UPF Semantics and Usage

Erich Marschner

Verification Architect

Mentor Graphics

A Deeper Look at UPF Power Intent

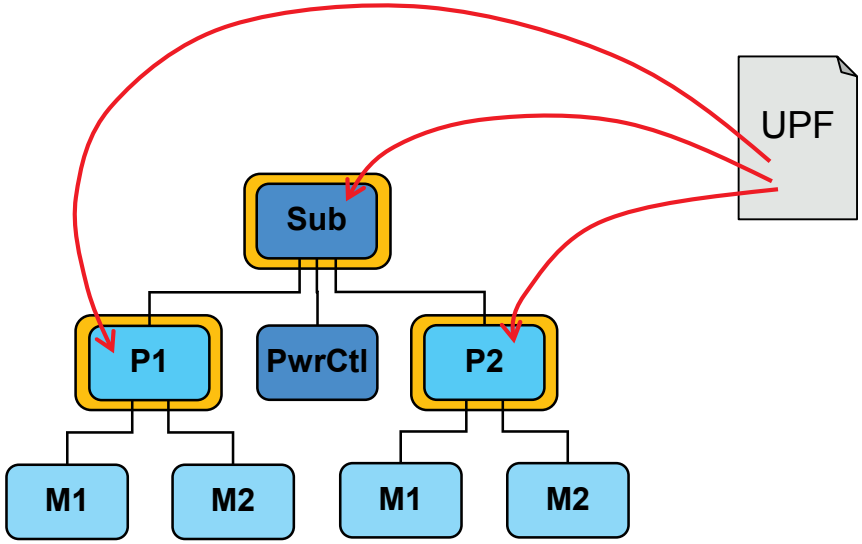
- Logic Hierarchy
- Power Domains
- Power Domain Supplies
- Supply Sets
- Supply Connections
- Power Related Attributes
- Power States and Transitions
- Power Domain State Retention
- Power Domain Interface Management
- Supply Network Construction
- Supply Equivalence

Logic Hierarchy

- **Design Hierarchy**
  - A hierarchical description in HDL
- **Logic Hierarchy**
  - An abstraction of the design hierarchy (instances only)
- **Scope**
  - An instance in the logic hierarchy
- **Design Top**
  - The topmost scope/instance in the logic hierarchy to which a given UPF file applies

Logic Hierarchy

- **Design Hierarchy**
  - Instances, generate stmts, block stmts, etc.
- **Logic Hierarchy**
  - Instances only
  - UPF objects
    - Created in instance scopes
    - Referenced with hierarchical names
- **Mapping to Floorplan**
  - May or may not reflect implementation
    - Depends upon the user and tools



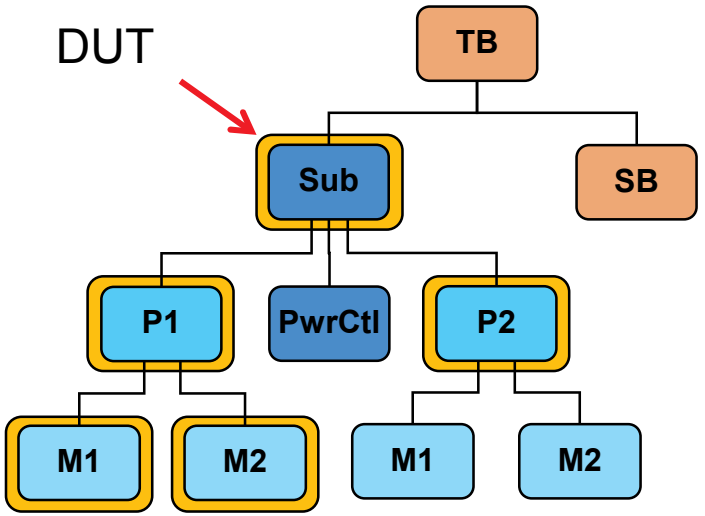
Navigation

- **set_design_top**

```
set_design_top TB/Sub
```

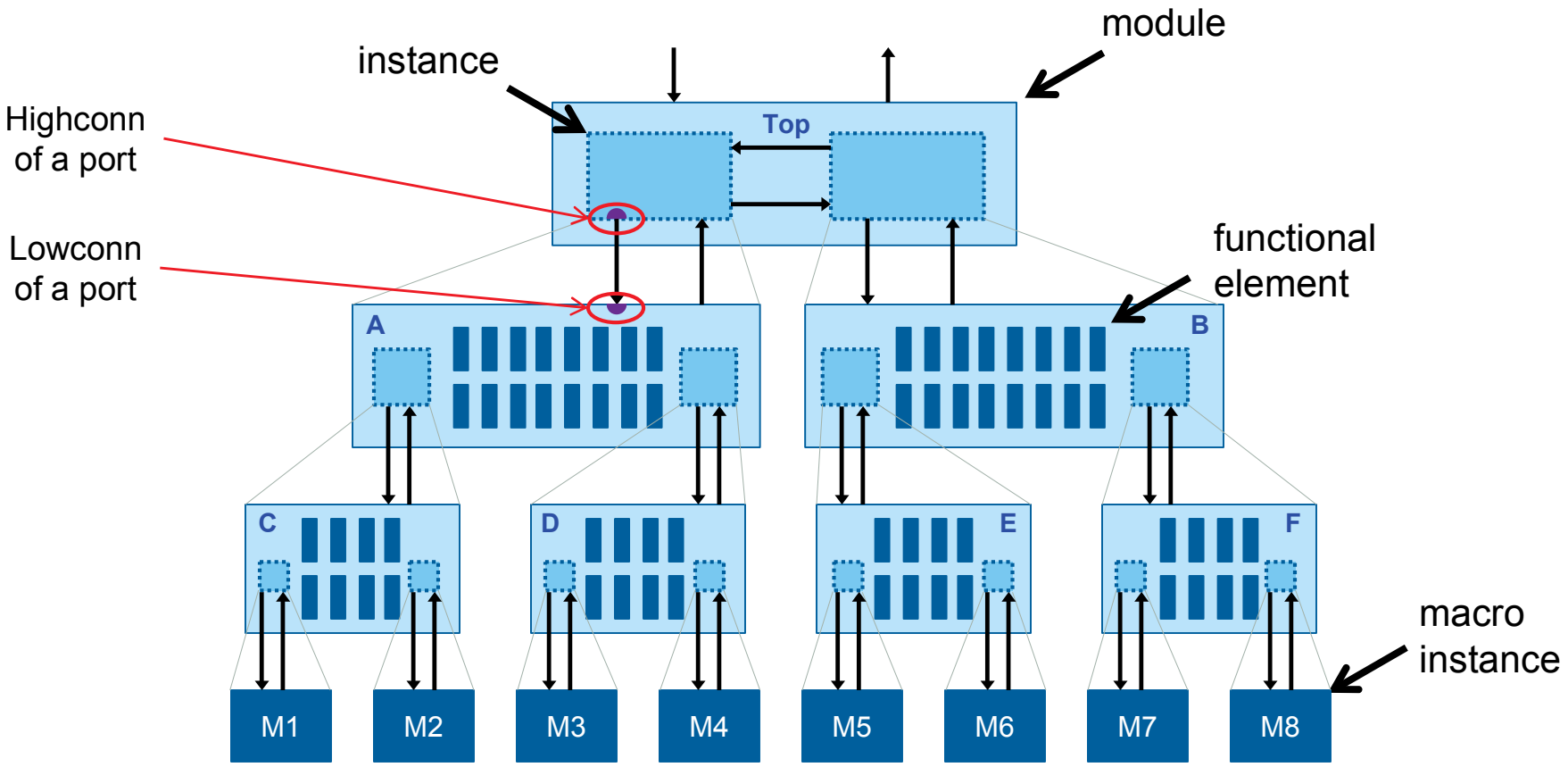
- **set_scope**

```
set_scope .
set_scope P1/M1
set_scope ..
set_scope M2
set_scope /P2
```



Note:
These are all instance names

Logic Hierarchy

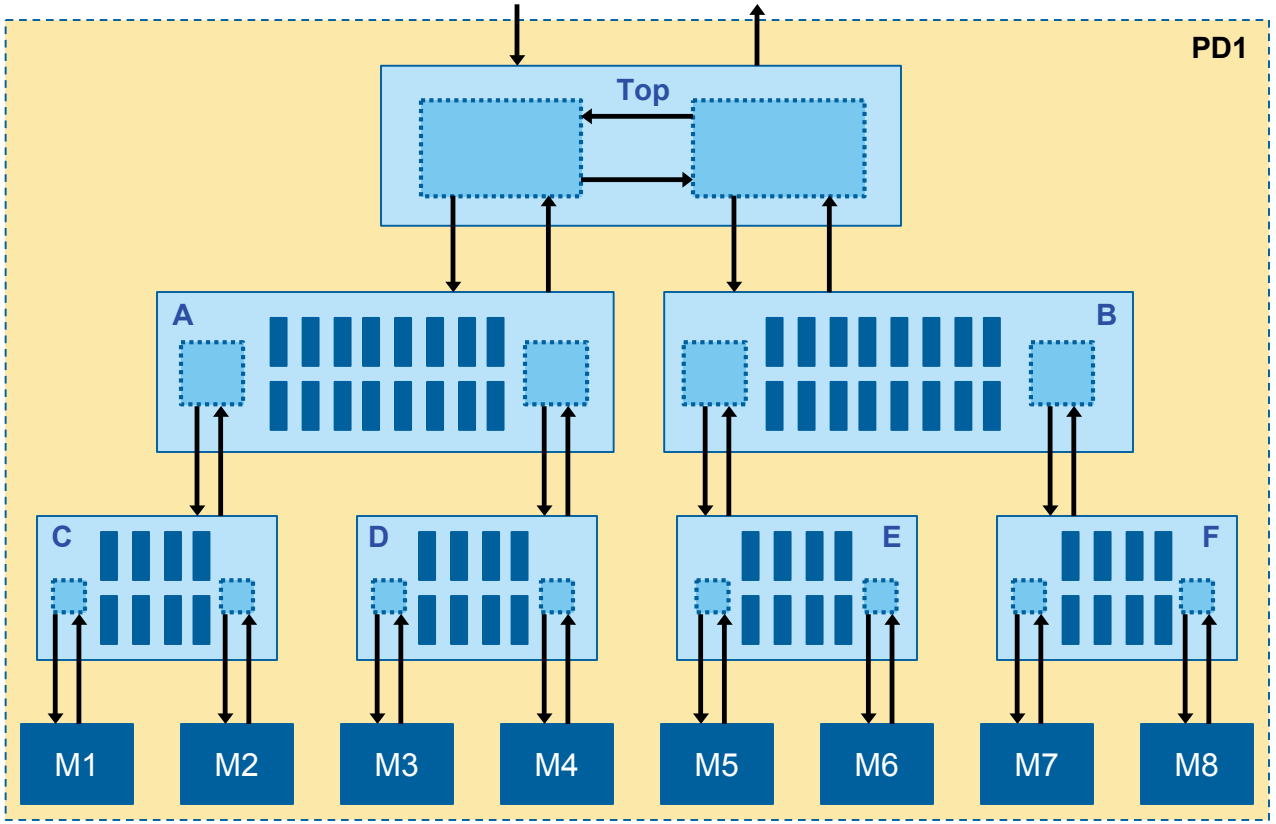


Power Domains

- **Partition the Logic Hierarchy**
  - Every instance must be in (the extent of) exactly one domain
- **Can be further partitioned**
  - A subtree of the design can be carved out as another domain
- **Unless declared "atomic"**
  - Atomic power domains cannot be further subdivided
- **Can be composed into larger domains**
  - If all subdomains have the same primary power supply
- **Have an upper and a lower boundary**
  - Boundaries represent a change in primary supply

