ADDRESSING THE CHALLENGES OF GENERICALLY SPECIFYING POWER INTENT WITH MULTI-RAIL MACROS

Brandon Skaggs
Cypress Semiconductor

Problem Statement

The Unified Power Format (UPF) language is well suited for describing power distribution networks for standard cells such that connections can be made automatically; however, **cell macros with multiple power rails often require explicit, manual connections that are often technology-dependent--reducing the portability of the design**.

Methods for abstracting these connections are discussed, including recommended enhancements to the *Liberty* Library specification and IEEE 1801 UPF standards to better support such cells.

Currently-Defined Liberty Power Attributes

Table 1: Cell Type & Function

Function	Attribute	Value	Purpose
Retention	retention_cell	retention_cell_style (string)	Cell has retention capability. Allows use of more than one type of retention element.
	retention_pin	save/restore/ save_restore	Maps a cell pin to the save, restore, or both save & restore actions
	save_action/restore_action	L(low)/H(high)/ R(rise)/F(fall)	Indicates the trigger events assoc. with save/restore
Level-Shifting	is_level_shifter	true/false	Cell is a level shifter
	level_shifter_type	LH/LH/HL_LH	Specifies cell shifts low-high, high_low, or both (default)
	input_voltage_range	floating value	Allowed range of input pin
	output_voltage_range	floating value	Allowed range of output pin
Isolation	is_isolation_cell	true/false	Cell is an isolation cell
	isolation_cell_data_pin	true/false	Input data pin
	isolation_cell_enable_pin	true/false	Isolation control
Macro	is_macro_cell	true/false	Defines macro cells
	is_isolated	true/false	Specifies that a macro pin does not require external isolation
	isolation_enable_condition	Boolean expr	Enable isolation of macro
	is_analog	true/false	Specifies an analog pin

Table 2: Power-related Pin Attributes

Attribute	Value	Purpose
related_power_pin	pg_pin_name	Associates a logic pin to its related supply
related_ground_pin	pg_pin_name	Associates a logic pin to its related ground
pg_type	primary_power	Specifies that pg_pin is a primary power source (default)
pg_type	primary_ground	Specifies that pg_pin is a primary ground source
pg_type	backup_power	Specifies that pg_pin is a backup (secondary) power source (for retention register, always-on logic, etc.)
pg_type	backup_ground	Specifies that pg_pin is a backup (secondary) ground source (for retention register, always-on logic, etc.)
pg_type	internal_power	Specifies that pg_pin is an internal power source for switch cells
pg_type	internal_ground	Specifies that pg_pin is an internal ground source for switch cells
pg_type	nwell	Specifies regular n-wells for substrate-bias modeling
pg_type	pwell	Specifies regular p-wells for substrate-bias modeling
pg_type	deepnwell	Specifies isolation n-wells for substrate-bias modeling
pg_type	deeppwell	Specifies isolation p-wells for substrate-bias modeling
std_cell_main_rail	True	Set on primary power pin to identify main cell power

Multi-Rail Macros

Cells containing multiple primary or bias connections, however, require special attention; multiple *primary_power* or *primary_ground* connections can exist. The supply pin ***pg_type* attribute information is not enough** in itself for tools to conclusively make appropriate connections.

Table 3: SRAM types with Supply Pins, *pg_type*

Type	Supply Pin	Pin pg_type	Function
Vendor A	VDDRET	backup_power	Retention power
	VDDB	primary_power	nwell bias (always on)
	VDD	primary_power	Periphery power (switchable)
	VSS	primary_ground	Common ground
Vendor B	VDDCE	backup_power	Core power (always on)
	VDDPE	primary_power	I/O power (switched)
	VSSE	primary_ground	Common ground
	BIASCNW	backup_power	Channel nwell bias (always on)
	BIASNW	backup_power	nwell bias / switched off
	BIASPW	backup_ground	pwell bias

The inability to make definitive, attribute-based assignment of power pins leads to the verbose and tedious practice of explicitly specifying supply connections directly with UPF syntax (Figure 1). These connections are **technology-dependent**—since the number and names of supply pins can vary across technologies or vendor implementations.

