Golden UPF : Preserving Power Intent From RTL to Implementation

Himanshu Bhatt, Harsh Chilwal

Synopsys

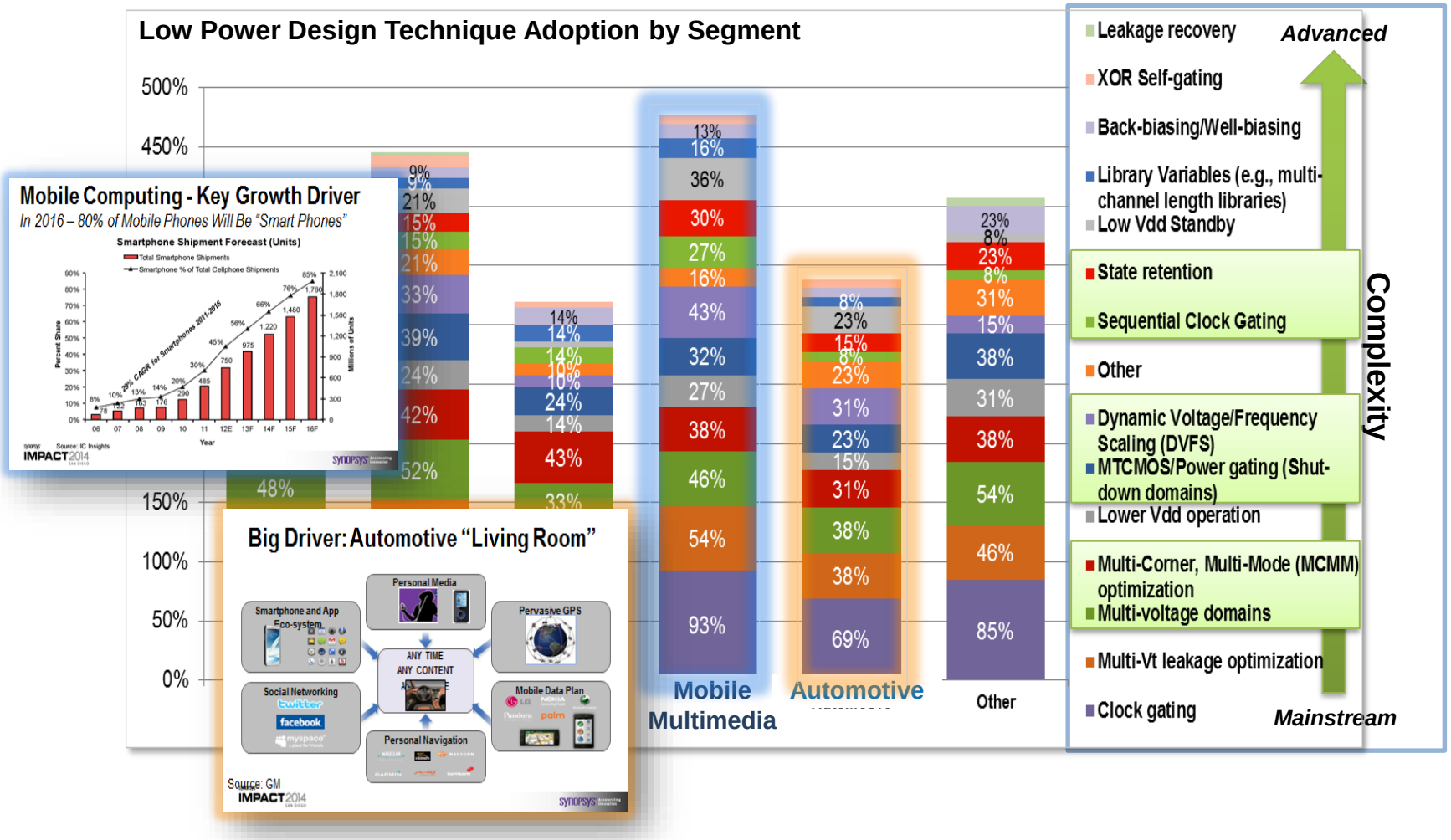
accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Agenda

- Overview of Power Aware Verification
- Why Golden UPF
- Golden UPF flow in LP simulator
- Golden UPF flow across other tools

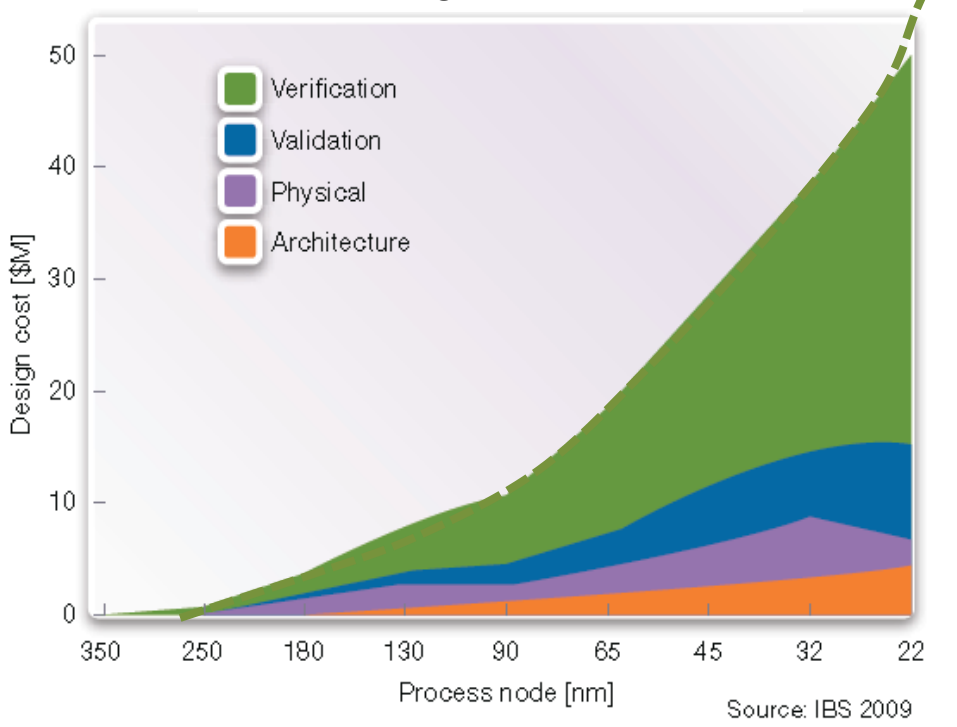
Low Power is Everywhere



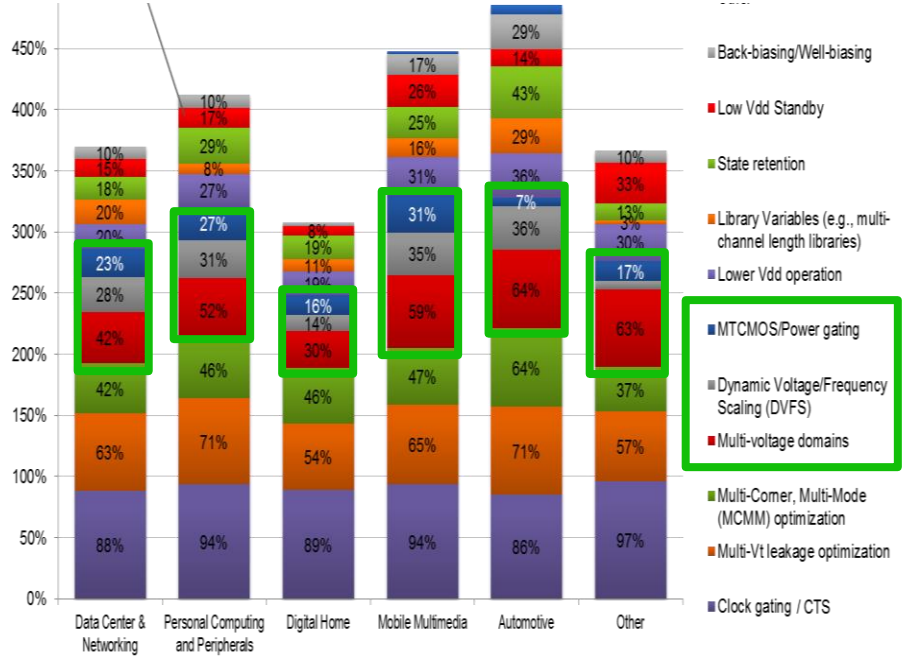
Mobile SoCs Driving Low Power Complexity Growth

