2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Hybrid Flow: A smart methodolgy to migrate from traditional Low Power Methodology

Rohit Kumar Sinha, Intel India Pvt Ltd, Bangalore, India (*rohit.kumar.sinha@intel.com*)

N, Prashanth Intel India Pvt Ltd, Bangalore, India (*prashanth.n@intel.com*),

Abstract— *Traditionally most of the SoC design companies have adopted merged based low power implementation methodologies but just as the complexity of an SoC demands a well-structured hierarchical approach to design and verification of its functional specification, the complexity of the power management infrastructure for a SoC requires a hybrid methodology. Because of the strict tape-out schedules and dependencies on internal and external IP teams, transitioning from merged to hierarchical and from UPF1.0 to UPF2.0 for the complete SoC design could be of huge risk as there are lots of uncertainties in terms of QoR. In order to mitigate such risk, we adopted a hybrid methodology which ensures complete reliability and smooth transition. Hybrid flow is essentially a mechanism to enable both merged as well as hierarchical UPF based on partitions and also, it enables to use both partition level UPF1.0 and UPF2.0 syntax in the SoC design Flow*

I. Introduction to Hybrid Low Power Methodology

In this paper, we propose a smooth and reliable hybrid approach which has been validated across all the design flows- static checker in the front end, functional verification, emulation, synthesis flows & APR flows. We will present the 5 most prominent challenges posed by migration from merged based to hierarchical low power implementation flow and from UPF1.0 constructs to UPF2.1 based phased approach and how those can be overcome in hybrid flow.

The key challenge that are identified and resolved with the proposed approach are –

- Development of bottom-up hierarchical PSTs from low power specification in the hybrid flow
- Isolation strategies correctness and consistency in hierarchical UPF2.x based flow
- Handling of domain dependent supply set and domain independent supply nets and handling buffers Across Scoped/Hierarchical Voltage Areas
- Handling of instrumental assertion code in UPF2.x syntax in functional verification and emulation environment
- Handling of SRSNs which are now set using the set_port_attribute command. If the IP provides the SRSNs, using an automation script we can migrate from SRSNs to set_port_attributes for setting SRSNs.

Intel standard flow has been using merged low power implementation flows using UPF1.0 syntax for power intent specification but because of the adoption of sophisticated power management architectures, there is a strong need to migrate to the hierarchal implementation approach using the latest IEEE P1801-2013 (UPF 2.1) low power specification. But since in a complex SOC, IP and its UPFs are sourced from multiple vendors (both internal & external channel), it is not easy to mandate IP vendors to provide UPFs in a specific format. That's why there is a strong need to develop a HYBRID flow such that SoC methodology is well equipped for handling merged as well as hierarchical based low power implementation flows while handling both versions-UPF1.0 and UPF2.x syntax through automation such that IP handoffs in either format can be consumed for Front end static flow, functional verification, emulation, synthesis & APR flows.

At Intel, the design handoff process consists of multiple phases. The typical design hierarchies for integrating IP blocks into a SoC are:

IP block → Unit →Partition→SoC

1. Process

   a) The power architecture and power domains & voltage area (VAs) are well understood, documented and verified
   b) The approved and recommended settings are used for power intent specifications are used for the validation.

1.1. <u>IP-level</u>

   a) IP will design the power intents for their blocks depending on the power architecture documents
   b) The power intents designed should be tested thoroughly through the various phases like Front end static checks using the static check tools, functional verification and regressions involving these power intents.
   c) All the static checks and functional verification results should be clean without any errors and it should be of IP-handoff quality.
   d) As per IP signoff process, IP UPFs & low power check waivers should be provided to the partition owners along with the other IP collaterals.

1.2. <u>Unit-level</u>

a) Partition owners should gather the IP UPFs from the appropriate IP owners. For the Soft IPs, the UPF path needs to be known. For hard IPs, both the UPF and liberty file information is needed. The hard IP UPFs are needed for functional verification purposes.
b) Also if any SRSNs needs to be set at IP level, they should be obtained from IP owners.
c) The IP UPFs needs to be loaded at Unit level using load_upf commands.

