Low-Power Verification Methodology using UPF Query functions and Bind checkers

Madhur Bhargava, Durgesh Prasad

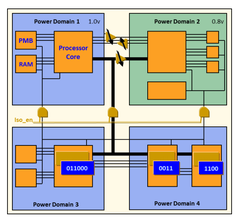
Mentor Graphics

accellera SYSTEMS INITIATIVE™

2014 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

INTRODUCTION

- Electronic devices have become complex and energy aware
- Require sophisticated power management architectures and strategies
  - If not applied properly, will affect design functionality
  - Complex protocols, many power modes need to be verified
- Need for advanced & efficient power aware verification
  - Catch low-power bugs at early stages and save design cycles
  - Power intent specification format (UPF) is used to define power management logic – without modifying HDL



Low-Power Verification

- Static Verification
  - **Catch all structural errors** – correct placement and connections of PA Cells
- Dynamic Verification
  - **Protocol checking –** Eg. Check PA Cells are enabled and active at the required time
  - **Power intent checking** – Verify power intent specified in UPF against implemented logic.
  - **Power intent coverage** – Check if all coverage goals are met for all power aware objects

Assertion Based Low-Power Verification

(Using System Verilog Assertions)

- Validate power control logic sequence
- Monitor and check transitions in power control signals
- Provide Coverage Data

EDA Vendors (Tool-Generated assertions) Common Protocol Errors

Custom Low-Power Assertions (Ensure all test scenarios are covered)

MOTIVATION FOR METHODOLOGY

- Tool-generated assertions used for low-power verification
  - Some specific scenarios may not be met
  - New protocols may require new assertions not yet supported by any tool

→ Custom Assertions & Coverage Items

- Custom Assertions & Coverage Items: Club in Checker Module and instantiate into design using UPF command "bind_checker"
  - Low-Power assertions require access of power objects – only present in UPF and not HDL (RTL Stage)
  - Some property checking require design/power control signals spanning across multiple domains.
  - Scope/Inputs of checker module defined in UPF – any change in UPF can break these assertions

→ Model assertions (Immune from change in UPF) ↓ Need for methodology

UPF CONCEPTS & COMMANDS REQUIRED IN METHODOLOGY

bind_checker	query_*	find_objects
Instantiate checker module (having low-power assertions) into design hierarchy without modifying the design code. Allows one to one port mapping of the checker module to actual power object/control signal.	(query_isolation, query_retention, query_power_domain) Search and get handle of power management object including strategies (isolation/retention/power switch), power domains, supply nets, supply ports.	Allows to query the design (HDL) elements. Provides good deal of filtering support to extract relevant elements.

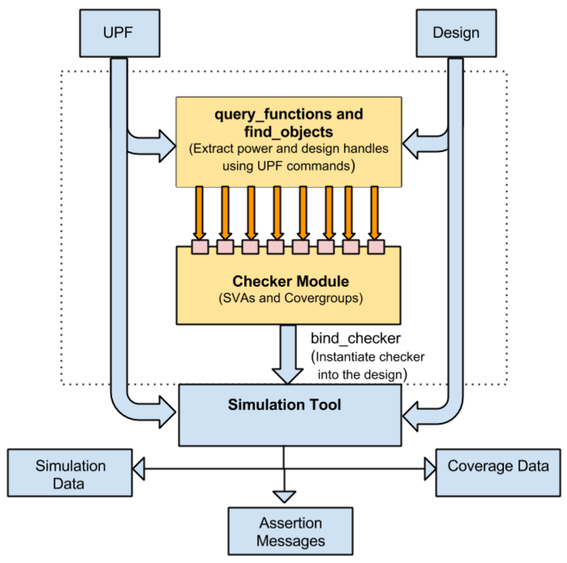
METHODOLOGY

Assertions and covergroups needed for low-power verification are packed in a checker module. This module is then instantiated in the design scope using UPF command "bind_checker". Since these SVAs and covergroups reside inside checker module, the required design/power signals need to be passed as actual argument to the checker module. This is achieved by using UPF commands query functions and bind checkers.

- Failing Assertions – Indicate functional issue or a low-power bug
- Coverage Data – Help achieve verification closure

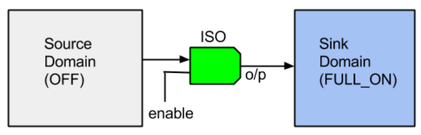
Steps for Methodology

- Model protocols/power intent to be checked
  - Use SystemVerilog assertions and club them together in a checker module
- Define interface based on UPF commands
  - Extract out required power/control signals (UPF objects) from power architecture using UPF query_* commands. Extract out design signals (HDL objects) using UPF command find_objects.
  - Pass on above handles to checker module and instantiate it in the design with help of UPF command "bind_checker".



