

Low-Power Verification at Gate Level for Zen Microprocessor Core

Baosheng Wang, Keerthi Mullangi – AMD

Raluca Stan, Diana Irimia – AMD Service Provider (Silicon
Service SRL)

Outline

- Introduction
 - Key targeted low power features
- Power Aware Gatesim (PAG)
 - Advantage of co-sim models
- Reset Checker at Gates
- Real Power Up Sequence
- Automation
- Results
- Summary

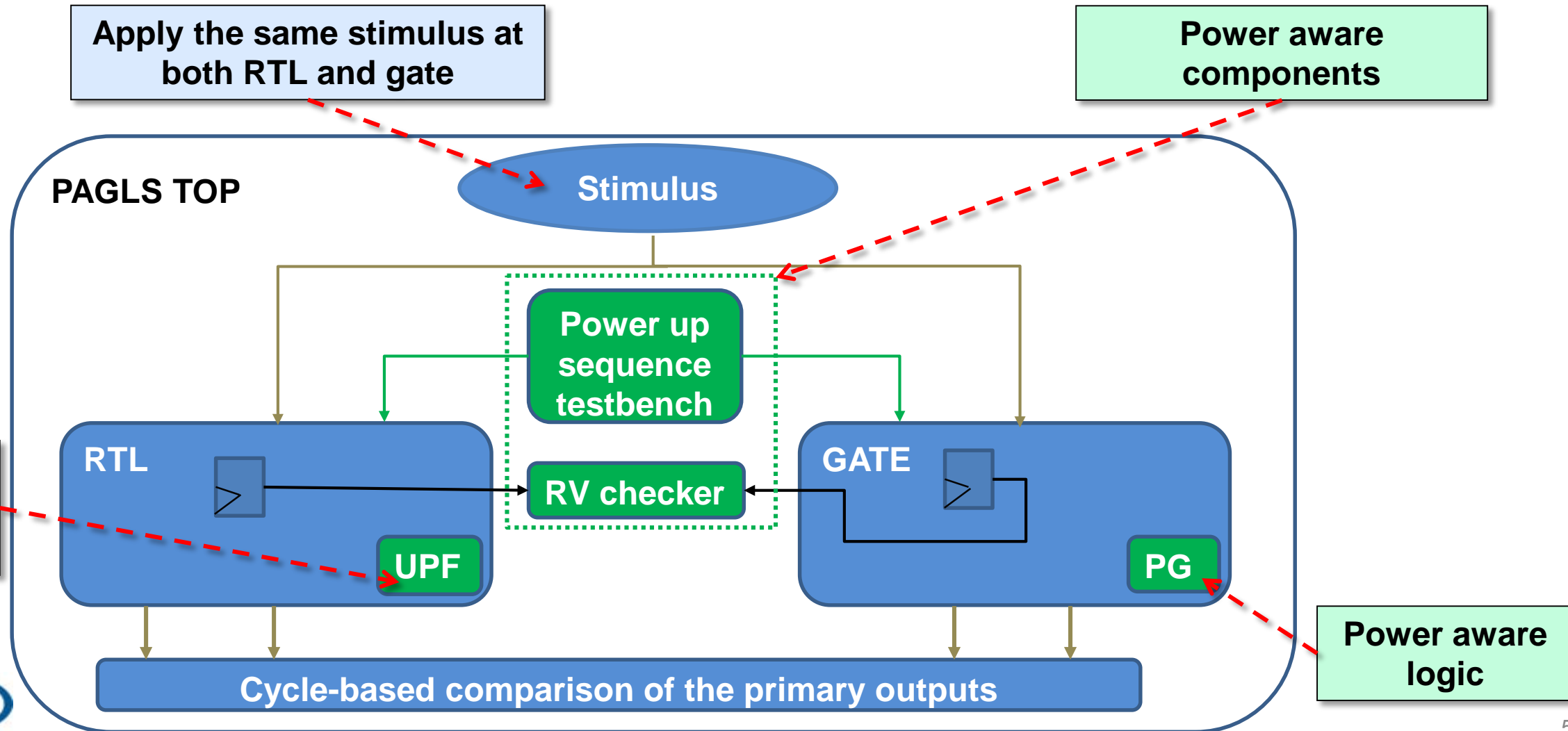
Introduction

- AMD “Zen” microprocessor
 - High-performance and low power x86 core
 - Energy efficient 14LPP FinFET
 - 1.4B transistors core complex unit
 - Shared 8MB L3 cache and four cores
- Low-power features verified at Gate level
 - Scan shift reset
 - Power gating
 - Power on clock/reset

