Solving Chip-Level Verification Challenges

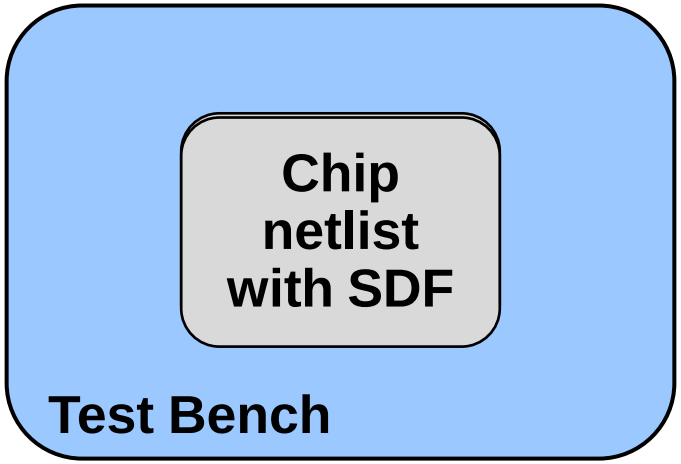
Balaji Kaliraj
Lavanya Rekha
Broadcom Corporation

Overview

- **Gate-level simulations**
  - Challenges seen
  - Test bench configurations
  - Simulation time
  - Timing closed PD blocks
  - Statistics
- **The Package-Aware Test Bench (PAT)**
  - Chip-Aware and Package-Aware Test Benches
  - The need for PAT
  - Methodology
  - Advantages

Gate-Level Simulations

- RTL simulations verify the functionality of the design.
- To verify interface timing across modules, gate-level simulations are required.
- Timing at clock domain crossings is verified.
- Standard Delay Format (SDF) annotated netlist is used instead of RTL.



