2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Unified Architecture of L1 and L2 Cache with Low Power Extensions for Multi-Core UVM-based Library Package

Avnita Pal, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (avnita@siliconinterfaces.com)
Priyanka Gharat, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (priyanka@siliconinterfaces.com)
Sastry Puranapanda, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (sastry@siliconinterfaces.com)
Darshan Sarode, VLSI Design Engineer, Silicon Interfaces, Mumbai, India (darshan@siliconinterfaces.com)

***Abstract*— This Paper demonstrates the continuum for multi-Core architecture integrating UPF-based Low Power methodologies and strategies for L1 and L2 Cache transitioning in different modes, like Off, Sleep, Dormant, and Retention in PowerUp/Down sequence within the UVM Low Power Package with in-built ASM routines and incorporates these within low power UVM classes using SystemVerilog and DPI. This addresses the limitation of previous works (referenced) to incorporate multi-Core low-power libraries (which in turn have the classes for SOC environment Devices, Buses and Memory) for low-power strategies. These low power libraries are imported in UVM environment.**

Keywords*— *Power Management, Low Power, UVM (Universal Verification Methodology), Functional Verification, SystemVerilog, Unified Power Format (UPF), L1 Cache, L2 Cache, DPI (Direct Programming Interface), PowerUp, PowerDown, Assembly Language (ASM), SoC (System on Chip)

I. Introduction

The prevailing practice of introducing Power Architecture subsequent to Functional Verification lacks optimization. For optimal results, it's recommended to integrate Power Architecture right from the outset, along with Methodologies centred around Functional Verification, Coverage, and Low Power Implementation. Prior studies have demonstrated the efficacy of a unified approach employing a single platform, such as UVM. This approach encompasses the development of device library components, encompassing low-power strategies, Functional Verification Methodology, and UPF-based Low Power Architecture. By adopting this comprehensive strategy, designers and verification engineers benefit from a holistic approach right from the initiation of the design and verification process.

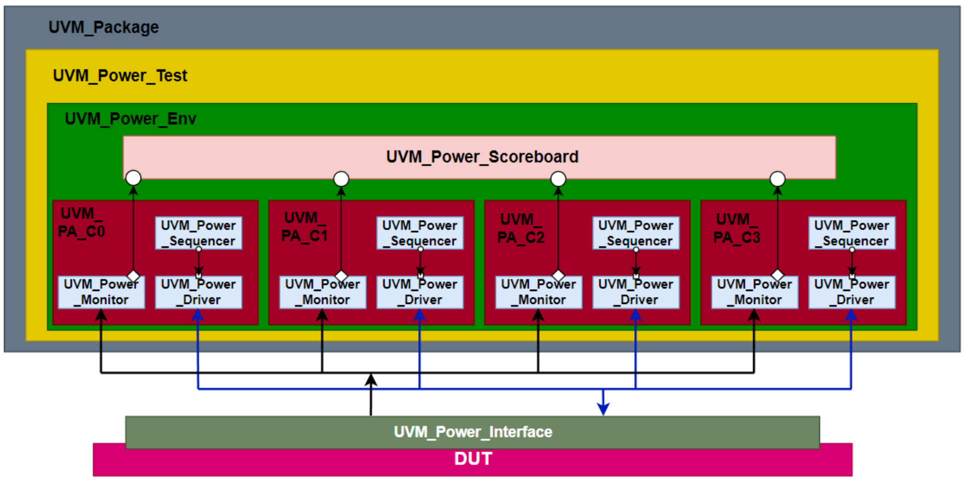


Fig I(a). UVM Low Power Hierarchical Structure

Previous works have utilized a unified platform, such as UVM, to develop library components for devices, encompassing low power strategies, Functional Verification Methodology, UPF-based Low Power Architecture and utilizing in-built ASM (Assembly Language) routines for performing PowerUp/Down of multi-Core. This package now incorporates two levels of cache memory - L1 and L2. L1 cache is on board cache which is small and fast in transferring and fetching the data with low latency and L2 cache is the external cache its integrated with SCU (Snoop Control Unit) controlling up to 4 cores within the clusters to provides the duplicate copies L1 data cache tags for coherency support and high latency which acts as a secondary buffer for fast accessed data. integrated approach empowers designers and verification engineers with a comprehensive strategy right from the initiation of the design/verification process.

