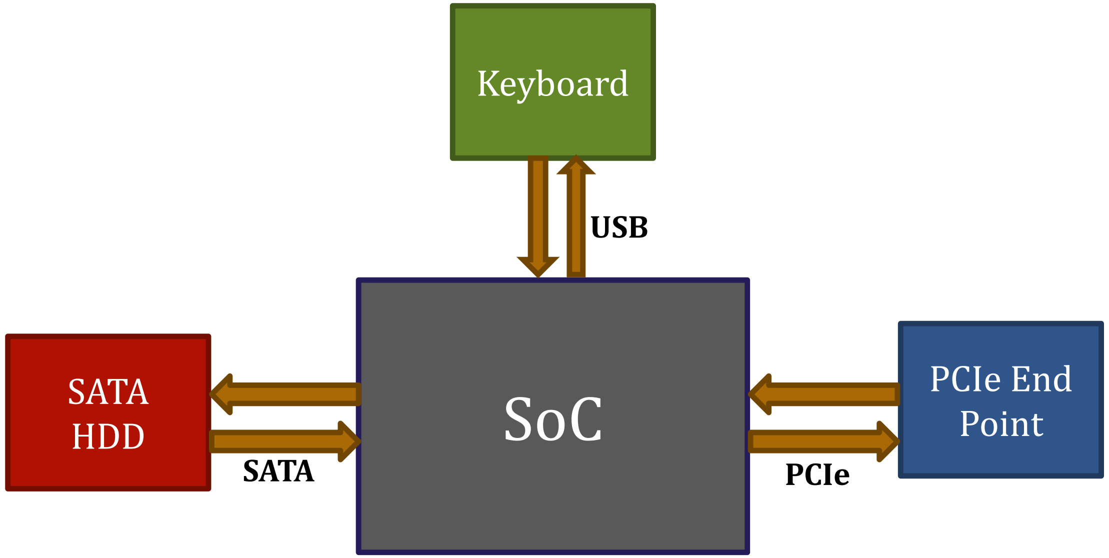




Layered Testbench Architecture

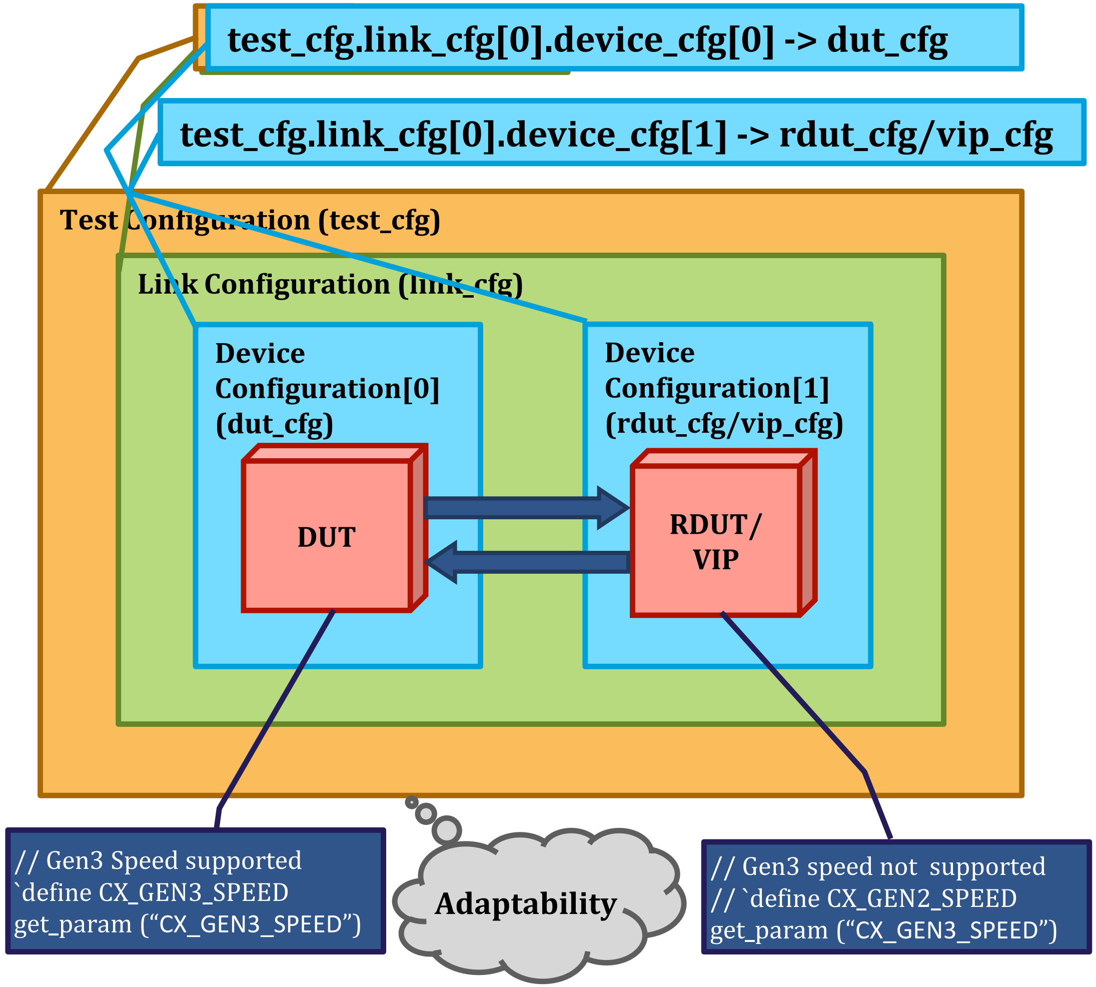


Scalable and Expandable Architecture



Layered Testbench Architecture

Configuration Hierarchy


test_cfg.link_cfg[0].device_cfg[0] -> dut_cfg
test_cfg.link_cfg[0].device_cfg[1] -> rdut_cfg/vip_cfg