Low Power Complexity Growth is Driving Design Cost

Overall Design Cost Breakdown



LP Complexity Growing Across All Segments



Source: 2011-2012 Global SNUG surveys

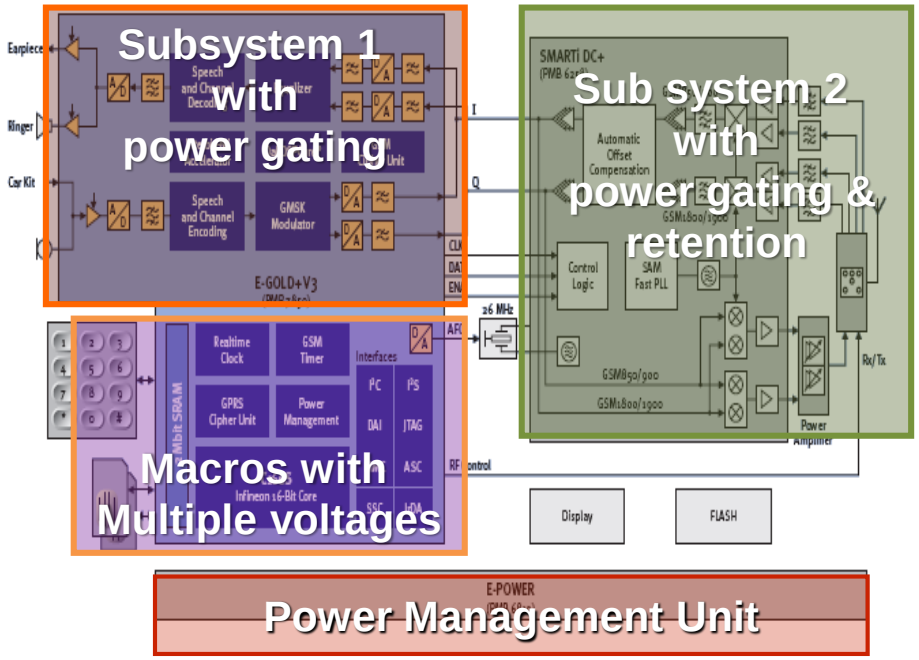
Low power adoption causes huge increases in verification complexity

Low Power Verification Complexity

Overview of Challenges

Verification challenges of Low Power (LP) design using IEEE 1801 (UPF)

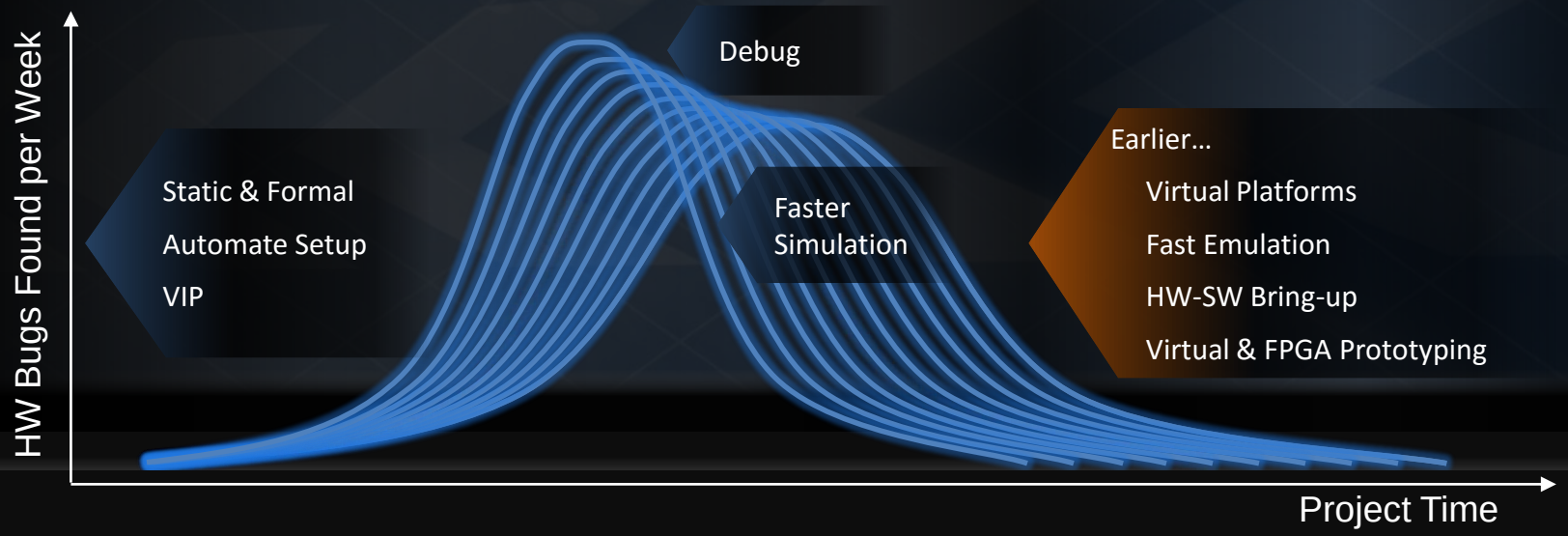
- Accurate Power Aware simulation of LP states (shutdown, standby, retention…) and low power cells
- Coverage of new LP states at sub system and system level
- Protocol checks for transitions in and out of low power states
- Debug of low power verification



Source: 2011-2012 Global SNUG surveys

Advanced Low Power Design Causes Huge Increase in Verification Complexity

Must ‘Shift-Left’ in Context of Low Power!



LP Static Checking, Power-Aware CDC, Lint, Formal, Simulation, Debug and Emulation

Introduction

- GUPF file refers to the original (RTL) UPF file that is provided by user together with their (RTL) netlist
- In current UPF flow
  - GUPF is updated in implementation tools along with netlist changes. In addition to the netlist output, implementation tools output the updated UPF file which is in sync with the changed netlist
  - The two outputs are passed to downstream UPF tools
  - The updated UPF file could be very different with GUPF file, in the following ways:
    - In updated UPF:
      - It reflects corresponding netlist object name changes, object deletion and object addition
      - Wildcards are expanded
      - UPF comments are lost
      - Tool specific condition statements which are not executed by implementation are lost
      - Tool derived UPF power intent is added

Why Golden UPF

- The Golden UPF flow offers the following advantages:
  - Golden UPF file remains unchanged throughout the flow, which keeps the form, structure, comment lines, and wildcard naming used in the UPF file as originally written at RTL stage
  - Golden UPF preserves user comments
  - User can use tool-specific conditional statements to perform different tasks in different tools. Such statements are lost in the traditional UPF-prime flow
  - Changes to the power intent are easily tracked in the supplemental UPF file

Why Golden UPF (contd.)

- User wants to be able to source RTL UPF at different points in the flow
  - RTL UPF is structured
  - More concise and easier to read
  - Has user comments
  - Has conditional sections not evaluated by implementation tools
  - Has wildcards and find/query operations