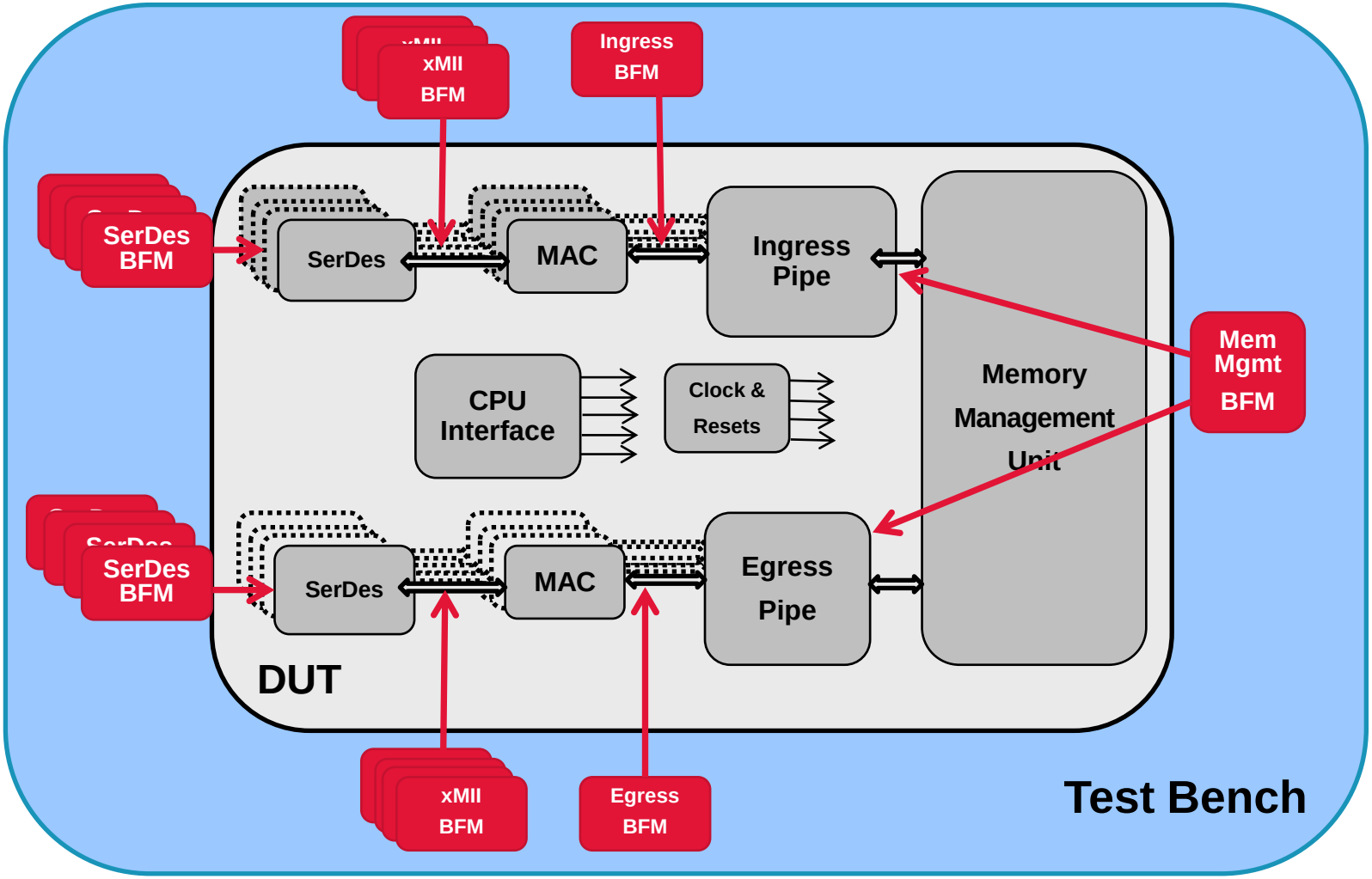
Challenges

- The design is huge.
- The tool may crash while trying to load the entire SDF annotated design.
- Running the simulation with the complete netlist is practically impossible.
- Simulation on full chip test bench takes approximately 40 CPS.

