UVM-based extended Low Power Library package with Low Power Multi-Core Architectures

Avnita Pal, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (avnita@siliconinterfaces.com)
Priyanka Gharat, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (priyanka@siliconinterfaces.com)

Abstract—This paper explores integration of Unified Power Format (UPF) and Universal Verification Methodology (UVM) to simplify the functional verification and power management process. Specifically, focuses on incorporating low power design features within multi-Core architectures using in-built low power routines in Assembly Language (ASM). The proposed Low Power UVM Package contains classes for SOC environment like Devices, Buses, and Memory, which can implement low power strategies using SystemVerilog and DPI extension within proposed Low Power UVM Package during the Run Phase of UVM Agents. The aim is to enable efficient power management and bridge the gap between functional verification and power management engineers.

Keywords—Power Management, Low Power, UVM (Universal Verification Methodology), Functional Verification, SystemVerilog, Unified Power Format (UPF) DPI (Direct Programming Interface), PowerUP, PowerDown, Assembly Language (ASM), SoC (System on Chip).

I. INTRODUCTION

The current approach of incorporating Power Architecture after Functional Verification is not optimal. It should be integrated into the strategy from the beginning, along with Methodologies based Functional Verification and Coverage, and Low Power Implementation. In previous works, a single platform, such as UVM, was used to develop library components for devices, including low power strategies, Functional Verification Methodology, and UPF-based Low Power Architecture. This allowed designers and verification engineers to have a comprehensive strategy from the start of the design/verification process.

Why not expand the use of UVM-based Object Classes to include UVM-based Classes for Cores, Multi-Cores (such as ARM, Intel, and Open Source), Bus Interface for signals (such as AMBA AXI, CHI, PCIe, and Wishbone), Memory and Devices? These could be registered for reusability and constructed within the UVM Environment. These classes could then be utilized in the Build, Connect, and Run phases, providing SOC Verification Engineers with ready-to-use classes that can be extended to the SOC Design being tested.

![Image](image_url)  
Figure 1. UVM Low Power Hierarchical Structure, notional diagrammatic representation of multi-Core the single UVM Power Routines will select the Core to operate.
II. IMPLEMENTATION DETAILS AND FLOWCHART

The creation of an effective Power Management structure can be achieved by utilizing UVM classes that include various tasks such as creating power domains, defining different scopes, and supplying nodes for each domain. These classes can be used as a library and extended to create structures based on the architecture of the device under test. By incorporating these classes, the Power Management structure can be efficiently managed and adapted to suit the specific needs of the device.

The Power Management structure is dependent on the Power state of each domain, which triggers various virtual functions, tasks, and sub-routines. A top-level UVM_power_pkg is incorporated into the test bench for a multi-core and extended based on the specifications defined in UPF to perform PowerUp and PowerDown of any Core sequences. This approach ensures that the Power Management structure is fully integrated into the verification process, providing efficient and effective power management for the device under test.

![Diagram of UVM_LP_PKG](image)

Figure 2. Architecture of Low Power UVM Package Library

III. IMPLEMENTATION DETAILS FOR ARM MULTI-CORE LOW POWER OPERATIONS

ARM has incorporated low-power principles, including UPF, and provided API calls for managing multi-core operations and transitioning between states. The company has also created Power Domains, such as PD_CPU, PD_L1, PD_L2, and PD_CORTEX, which can be managed using register bit enabling and output clamps activation/deactivation, making it unnecessary to create UPF-based Power Domains. These features are leveraged to manage power-down and power-off operations in compliance with ARM's recommendations. Power Domain Classes are extended to the Low Power UVM Package as a library, enabling their use in the Power Management architecture for Memories, Bus Interface cores, and other components. The Power Domain classes for the ARM Cortex A53 processor, such as PDCORTEX, PDCPU, PDL2, and so on, are created by ARM and can be utilized in the multi-core environment through factory registration in build_phase, run_phase, and connect_phase.

By incorporating these classes, the Power Management structure can efficiently manage power-down and power-off operations in compliance with ARM's recommendations. The use of these classes enables the creation of a comprehensive Power Management structure, with support for multiple cores and power domains. The architecture can be implemented using the Low Power UVM Package as a library, making it easier to integrate into the verification process.

ARM provides assembly code instructions for executing PowerUp and PowerDown routines. The ARM Cortex A53 Processor/Cluster development environment supports multiple core states, including Ready(D3_Hot), Normal, Standby, Retention, Dormant, Deep Sleep (D3_Cold), which can involve one, two, or even three-step state transitions. Power UP
and Power Down routines are called to operate on single or multiple cores and are implemented in both C Language and assembly language.

To leverage and reuse the C code, these Power Down routines are imported into the low power UVM SV package using Direct Programming Interface (DPI-C). This interface allows for the calling of C functions from SystemVerilog using "DPI" declarations. By importing C code into the UVM SV package, it is possible to reuse existing code and take advantage of the functions implemented in C to improve the efficiency and effectiveness of the verification process.

