Exploring Formality with UPF for Low-Power verification

Zhang Chaochao (IC design staff engineer)
Bridgetek Pte Ltd

Background

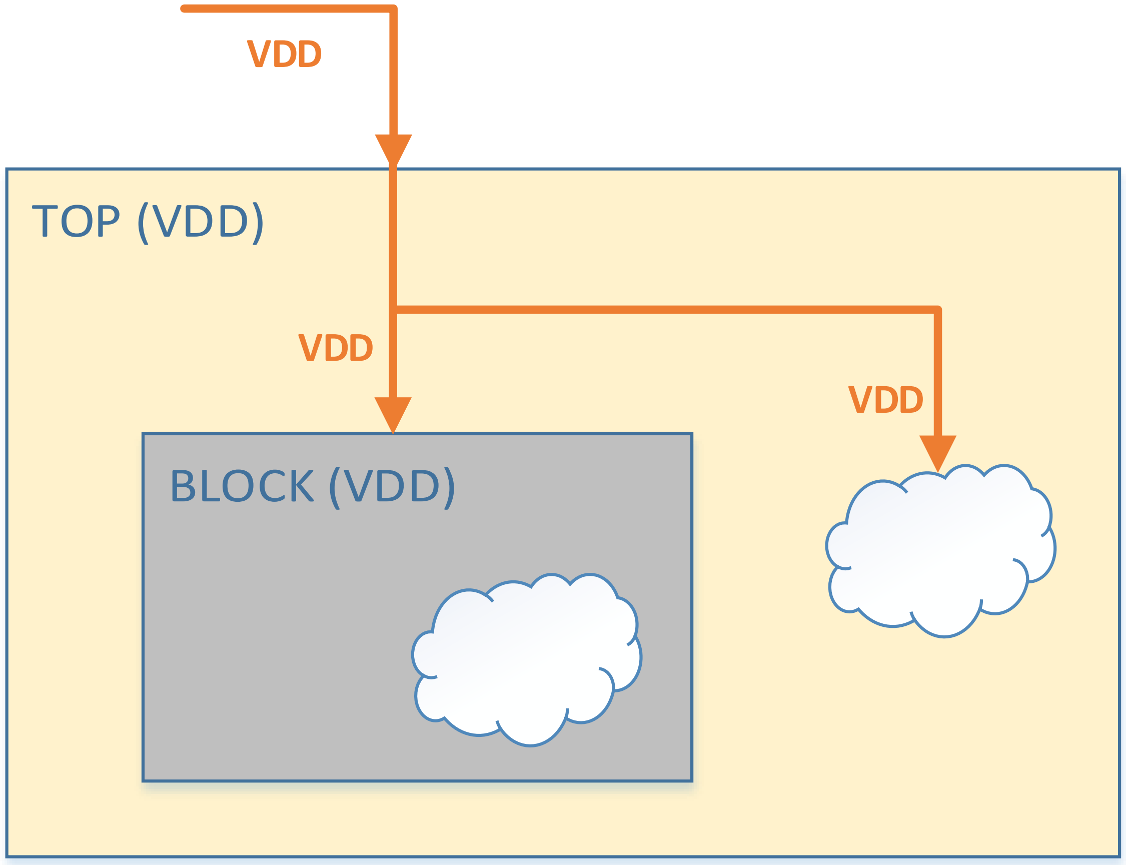
Problem to be solved

- A low power design utilizing UPF
- Formality without UPF
  - Checks logic equivalence only
  - Troublesome manual setup
- Formality with UPF
  - Checks both logic & power equivalence
  - Setup is automatic in Formality