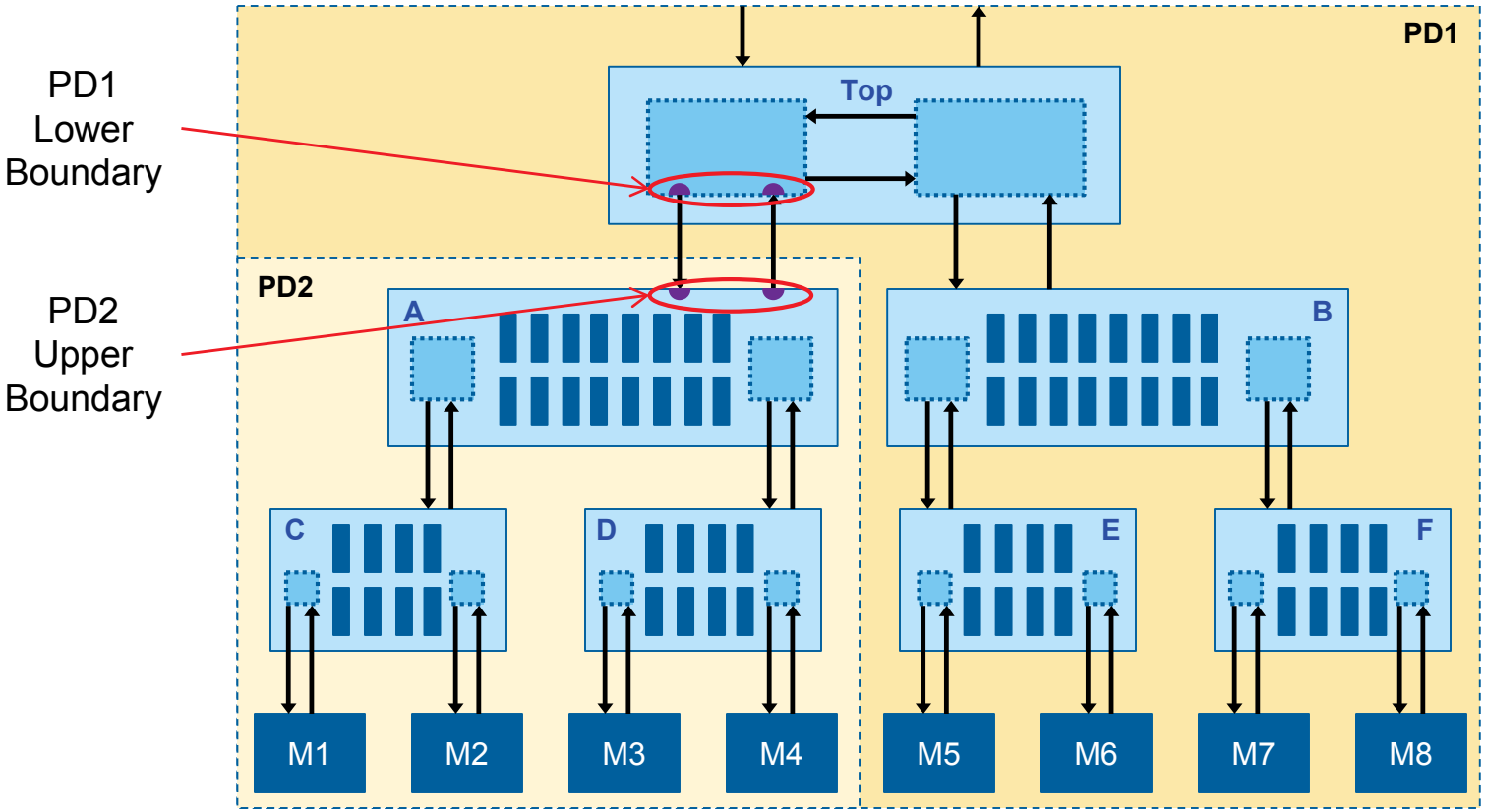
Partitioning the Logic Hierarchy - 1

```
create_power_domain PD1 -elements {.} …
```



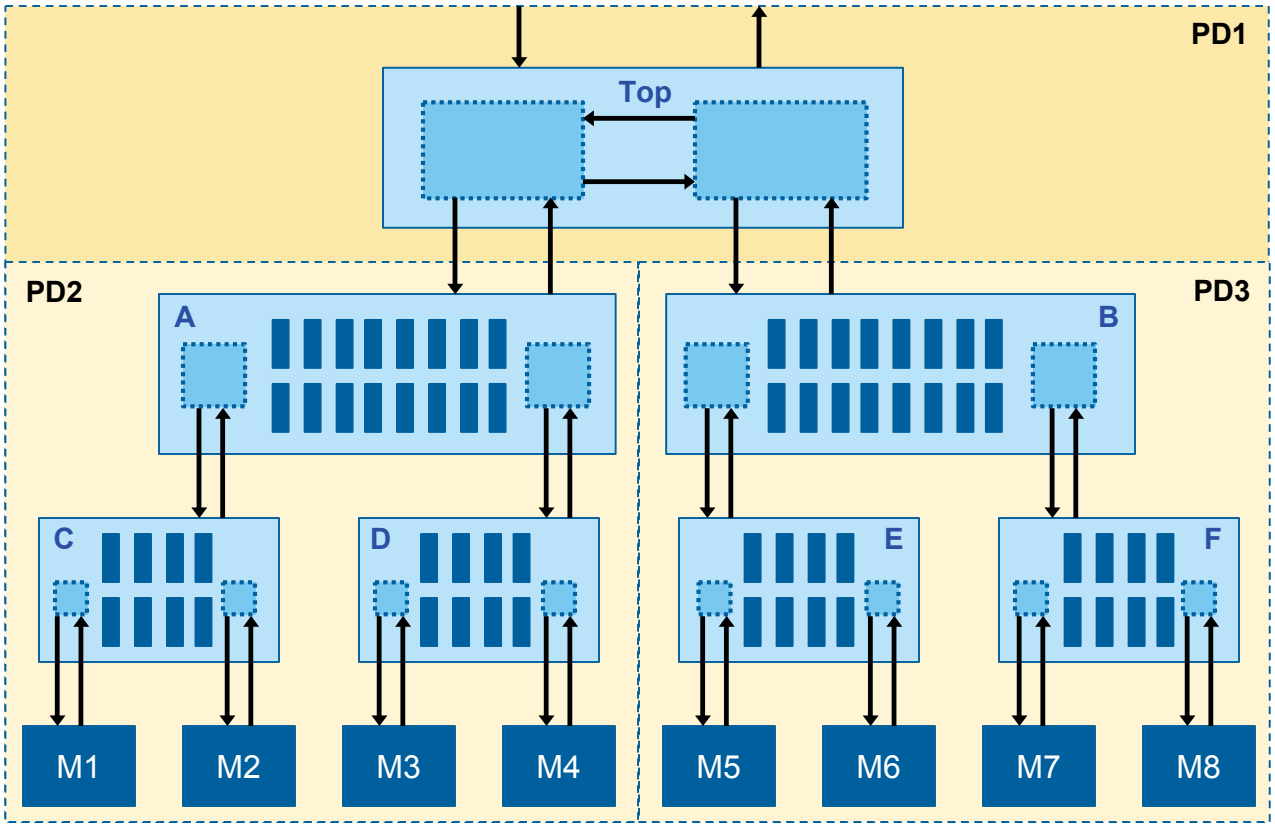
Partitioning the Logic Hierarchy - 2

```
create_power_domain PD2 -elements {A} …
```



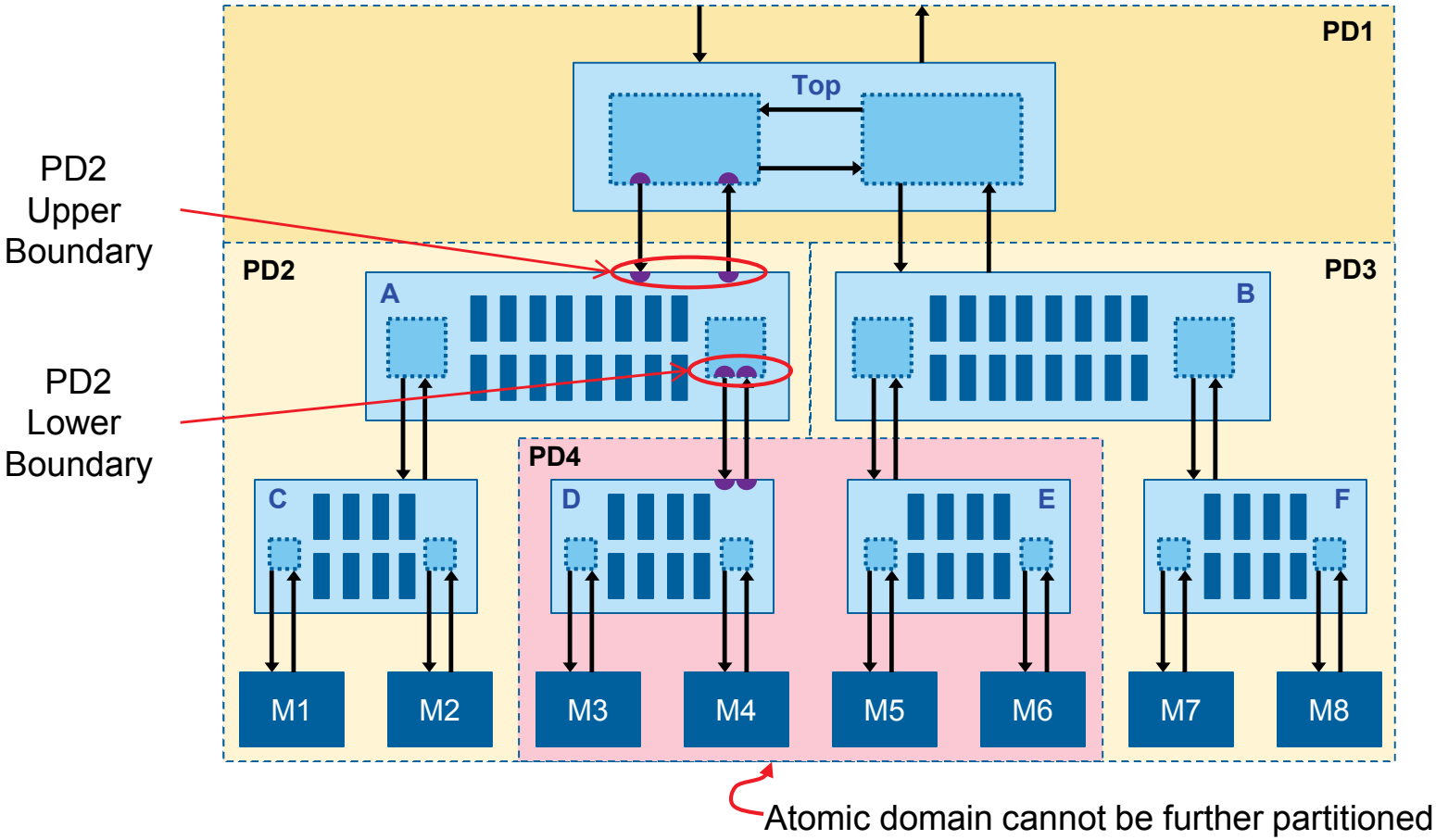
Partitioning the Logic Hierarchy - 3

```
create_power_domain PD3 -elements {B} …
```



