# Hierarchical UPF: Uniform UPF across FE & BE

Dipankar Narendra Arya, Balaji Vishwanath Krishnamurthy, Aditi Nigam, Ali Tahir
Intel Technology

***Abstract*- Power intent specification of a System-on-Chip (SoC) using IEEE 1801 Unified Power Format (UPF) is a complicated task and requires coordinated efforts from RTL, validation, and BE (Back End) teams. Ideally, the same set of UPF files (hierarchical UPF) should be consumed across the different SoC domains, however, due to technological limitations in BE tools and flows, this was not done till today. The tools, flows and methodologies used in past projects dictate that the BE should use merged UPF, which requires 70% extra effort for its generation and about 2 months for the initial bring-ups with multiple configurations and manual hacks needed in Front End (FE). This also leads to a common scenario where verification is done on hierarchical UPF whereas the BE implementations are done on merged UPF. But the process of validating the logical equivalence of merged UPF with the hierarchical UPF is cumbersome and never foolproof. Ultimately this results in higher TAT (turn-around-time) and possible silicon escapes. But with the advancement of tools & methodologies, the industry is now going forward with adopting the hierarchical UPF for all the tools and flows, while most projects in Intel are still using hierarchical UPF for FE and merged UPF for BE implementations. So, in this paper, we describe the learnings from the "proof of concepts" (POC) of using hierarchical UPF for the BE implementations. The recommendations/guidelines for the seamless consumptions of hierarchical UPF, which is being followed by the current SoC, are also listed down. These guidelines empowered the adoption of hierarchical UPF for the current SoC, allowing us to consume the 3rd party IP UPF files directly after the integration, which is the right thing to do and was not possible earlier; and in turn, reducing the manual work required for generating and validating the merged UPF files.***

***Keywords*: Unified Power Format (UPF), hierarchical UPF, SoC, merged UPF**

## I. INTRODUCTION

Traditionally, hierarchical UPF is the original way of writing the UPF where lower-level blocks (IPs) are designed independently from the upper-level blocks (SoC partitions). The complete power intent of the lower-level block is specified at that scope and later integrated at the SoC/par level UPF as described in Fig 1.

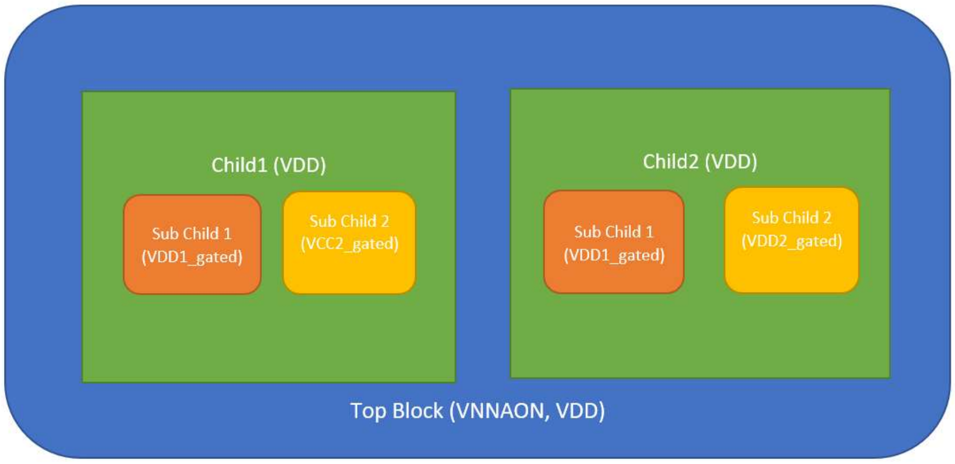

Figure 1. Block representation of a small design

Figure 1 contains two instances of “child” block – child1 and child2. Figure 2 below is the UPF representation of “child” block. Figure 3 contains the snippet of the UPF of the “top” block from Fig 1. It shows the loading of the UPF of the child instances (hierarchical format).

```
create_supply_net VDD
create_supply_net VSS
create_supply_port VDD
create_supply_port VDD
connect_supply_net VDD -ports VDD
connect_supply_net VSS -ports VSS
create_supply_set  ss_VDD -function {power VDD} -function {ground VSS}
create_power_domain PD -elements {.} -supply “primary ss_VDD”
```

