2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Simplifying Hierarchical Low power designs using Power Models in Intel Design

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

***Abstract*: In the Client SoC world, IP designs are sourced from both internal and external channels and SoC team faces multiple challenges in building the bottom-up SoC UPF and in performing power aware functional verification in the hierarchical low power design flow since there are quite a few HIPs in the design which are imported with their own implementation as well as simulation models. In this paper, we will demonstrate how to replace the traditional process of building SoC UPF for hard macros with a much more efficient method by enabling power models in the hierarchical UPF. Also, this paper focusses on more accurate low power verification across all the frontend flows such as static verification, power aware simulation as well as power aware emulation.**

Keywords--Power Aware; Low Power Design; Hard Macros; HIPs, UPF; Liberty; Design Implementation; Function Simulation; Emulation

I. INTRODUCTION

Modern low power designs involve many Hard IP's having number of instances with their power management architecture. For soft IP's, the power intent can be defined in UPF Format using traditional Hierarchical UPF writing approach, in which the power intent is divided at the IP boundaries and written in separate files. For power aware simulation at the RTL stage, simulation models are required which defines the power management architecture of the IP. For Hard IP, the behavioral model which is shipped with the IP is used which contains only the internal functionality and not the power aware behavior in it. Here, UPF power model building approach comes into picture which specifies the power intent of Hard IP in 2 parts, one which describes the internal behavior and one which gives the interface information or connectivity with the other IP'. So, the UPF based simulation can be done at the RTL stage for the Hard IP's also. Here are some things, which should be taken care of while moving to modular UPF approach for Hard IP's:

- Ensure the correctness and consistent results with Hard IP power models across low power static checker, power aware simulations and power aware emulation.
- Handling of domain dependent supply set and domain independent supply
- Handling of instrumental assertion code in UPF2.0 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
- Equivalence checking between the traditional hierarchical UPF flow and the UPF flow with power models.

The Unified Power Format (UPF) is the standard for specifying the power control logic and its design connections. When the power logic is not instantiated in a design, the power intent is absent in the RTL and is extracted from the UPF file during synthesis.

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

Figure 1 shows a typical power aware design flow with UPF. UPF is available at RTL level and is not embedded in the RTL design. UPF contains design's power architecture and additional power elements. Power elements become part of the design after synthesis is complete.

