# Speed up PCIe® System Verification using Hardware Acceleration

Kanwarpreet Grewal

Cadence Design Systems

# Disclaimer

**Presentation Disclaimer**: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

# Acknowledgements


- Co-authors for this paper
  - ✓ David Pena, Cadence Design Systems
  - ✓ Pete Heller, Cadence Design Systems

# Agenda

- PCI Express® verification: When to simulate and when to accelerate
- Value of hardware acceleration for PCI Express SoC/System Level Verification
- How to architect PCI Express VIP for acceleration
- Typical test flow for PCI Express acceleration VIP
- Maximizing acceleration performance
- Results
- PIPE vs. serial interface, error injection

# Simulation Satisfactory for Block Level Verification

- Simulation is the most commonly used method for verifying PCIe® protocol compliance
  - ✓ Requires thousands of small tests for all different aspects of the protocol
  - ✓ Corner case protocol testing
  - ✓ Provides needed functional coverage information
- Works well at block level
- Simulations can run at KHz speeds
- Test runs can be parallelized using server farms

# Simulation Is Not Sufficient for SoC/System Level

- System level/SoC level integration involves fewer, but longer running tests
- Some tests require multiple millions of cycles
- Long tests cannot be broken into smaller units for parallelization
- Simulation performance not adequate for system level
  - ✓ 10 to 100 cycles per second
- SW development/debug requires quick turnaround time

# Acceleration Necessary for SoC/System Level Verification

- System level integration has a different focus (relative to block level)
  - ✓ Assumes protocol compliance verification already completed
  - ✓ Focus on data integrity and connectivity when blocks come together in the system
  - ✓ Firmware/Software integration and verification frequently a priority
- **I.e., system level and block level verification needs for PCIe systems are *very different***
  - ✓ **High speed execution of longer tests is necessary**
  - ✓ **Connection to virtual platforms is often necessary**

# Verification Methods Compared and Contrasted

## Simulation

### Software simulation

+ Accurate
+ Flexible
- Not fast enough for large designs
- Not fast enough for embedded software

### Acceleration

+ Faster
+ Handles large designs
+ Debug via simulator UI
- Not fast enough for embedded software
- Set up time for porting your testbench

## Prototype

### In-circuit emulation

+ Fast enough for embedded software
+ Real world stimulus
+ Operates at RT-level
- Slower than real-time speed

### FPGA prototype

+ Inexpensive
+ Fast
- Long implementation time
- Little, if any debug capability
- Error prone
- Hard to change

**Hardware Verification Platform**

# Emulation Pros and Cons

Advantages:

- Much faster than simulation, especially for large designs
- Verifies functionality in actual system environment
- Finds functional system problems prior to tape out
- Allows concurrent development of hardware, software and systems
- Enables efficient post-silicon debug

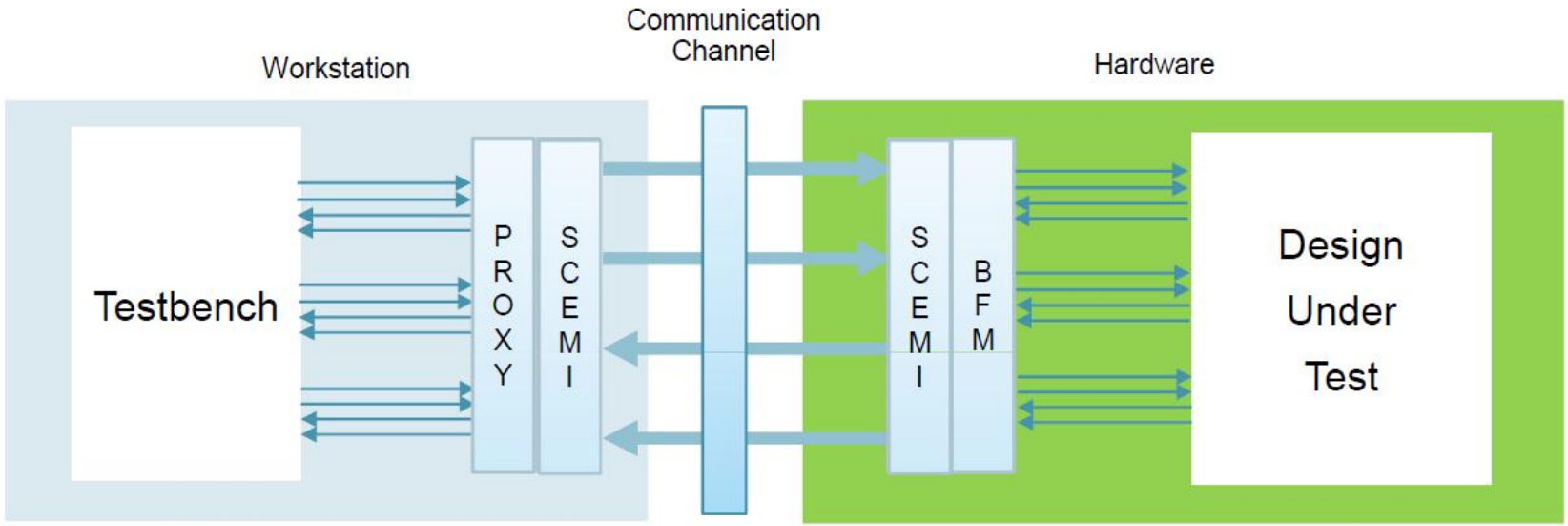
Disadvantages:

- Requires ability to set up target testing environment
- May require external interfaces of design to be developed early

# Why Accelerate?

- Supplements use model of in-circuit emulation
  - ✓ Can be done with incomplete design, or just parts of design
  - ✓ Avoids need for any external target hardware
- Can be as fast as emulation
  - ✓ But getting best performance requires an appropriately structured testbench
- Can leverage external target hardware and virtualized system components

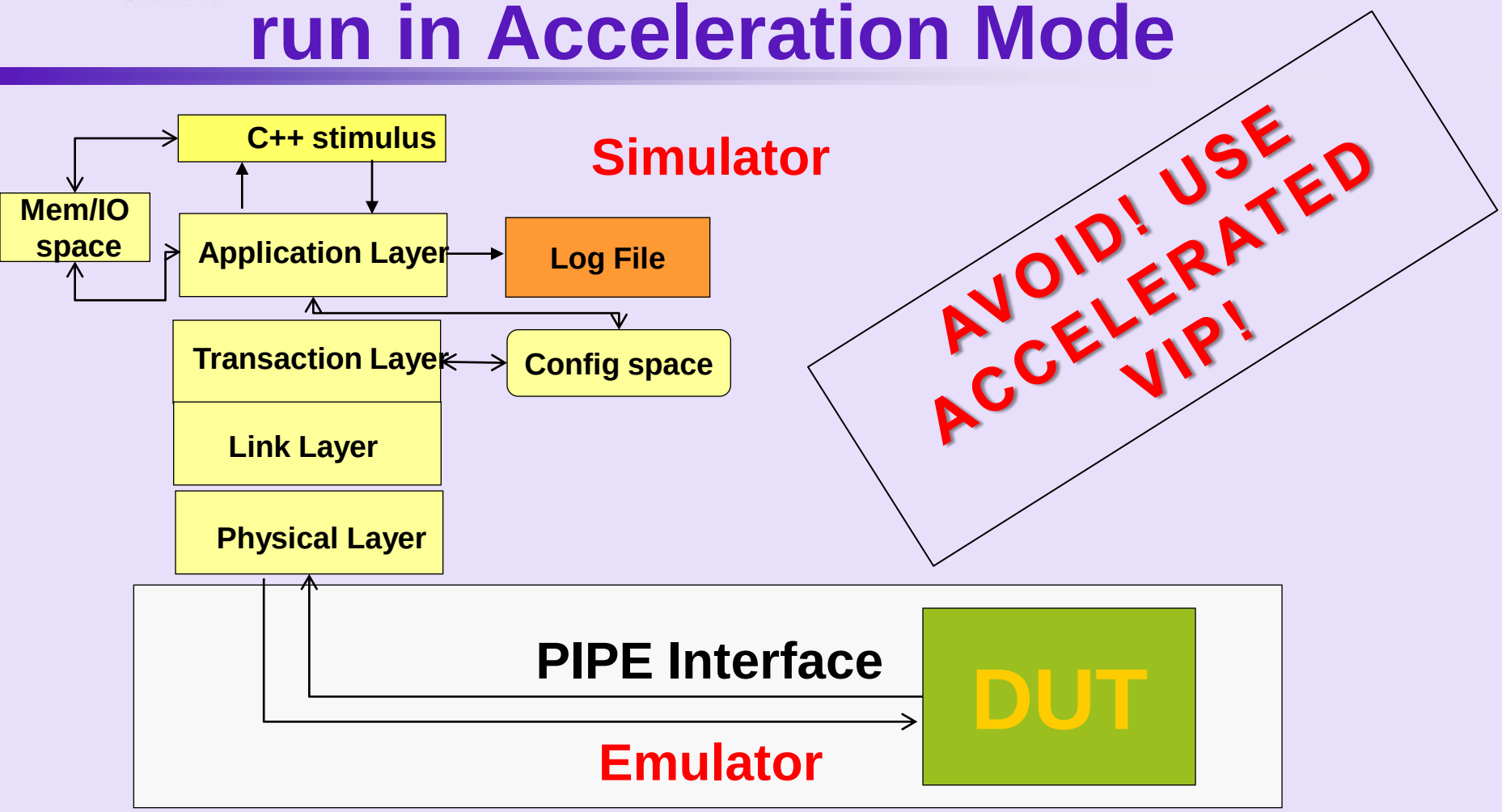
# Transaction Based Acceleration (TBA) Concepts