Test Configuration (test_cfg)
Link Configuration (link_cfg)
Device Configuration[0] (dut_cfg)
DUT
Device Configuration[1] (rdut_cfg/vip_cfg)
RDUT/ VIP
Adaptability

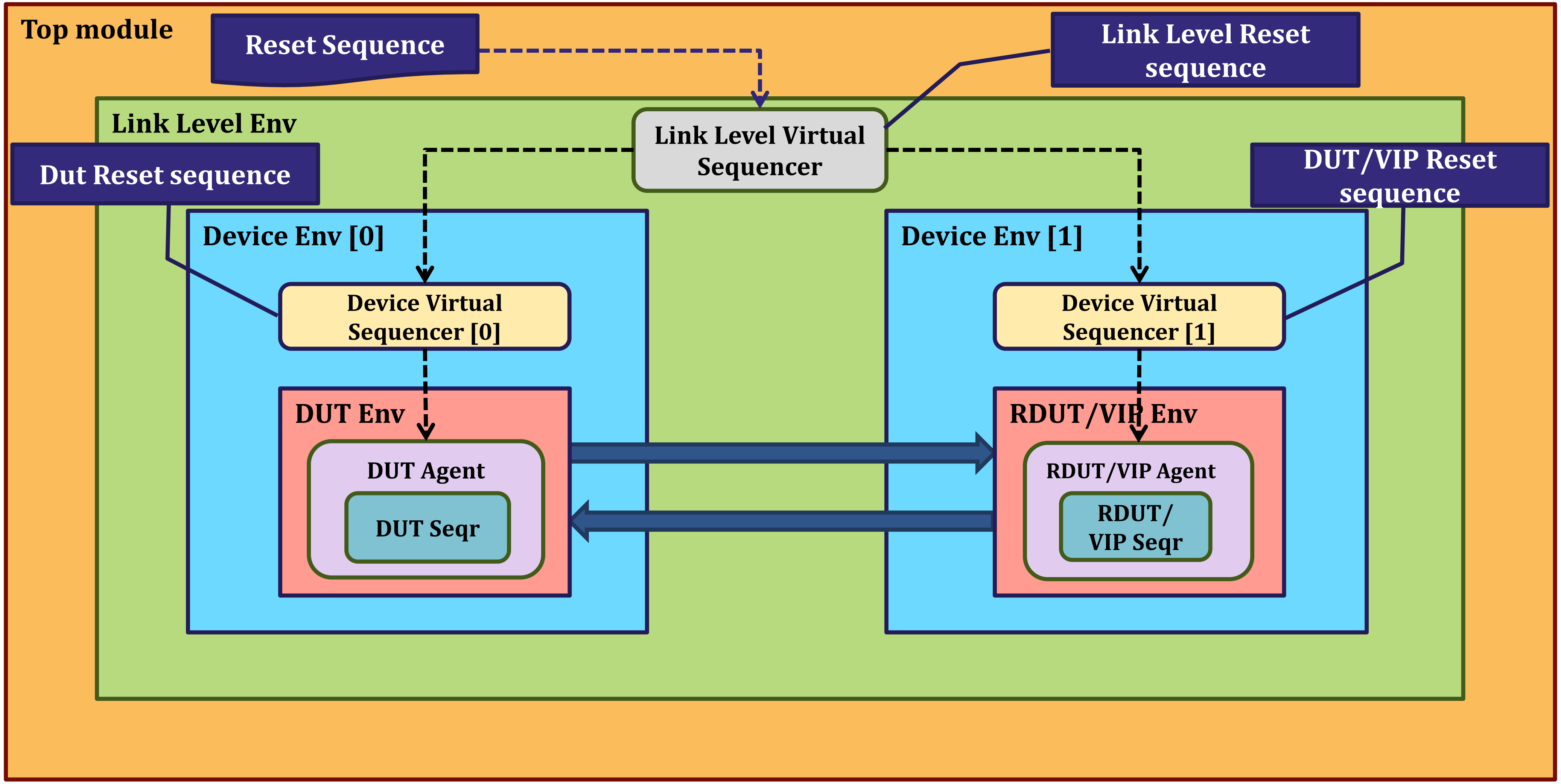
// Gen3 Speed supported
`define CX_GEN3_SPEED
get_param (“CX_GEN3_SPEED”)