1.3. Partition level

d) The creation of UPFs at partition level is based on hierarchical approach
e) The partition owner is supposed to load the unit UPFs. Also the static checks should be run and the UPFs should be clean without any Errors.
f) If any Errors are reported at IP level during the partition level runs, they should be reviewed with IP team and the appropriate fix needs to be obtained.
g) All the static checks, functional verification and emulation results should be clean.

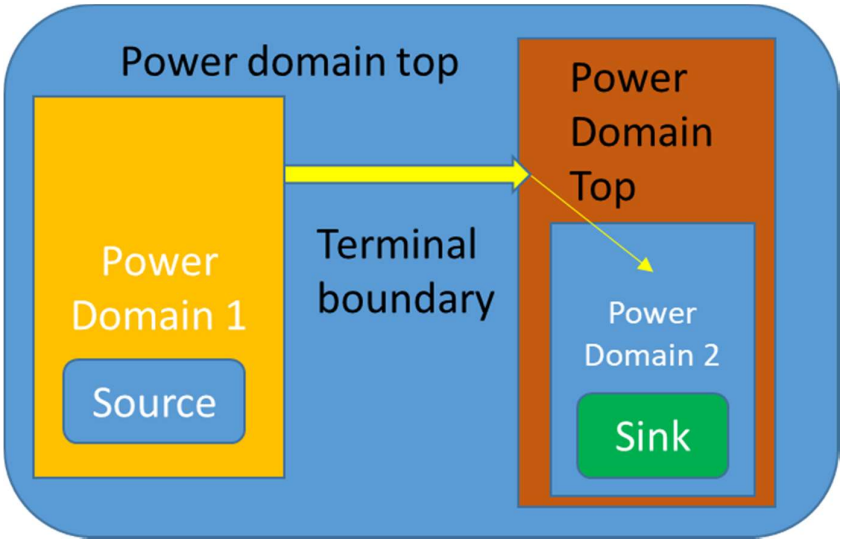
1.4. SoC level

h) SoC level UPF loads the partition level UPFs. Here also, the hybrid UPF methodology is adopted.
i) All the static checks, functional verification and emulation results should be clean.

II. Hybrid Flow Implementation Challenges

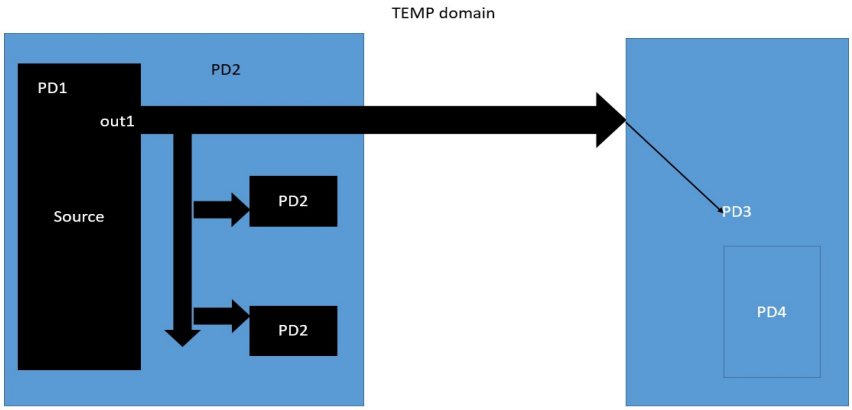
There are multiple challenges faced in implementing hybrid flow methodology. Please find below some of the challenges

1. Terminal Boundary Issue : Isolation cell with source and sink strategy got matched but isolation cell is not placed due to crossover contain a wire which belongs to terminal_boundary



But due to terminal boundary set on BLK1 element and signal is crossing via wire which is used in BLK1 is breaking this connection .