**Transaction Based Acceleration**

- Reduces communication channel overhead from signal based to transaction based
- Leverages fast hardware for DUT execution

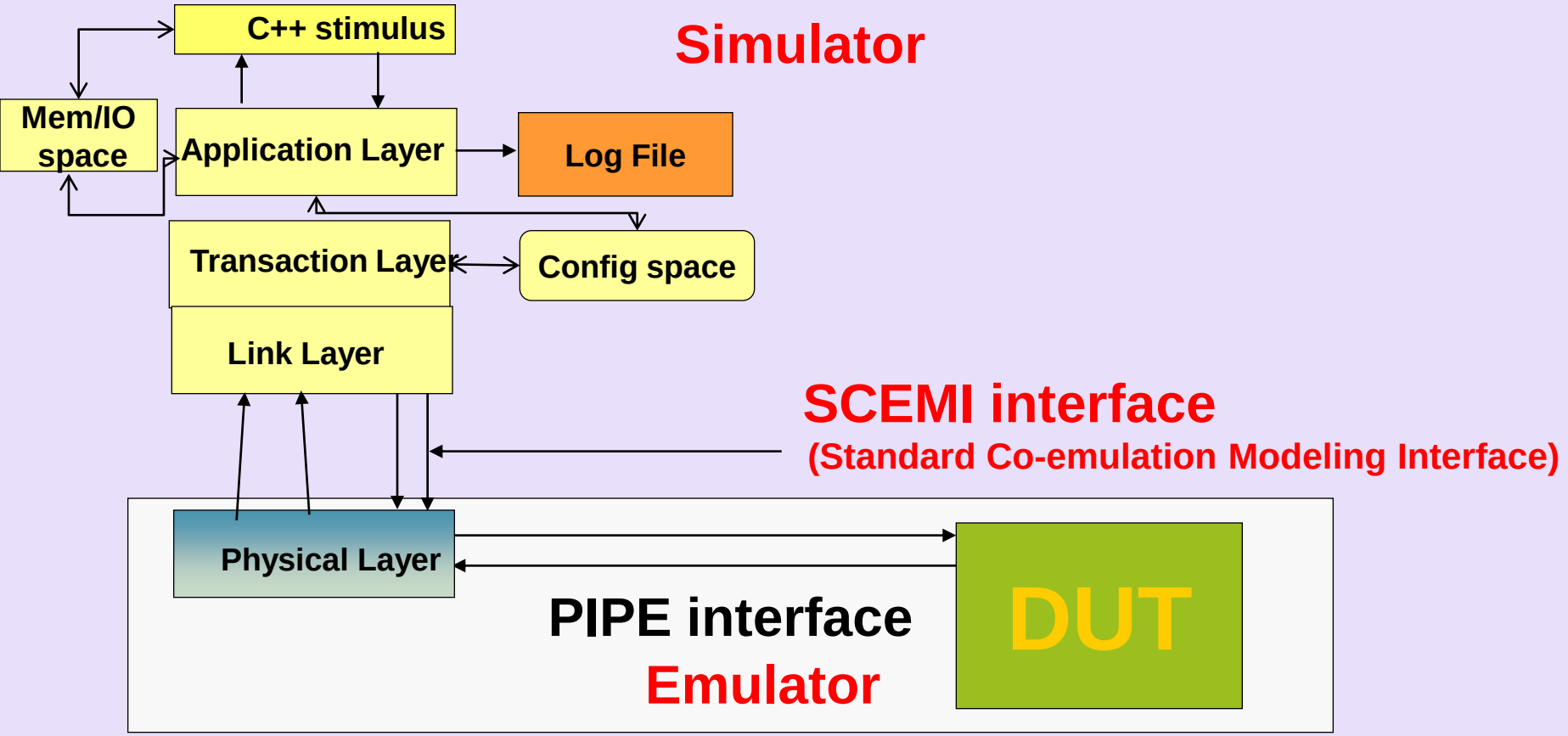
# PCI Express Simulation VIP run in Acceleration Mode



# PCI Express Simulation VIP run in Acceleration Mode

- Maximum control
  - ✓ All corner cases can be tested
- Minimum speed
  - ✓ HW resources wasted on simulation speeds
- Signal based Acceleration
  - ✓ For every symbol there would be HW-SW sync
- Acceleration Speed 1 to 2X sim speed
- With some changes speed can be increased
  - ✓ Buffering on TX/RX lanes