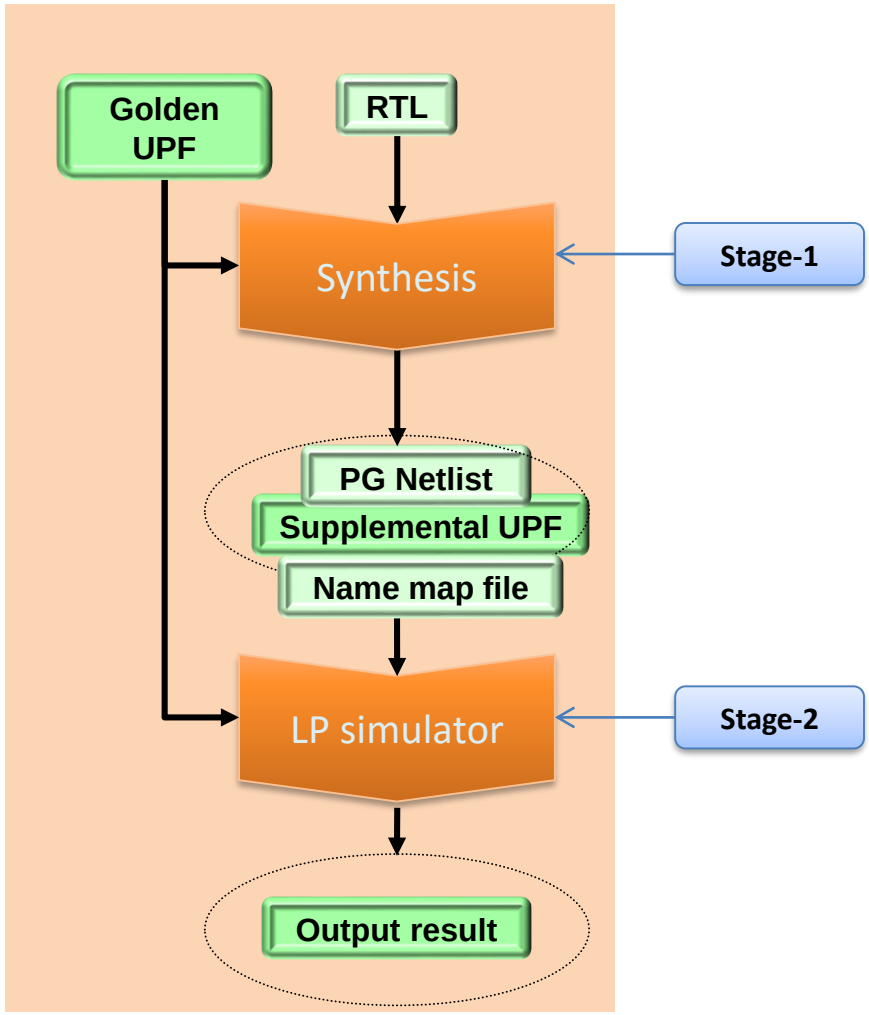
Original (RTL) UPF

```
# comment
…
If ( SIM ) {
…
}
# comment
create_power_domain TOP
create_power_domain PD1 –element U*
…
```

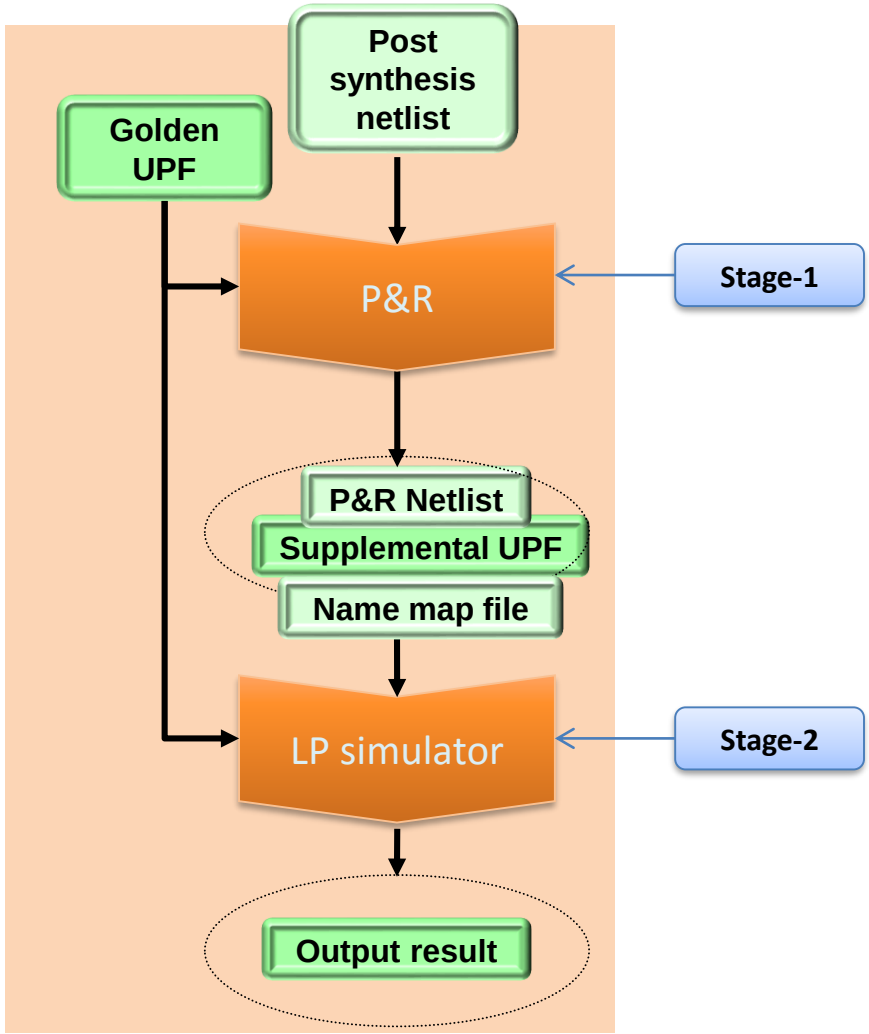
UPF'

```
…
create_power_domain TOP
create_power_domain PD1 –elements \
  {U1 U2 U3}
…
```

PA-GLS used by LP simulator (post synthesis)



PA-GLS used by LP simulator (post P&R)



Golden UPF Flow

- STAGE- 1 :
  - Give RTL + UPF to synthesis
- Example synthesis commands:

```
set enable_golden_upf true
analyze -format verilog rtl_design.v
load_upf Golden_upf.upf
write -f verilog -hierarchy -pg -output NETLIST.v
save_upf -supplemental supplemental_upf_dc.upf -include_supply_exceptions
```

- Synthesis tool will spit it out Netlist design and Supplemental UPF

Golden UPF Flow (contd.)

Name map file content

- One challenge of the Golden UPF reapplication on post implementation netlist is object name changes
- Existing netlist object names can be changed by implementation tools when certain netlist transformation happens
- If the original object name is referred by Golden UPF file and the original name got changed, in Golden UPF reapplication, UPF tools will mostly not be able to find the netlist object based on the original name

Example :

```
create_power_domain TOP
create_power_domain PD1 –element U1/U2
set_retention ret_TOP –domain TOP –elements {A_reg}
```

  - One of the common default name change rule is: “Hierarchical separator: {/ _}”
  - The Golden UPF file has:
    During netlist changes in some synthesis tool, U1/U2 is ungrouped, and its name becomes U1_U2. In Golden UPF file reapplication, all UPF tools will find U1_U2 when U1/U2 is queried if they all apply this naming rule that synthesis tool used.

Golden UPF Flow (contd.)

Supplemental UPF content

- During netlist implementation, implementation tools may derive new UPF power intent:
  - New –no_isolation isolation strategy on newly created ports along UPF control signal paths
  - New root cell addition for existing power domain during physical always on synthesis
  - Exception power connection on UPF PM level shifter and always on cells.

```
set_isolation snps_no_iso__0 -domain core1_pd -elements I_CORE1_I_CORE11/iso -no_isolation
connect_supply_net top_vdd -ports { core22_in1_1__UPF_LS/VDD }
create_power_domain core1_pd -elements {i_core1_i_core11/U3}  -update
```

Golden UPF Use Model - Simulation

- LP simulator must have the below inputs
  - NETLIST design
  - power.config file having:
    - Name mapper file
    - Golden UPF file
    - Supplemental UPF file
- Example :
  - **vcs** testbench.v vclp_tf03.v -sverilog -v litmus/vcst_xt_db2vlg.v -timescale=1ns/1ns -mvsim=incr -nopowerelab +define+UPF_1_0 ",
  - **powerelab** -power_config power.config

Golden UPF Use Model – Simulation (contd.)

power.config file content

- Power config file contain below commands:
  - Library path for Cell which is used in Netlist
  - DB file Names at available at Library path
  - UPF Name mapper File
  - read_upf command with “Golden upf” and “supplemental upf”

- Example :
  - set search_path "./ /remote/arch-proj/testcases/pv_dbs/harsha/einfochips/mapping_files/lib/stdcell_hvt/db_nldm /remote/arch-proj/testcases/pv_dbs/harsha/einfochips/mapping_files/lib/stdcell_rvt/db_nldm /remote/arch-proj/testcases/pv_dbs/harsha/einfochips/mapping_files/lib/stdcell_lvt/db_nldm "
  - set link_library " saed90nm_max.db saed90nm_max_lvt.db saed90nm_max_hvt.db saed90nm_max_hth.db saed90nm_max_hthn_lsh.db saed90nm_max_hth_rd.db saed90nm_max_hth_rdr.db saed90nm_max_hth_rdsr.db "
  - set upf_name_map {{**top vclp_tf03.v.nmf**}}
  - read_upf vclp_tf03.upf -supplemental **tf3_supplemental.upf** -scope top -strict_check false -target icc_netlist