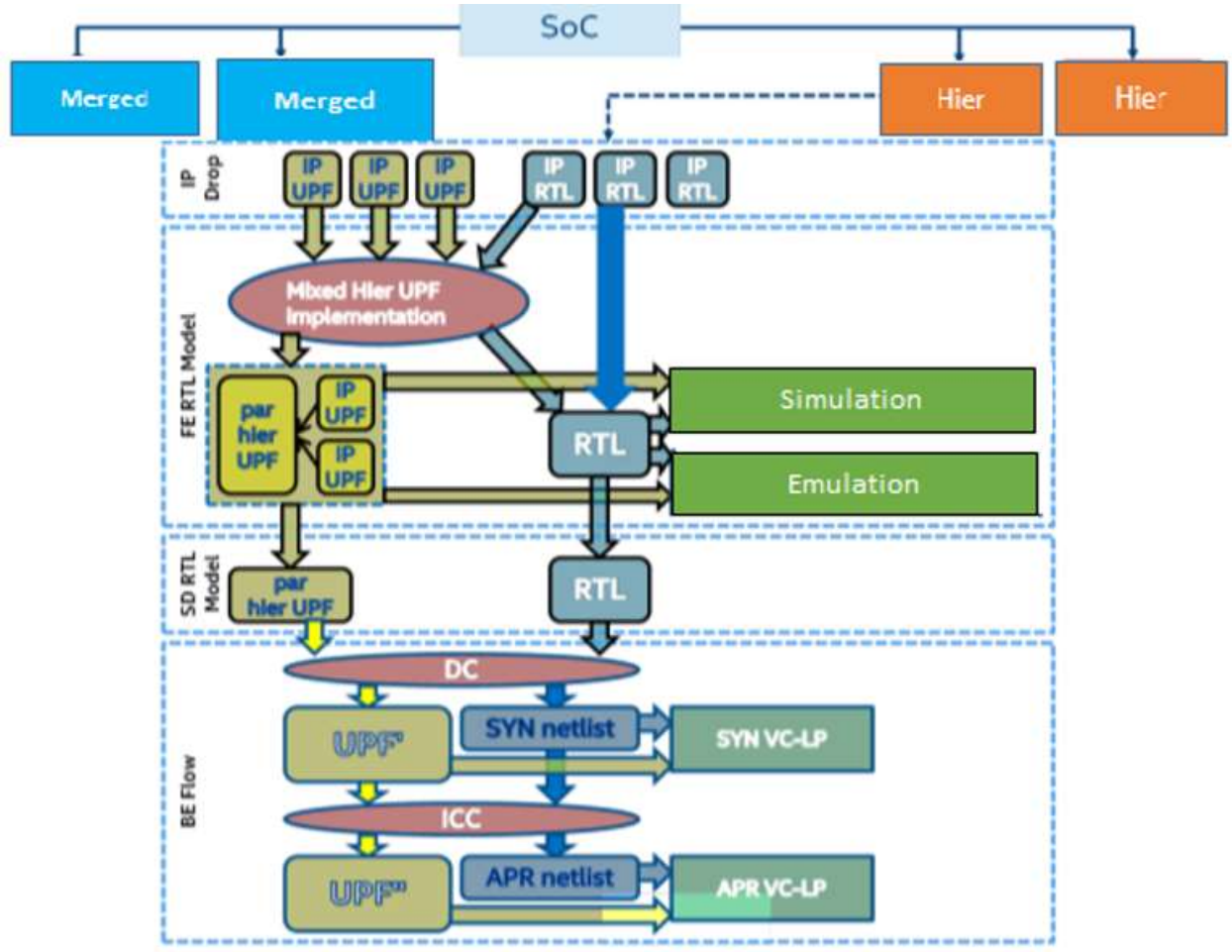

Figure 1. Hierarchical UPF flow in Client SoC

II. POWER AWARE SIMULATION OF HARD IP'S

The power aware simulation of soft IP can be done with the conjunction of Behavioral model (HDL) with the power intent specified in UPF format. But when the soc design integrates various multi-rail Hard IP's whose power supplies behavior impacts the power simulation in their own way. This kind of behavior may not be defined in UPF format. That's why Hard IP's requires simulation models with embedded power aware behavior. The PA simulation models for Hard Macros which contains both the design functionality as well as the power management behavior results in some extra supplies in the model interface, then the UPF semantics are needed to be disabled in this case. So, for this case the power aware simulation depends only on the HDL model only.

Limitations of using behavioral HDL models for power aware simulations:

- Problem in extending the traditional non-power aware behavioral models with power aware functionality due to the high complexity of behavioral models of Hard IP's
- Power management information such as power states, voltage values are difficult to be captured in HDL

So, the behavioral models are not fully able to describe the power management features causing requirement of a UPF model containing power aware information which can support the UPF semantic based simulation tools and also to perform static checks at the RTL stage. Hence, for the UPF based simulation at the RTL stage, additional power management information is required which is given in terms of power aware models.

Power model is all the more effective when you have the of variety of pre-implemented complex IPs such as dekel, phy IPs or fia IPs in a SoC often lack appropriate power aware simulation models, forcing users to rely on gate level or analog simulations for complete power aware verification. This limits the effectiveness of power aware simulations at RTL.

III. POWER AWARE MODELLING OF HARD IP

As we have seen that for the Hard IP, the simulations models can't be fully expressed as HDL behavioral models or using UPF semantics. To overcome this barrier we need to combine both HDL model and UPF things in the same model which is called ad Power Aware Model.

A. Power Aware Model Components:

Power Management Interface:

For Hard IP, the interface information contains information about supply pins at macro boundary, isolation or level shifting cells, power states information etc. These can be expressed in UPF or liberty. It includes the information of the top level power domain and connections with it.

Interface information mainly contains information about:

Supply Information- Pins at the boundary of IP need to be connected to the supplies correctly. Pg_type liberty attribute defines the function of the pin. This supply type info will help in making correct mapping of supply values coming from UPF.

Related Supplies- Integration of the Hard IP requires the pins information of the drivers and receiver supplies which can be specified with the related supply attributes defined for those pins of the Hard IP.

Power States- There can be different power modes in which IP can operate. These power states are specified using add_power_state, create_pst and add_pst_state UPF commands. During integration, these information helps in ensuring correct power mode of the IP.

Interface Protection Cells- If the IP already consists of the isolation cells inside it, then this need to be conveyed during integration to avoid duplication of the cells.

1) Power Management Behavior:

The behavior is generally captured in HDL and traditional HDL models which are shipped with IP are generally reused.

B. Hard IP interface information in UPF format:

In the traditional Hierarchical UPF approach, the power intent is written in separate files. The division of the power intent is done at the IP boundaries so that we can use the standalone UPF file also. If there are multiple instances of the IP, then the IP UPF is needed to be instantiated or loaded using load_upf command wherever it is required followed by the connecting required supplies as per instance basis. If there are 100 instances of one IP then we have to write the load_upf command 100 times which increases the number of lines in the top UPF files and increases its verbosity.