// Gen3 speed not supported
// `define CX_GEN2_SPEED
get_param (“CX_GEN3_SPEED”)

```
class  SpeedChangeSeq extends LinkBaseVSeq ;
  …
  …
  if(p_sequencer.link_cfg[0].device_cfg[0].dev_params.get_val("CX_GEN2_SPEED”)  &&
      p_sequencer.link_cfg[0].device_cfg[1].dev_params.get_val("CX_GEN2_SPEED”))
       linkspeed = GEN2;

  else if(p_sequencer.link_cfg[0].device_cfg[0].dev_params.get_val("CX_GEN3_SPEED”) &&
           p_sequencer.link_cfg[0].device_cfg[1].dev_params.get_val("CX_GEN3_SPEED”))
      linkspeed = GEN3;

  else
       linkspeed = GEN1;

  // Setup the target link speeds for both devices.
  foreach(p_sequencer.link_cfg.device_cfg[ii]) begin
    if(p_sequencer.link_cfg.device_cfg[ii].deviceEnvType == DUT) begin
     `uvm_do_on_with( DutSetTrgtLinkSpeedSeq , p_sequencer.device_vseqr[ii],
                      { TrgtSpeed ==  linkspeed; })
    end

    if(p_sequencer.link_cfg.device_cfg[ii].m_deviceEnvType == VIP) begin
      `uvm_do_on_with( VipSetLinkSpeedSeq, p_sequencer.device_vseqr[ii],
                      { TrgtSpeed ==  linkspeed; } )
    end
  end
  …
  …
endclass
```