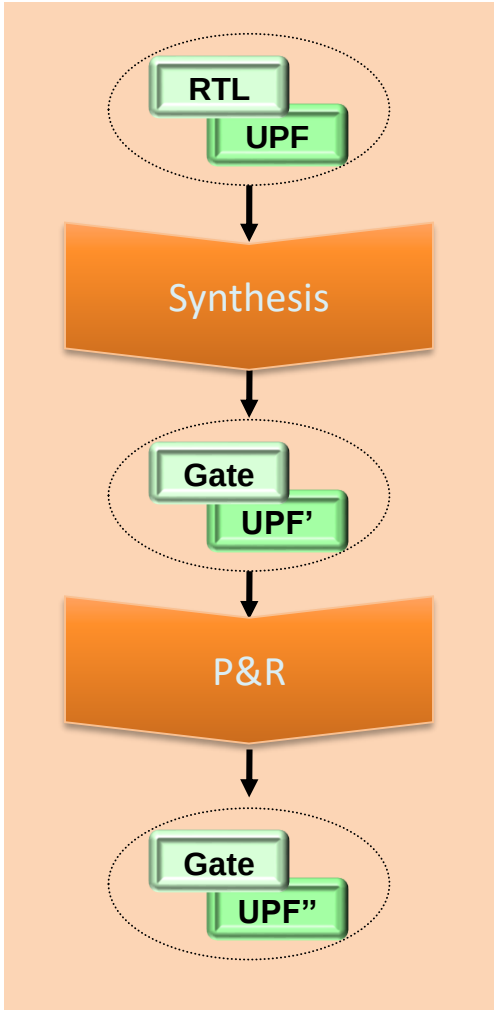
Golden UPF Use Model – Simulation (contd.)

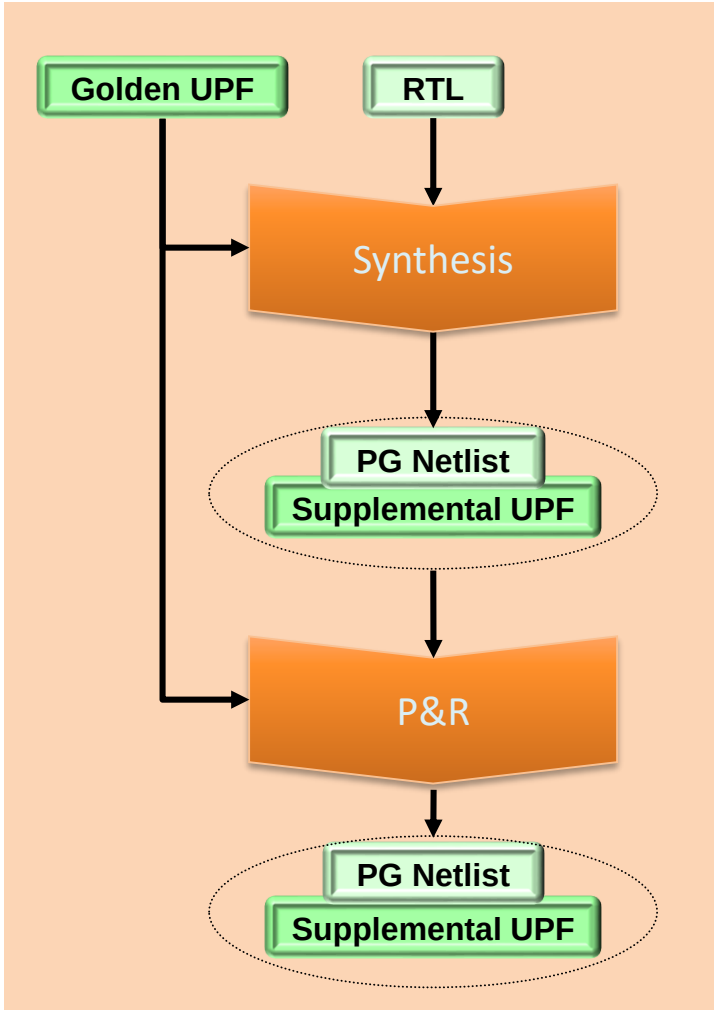
Name map file

- Content :
  - **Example :**

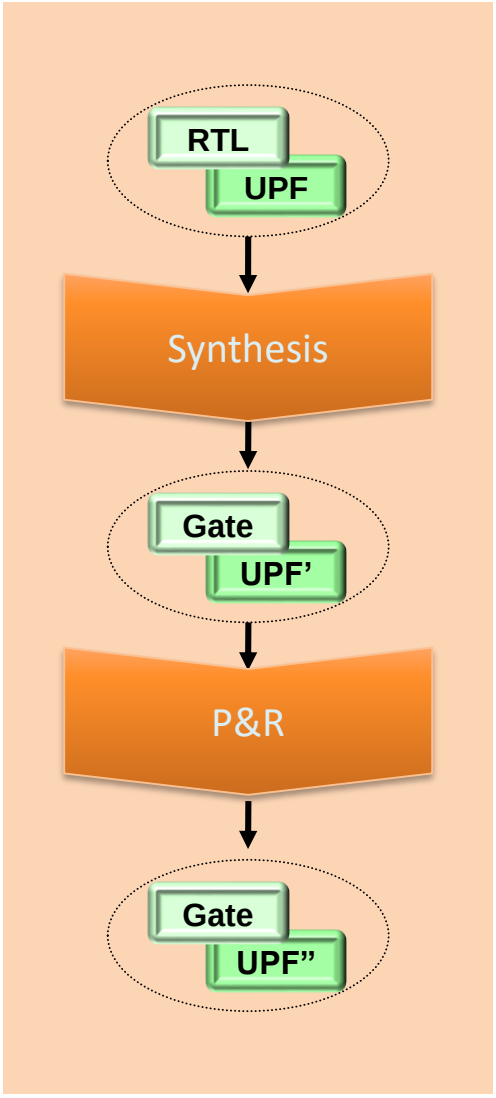
```
################################################################################
# Golden UPF name map file
#  Design    : top
#  Created By :  Design Compiler
#  Version   : J-2014.09-BETA3
#  Date      : Tue Aug 19 03:20:21 2014
################################################################################
set upf_name_map_version  v1.0
define_name_maps \
   -application golden_upf \
   -design top \
   -columns {class pattern options names} \
   [list cell c1 [list] [list blah]] \
   [list pin c1/counter* [list] [list {blah/counter[0]} {blah/counter[1]} {blah/counter[2]} {blah/counter[3]} {blah/counter[4]}
{blah/counter[5]} {blah/counter[6]} {blah/counter[7]} {blah/counter[8]} {blah/counter[9]}]] \
   [list cell c1/counter* [list] [list blah/counter_reg_0_ blah/counter_reg_1_ blah/counter_reg_2_ blah/counter_reg_3_
blah/counter_reg_4_ blah/counter_reg_5_ blah/counter_reg_6_ blah/counter_reg_7_ blah/counter_reg_8_ blah/counter_reg_9_]] \
   [list pin c1/nreset [list] [list blah/nreset]]
```