The necessity of low power verification at Gate level

- RTL verification limitations
 - Scan chain is *partially* modeled
 - Reset logic *not fully* verified
 - No power supply ports physically inserted
- Traditional Logical Equivalence Check (LEC) limitations
 - Scan chains are inserted *after* the LEC verification completes
 - Performs scan chain check *only at* the macro level
 - *Cannot* detect design optimization

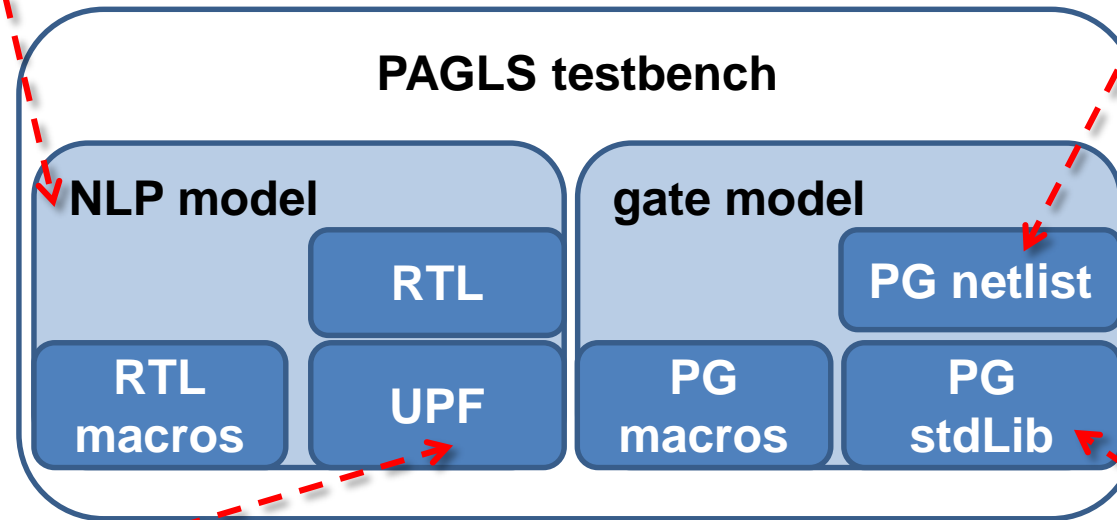
Power Aware Gate Level Simulation (PAGLS) Environment



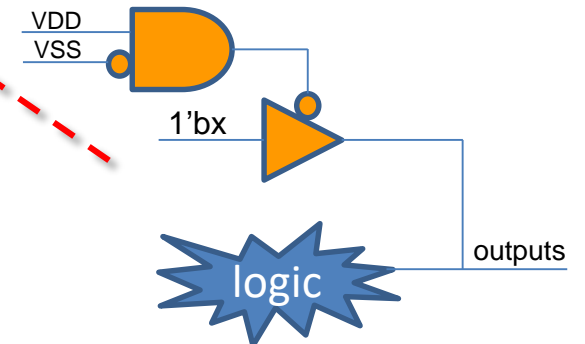
PAGLS configuration model

NLP tool is an external irritator used in UPF-based verification at the RTL level

PG netlist is the result of the conversion of the RTL design to gate-level description via a synthesis tool
All power ports are inserted and connected