Figure 2. UPF for the Child block (child_block.upf)

```
...
load_upf child_block.upf -scope child1
load_upf child_block.upf -scope child2
...
```

Figure 3. UPF for the Parent block (Hierarchical UPF)

**Flat UPF** is generated by flattening out the hierarchical UPF files into a single UPF file. All the children UPF components like power domains, power switches, isolations, level-shifters, etc. are brought to the same scope as that of the topmost level UPF. Figure 4 shows the flat UPF for the “top” block from Figure 1 (by flattening out the UPF from Figure 2 and Figure 3).

```
create_supply_net VNNAON
create_supply_net VDD
create_supply_net VSS
create_suppy_port VNNAON
create_supply_port VDD
connect_supply_net VNNAON -ports VNNAON
connect_supply_net VDD -ports VDD
connect_supply_net VSS -ports VSS
create_supply_set ss_VNNAON -function {power VNNAON} -ground {VSS}
create_supply_set ss_VDD -function {power VDD} -ground {VSS}
create_power_domain PD_top -elements {.} -supply “primary ss_VNNAON”
create_power_domain PD_child1 -elements {child1} -supply “primary ss_VDD”
create_power_domain PD_child2 -elements {child2} -supply “primary ss_VDD”
```

Figure 4. Flat UPF

**Merged UPF-** In the above flattened UPF, if the primary supplies of two or more domains are same, then they are treated as equivalent, and are merged to create a simplified UPF for the BE consumption.

```
create_power_domain PD_child_merged -elements {child1 child2} -supply “primary ss_VDD”
```

Figure 5. Merged UPF snippet

## A. *Traditional UPF generation flow in the SoC world & its Limitations*

An internal RTL, TB, and UPF integration tool is used to integrate the IPs, block-level RTL and UPF, and then create partition-level UPF files. The generated UPF files are hierarchical in nature. In an ideal SoC world, these generated hierarchical UPF files should be consumed for validation as well as for BE implementation. But, because of the tools, flows & methodology (TFM) limitations, downstream implementation tools (synthesis and P&R) could not easily consume hierarchical UPF files. Some of these limitations are - BE flows cannot always properly understand

IP level power intents, PST conflicts may arise because of PSTs in the child UPF, power domain explosions, feedthrough routing between different child blocks, etc.

Because of these limitations, BE flows require merged UPF files as they are easy and straightforward to consume. However, the generation of merged UPF files is a tedious task. It is done via a script that involves various configurations, is buggy, and often requires manual interventions. Fig. 6 below gives the overview of this flow. So, the TAT for the initial setup itself is around 7-8 weeks (as seen in the previous SoCs) and then later, a few days per iteration per UPF generation. There is no industry standard tool available for the generation of merged UPF because of the involved complexities.