2. For “shift operator” issue : Isolation cell with source and sink strategy got matched but isolation cell is not placed due to ">>" right shift operator in port connection as below
3. Instrumentation Code Issue : In Source-Sink isolation strategy, isolation will be skipped if signal(output/input) is matching with multiple sinks (sinks with different power domains) and VCS-NLP give’s “HETEROGENEOUS_FANOUTS” Warning (Isolation is skipped)



4. PST Merging Issue: In hybrid flow, the partition level PSTs are generated using automation but since there are multiple syntax of PST such as “add_pst_state” in UPF1.0 and “add_power_state” in UPF2.0, implementation flow needs handling of state names such that there is no conflict in the states being used at the SoC level.
5. Issue related to buffering Across Scoped/Hierarchical Voltage Areas
6. Issues with Macro/Cell Pushdown Into Scoped/Hier Voltage Areas
7. APR tools demands matching between states in top level PST and sub hierarchy PST

```
--------------------------------------------------------------------------------
PST Name : pst
Scope              : / (top scope)
--------------------------------------------------------------------------------
Supplies    : vcca                    | vccsb             | vccc              | vss
States
 (1) S_VCCB_on_LV : ps_VCCA_HV [1.200] | ps_VCCSB_HV[1.200] | ps_VCCC_LV[0.70] | ps_VSS_0[0.00]
 (2) S_VCCD_on_LV : ps_VCCA_LV [1.200] | ps_VCCSB_HV[1.200] | ps_VCCC_HV[0.70] | ps_VSS_0[0.00]
 (3) S_VCCB_on_LV : ps_VCCB_LV [1.200] | ps_VCCSB_HV[1.200] | ps_VCCC_HV[0.70] | ps_VSS_0[0.00]
 (4) S_VCCE_on_LV : ps_VCCC_LV [off]   | ps_VCCSB_HV[off]  | ps_VCCC_LV[0.70] | ps_VSS_0[0.00]
 (5) S_VCCE_on_LV : ps_VCCA_HV [off]   | ps_VCCSB_HV[off]  | ps_VCCC_LV[0.70] | ps_VSS_0[0.00]
 (6) S_ALL_RAILS : ps_VCCA_HV [1.200] | ps_VCCSB_HV[1.200] | ps_VCCC_LV[0.70] | ps_VSS_0[0.00]
```

8. Automations for hybrid flow-The add_power_state command defines power states of a supply set or a power domain. Currently in the hybrid methodology, we are generating the PSTs in UPF 1.0 format. The add_power_state commands are flexible and complex power state definitions can be constructed. If we properly understand the relationships between different supplies and functional verification states, we can easily migrate from UPF 1.0 to UPF 2.1 PSTs through a migration script.

III. Case Study On Actual Design

This methodology is implemented at three blocks in SoC. Currently all the four blocks use UPFs in 1.0 format. The type of blocks which we have used on our conversion are power management block, fabric block, mc main block and IOP block. Power management block controls all the control signals required for power management and power gating. The fabric partition acts as a connectivity partition. For the analysis, we are considering the UPF 1.0 run results as reference. We are converting the UPFs in 1.0 format to 2.1 hierarchical UPF format at each of the sub blocks within the main block. The runs which were performed are completed successfully without any issues. The analysis was also performed with the inclusion of source and sink for isolation strategies. The four blocks have different cell counts and are with varying levels of complexity as below. The power domain information is also mentioned in the below table. Out of 35 partitions, 5 partitions are in UPF 2.1 syntax and remaining are in UPF 1.0 syntax.

Block Type	Cell count	Number of power domains
BLOCK1	~48000	3
BLOCK2	~24000	3
BLOCK3	~87000	3
BLOCK4	~17000	3
BLOCK5	~4000	3

Table 3.1. Types of blocks at SoC level used for analysis, the cell counts of each block and the power domain information

The above partitions have 3 power domains. But the number of power domains vary across the different partitions. Among the above partitions, MC main partition has maximum number of units of 23 and each unit has its own power domain. But after power domain merging, we are getting total of 3 power domains.

