Verifying a Low Power Design

Asif Jafri
Verilab Inc.

verilab

Agenda

- Introduction
- Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Agenda

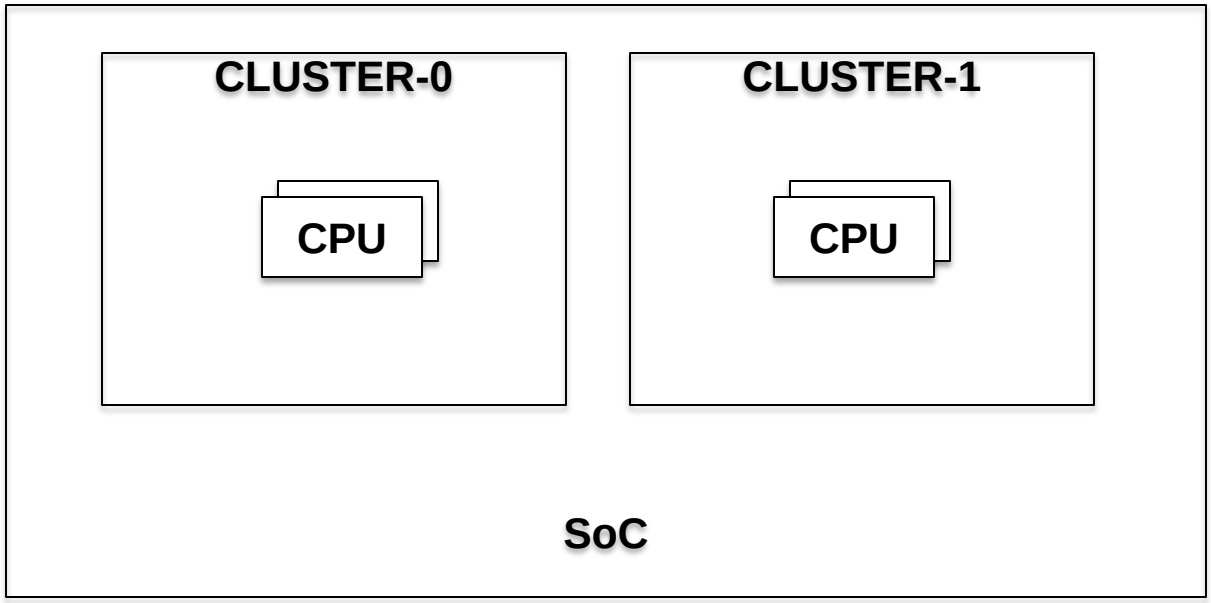
- Introduction
- Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Introduction

- Simulation based verification tools exist but what about low power verification
- Looking to start low power verification…

- Challenges created by some widely used low power design techniques
- Discuss Unified Power Format
- Extending your existing testbench
- Is this design verified

System Level



Cluster 0 Domain

Cluster 1 Domain

SoC Domain

CLUSTER-0	CLUSTER-1	SoC
OFF	OFF	OFF
ON	ON	ON
OFF	ON	ON
ON	OFF	ON

Asif Jafri

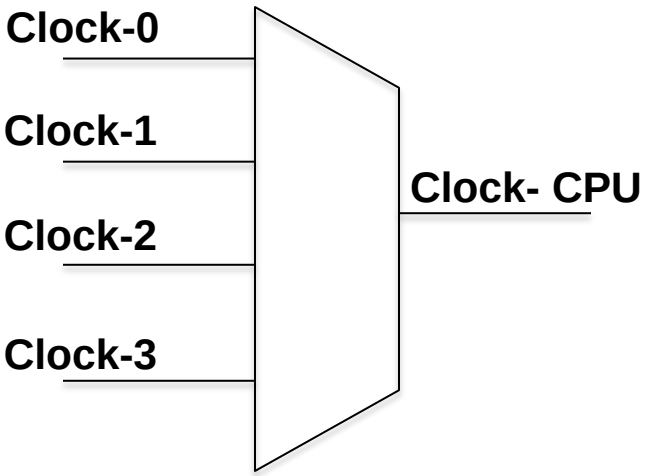
Agenda

- Introduction
- Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Frequency Scaling

$$P = ACV^2F$$

P: Power consumed
A: Activity factor
C: Switched capacitance
V: Supply voltage
F: Frequency



