REUSABLE UPF: Transitioning from RTL to Gate Level Verification

Durgesh Prasad (durgesh_prasad@mentor.com), Mentor, A Siemens Business
Jitesh Bansal (jitesh_bansal@mentor.com), Mentor, A Siemens Business
Madhur Bhargava (madhur_bhargava@mentor.com), Mentor, A Siemens Business

***Abstract*- United Power Format (UPF) is used in a design to specify the power intent of the design. The UPF file is used at all the stages in the verification starting from RTL verification to GLS to place & route stages however it is often the case that UPF needs to be modified at next verification stage. For example UPF at RTL level needs to be modified to be used at GLS level due to design-hierarchy changes, cell placements and cell connections. This creates problem of managing different UPFs at various stages, checking their equivalence to make sure the consistency. In this paper we want to highlight all the differences between an RTL UPF and a GLS UPF. We would also propose a methodology to write RTL UPF in such a way that minimal changes are required during gate level power verification.**

Keywords: UPF, netlist, RTL, Place & Route, Liberty, Implementation tool, State elements, ELS cells.

I. Introduction

Today's chips require a sophisticated verification methodology. The normal trend is to verify the chip at multiple level starting from verification at RTL then to Gate Level and further down to place & route netlist. RTL is the most abstract level of describing the design and provides very fast verification turnaround but at the same time it is not able to catch all the issues so designers tend to move to Gate level simulation. Gate level simulation overcomes the limitations of static-timing analysis and is increasingly being used because of complex timing checks at 40nm and below, design for test (DFT) insertion at gate level and low power considerations. Other reasons for running gate level simulation are reset verification, X optimism in RTL and basic heartbeat tests. The next step in chip design cycle is place & route in which the tool extract the components and nets from the netlist, place the components on the target device, and interconnect the components using the specified interconnections. After the place and route verification process is complete, the designer has an implementation of the design in the target technology.

Now a day's one of the very important aspect of verification is the low-power. Earlier power decisions were only taken at the place & route stage which was very late and any bug found in the design at that stage would take huge time to fix. To overcome this problem designer started applying power aspect very early in the design cycle starting at RTL itself. UPF emerged as the most accepted format to represent power intent of the design. Power Intent is specified in the UPF file and it is then applied at every stage of design cycle, starting at RTL level, then at gate-level and finally at place & route level too. However one of the main concern in application of power-intent is that UPF needs to be refined/modified up to some extent at every stage to keep it compatible with the netlist. This creates different flavors of UPF which lacks consistency and creates maintenance problem.

The motivation behind this paper is to explain the challenges why UPF needs to be changed when designer shifts from RTL to gate level verification and how we can write a better UPF at RTL itself so that it requires minimal or no changes when it goes to Gate Level netlist. To understand these reasons, first we need to understand the relevant basics of UPF.

A. UPF Basics

The UPF specification, *IEEE Std. 1801™-2013 Unified Power Format (UPF)*, allows designers to specify the power intent of the design. It is based on Tcl and provides concepts and commands that are necessary to describe the power management requirements for IPs or complete SoCs. Power Intent Specification in UPF is used throughout the design flow; however, it may get refined at various steps in the design cycle.

The following are some of the important concepts and terminologies used in the power intent specification:

- **Power domain** — A collection of HDL module instances and/or library cells that are treated as a group for power management purposes. The instances of a power domain typically, but do not always, share a primary supply set and are typically all in the same power state at a given time. This group of instances is referred to as the extent of a power domain. Following UPF command can be used to create a power domain on element cpu[0]/inst.

```
create_power_domain PD –elements {cpu[0]/inst}
```

- **Isolation cell** — An instance that passes logic values during normal mode of operation and clamps its output to some specified logic values when a control signal is asserted. It is required when the driving logic supply is switched off while the receiving logic supply is still on. Following UPF commands can be used to create isolation cell on the port "in1" of the domain PD.

```
set_isolation ISO –domain PD –elements {cpu[0]/inst/in1}–clamp_value 1\
            –isolation_supply_set AON.supply –isolation_enable {iso_en}
```

