Successive Refinement:

A Methodology for Incremental Specification of Power Intent

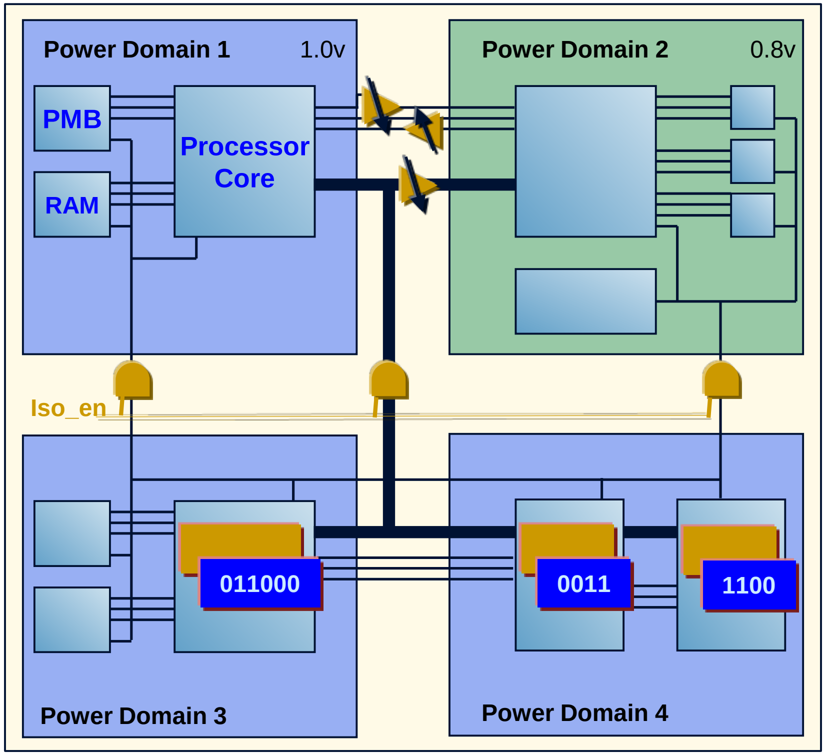
Adnan Khan, Eamonn Quigley, John Biggs - ARM Ltd.
Erich Marschner - Mentor Graphics

What is UPF?

- **An Evolving Standard**
  - Accellera UPF in 2007 (1.0)
  - IEEE 1801-2009 UPF (2.0)
  - IEEE 1801-2013 UPF (2.1)
  - IEEE 1801a-2014 UPF (2.2)
  - IEEE 1801-2015 UPF (3.0)
    - (In development now)
- **Based upon Tcl**
  - Tcl syntax and semantics
  - Can be mixed with non-UPF Tcl
- **And HDLs**
  - SystemVerilog, Verilog,
  - VHDL, and (in 3.0) SystemC
- **For Power Intent**
  - To define power management
  - To optimize power consumption
- **For Power Analysis (in 3.0)**
  - Component Power Modeling
- **For Verification**
  - Simulation or Emulation
  - Static/Formal Verification
- **For Implementation**
  - Synthesis, DFT, P&R, etc.

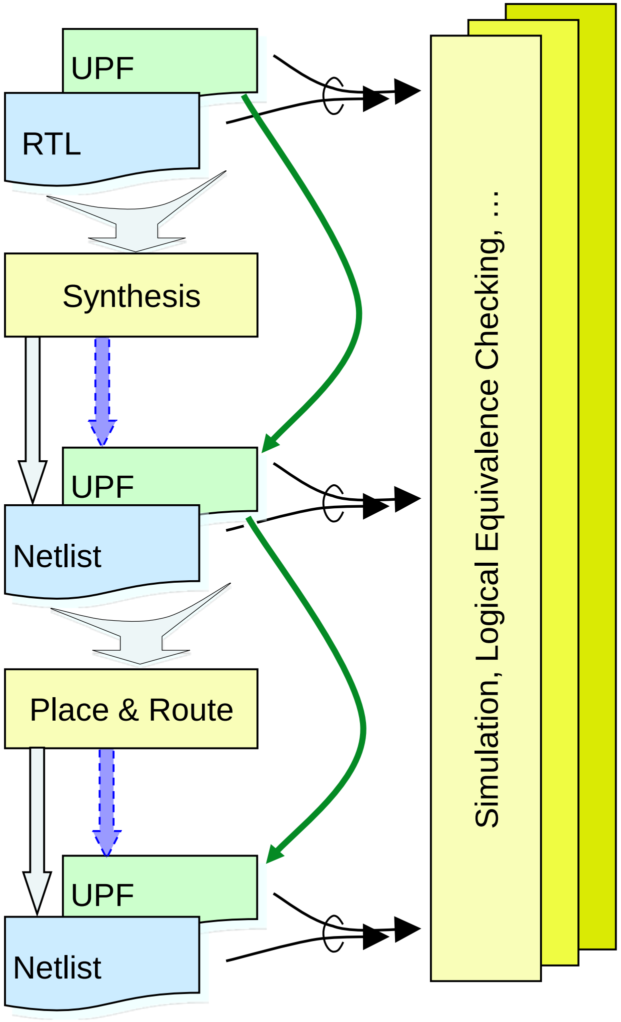
Power Mgmt Concepts

- Power Domains
  - Independently powered regions
  - Enable application of different power reduction techniques in each region
- State Retention
  - To save essential data when power is off
  - To enable quick resumption after power up
- Isolation
  - To ensure correct electrical/logical interactions between domains in different power states
- Level Shifting
  - To ensure correct communication between different voltage levels