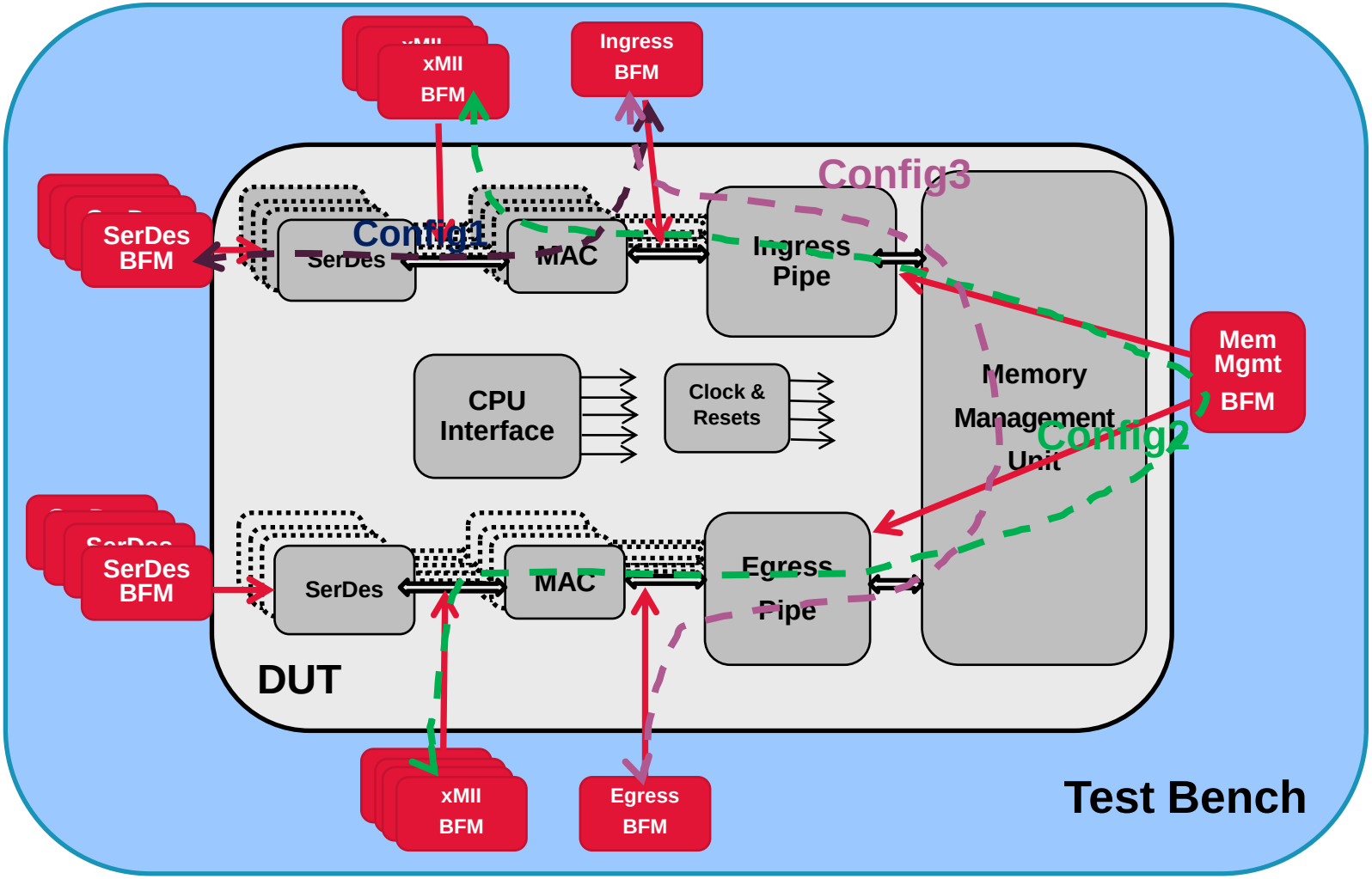
Test Bench Configurations

- These challenges are solved by having different flavors of test benches on which to run the gatesims.
- The design is functionally partitioned, with a netlist for blocks of interest and black-boxing of other modules.
- Decreases in overall design size make it easy to load the design and reduce simulation time.
- The configurations make sure every interface timing is covered.

Chip Level Test Bench



Three Test Bench Configurations



Timing Closed PD Blocks

- Having separate PD blocks (timing closed) from the Physical Design team helps in verifying timing between blocks (e.g., SerDes and MAC)
- We had seen cases where the SerDes netlist was not a part of the delivery from the PD team. SerDes RTL was used, but missed the verification with timing across interface, since the SerDes SDF(contains timing information) was not used.
- Now using a complete timing-closed wrapper with MAC + SerDes, the verification is more reliable and complete.

Simulation Time

- Simulation time for running gate-level simulations is exceptionally high.
- It can be greatly reduced with different test-bench configurations.
- Even after that, simulation takes a substantial number of hours (and sometimes days) to complete.
- The most time-consuming task in simulation is the initialization procedure.

Simulation Time (Cont.)

- The initialization procedure involves register/table configurations.
- Scripts have been developed to enable back-door slamming of register/tables in the netlist.
- Depending on the test bench configuration used, irrevelant tasks can be skipped.

Statistics

Test Bench Used	Chip RTL	Configuration A (Most of the Modules Included)	Configuration B (Lighter Configuration)
Compile Database Size (as per VCS Simulator)	4.2G	35G	10G

Test Bench Used	Chip RTL	Configuration A (Some Crucial Blocks Enabled)	Configuration B (Lighter Configuration)
Simulation Time (in Clocks per Second of the Simulator)	291 CPS	23 CPS	446 CPS

The Package-Aware Test Bench and Chip-Aware Test Bench

- A chip-aware test bench system requires clock/reset driver, bfm to drive/sample all signals/features that the chip supports.
- For each of the packages that a chip supports, some features are enabled and some are disabled. This is achieved in the chip-aware test bench using $plusarg/`ifndef and more manual effort.
- There is a need to test all different test benches separately.
- Increases the chances of *bugs* being left in certain test benches.
- A package-aware test bench requires only one test bench for all the package modes.
- The chip top is considered a die; logic to implement package(s) is instantiated in the test bench
- The common interfaces of all the packages are tied together and only one package mode is active at a time.