In the SoC design we can save power by turning off some cores in a cluster. When a core is turned off, its cache data becomes inaccessible, and it would take a long time to reload the data if required. To address this issue, we can use solution to map the behavior of the L1 and L2 caches simultaneously. These strategies are being build using functions defined in classes so, basically, we first look for the current state or next state in which the core will be transitioning of states if status is seen to be in Standby, Deepsleep, Dormant etc. Further these functions would be invoked within the inbuilt SV class methods using DPI. Strategies would help in preserving the data and making it accessible for other parts of the processor even before shutting down the core, the L1 data is stored to the L2 cache. This allows other parts of the processor to access the L2 cache data via a communication bus, while the core and L1 cache are turned off. This technique reduces power consumption significantly, while maintaining the responsiveness of the processor.

Why not leverage the power of UVM-based Object Classes to encompass Cores, Multi-Cores (e.g., ARM, Intel, and Open Source), Bus Interfaces for signals (e.g., AMBA AXI, CHI, PCIe, and Wishbone), Memory, and Devices? These classes can be registered for reusability and instantiated within the UVM Environment. Such an approach enables SOC Verification Engineers to have readily available classes that can be extended and utilized in UVM Phases for SOC design verification.

II. Methodology

Constructing a robust Power Management framework for multi-core design with L1 and L2 cache, it can be facilitated through the utilization of UVM classes that encompass a range of functions.

This collection of classes can be treated as a library and extended to establish structures aligned with the test device's architecture. By integrating these classes, the Power Management framework can be adeptly observed and applied to precisely match the distinctive requirements based on the state of multi-cores and cache.

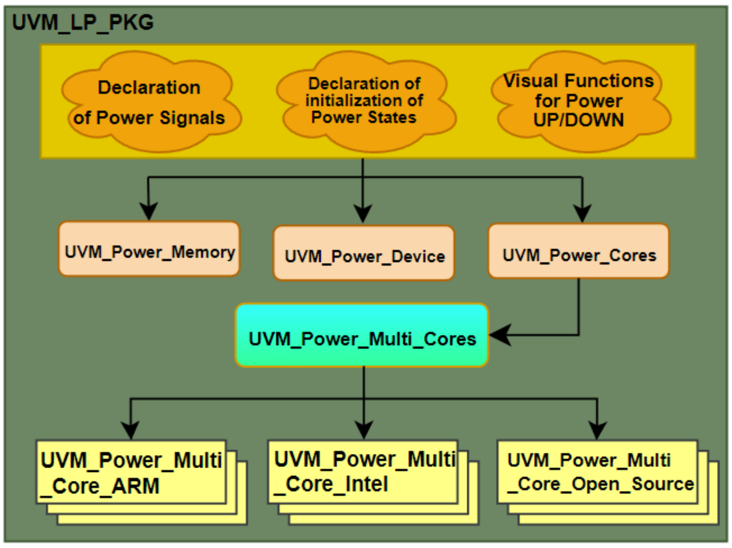

UVM_LP_PKG
Declaration of Power Signals
Declaration of initialization of Power States
Visual Functions for Power UP/DOWN
UVM_Power_Memory
UVM_Power_Device
UVM_Power_Cores
UVM_Power_Multi_Cores
UVM_Power_Multi_Core_ARM
UVM_Power_Multi_Core_Intel
UVM_Power_Multi_Core_Open_Source

Fig II(c). Architecture of Low Power UVM Package Library

The effectiveness of the Power Management structure hinges (supports) on the specific power state of each domain, which activates a cascade of virtual functions, tasks, and sub-routines. At the apex of this, a top-level component called UVM_power_pkg is seamlessly integrated into the test bench tailored for multi-core scenarios. It is systematically expanded in accordance with the specifications outlined in UPF. This augmentation equips it to execute PowerUp and PowerDown sequences for various Cores. This methodology guarantees the complete integration of the Power Management structure within the verification procedure. The result is a streamlined for effective power management system that efficiently achieves power aspects of the device being scrutinized.