****Golden UPF flow is not discussed here*

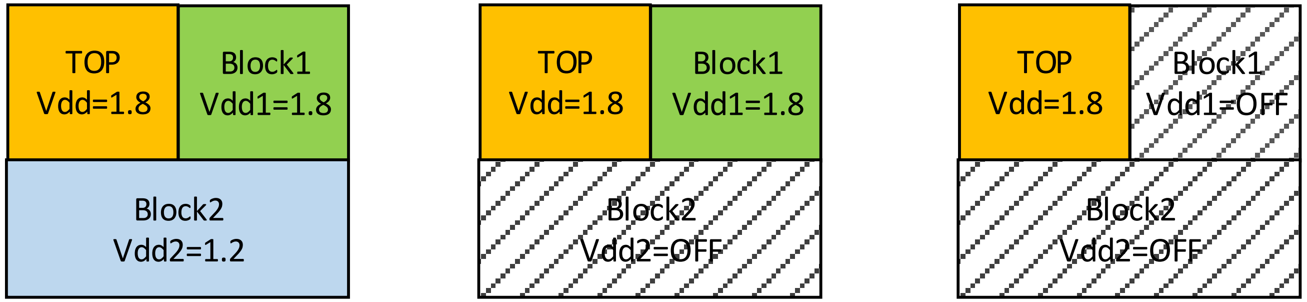
Designs with one power domain/supply

- All digital cells are on/off together
- RTL entry is enough to describe the hardware
- Formality for logic equivalence is enough



Low power designs

- Multi-power domains, multi-voltage supplies
  - Multiple supplies may turn on/off independently and with different voltage levels
  - UPF is required as supplement to RTL about power specification



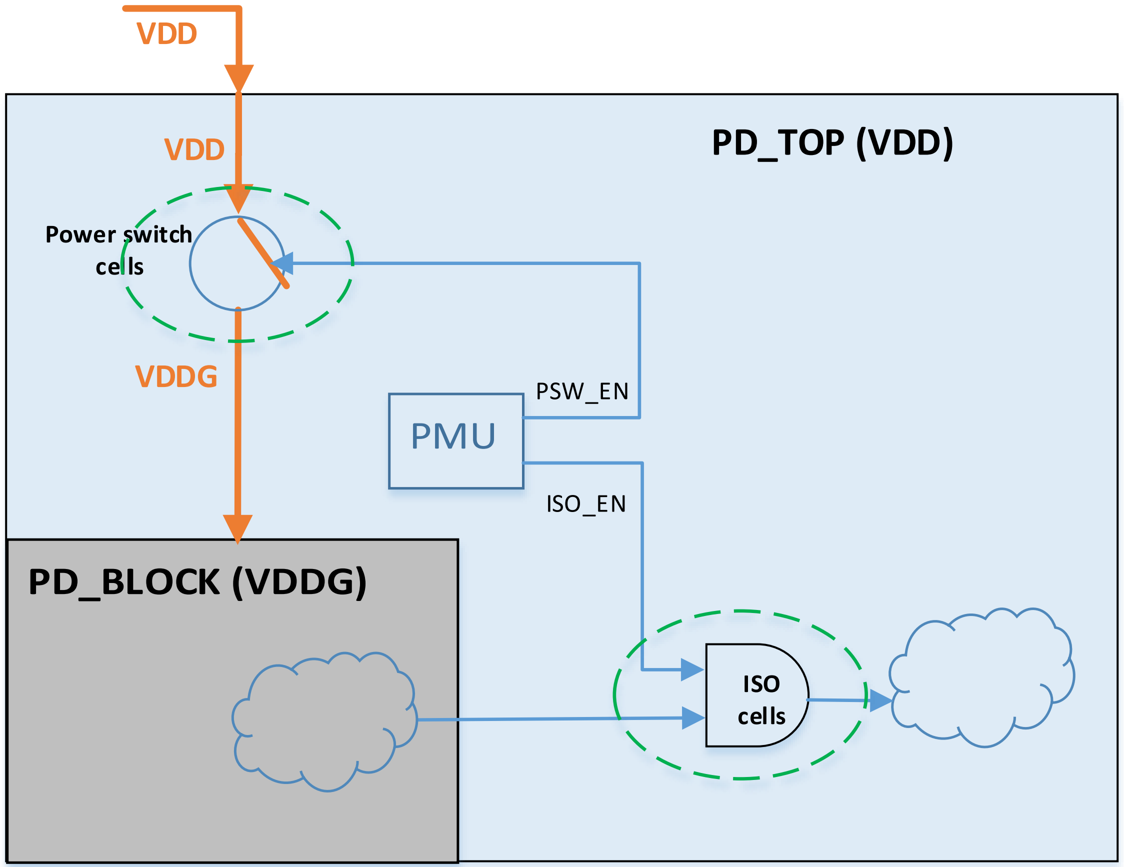
- Formality with UPF flow is required
  - Check through different power domains

What is UPF?