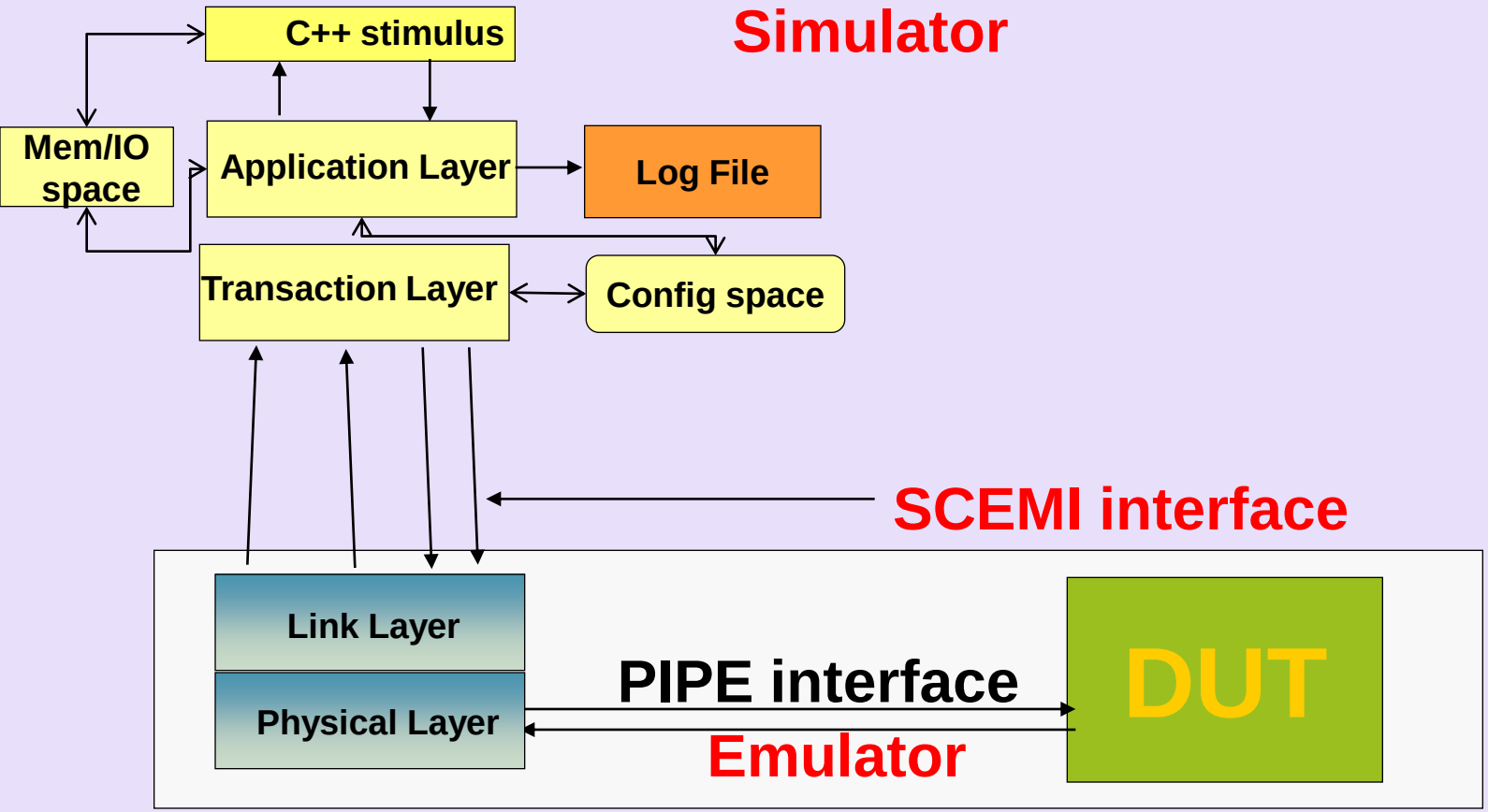
# PCI Express Acceleration VIP with Physical Layer in HW



# PCI Express Acceleration VIP with Physical Layer in HW

- Lesser control
  - ✓ Will not be able to control the LTSSM state transitions
  - ✓ Will not be able to control Training sequences
- More speed
  - ✓ Simulator can be completely inactive during link training/recovery/low power states
- HW-SW sync on every packet (TLP and DLLP)
- Buffering of packets can increase speed
- Flow control DLLP frequency changes to increase speed
- Acceleration Speed 30-50X sim speed