- **map_isolation_cell** — This command is used to specify the custom isolation cell for the signals of the corresponding set_isolation strategy. Although this command has been deprecated in UPF 2.1 and replaced by the **use_interface_cell** command, the basic functioning remains the same.

```
map_isolation_cell ISO_1 –domain PD –lib_model_name {iso_cell} \
                         –port "ISO iso_en" –ports "in IN1"
```

- **Level shifter** — An instance that translates signal values from an input voltage swing to a different output voltage swing. Following UPF command can be used to create level shifter cells on the output of a domain PD.

```
set_level_shifter LS –domain PD –applies_to output –input_supply_set
IN_ss –output_supply_set OUT_SS –internal_supply_set INT_SS
```

- **Retention** — Enhanced functionality associated with selected sequential elements or a memory such that memory values can be preserved during the power-down state of the primary supplies. Following UPF commands can be used to impart sequential behavior to specified sequential elements.

```
set_retention RET –domain PD –elements { cpu[0]/inst/qffr} \
–save_signal {save posedge} –restore_signal {restore negedge}
```

- **connect_supply_net** — This command is used to connect the supply nets (pg-pin) to various ports e.g pa-cell ports.

```
connect_supply_net vdd –ports{in1_UPF_ISO/vdd_sub}
```

- **bind_checker** — This command is used to bind a user's custom assertions, coverage models, or debug models to various UPF objects in their design, such as isolation cell, retention cell, supply sets, and power domain.

  A sample code snippet:

```
bind_checker checker_instance_name \
      -module checker_module_name \
      -bind_to target_instance \
      -ports {{formal_port1_name power_object_handle} \
      {formal_port2_name power_control_signal}}
```

  In the sample code, the `bind_checker` UPF command instantiates the checker module, *checker_module_name*, into the design hierarchy, *target_instance*, with the instance name,

checker_instance_name, without actually modifying the design code or introducing any functional changes. The *-ports* option maps actual ports to formal ports.

- **find_object -** The find_objects command searches for instances, nets, ports, supply ports or processes that are defined in the design hierarchy. For example following command can be used to search an instance containing keyword ff in hierarchy "cpu[0]/inst"

```
find_objects cpu[0]/inst –pattern "*ff*" –object_type inst \
     –transitive true
```

- **query_commands** — The UPF 2.0 and 2.1 standards provide toolset of query commands (for example, `query_power_domain`, `query_isolation`, and `query_retention`), which can be used to search and get the handle of power management objects, including strategies (isolation/retention/power switch), power domains, supply nets, and supply ports.

  A sample code snippet:

```
# Get handle of isolation strategy 'PD_ISO1' defined in domain 'PD'
query_isolation PD_ISO1 -domain PD
```

  The return value of the `query_isolation` command can be used to get isolation strategy details such as isolation enable signal, its elements, isolation power, isolated source domain power etc.

- **save_upf -** This UPF command provides a way to write back the tool interpretation of UPF in a legitimate UPF format in the file specified by the command**.**

```
For example:
set_isolation ISO –domain PD –source PD1.primary
save_upf saved.upf

saved.upf contents :
set_isolation ISO –domain PD \
 –elements { p1 p2 …. (all ports with source PD1.primary) }
```

B. UPF verification at RTL vs GL netlist

UPF based verification at RTL level consists of creating power domain, inserting power aware cells like isolation, level-shifter and retention cell and finally defining a supply network to propagate power. Please note that the power aware cells placed in the design by application of UPF at the RTL stage are mostly simulation models and they work as a placeholders for actual cell which would come later during gate level. RTL power-aware verification ensures that power-aware cells are placed at all the required ports/state elements and power distribution network is valid.

In Gate Level Netlist power-aware cells are already part of the design and UPF only complements them. There are two flavors of gate level netlist

1. Pg-connected netlist: The pg-pins of the power aware cells are present in the cell and they are connected in the netlist itself. The use of UPF is to validate that pa-cells are placed at all possible location as specified in RTL UPF and supply network is properly embedded in netlist as per UPF.

