2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

May the powers be with you! – Unleashing powerful new features in UPF IEEE 1801

Srinivasan Venkataramanan, DV Consultant, VerifWorks LLC,
srini@verifworks.com,
Ajeetha Kumari, VerifWorks
Thejas @CVC

accellera SYSTEMS INITIATIVE

VerifWorks LLC
Compassion. Innovation. Trust

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

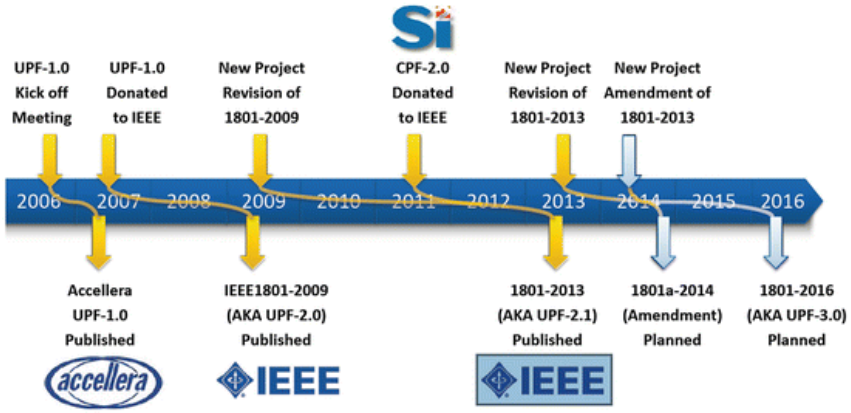
Agenda

- Introduction
- Sample SoC, UPF basics
- Model Simstates
- Sim Reply control
- Sim Assertion Control
- Information Model In UPF
- Automating PST coverage closure
- Summary

accellera SYSTEMS INITIATIVE

Introduction

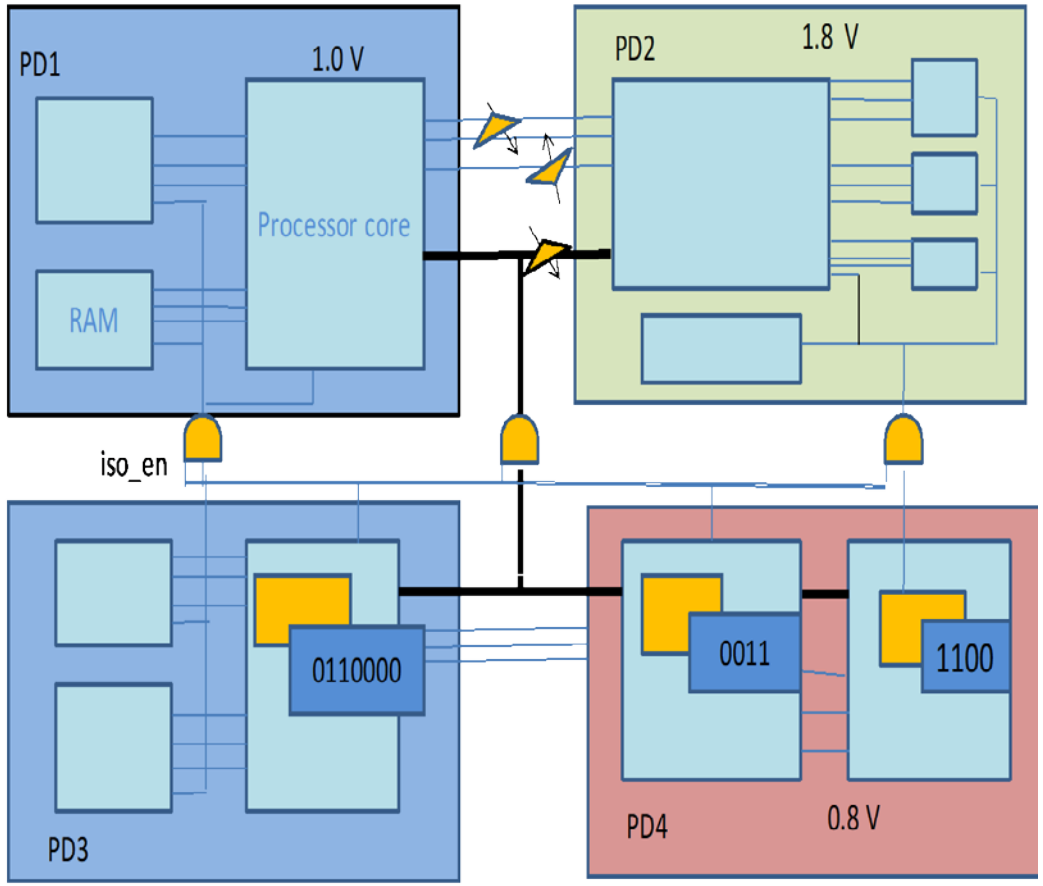
- Low power requirements - omnipresent now
- IEEE 1801 UPF - a TCL based language to capture Low Power requirements
  - Various levels of abstraction
  - Various tools consume & update UPF code
- UPF evolution:
  - 1.0, 2.0, 2.1 and 3.0
  - New features, enhancements, deprecations



