In post-synthesis, gate-level netlist (GL-netlist), power aware (PA) simulation, the fundamental focus is to identify PA specific cells already present in the netlist. The associated UPF with the netlist design, determines the supply network and power connectivity to these special PA cells, and aid to keep their outputs from being corrupted. Hence, the GL-netlist-based power aware simulation (PA-SIM) input requirements are mostly the same as for RTL simulation. However, the design-under-verification at this stage is the GL-netlist from synthesis, so logic gates from standard, multi-voltage (MV), and Macro cell Liberty libraries are already inserted or instantiated in the design. Hence PA-SIM at post-synthesis also requires Liberty libraries as input in order to accumulate different cell-level attributes and power down functions. The Questa® PA-SIM tool utilizes the libraries to extract different attributes and functionalities which can be summarized as follows.

One significant aspect of GL-netlist PA-SIM is that all cell instances are interpreted as containing drivers because these cells are usually leaf level cells or they are an instance that has no descendants. As a result, buffer cell instances in a GL-netlist will cause corruption when powered down. This contrasts with RTL Verilog ‘buf’ primitives that do not represent drivers and, therefore, do not cause corruption when powered down, but they do not isolate and may propagate corrupted signals from upstream drivers.

During GL-netlist PA-SIM, corruption will occur on the output ports and sequential logic of any detected gate-level cells. In addition, power aware simulation automatically treats a module as a gate-level cell if the module contains the `celldefine attribute or the `specify blocks in HDL code. Even these cells are not defined in the Liberty syntax; the standard processing of driver-based corruption is still applied to these cells, similar to that for RTL cell designs.

UPF 1801-2013 or UPF 2.1 LRM provides a dominant mechanism to devise a driver-based corruption of any HDL cell—even when there is no `celldefine or `specify block—through the set_design_attribute [-attribute {name value}] command. PA-SIM treats all module, entity, or design elements as a gate-level or leaf cell when the following syntax is applied on them.

Example 1.1 Leaf-level or Gate-level Cell Treatment of Design for Driver-Based Corruption

Define through UPF File:
```
set_design_attributes -models FIFO
-attribute (UPF_is_leaf_cell TRUE)
```

Define through HDL Annotation:

SystemVerilog or Verilog Attribute Specification:
```
(* UPF_is_leaf_cell="TRUE" *)
module FIFO (<port list>);
```

VHDL Attribute Specification:
```
attribute UPF_is_leaf_cell : STD.Standard.String;
attribute UPF_is_leaf_cell of FIFO : entity is "TRUE";
```

Though the latest UPF 1801-2015 or UPF 3.0 LRM revised the syntax for leaf-level and gate-level cell
definition to enable driver-based corruption through `UPF_is_hard_macro` instead of `UPF_is_leaf_cell` attributes, the semantics and use model remain identical.

During GL-netlist PA-SIM, apart from detecting the standard and Macro cells and applying corruption accordingly, the simulator is also required to automatically identify special power management or MV cells—such as, isolation (ISO), level-shifter (LS), and retention flops (RFF)—in the design. The detection of MV cells is primarily done through the cell-level attributes available in the corresponding Liberty libraries and are usually cross-compared with the corresponding definition of strategies in UPF. Recalling the syntax and examples of ISO, LS, and RFF from UPF LRM, and also the fact that a GL-netlist contains at least ISO, LS, and RFF cells through synthesis, most of these cells are already specified either through following UPF commands and options or through tool auto detection processes.

**Example 1.2 ISO, LS, RFF Automatic Detection in GL-netlist Simulation through UPF Command**

ISO cells:
```plaintext
set_isolation strategy_name
   [-instance [[instance_name port_name]*]]
```

Where the `<instance_name>` is a technology leaf-cell instance and the `<port_name>` of the logic port that it isolates.

LS cells:
```plaintext
set_level_shifter strategy_name
   [-instance [[instance_name port_name]*]]
```

Similarly where the `<instance_name>` is a technology library leaf-cell instance and the `<port_name>` of the logic port that it level-shifts.

RFF cells:
```plaintext
set_retention retention_name
   [-instance [[instance_name][signal_name]*]]
```

Here in this case as well the `<instance_name>` is a technology library leaf-cell instance and the optional `<signal_name>` is the HDL signal that controls retention. If this instance has any unconnected supply ports or save and restore control ports, then these ports need to have identifying attributes in the cell model, and the ports shall be connected in accordance with this `set_retention` command.

In GL-netlist PA-SIM, the tool’s auto detection process of MV cells actually refers to the cells that are not specified through the `-instance` but through Liberty or other attributes. Hence, for the rest of the cells that are not specified in the UPF file, GL-netlist PA-SIM automatically detects the right UPF strategy to which they belong and treats them in a similar way to cells of that strategy specified with an `-instance` argument. Questa® PA-SIM detects power management cells based on one of the following.