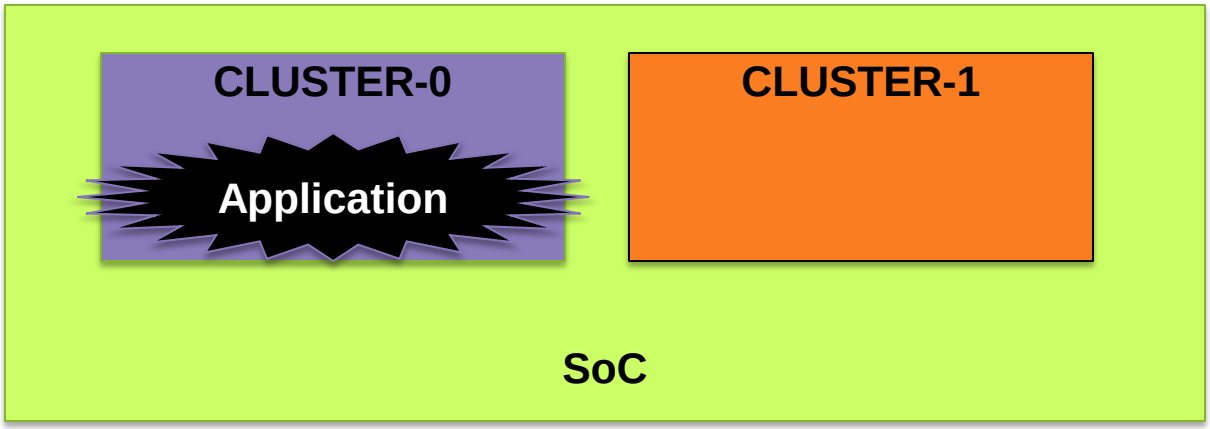
- Frequency control registers in SoC domain
- Directed tests used to test all frequencies
- Power Gating

CLUSTER-0	CLUSTER-1	SoC
OFF	OFF	OFF
ON	ON	ON
OFF	ON	ON
ON	OFF	ON

Agenda

- Introduction
- Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

big-LITTLE Switching



- Switching between high-end performance and energy efficiency
- Requires virtualizer software to be run
- Running in simulation would take multiple days
- Best tested in Emulation environment
  - Emulation used for OS bring up
  - Extremely fast proof of concept

Asif Jafri

Agenda

- Introduction
- Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Power Aware Simulations

- Unified Power Format (UPF)
  - IEEE standard 1801-2009, based on Accellera's unified power format
  - Describes low power intent of a design
  - Input to multiple tools
    - Simulation
    - Formal Verification
    - Synthesis
    - Place-and-route

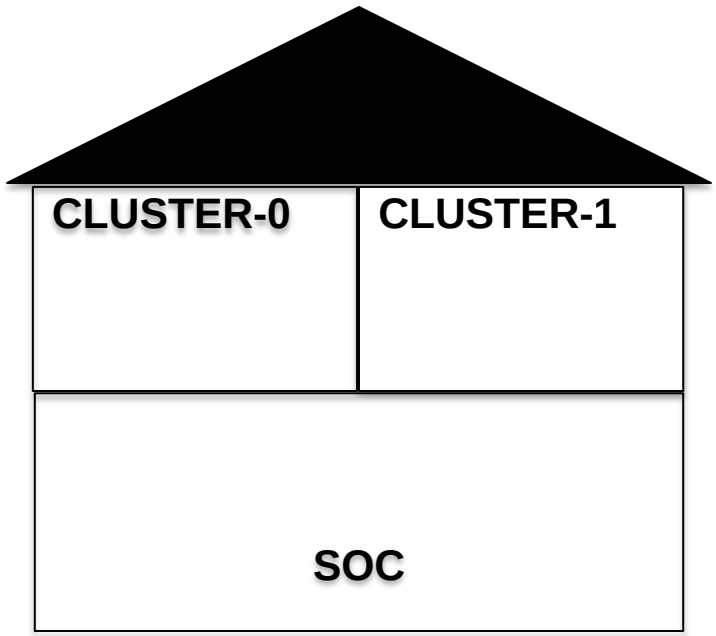
Unified Power Format

- Supply Ports
  - Main Supply to the house
  - Provides power state and voltage value
  - Power state (ON, OFF)
  - Voltage (0.8v 1.0v etc.)


```
create_supply_port VDD_CLUSTER0
create_supply_port VSS_CLUSTER0
```