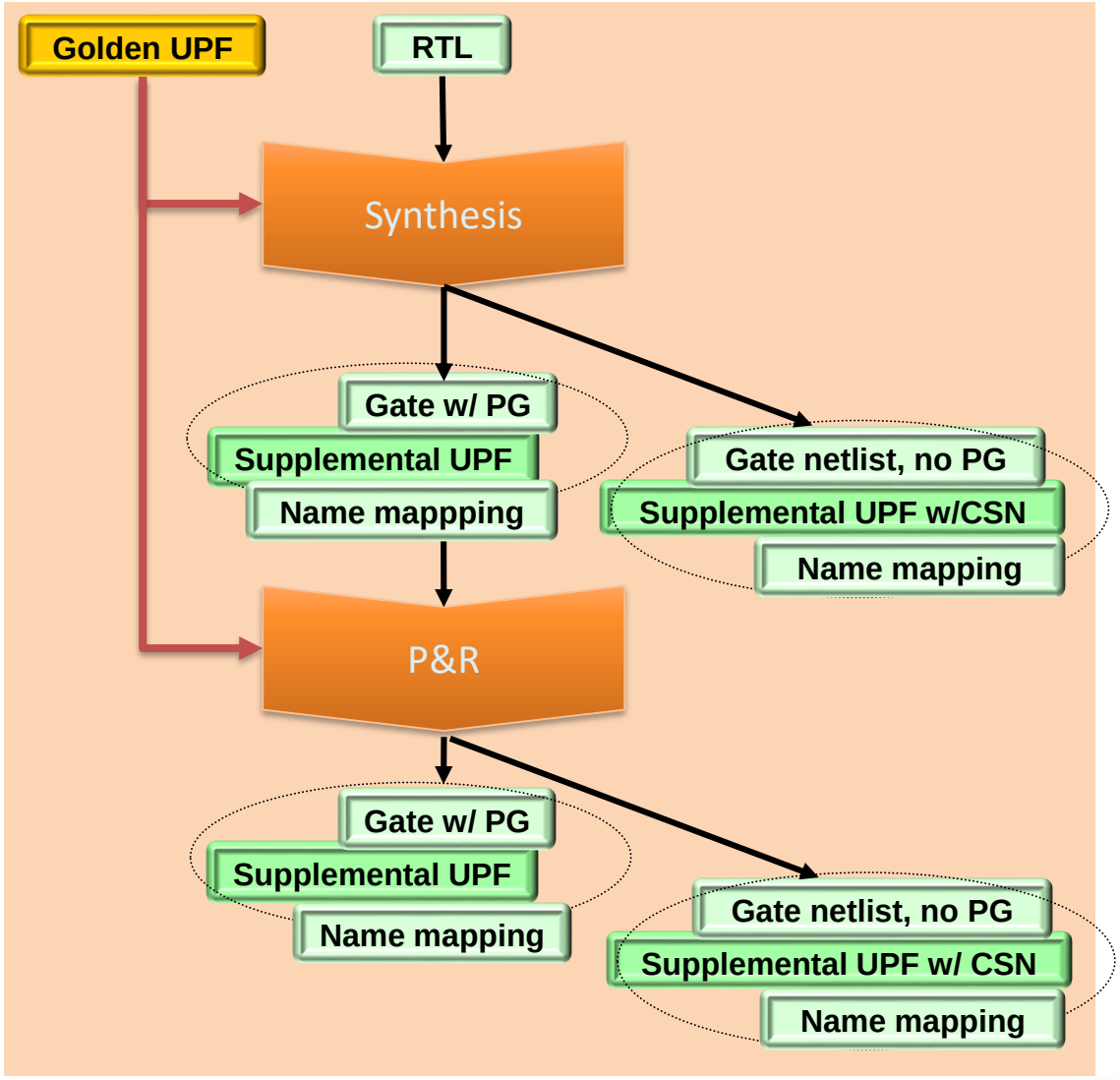
Existing UPF Flow vs. Golden UPF Flow



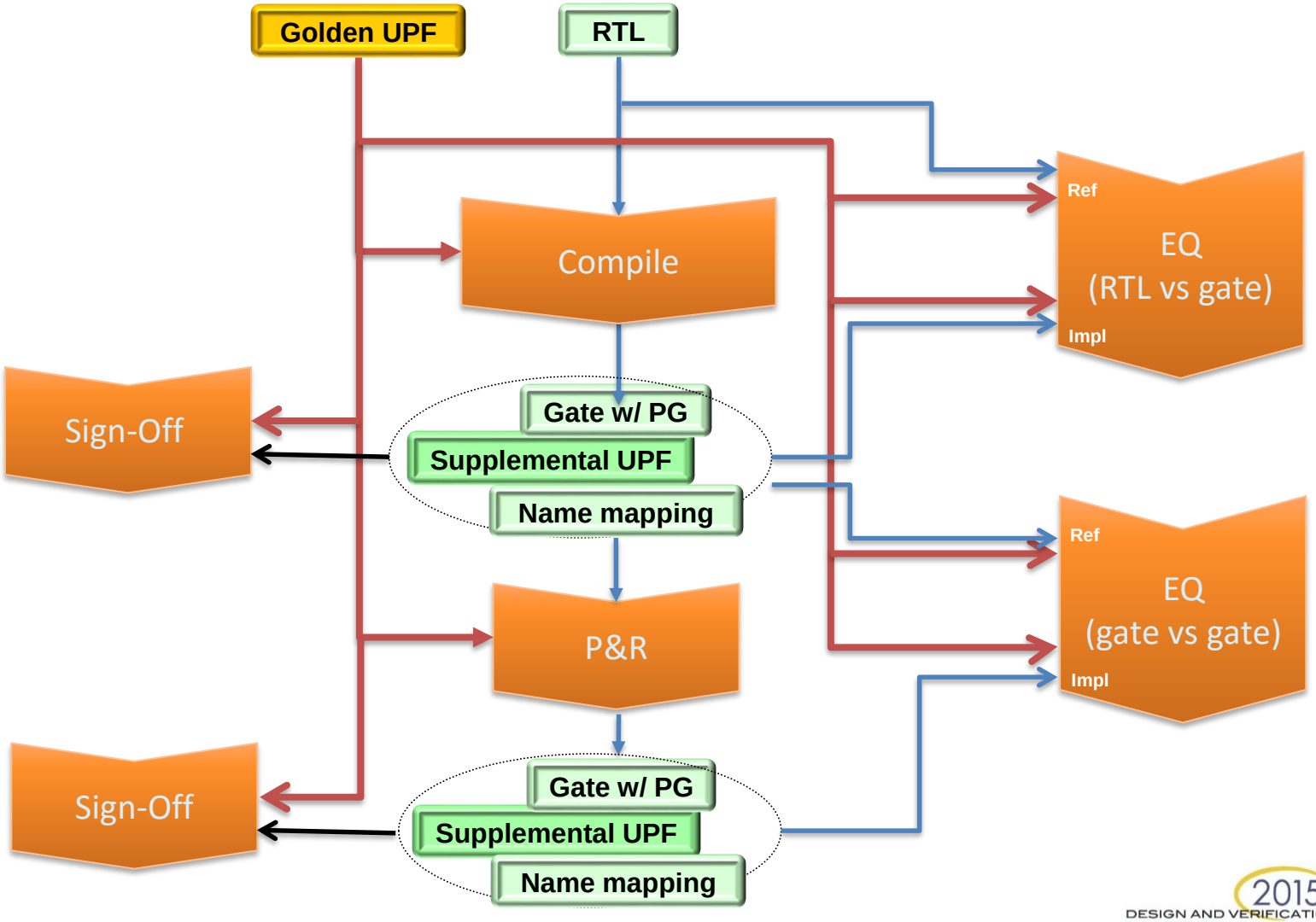


Existing UPF Flow vs. Golden UPF Flow

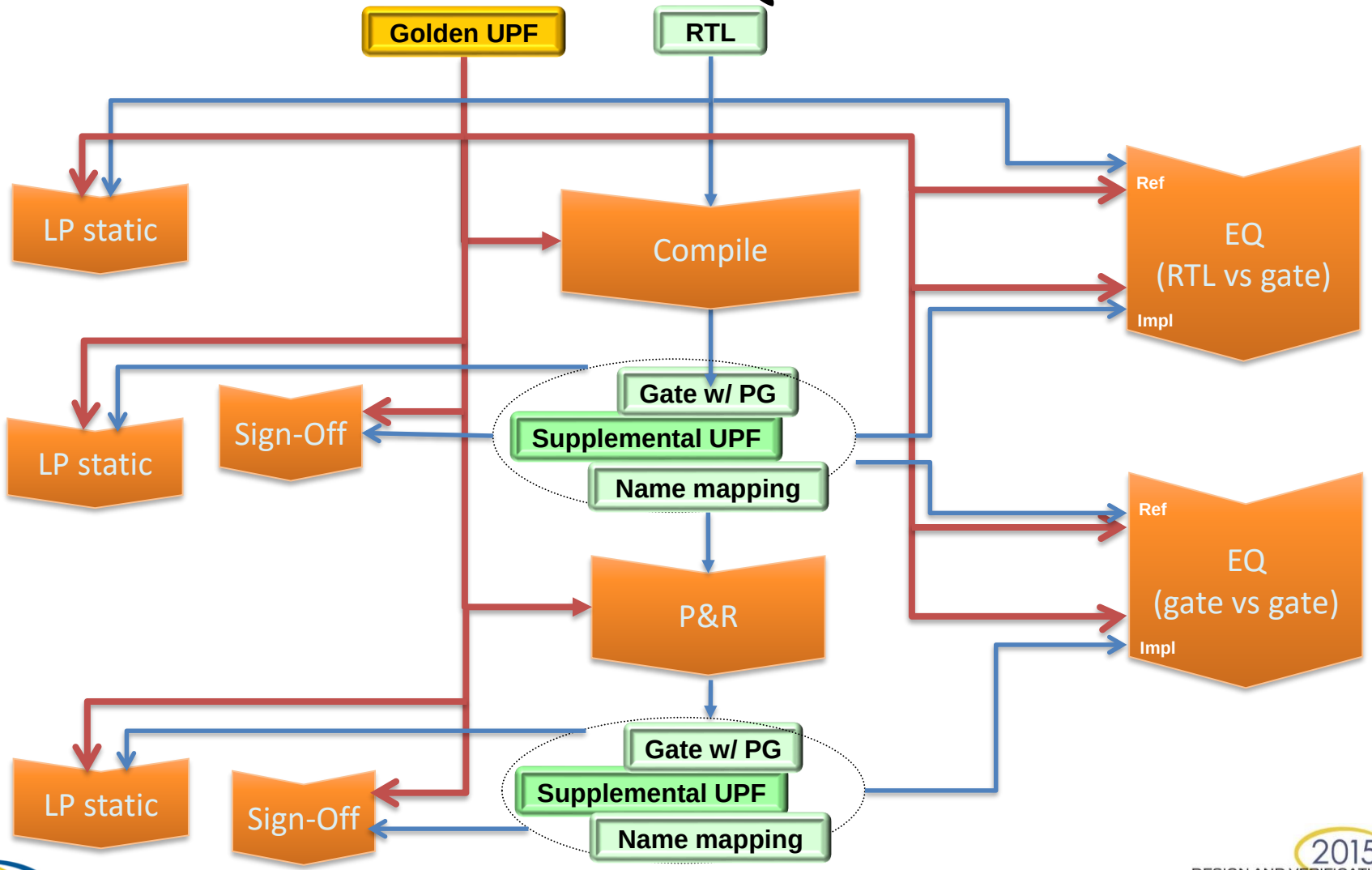




