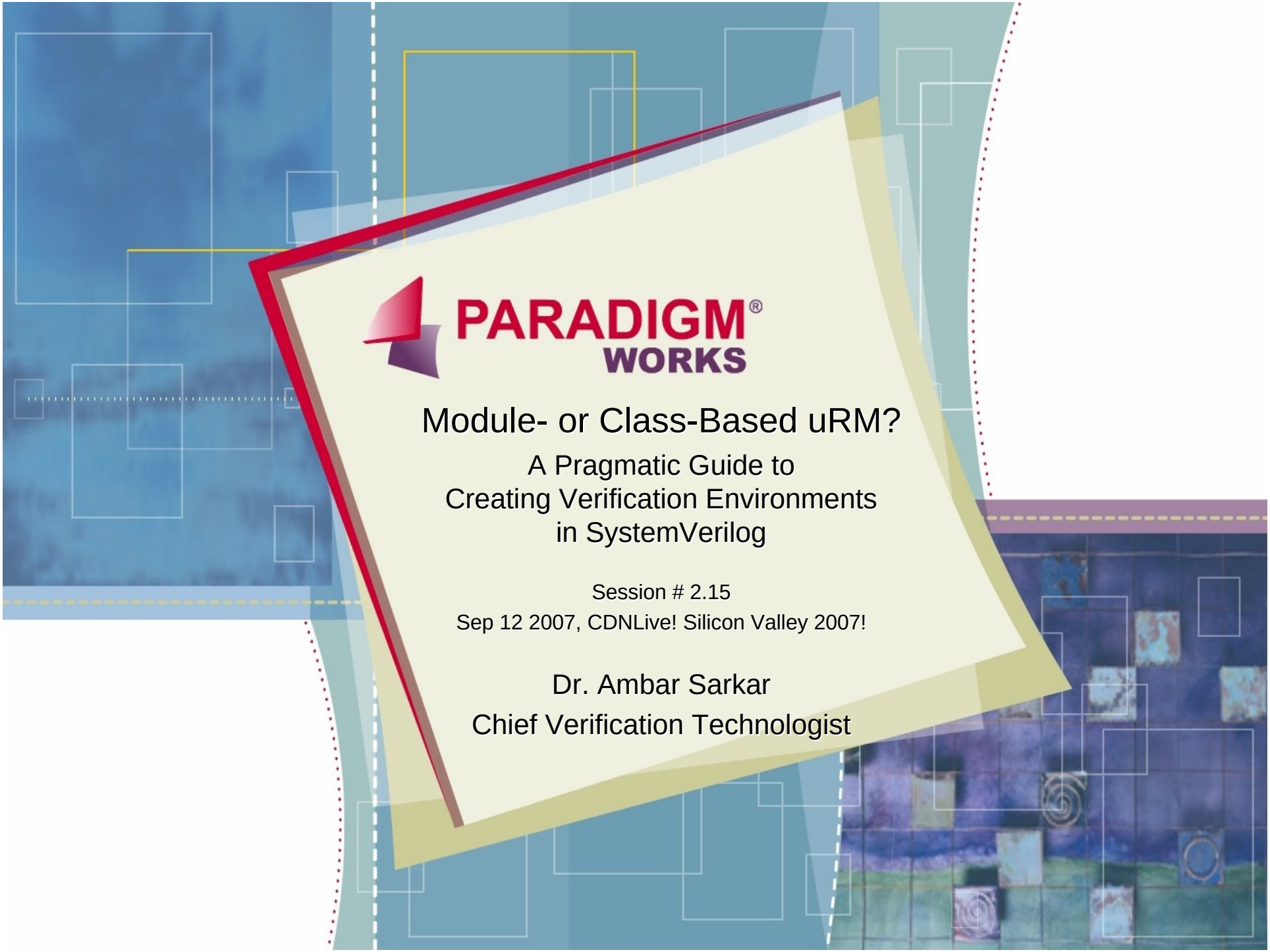


Module- or Class-Based uRM?