2. Non-pg netlist: The pg-pins of the cell are not connected via HDL hierarchy rather it is expected from simulation tools to connect the pg-pin. At this stage they provide liberty information about the cells. This helps verification tool to do automatic connection of these cells and handle their corruption. Sometimes these connections are provided explicitly in the Gate Level UPF via

`connect_supply_net` UPF commands. In contrast to RTL now the netlist has actual implementation model for simulation.

Designers use various methodology to perform Gate level power aware simulation. One of the way is to spit out a new UPF for Gate Level netlist after low-power RTL verification but this method is not very reliable because Gate Level UPF might not be equivalent to RTL UPF due to synthesis tool issues and UPF interpretation. Also designer need to do equivalence checking of the two UPF.

Another way some Implementation tool propose is to have the Golden RTL UPF and generate additional information (in terms of configuration file or new UPF file) for gate level simulation [4]. This approach minimizes the synthesis tool's bug risk but still doesn't give designer much confidence. The most sought after approach is to re-use the RTL UPF for gate level simulation too.

II. BENEFIT OF REUSABLE UPF

There are lots of benefits of re-using the UPF written for RTL simulation at the gate level as well. The important benefits are as follows:

- Concise and easy: RTL UPF is very concise and simple to read. The main reason of writing design in RTL itself is being concise as compared to its Gate Level Netlist. Another reason is to insert power-aware cells user doesn't need to instantiate them one by one, users can simply write a single policy to instantiate multiple cells. Users can put his comments in the UPF file for better readability and maintenance. All of these benefits are lost if UPF is generated by tool.
- Tool specific changes are preserved: User might have done tool specific changes for various vendors. This is required because UPF is still evolving and hence in spite of IEEE standard the interpretation of various UPF commands/concept differ across vendors. Also user might have done tool specific changes because he want to use same UPF for structural checking tools as well as simulation tools.
- Requirement of logical equivalence is eliminated: If user is dealing with a single UPF at gate level as well as RTL level then there is no requirement of doing logical equivalence for UPF but if this is not the case then he needs to do this equivalence check to ascertain that his gate level UPF is actually the same implementation of his RTL UPF.
- Coverage of UPF remains intact: Now a days designers do the coverage of UPF objects and save it to achieve coverage closure. If this complete UPF is going to change during Gate Level Netlist then he has to again repeat this step on new UPF and merge the results back to achieve coverage closure.

III. RELEVANT DIFFERENCE in Power Aware RTL and GLS Netlist

A. Language style

RTL and GLS are very two different type of netlist. RTL is more abstract in nature and consist mainly of always and assign blocks. GLS is closer to silicon and consist mainly of cells and gates. Though logical intent is same in both netlist but language style is different. In GLS, there are no generates or vectors, everything is flattened out.

RTL	GLS

`module dut(input [1:0]in1,output [1:0] out1);`
`genvar i;`
`generate`
`for(i = 0; i < = 1; i = i + 1) begin : gen`
`mid mid(in1[i],out1[i]);`
`end`
`end generate`
`endmodule`
	`module dut(input \in1[1],\in1[0] ,`
	`output \out1[1] ,\out1[0]);`
	
	`mid \gen[0].mid (\in1[0] , \out1[0] );`
	`mid \gen[1].mid (\in1[1] , \out1[1] );`
	
	`endmodule`

B. Low power cells

In RTL low power simulation logical intent is specified by RTL netlist and power intent is specified by UPF. Simulation tools do the low-power verification by superimposing power intent specified by UPF file on the logical intent of RTL netlist. All the power aware activities - corruption, isolation, retention are done using the virtual low-power cells. RTL netlist is not modified in any way. It enables users to re-use the same logical design with different power intent. In the synthesis phase, the UPF file is also passed along with the RTL netlist. The synthesis tool infers the low-power behavior specified in the UPF and adds the following low-power cells to the design:

- Isolation cells to signals crossing power domains
- Level shifters to signals crossing voltage domains
- Replacement of all flops with retention flops where specified

GLS netlist not only have logical content but also some power intent. In GLS low power verification, there is no need to add virtual cells for power activities. They are already present in design and just need power up and down behavior for low power verification.