UPF 1.0 Design Flow

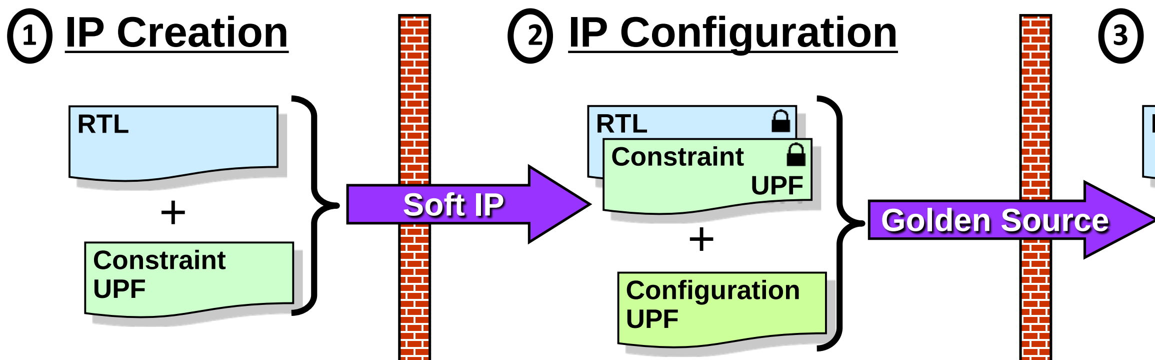
- RTL is augmented with UPF
  - To define power management architecture
- RTL + UPF verification
  - To ensure that power architecture completely supports planned power states of design
  - To ensure that design works correctly under power management
- RTL + UPF implementation
  - Synthesis, test insertion, place & route, etc.
  - UPF may be updated by user or tool
- NL + UPF verification
  - Power aware equivalence checking, static analysis, simulation, emulation, etc.

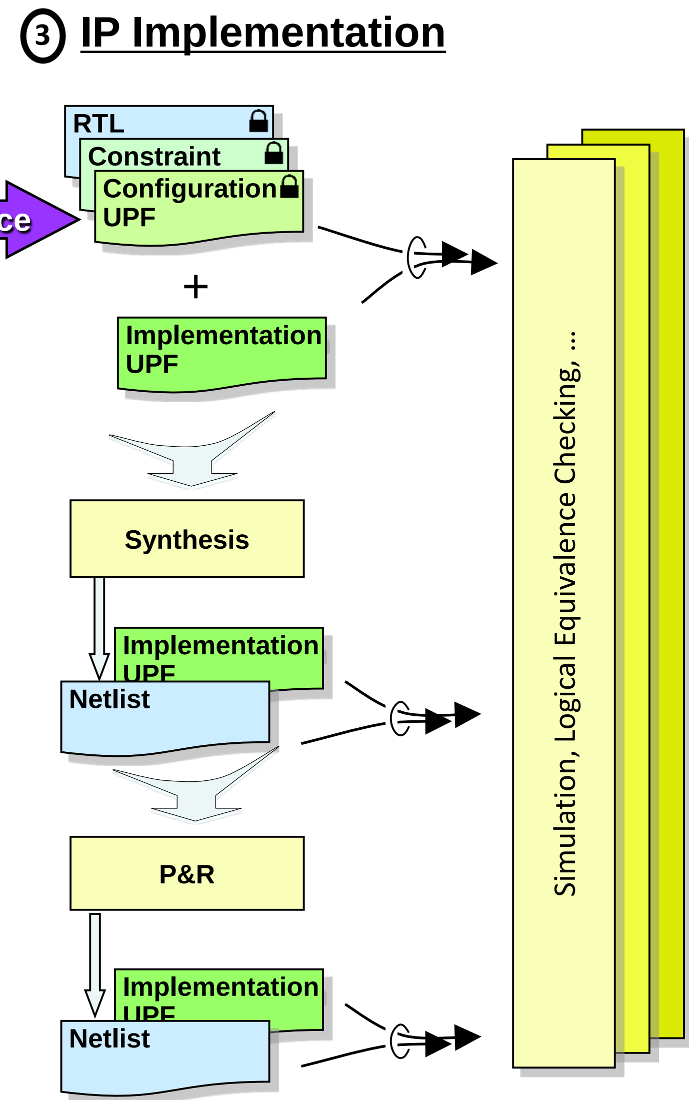


Issues With This Flow

- UPF 1.0 is implementation-oriented
  - Verification requires target technology information
- But target technology may be unknown until later
- This leads to
  - Making assumptions that turn out to be invalid
  - Having to redo verification again later
  - Delaying verification until late in the flow
  - Doing less than thorough verification
  - Having to redo verification entirely if retargetting

Successive Refinement





IP Provider:

- Creates IP source
- Creates low power implementation constraints

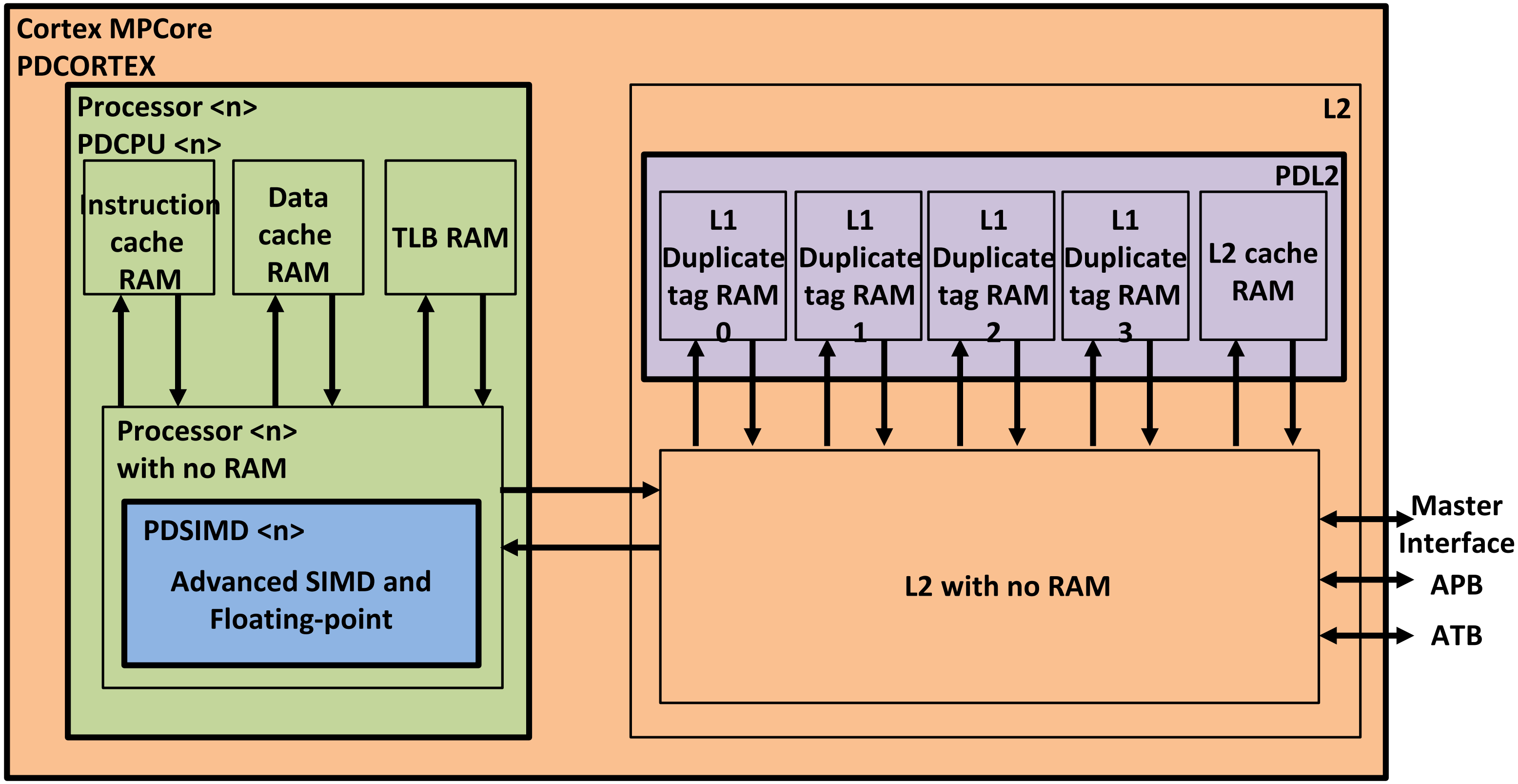
IP Licensee/User:

- Configures IP for context
- Validates configuration
- Freezes “Golden Source”
- Implements configuration
- Verifies implementation against “Golden Source”

UPF Layers

- Constraint UPF
  - Power intent inherent in the IP
    - power domains/states/isolation/retention etc.
  - Part of source IP, travels with RTL
- Configuration UPF
  - Application-specific configuration of instances
    - supply sets, power states, logic expressions, etc.
  - Required for simulation
- Implementation UPF
  - Technology-specific implementation of system
    - supply nets/ports, switches, etc.
  - Required for implementation

ARM® Cortex®-MPCore Processor IP



Constraint UPF File

- Define Atomic Power Domains
- Define Retention Constraints
- Define Isolation Constraints
- Define Fundamental Power States
- Define Abstract Retention States
- Define Power State Constraints

These Specify How the IP Can and Cannot be Used in a System

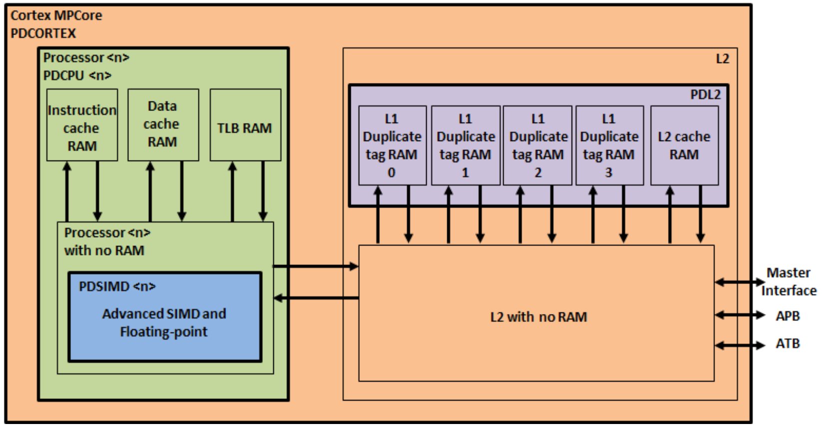
Atomic Power Domains

```
# Create the L2 Cache domain
create_power_domain PDL2 –elements \
  “u_ca_l2/u_ca_l2_datarams \
   u_ca_l2/u_ca_l2_tagrams \
   u_ca_l2/u_cascu_l1d_tagrams” \
  -atomic

# Create the SIMD power domain
create_power_domain PDSIMD0 \
  –elements “u_ca_advsimd0” -atomic

# Create the CPU0 power domain
create_power_domain PDCPU0 \
  –elements “u_ca_cpu0” -atomic

# Create the cluster power domain
create_power_domain PDCORTEX \
  –elements {.} -atomic
```