UPF (Unified Power Format)

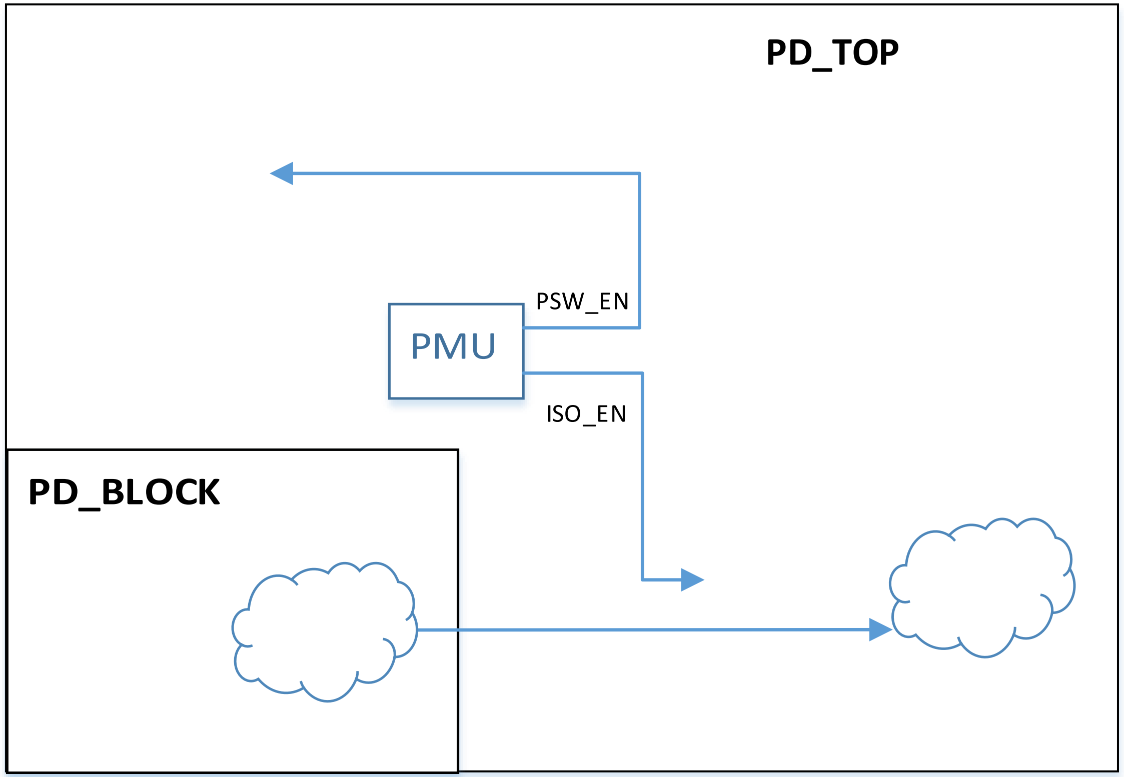
- UPF specifies the power intent of the design
  - Power supplies and power domains
  - Power State Table for legal combination of port state
  - Control of power switch cells for gated power
  - Control of isolation cells for isolation strategy
  - Many others …

UPF example



- 2 power domains
- VDD is power supply
- VDDG is the gated power supply
- PSW_EN controls to turn on/off VDDG
- ISO_EN controls if to force 0 the signal from PD_BLOCK(VDDG) to PD_TOP(VDD)
- PMU is power management unit
- Power state table