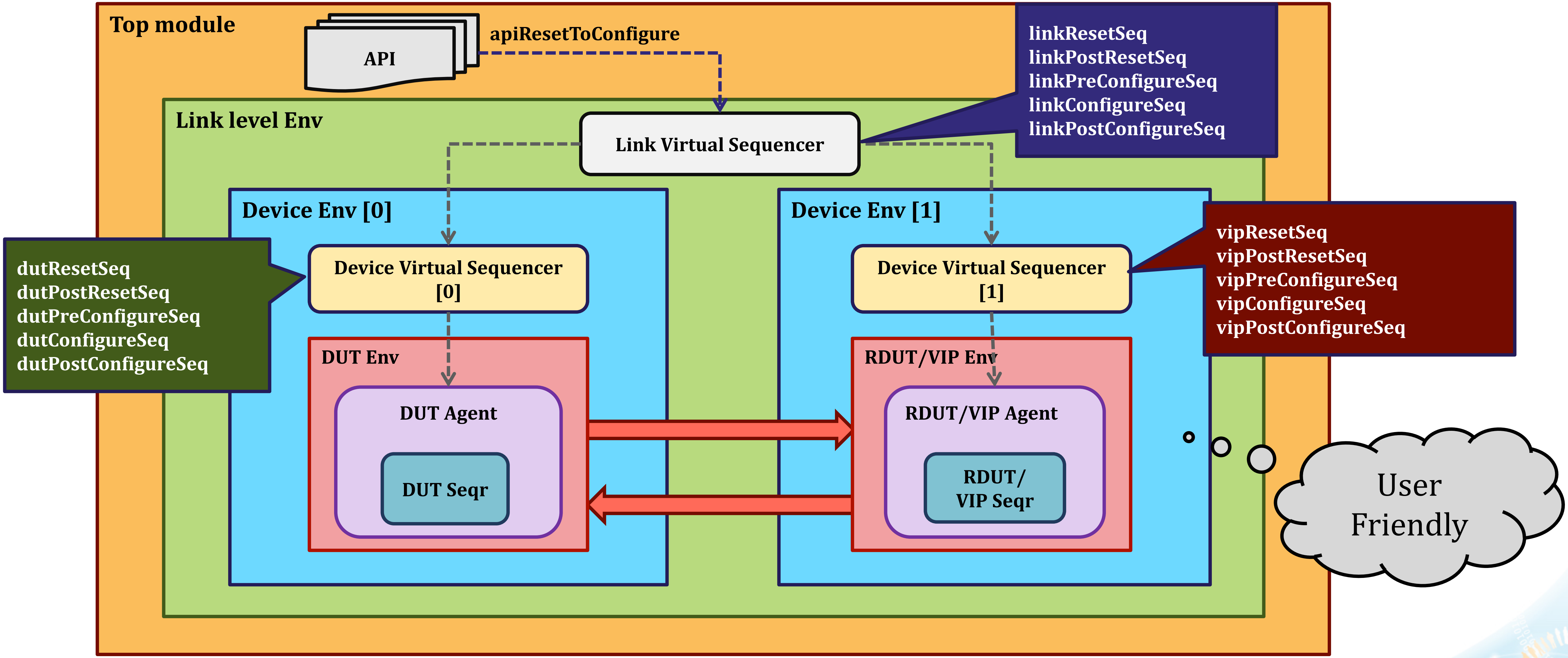
Layered Testbench Architecture

Sequencer Hierarchy