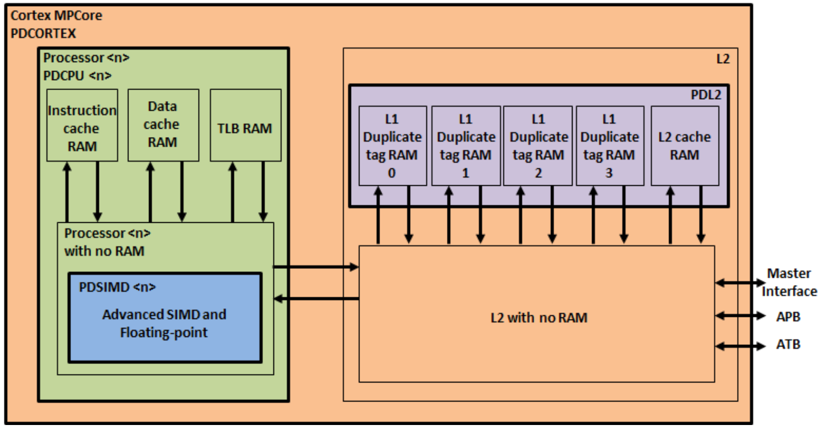


Retention and Isolation Constraints

```
set_retention_elements PDCPU0_RETN –elements “u_ca_cpu0”

# default is isolate low
set_port_attributes –elements “u_ca_hierarchy” \
  -applies_to outputs \
  -clamp_value 0

# some ports need to be clamped high
set_port_attributes \
  -ports “u_ca_hierarchy/out1” \
  -clamp_value 1
```



Domain/Supply Power States

```
add_power_state PDSIMD0 –domain \
  -state {RUN -logic_expr {primary == ON}} \
  -state {SHD -logic_expr {primary == OFF}}

add_power_state PDSIMD0.primary –supply \
  -state {ON  -simstate NORMAL \
    -supply_expr {power == FULL_ON && ground == FULL_ON}} \
  -state {OFF -simstate CORRUPT \
    -supply_expr {power == OFF}}
```

Fundamental

Optional

```
add_power_state PDSIMD0 –domain –update \
  -state {RET \
    -logic_expr {primary==OFF && default_retention==ON}}
```

IP Power States

```
add_power_state PDCORTEX -domain \
  -state {RUN \
    -logic_expr {primary==ON  && PDL2==RUN && PDCPU==RUN}} \
  -state {DMT \
    -logic_expr {primary==OFF && PDL2==RUN && PDCPU==SHD}} \
  -state {SHD \
    -logic_expr {primary==OFF && PDL2==SHD && PDCPU==SHD}}
```

Fundamental

Optional

```
add_power_state PDCORTEX –domain -update \
  -state {CPU0_RET \
    -logic_expr {primary==ON  && PDL2!=SHD && PDCPU==RET}} \
  -state {L2_RET \
    -logic_expr {primary==ON  && PDL2==RET && PDCPU!=SHD}}
```

Power State Constraints

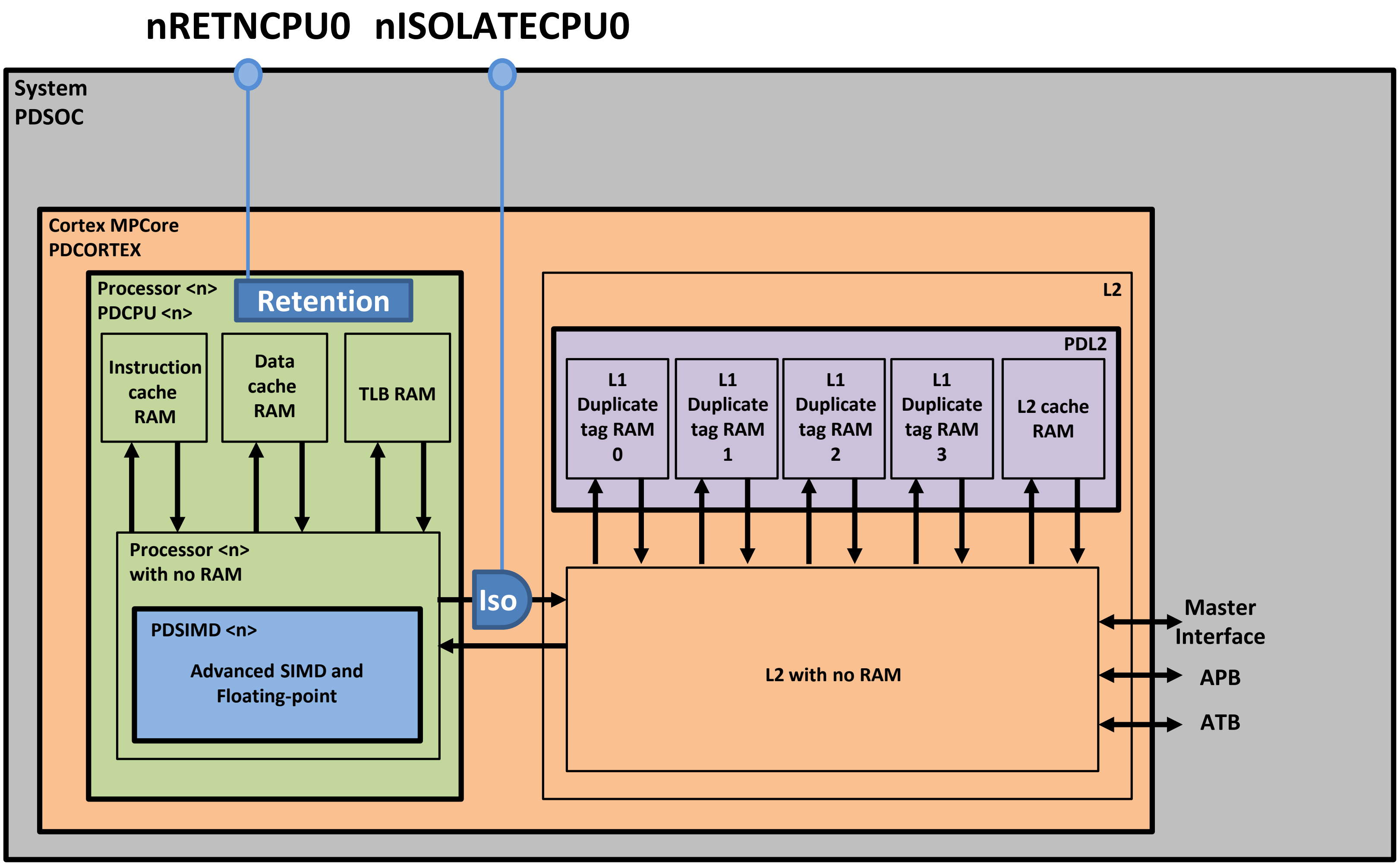
- Specific Illegal Power States

```
add_power_state PDCORTEX -update \
  -state CPU0_RET_ONLY -illegal   \
    {-logic_expr {primary == ON && PDL2    == RUN &&
                  PDCPU0 == RET && PDSIMD0 == RUN}}
```