Traditional Hierarchical UPF Example:

Soc.upf

```
create_power_domain soc_pd -include_scope
create_supply_net VCC
create_supply_net VSS
……..
Load_upf ip.upf -scope {I1}
Load_upf ip.upf -scope {I2}

Load_upf ip.upf -scope {I3}
……..
Connect_supply_net VCC -ports { \
I1/Vcc \
I2/Vcc \
I3/Vcc }
Connect_supply_net VSS -ports { \
I1/Vss \
I2/Vss \
I3/Vss }
```

hard_ip.upf

```
create_power_domain ip_pd
create_supply_port vcc -direction in -domain pd
create_supply_net vcc -domain pd
connect_supply_net vcc -ports vcc
……..
create_supply_port vss -direction in -domain pd
create_supply_net vss -domain pd
connect_supply_net vss -ports vss
……..
set_domain_supply_net pd ungated_rf - primary_power vcc -
primary_ground vss
```

As explained above, the SOC upf essentially loads all the SIP UPFs and Hard IP UPFs. In the current design flow, all the IP level UPFs are loaded in the partition and partition level UPFs are loaded in the Subsystem. Subsystems are loaded as HIPs at the SOC Level. In the SOC, there are multiple flows for Verification and Implementation.
In the functional verification, hard IP UPFs are read whereas in the design implementation, hard macros liberty files are read.

In case of power model, it is implemented differently.
Instead of writing load_upf command for each instance of IP, we can create a power model for that IP.upf using command begin_power_model and end_power_model and wherever the IP instance is needed to be loaded then we can use apply_power_model command (single line command) which loads the IP.upf for any number of instances which we can specify using *-elements* option.
Below is the example of how power model is implemented and used in the SOC world. In this paper, we have validated the power model across multiple flows- static checker, functional simulation, emulation flows and design implementation flows.

UPF with Power Model:

Soc.upf

```
begin_power_model macro_cell -for { c74p13rflfc1r2w184x128h }
create_power_domain pd_ungated_rf
create_supply_port vcc -direction in -domain pd_ungated_rf
create_supply_net vcc -domain pd_ungated_rf
connect_supply_net vcc -ports vcc
set_domain_supply_net pd_ungated_rf -primary_power_net vcc -primary_ground_net
vss
add_port_state vcc   -state "vcc_on_hv $VCC_SUPPLY_HV" -state "vcc_on_lv
$VCC_SUPPLY_LV" -state "vcc_ret $VCC_SUPPLY_RET"
……
……
create_pst pst_pd_rf   -supplies          "vcc                    vss"
add_pst_state hv -pst pst_pd_rf -state { vcc_on_hv  ps_VSS_0p0 }
add_pst_state lv -pst pst_pd_rf -state { vcc_on_lv  ps_VSS_0p0 }
add_pst_state all_off  -pst pst_pd_rf -state { vcc_ret  ps_VSS_0p0  }
end_power_model
```

hard_ip.upf

```
create_power_domain soc_pd  -include_scope
create_supply_net VCC
create_supply_net VSS
……..
apply_power_model macro_cell -elements { \
I1 \
I2 \
I3 \
}  -port_map { {I1/vcc VCC} {I2/vcc VCC} {I3/vcc VCC} {I1/vss VSS} {I2/vss VSS} {I3/vss VSS}  }

add_port_state "I1 /vcc" \
    -state "SIPPCIE2_CORE_PGD_ON_MIN   $SIPPCIE2_VCCSA_SUPPLY_MIN" \
    -state "SIPPCIE2_CORE_PGD_ON_MAX   $SIPPCIE2_VCCSA_SUPPLY_MAX" \
    -state "SIPPCIE2_CORE_PGD_OFF   off"
add_port_state "I2 /vcc" \
    -state "SIPPCIE2_CORE_PGD_ON_MIN $SIPPCIE2_VCCSA_SUPPLY_MIN" \
    -state "SIPPCIE2_CORE_PGD_ON_MAX $SIPPCIE2_VCCSA_SUPPLY_MAX" \
    -state "SIPPCIE2_CORE_PGD_OFF   off"

create_pst pt -supplies   " I1/vcc    I2/vcc "

add_pst_state s1 -pst pt -state { SIPPCIE2_CORE_PGD_ON_MIN SIPPCIE2_CORE_PGD_OFF  }
```

The power model command pair begin and end power model defines a boundary for the Hard IP. The created model here is named as macro_cell for the contents of Hard IP ***c74p13rflfc1r2w184x128h.***

D. Hard IP with Multiple instances:

Since the IP can have large number of instances, in the traditional Hierarchical UPF writing approach, a large number of load_upf command line need to be written in the upf file to load the IP upf. This increases UPF verbosity and probability of error while integrating the power intent of the IP. Using power model approach we can reduce the number of lines in upf file hence keeping power intent short and simple.