- We highlight few key new features/powers of UPF 3.0

Concepts in Low Power DV

- Power Domains
- Isolation mechanism
- Level Shifters
- State Retention
- Power State Tables



Credits: Mentor Graphics

Sample SoC – first-cut UPF

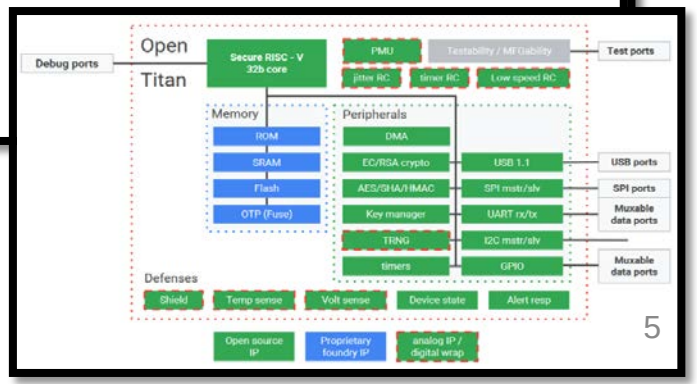
- Ref: Google's Opentitan RoT chip
- Listing power domains is easy
  - Architecture
  - Why that PD? – different topic
- Coding them in UPF can be tedious
  - First timers
  - Too many options, verbose TCL
- We used a UPF generator
  - DVCreate-LP
  - List all hierarchies (upto a depth)
  - Let users pick

```
create_power_domain TOP_PD

create_power_domain RISC_V_PD -elements {top_earlygrey.core }
create_power_domain MEM_SUBSYS_PD -elements {top_earlygrey.tl_adapter_rom \
  top_earlygrey.u_rom_rom \
  top_earlygrey.tl_adapter_ram_main \
  top_earlygrey.u_ram1p_ram_main \
  top_earlygrey.tl_adapter_eflash \
  top_earlygrey.u_flash_eflash \
}

create_power_domain USB_11_PD -elements {
  top_earlygrey.gen_usbdev.udev \
}


create_power_domain PERI_PD -elements {top_earlygrey.uart \
  top_earlygrey.gpio \
  top_earlygrey.spi_device \
  top_earlygrey.rv_timer \
}
```

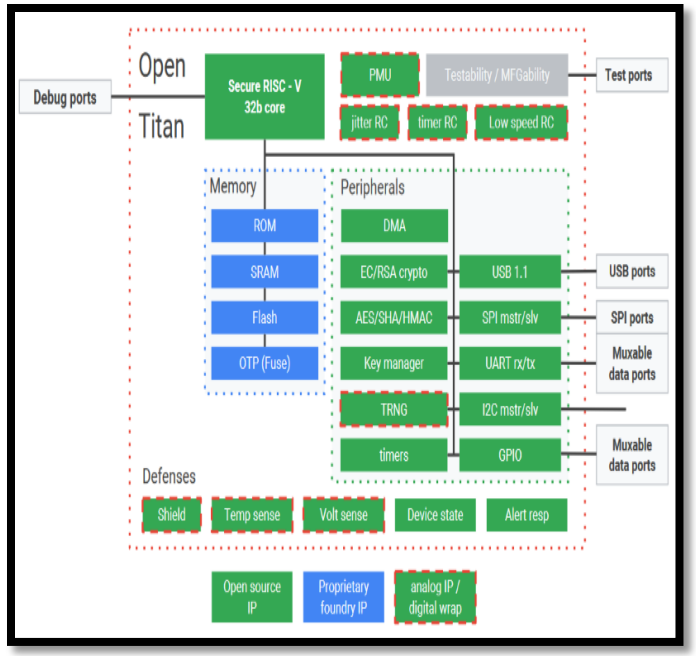


Credits: Google

Power State Tables - PSTs

- Next step - identify the necessary power strategies
- Power State Table (PST) - necessary to arrive at appropriate strategies
- PST - a detailed list of various modes in which the design is intended to be used.