C. Liberty information

GLS netlist consist of cells which has their own power/timing specifications. These specifications are provided by liberty files and they should be followed thoroughly for correct behavior of cells. For low power GLS verification these liberty specifications along with UPF are superimposed on GLS netlist.

Sample liberty file –

```
cell (ISOHID1BWP) {
  area : 1.0584;
  cell_footprint : "isohid1";
  is_isolation_cell : true;

  pg_pin (VDD) {
    pg_type : primary_power;
    voltage_name : COREVDD1;
  }

  pg_pin (VSS) {
    pg_type : primary_ground;
    voltage_name : COREGND1;
  }
  pin(I) {
    direction : input;
    isolation_cell_data_pin : true;
    related_ground_pin : VSS;
    related_power_pin : VDD;
    capacitance : 0.0005546;
  }
  pin(ISO) {
    direction : input;
    isolation_cell_enable_pin : true;
    related_ground_pin : VSS;
    related_power_pin : VDD;
    capacitance : 0.0005649;
  }
  pin(Z) {
    direction : output;
    power_down_function : "!VDD + VSS";
    function : "(I+ISO)";
    related_ground_pin : VSS;
```

```
      related_power_pin : VDD;
      max_capacitance : 0.03277;
    }
  }
```

Relevant Power Information -

- ***pg_pin:*** Use the pg_pin group to specify power and ground pins. The library cells can have multiple pg_pin groups. A pg_pin group is mandatory for each cell. A cell must have at least one primary_power pin specified in the pg_type attribute and at least one primary_ground pin specified in the pg_type attribute.

- ***pg_type:*** Use the optional pg_type attribute to specify the type of power and ground pin. The valid values are primary_power, primary_ground, backup_power, backup_ground, internal_power, and internal_ground.

- ***related_ground_pin/related_ground_pin:*** The related_power_pin and related_ground_pin attributes are defined at the pin level for output, inputs and inout pins to specify the power supplies of pins.

- ***power_down_function:*** The power_down_function string attribute is used to identify the condition when an output pin is switched off by pg_pin and to specify the Boolean condition under which the cell's output pin is switched off (when the cell is in "off" mode due to the external power pin states). If the power_down_function is "1" then X is assumed on the pin.

IV. CHALLENGES in RE-USING UPF

To summarize the various problems occurring in re-using the RTL at GLS are:

1. Design elements are flattened in GLS netlist. So, the signals and instances referred in RTL UPF might require changes when being applied to GLS netlist.

2. The state elements which were signals reg/logic/std_logic at RTL have now changed to instances to implement retention/state behavior. So referring them in UPF might needs to be changed.

3. Isolation/level-shifter/repeater cells are now actually part of the netlist and hence their corresponding strategy is not much meaningful. Also synthesis tool might do the optimization of clubbing isolation and level-shifter cell to create an ELS cell.

4. Additional buffers/always-on cells/feedthrough cells are inserted in the design and they might be on the path of an isolation cell .This potentially changes the source/sink behavior of that isolation cell.

5. Simulation tools need to infer these existing low power cells and use them for simulation instead of inserting their own cells. Cell inferencing requires intelligence and may require extra information through liberty files or from HDL attributes.

6. Power specifications present in Liberty files may be different from UPF specifications. Corruption semantics are defined by power_down_functions which do not exist at RTL stage.

Due to these differences writing a re-usable UPF requires the deep understanding of UPF and various differences in the two netlist.