- All Undefined Power States are Illegal

```
add_power_state PDCORTEX -update -complete
```

MPCore Instance in System



Configuration UPF File

- Load Constraint UPF for each Instance

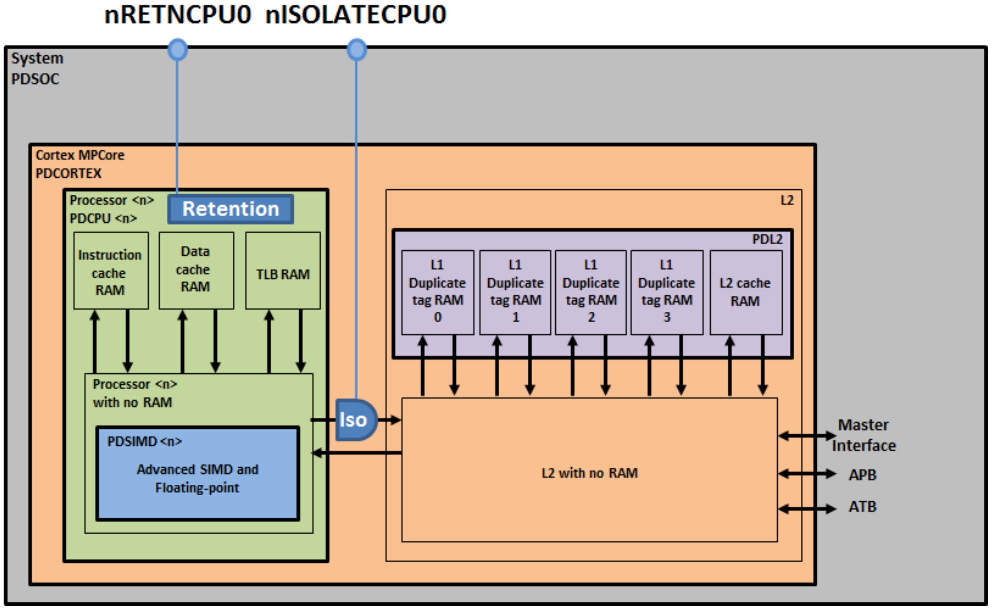
```
load_upf -scope MPCoreInst CORTEX_constraints.upf
```

- Define Control Logic
- Define Retention Strategies
- Define Isolation Strategies
- Update Power States with Control Conditions
- Refine Power States as Required

All Configuration Info Will Be Checked Against Constraints

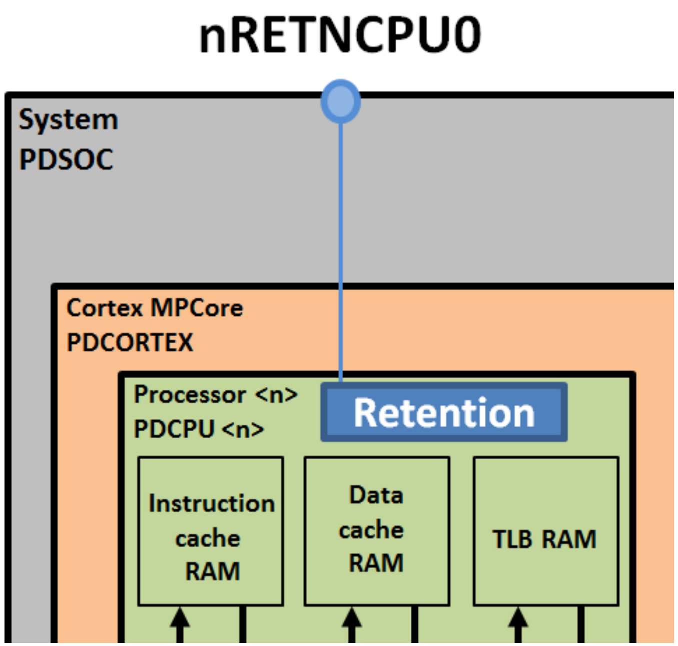
Control Logic

```
create_logic_port -direction in nRETNCPU0
create_logic_net nRETNCPU0
connect_logic_net nRETNCPU0 -ports nRETNCPU0
```