A Pragmatic Guide to Creating Verification Environments in SystemVerilog

Session # 2.15

Sep 12 2007, CDNLive! Silicon Valley 2007!

Dr. Ambar Sarkar

Chief Verification Technologist

PARADIGM® WORKS

Agenda

- Introduction
  - Module-based vs. Class-based URM
- Choosing the right verification methodology
  - Help maximize design team contribution
- Case Study
  - Hybrid environment
- Conclusion

PARADIGM® WORKS

Introduction

- What is URM?
  - Universal Reuse Methodology
  - Helps create functional verification environments
    - Captures effective practices
    - Highly reusable
- Based on
  - Constrained random stimuli
  - Coverage driven closure
  - Modular design
  - Well defined interfaces among components
- Two flavors of URM
  - SystemVerilog Module based URM (MB-URM)
  - SystemVerilog Class based URM (CB-URM)

PARADIGM® WORKS

Module Based vs. Class Based

Module-based URM

- Major verification elements are defined as modules
- Does not require mastery of advanced features or OO skills
- Defined for stand-alone verification environments
- Easy for designers to create block and chip-level verification environments
- Not open source(?)

Class-based URM

- Major verification elements defined as classes
- Requires OO skills
- Heavily focused on reusable components
- Suitable for creating more sophisticated verification environments.
- Open source

PARADIGM® WORKS

Maximizing Design Team Contribution

- Why should design team contribute to verification?
  - Schedule pressure
  - Better resource utilization
  - Corner case generation
- Barriers against contribution
  - Ad-hoc verification environment by designer
  - Verification team cannot effectively reuse ad-hoc code
  - Design team cannot use verification code
- Want to have a common environment
  - Both teams contribute
  - Both teams reuse each other's component

PARADIGM® WORKS

Maximizing Design Team Contribution

- The design team should easily create environments and tests as needed.
- Both teams should be able to execute their verification components together.
- Be able to run test sequences from each together. For example,
  - Design team: initialization sequence of the chip
  - Verification team: traffic pattern
  - Reuse scenario: init followed by traffic pattern
- Reuse all self-checks implemented in either environment

PARADIGM® WORKS

Choosing MB-URM or CB-URM

- All using MB-URM
  - Easier methodology
  - But may not be heavy duty enough
- All using CB-URM only
  - Well suited for reuse
  - But design team needs to tackle big learning curve
- Hybrid- CB-URM and MB-URM
  - Both teams in their comfort-zone
  - But how to integrate?