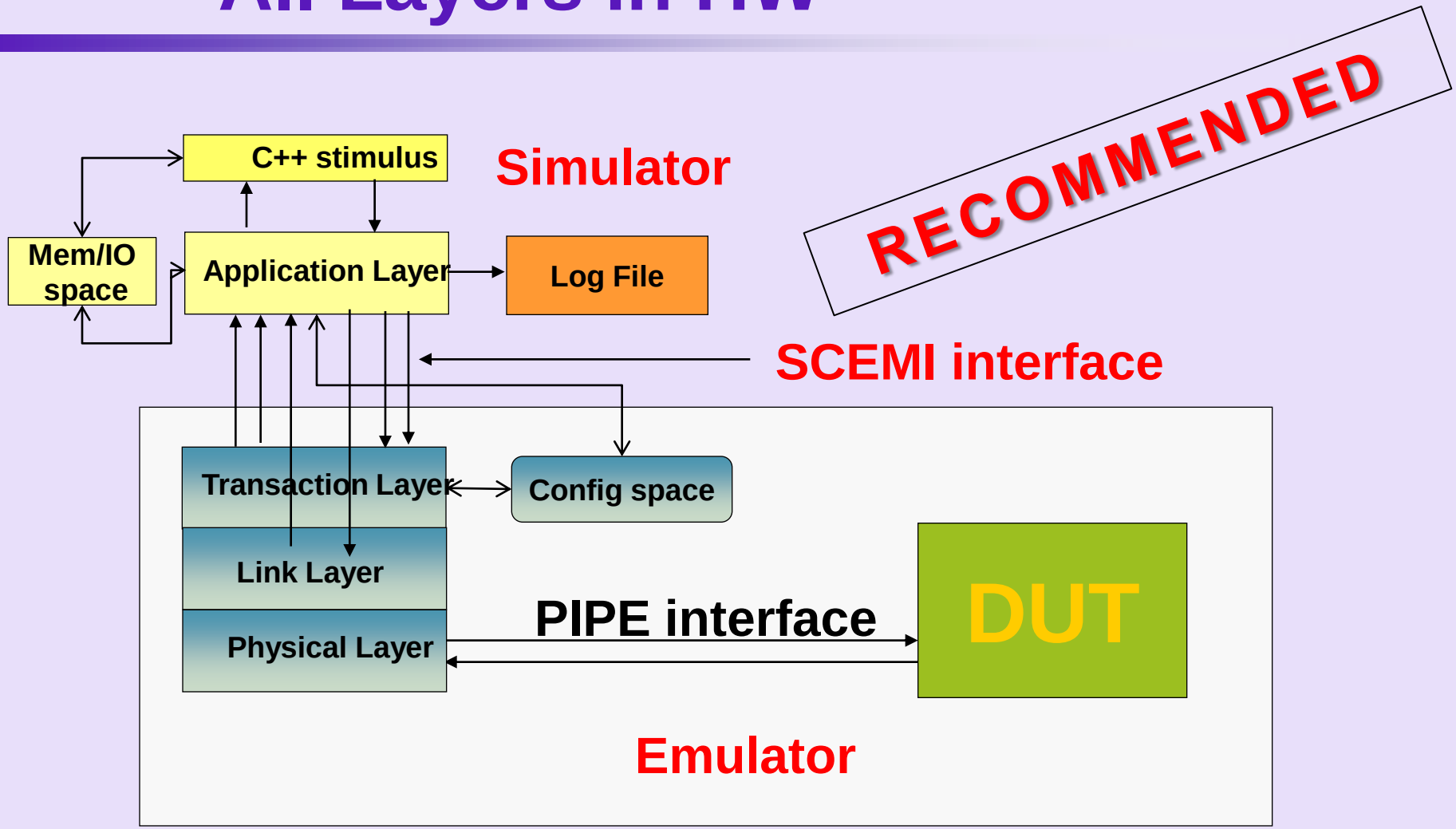
# PCI Express Acceleration VIP LL+ PL in HW



# PCI Express Acceleration VIP LL+ PL in HW

- Lesser control
  - ✓ Difficult to control TLP and DLLPs as they cross LL
  - ✓ Cannot control symbols and LTSSM states
- More speed
  - ✓ Simulator is inactive even during DLLP RX/TX
  - ✓ Flow control updates do not cause HW-SW sync
- HW-SW sync on every TLP
- SW-HW sync on Configuration transactions also
- Acceleration Speed 50-100X sim speed

# PCI Express Acceleration VIP All Layers in HW



# PCI Express Acceleration VIP All Layers in HW

- Least control
  - ✓ Difficult to control TLPs, DLLPs, TS, LTSSM
  - ✓ Difficult to control packet latencies
- Highest speed
  - ✓ HW-SW sync only on TLPs which need to access memory
  - ✓ Even configuration requests are handled in HW
  - ✓ Turn off debugging/logging for max gains
- >100x performance increase with the recommended architecture

# PCI Express Acceleration VIP Typical Execution flow

- During link training the SW layer can be inactive
- When L0 is reached the HW layer gives control to SW
- The test thread starts and sends transactions to HW
  - ✓ The test thread may also send Self configuration to its own configuration space
  - ✓ Or fill its own memory
- When a transaction is received a callback function is called in SW