Unified Power Format

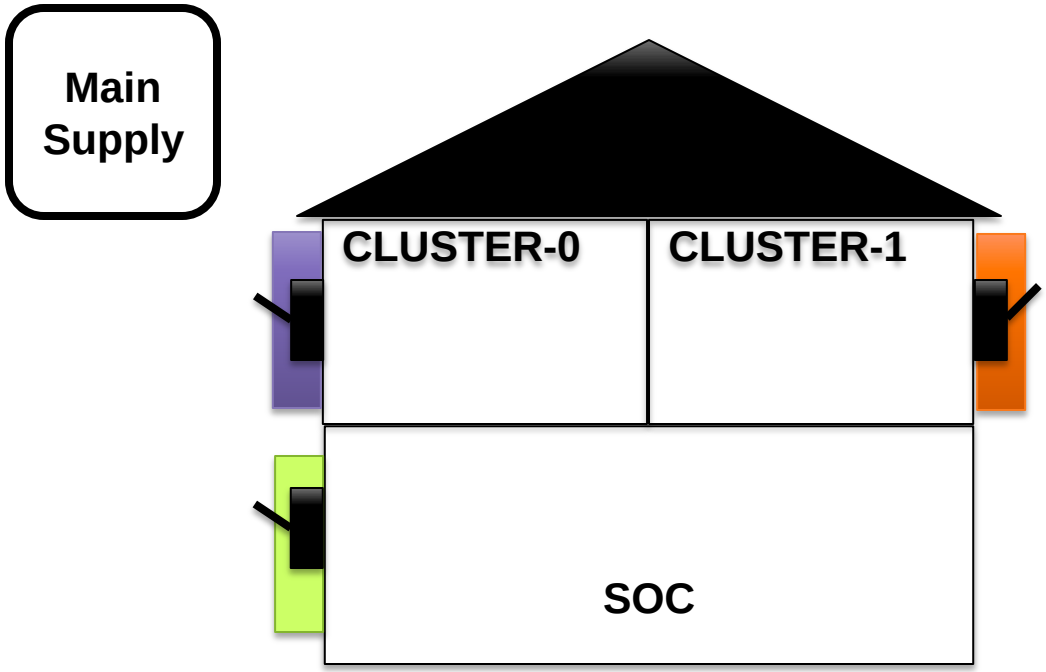
- Power Domains
  - The various rooms are the domains of the house




```
create_power_domain TOP
create_power_domain CLUSTER0
create_power_domain CLUSTER1
```

Unified Power Format

- Power Switches
  - Circuit breakers to various rooms.
  - They control various components in the room