Retention Strategies

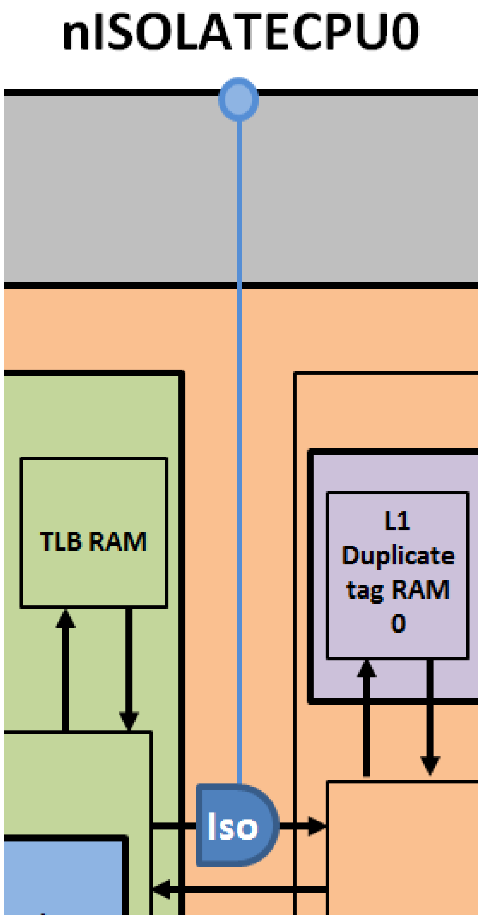
```
set_retention ret_cpu0 -domain PDCPU0 \
  -retention_supply_set PDCPU0.default_retention \
  -save_signal "nRETNCPU0 negedge" \
  -restore_signal "nRETNCPU0 posedge"
```



Isolation Strategies

```
# -------- cpu clamp 0 ---------
set_isolation iso_cpu_0 -domain PDCPU0 \
  -isolation_supply_set PDCORTEX.primary \
  -clamp_value 0 \
  -applies_to outputs \
  -isolation_signal nISOLATECPU0 \
  -isolation_sense low

# -------- cpu clamp 1 ---------
set_isolation iso_cpu_1 -domain PDCPU0 \
  -isolation_supply_set PDCORTEX.primary \
  -clamp_value 1 \
  -elements "u_ca_hierarchy/out1" \
  -isolation_signal nISOLATECPU0 \
  -isolation_sense low
```



Update Domain Power States

```
add_power_state PDSIMD0 –domain -update \
  -state {RUN \
    -logic_expr {nPWRUP_SIMD0==0 && nPWRUPRetn_SIMD0==0}} \
  -state {RET \
    -logic_expr {nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0==0 && \
                 nRETN_SIMD0 ==0 && nISOLATE_SIMD0  ==0}} \
  -state {SHD \
    -logic_expr {nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0== 1}}
```

Update Domain Power States

in Constraint UPF

```
add_power_state PDSIMD0 -domain \
  -state {RUN -logic_expr {primary == ON}} \
  -state {SHD -logic_expr {primary == OFF}}
  -state {RET \
    -logic_expr {primary==OFF && default_retention==ON}}
```

in Config UPF

```
add_power_state PDSIMD0 -domain -update \
  -state {RUN \
    -logic_expr {nPWRUP_SIMD0==0 && nPWRUPRetn_SIMD0==0}} \
  -state {RET \
    -logic_expr {nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0==0 && \
                 nRETN_SIMD0 ==0 && nISOLATE_SIMD0  ==0}} \
  -state {SHD \
    -logic_expr {nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0== 1}}
```

Update Effect

```
add_power_state PDSIMD0 –domain \
   -state {RUN -logic_expr {primary == ON} && \
                 nPWRUP_SIMD0==0 && nPWRUPRetn_SIMD0==0} \
   -state {SHD -logic_expr {primary == OFF} && \
                 nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0== 1}
   -state {RET \
     -logic_expr {primary==OFF && default_retention==ON && \
                 nPWRUP_SIMD0==1 && nPWRUPRetn_SIMD0==0 && \
                 nRETN_SIMD0 ==0 && nISOLATE_SIMD0  ==0}}
```

Update Supply Power States?

in Constraint UPF

```
add_power_state PDSIMD0.primary –supply \
  -state {ON  -simstate NORMAL \
    -supply_expr {power == FULL_ON && ground == FULL_ON}} \
  -state {OFF -simstate CORRUPT \
    -supply_expr {power == OFF}}
```

in Config UPF

```
add_power_state PDSIMD0.primary –supply -update \
  -state {ON  -logic_expr {nPWRUP_SIMD0 == 0}} \
  -state {OFF -logic_expr {nPWRUP_SIMD0 == 1}}

add_power_state PDSIMD0.primary –supply \
  -state {ON  -simstate NORMAL \
    -supply_expr {power == FULL_ON && ground == FULL_ON}} \
    -logic_expr {nPWRUP_SIMD0 == 0} \
  -state {OFF -simstate CORRUPT \
    -supply_expr {power == OFF}} \
    -logic_expr {nPWRUP_SIMD0 == 1}
```