UPF describes the power aware logic and reflects the power intent of the design



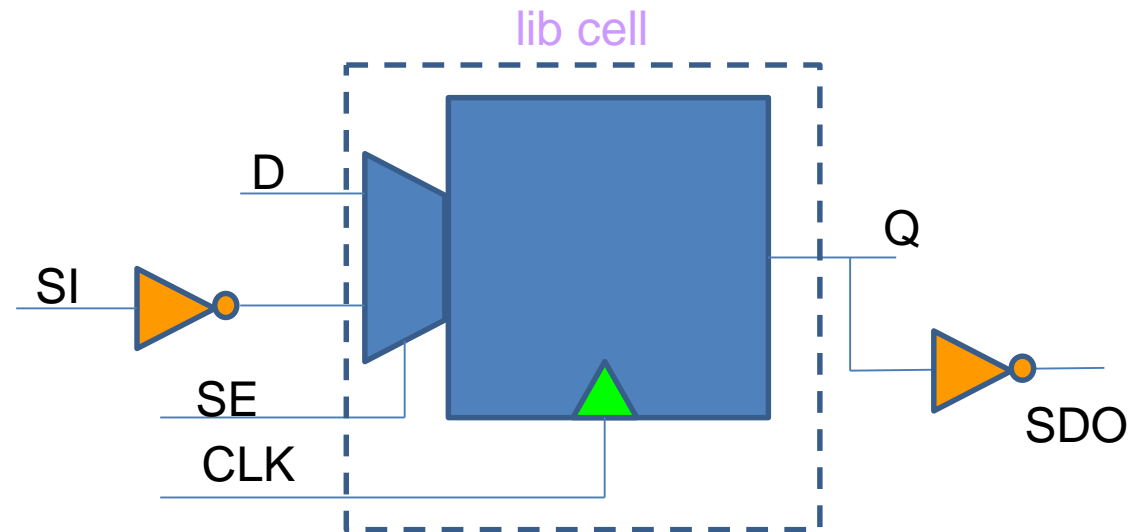
Advantages of the co-simulation environment

- Reuse of infrastructure – *same verification tools*
- Reuse of stimulus – *reproduce any scenario*
- Stimulus sampling – *testcase variety and quality*
- Plug-and-play execution and reconfiguration – *same built-in template*

Scan Shift Reset Design Methodology

- Scan shift reset (SSR)
 - Performs a synchronous reset of the design
 - All the scannable flops in the design are in a known state
- Example of reset1 flop

```
dff #(.RVAL(1)) dff_inst ( ... );
```



Reset checker methodology

- Reset value (RV) checker
 - Performs the flop checks *inside* the macros
 - Compares the RV of all the scannable flip-flops after SSR
 - Detects inverters inserted during design optimization into scan chains

| RTL .rv parameter | LEC mapping | RTL equals Gate | RTL non-equals Gate |
|-------------------|-------------|-----------------|---------------------|
| 0 | direct | OK | mismatch |
| 0 | inverted | mismatch | OK |
| 1 | direct | mismatch | OK |
| 1 | inverted | OK | mismatch |