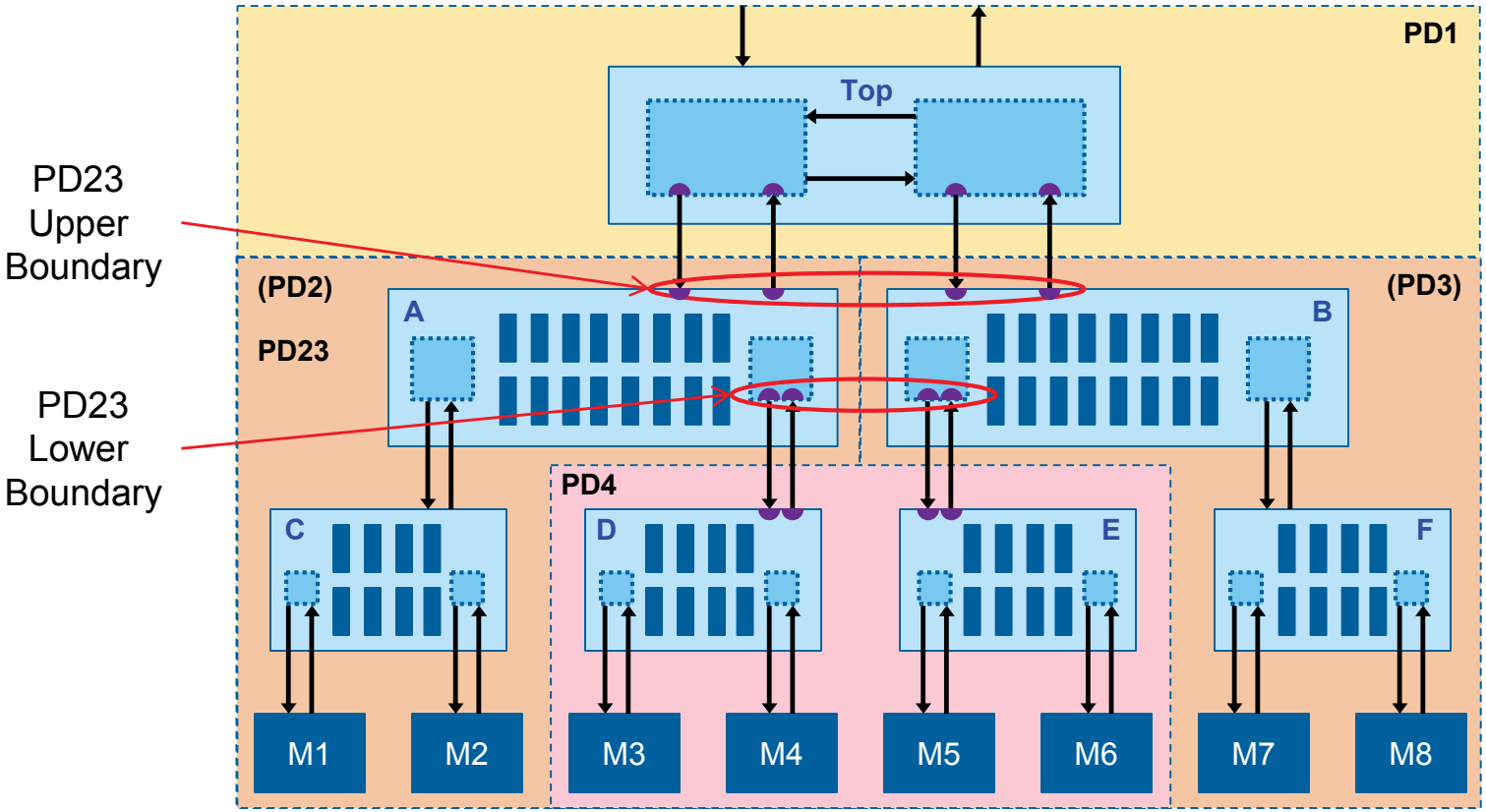
Partitioning the Logic Hierarchy - 4

```
create_power_domain PD4 -elements {A/D B/E} -atomic …
```



Partitioning the Logic Hierarchy - 5

```
create_composite_domain PD23 -subdomains {PD2 PD3} …
```



Power Domain Boundaries

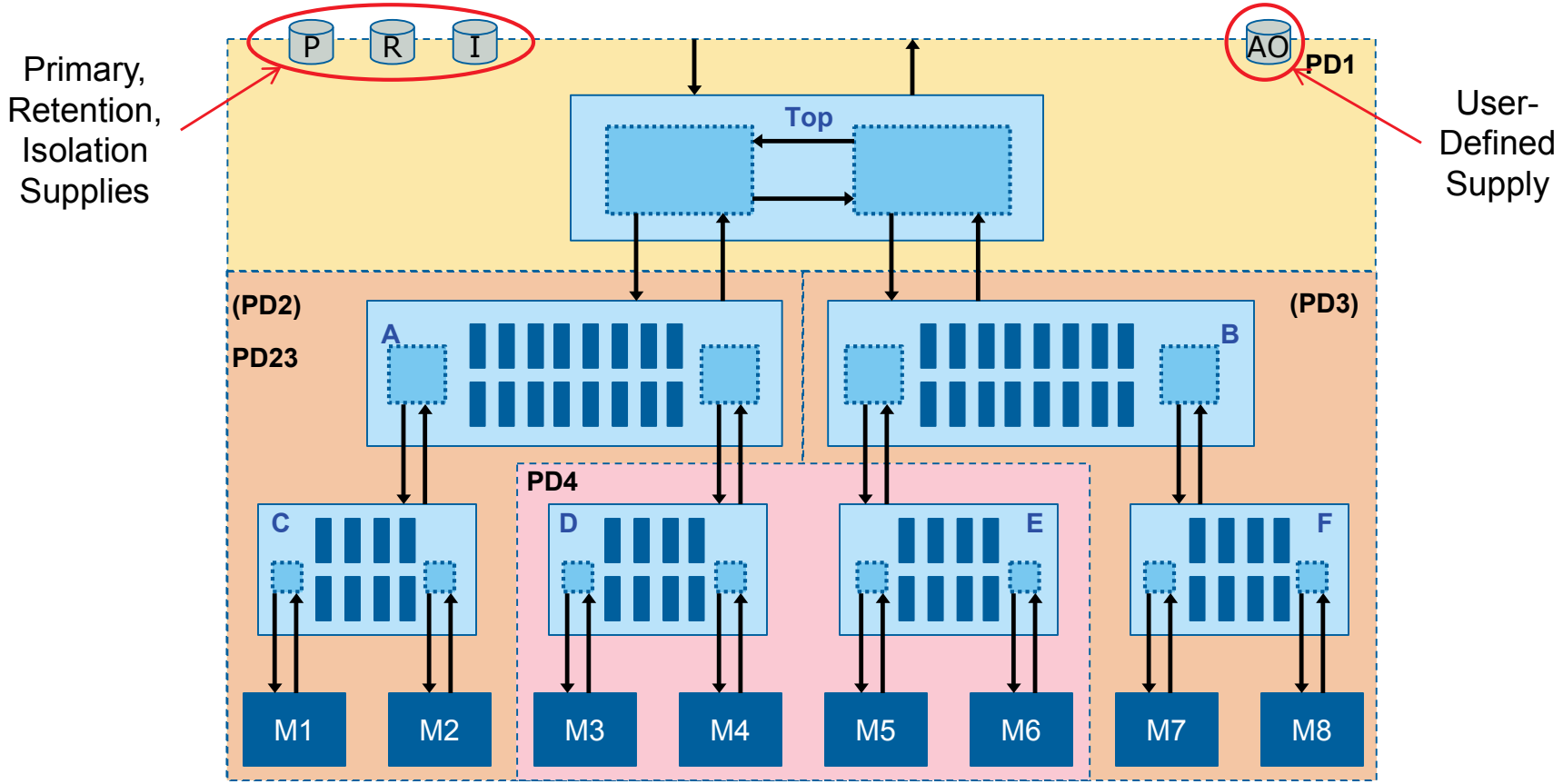
- **Define Domain Interfaces**
  - Isolation/Level shifting are only inserted at power domain boundaries
- **Upper Boundary**
  - Includes lowconn of declared ports of "top-level" instances
- **Lower Boundary**
  - Includes highconn of ports of instances in another domain
  - Includes macro instance ports with different supplies
- **Macro Instances**
  - May have multiple supplies
  - Each port may have a different supply

Domain Supplies

- **Primary supply**
  - Provides the main power, ground supplies for cells in the domain
  - Can also provide additional supplies (nwell, pwell, …)
- **Default retention supply**
  - Provides a default supply for saving the state of registers
- **Default isolation supply**
  - Provides a default supply for input or output isolation
- **Additional user-defined supplies**
  - Can be defined for particular needs (e.g., hard macros)
- **Available supplies**
  - Can be used by tools to power buffers used in implementation

Power Domain Supply Sets

```
create_power_domain PD1 -supply {AO} …
```

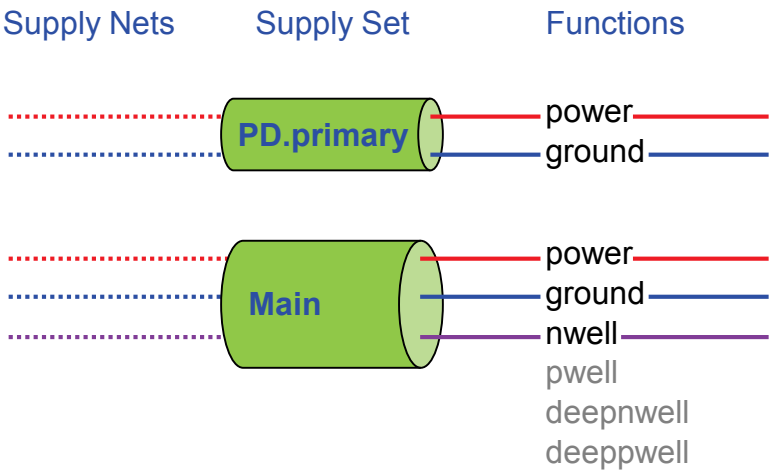


Supply Sets

- **Consists of a set of up to 6 supply "functions"**
  - power, ground, nwell, pwell, deepnwell, deeppwell
- **Represent a collection of supply nets**
  - One supply net per (required) function
- **Can be "global" or "local" to a power domain**
  - Power domains have a few predefined supply set "handles"
- **Can be associated with one another**
  - To model supply connections abstractly
- **Have power states with simstates**
  - Determine domain functionality in power aware simulation