# PCI Express Acceleration VIP Typical Execution Flow

- The test may have multiple test threads
  - ✓ *first_l0_thread*: for transactions after L0 is hit first
  - ✓ *second_l0_thread*: for transactions after L0 is hit post speed change (5GT/s or 8GT/s)
- Test may also read from its own memory/cfg space
- Test cannot contain any waits/clock controls
  - ✓ All waits/clock controls in HW
  - ✓ HW can send triggers to SW at appropriate times using SCEMI/DPI

# Passing Data Across HW-SW Boundary

- SCEMI stands for Standard Co-emulation Modeling Interface
- Provides mechanisms to pass data/control between HW and SW
- While sending from SW to HW the C layer calls a *`send`* function with a bit vector as the argument. The verilog bfm can receive it in an always block using *`receive`*.
- When sending data from HW to SW the verilog bfm calls a *`send`* function. The C proxy gets a callback notification and can call *`receive`* in the registered callback function.

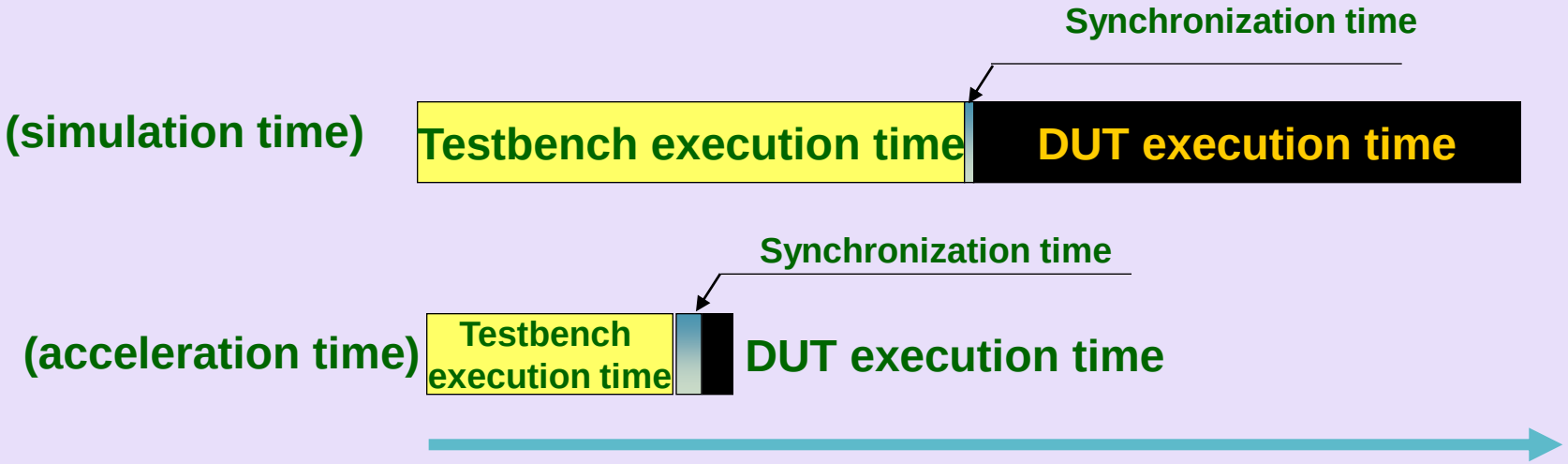
# Passing Data Across HW-SW Boundary

- Various mechanisms for data shaping, query functions and variable length messages
- SCEMI buffer depth/width is configurable for maximum acceleration
- SCEMI buffers can be configured to reactive/batching modes

# Maximizing Overall Performance

- Minimize time spent in the testbench on the host
- Optimize the use of hardware accelerator
- Reduce synchronization time
  - ✓ the times when the host (called SW side) and the accelerator (called HW side) synchronize passing data back and forth