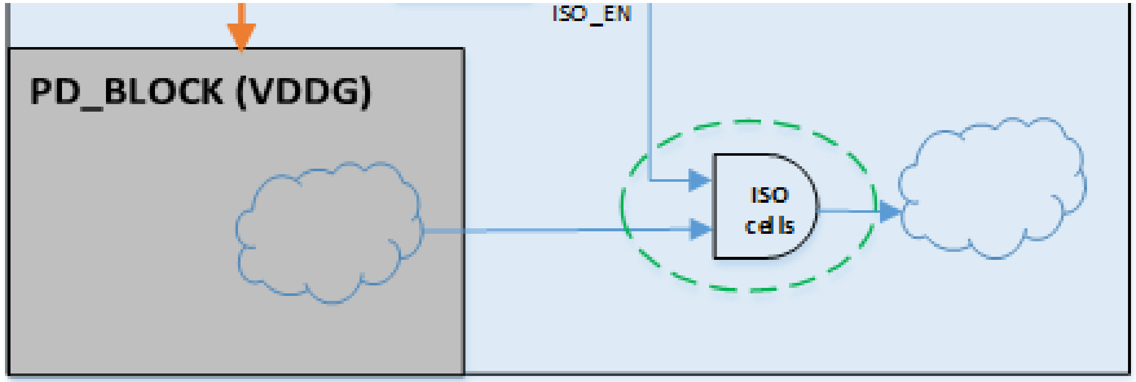
RTL before synthesis



Formality UPF flow

Formality without UPF

- Without load_upf, RTL vs. netlist will NOT be equivalent
- For example, the insertion of ISO cell will change the behavior
- Additional settings like ISO_EN=1 could make it pass
- It is checking logic only

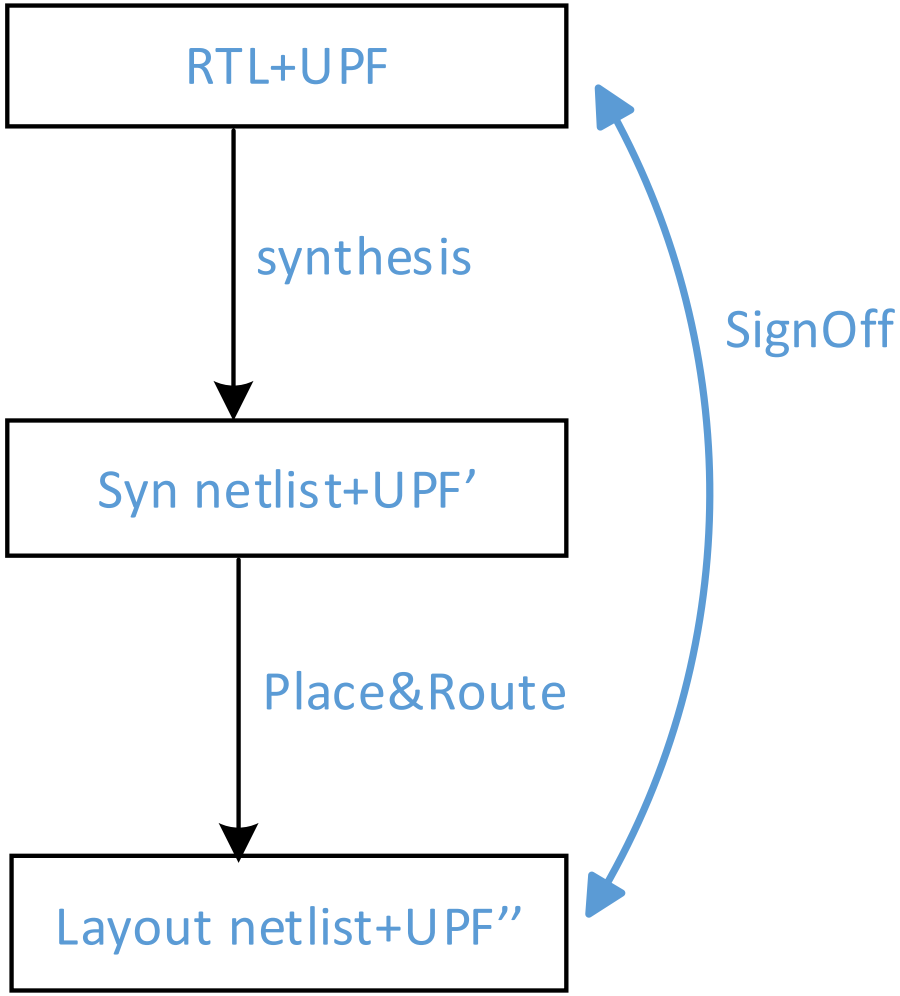


Formality UPF flow

- load_upf will enable Formality UPF flow
- Formality will automatically setup with UPF
  - Those cells like isolation cells will be handled based on UPF
- It will do equivalence check on UPF specifications