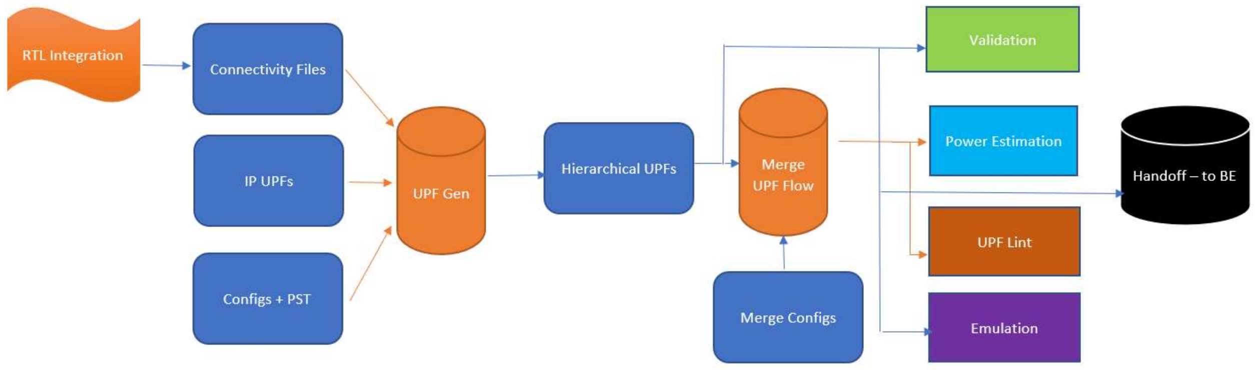

Figure 6. Current UPF Generation flow for SoC

In hierarchical UPF, loading of all the child UPF files are protected by conditional variables, so if a portion of design is needed to be blackboxed during verification, it can be easily done by changing the values of those conditional variables (UPF stubbing). That is why merged UPF files can’t be used for verification as selective enabling/disabling is not possible in it.

Additionally, we need to make sure hierarchical and merged UPF files are equivalent. The only way to check this equivalence is by using the Formal Equivalence (FEV) checking tool. It is done by consuming the same set of RTL files in the FEV tool for the golden as well as the revised side, and giving hierarchical UPF files to one side, and merged UPF to other side. This itself is not a full-proof solution, as it involves a lot of extra mapping commands to manually map power intent cells like isolations, level-shifters and power switches between hierarchical vs merged side. If the mapping itself is wrong, which has happened in past projects, can result in a false match. Also, it doesn’t catch all kinds of differences, and there are also a lot of false non-equivalencies like power-grid (which arises because of merging of equivalent supply networks/PDs), and some corner cases involved with PSTs (Power State Tables). Another issue was seen in the early stages of one of the previous projects - the tool was unable to perform retention cell related mappings in the UPF, and the merge script missed putting the retention cells in the generated UPF files. This miss was caught by BE, and the bug got fixed in the later releases of the FEV tool. Since the debugging and setting up these equivalence check runs is complicated, this approach is risky and time-consuming.

Therefore, even though merged UPF files simplify BE implantations, we can never be fully confident whether the implemented UPF files are fully accurate. Complete equivalence can only be established when the gate-level simulations (GLS) are done on the SoC, which again comes very late in the design cycle, typically during the code-freeze timeframe.

Also, in case of any BE feedback on the merged UPF, this whole cycle of generating hierarchical UPF, deriving merged UPF, functionally verifying/validating, and checking their equivalencies needs to be performed again. This in turns increases TAT (turn-around-time) many folds for the previous SoC project.

## II. POC ON HIERARCHICAL UPF

Consuming hierarchical UPF files for BE implementation will allow us to follow the basic principle of using the same source UPF files everywhere in the design cycle. TAT time for the UPF bring up will reduce, as there would not be a need for any effort to generate and validate the merged UPF files. Any BE feedback on the UPF can be easily implemented.

A strategic decision was made to move to hierarchical UPF for BE for the next SoC project. To arrive at this decision, a POC (proof of concept) was done on one of the most complex partitions (from a hierarchical UPF based implementation point of view) from the current SoC project.

The partition covered most of the complicated power intent scenarios such as the presence of equivalent power switches (which would have been merged into a single switch during merged UPF generation), isolation strategies with both parent and self-locations, ELS (enable level shifters) cells, multiple SIPs (Soft IPs) and HIPs (Hard IPs). All the major tools and flows like FE LP Static-checks, FE handoff Synthesis, BE full-scale Synthesis, APR, BE LP Static-checks, LVS (Layout vs. Schematic), RV (Reliability Verification), and FEV were run.

### *A. Issues, Workarounds and Additional Settings required during POC*

Following are the list of issues/additional settings which were seen across these tools and flows:

- FE LP Static Checks: No Issues seen.
- FE handoff Synthesis:
    - Last gen Syn tool was not able to insert isolation strategies for child UPF at parent location, which was indeed a bug. The same was not seen with next-gen Syn/Apr tool, which is the POR (the plan of record) tool now.
    - Missing connect supply nets for internal nets resulted in errors, because voltage levels for those supply nets were not defined.
- BE full-scale Synthesis:
    - Same as above
- APR:
    - There were flows crashes initially, which were later found out to be due to improper buffer and inverter placements.
    - Additional options were required to merge equivalent voltage areas (VA), and equivalent power-switches, shared_voltage_area attributes, creating power switch array for the merged PSWs (power-switches), etc.