Mode	RISC_V_PD	MEM_SUBSYS_PD	PERI_PD	USB_11_PD	TOP_PD
ALL_ON	ON	ON	ON	ON	ON
LP0	ON	ON_LOW	ON_LOW	ON_LOW	ON
LP1	ON_LOW	OFF	OFF	ON_LOW	ON
SLEEP	OFF	OFF	OFF	OFF	ON



Concept of SIMSTATE in UPF

- Simstate defines precise simulation semantics in a given power state.
- Used to describe the expected behavior of the cells connected to thi supply set.
- UPF 1.0 – had basic support:
  - NORMAL (When ON)
  - CORRUPT (When OFF)
- Designs have more than the above two states:
  - Normal as long as there is no new activity/change etc.

```
add_power_state -supply PD2.primary \
    -state { NORMAL_ss -supply_expr  \
        { (power  == {FULL_ON 3.0} ) && \
            (ground == {FULL_ON 0} ) } } \
    -state { C_ON_ACT_ss -supply_expr  \
        { (power  == {PARTIAL_ON 2.5} ) && \
            (ground == {FULL_ON 0} ) }  \
        -simstate CORRUPT_ON_ACTIVITY } \
    -state { C_ON_CHA_ss -supply_expr  \
        { (power  == {PARTIAL_ON 2.0} ) && \
            (ground == {FULL_ON 0} ) }  \
        -simstate CORRUPT_ON_CHANGE } \
```

CORRUPT_STATE_ON_CHANGE	Corrupt	
NOT_NORMAL	Deferred – tools may provide an override mechanism, default == CORRUPT	Deferred – tools may provide an override mechanism, default == CORRUPT

Power Mode changes in simulation

```
function bit supply_on
  ( string pad_name,
    real value = 1.0);
endfunction : supply_on

function bit supply_off
  ( string pad_name);
endfunction : supply_off
```

```
16  glue_logic g_0();
17
18  initial begin : supply_init
19    int status;
20    status = supply_on("VDDH", 1.8);
21    status = supply_on("VSS", 0.0);
22    status = supply_on("core_0/VDD_CPU", 1.8);
23    status = supply_on("core_0/VSS_CPU", 0.0);
24    status = supply_on("core_1/VDD_CPU", 1.8);
25    status = supply_on("core_1/VSS_CPU", 0.0);
26    status = supply_on("VSS_IP1", 0.0);
27    status = supply_on("VDD_IP1", 1.08);
28    status = supply_on("VSS_IP1", 0.0);
29    status = supply_on("VDD_IP2", 1.8);
30    status = supply_on("VSS_IP2", 0.0);
31  end : supply_init
```