III. Implementation Details for Multi-Core Low Power Operations With L1 And L2 Cache

Multi-cores like ARM has seamlessly integrated principles of low power, including the Unified Power Format (UPF), into its operations. It has introduced API calls that facilitate the management of multi-core functionalities and transitions between different operational states. Multi-cores have pre-defined Power Domains such as PD_CPU, PD_L1, PD_L2, and PD_CORTEX.

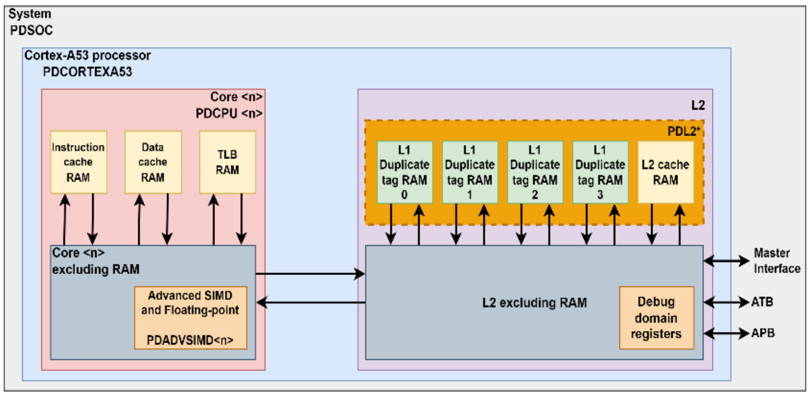


Fig III(a). ARM Cortex A53 Power Domain Block Diagram

These domains can be effectively controlled using register bit enabling and activation/deactivation of output clamps. STANDBYWFI[n] signal serves as an indicator that an individual core has entered an idle and low-power state. Its purpose is to allow the power management controller to deactivate the power supply to that specific core once the STANDBYWFI[n] signal has been activated. And the deactivation or Power Down of the core is done by the ASM routines defined by ARM. So, to call these routines within UVM classes we have made use of DPI import functions. This mechanism helps optimize power consumption in the system by enabling the shutdown of power to cores that are not actively in use, thereby contributing to more efficient energy utilization.

Leveraging these capabilities, for L2 cache the controlling register bit STANDBYWFIL2 signal serves as a crucial indicator of the collective idle and low-power state of both individual cores and the L2 memory system. This signal effectively communicates to the power management controller that all components, including the individual cores and the L2 memory system, have transitioned into an energy-efficient idle state. When the STANDBYWFIL2 signal is asserted, it signifies an opportune moment for the power management controller to initiate power removal from the processor. In this case the routines required for transitioning to idle state are being invoked using DPI import statements within UVM classes which indirectly calls ASM function defined by multi-core for example ARM. The DPI calls are being shown in section IV.

```
class uvm_power_L2_RAM extend uvm_power;
      //L2 Cache Standby state
      rand bit STANDBYWFIL2;
      //Read no snoop control signal ARLOCKM=1(For ACE Interface)
      (Inner/outer shareable Cache) and FOR Load/store
      rand bit ARLOCKM;
      //Write No snoop Control signal AWLOCKM= 1(For ACE Bus Interface transactions)
      (Inner/Outer write through) for load/store
      rand bit AWLOCKM;

      virtual task L2_cache_operation;
            if(STANDBYWFIL2 = 1'b1)
                  $display("asserted to indicate that the L2 memory system is idle");
                  $display("L2 will be able to access the data from other resources");
            if(L2_received_data = L1_send_data)
                  $display("checking the data matching status between L1 and L2");
      endtask
endclass
```

ARM efficiently handles power-down and power-off procedures, adhering to its recommended practices. Power Domain Classes are extended into the Low Power UVM Package as a library. This strategic move enables their seamless integration into the Power Management framework for components like Memories, Bus Interface cores, and others. These classes are furnished by Low Power UVM class and can be conveniently applied in a multi-core context through factory registration, constructing and can be executed during build_phase, run_phase, and connect_phase stages.