Supply Sets

- A group of related supply nets
- Functions represent nets
  - which can be defined later
- Electrically complete model
  - power, ground, etc.
- Predefined supply sets
  - for domains
- User-defined supply sets
  - for domains (local)
  - standalone (global)
- Supply set parameters
  - for strategies



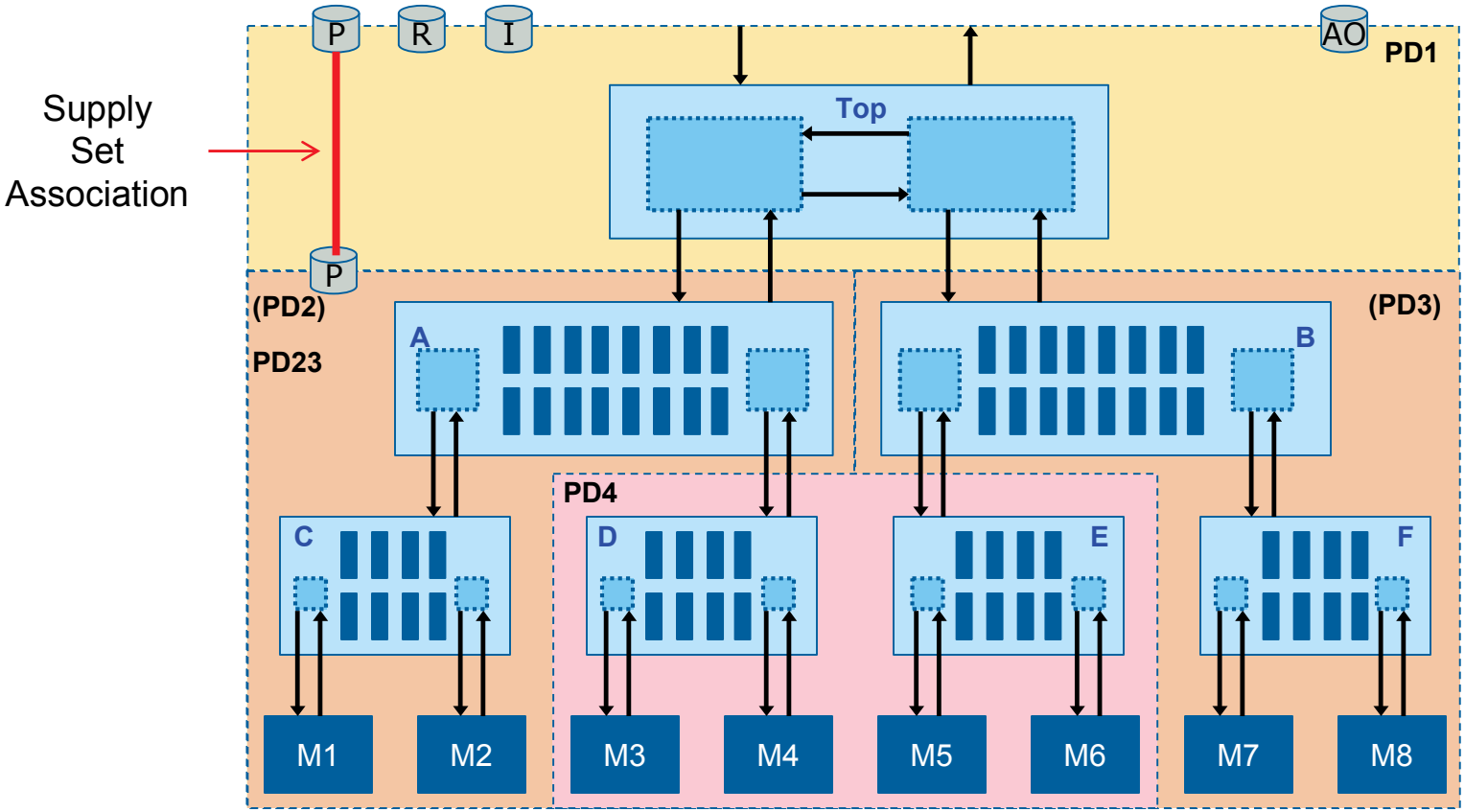

```
create_power_domain PD \
  -supply {primary} \
  -supply {backup}
```
(predefined)
(user-defined)

```
create_supply_set Main \
  -function {power} \
  -function {ground} \
  -function {nwell}
```
(user-defined)

```
set_isolation ISO -domain PD \
  -isolation_supply_set PD.backup
```

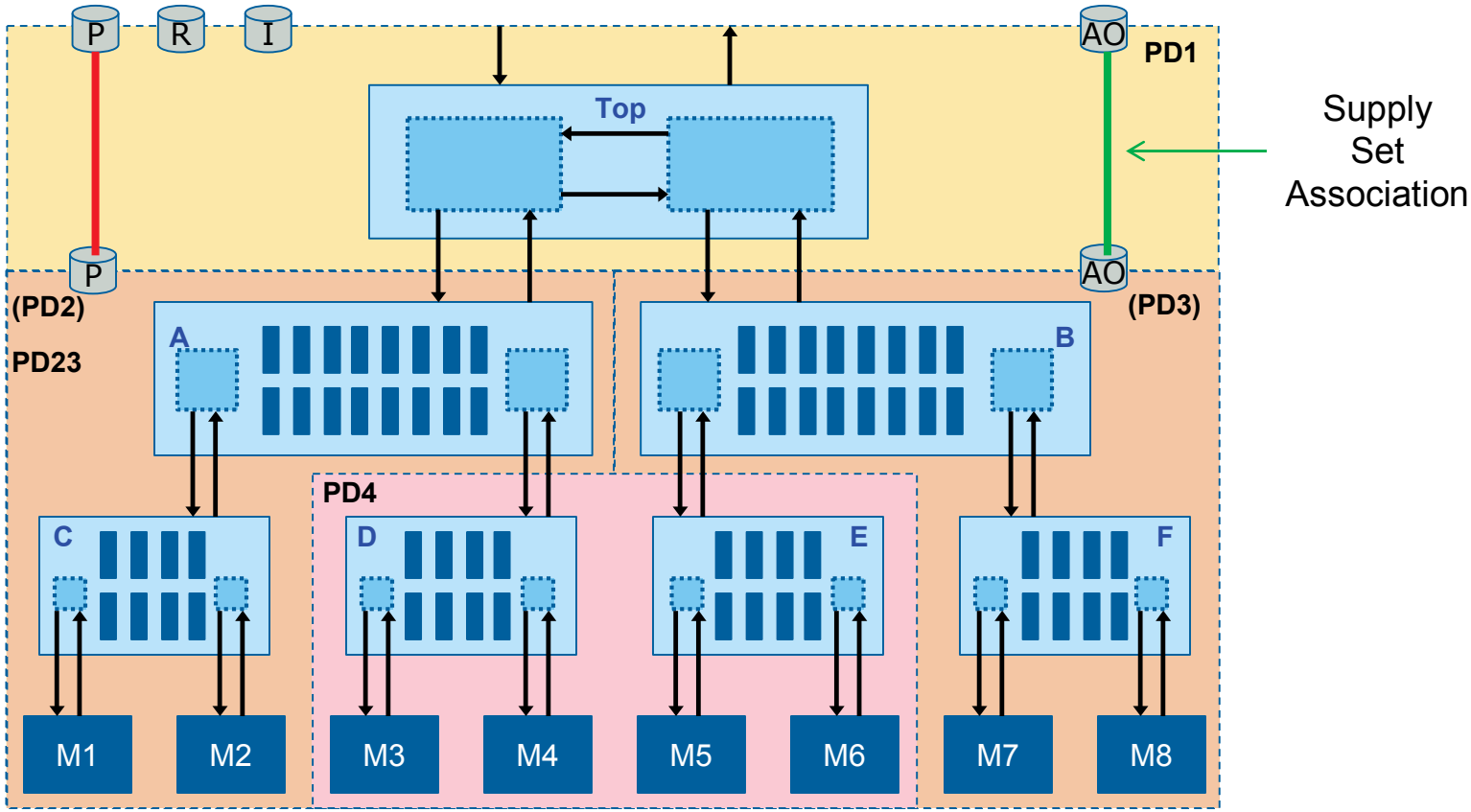
Associating Supply Sets 1

```
associate_supply_set PD1.primary -handle PD2.primary
```



Associating Supply Sets 2

```
associate_supply_set PD1.AO -handle PD3.AO
```



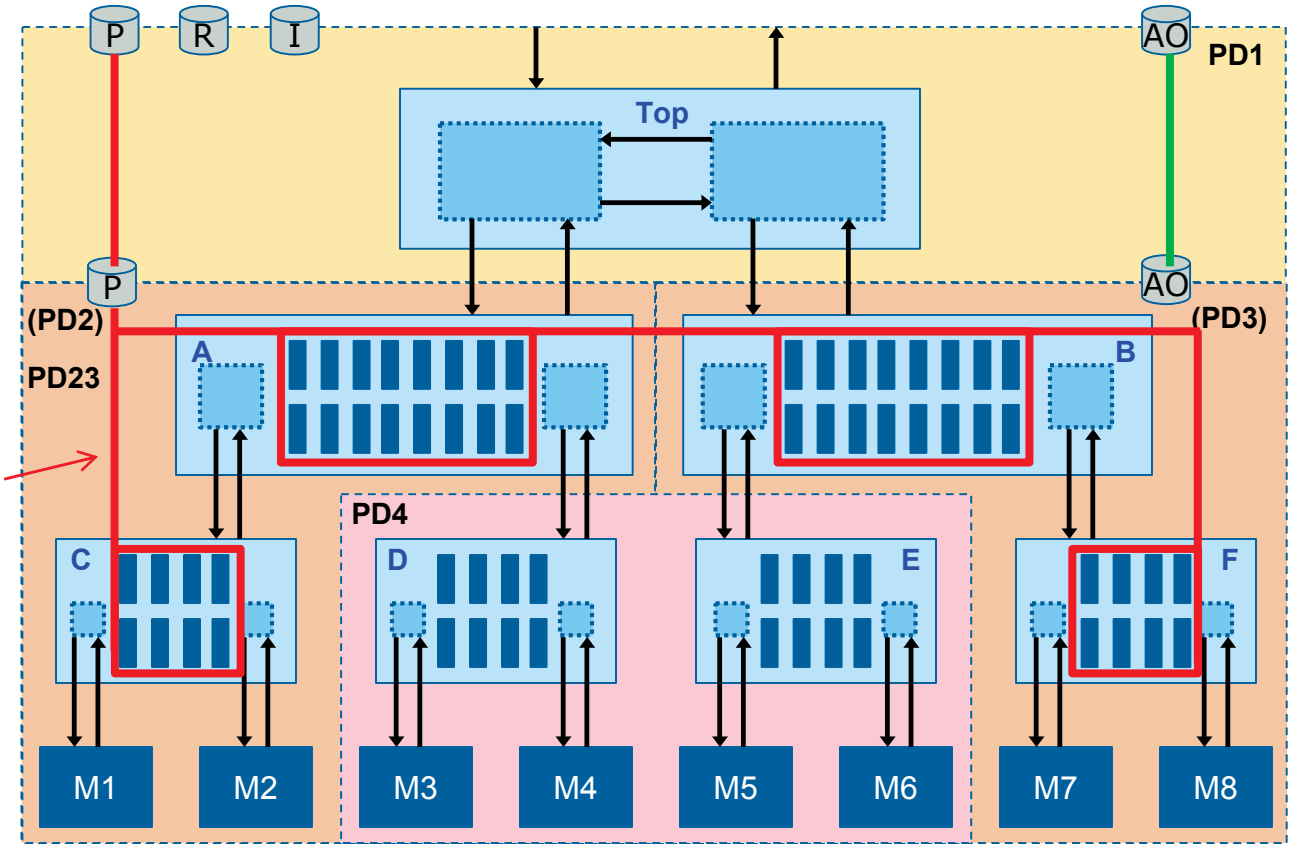
Supply Connections

- **Implicit connections**
  - Primary supply is implicitly connected to std cells
- **Automatic connections**
  - Supplies can be connected to cell pins based on pg_type
- **Explicit connections**
  - Supplies can be connected explicitly to a given pin
- **Precedence rules apply**
  - Explicit overrides Automatic overrides Implicit
- **Supply states determine cell behavior**
  - Cells function when supply is on,
  - Cells outputs are corrupted when supply is off

Implicit Supply Connections

...