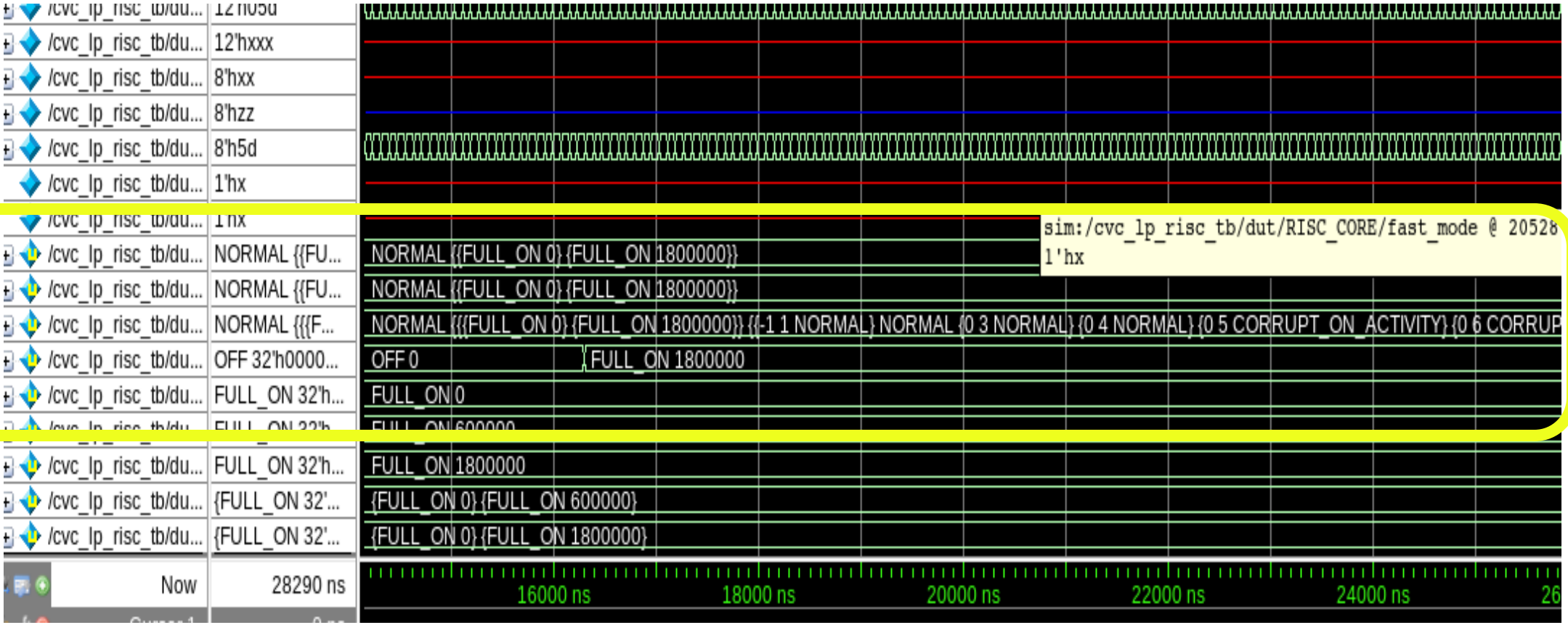
`g2u_lp_supply_on ("VDD_IP1", 1.08)

```
egin
  int status;
  status = supply_on ("VDD_IP1", 1.08);
  if (!status) begin
    `uvm_error ("VDD_IP1",
    "Unable to turn supply_on on given supply net, check net name")
  end
end
```