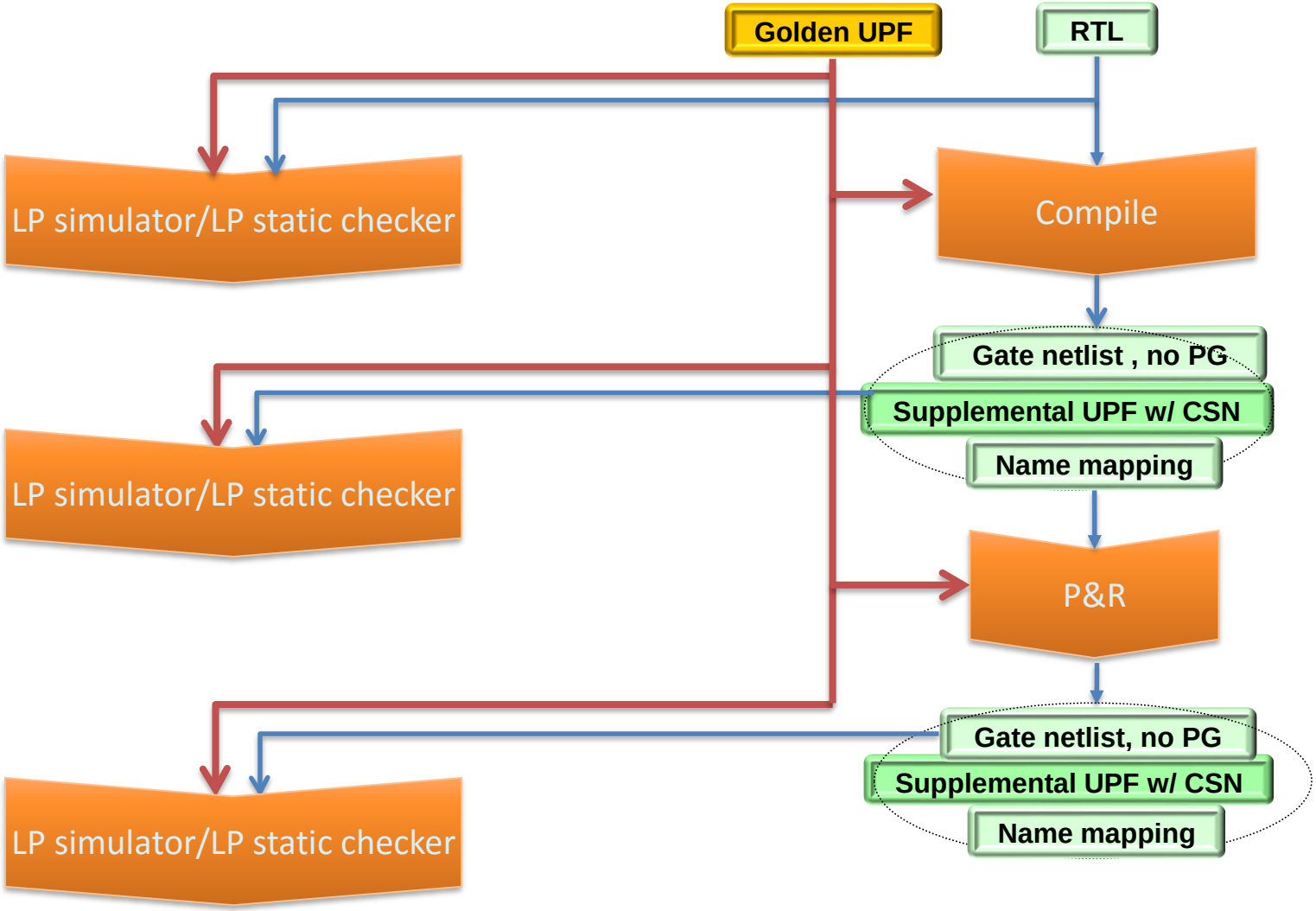
Golden UPF Flow: Sign-Off Analysis and EQ



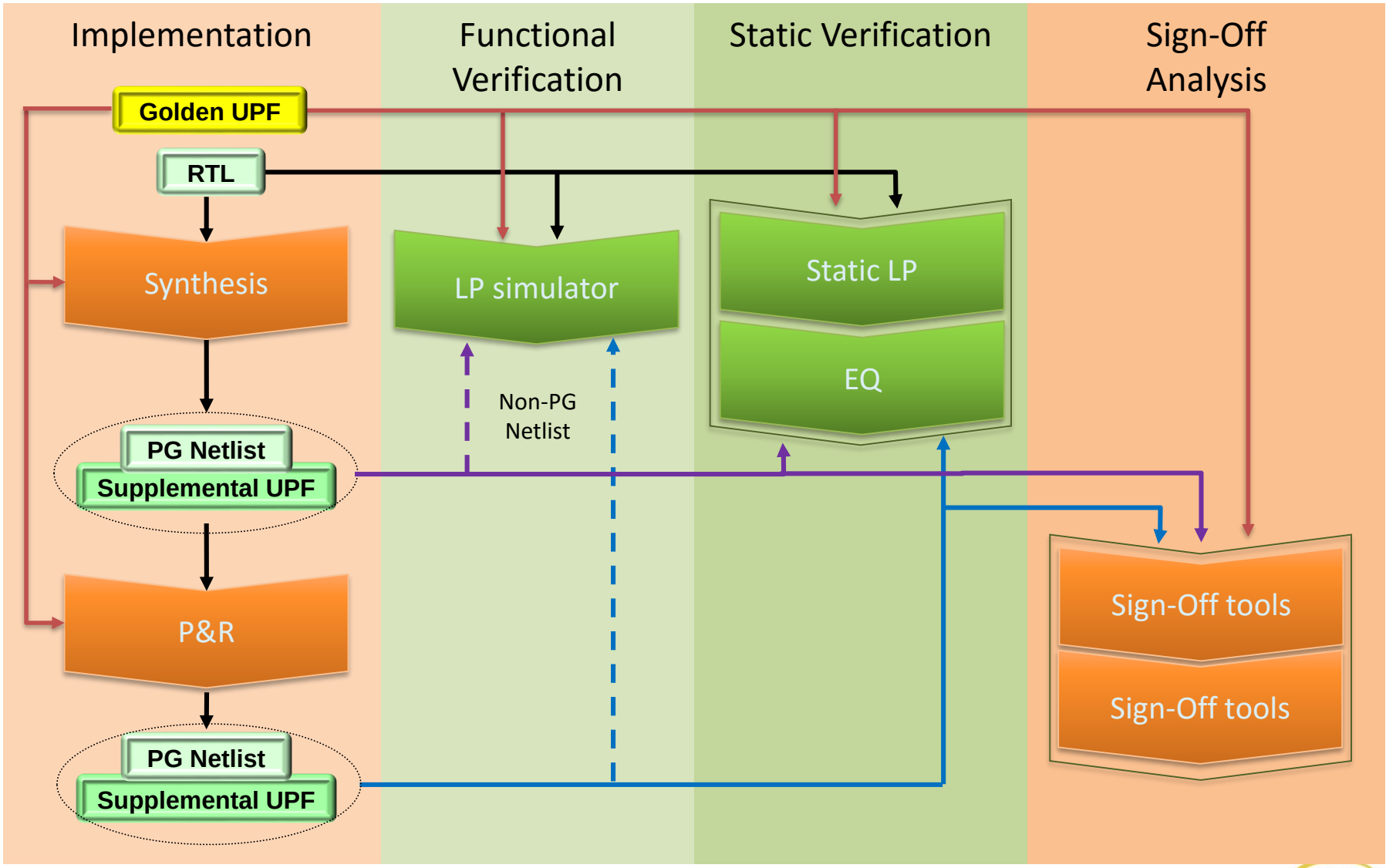
Golden UPF Flow: Sign-Off Analysis and EQ