The results were analyzed based on the following criteria:

- Compare the results (Errors/Warnings/Fatal) in the test log for 4 standard Intel flows
- Front End Static checks using VCLP
- Functional verification using VCS
- Emulation Flow using Zebu
- Synthesis and implementation flow using Design Compiler and ICC2

Initially the analysis was mainly focused on the VCS results and on merged UPF in 2.0 format. Due to the presence of hetero fan outs under certain cases and due to assertions, some of the isolation strategies were getting missed in our VCS runs as shown in the Table 3.2

Design: Block B	**Result without VCS patch**	**Result with VCS patch**
_mc_main//*_stf_data_out_hub_ctrl_STF_PKT	Iso logic missing	Iso logic inserted by VCS
_mc_main//*_stf_data_out_child_ctrl_STF_PKT	Iso logic missing	Iso logic inserted by VCS

Table 3.2. Missing isolations case due to right shift operators

Type of Block	**Right shift operators (**(Total Missing Low Power Cells)	**Terminal boundary issue0**(Total Missing Low Power Cells)	**Missing isolations for Instrumentation code**
BLOCK1	~50	~75	Found
BLOCK2	~75	~70	No instrumental code (assertion Code)
BLOCK3	~25	~10	No instrumental code (assertion code)

Table 3.3. Missing isolations in all cases

Also as explained in the previous sections, the terminal boundary issue is resolved after defining terminal boundary for 3 blocks and at SoC level. Now all the missing isolations are now inserted in VCS. Hetero fan out issues are resolved after adding the design attribute constraints for instrumentation code.

The next level of analysis is completely focused on conversion of merged UPFs in 1.0 format to 2.1 hierarchical UPFs and performing different levels of analysis as mentioned above. The sub block level UPF generation is done by using the automation script. Then a hierarchical UPF is developed at the block level. Some of the samples of UPF development are as shown below. The important constructs of UPF 2.1, which are the supply sets, set_design_attributes and add_power_states have been used. A supply set is a collection of supply nets. A supply set is a unified and progressively defined bundle of supply nets that is not specific to a particular power domain. add_power_state command can be used to specify the voltage values over the supply sets.

```
create_supply_set VCCA –function {power vcca} –function {ground vss}
create_supply_set VCCB –function {power vccb} –function {ground vss}
create_supply_set VCCC –function {power vccc} –function {ground vss}
create_supply_set VCCD –function {power vccd} –function {ground vss}
create_supply_set VCCE –function {power vcce} –function {ground vss}
create_supply_set VCCf –function {power vccf} –function {ground vss}
```

Fig 3.1 Supply sets used in UPF

```
Set_isolation punit_vccsa_vccsb_iso_1
                -domain "vccsb_domain_merge" \
                -isolation_supply_set "VCCSB" \
                -isolation_signal "punit_*/*/ptpclk/*soft_supply" \
                -isolation_sense "low" \
                -location "self" \
                -elements [list \
                "punit_*/*/ptpclk/PII*" \
                "punit_*/*/ptpclk/XXBclk*" \
                "punit_*/*/ptpclk/tap*" \
```

Fig 3.2 Supply sets used in set_isolation command

```
add_power_state "VCCA" –state "ps_VCCA_LV" –supply_expr \{power= `\{FULL_ON, 0.8\|\}–simstate NORMAL"
add_power_state "VCCA" –state "ps_VCCA_UV" –supply_expr \{power= `\{FULL_ON, 1.4\|\}–simstate NORMAL"
add_power_state "VCCA" –state "ps_VCCA_MV" –supply_expr \{power= `\{FULL_ON, 0.95\|\}–simstate NORMAL"
add_power_state "VCCA" –state "ps_VCCA_OFF" –supply_expr \{power= `\{OFF\} –simstate CORRUPT"
```