Can be coded inside UVM SEQ

SIMSTATEs in waveform

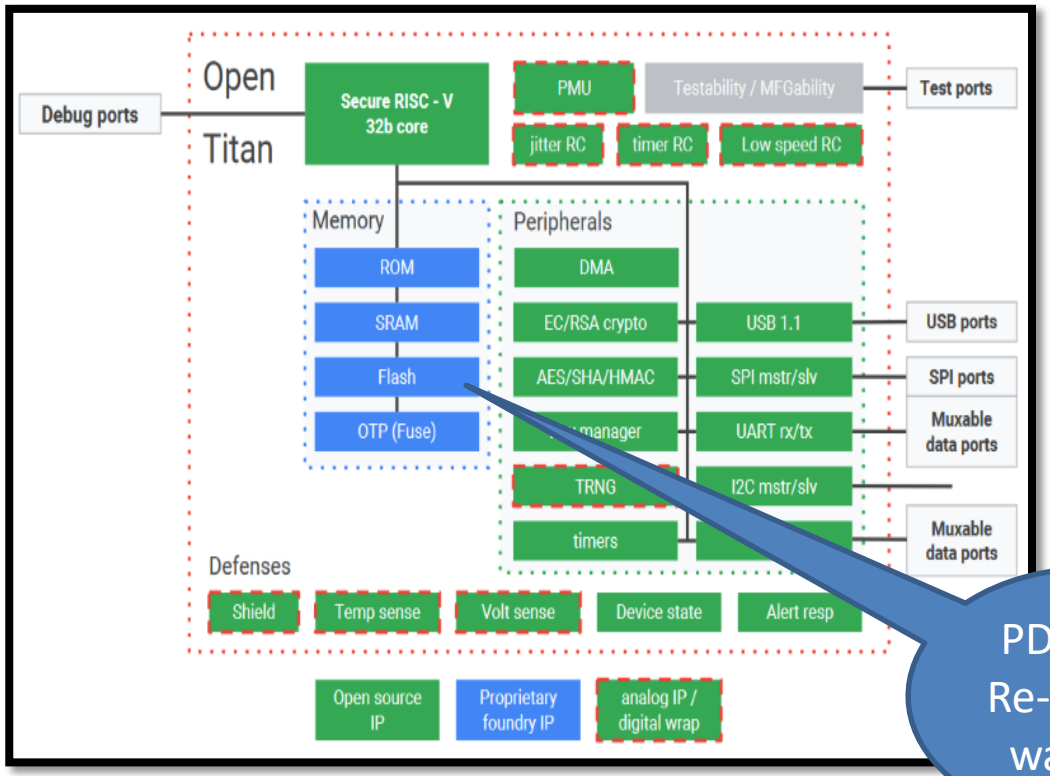
- Mentor's Questa SIM



Sim Reply Control

- Verilog semantics of *initial* block ensures that these blocks get executed at time 0 of a simulation
- What about ON → OFF → ON transition?
  - Re-load memories ($readmemh)
  - Re-initialize PLL/ADC/DAC etc. parameters/states
  - Verilog-only semantics do NOT do these ☹
- sim_replay_control - enables this modelling

Sim Reply Control

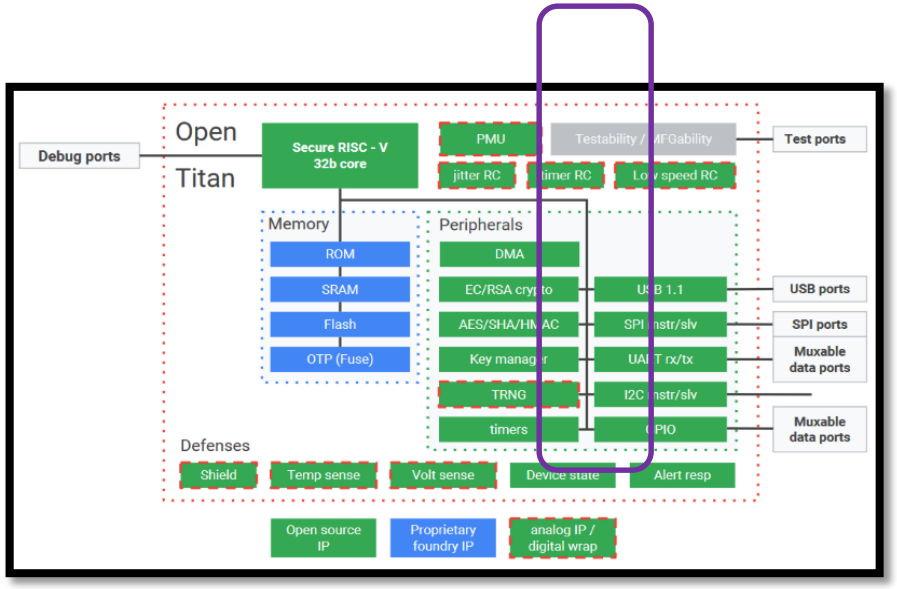



```
add_power_state -supply PD_MEM.primary \
    -state { NORM_pst_ss -supply_expr  \
 { (power  == {FULL_ON 1.8} ) && \
  (ground == {FULL_ON 0} ) } }

sim_replay_control \
              -elements {*}  \
              -domain PD_MEM \
              -model top_earlygrey \
```