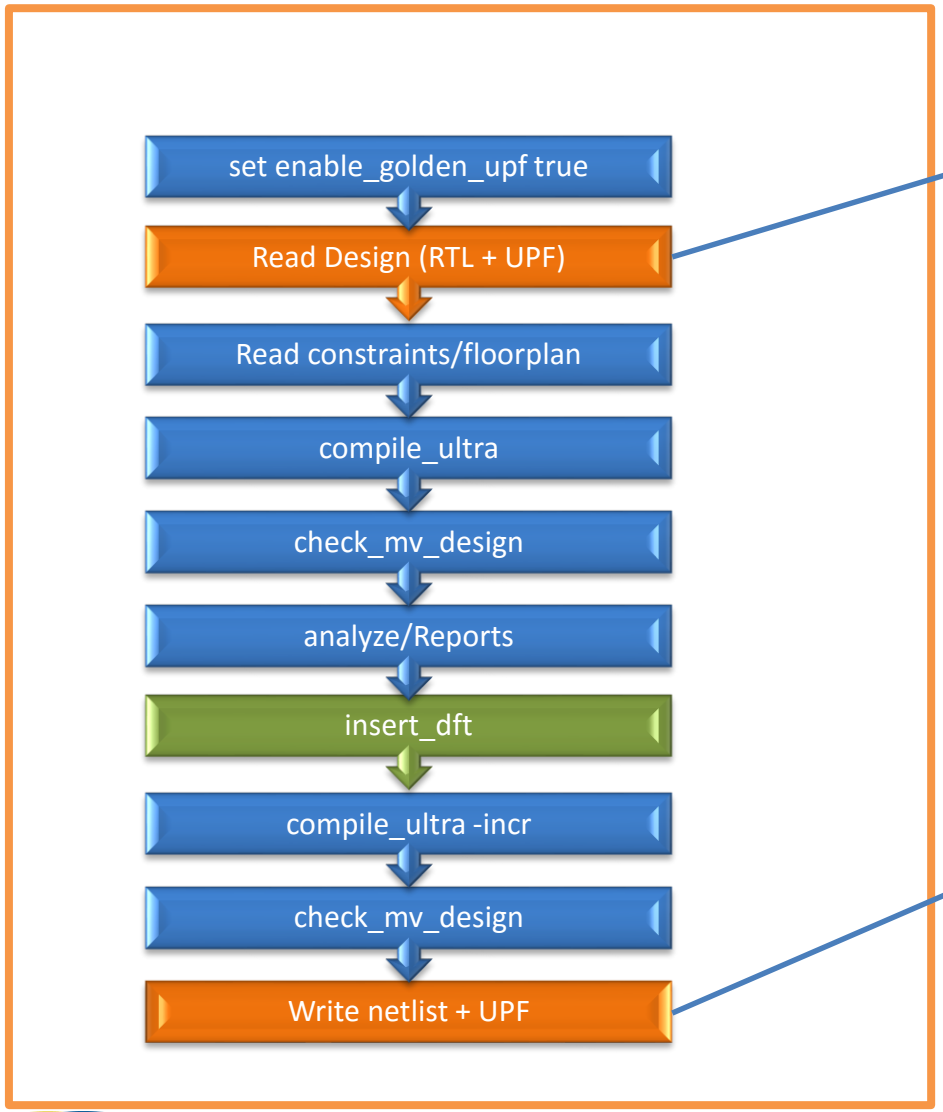
Golden UPF Flow: Verification Tools



Golden UPF Advanced Low Power Flow



RTL flows with Golden UPF



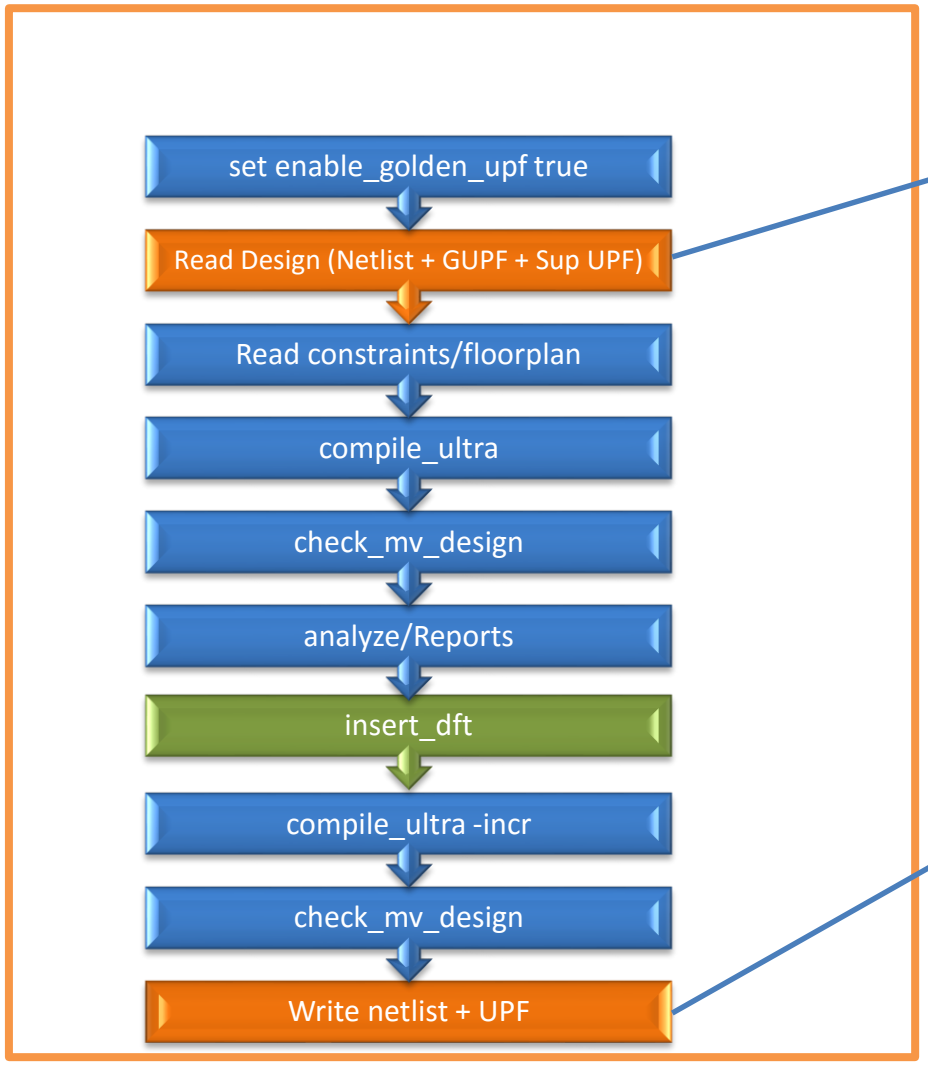
Inputs:
Design : RTL or RTLPG
UPF: Golden UPF

UI:

```
load_upf upf_file
-strict_check true
```

Outputs:
Netlist: Verilog or Verilog (PG) or DDC
UPF: Supplemental UPF or UPF'