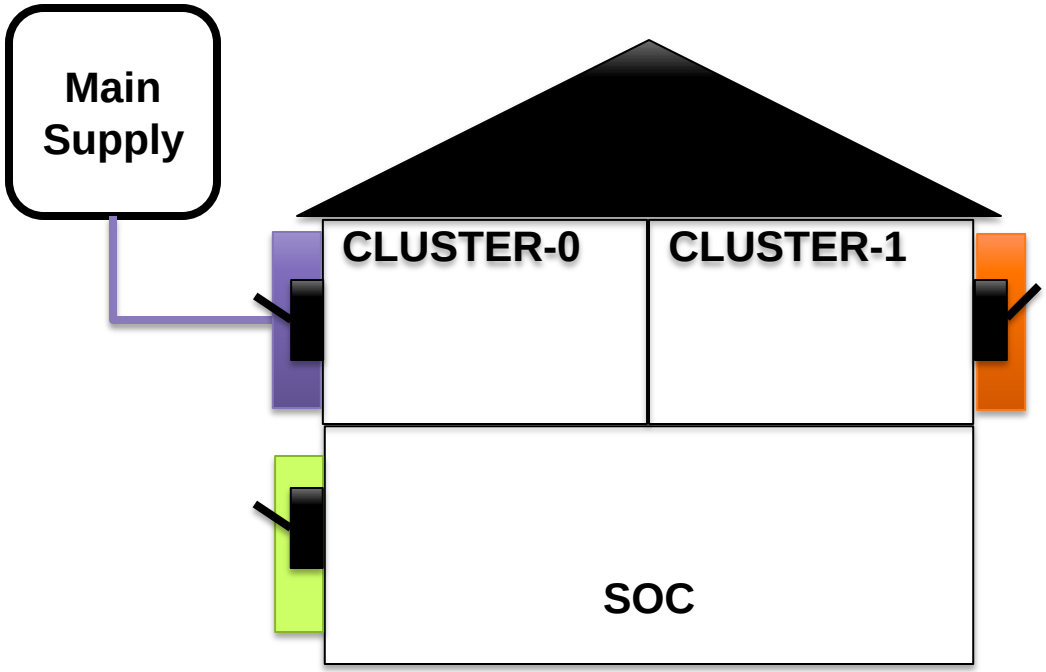

```
-create_power_switch  main_sw \
            -domain CLUSTER0 \
            -input_supply_port {in VDD_CLUSTER0} \
            -output_supply_port {out VDD_O_CLUSTER0} \
            -control_port {inst_on inst_on} \
            -on_state {state2001 in {!list_on}}
```

Asif Jafri

verilab

Unified Power Format

- Supply Nets
  - Wiring between main supply and breaker

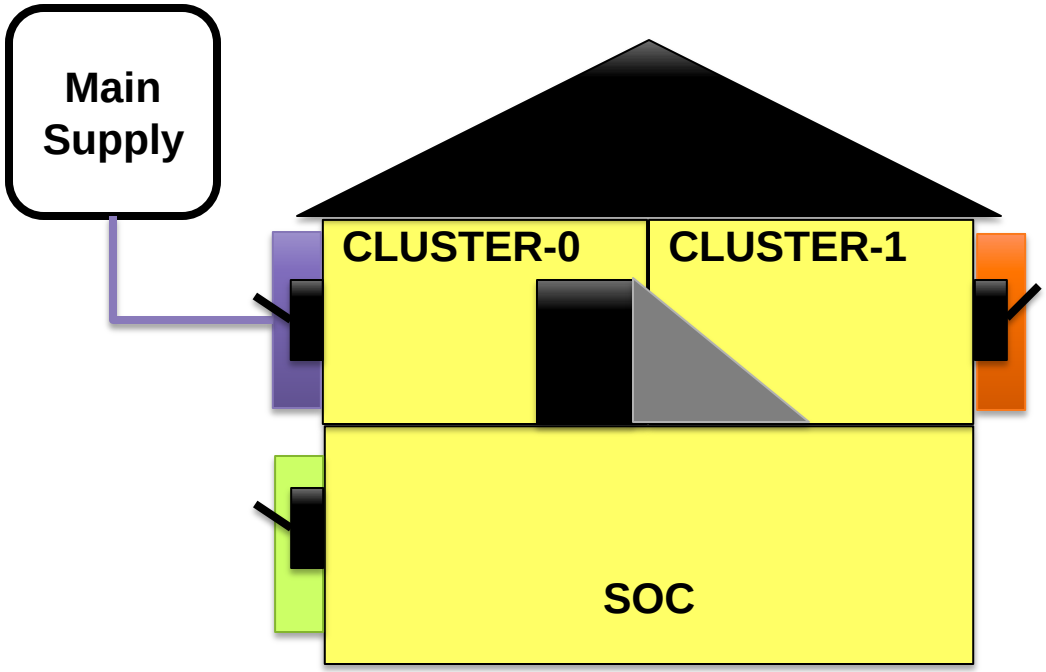



```
create_supply_net   net_VDD_CLUSTER0 -domain CLUSTER0
connect_supply_net net_VDD_CLUSTER0 -ports   VDD_CLUSTER0
```