# Maximizing Overall Performance

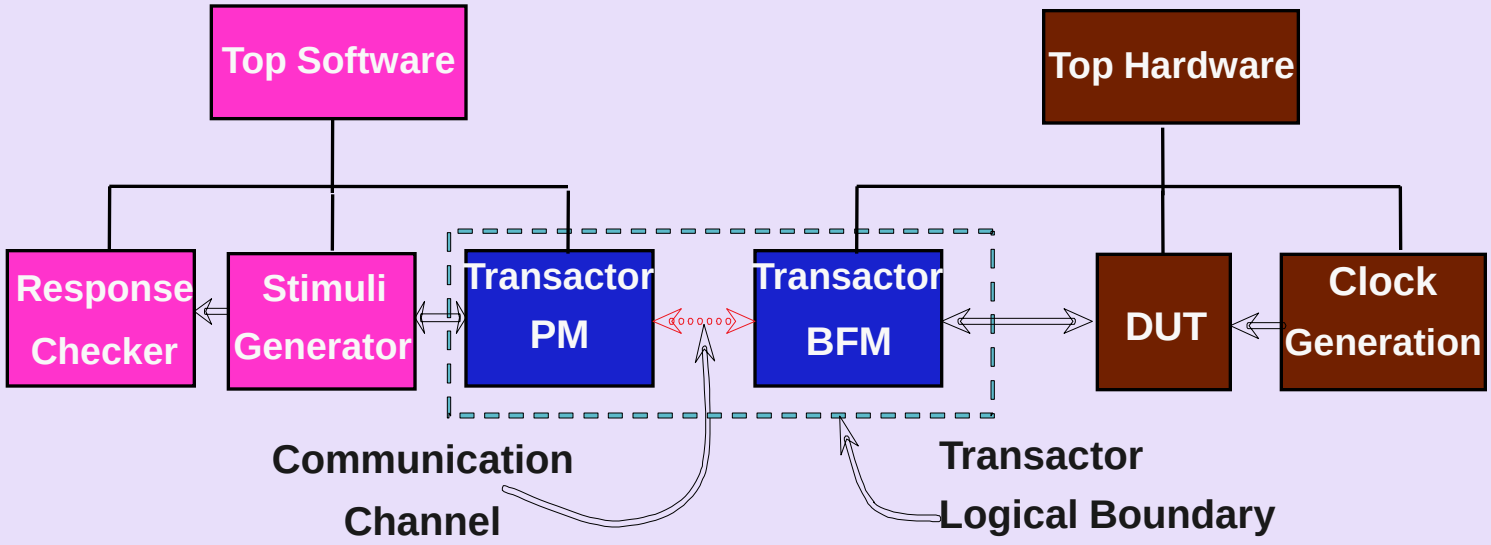



- An example:
  - ✓Lets say 10% is time spend in testbench and 90% in DUT
  - ✓Then in Acceleration the maximum gain will be 10X
  - ✓So to get to 100X the testbench time should be less than 1%!

Very severe constraints what part can be in SW!

# Principles to Maximize Performance

- The most active part of the testbench (BFMs/monitors) are running on the accelerator
  - ✓ BFMs and Monitors run at its peak emulation speed
- The BFMs and monitors encapsulate only the interface protocol specific knowledge
- The BFMs and monitors are the only testbench components requiring clocks
  - ✓ Avoiding synchronization with the SW side on every clock edge
- BFMs and monitors can provide/gather “transaction data” over multiple clock cycles
  - ✓ During these periods the HW side can run w/o interruption
- The testbench residing on the SW side is abstracted
  - ✓ Runs significantly faster with the BFM and the Monitor relegated to the HW side



# Principles to Maximize Performance (cont)

- Buffering mode can be used by non-reactive models
  - ✓ Controlled by the user for each communication channel
  - ✓ Supports batching multiple transactions into the buffer reducing the number of HW/SW side synchronizations
- HW/SW communication does not take place during idle periods
  - ✓ The more idle periods that occur on certain interfaces, the more performance improves
- Interaction between the HW side and SW side is fully asynchronous
  - ✓ Only happens when the HW side requests a new transaction or produces a new transaction
- Additional performance can be obtained when
  - ✓ the stimulus could be pre-generated
  - ✓ DUT response could be provided for post-processing checking

# Results Using Cadence's PCI Express Accelerated VIP

Cast Study #1

- Major systems and semiconductor company in Asia
- 10 Million gate design
- Performance achieved:
  - ✓ >400X speedup relative to simulation
  - ✓ 1.3 MHz execution speed to enable software development
  - ✓ Successfully completed long test unable to run in simulation
- Found several bugs not identified in simulation or FPGA prototype

# Results Using Cadence's PCI Express Accelerated VIP

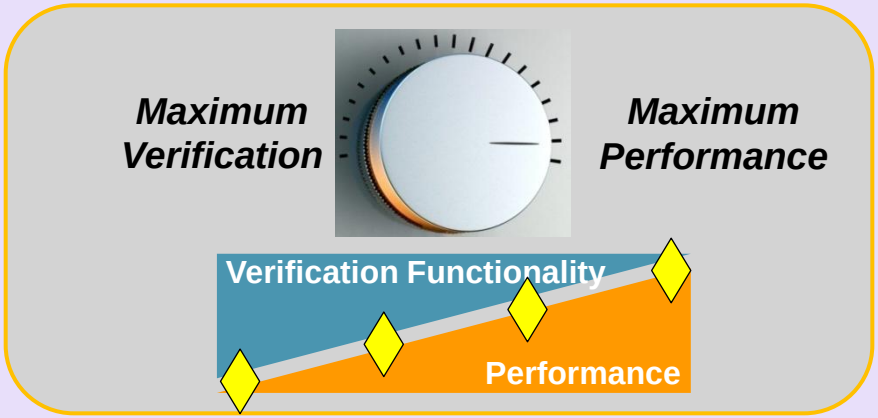
Cast Study #2

- Major systems and semiconductor company in Japan
- 28 Million gate design
- Performance achieved:
  - ✓ >500X speedup relative to simulation

# Speed vs Control – Deciding Acceleration VIP Architecture

- At block level verification speed is not the primary goal
- Need control to test every possible protocol scenario
- Need complex error injection, coverage, randomization, etc.