CASE STUDY

Isolation Protocol Checking – Whenever the driving logic supply is switched off while the receiving logic supply is still ON, an isolation cell is required. One of thing to be verified is that the **output port (op) is clamped to golden expected value throughout the duration isolation enable is asserted.**



Step1 - Write checker module

- Above check can be expressed in the form of SVA which is written inside a checker module

Step2 – Define interface

- Checker module requires the handle of isolated signals, isolation_enable, clk and parameter values.
- Extract these handles from power architecture (using query_*) and pass these as actual to the formal port names in the checker module.
- Attach the checker module to the design using the bind_checker command.

Checker module:

```
module checker_isolation(input op, iso_en, clk) ;
   parameter int clamp_value      = 1 ;
   parameter isolated_signal_name = "" ;
   parameter iso_strategy_name    = "";
   always@(posedge clk)
     if(iso_en)
       assert (op == clamp_value) else $error("isolated signal '%s'
for isolation     strategy '%s' is not clamped(%b) correctly",
isolated_signal_name, iso_strategy_name, clamp_value);
endmodule
```

Tcl Proc:

```
proc chk_isolation_properties {
foreach domain [query_power_domain *] {
   foreach isolation [query_isolation * -domain $domain] {
      array set Iso_Strat[query_isolation * -domain $domain]
      foreach iso_sig $Iso_Strat(elements) {
        bind_checker
chk_$Iso_Strat(isolation_name)_$domain(domain_name)\
          -module checker_isolation
          -ports [list \
               [list op $iso_sig] \
               [list iso_en $Iso_Strat(isolation_signal)]\
               [list clk clk]\
               ] \
           -parameters [list \
                    [list clamp_value $Iso_Strat(clamp_value)] ...]
      }
   }
}
```

ADVANTAGES OF METHODOLOGY

Immune to change in UPF	• As query_*, bind_checker commands used in methodology are UPF commands, so any change in UPF will automatically reflect on output of these commands and will be fed to bind_checker
Highly programmable and easy to use	• Based on Tcl, so can be embedded in UPF file to automate the verification process
Access to all power objects and design signals	• As it relies on method to query and extract information from UPF

EXTENTIONS REQUIRED IN UPF CONCEPTS/COMMANDS

query_* commands

Extend these commands to return the object handle as the **full hierarchical path of the object referenced from the design top** - methodology require the handle of power/control signals needs to be the full hierarchical path.

Some **query_* command need to be extended to provide additional information** which is not as defined, per the UPF LRM. For example, query_power_domain needs to be extended to provide the primary supply (power/ground) of the queried power domain.

bind_checker command

Certain SVAs and covergroups require object name or constant values to give intuitive messages, which are passed on as parameters to the checker module. **Extension of "bind_checker -parameters" needs to be added.**

Support for expressions in -ports in bind_checker: In certain cases, the actual port of checker module can be an expression composed of power objects extracted from power architecture. For example, save_condition/restore_condition of set_retention.

CONCLUSION

- SystemVerilog assertions and Cover groups
  - used to achieve the verification closure of low-power designs
- Some EDA vendors provides fixed set of low-power assertions and coverage
  - Still a need for custom low-power assertions and coverage items
- UPF command bind_checker can be used
  - Standalone usage doesn't provide a strong way of writing custom assertions and cover groups
- Suggested methodology using bind_checkers, query commands and find_objects
  - Allows to write some of the very powerful low power assertion having considerable immunity from any change in the UPF or the design
  - The kind of flexibility the methodology provides in writing assertions and covergroup would be a leap forward in low-power verification.

Note: The purpose of the query_* command in the methodology is just to extract out the handles of power/control signals from the power architecture. However any other way apart from query_* commands can also be used in the methodology to extract out the same information.

In fact UPF 2.0 query function definitions were somewhat ambiguous and they have been moved to an appendix in UPF 2.1. The P1801 working group is working on an information model and API that will serve as the basis for a new set of query functions in UPF 3.0.

REFERENCES

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

Rudra Mukherjee, Amit Srivastava, Stephen Bailey: "Static and Formal Verification of Low Power Designs at RTL using UPF", DVCon 2008.

Author's Contact Information

Name: Madhur Bhargava
Email: madhur_bhargava@mentor.com
Company: Mentor Graphics

Name: Durgesh Prasad
Email: durgesh_prasad@mentor.com
Company: Mentor Graphics