GL netlist flows with Golden UPF



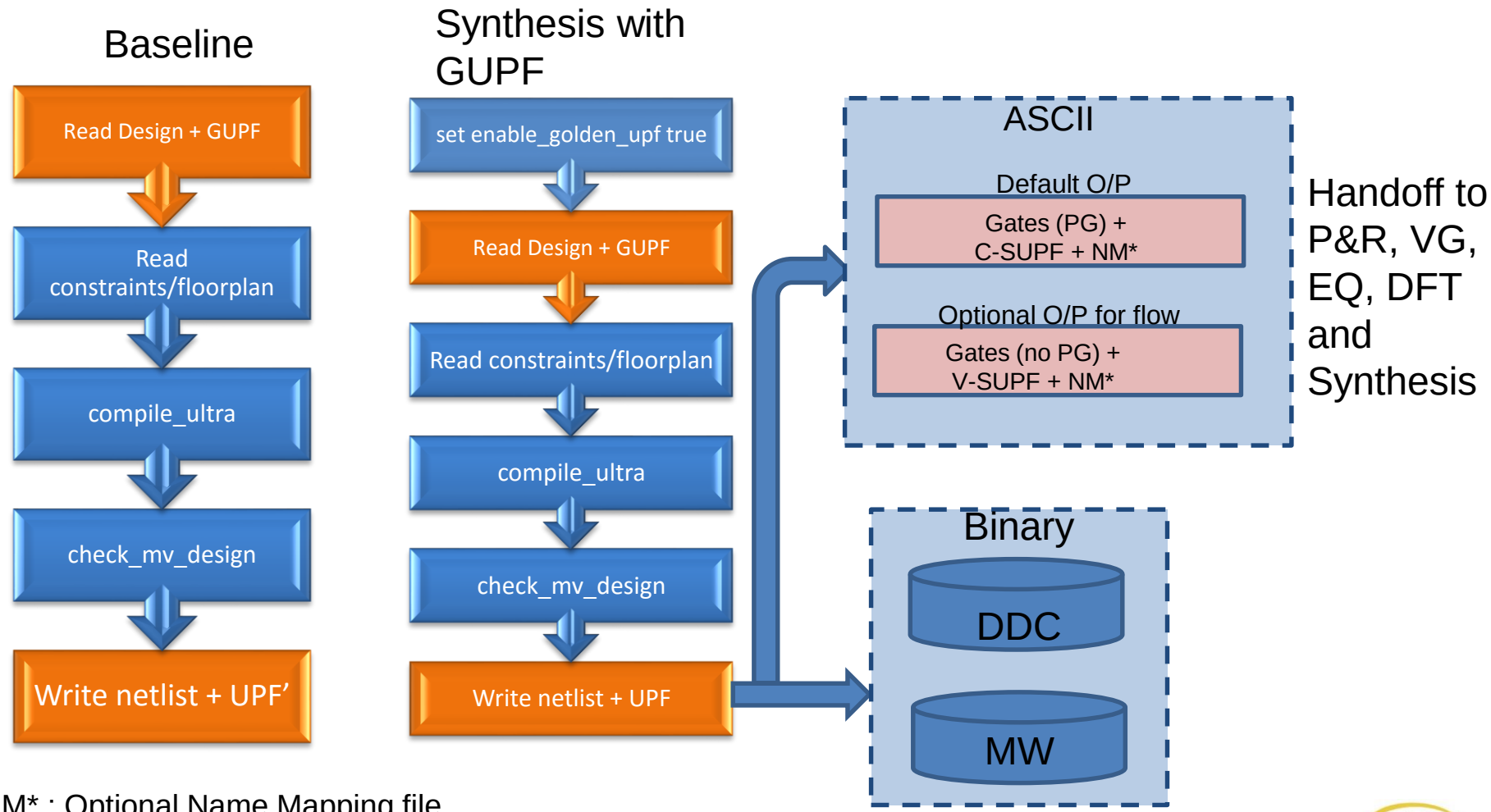
Inputs:
Netlist : Gates or Gates (PG)
UPF: Golden UPF + Sup UPF

UI:

```
load_upf upf_file
-strict_check false
 [-supplemental
<supplemental_upf>]
```

Outputs:
Netlist: Verilog or Verilog (PG) or DDC or Milkyway
UPF: Supplemental UPF or UPF'

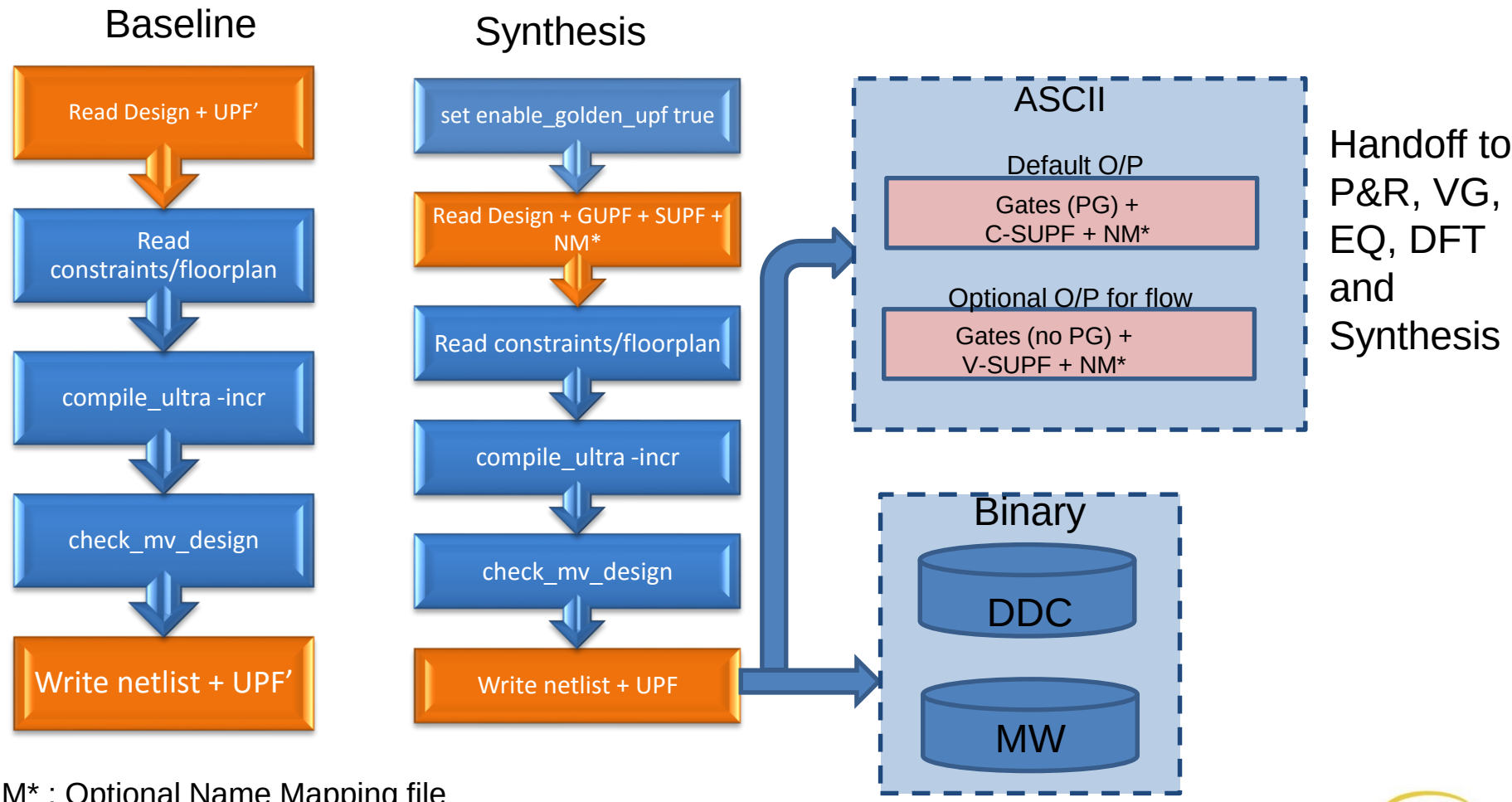
RTL + GUPF Validation flows (RTL)



NM* : Optional Name Mapping file

Validation flows (Gate netlist)

*Gates + GUPF + SUPF + NM**



NM* : Optional Name Mapping file

Questions