Sim Assertion Control

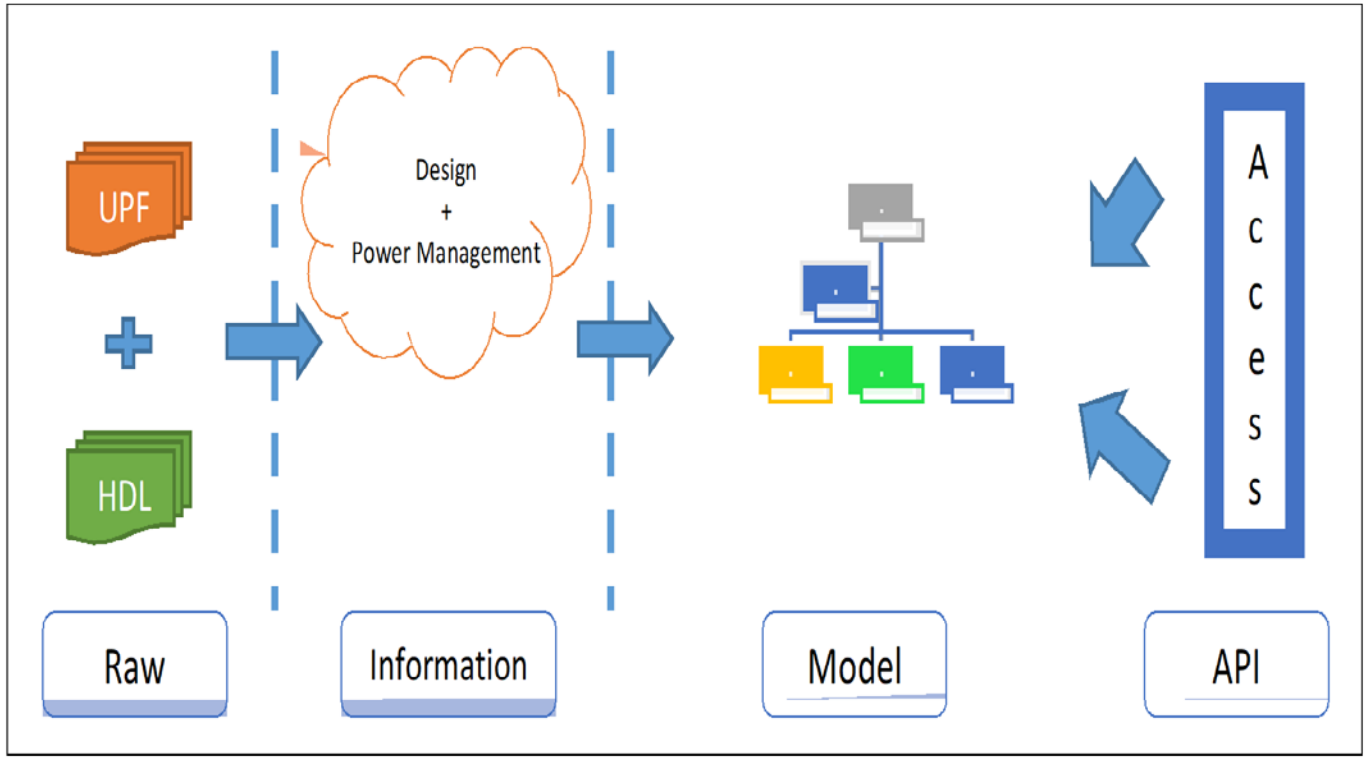
- Consider the assertions inside power down logic, they need special care to ensure they do not provide false negatives.
- UPF 3.0 provides control over these assertions written in SVA.




```
sim_assertion_control \
        -elements {*} \
        -domain PD_IP1 \
        -model chip_top \
        -type suspend
```

Introduction – UPF information model

- Reflection APIs – great for automation
- UPF has Information Model API
- Query:
  - PDs
  - Strategies/ISO/LS etc.
  - PSTs
- Applications:
  - Custom LP rule checker
  - Automated PST closure etc.