Use with Caution: Implies a Switch, but does not define one

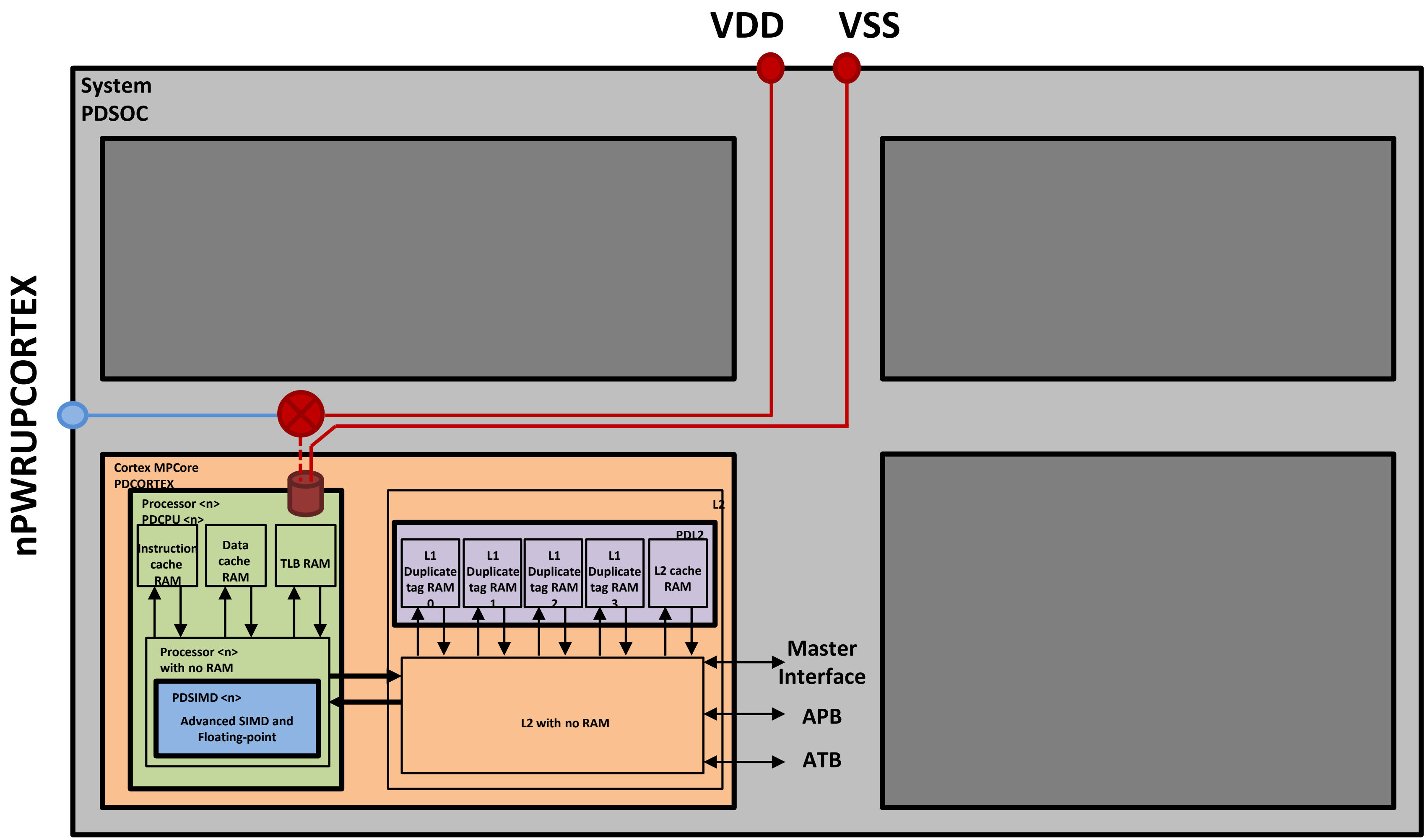
Refine Power States

- What if there are two different retention states?
  - Sleep: clock gated, domain isolated, reverse biased
  - Deep Sleep: power gated, state saved in balloon latch
- Use branching refinement

```
add_power_state PDSIMD0 –domain -update \
  -state {SLEEP \
    -logic_expr {PDSIMD0 == RET && nPDSIMD0_SLP == 2`b10} \
  -state {DEEPSLEEP \
    -logic_expr {PDSIMD0 == RET && nPDSIMD0_SLP == 2`b01}
```

For more information, see DVCon 2015 paper “Unleashing the Full Power of UPF Power States” by E. Marschner, J. Biggs

System Power Distribution



Implementation UPF File

- Load Configuration UPF for the System

```
load_upf SoC_configuration.upf
```

- Create Supply Network Elements
- Create Power Switches
- Update Supply Sets with Supply Nets
- Update Power States with Voltages
- Specify Other Technology Info as Required

Configuration Imposes Requirements on the Implementation

Define Supply Network

```
create_supply_port VDD
create_supply_port VSS

create_supply_net  VDDCORTEX -domain PDCORTEX
create_supply_net  VDDCPU0 -domain PDCPU0
create_supply_net  VDDRCPU0 -domain PDCPU0

connect_supply_net …
```