Formality UPF flow : Compare

- Signoff
  - RTL+UPF vs. Layout PG netlist
    - ❖Layout netlist has physical power pin connectivity.
  - RTL+UPF vs. Layout nonPG netlist+UPF”
    - ❖Layout netlist power intent is controlled by UPF”.
    - ❖More items are checked like power state table.
- Optional: compare with syn netlist

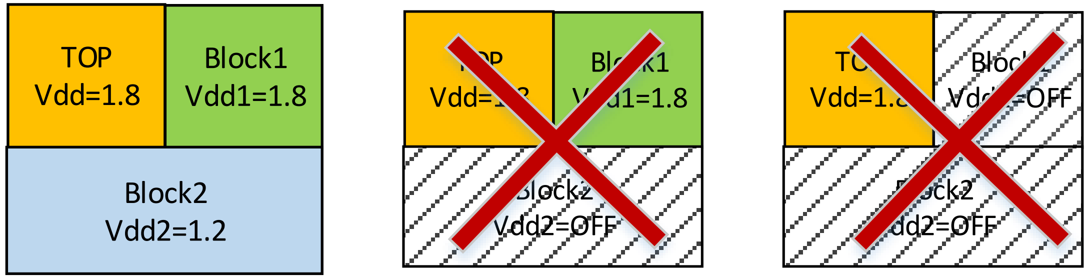


Formality UPF flow : UPF’/UPF”

- UPF is an input from designer
- UPF’ is from synthesis output
  - It additionally reflects changes occurred during synthesis, like isolation cells
- UPF” is from layout output
  - It additionally reflects changes occurring during physical implementation, like power switch cells

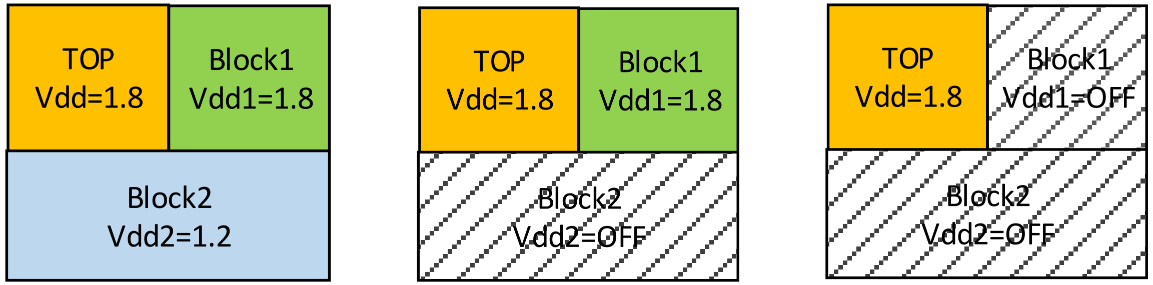
Formality UPF flow (all supplies on=true)

- fm_shell> set verification_force_upf_supplies_on true
  - True is the default setting
  - Verify where all UPF supplies are forced on
  - This is used for initial debugging

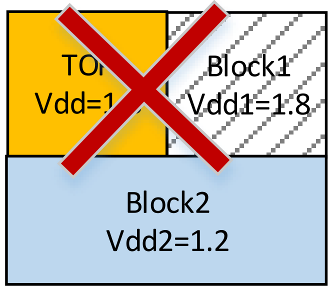


Formality UPF flow (all supplies on=false)

- fm_shell > set verification_force_upf_supplies_on false
  - Verify all possible power states as defined by UPF files
  - This is a complete low power verification of the design, which is SignOff setting