RV checker sample code

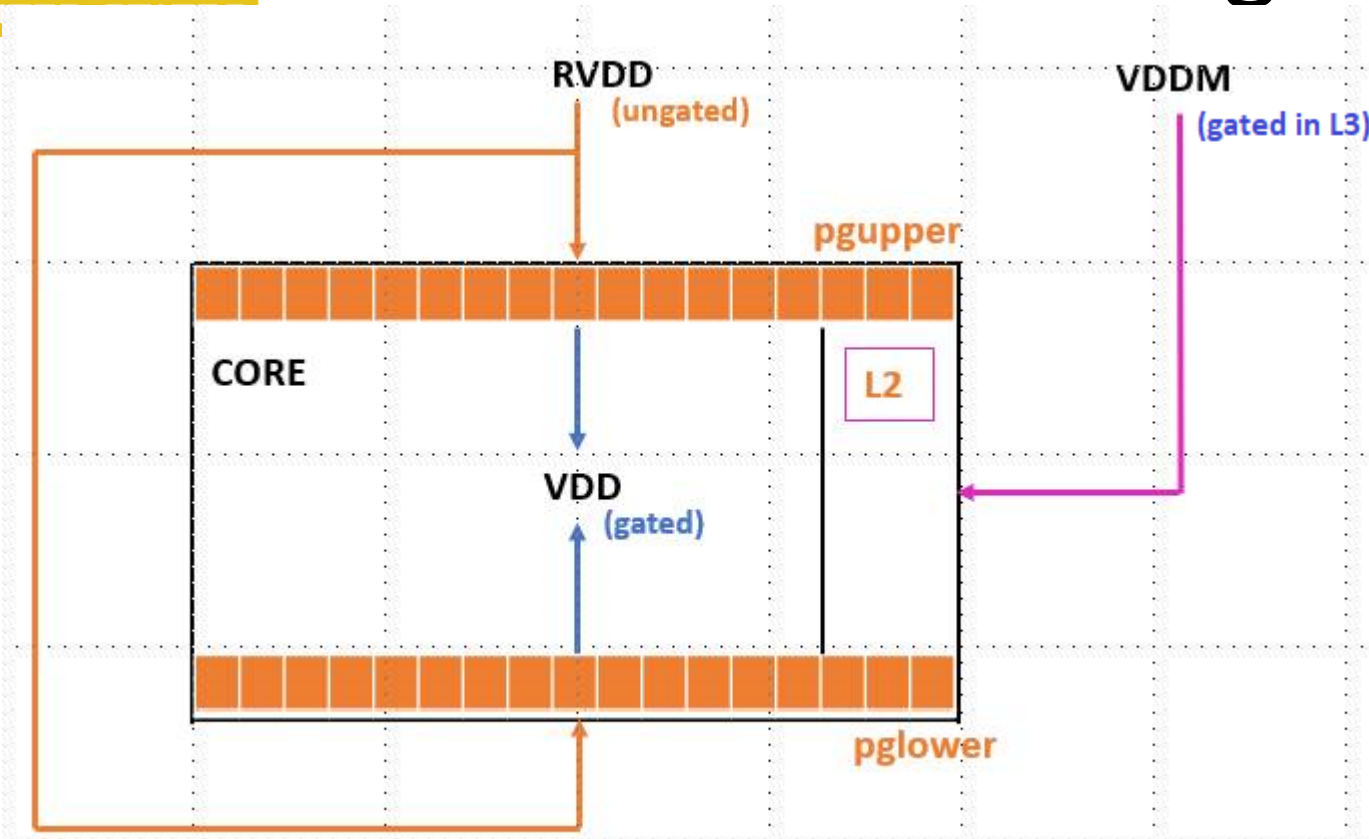
direct mapping
between state point
for “QB” output port

```
always @(negedge SSE) begin
    if ( GATE_FF_instance0.QB !== RTL_FF_instance0.dff_q) begin
        $display (":RV checker %m RV Value mismatch, RTL
statepoint CAN'T BE DIFF as Map Direct GATE statepoint upon
scan);
        rv_check_err_cnt = rv_check_err_cnt + 1;
    end else
        $display ("Map Direct RTL GATE statepoint matched");

    if ( GATE_FF_instance1.Q === RTL_FF_instance1.dff_q) begin
        $display (":RV checker %m RV Value mismatch, RTL
statepoint CAN'T BE SAME as Map Invert GATE statepoint upon
scan reset");
        rv_check_err_cnt = rv_check_err_cnt + 1;
    end else
        $display ("Map Invert RTL GATE statepoint matched");
end
```

inverted mapping
between state point
for “Q” output port

Power Gating Design in Zen



- Three Power Domains
 - RVDD: raw power supply
 - VDD: gated power supply
 - VDDM: gated power supply, for memory array retention only
- Power gating is achieved in UPF with power_switch
- Ring Style
- Digitally Controlled

```
create_power_switch pgheader -domain PD_TOP
-input_supply_port {RVDD RVDD}
-output_supply_port {VDD VDD}
-control_port {RUN_X CONTROL_SIGNAL}
-on_state {vdd_on RVDD {!RUN_X}}
-off_state {vdd_off {RUN_X}}
```