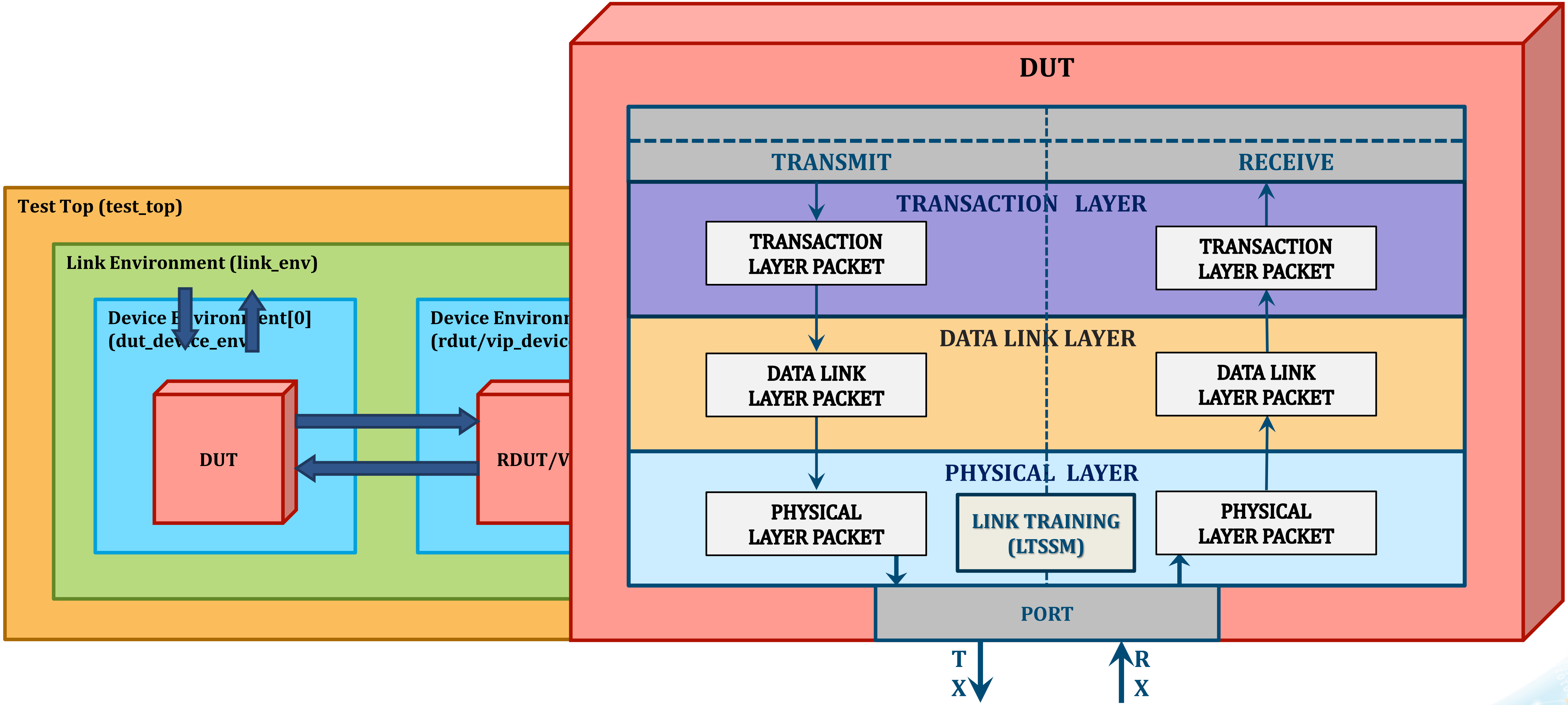
Layered Testbench Architecture

APIs (Application Programming Interface)