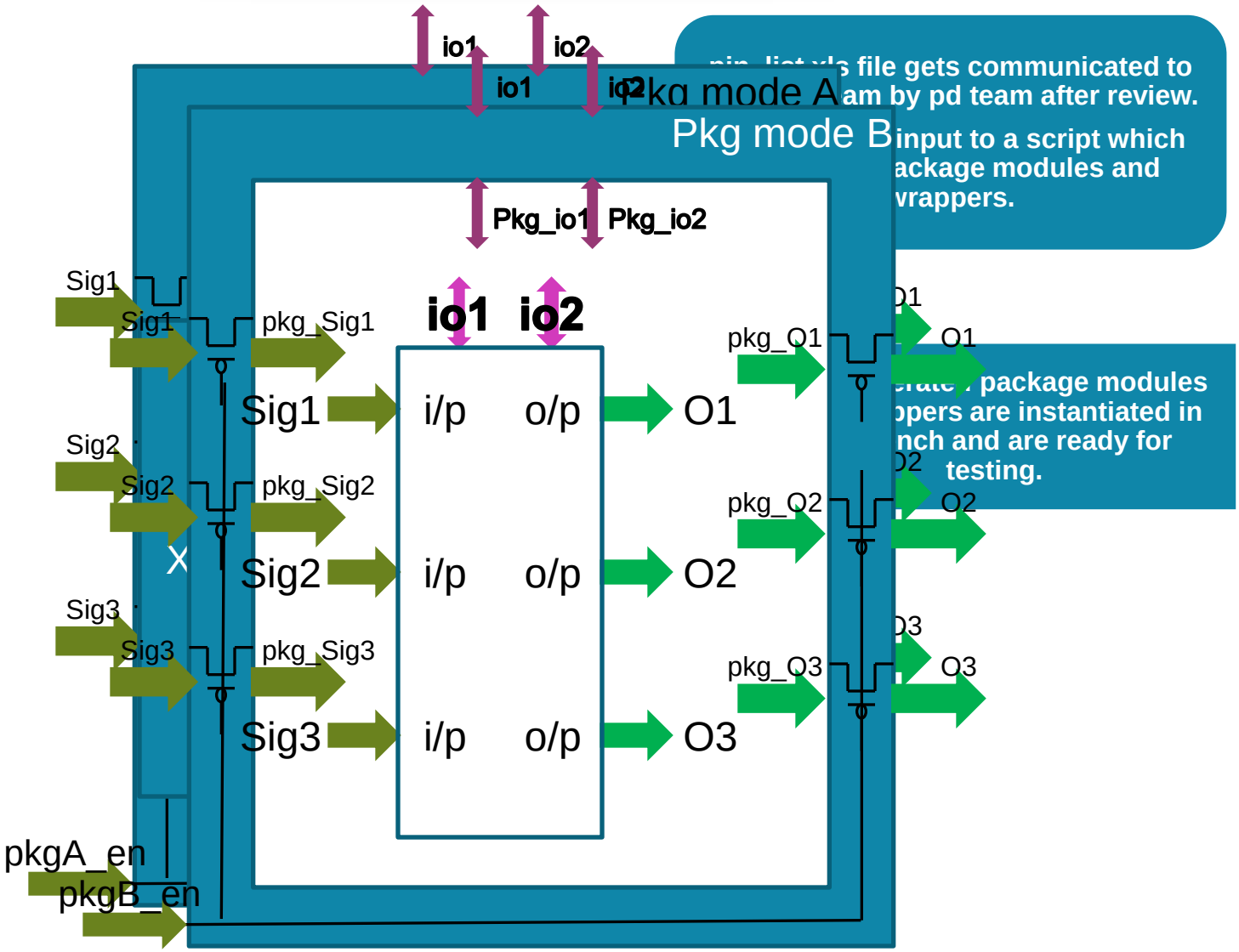
The Need for PATs

- With more packages added for every chip that a design has, it becomes necessary to implement package intelligence in test benches.
- There’s also a need to minimize the number of chip-level test benches.
- We must be able to run almost all the top-level test cases in all the packages that our chip supports.
- Chip initialization routines and configurations must be dependent on the package we select.

Methodology

- **Pass-transistors to interconnect test bench and DUT:**
  - Equivalent to interfaces in Sverilog.
  - Can be used for inputs/outputs and inouts, unlike the ***ternary operator*** (**? :**), which can be used only for inputs.
- **A new module "*chip_pkg*" is created based on the pinout extract provided to package team. This extract has the following interfaces:**
  - All inputs, outputs, and inouts of chip_top
  - All inputs, outputs, and inouts of chip_top, with their direction reversed
  - One input for package selection
  - Glue-logic to feed through signals based on the package_selection input
- **Chip_pkg instantiated multiple times (for as many packages as we support) in the test bench.**
- **Package selection input vector that is one-hot encoded can be triggered based on runtime plusargs.**

PAT: the Package-Aware Test Bench



PAT Advantages

- Run-time selection
- Integration of all packages into one test bench using pass transistors; this calls for *thin code.*
- Reduced *simulation overhead.*
- *Automated* method of creating/integrating packages minimizes the chances of leaving a bug behind.
- With this implementation, all test cases can be run in different package modes using run-time option with almost *no penalty* on *simulation run time*.
- Clocks, resets, and feature-enables that are supposed to be disabled/enabled for a certain package are automatically taken care of (through scripts).

Thank You!