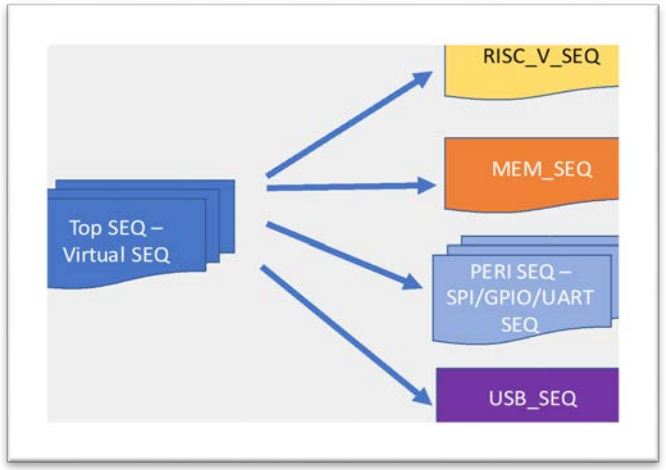
Automating Power-coverage closure

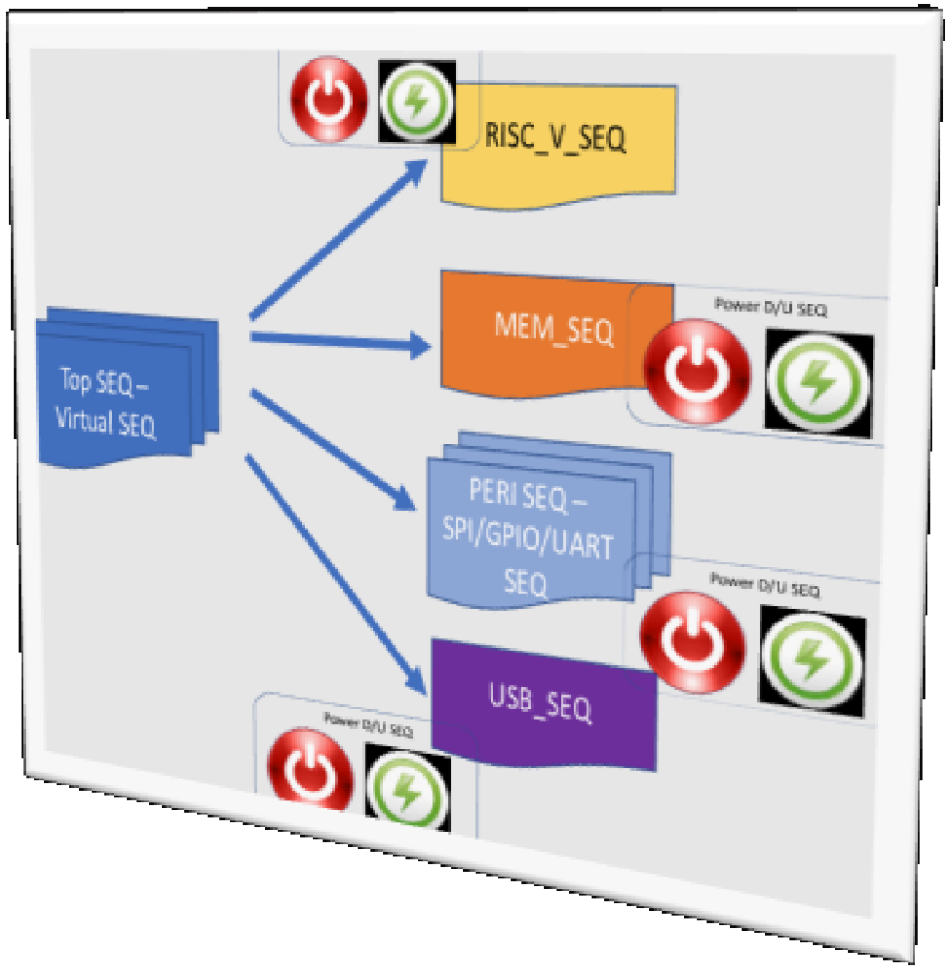
- UVM is widely used in TB domain
- PSTs define a state-space for UVM to hit/cover
- Manual coding of UVM sequences to hit each state/transition
  - Time consuming
  - Error prone, debug cycles
- We used a tool to generate UVM sequences
  - Inputs: UPF PST, config knobs, UVM base SEQ
  - Use Information Model/custom API to query the PST
  - Generate UVM sequences

Automating PST coverage closure

Mode	RISC_V_PD	MEM_SUBSYS_PD	PERI_PD	USB_11_PD	TOP_PD
ALL_ON	ON	ON	ON	ON	ON
LP0	ON	ON_LOW	ON_LOW	ON_LOW	ON
LP1	ON_LOW	OFF	OFF	ON_LOW	ON
SLEEP	OFF	OFF	OFF	OFF	ON

UPF Information Model





2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Summary, Q&A

- Generate first-cut UPF via tool
- SIMSTATE enhancements explained
- New simulation control features introduced with use cases
- Information model in UPF
  - Programming API to Low Power structures
  - Next generation innovation in LP Design Verification
- THANKS!
- Questions?

accellera SYSTEMS INITIATIVE