Asif Jafri

Unified Power Format

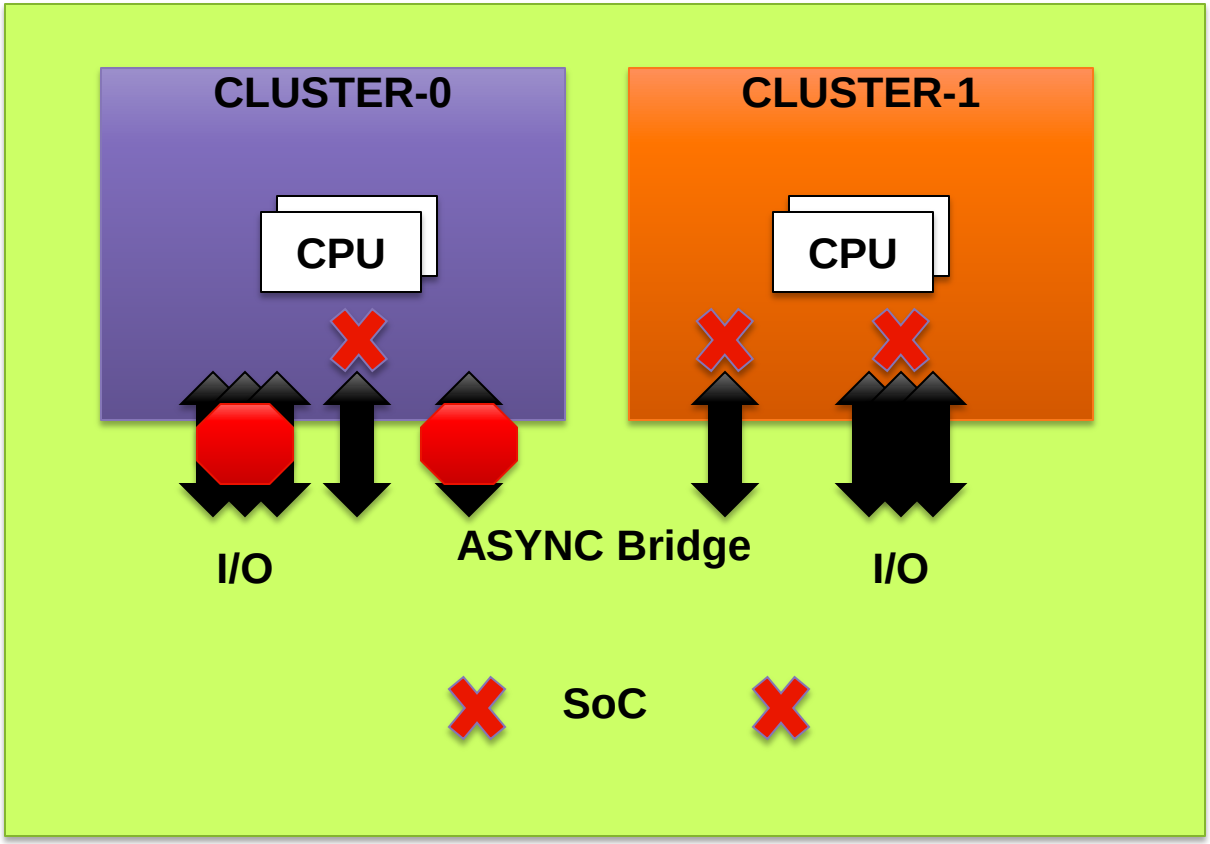
- Isolation
  - Doors and blinds




```
set_isolation CLUSTER0_SIG -domain CLUSTER0
```

Asif Jafri

Power Aware Testing



Unified Power Format

- Hierarchical scope for UPF files

```
set_scope <PATH TO UPF FILE>
```

```
load_upf   $env(DESIGN_ROOT)/upf/CLUSTER0.upf
           -scope <path to cluster0 core>
```

- VCS Command Line (Native Low Power)