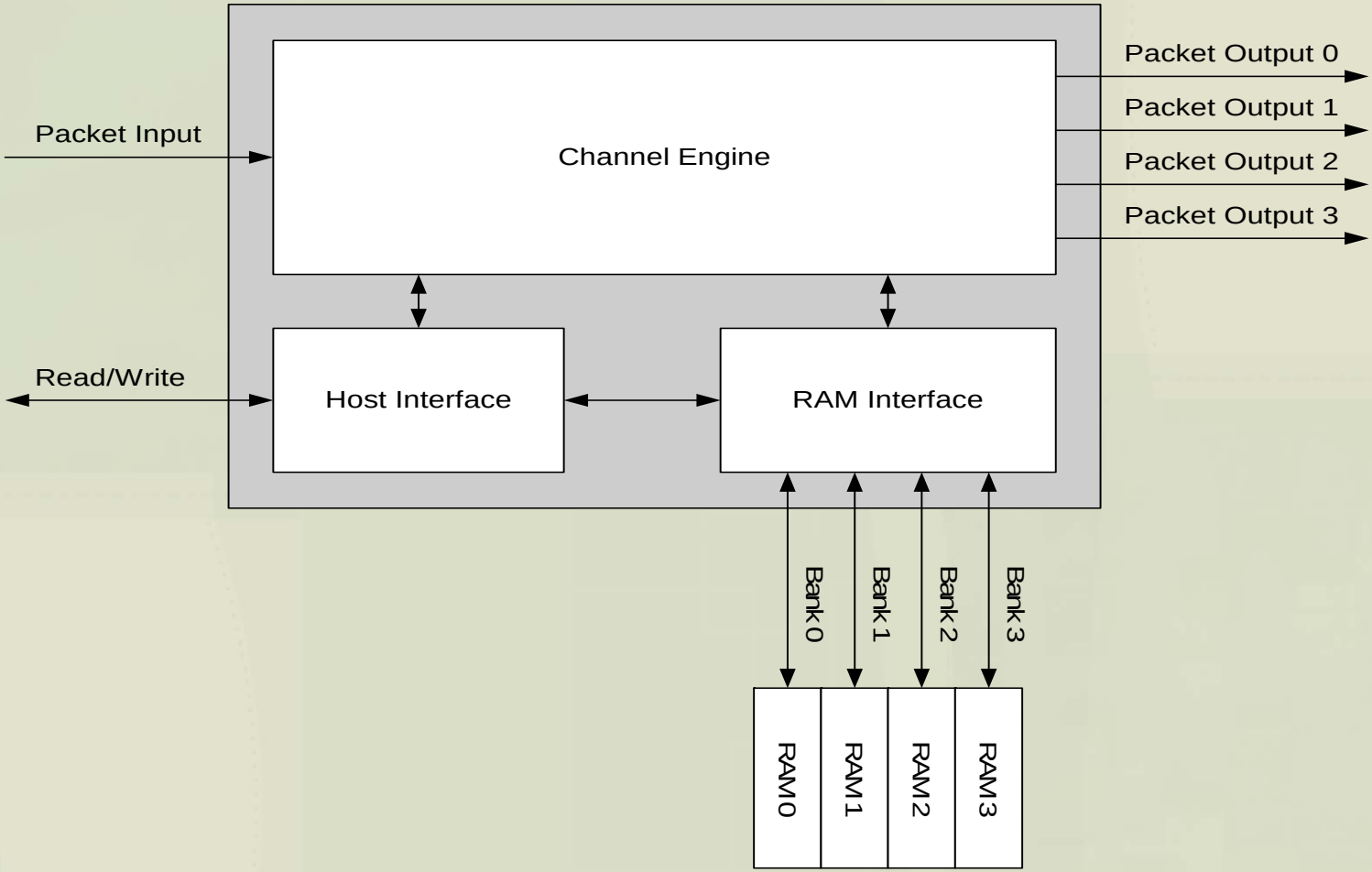
PARADIGM® WORKS

Challenges in Integrating MB-URM with CB-URM

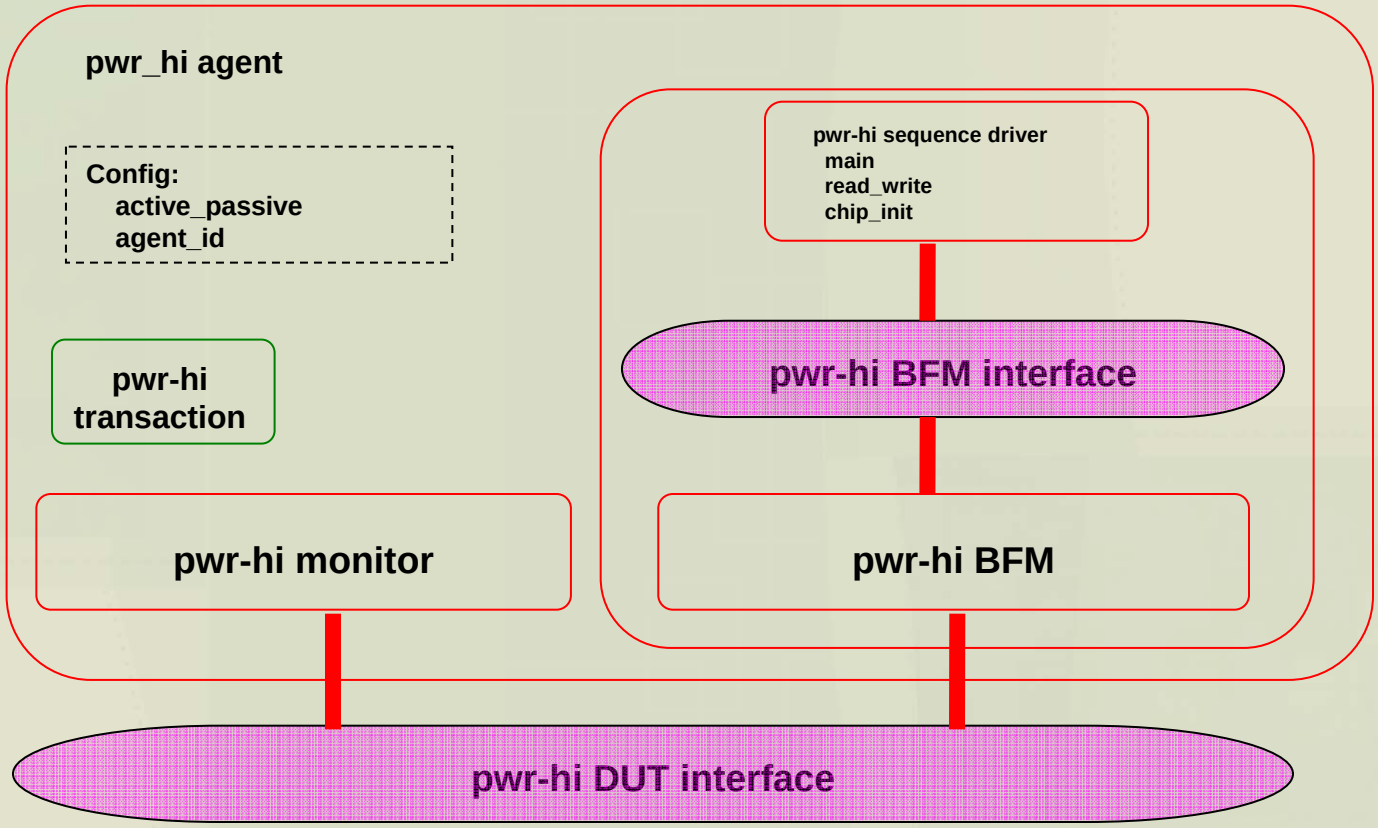
- Choosing the right flavor of the MB-URM:
  - the simple testbench environment
    - No agents or sequences, environments, but straightforward
    - Harder to maintain, ad-hoc
  - the general testbench environment.
    - More self-contained, with agents, sequences, environment
    - Some work needed in creating components
- Interaction between CB-URM and MB-URM elements
  - CB-URM BFM
    - SystemVerilog interface to talk to DUT
    - TLM port to communicate to other components
  - MB-URM BFM
    - SystemVerilog interface to talk to DUT
    - SystemVerilog interface to talk to other components
- MB- and CB-URM use different base libraries
  - Resolving conflicts?