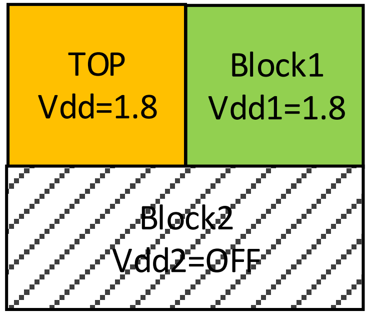
3 legal combinations as defined in UPF



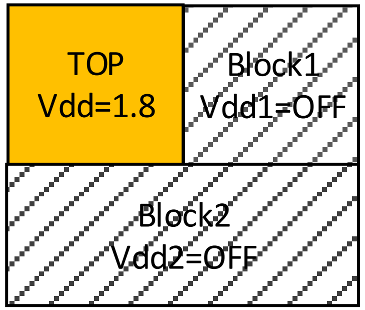
Undefined in UPF, won’t be verified

Formality UPF flow: failure example

- Inconsistent ON/OFF of power supplies: reference with Block1=ON, implementation with Block1=OFF
  - The cut points in implementation/Block1='X', while in reference they are defined.
  - Failure is reported with verification_force_upf_supplies_on=false, but NOT when it is true.



Reference Block1=ON



Implementation Block1=OFF

Formality issues

Issue #1: Layout PG netlist + UPF'' (2 errors)

- Run: RTL+UPF vs. Layout PG netlist+UPF'' (layout)
- FM_UPF-232 error: Name space conflicts between Layout PG netlist & UPF''
  - This is because PG netlist has the PG connection. And UPF'' tries to re-connect.
  - Upf_allow_rtl_pgnet_name_space_conflict=true will merge duplicated connections.
- FM_UPF-206 error: UPF'' supply port does not exist on those power switches.
  - UPF'' has connect_supply_net on all power switches. Formality merges power switches.
  - Disable merge of power switches by hdlin_merge_parallel_switches=false.
  - However, disable merge of power switches makes Formality hard to pass.

Issue #1: Layout PG netlist + UPF'' (-target)

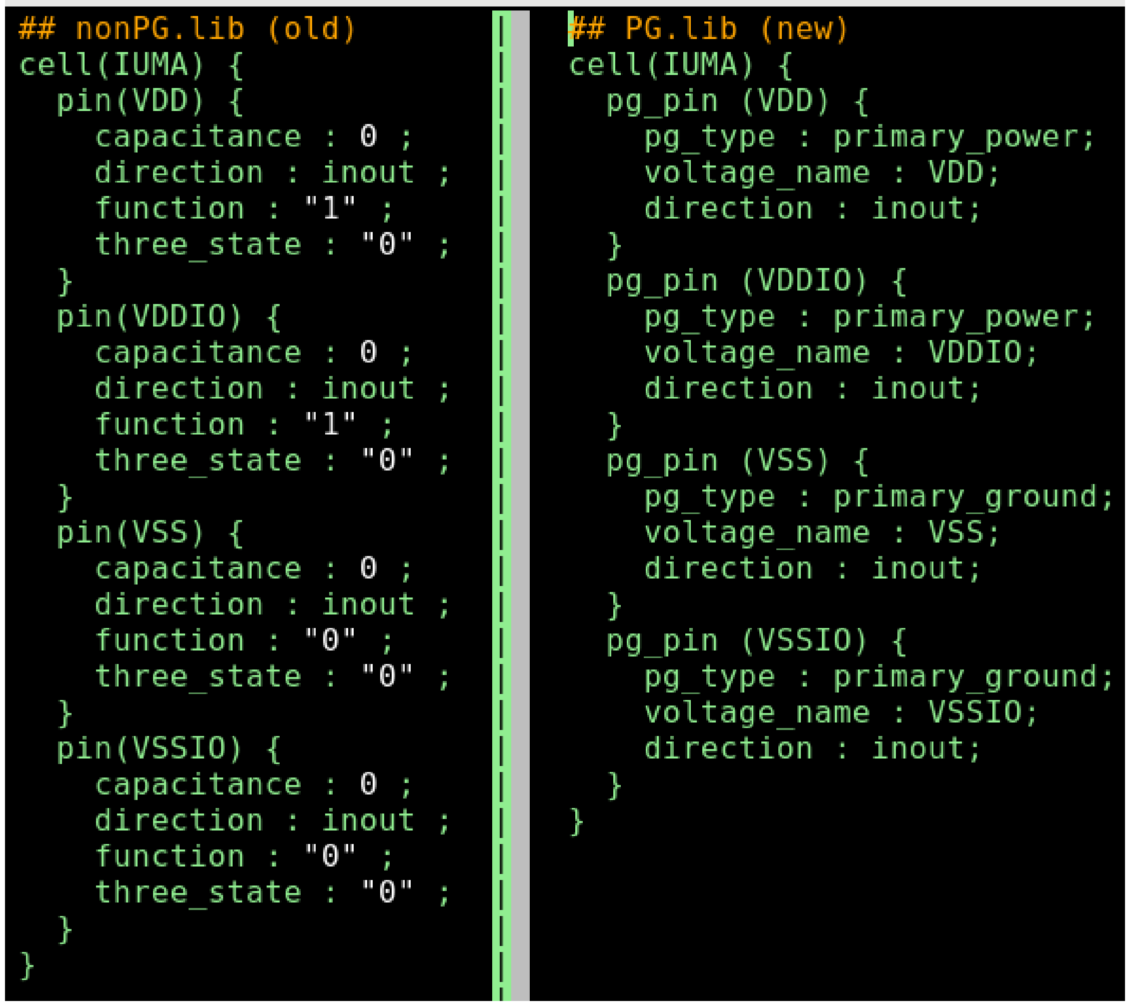
- A better alternative: the dedicated option for PG netlist + UPF'' run
  - Load_upf –target pg_netlist
- The 2 errors will not appear.
- Formality can pass easily.