Define Power Switches

```
create_power_switch ps_CORTEX_primary -domain PDCORTEX \
  -input_supply_port { VDD VDD } \
  -output_supply_port { VDDCORTEX VDDCORTEX } \
  -control_port { nPWRUPCORTEX nPWRUPCORTEX } \
  -on_state { on_state VDD {!nPWRUPCORTEX} } \
  -off_state { off_state {nPWRUPCORTEX} }
```

Bind Supply Sets to Nets

```
create_supply_set PDCPU0.primary -update \
  -function {power VDDCPU0} -function {ground VSS}

create_supply_set PDCPU0.default_retention -update \
  -function {power VDDRCPU0} -function {ground VSS}
```

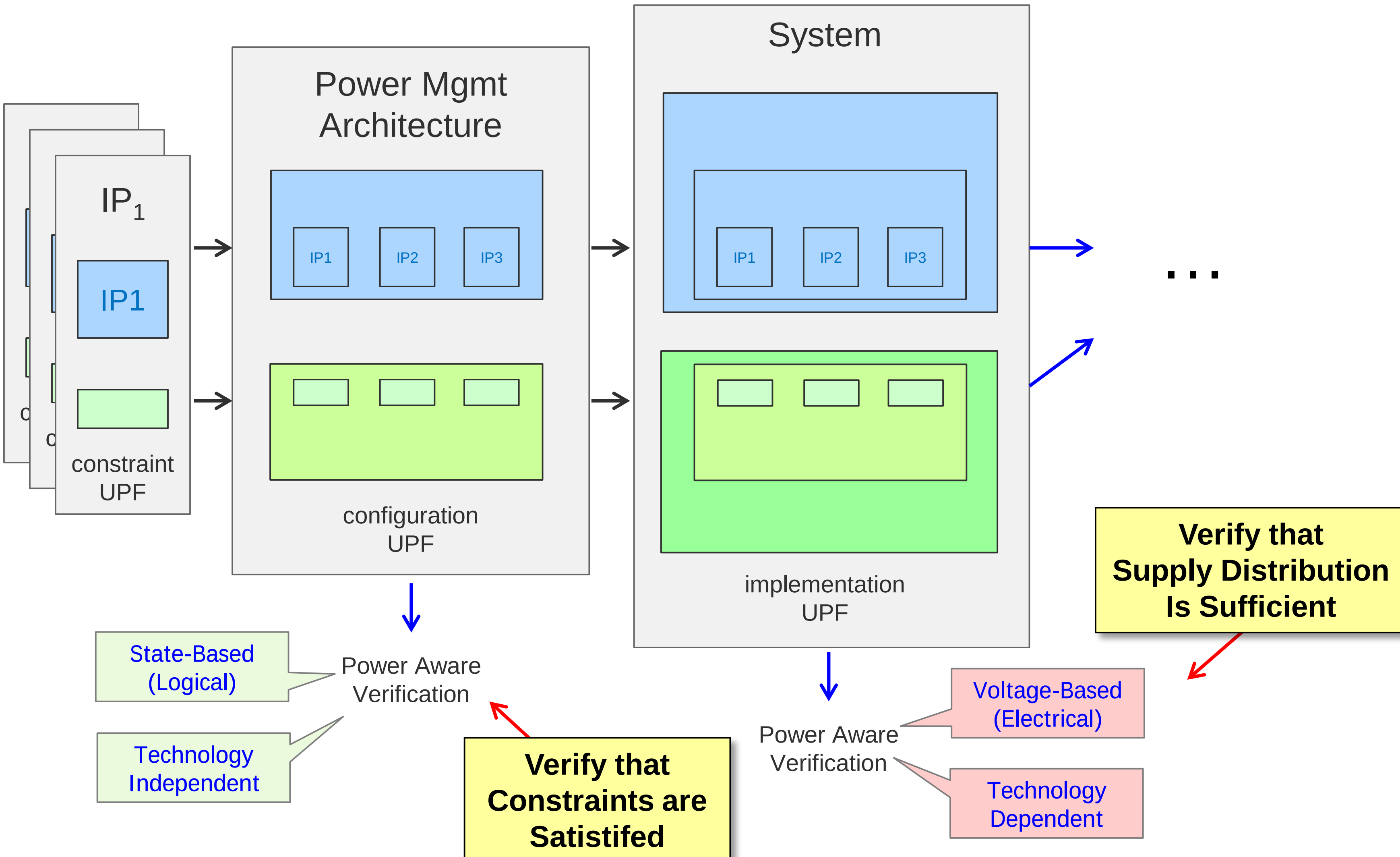
Update Supply Power States

```
add_power_state PDCPU0.primary –supply -update \
   -state {ON  -supply_expr {power  == {FULL_ON 0.81} && \
                             ground == {FULL_ON 0.00}}} \
   -state {OFF -supply_expr {power == OFF || \
                             ground == OFF}}
```

Other Technology Info

- Technology Info
  - Add threshold information to level shifter strategies
- Library Cells
  - Map strategies to available ISO, LS, RET cells
- Location
  - Add -location info to strategies based on available cells
- Port States and PSTs
  - Add primary supply constraints for the system

Incremental Verification



Summary

- Successive Refinement enables
  - Clear communication between IP Provider and Consumer
    - Decreased risk and more successful usage of IP
  - Separation of Logical Design from Implementation
    - Earlier verification, before technology is known
    - Easier retargeting to different technologies
    - Easier debugging at each stage
  - Preservation of Verification Equity
    - No need to re-verify logical configuration for new technology