Fig 3.3 Power states used in UPF

One needs to identify the different sub blocks and their power domains and the required isolation or level shifter strategies if there are any crossings between the units. So we have identified the sub blocks for each of the main block and the associated power domains for each of the block mentioned above. We have followed all the five criteria mentioned in the top. The snapshot of the hierarchical UPF conversion is as shown in Fig. 3.4

```
Load_upf $::env($MODEL_ROOT)/source/*/pgpunt_wrap.upf –scope punit_wrap
connect_supply_net "VCCA" –ports "pgpunt*/vcca"
connect_supply_net "VCCB" –ports "pgpunit*/vccb"
connect_supply_net "VSS" –ports "pgpunt*/vss"

Load_upf $::env($MODEL_ROOT)/source/*/soc_pg_pwrup_cell_wrap.upf –scope soc_pg_punit_wrap
connect_supply_net "VCCA" –ports "soc_pg_pgpunit*/vcca"
connect_supply_net "VCCC" –ports "soc_pgpunit*/vccc"
connect_supply_net "VCCD" –ports "soc_pgpunit*/vccd"
connect_supply_net "VSS" –ports "soc_pgpunit*/vss"
```

Fig 3.4. Hierarchical UPF conversion using the load_upf commands

After the hierarchical UPF conversion, the first step was to make the VCLP runs for each of the blocks mentioned above. The results of the runs pointed out certain issues, which are explained below

- The UPF_PRIMARY_UNAVAIL errors even though primary power net is declared for the domain.
- The UPF_CSN_UNAVAIL errors even though the supply nets are part of supply set functions.
- The ISO_SUPPLY_UNAVAIL errors where the UPF supply is flagged as missing but it is already part of ground functions of supply sets.

All these issues were not reported in UPF 1.0 runs but were reported in UPF 2.1 runs. These issues are reported to the Synopsys team and have been acknowledged by the Synopsys team. Apart from these, the VCS and VCLP run results seemed consistent across both the type of UPFs. The number of level shifters and isolation cells inserted are consistent across both the merged and hierarchical UPF runs.

Challenges faced in the Physical Design

1. IC Compiler 2 demands matching between states in top level PST and sub hierarchy PST. There were some cases where certain voltages were missing in sub hierarchy PST or even without any states at all. The solution for this is to disable lower level PST.

2. Each IP comes with a set of available supply nets (secondary nets) which can vary. Ultimately a group of hierarchical PDs in the same VA will share same primary supplies but if there is a mismatch in secondary domains, as long as at least one domain contains the required supply as available supply, tool can use the hierarchy of the domain to put the cell anywhere in the VA physical shape.