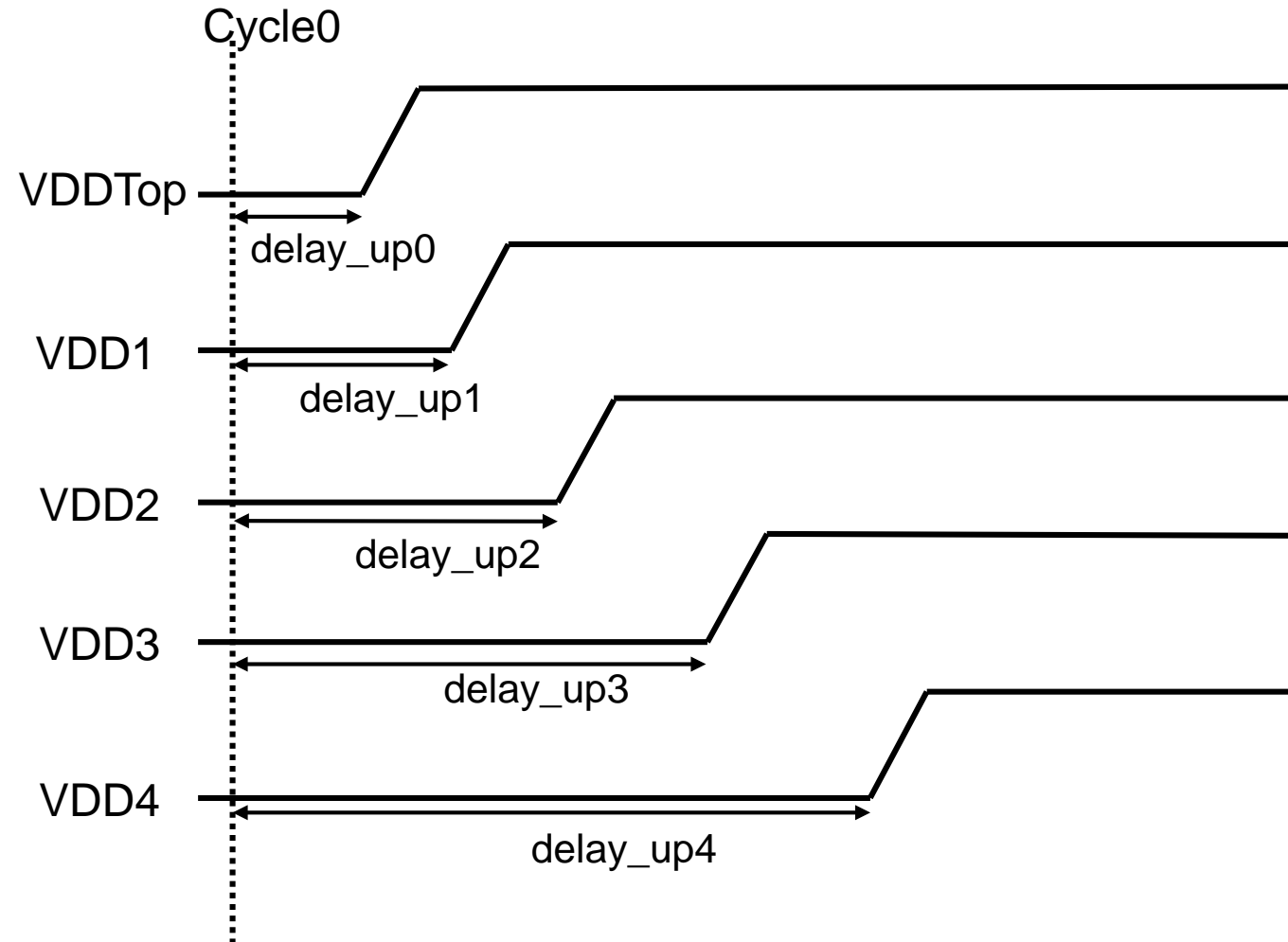
Real Power Up Sequence

- In a design, all the power rails won't power up from 0 time
 - Randomization sequence of the power supplies is defined according to Spec
- Clocks are verified during power up
 - how mesh clocks make the transition from clock generator to local clocks
 - how clocks propagate when the power supply is high
- Also, checks if level shifters are isolated correctly

| Power rail | Package power supply | Comment |
|------------|----------------------|--|
| PowerRail1 | VDDTop | VDDTop has to be powered up before all the power supplies |
| PowerRail2 | VDD1, VDD2 | VDD1 and VDD2 can be powered up at any sequence after PowerRail1 |
| PowerRail3 | VDD3, VDD4 | VDD3 and VDD4 can be powered up at any sequence after PowerRail2 |

Real Power Up Sequence

- Power ramps up gradually
 - Analog and mixed clock macros in the Zen core demand this process in simulation models
 - Achieving this by adding a delay element in the test sequence
 - Starting from cycle0, the delay numbers are randomized ensuring there is a gap of 20-30 clock cycles between each rail ramp up.