V. WRITING REUSABLE UPF

A. Hier-path related issues in RTL vs GLS UPF

1. Specifying any vector signal in UPF: A vector signal "A" can be referred in UPF by single occurrence as 'A' or multiple occurrences as A[0],A[1],A[2]. It is always a better choice to list down every bit in UPF although it makes the UPF cumbersome but it will help in GLS verification. Remember than in GLS the

signal will get flattened to \A[0] ,\A[1]… . So writing it in bit wise manner will keep the UPF consistent with GLS.

2. Hier-path scope difference: Sometimes the scope of the signal gets changed when translated to its gate level. For example in below code the scope of the flop srpg_flp1 changes from dft_inst to its generate scope. We couldn't find any simple and LRM compliant way to solve this inconsistency. Our recommendation is to write the elements in the gate level UPF form. Some of the EDA vendors support the UPF written in gate level form to be applied in RTL via more intelligent processing.

RTL state element	Gate Level State Element	Gate Level UPF	RTL UPF

module dft(…)
reg srpg_flp1;
 	
generate
if(NON_SRPG==1)
begin : srpg
always@(posedge clk or negedge rst_t)
begin
if(!rst_t)
srpg_flp1 <= 1'b0;
else
srpg_flp1 <= enable;
end
	module srpg (….
	srff_dff srpg_flp1 (….);
		set_retention ret1 \
		-domain pd \
		-elements {…dft_inst/srpg/srpg_flp1}
			set_retention ret1 \
			-domain pd \
			-elements {…dft_inst/srpg_flp1}

3. Hier-path separator "." for generate hierarchy: A hier-path containing generate needs to be written using "/" hier-path separator as per UPF LRM, but when the RTL is flattened into netlist then the generate names are collapsed into the child instance names. So the hier-path of instance in RTL "/tb/top/gen[0]/mid_inst" would be "/tb/top/\gen[0].mid_inst " in GLS. So its is always a better choice to separate the generate hierarchies with "." Instead of "/" , although it is not the LRM compliant way. We would like to see allowing the "." Just like "/" in UPF LRM.

4. Use find_object command wherever possible, since find_object command supports wildcard based search also so a little change of name in GLS would not be a problem for RTL UPF.

RTL	GLS	RTL UPF
For(i=0;i<num;i=i+1) Begin:cfg_gen Hm_cfg hm_cfg1_mem1(…) Hm_cfg1_mem2(…) End	Hm_cfg \cfg_gen[0].hm_cfg1_mem1(…); Hm_cfg \cfg_gen[0].hm_cfg1_mem2(…);	Set ret_exclude_list [join [find_objects . –transitive true –pattern *hm_cfg1_mem* -object_type instance]]

B. PA cell inferencing

1. In RTL stage all the PA cells like isolation/level-shifter are not present in design. Only strategies are defined in UPF and tool inserts and power them automatically. But in Gate level Netlist cells are already present in the netlist and tool needs to connect it's supply pins properly. For the cases of retention in fact the retention flops which were represented as signal in the RTL becomes instance name in Gate Level as shown in above figure 2. For proper pg-pin connections of these cells tool needs to associate them with correct strategy. To ease the tool's job following items are recommended:
    a. If designer already know the name of actual implementation cell going to be used during gate level but doesn't have the actual cell then he should specify additional UPF command map_*/use_interface_cell to specify the "-lib_cells" option. This will help Gate Level simulation tool to identify and associate the cells correctly with strategy. If the actual implementation model is available during the RTL simulation itself then use the "-lib_model_name" option of these map_* command to simulate the implementation model at RTL itself.
    E.g `map_isolation_cell ISO1 –lib_model_name {iso_buf_mux`

2. Specify the signal/ports on which the strategy is going to be applied. For isolation and level shifters it is not possible in first go to list out these signals individually because writing these strategies as -source/-sink is easier, concise and more relevant. The same -source/-sink tends to break in Gate Level Netlist because of additional buffers, AON and other elements in actual source/sink path. So we would recommend once the RTL UPF with –source/-sink is verified then change the UPF strategies to be per signal basis taking help from tool's internal report or using save_upf approach. Specifying the retention elements in per signal basis helps a lot for set_retention strategy because the sequential elements which are represented as signals during RTL would get converted to retention cell instance.