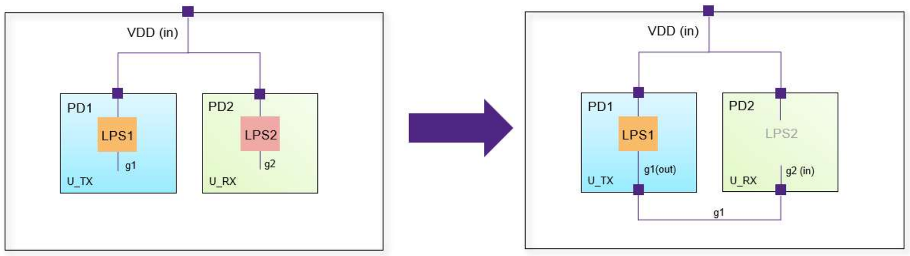

Figure 7: Handling Equivalent Power Switches in APR

- BE LP Static Checks:
    - One minor bug was found for the "shared_voltage_area attribute". The tool was unable to comprehend this option if it was used for multiple elements in separate lines. Ticket was filed on the vendor and subsequently, this got fixed in the later versions of LP Static Check tool.
    - There were many violations related to merging of PSWs in voltage areas. These were found to be expected and could be waived.
- LVS Checks:

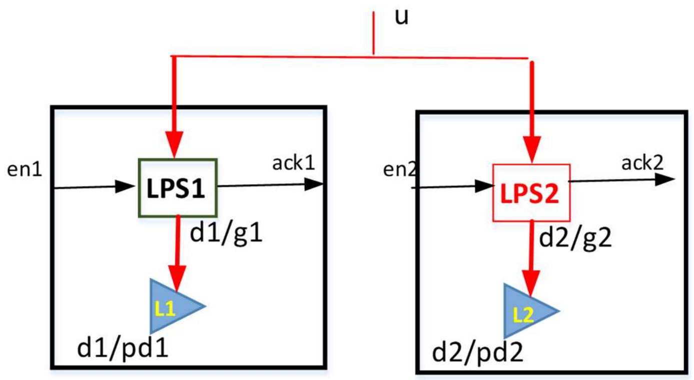

Figure 8: RTL + FE UPF view

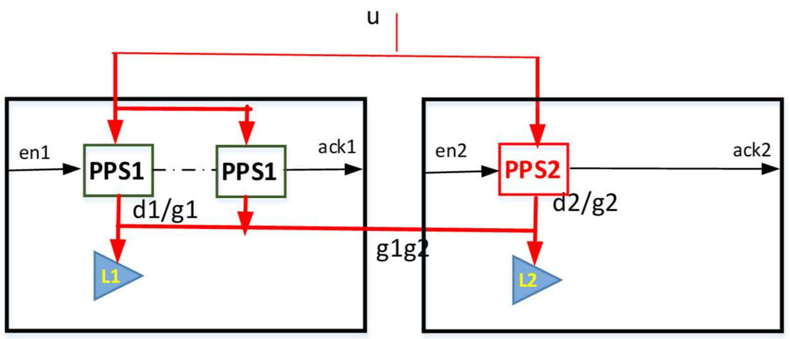

Figure 9: BE logical view

  - Output nets of equivalent power switches (LPS1 and LPS2 in Fig. 8) - supply nets d1/g1 and d2/g2 once get converted to power nets, are two distinct power nets that are not shorted by default. However, in a single VA, we can either create d1/g1 power-grid or d2/g2 power-grid. If the floorplan tool uses d1/g1 as the primary power of the VA, dg2/pd2 domain cells will be logically connected to dg2/g2 power net but physically connected to dg1/g1, thus creating an LVS open. There are two solutions for this-
    - Use UPF 2.x set_equivalent command in the UPF to short the output supply nets

      *set_equivalent -nets {d1/g1 d2/g2} -function_only*
    - Port out the gated supplies to the top-level UPF and short them there. This is the approach we followed here.
- RV Checks: No Issues
- LP FEV:
  - Some power-grid violations related to merging of power switch VAs in APR came up, which was expected.
  - Since our test case UPF didn't had biasing, but the process node was with biasing, there were many violations related to that.