Power Up Sequence Sample Code

```
task rampUp_Seq1();
fork
#(delay_up0)      set_vddtop  = 1'b1;
#(delay_up1)      set_vdd1   = 1'b1;
#(delay_up2)      set_vdd2   = 1'b1;
#(delay_up3)      set_vdd3   = 1'b1;
#(delay_up4)      set_vdd4   = 1'b1;
join
Endtask

always_comb @* begin
VDDTop  = (set_vddtop== 1'b1) ? 1'b1 : 1'b0;
VDD1    = (set_vdd1 == 1'b1) ? 1'b1 : 1'b0;
VDD2    = (set_vdd2 == 1'b1) ? 1'b1 : 1'b0;
VDD3    = (set_vdd3 == 1'b1) ? 1'b1 : 1'b0;
VDD4    = (set_vdd4 == 1'b1) ? 1'b1 : 1'b0;
end
```

Test bench

Note: delay_up0<delay_up1<...<delay_up4

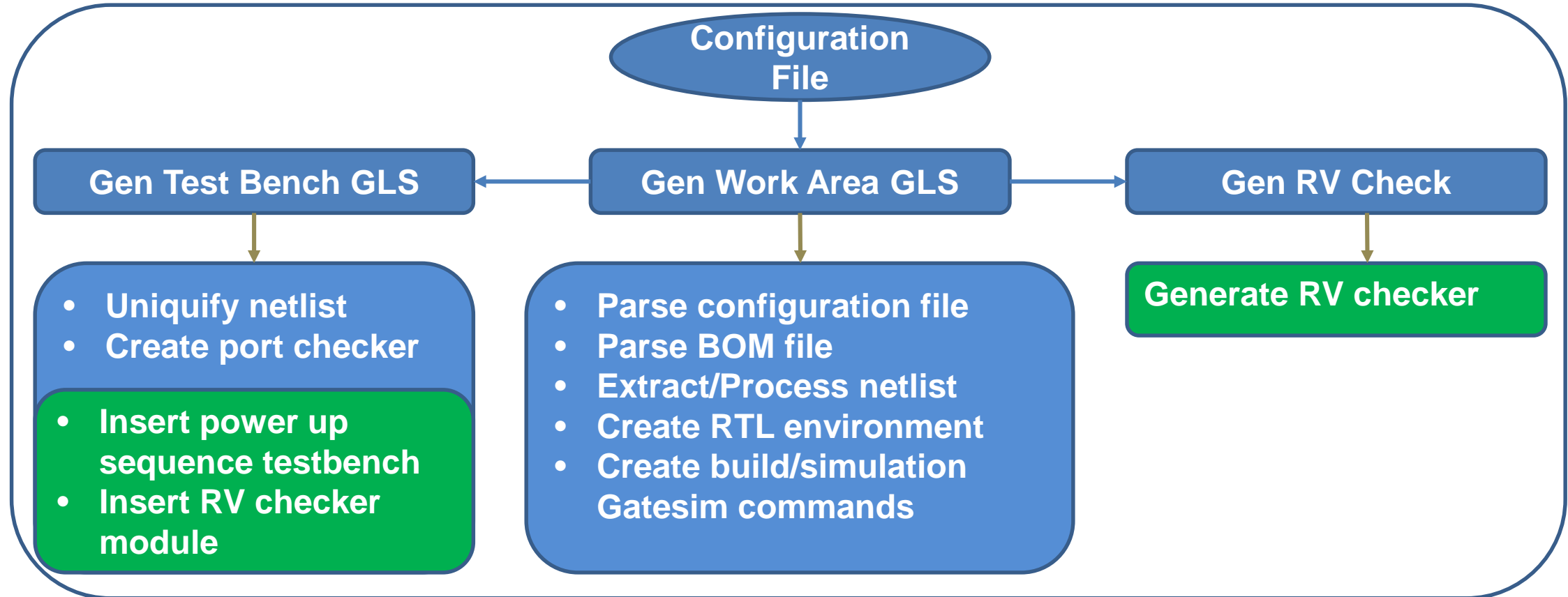
| | | | | |
|--------------------|-----------|--------|---|--|
| connect_supply_net | SN_VDDTop | -ports | {tb_nlp_power_seq_inst/VDDTop VDDTop } | |
| connect_supply_net | SN_VDD1 | -ports | {tb_nlp_power_seq_inst/VDD1 VDD1 } | |
| connect_supply_net | SN_VDD2 | -ports | {tb_nlp_power_seq_inst/VDD2 VDD2 } | |
| connect_supply_net | SN_VDD3 | -ports | {tb_nlp_power_seq_inst/VDD3 VDD3 } | |
| connect_supply_net | SN_VDD4 | -ports | {tb_nlp_power_seq_inst/VDD4 VDD4 } | |