    Example:
    ```
    set_retention RET –domain PD -elements {srpg_flp1 srpg_flp2 }
    ```

    Please note that same retention intent can be written in more abstract manner e.g
    ```
    set_retention RET –domain PD \
       –exclude_elements {all elements except srpg_flp1 srpg_flp2}
    ```

Writing UPF in manner 2 might create problem during Gate Level because of strategy association complexities.

C. Supply connections to Isolation/Level_shifter/ELS cells

In a GLS netlist an Isolation/level-shifter/ELS cell could be dual rail or single rail. In case of dual rail, the cell has a primary as well as a backup power connection however the same is not possible at RTL stage because it is a tool inserted cell which uses isolation strategy power as its single power source. This disparity can cause mismatch in Gate Level vs RTL simulation because corruption of these cells are being driven by different power source in the 2 stages.

Unfortunately UPF doesn't provide any way to specify the dual connection, neither has it specified the way to define ELS cells which are mostly multi-rail cells. The dual rail for these cells are mainly their source and sink power rails. So getting the information at RTL stage about the source/sink power domains would help mitigate the issues arising due to multiple rails. UPF does provide the concept of UPF GENERICS to access these kind of information. Some vendors have gone further to enhance these GENERICS to access the isolation/level-shifter source and sink supplies. These GENERICS can further be used with bind_checker to write relevant checks to catch the Gate Level scenarios in RTL UPF itself.

Example: How to write a RTL UPF to catch a Gate Level isolation cell issue arising due to isolation enable not being active while the source domain was off.

```
foreach ISO_STTG [query_isolation * -domain PD]
set query_output [ query_isolation $ISO_STTG –domain PD –details].
array set isolation_detail [join $query_output]
```

```
set src_pwr_port [list src_pwr $isolation_detail(upf_source_domain_pwr)]
set src_gnd_port [list sink_pwr $isolation_details(upf_source_domain_gnd)]
…
set ports_list { … $src_pwr_port $src_gnd_port }

bind_checker $instance_name \
    -module checker_isolation \
    -elements $elements
    -ports $port_list
    …
```

In the above example the upf_generics "`upf_source_domain_pwr`"/"`upf_sink_domain_pwr`" are used to extract the source power and ground rail for every isolation cell. Finally these rails are passed as a port via `bind_checker` command to be used in the checker module "checker_isolation".

VI. CONCLUSION

We have presented various cases on how to write RTL UPF so that it is re-usable at Gate Level. However not all cases can be handled in LRM complaint way or without tool additional processing. To summarize, migration from RTL to Gate Level low-power verification requires below items:

1. UPF should be written in such a way that it remains usable and valid for Gate level simulation too.
2. Simulation tools needs to do some additional as well as relaxed processing of same UPF to make it work in GLS.
    a. Lenient in UPF semantics checking.
    b. Use liberty files for missing power specifications
    c. Enhance the search criteria of design elements to search flattened paths
    d. Automatic cell inferencing to avoid duplicate insertion of low power cells and proper connections of existing cells
3. UPF needs some enhancement to make the process of transition from RTL to GLS seamless and easy.
    a. Flexibility to choose path separator. Currently it is "/" but due to synthesis requirements it should allow "." also.
    b. Construct to specify ELS cells.
    c. UPF generics to specify isolation/level-shifter source power and sink power.
    d. Rules to clearly define the connection of power aware cells for non-pg connected netlist.

VII. References

[1] IEEE Std 1801™-2015 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 05 Dec 2015.

[2] Madhur Bhargava, Durgesh Prasad, "Low-Power Verification Methodology using UPF Query functions and Bind checkers.", DVCon Europe 2014

[3] Durgesh Prasad, Jitesh Bansal, "Upf Generics References: Unleashing the Full Potential", DVCon USA 2015

[4] Himanshu Bhatt,Harsh Chilwal, "Golden UPF : Preserving Power Intent From RTL to Implementation", DVCon Europe, 2015