Hierarchical UPF	Modular UPF with Power Models
Loading 10 instances of c74p13rflfc1r2w216x128h.upf→ 10 load_upf commands→ load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_1 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_2 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_3 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_4 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_5 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_6 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_7 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_8 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_9 load_upf $ROOT_DIR/c74p13rflfc1r2w216x128h.upf -scope pxtss_rf_10 2 connect_supply_net commands→ connect_supply_net VCC -ports { pxtss_rf_1/vcc pxtss_rf_2/vcc pxtss_rf_3/vcc pxtss_rf_4/vcc pxtss_rf_5/vcc pxtss_rf_6/vcc pxtss_rf_7/vcc pxtss_rf_8/vcc pxtss_rf_9/vcc pxtss_rf_10/vcc } connect_supply_net VSS -ports { pxtss_rf_1/vss pxtss_rf_2/vss pxtss_rf_3/vss pxtss_rf_4/vss pxtss_rf_5/vss pxtss_rf_6/vss pxtss_rf_7/vss pxtss_rf_8/vss pxtss_rf_9/vss pxtss_rf_10/vss }	Creating power model for c74p13rflfc1r2w216x128h.upf → 1 apply_power_command→ apply_power_model c74p13rflfc1r2w216x128h -elements { pxtss_rf_1 pxtss_rf_2 pxtss_rf_3 pxtss_rf_4 pxtss_rf_5 pxtss_rf_6 pxtss_rf_7 pxtss_rf_8 pxtss_rf_9 pxtss_rf_10} l -port_map { {pxtss_rf_1/vcc VCC} {pxtss_rf_2/vcc VCC} {pxtss_rf_3/vcc VCC} {pxtss_rf_4/vcc VCC} ..... {pxtss_rf_1/vss VSS} {pxtss_rf_2/vss VSS} {pxtss_rf_3/vss VSS} {pxtss_rf_4/vss VSS} ... }

Figure 2. Comparison of HIP UPFs snippet in 2 approaches

This figure shows an IP.upf which is loaded 10 times for 10 different instances in Hierarchical approach. While in the other hand, instantiation of those 10 instances is done in single UPF command line.

We have considered Client SoC where there are more than 1189 IPs consisting of 384 HIPs and remaining SIPs for IPs, globals etc. We implemented power aware verification model for 1 of the Subsystem with 3 main partitions Controller, PHY and Fabric partition.

Block Type	Cell count	Number of power domains
Controller	46942	3
PHY	43169	4
Fabric partition	17960	3

Table I. Block Specifications

These HIP's having large number of instances have been compiled and simulated by using both the traditional hierarchical load_upf approach and modular_upf approach. There is improvement seen in the simulation time which is decreased by around 1\3rd of the original simulation time.

IV. KEY BENEFITS

The hierarchical power intent with UPF2.1 using power models for hard IPs were implemented for the next gen Client SoCs and HIP power models for hierarchical UPFs were validated for FE Static Checker Tool in the front-end design, functional verification for the power aware simulation and vendor emulation tool for the power aware emulation. We replaced all the import of Hard IP UPF and its interface connectivity

and converted into power models such that all the power models could be reused for each instantiation with great simplicity. Following are the advantages that we witnessed in the hierarchical low power design

- Easy to define the power interface information of IP by making its power model using power model commands i.e, *begin_power_model* and *end_power_model* commands. Single *apply_power_model* command will provide the power interface information to the top upf.
- Number of hand-written UPF lines is greatly reduced thereby eliminating the manual error in writing thousand lines of Hard IPs loading as well as interface connectivity.
- Front Tools flags to the point violation which eases debug ability while reducing noise in cleaning up the issues.
- Simulation time for the power aware simulation in vendor tool was reduced by 30%
- Power Aware simulation model for HIPs are more accurate as issues related to interface supplies and their related supply can be easily caught in Front Static Checker tool upfront

V. CONCLUSION

The hierarchical low power intent with power models comprising of power management interface and power management behavior enable more accurate design flow verification and implementation. This allows designer to build the UPF infrastructure in a much efficient manner wherein power models can be reused for many instantiations. This helps in easy readability and debug ability and it helps to catch complicated functional bugs early in the design flow thereby reducing the need for costly re-spins. In this paper, we have demonstrated how to build power models for hard IPs with much more concise UPF that can be used at front end static verification, low power simulations and power aware emulations.

REFERENCES

[1] IEEE Std 1801™-2013 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 29 May 2013

[2] DVCon 2018 Power Aware Models: Overcoming barriers in Power Aware Simulation, Mohit Jain, Amit Singh, J.S.S.S Bharath, STMicroelectronics, Amit Shrivastava, Bharathi Jain, Synopsys

[3] DVCon 2018 Hybrid Flow: A smart methodology to migrate from traditional Low Power Methodology, Rohit Kumar Sinha, Prashanth N