UPF

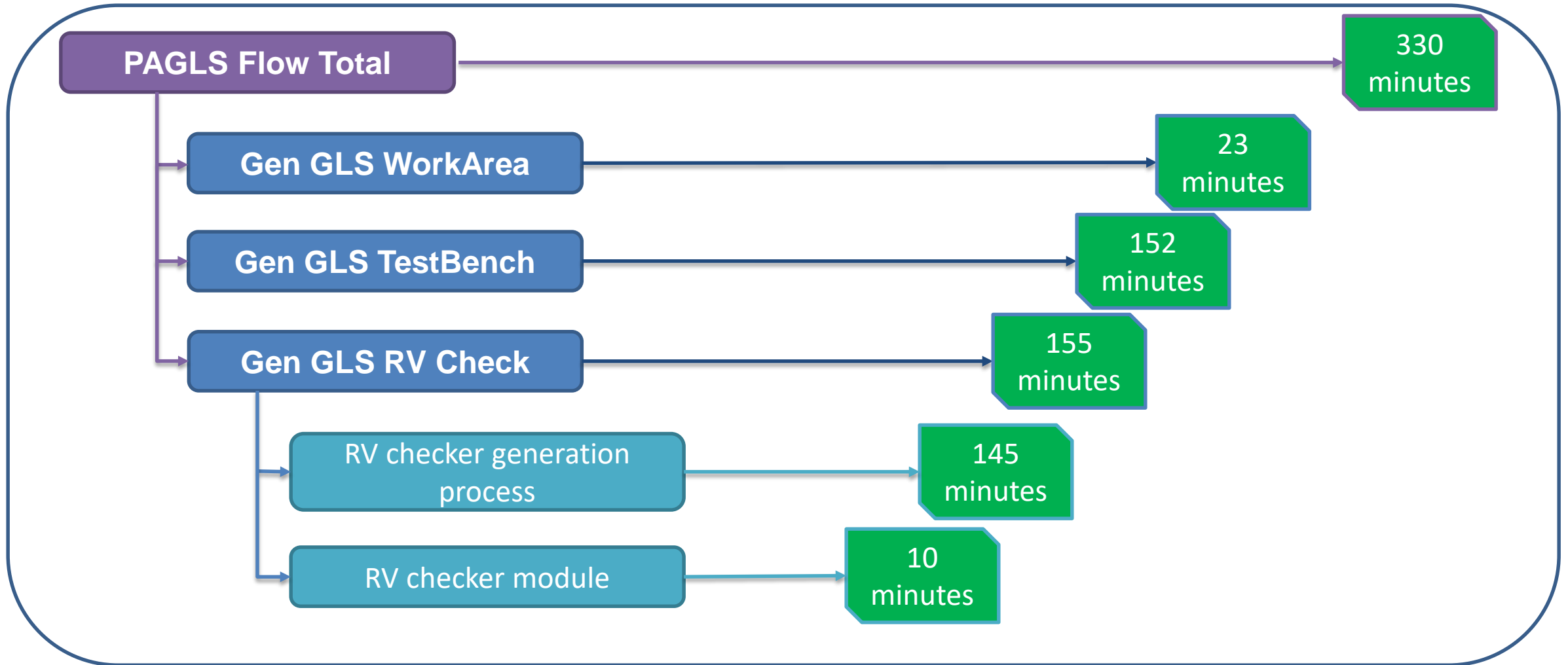
Real Power Up Sequence at Gate Level

- Re-use of NLP test bench in PAGLS
- Testbench Drives the power ports in netlists and UPFs
- Randomization and delays are applied in PAGLS
- Verify the clocks on powerup , isolation strategies