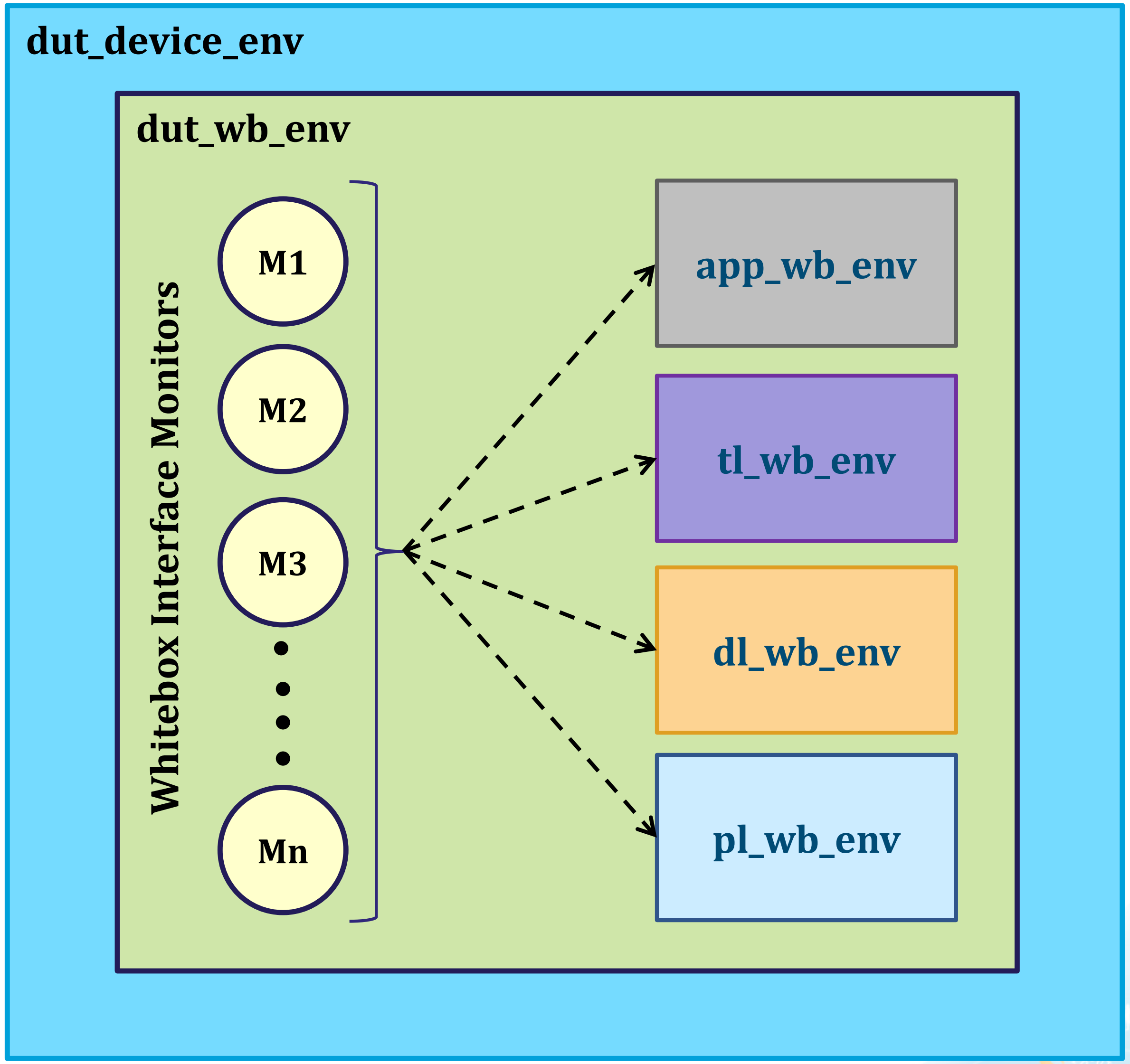
Whitebox Layers

Whitebox Layers

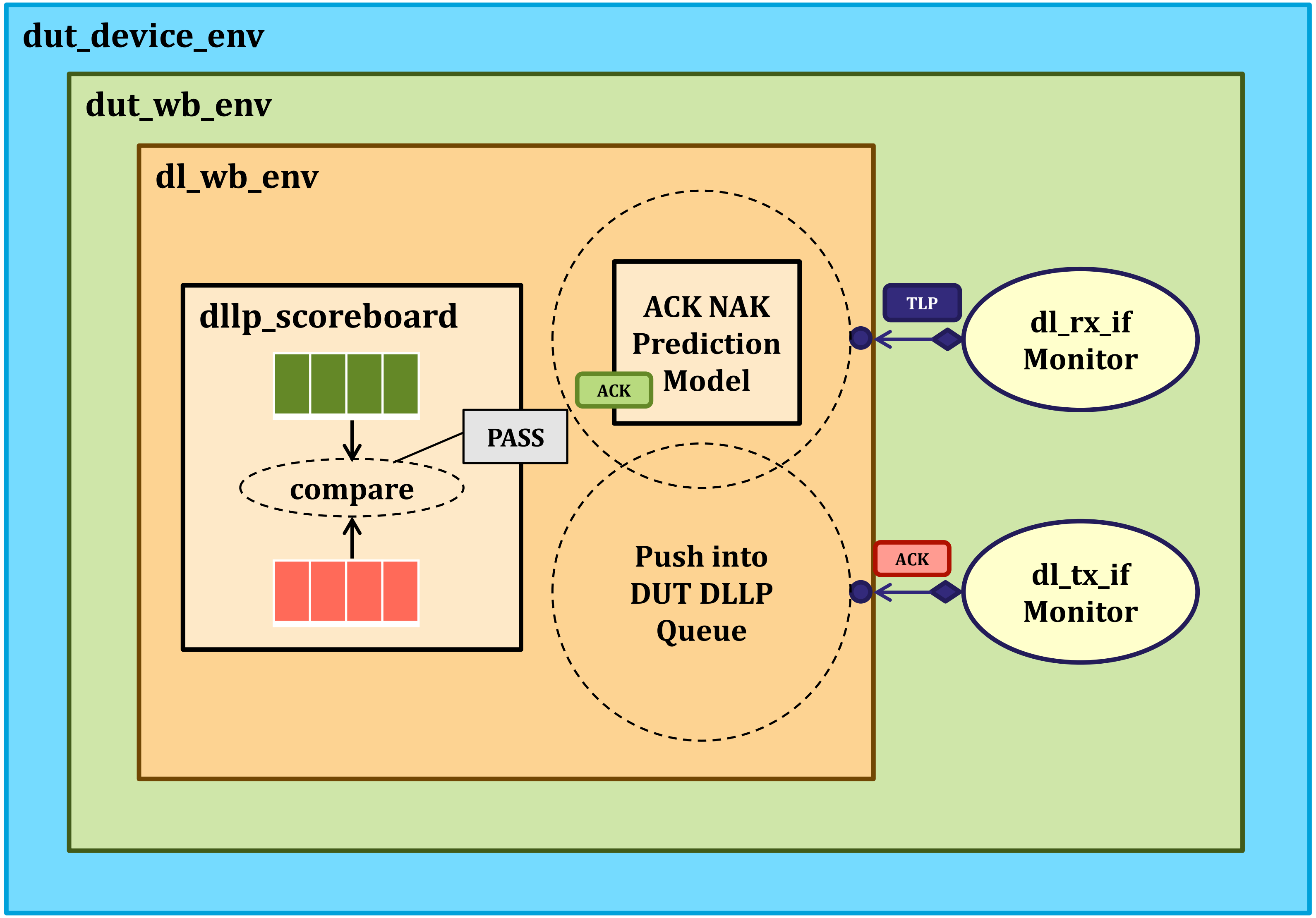


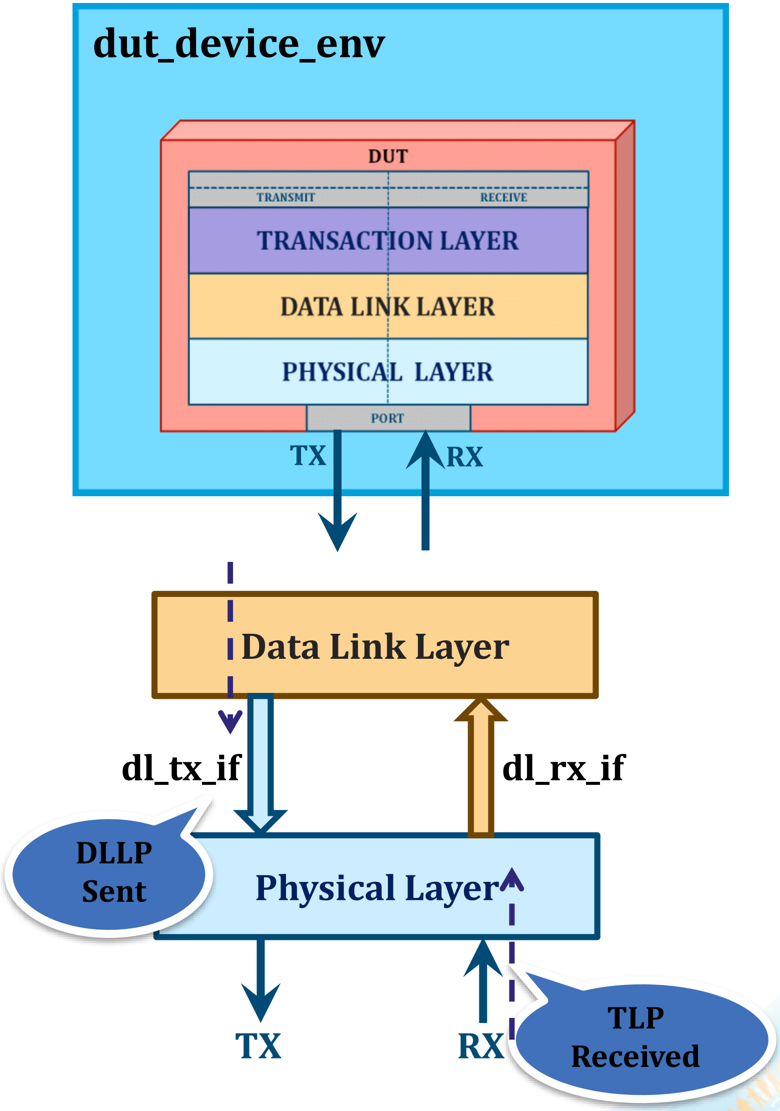
Whitebox Layers

- Whitebox checkers force the emulation of DUT functionality into logical blocks
  - Often the emulation is distributed throughout the environment and testcases
- Whitebox checkers are active for all tests
  - Whitebox checks can be active during error injection tests
  - Typically End-2-End Scoreboards must the turned off during error injection
- Whitebox checkers point to the source of the bug with much greater level of granularity
- Although initial development time is required, they can greatly reduce debug effort in a project life cycle
- Optimal point for positioning of coverage code