Figure 1: Manual connections to SRAM supply ports

```
// Vendor A supply connections
connect_supply_net vcc_arr_out -ports {sram/VDDRET}
connect_supply_net vccdpslp    -ports {sram/VDDB}
connect_supply_net vcc_per_out -ports {sram/VDD}
connect_supply_net vssd        -ports {sram/VSS}

// Vendor B supply connections
connect_supply_net vcc_arr_out -ports {sram/VDDCE}
connect_supply_net vcc_per_out -ports {sram/VDDPE}
connect_supply_net vssd        -ports {sram/VSSE}
connect_supply_net vccdpslp    -ports {sram/BIASCNW}
connect_supply_net vcc_per_out -ports {sram/BIASNW}
connect_supply_net vssd        -ports {sram/BIASPW}
```

Current Approaches to the Problem

A. *Querying for Supply Pin Availability*

UPF 2.0 and newer UPF specifications provide a *find_objects* command; this allows for the querying of design objects (signals, instance names, ports, etc.) at a specified level of hierarchy matching an arbitrary regular expression. **Conditional connection of supply pins can be achieved based on their existence**:

Figure 2: Query and Conditionally Connect Supply Nets

```
set vddce_port [find_objects my_ip/sram -object_type port \
                        -pattern VDDC*]
if {[llength $vddce_port] ne 0} {
   connect_supply_net   vcc_arr_out -ports "$vddce_port"
}
```

Rules for the connection of all possible supply nets within the set of supported technologies could be provided; however, new technologies must be manually added, and care must be taken to handle connections where technology pin names overlap but conflict.

Current Approaches (cont.)

B. *Sourcing Tech-specific Connections via Hierarchical Flow*

Another option would be the direct **sourcing of an implementation-specific UPF by a reference UPF**; rather than making the technology-specific connections directly within a UPF file, the technology-specific connections would be referenced from a technology-specific UPF—that would be selected based on environment/configuration settings:

Figure 3: Including Tech-Specific Content via Hierarchical Reference

```
set_scope my_ip
load_upf ${DESIGN}/ip/rtl/ip_top/${TECH}/ip_top.upf
set_scope .
```

Note that **the selection of the technology-specific file is still required**; there must be knowledge somewhere in the system of what the implementation is. However, as shown above, a solution where the technology-specific file path is determined from configuration variables can be envisioned.

C. *IP-based UPF Generators*

One alternative to providing a library of static UPF files for supported technologies is to **dynamically generate technology-specific UPF files with automation scripts** capable of distinguishing among the supported technologies and outputting power-intent in line with the selected implementation. While functionally no different than the approach of sourcing technology-specific UPF files within a hierarchical flow (described above in Section B), maintenance can be more manageable, as the power-intent code that is common across technologies need only be updated in one 'generator' script.

Proposed Liberty Standard Attribute Additions

The difficulty in mapping the small number of *pg_type* possibilities listed in Table 2 to multi-rail macro power connections can be reduced simply by providing **more options for *pg_type*** to take on—as well as a well-defined method for mapping these to new or existing UPF concepts.