Implicit Connections: power, ground only

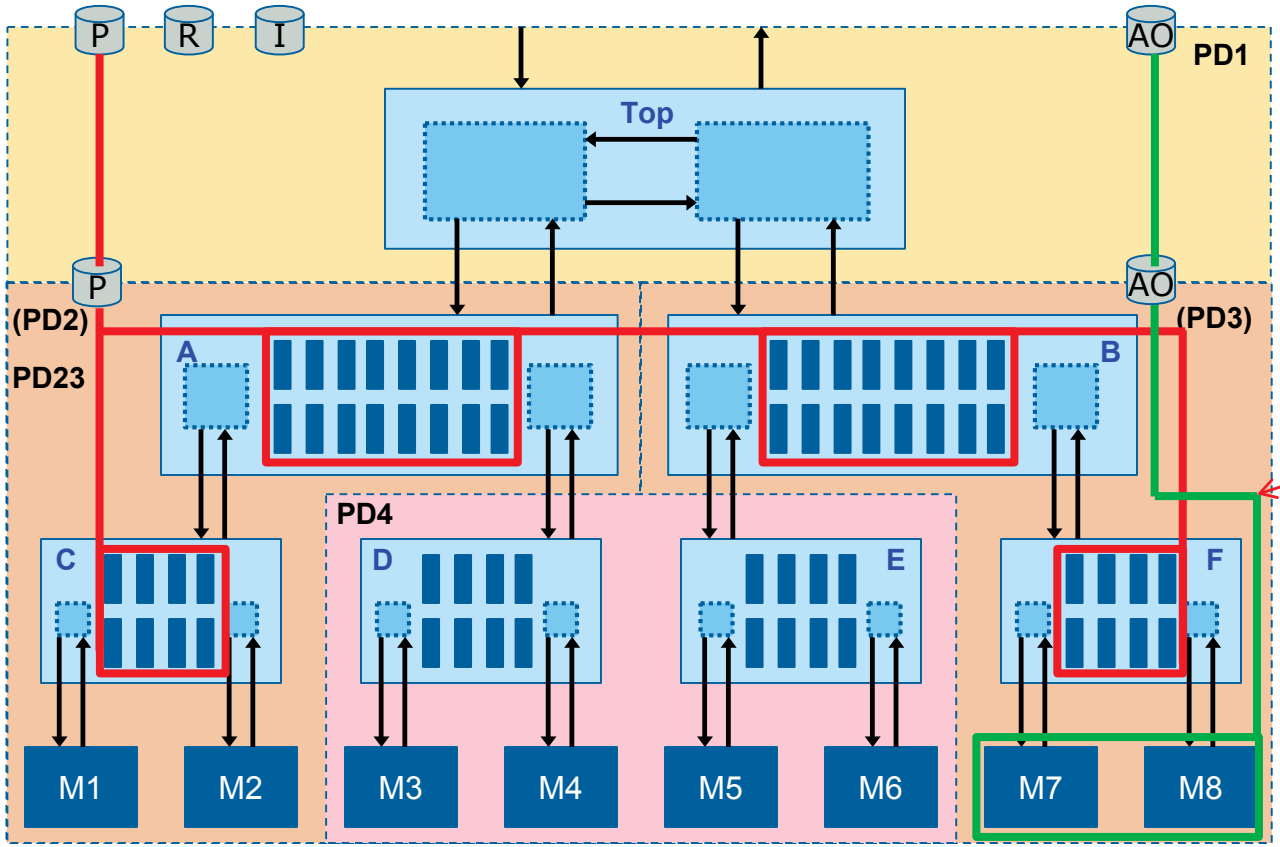


PG Types

- **Describe the usage of supply pins of cells, macros**
  - primary_power, primary_ground
  - backup_power, backup_ground
  - internal_power, internal_ground
  - nwell, pwell, deepnwell, deeppwell
- **Typically defined in Liberty library models**
  - primary power/ground are common to all
- **Can also be defined in HDL or UPF**
  - Using attributes …
- **Drive implicit and automatic supply connections**
  - Each function of a domain supply set maps to a pg_type

Automatic Supply Connections

```
connect_supply_set PD3.AO -elements {B/F/M7 B/F/M8}
```



Automatic Connections based on pg_type

PG Type-Driven Connections

- **Automatic connection**

```
connect_supply_set PD3.AO -elements {B/F/M7 B/F/M8} \
  -connect {power primary_power} \
  -connect {ground primary_ground}

connect_supply_set PD3.AO -elements {B/F/M7 B/F/M8} \
  -connect {power backup_power} \
  -connect {ground backup_ground}
```

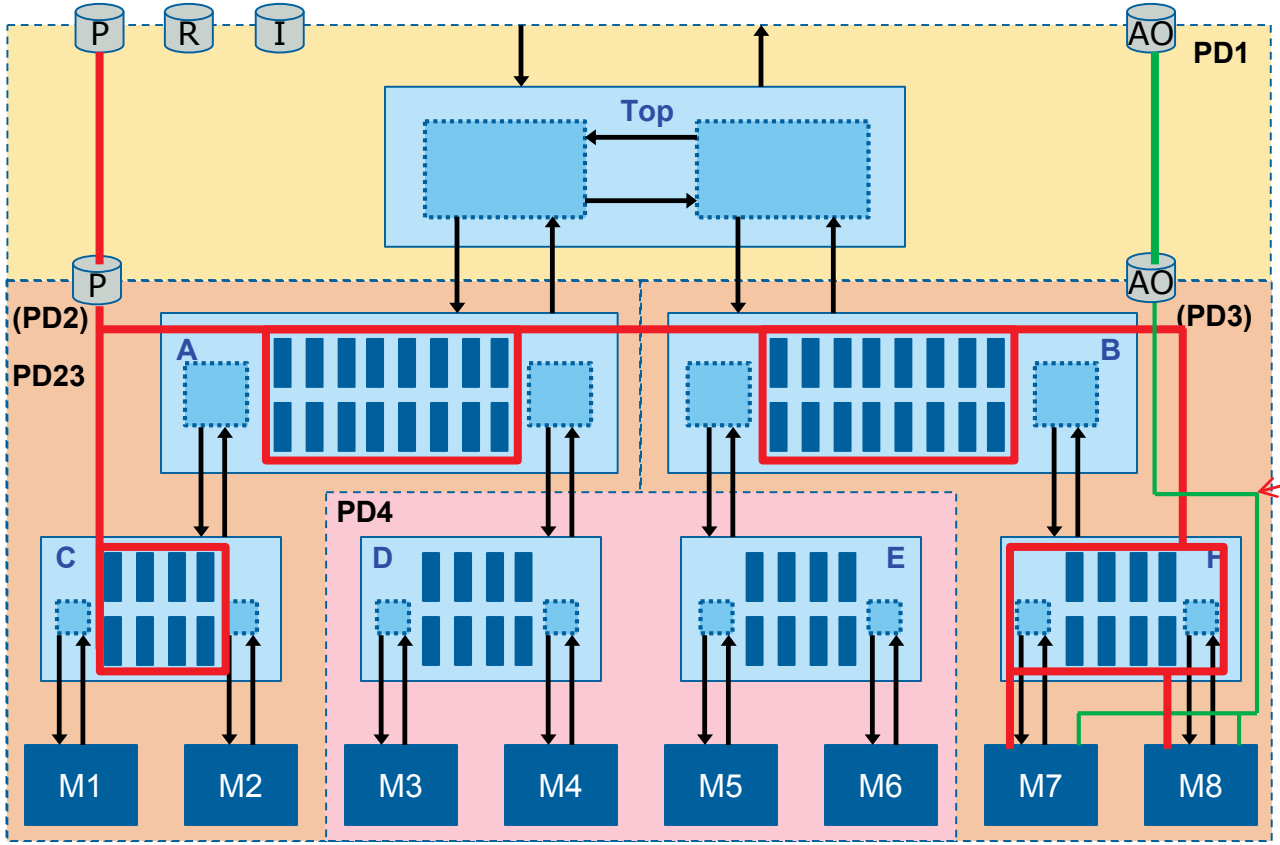
- **Implicit connection**

– Equivalent to

```
connect_supply_set PD2.primary -elements {.} \
  -connect {power primary_power} \
  -connect {ground primary_ground}
```

Explicit Supply Connections

```
connect_supply_net PD3.AO.power -ports {B/F/M7/VDDB …}
```



Power Attributes

- **Characteristics of a port or design element**
  - That relate to power intent or implementation
- **Defined in UPF, HDL, or Liberty**
  - Liberty and HDL attributes are imported into UPF
- **Used to identify power supplies for ports**
  - Related supplies for ports and cell pins
- **Used to specify constraints for IP usage**
  - Clamp value constraints for isolation of ports
- **Used to specify structure and behavior information**
  - Hierarchy leaf/macro cells, net connections, simstate use

Predefined UPF Attributes

- **Supply Attributes**
  - UPF_pg_type
  - UPF_related_power_port
  - UPF_related_ground_port
  - UPF_related_bias_ports
  - UPF_driver_supply
  - UPF_receiver_supply
- **Isolation Attributes**
  - UPF_clamp_value
  - UPF_sink_off_clamp_value
  - UPF_source_off_clamp_value
- **Structural Attributes**
  - UPF_is_leaf_cell
  - UPF_is_macro_cell
  - UPF_feedthrough
  - UPF_unconnected
- **Behavioral Attributes**
  - UPF_retention
  - UPF_simstate_behavior

Attribute Definitions

UPF

```
set_port_attributes -ports Out1 \
  -attribute \
    {UPF_related_power_port "VDD"

set_port_attributes -ports Out1 \
  -attribute \
    {UPF_related_ground_port "VSS"


set_port_attributes -ports Out1 \
  -related_power_port "VDD" \
  -related_ground_port "VSS"
```

HDL

SystemVerilog or Verilog-2005

```
(* UPF_related_power_port = "VDD",
   UPF_related_ground_port = "VSS" *)

  output Out1;
```

VHDL

```
attribute UPF_related_power_port of
  Out1: signal is "VDD";

attribute UPF_related_ground_port of
  Out1: signal is "VSS";
```