**List 1.2 Liberty Cell-Level Attributes**
- `is_isolation_cell`
- `is_level_shifter`
- `retention_cell`

**List 1.3 Library Cell Name from UPF Commands**
- `map_isolation_cell` isolation_name
  ```plaintext
  [-lib_cells lib_cells_list]
  ```
- `map_level_shifter_cell` level_shifter_strategy
  ```plaintext
  [-lib_cells list]
  ```
- `map_retention_cell` retention_name_list
  ```plaintext
  [-lib_cells lib_cells_list]
  ```
- `use_interface_cell-cell-strategy`
  ```plaintext
  list_of_isolation_level_shifter_strategies
  [-lib_cells lib_cells_list]
  ```

Note that `map_isolation_cell` and `map_level_shifter_cell` are deprecated from UPF LRM 3.0 with `use_interface_cell` command. Unlike `map_isolation_cell` and `map_level_shifter_cell`, the `use_interface_cell` can be used to manually map any ISO, LS, or combined isolation level-shifter (ELS) cells.

**List 1.4 UPF name_format Command for Defining Names of Implicit Objects**
- `[-isolation_prefix string]`
- `[-isolation_suffix string]`
- `[-level_shift_prefix string]`
- `[-level_shift_suffix string]`

**List 1.5 Synthesis Pragmas**
- `isolation_upf`
- `retention_upf`
Although the GL-netlist PA-SIM does not have any exceptions from the fundamental concept of PA-SIM at the RTL, the Questa® PA-SIM tool procedure requires additional commands to process the information discussed above (Lists 1.2 and 1.3).

**Tool Procedures for Liberty Processing at GL-netlist:**

Compile: No change  
Optimize: `vopt` requires including either  
"vopt-pa_libertyfiles" or  
"vopt-pa_dumplibertydb"  
Simulate: No change

The following list explains the Liberty library referencing methods for GL-netlist PA-SIM.

**List 1.6 Liberty Referencing in PA-SIM at GL-netlist**

- `pa_libertyfiles`- Specifies the Liberty files to read. It is also possible to specify multiple files by separating file names with a comma.  
  e.g., `vopt -pa_libertyfiles=a.lib,b.lib`  
- `pa_dumplibertydb`- Specifies the name of the Liberty attribute library database for future reference.  
  e.g., `vopt -pa_dumplibertydb=lib_datafile`

Apart from detecting standard, Macro, and MV cells, PA-SIM also is required to virtually infer missing MV cells in the design. In general the virtual inferring process is limited to RTL where physical MV cells are not instantiated yet. Inferring may also be required at the Mixed-RTL, where some of the MV cells are still missing. Hence during GL-netlist, such virtual inferring is redundant. However, PA-SIM provides user controllability through tool procedures where it is possible to control the inference.

**Tool Procedures for Controlling Virtual Inferring of MV Cells:**

Compile: No change  
Optimize: `vopt` requires adding one of the following to disable auto inference:  
"vopt -pa_disable=insertiso" - Disable ISO cell insertion  
"vopt -pa_disable=insertls" - Disable LS cell insertion  
"vopt -pa_disable=insertret" - Disable RFF cell insertion  
Simulate: No change

Using one of the above procedures, based on the requirements, will allow the tool not to infer the appropriate cells virtually at any design abstraction level. But since physical MV cells are already inserted in post-synthesis GL-netlist designs, hence using a tool procedure during optimization as follows will instruct the tool to disable all three virtual insertions for ISO, LS, and RFF at once.

**Required Tool Procedure for GL-netlist PA-SIM:**

Optimize: `vopt` requires to include “vopt -pa_gls”

Hence the GL-netlist PA-SIM as well as Mixed-RTL mechanisms can be summarized as follows:

**List 1.7 Summarization of PA-SIM Mechanism for GL-netlist and Mixed-RTL Design**

- Detect standard and Macro cells in the netlist and perform corruption based on the driver, UPF strategies, or power down functionalities from Liberty.  
- Detect MV cells in the netlist and match them with corresponding UPF strategies at that point in the design.  
- Virtually infer MV cells, if missing in the netlist, based on UPF strategies.  
- Conduct automated power aware sequence checks and testbench-based simulation similar to RTL PA-SIM.

So once the cell detection or inferring process is completed as discussed above, the tool conducts power aware simulation on the GL-netlist similar to the RTL design. Although the Liberty file is required as additional input in the GL-netlist, however it is recommended to use the same testbench from RTL stage to confirm verification consistency.
VERIFICATION ACADEMY
The Most Comprehensive Resource for Verification Training

32 Video Courses Available Covering

- UVM Debug
- Portable Stimulus Basics
- SystemVerilog OOP
- Formal Verification
- Intelligent Testbench Automation
- Metrics in SoC Verification
- Verification Planning
- Introductory, Basic, and Advanced UVM
- Assertion-Based Verification
- FPGA Verification
- Testbench Acceleration
- PowerAware Verification
- Analog Mixed-Signal Verification

UVM and Coverage Online Methodology Cookbooks
Discussion Forum with more than 8250 topics
Verification Patterns Library

www.verificationacademy.com