Issue #2: nonPG lib (RTL fix)

- Our case: IO PAD is of nonPG lib
  - The power pins of PADs is defined as some signal pin
- Issues
  - DC will issue errors if UPF defines PG connection for IO PAD power pins
  - It requires power pins of PADs to be pg_pin instead of signal_pin
- Solution (used in tape out)
  - RTL connects those IO PAD VDD/VSS directly

Issue #2: nonPG lib (converted PG lib)

- Convert IO PAD lib: nonPG.lib -> PG.lib
  - – add_pg_pin_to_lib $nonPG.lib –output $PG.lib …
- UPF connects power pins of IO PADs



```
## nonPG.lib (old)
cell(IUMA) {
  pin(VDD) {
    capacitance : 0 ;
    direction : inout ;
    function : "1" ;
    three_state : "0" ;
  }
  pin(VDDIO) {
    capacitance : 0 ;
    direction : inout ;
    function : "1" ;
    three_state : "0" ;
  }
  pin(VSS) {
    capacitance : 0 ;
    direction : inout ;
    function : "0" ;
    three_state : "0" ;
  }
  pin(VSSIO) {
    capacitance : 0 ;
    direction : inout ;
    function : "0" ;
    three_state : "0" ;
  }
}
```

```
## PG.lib (new)
cell(IUMA) {
  pg_pin (VDD) {
    pg_type : primary_power;
    voltage_name : VDD;
    direction : inout;
  }
  pg_pin (VDDIO) {
    pg_type : primary_power;
    voltage_name : VDDIO;
    direction : inout;
  }
  pg_pin (VSS) {
    pg_type : primary_ground;
    voltage_name : VSS;
    direction : inout;
  }
  pg_pin (VSSIO) {
    pg_type : primary_ground;
    voltage_name : VSSIO;
    direction : inout;
  }
}
```

Summary

- This presentation has covered
  - A brief introduction on UPF
  - The necessity and benefits to include UPF in Formality
  - Some settings & issues for Formality UPF

- Working through Formality UPF has improved the signoff flow
  - Better understanding on the principle of Formality
  - More aware of UPF impact on Formality
  - Enhanced debugging skills in Formality UPF

Conclusion

- Running Formality with UPF has enhanced our consistency check
  - Including UPF, the formality setup becomes automatic
  - It checks for all the legal combinations of supply ON/OFF as defined in UPF

➡ Formality with UPF is now integrated in our standard design flow

References

- https://spdocs.synopsys.com/dow_retrieve/qsc-v/dg/forecoolh/V-2023.12-SP2/forecoolh/smvfug/upf_script_examples/upf_script_examples.html
  - The link contains a few examples about UPF scripts.
- https://solvnetplus.synopsys.com/s/article/Adding-PG-Pin-Syntax-to-Logical-Libraries-Application-Note-1576148257314
  - This link is the application note for add_pg_pin_to_lib (convert nonPG to PG)

Bridgetek
BRIDGING TECHNOLOGY

snug

THANK YOU

Our Technology,
Your Innovation™

Questions & Answers