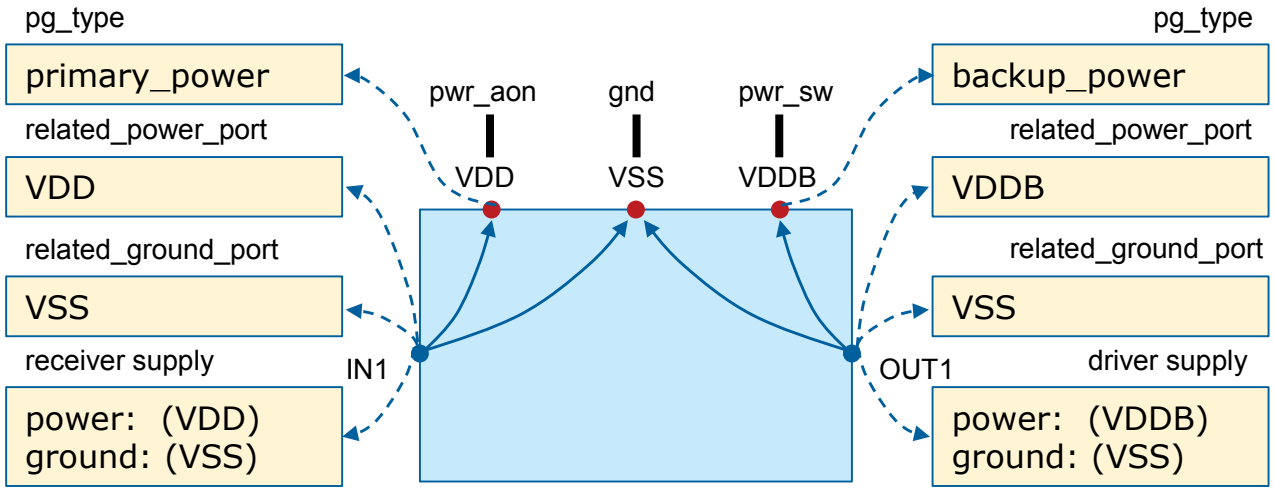
Liberty

- related_power_pin, related_ground_pin
- pg_type, related_bias_pins, is_macro_cell, etc.

UPF Attribute Usage

- **Supply Attributes**
  - UPF_pg_type
  - UPF_related_power_port
  - UPF_related_ground_port
  - UPF_related_bias_ports
  - UPF_driver_supply
  - UPF_receiver_supply

- **Used to specify**
  - cell/macro supply port types
  - logic port related supplies
  - primary IO port related supplies
  - driver/receiver supply sets
    - can be defined only in UPF
    - no supply set data type in HDL or Liberty



UPF Attribute Usage

- **Supply Attributes**
  - UPF_pg_type
  - UPF_related_power_port
  - UPF_related_ground_port
  - UPF_related_bias_ports
  - UPF_driver_supply
  - UPF_receiver_supply

- **Used to specify**
  - cell/macro supply port types
  - logic port related supplies
  - primary IO port related supplies
  - driver/receiver supply sets
    - can be defined only in UPF
    - no supply set data type in HDL or Liberty

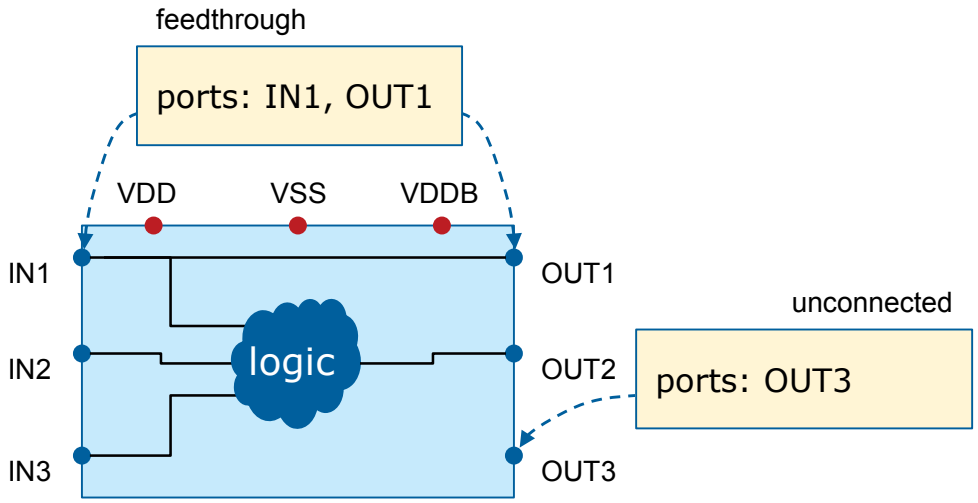
- **Isolation Attributes**
  - UPF_clamp_value
  - UPF_sink_off_clamp_value
  - UPF_source_off_clamp_value

- **Used to specify**
  - clamp value requirements in case source is powered off when sink is powered on
    - used to define power constraints for IP blocks

UPF Attribute Usage

- **Used to identify**
  - leaf cells in the hierarchy
  - macro cells in the hierarchy
  - feedthrough paths through a macro cell
  - unconnected macro ports

- **Structural Attributes**
  - UPF_is_leaf_cell
  - UPF_is_macro_cell
  - UPF_feedthrough
  - UPF_unconnected



UPF Attribute Usage

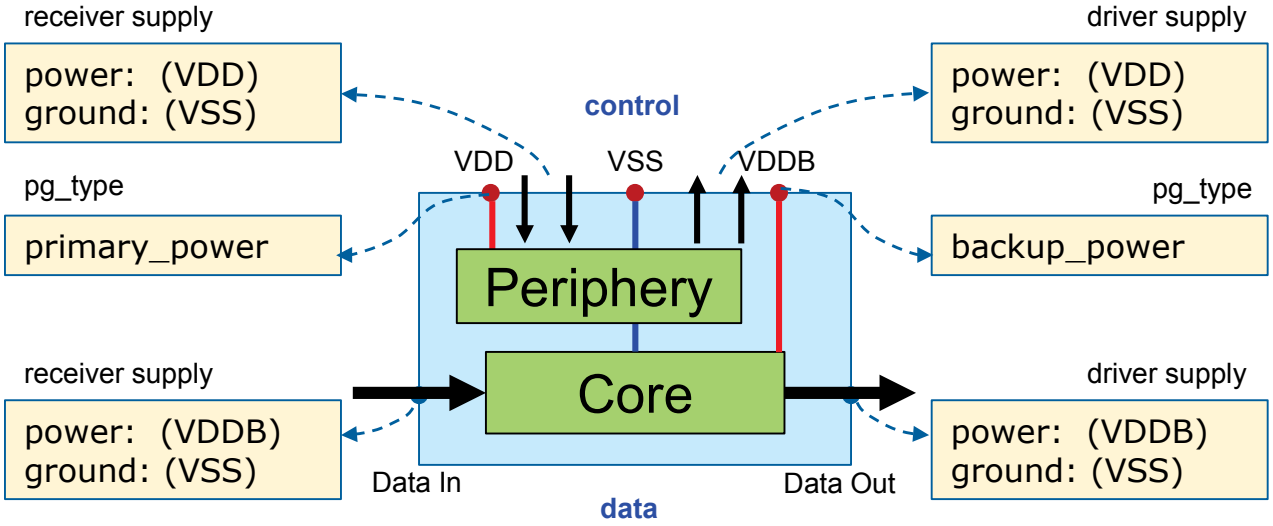
- **Used to identify**
  - leaf cells in the hierarchy
  - macro cells in the hierarchy
  - feedthrough paths through a macro cell
  - unconnected macro ports

- **Structural Attributes**
  - UPF_is_leaf_cell
  - UPF_is_macro_cell
  - UPF_feedthrough
  - UPF_unconnected

- **Used to define**
  - whether state retention is required for a given element
  - whether simstates determine power aware behavior (i.e., corruption)

- **Behavioral Attributes**
  - UPF_retention
  - UPF_simstate_behavior

Hard Macro Supplies



- **Modeled with Attributes**
  - Attributes of cell pins:
    - PG type attributes
    - Related supply attributes
    - In UPF, HDL, or Liberty
  - Imply anonymous supply sets

In a memory cell with separate supplies for peripheral logic and memory core, different ports will have different driver supplies or receiver supplies.

Supply Power States

- **Defined on supply sets**
  - In particular, power domain primary supply
- **Represent how cells behave in various situations**
  - When / whether cell outputs are corrupted
- **Defined by a logic expression**
  - State holds when logic expression is TRUE
- **Also may include a supply expression**
  - Defines the legal values of supply set fns when in that state
- **Also includes a simstate**
  - Simstate defines precise simulation semantics in this state
- **Not necessarily mutually exclusive!**

Supply Set Power State Definition

- **Simple**

```
add_power_state PD1.primary -supply \
  -state {ON  -logic_expr {PwrOn}  -simstate NORMAL} \
  -state {OFF -logic_expr {!PwrOn} -simstate CORRUPT}
```

- **More Complex**

```
add_power_state PD1.primary -supply \
  -state {RUN -logic_expr {PwrOn && !Sleep && Mains} \
    -simstate NORMAL} \
  -state {LOW -logic_expr {PwrOn && !Sleep && Battery} \
    -simstate NORMAL} \
  -state {SLP -logic_expr {PwrOn && Sleep} \
    -simstate CORRUPT_ON_CHANGE} \
  -state {OFF -logic_expr {!PwrOn} \
    -simstate CORRUPT}
```

Simstates - Precedence and Meaning

lower

- **NORMAL**
- **CORRUPT_STATE_ON_CHANGE**
- **CORRUPT_STATE_ON_ACTIVITY**
- **CORRUPT_ON_CHANGE**
- **CORRUPT_ON_ACTIVITY**
- **CORRUPT**

higher

- **NORMAL**
  - Combinational logic functions normally
  - Sequential logic functions normally
  - Both operate with characterized timing
- **CORRUPT_STATE_ON_CHANGE**
  - Combinational logic functions normally
  - Sequential state/outputs maintained as long as outputs are stable
- **CORRUPT_STATE_ON_ACTIVITY**
  - Combinational logic functions normally
  - Sequential state/outputs maintained as long as inputs are stable
- **CORRUPT_ON_CHANGE**
  - Combinational outputs maintained as long as outputs are stable
  - Sequential state/outputs corrupted
- **CORRUPT_ON_ACTIVITY**
  - Combinational outputs maintained as long as inputs are stable
  - Sequential state/outputs corrupted
- **CORRUPT**
  - Combinational outputs corrupted
  - Sequential state/outputs corrupted

Domain Power States

- **Defined on power domains**
  - In particular, power domains representing an IP block
- **Represent aggregate state of supplies, subdomains**
  - Abstract functional/power modes of a component
- **Defined by a logic expression (like supply set states)**
  - Typically refers to power states of other objects
- **Does NOT include a supply expression**
  - Supply expressions are only for supply set power states
- **Does NOT include a simstate**
  - Simstates are only for supply set power states
- **Not necessarily mutually exclusive!**

Domain Power State Definition

Examples

```
add_power_state PD_TOP -domain \
  -state {Normal \
    -logic_expr \
      {primary == ON && \
       backup  == ON && \
       PD_mem  == UP} } \
 -state {Sleep \
    -logic_expr \
      {primary == OFF && \
       backup  == ON && \
       PD_mem  == UP} } \
 -state {Off\
    -logic_expr \
      {primary == OFF && \
       backup  == OFF && \
       PD_Mem  == DOWN} }
```