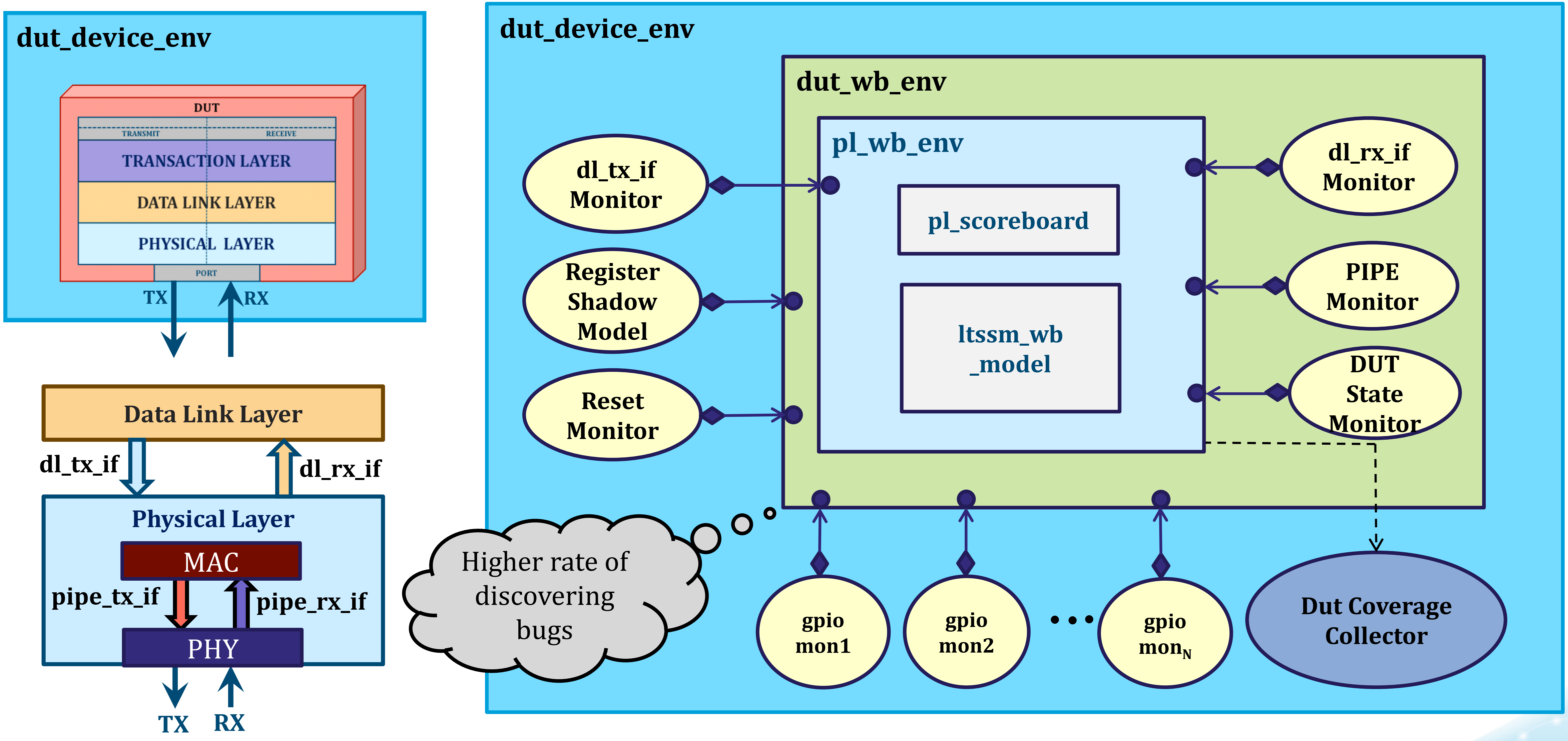


Whitebox Layers





Whitebox Layers



Higher rate of discovering bugs

Advantages and Summary

Advantages

- Adaptability
- Expandable architecture
- Works well with different IP variations
- User friendly
- Higher rate of discovering bugs

Summary

- The layered architecture provides an approach to create flexible, scalable and manageable testbench to verify point to point protocol with many complex functionalities.

- This architecture has an ability to mould itself for various topologies (i.e. DUT-to-VIP or DUT-to-DUT connection), different IP variations and a range of protocols or designs. This makes it highly reusable.

- Whitebox environment helps us to find many corner case design bugs, hence stepping up the design quality significantly.

Q & A

Thank You