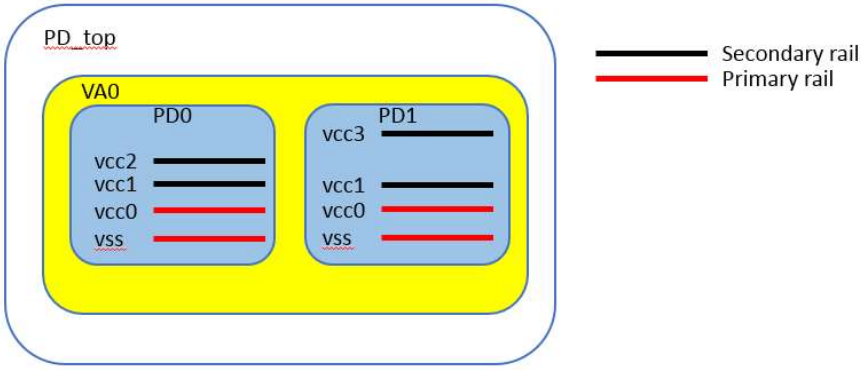


Fig 3.8 Missing Supply states

3. In versions before ICC2 2016.12, there is no support for choosing the hierarchy where the power switch will be inserted – therefore, if it is inserted in a hierarchy which is not the domain specified in the UPF rule. Solution lies in setting specific hierarchy to be used for SW insertion by using the **contains_switches** attribute of **set_design_attributes** command in UPF

```
create_power_switch sw_*_vcca_PGD –domain *_wrap/*/pd_*_vcca_PGD
-output_supply_port {gtout npk_wrap/npk/vccagnpk} –input_supply_port {vcc_in
*_wrap/*/vccs”
Error: Supply net *_wrap/*/vcca cannot be connected to the pin
Soc_*/*/vcc_in in domain soc_*/*/*_wrap
Error: Problem in connect_supply_net
```

4. Using shadow domain solution for optimization in ICC2.

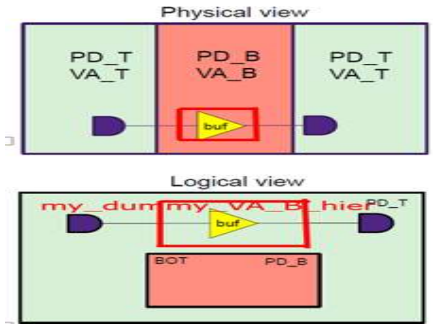


Fig 3.10 Shadow Domain Concept

Moving from the traditional 1.0 UPF methodology to hybrid UPF 2.1 methodology has certain advantages

1. As the complexity of SoC is increasing, it needs a well-structured hierarchical approach for the design and verification of its functional specification. Similarly, the complexity of the power management infrastructure for a SoC requires a hierarchical methodology

that supports partitioning, parallel development, and reuse. The hybrid methodology which we have used in our case study supports this.

2. The UPF complexity is very much reduced. The UPF structure is precise and easy to read. It is evident from the conversion of the UPFs in 1.0 to 2.1 in our case study.
3. We have used supply sets during the conversion of UPF 1.0 to UPF 2.1 syntax. The supply sets provides an abstraction and allows designers to define their power intent without having to create the actual supply nets which may not be known at the very early stages of the design.
4. Usage of hierarchical flows at all the stages of design flow eliminates the usage of two different UPFs. Hierarchical UPFs at front end side and merged UPFs from physical design side.
5. Usage of add_power_state which is a more powerful way to capture the relationship between the supply nets which we have used in the conversion of power state tables in 1.0 to UPF 2.1.
6. Usage and maintenance of two different kinds of UPFs is avoided, hierarchical UPFs at front end side and merged UPFs from physical design side.
7. Even if we want to use the repeaters in our design, the voltage and placement of repeaters can be done using the set_repeater command. UPF 1.0 does not support this option.
8. Support for the creation of atomic power domains and support for separate power modeling for the hard IPs.
9. The usage of set_design_attribute allows the propagation of power information to the lower levels of design hierarchy and even for the definition of set related supply nets on the ports in our design, which we have used in our hierarchical UPF building.

Future scope

The new UPF 3.0 standard offers additional enhancements over the UPF 2.1 constructs. We have been constantly working with our UPF support team to work on implementing the UPF 3.0 constructs for hierarchical UPF and unit UPF building. Some of the commands which are being tested are as shown below:

- set_port_attributes
- add_power_state
- create_power_state_group

We are testing the usage of different UPF switches related to the above UPF commands. These commands will be tested across the different phases as mentioned earlier.

IV. References

[1] Erich Marschner, John Biggs, Unleashing the Full Power of UPF Power States, DVCon 2015

[2]Stepping into UPF 2.1 world: Easy solution to complex Power Aware Verification, DVCon 2014

[3]Amit Srivastava, Rudra Mukherjee, Erich Marschner, Chuck Seeley, Mentor Graphics, Sorin Dobre, Qualcomm, DVCon 2012: 131-II287: Low Power SoC Verification: IP Reuse and Hierarchical Composition using UPF

[4] E. Marschner, December 6, 2012, “The Next UPF”, Semiconductor Engineering, http://semiengineering.com/the-next-upf.