Examples

```
add_power_state PD_Mem -domain \
  -state {UP \
    -logic_expr {primary == ON}} \
  -state {RET \
    -logic_expr {retention == ON}} \
  -state {DOWN \
    -logic_expr {retention == OFF}}
```

PD_TOP	.primary	.backup	PD_MEM
Normal	ON	ON	UP
Sleep	OFF	ON	RET
Off	OFF	OFF	DOWN

Power Management Strategies

- **Retention strategies**
  - Identify registers to retain, controls/conditions, and supplies
  - Must satisfy any retention constraints (clamp value attributes)
- **Repeater strategies**
  - Identify ports to be buffered and their supplies
  - Input and output ports can be buffered
- **Isolation strategies**
  - Define how to isolate ports where required - control, supplies
  - Actual isolation insertion is driven by source/sink power states
- **Level shifter strategies**
  - Define how to level-shift ports where required - supplies
  - Actual level shifter insertion is driven by threshold analysis

Retention Strategies

Balloon Latch

```
set_retention BL -domain PD1 \
  -elements {…} \
  -save_signal … \
  -restore_signal … \
  -retention_supply …
```

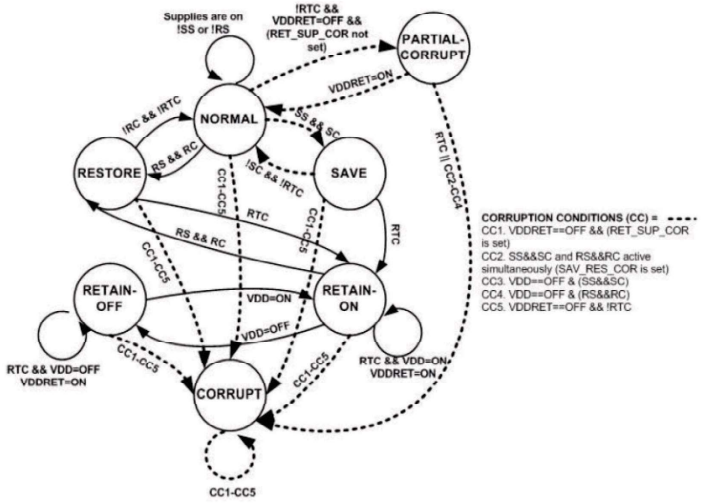


Figure 1—Retention state transition diagram for balloon-style retention

Live Slave Latch

```
set_retention LSL -domain PD1 \
  -elements {…} \
  -retention_condition … \
  -retention_supply …
```

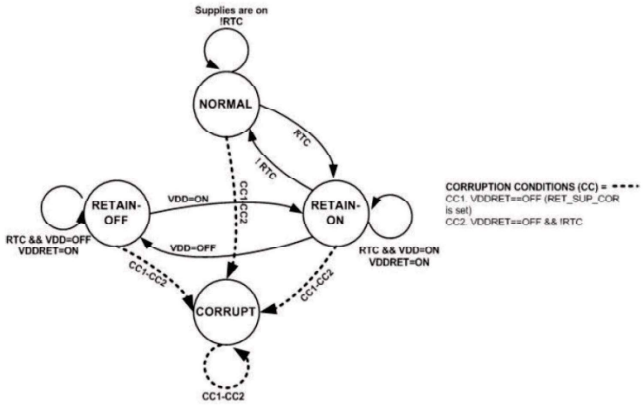


Figure 2—Retention state transition diagram for master/slave-alive style retention

Isolation Strategies

- **Specifying Ports**
  - Using `-elements` and `-exclude_elements`
  - Using filters: `-applies_to`, `-diff_supply_only`, `-sink`, `-source`
- **Precedence Rules**
  - More specific rules take precedence over more generic rules
- **Specifying Location**
  - Using locations `self`, `parent`, `other`, and `fanout`
- **Handling Fanout to Different Domains**
  - Using `-sink` to isolate different paths
- **Isolation Supplies and Cells**
  - Location affects default isolation supply and usable cell types

Specifying Ports

- **Elements list includes ports**

```
-elements { <port name> }
-elements { <instance name> }
-elements { . }
```

[if no -elements list, default is all ports of domain]

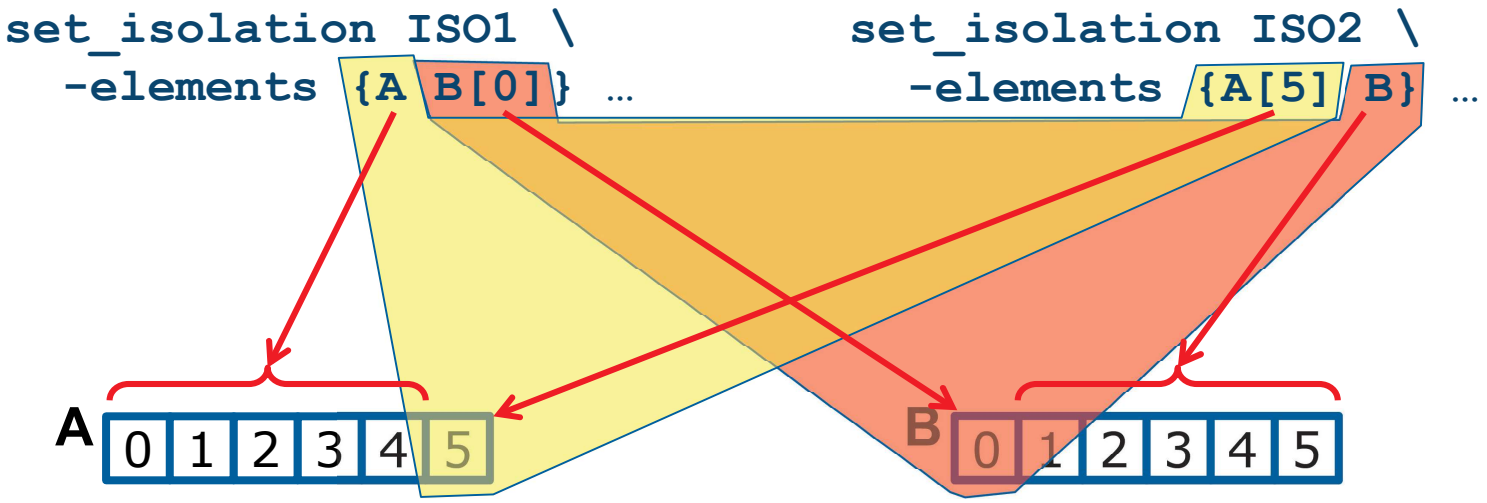
- **Exclude elements list excludes ports**

```
-exclude_elements { … }
```

- **Filters further limit the set of ports**

```
-applies_to <inputs | outputs | both>
-source <domain name> | <supply set name>
-sink <domain name> | <supply set name>
-diff_supply_only
```

What Happens if Multiple Strategies?



Precedence Rules for Strategies

lower

- Strategy for all ports of a specified power domain

```
set_isolation ISO1 -domain PD \
  …
```

- Strategy for all ports of a specified power domain with a given direction

```
set_isolation ISO2 -domain PD \
  -applies_to inputs …
```

- Strategy for all ports of an instance specified explicitly by name

```
set_isolation ISO3 -domain PD \
  -elements {i1} …
```

- Strategy for a whole port specified explicitly by name

```
set_isolation ISO4 -domain PD \
  -elements {i1/a i1/b} …
```

- Strategy for part of a multi-bit port specified explicitly by name

```
set_isolation ISO5 -domain PD \
  -elements {i1/a[3] i1/b[7]} …
```

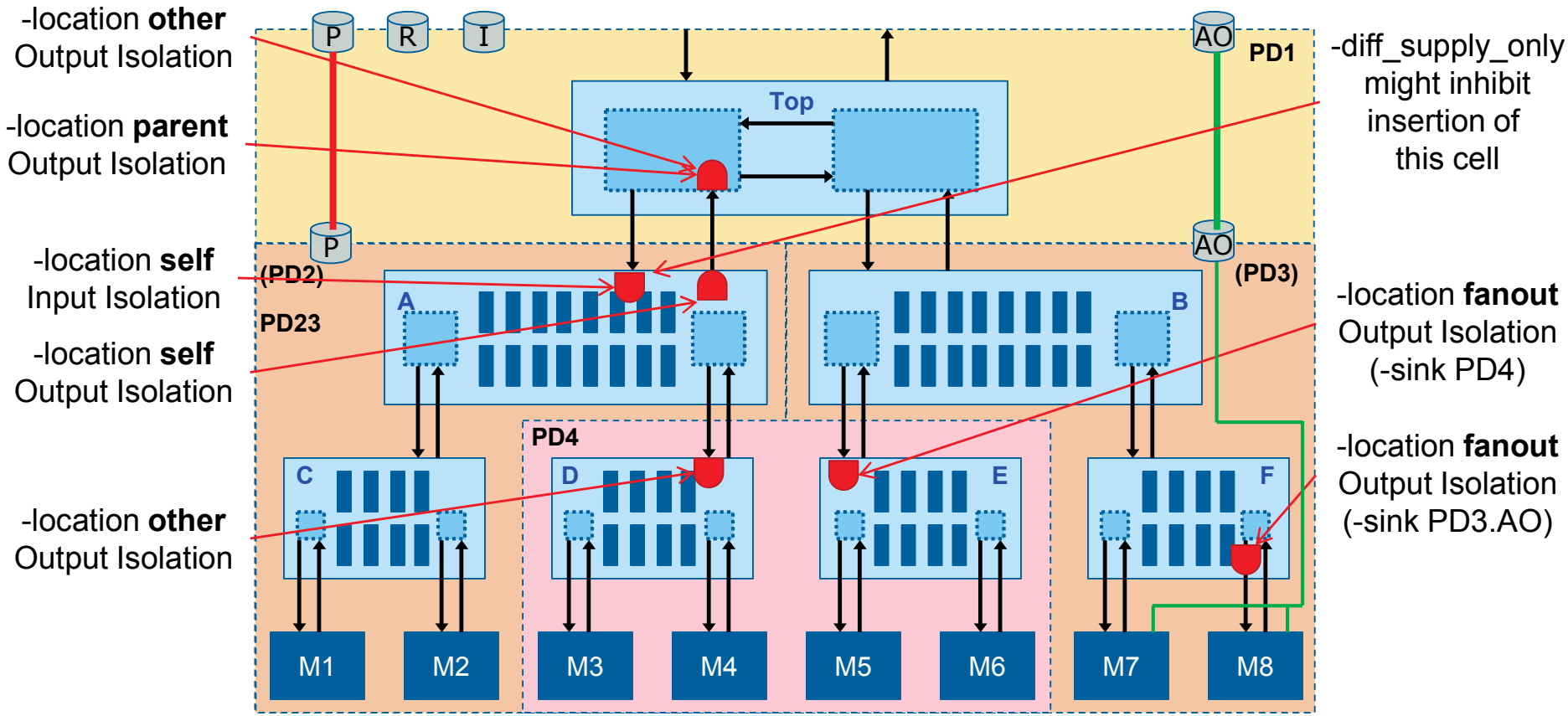
higher

Interface Cell Locations

- **Self**
  - The domain for which the strategy is defined
- **Parent**
  - The domain "above" the self domain
- **Other**
  - The domains "above" and "below" the self domain
- **Fanout**
  - The domain in which the receiving logic is contained