PARADIGM® WORKS

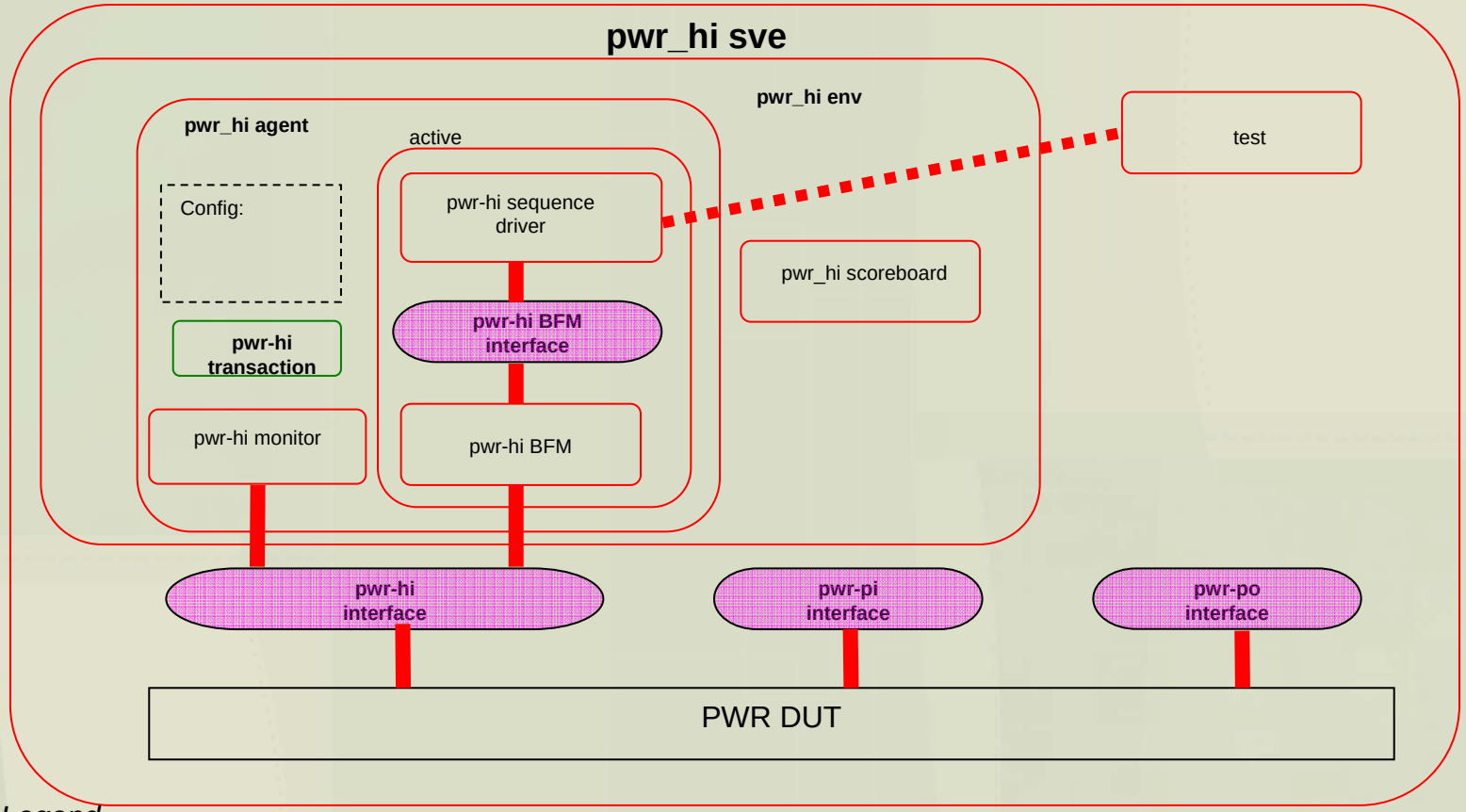
Case Study- PW Router



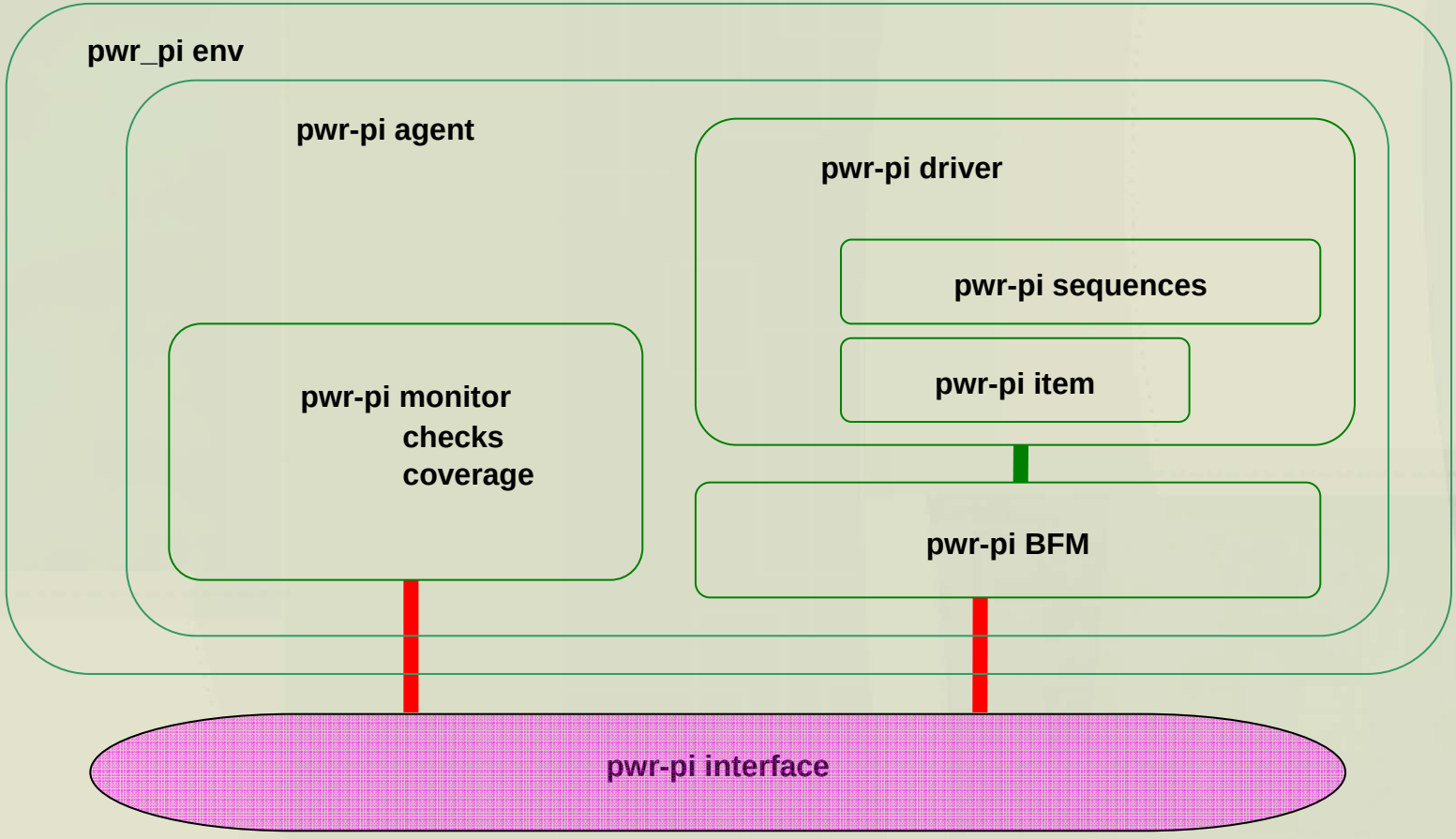
MB-URM agent



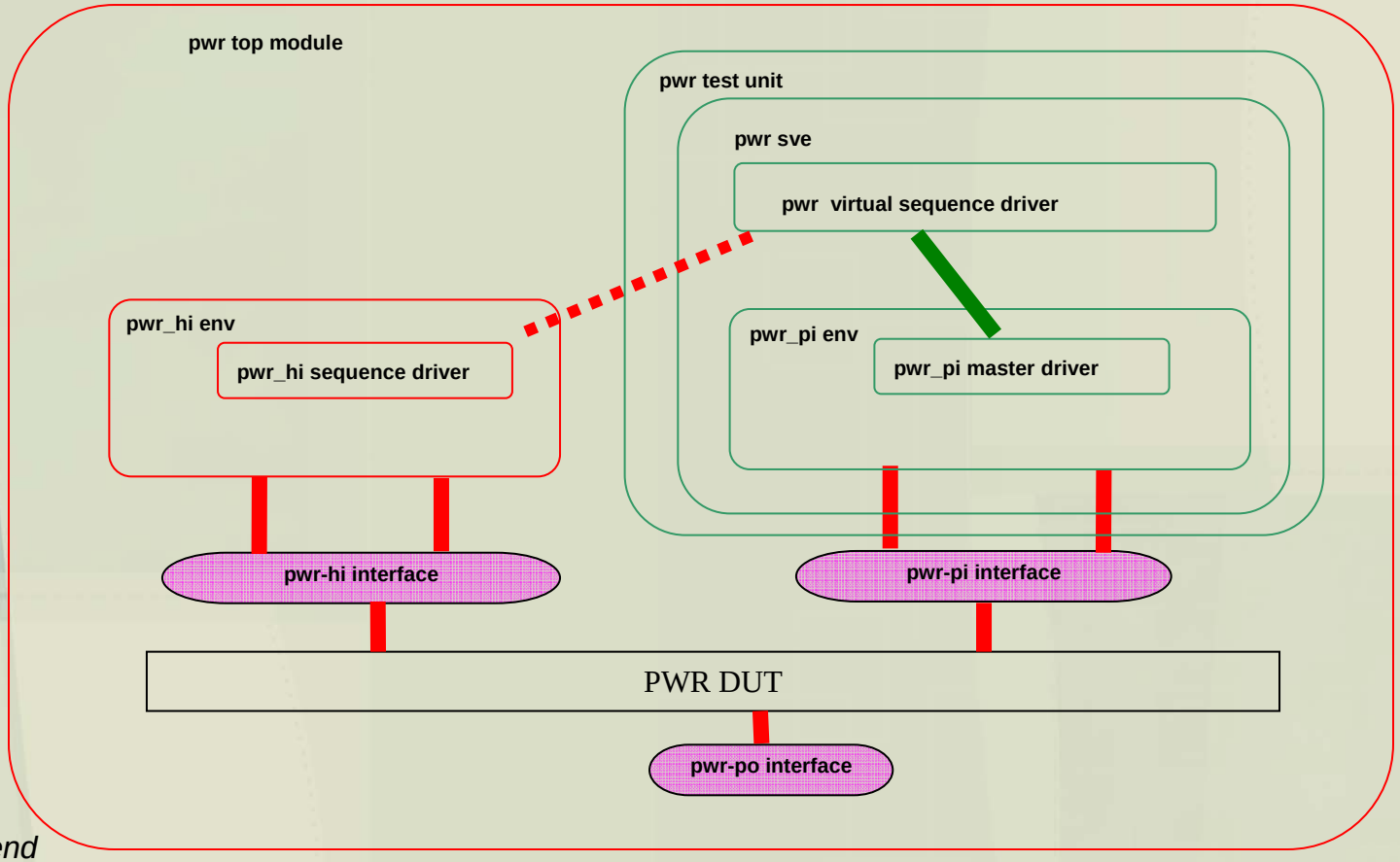
MB-URM environment



CB-URM Environment



Hybrid Environment



Legend

SystemVerilog Interface — *Connect via TLM interface* — *Connect via SystemVerilog interface*

Class instance — *Module instance* — *Direct task call*

Using MB-URM and CB-URM Sequences Together

```
class hybrid_seq extends urm_sequence;
    ...
    `urm_sequence_utils(hybrid_seq, pwr_pi_master_driver)

    send_pkts_seq pkts2port0; // Sequence defined in CB-URM

    virtual task body();
    begin
            ...

            $root.pwr_tb_top.pwr_hi_env.masters.agent[0].active.seq_driver.init(
    );
            `urm_do(pkts2port0)
            ...
    end
  endtask : body
endclass : hybrid_seq
```

PARADIGM® WORKS

Conclusion

- Combining MB-URM and CB-URM is straight-forward
- Recommended approach
  - Design team uses MB-uRM
    - Gets started creating its own unit-level tests.
  - Verification team uses CB-uRM
    - Implements the majority of the verification effort,
    - Reuses the MB-uRM components developed as needed.
  - Once the RTL development stage is over
    - Designers verify other blocks
    - Designers add more complex sequences

cadence™

cadence designer network

cdn LIVE™

CDNLive! 2007 Silicon Valley