### *B. UPF Handoff to BE - Localized UPF and streamlined *.cfg files*

Usually, FE UPF files are heavily dependent on variables, environment settings, and central paths. It would be a challenge for BE to be able to consume these UPF files directly as the execution environment is completely different. So, to make the shipping and consumption of these UPF files easy, we have enabled a flow known as **localized UPF**. This flow copies all the child UPF files of a partition in a single directory and updates the "load_upf" paths accordingly. Also, we have minimized the usage of custom variables and streamlined the .cfg files (configuration files) in accordance with the Intel-wide approved templates to avoid potential conflicts.

## III. Results And Learnings

Overall, no roadblocking issues which could potentially hamper BE implementation were discovered. Therefore, for the upcoming SoC project, the result was presented to the management, and the decision was made to use the hierarchical UPF for BE implementation as well. Merge UPF generation flow was still enabled as a backup to account for if any major issue is observed for a partition during BE flows.

Using the hierarchical UPF has a separate set of challenges, however, we could overcome them with strict compliance to certain guidelines. Below are the guidelines which we published for FE integration as well as IP teams well in advance for the easy execution-

- Strict compliance to the Intel-wide UPF 2.1 templates. Mix and match of UPF versions are not allowed.
- Well-biasing should be done for all the supply sets present in all the UPF files. Even a single miss will cause implementation tools to fail. There isn't any rule in the POR LP Static-check tool to detect this.
- The value of attributes used in "set_design_attributes" commands should be consistent in all the UPF files. For eg. – if the value is of the attribute is "TRUE" in the topmost UPF file, the child UPF can not have "FALSE" value.

  *set_design_attributes -elements {.} -attribute {<attribute_name> TRUE}*
- Usage of deprecated (and legacy) UPF constructs can cause errors during Synthesis. Some of these constructs are "set_domain_supply_net", "extra_supplies_#" etc. Again, these can-not be caught during LP Static-check runs.
- All the gated supplies from the IP should be ported out to the par level. This is done to short the gated supplies in case of equivalent power switches and also to have a complete PST at the par level.

- Add protection around “add_power_state” command and PSTs for IP UPF files. This is to avoid PST related conflicts during implementation. This means the topmost PST should contain all the legal combinations.
- “connect_supply_nets” should be done for all ports and nets. Any missing CSN will result in failures in BE tools, and violations in LP static-checks.
- Internal supply nets should be brought to the par level. This is to enable adding filler cells etc. during APR on those supply nets.

During the initial stages of the current SoC project, there were multiple failures during execution, because these guidelines were not properly followed.

## IV. Summary

The POC done on a complex partition from the previous SoC project showed no major challenges or degradation in QoR. So, it was decided that hierarchical UPF will be the POR for the ongoing SoC project with certain guidelines for all the partitions, with merged UPF as a backup in case of any unsolvable TFM issues for BE.

Initial bring-up/integration of the FE UPF took some effort because many of the IP UPF files were not adhering to the guidelines. However, till writing this paper, no Show Stopper issue is found during the current SoC execution.

This adoption has resulted in a significant reduction of the time and resources required to configure and generate the merged UPF files, validate their equivalencies with the corresponding hierarchical UPF files, and consume external/3rd party IP UPF files (used as is in hierarchical flows). This has significantly simplified the FE flows and TAT for the UPF integration is reduced by 60-70%, resulting in a saving of 8-10 weeks for initial bring-up and 2-3 days for each of the subsequent drops. This has also ensured that the basic principle of consuming common UPF files across the design is followed. Additionally, LP static checks are now being directly run on the IP UPF files, resulting in faster debugs and fixes in the SoC design cycle.

## Acknowledgment

Thanks to William G. Crocco, UPF-MPP WG Chair, for the continuous support and the guidance. Thanks to Michael Fish, Ronen Yosef and Adi Yacobson to share their learnings and inputs. Thanks to Sachan Deepmala for pushing for hierarchical UPF as the POR UPF methodology for the ongoing SoC.

## References

[1] IEEE 1801-2013, "IEEE Standard for Design and Verification of Low-Power Integrated Circuits".