- At system level speed is of primary importance
- Need transaction level control. No need for byte/symbol level.



# Serial vs. PIPE for Acceleration

**SERIAL**

- This is the real interface between PCIe components
- Includes 8b-10b convention and serialization/deserialization
- A serial interface has a clock which is 10 times faster than core clock( clock at which the rest of the components function)
  - ✓ Acceleration degradation

**PIPE**

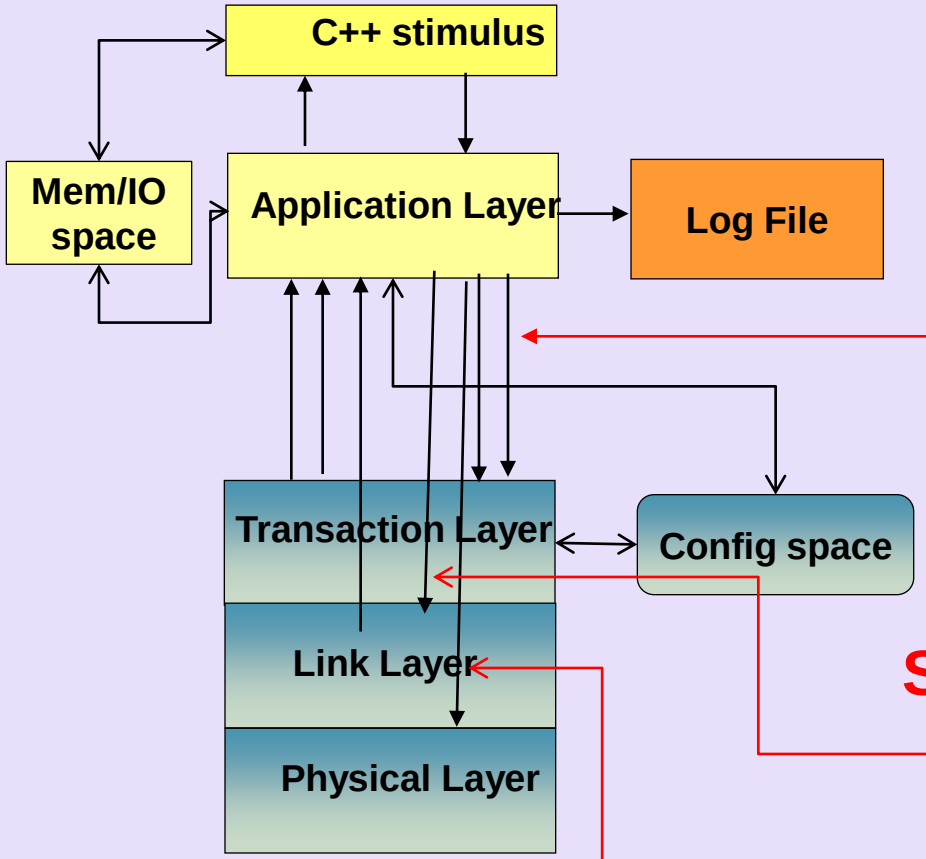
- Separation of Mac-Phy interface
- Parallel interface at symbol clock
- Clock frequency is equal to core clock
  - ✓ No Acceleration degradation

**PIPE interface is recommended to achieve acceleration speeds. SERIAL interface can be verified with a few short tests (i.e., only 1-2% of total tests).**

# Error Injection for PCIe Acceleration VIP

- Error injection is an important aspect of verification
- For Acceleration VIP errors have to be injected in rtl on emulator
  - ✓ Enable some basic error injection
  - ✓ Not as extensive as errors in simulation VIP
- Errors have to be injected at proper places
  - ✓ CRC error injection after CRC is calculated ( LL->PL)
  - ✓ TLP field errors after packet framing but before CRC calculation( TL->LL)
  - ✓ Can also insert errors on received packets to make VIP behave in a certain way

# Error Injection for PCIe Acceleration VIP




SCEMI interface

SCEMI channel for TLP framing error

SCEMI channel for CRC error

# Summary

- Acceleration is needed for SoC verification
- PCI Express Accelerated VIP enables verification of complex systems with PCI Express Interface
- Using accelerated VIP is different from simulated VIP—new assumptions and new objectives
- Reaching performance goals may require testbench restructuring

Thank you for attending the
PCI-SIG Developers Conference 2012

For more information please go to
www.pcisig.com