PAGLS Flow Automation



PAGLS Flow Automation



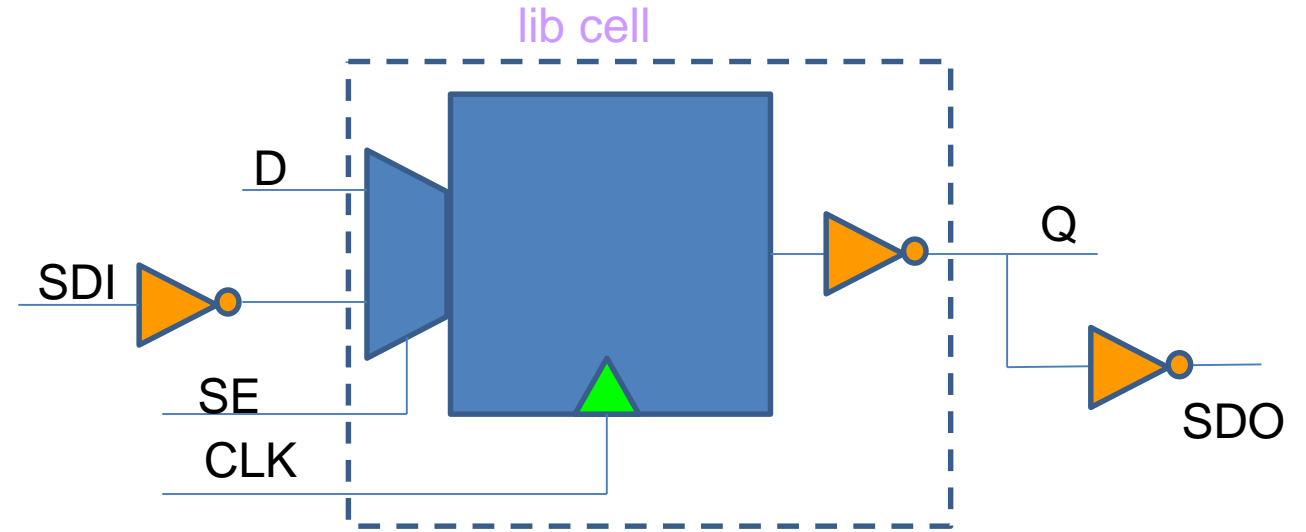
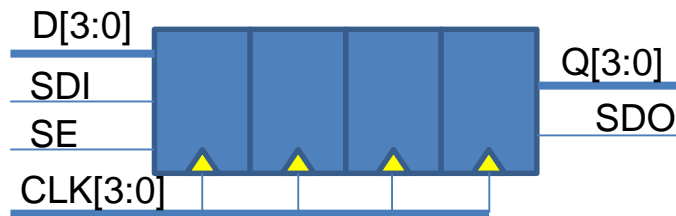
Results and Performance

- Issue 1: PD scan insertion tool fails to consider the extra inverter in the lib cell
- Issue 2: RTL fails to consider quad flops

```
dff #(.RVAL(4'b0101)) dff_inst ( ... );
```



```
dff #(.RVAL(4'b0000)) dff_inst ( ... );
```



- PAGLS performance downgrade over RTL NLP run

| Zen | Runtime | Memory Allocation |
|-------------------|-----------|-------------------|
| Build | 5x | 5x |
| Simulation | 2x | 2x |

Summary

- Capabilities at gate level
 - Perform a synchronous reset in the design
 - Compare the RTL and Gate output scannable flops inside the macros
 - Verify low-power structures added during synthesis by applying real power up sequence
- Challenges
 - PG netlist is available very late in the design cycle
 - Gate model has a slower runtime and a higher memory footprint
- Future improvements
 - Perform detailed investigations during runtime using the simulator profiling

Copyright & Disclaimer

- AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. © **2019** *Advanced Micro Devices, Inc. All rights reserved.*
- **Disclaimer:**
 - The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.
 - THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.