IV. PRELIMINARY RESULTS AND SOURCE CODE

In sub-sections "A", "B", "C", and "D," we demonstrate our approach to utilizing ARM multi-Core sequences for low power operations. We achieve this by using ASM and wrapping it within SystemVerilog-based classes through DPI and C. Our routines utilize ARM Core Assembly language for PowerDown and PowerUp of the core or multi-Core, along with its Low Power Management functions. These ARM multi-Core functions are accessed using ASM within the ARM Development Environment, which includes the commented include statement for "ARMv6T2.h," that would be uncommented to run in the environment. We call ARM ASM code in C functions, which are then utilized in SystemVerilog through DPI for extending classes in Low Power extension to UVM.

A. UVM Low power ASM Routine

This is the source code for executing PowerUP and PowerDOWN routine in Assembly Language. It is designed to follow the 32- and 64-bits Instructions set. The ASM calling the desired Header files and include DPI header file.

```c
#ifndef CLOSETARGET
#define CLOSETARGET

typedef enum { CLEAN_BY_SETWAY, 
               INVALIDATE_BY_SETWAY, 
               CLEAN_INVALIDATE_BY_SETWAY, 
               CLEAN_BY_VA_TO_PUC,
               ....
} cmo_type_t;

typedef enum { wfi, not_of_wfi, 
               wfe, not_of_wfe,
               } power_standby_methods_t;

void disable_cache() { 
  asm volatile ( 
    "mr x0, SCLTR_EL3\n" 
    "bic x0, x0, #1 << 12)\n" 
    "bic x0, x0, #1 << 2)\n" 
    "mr SCLTR_EL3, x0\n"
    "isb" );
}
```

B. UVM Low power DPI package

This Assembly Code is being enveloped using DPI calls into an SV package by declaring source file for SystemVerilog package “uvm_lpm_core_pd_pkg”. The C code shown in section A is being called inside “uvm_lpm_core_pd_pkg” package using import “DPI-C” keyword. This helps to provide the connectivity using ARM routines defined using C assembly language in System Verilog.
C. UVM Low power Scenario Package

The below uvm_power_pkg package includes SystemVerilog file which imports DPI functions mentioned in uvm_lp_core_pd_pkg package. Further the same class is being registered in UVM factory. It also includes the members and methods which performed all the power related routines.
D. Functional Description for Power Domains for Power UP and Power DOWN

Referring to the architecture of ARM Cortex A53 using different power domains such as PDCORTEXA53, PDL2, PDCPU, PDL1, etc. are considered and their relevant power routines functions are being called through UVM as shown in below source code.

![Figure 3. ARM Cortex A53 Power Domain Block Diagram](image)

In the sample test case, the user can utilize the library package at different levels. Class and Functions described in section A, B and C are being called in `uvm_power_multicore` class.

```verilog
module tb;
    import uvm_power_pkg::*;
    import uvm_lp_core_pd_pkg::*;
    uvm_power_multicore pd;
    initial begin
        pd=new();
        pd.core_power_down();
    end
endmodule

class uvm_power_memory extends uvm_power;
---
class uvm_power_device extends uvm_power;
---
endpackage

import uvm_lp_core_pd_pkg::*;

class uvm_power_core extends uvm-power;
    function new();
        super.new();
    endfunction
    virtual task uvm_disable_cache_core;
        disablecache_func();
    endtask
---

endclass

class uvm_power_multicore extends uvm_power_core;
    typed struct {
        bit [3:0]NO_OF_CORES;
        bit [3:0]NO_OF_CORES_IN_CLUSTER;
        bit [3:0]NO_OF_CLUSTER;
        bit [2:0]NO_OF_CORES_IN_PROC;
    } multi_core;

    virtual task core_power_down;
    begin
        uvm_disable_cache_core();
        uvm_clean_invalidate_cache();
    end
endtask
```

This class shall be registered in UVM factory of low power package. The full implementation needs to be done in an ARM environment in close collaboration and cooperation from ARM in the ARM Cortex Development environment. So, that would permit us to PowerUp and PowerDown cores. In this paper, the outputs are being observed using $display and C printf (using DPI-C) to check the results. Further, the assembler code which is essential for testing will run on ARM Development Environment.
V. CONCLUSION

In conclusion, the paper proposes the use of ARM ASM Environment for designing Low Power routines for multi-Core as a case study, which can also be applied to other multi-Cores like Intel or ARC. The paper suggests that routines can be built for Bus Interface signals, Memory, and Device needs to be written, as the need for smaller and low power designs increase. The paper emphasizes the importance of implementing power architecture strategy and verification as an integral part of the design process, rather than an afterthought post-functional verification, to avoid unwanted re-spins that can be detrimental to costs and time-to-market guidelines. The paper concludes by recommending that low power classes for multi-Core should be available in the low power extension of UVM Libraries to enable SOC designs to have a UVM-like verification test bench.

REFERENCES