Isolation Cell Locations

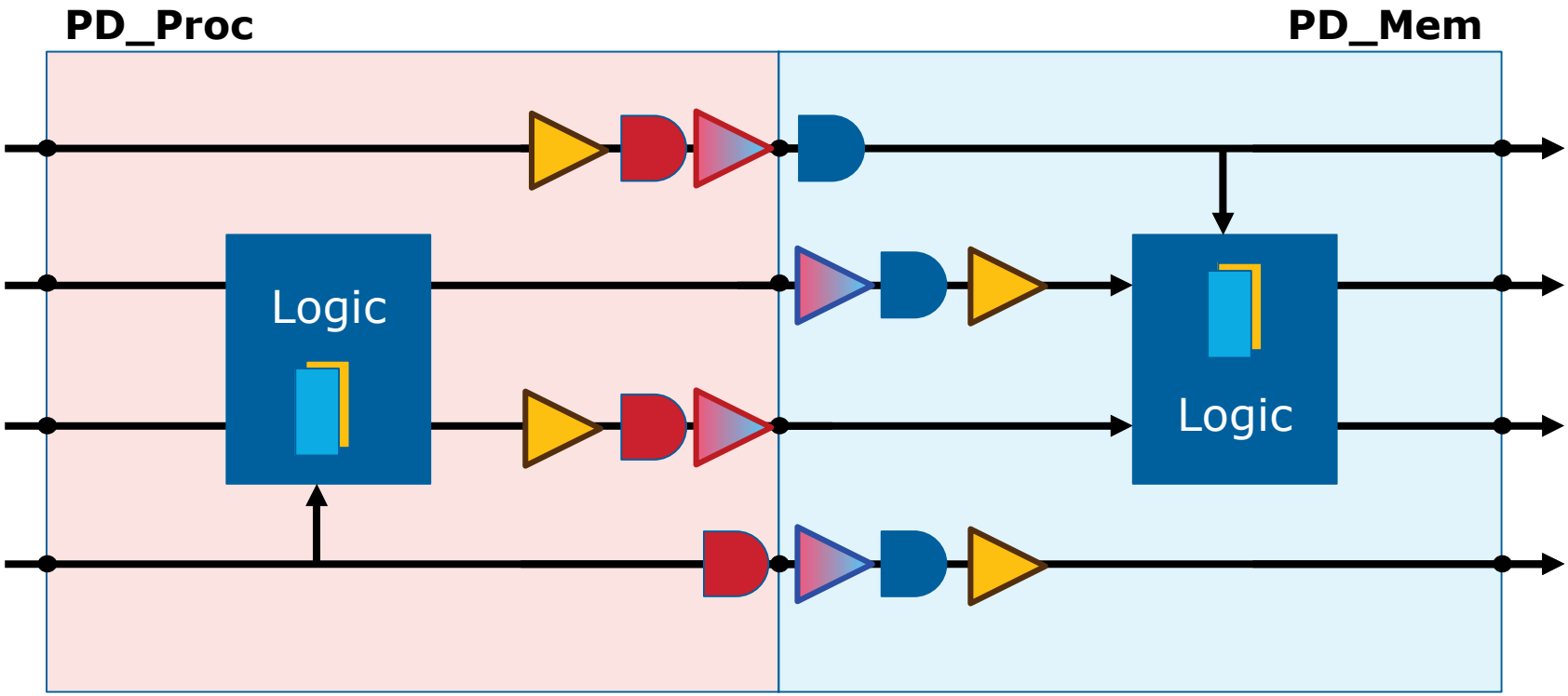
```
set_isolation ISO1 -domain PD2 -location …
```



Other Isolation Cell Parameters

- **Clamp Value**
  - specified with
    ```
    -clamp_value < 0 | 1 | any | Z | latch >
    ```
- **Control**
  - specified with
    ```
    -isolation_signal <signal name>
    -isolation_sense <high | low>
    ```
- **Supply**
  - specified with
    ```
    -isolation_supply <supply set name>
    ```
  - if not specified, uses default_isolation supply of location
    - can be a single-rail cell if containing domain is always on when enabled
    - otherwise typically requires a dual-rail cell

Strategy Execution Order



Retention, then **Repeater**, then **Isolation**, then **Level Shifter**

Strategy Interactions

	Retention	Repeater	Isolation	Level Shifter
Retention	--	affects	affects	affects
Repeater	affected by	--	affects	affects
Isolation	affected by	affected by	--	affects
Level Shifter	affected by	affected by	affected by	--

Strategies may change driver and/or receiver supplies of a port

This may affect `-source/-sink` filters of subsequently executed strategies

Supply Ports/Nets

- **Represent supply ports, pins, and rails**
  - Primary supply inputs, supply pins of cells, nets in between
- **Are connected together to create supply network**
  - Together with power switches to control power distribution
- **Deliver power/ground/etc. supplies to domains**
  - Delivered values determine how domain functions
- **Have and propagate {state, voltage} values**
  - States are UNDETERMINED, OFF, PARTIAL_ON, FULL_ON
  - Voltages are fixed-point values with microvolt precision
- **Examples**
  - {FULL_ON 1.2} {PARTIAL_ON 0.81} {OFF}

"Power" (Supply) Switches

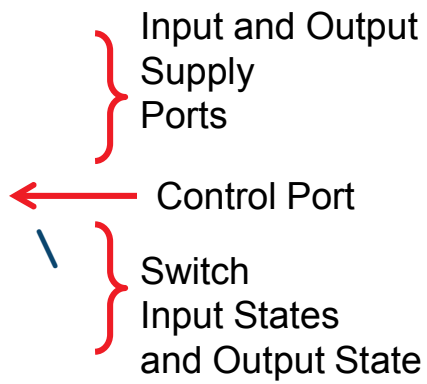
- **Have one or more supply inputs**
  - Defined with `-input_supply_port`
- **Have one supply output**
  - Defined with `-output_supply_port`
- **Have one or more control inputs**
  - Defined with `-control_port`
- **Have one or more control states**
  - Defined with `-on_state` or `-on_partial_state`
  - Also can include `-error_state` and/or `-off_state`
- **Conditionally propagate input supply values to output**
  - Based on which control states are active

Power Switches

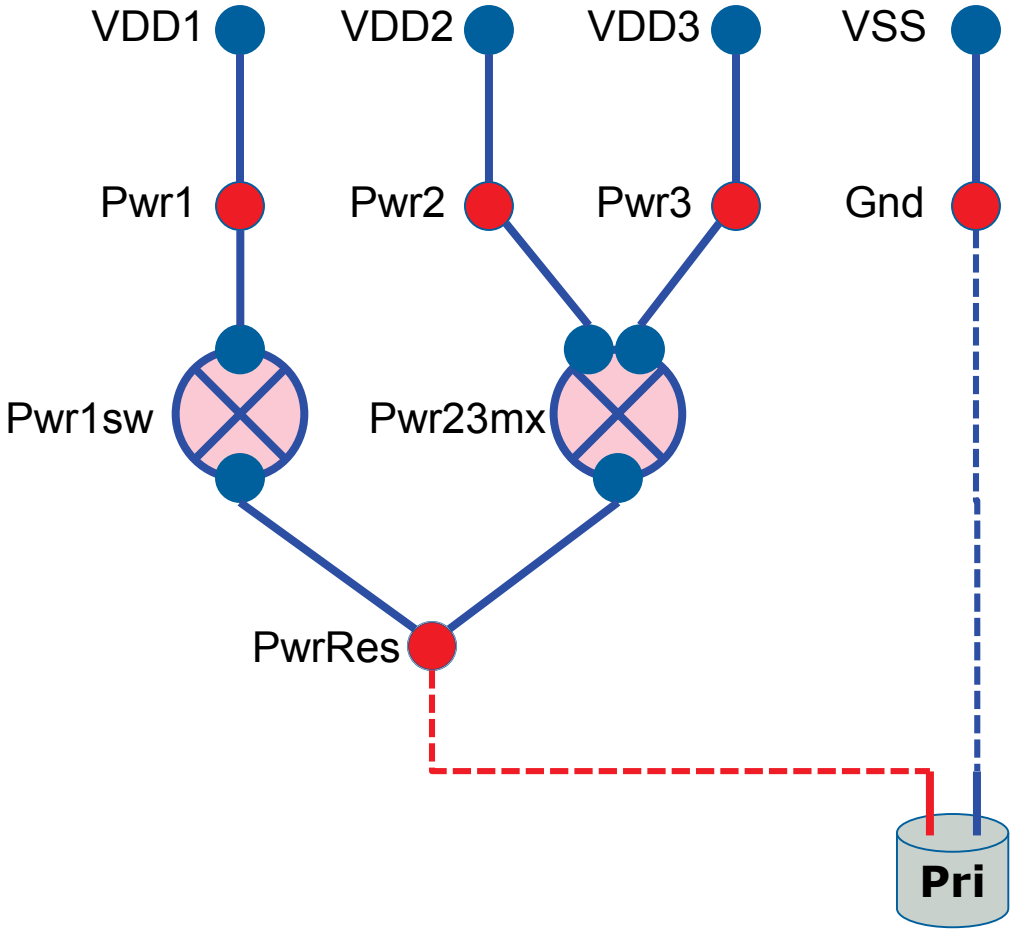
- **Examples**

```
create_power_switch Simple \
  -output_supply_port {vout} \
  -input_supply_port {vin} \
  -control_port {ss_ctrl} \
  -on_state {ss_on vin { ss_ctrl }} \
  -off_state {ss_off { ! ss_ctrl }}

create_power_switch TwoStage \
  -output_supply_port {vout} \
  -input_supply_port {vin} \
  -control_port {trickle_ctrl} \
  -control_port {main_ctrl} \
  -on_partial_state {ts_ton vin { trickle_ctrl }} \
  -on _state {ts_mon vin { main_ctrl }} \
  -off_state {ts_off { ! trickle_ctrl && ! main_ctrl }}
```



Supply Network Construction



Commands

```
create_supply_port …
create_supply_net …
connect_supply_net …

create_power_switch …
connect_supply_net …

create_supply_net \
  -resolved …
connect_supply_net …

create_supply_set \
  -update
```

Supply Equivalence

- **Supply Ports/Nets/Functions**
  - **Electrically equivalent** if same/connected/associated
  - **Functionally equivalent** if
    - they are electrically equivalent, or
    - they are declared functionally equivalent
      - example: outputs of two switches that have same input and control
- **Supply Sets**
  - **Functionally equivalent** if
    - both have the same required functions, and corresponding required functions are electrically equivalent; or
    - both are associated with the same supply set; or
    - they are declared functionally equivalent
      - Declaration works for verification only; must be explicitly connected for implementation

A Deeper Look at UPF Power Intent

- **Logic Hierarchy**
- **Power Domains**
- **Power Domain Supplies**
- **Supply Sets**
- **Supply Connections**
- **Power Related Attributes**
- **Power States and Transitions**
- **Power Domain State Retention**
- **Power Domain Interface Management**
- **Supply Network Construction**
- **Supply Equivalence**

For more details, read the IEEE 1801-2013 UPF spec, especially Clause 4, UPF Concepts