Table 4: Proposed New Liberty Attributes for Multi-Rail Macros

Attribute Value	Function	UPF Automatic Assignment
macro_io_primary_power	I/O padring power	PD.primary.power
macro_io_backup_power	I/O padring power (dual rail cell)	PD.default_retention.power
macro_io_ground	I/O padring ground	PD.primary.ground
macro_core_primary_power	Core macro power	PD.primary.power
macro_core_backup_power	Core macro power (dual rail cell)	PD.default_retention.power
macro_core_ground	Core macro ground	PD.primary.ground
macro_isolation_primary_power	Isolation supply	PD.primary.power
macro_isolation_backup_power	Isolation supply (dual rail cell)	PD.default_isolation.power
macro_isolation_ground	Isolation ground	PD.default_isolation.ground
macro_retention_primary_power	Retention supply	PD.primary.power
macro_retention_backup_power	Retention supply (dual rail cell)	PD.default_retention.power
macro_retention_ground	Retention ground	PD.default_retention.ground
macro_reference_primary_power	Analog reference supply	PD.primary.power
macro_reference_backup_power	Analog reference supply (dual rail cell)	PD.default_isolation.power
macro_reference_ground	Analog reference ground	PD.default_isolation.ground

Proposed UPF Standard Enhancements

There are two paths for improving multi-rail macro connections via IEEE 1801 / UPF language enhancements: (1) improving the likelihood of automatic connections based on *Liberty pg_type* attributes and (2) providing more flexible methods for mapping supply sets to *Liberty pg_types*.

(1) *Create a 'default_backup' Supply Set Handle*
The *create_power_domain* command already provides a method for mapping a supply set to an arbitrary *supply set handle*, and earlier revisions pre-defined handles named *default_isolation* and *default_retention* for providing supply sets for isolation- and retention-related cells within a domain. However, these supply sets were only mapped to cells defined to be isolation or retention cells that were inserted as part of an isolation or retention strategy.

A more flexible option would be to **provide a default supply set handle named *default_backup***; this supply set would be automatically connected to the *pg_pin*s of cell macros within the domain whose *pg_type* are *backup_power* and *backup_ground*—**without the requirement that the cell be of a specific retention or isolation cell type**. While this simple mapping would not be sufficient for many multi-rail macros, the capability could allow simple macros to have their power mapping performed automatically.

(2) Extend 'create_power_domain -define_func_type'
create_power_domain also provides a method for mapping a domain's *primary* supply set handle's functions (i.e., *power, ground, nwell, etc.*) to a specific *pg_pin_type* attribute (*primary_power, primary_ground, etc.*). However, this option is only available for the *primary* supply set for a domain.

What is needed is the capability to **map any existing supply set handle's functions to any arbitrary *pg_pin_type* attribute**—including the *user_pg_type* attribute that can also be applied to a *pg_pin*. In fact, support for *user_pg_type* in this way could allow companies to assign attributes based on internal naming conventions—standardizing the interface to vendor-supplied IP in such a way as to allow their UPF power descriptions to then be vendor agnostic.

Example Usage

Table 5: SRAMs types with New *pg_type* Attributes

Type	Supply Pin	Pin pg_type	Function
Vendor A	VDDRET	macro_core_backup_power	Retention power
	VDDB	macro_retention_backup_power	nwell bias (always on)
	VDD	macro_io_primary_power	Periphery power (switchable)
	VSS	macro_core_ground	Common ground
Vendor B	VDDCE	macro_core_backup_power	Core power (always on)
	VDDPE	macro_io_primary_power	I/O power (switched)
	VSSE	macro_core_ground	Common ground
	BIASCNW	macro_reference_backup_power	Channel nwell bias (AON)
	BIASNW	macro_reference_primary_power	nwell bias / switched off
	BIASPW	macro_reference_ground	pwell bias

Figure 4: UPF Mapping new *pg_type*s to Supply Sets

```
create_supply_set SRAM_SS -function {power vcc_arr_out} \
   -function {ground vccdpslp} \
   -function {nwell vcc_per_out}\
   -function {pwell vssd}
create_power_domain SRAM_PD -supply {primary SRAM_SS} \
   -define_func_type {                                 \
      {SRAM_SS.power  macro_core_backup_power}         \
      {SRAM_SS.ground macro_reference_backup_power}    \
      {SRAM_SS.nwell  macro_io_primary_power}          \
      {SRAM_SS.pwell} macro_core_ground}}
```