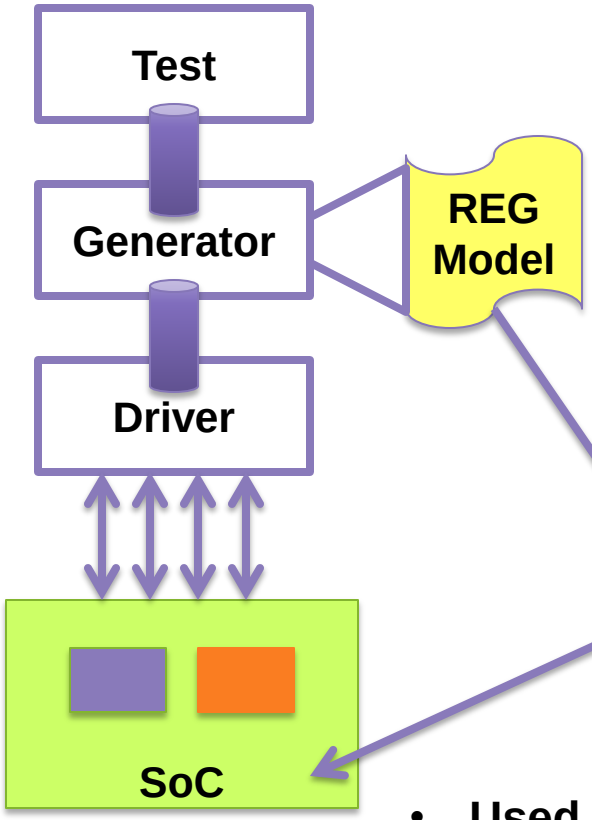
```
vcs –upf <top level upf file>
    -power_top <module scope for top level upf>

    -sverilog –debug_all –f …
```

Agenda

- Introduction
- Dynamic Voltage Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Testbench



Power Controller

```
import UPF::*;

task power_up();
  power up sequence;
  supply_on ("VDD_CLUSTER0", 1.2);
endtask

task power_down();
  power down sequence;
  status_reg.READ();
  supply_off("VDD_CLUSTER0");
endtask
```

- Used to control and monitor system from outside
- Used to switch OFF or switch ON power supply to supply ports

Asif Jafri

Power Down Sequence

- Save context
- Clear and Invalidate caches
  - Prevent any new data cache snoops or data cache operations from other processors
- Put CPU's in standby mode
  - The processor waits for all instructions in the processor to complete before entering idle or low power state

Power Down Sequence

- Power down async bridge — Controller
  - Write to control register in SoC domain
- Assert reset to cluster — Controller
  - Write to control register in SoC domain
- Isolate cluster being powered down — Controller
  - Write to control register in SoC domain
- Remove power — Controller
  - supply_off(“VDD_CLUSTER0”);

Power Up Sequence

- Enable power
  - supply_on (“VDD_CLUSTER0”, 1.2);
- Supply clocks to cluster
  - Write to control reg in SoC domain
- Remove isolation
  - Write to control reg in SoC domain
- Remove reset
  - Write to control reg in SoC domain
- Run boot code to restore context

Controller

Controller

Controller

Controller

Test

Agenda

- Introduction
- Dynamic Voltage Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Tests

- Power up and power down test for each cluster
  - Testing basic power down and power up sequences
- Power up and power down with context save and restore
  - System can indeed be brought back to state before power down
- Random Power up and down
  - Testing async bridges
  - Corner cases

CLUSTER-0	CLUSTER-1	SoC
OFF ✓	OFF ✓	OFF ✓
ON ✓	ON ✓	ON ✓
OFF ✓	ON ✓	ON ✓
ON ✓	OFF ✓	ON ✓

Agenda

- Introduction
- Dynamic Voltage Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Check that …

- …clocks are switched off in powered down cluster

```
property check_clk;
  @(Cluster0_clk) (cluster0_pwr_ctrl === 1'b1);
endproperty
```

In SoC domain

- …resets are asserted in powered down cluster

```
property check_rst;
  @(Cluster0_rst) (cluster0_pwr_ctrl === 1'b1);
endproperty
```

Check

- All existing assertions in clusters were used
- Assertions in powered down cluster automatically disabled

Agenda

- Introduction
- Dynamic Voltage Frequency Scaling
- big.LITTLE switching
- Power Aware Simulations
  - Unified Power Format
  - Testbench
  - Tests
  - Checks
- Summary

Summary

- Various techniques used to reduce power consumption
- Power Aware Simulations useful for testing isolation, power gating
- big-LITTLE processing best tested by emulation
- Re-use most of your existing testbench
- Make use of directed tests, assertions, functional coverage and code coverage to prove functionality

Thank You

Q&A

Asif Jafri
verilab