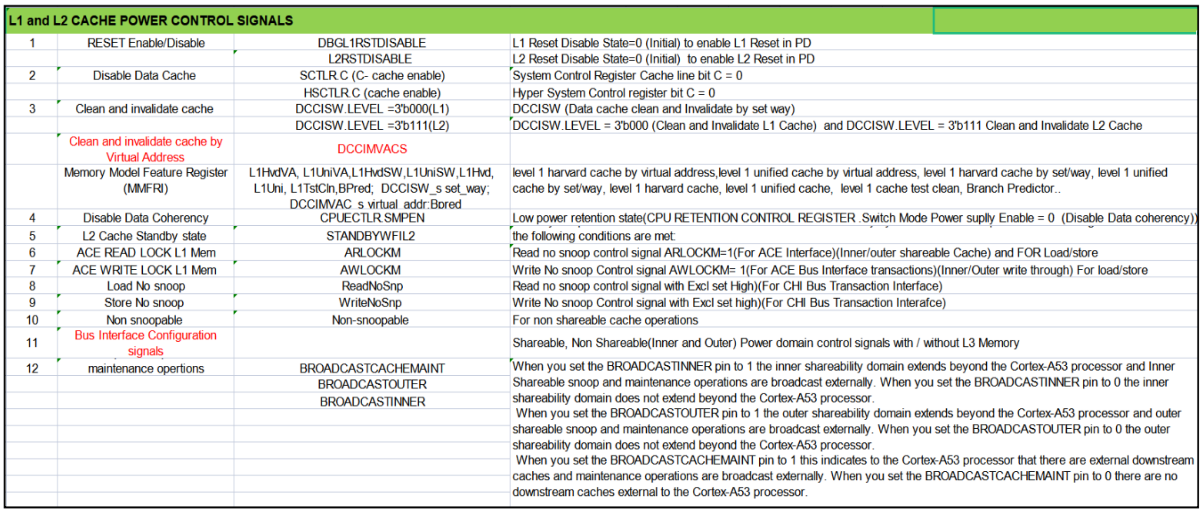
By integrating these classes, the Power Management structure gains the capability to effectively handle power-down and power-off operations while adhering to ARM's recommended practices. The utilization of these classes facilitates the development of a holistic Power Management framework that accommodates multiple cores and power domains. This architectural approach can be realized through the adoption of the Low Power UVM Package as a library, simplifying its integration into the verification process.

ARM offers assembly code instructions designed for executing PowerUp and PowerDown routines. Within the ARM Cortex A53 Processor/Cluster development environment, diverse core states are supported, encompassing Ready (D3_Hot), Normal, Standby, Retention, Dormant, and Deep Sleep (D3_Cold). These states can entail multi-step transitions involving one, two, or even three stages.

Power Up and Power Down routines are utilized to manage the functionality of individual or multiple cores. These routines are developed in both C Language and assembly language to ensure comprehensive coverage. The behavior of the L1 and L2 caches is contingent on the present state of the Core and the intended state it needs to transition to. For instance, transitions like Normal to Standby or from Retention to Off. The management of L1 and L2 caches can differ, with the L1 potentially being powered down while the L2 remains active. In instances where a Power Down sequence is initiated, system memory retention becomes crucial to preserve memory contents.

To harness the benefits of existing C code and promote reusability, the Power Down routines are integrated into the low power UVM System Verilog (SV) package through the Direct Programming Interface (DPI-C). The DPI-C serves as an interface that enables the invocation of C functions from System Verilog by employing "DPI" declarations. Through the incorporation of C code into the UVM SV package, the opportunity arises to tap into pre-existing code and leverage the functionalities established in C. This integration optimizes the efficiency and efficacy of the verification process.

Even after having control on the power management using ASM routine of L1 and L2 cache with multi-core system. There is also data in system memory on which one needs to have control in terms of accessing it with help of SCU (snooping control unit). This System Coherency Unit (SCU) within the ARM processor plays a pivotal role in maintaining cache coherence within multi-core configurations. It integrates the synchronization of data across cores, thereby elevating system reliability. Following are some important internal signals BROADCASTINNER: Facilitates inner-level cache broadcasts. BROADCASTOUTER: Initiates outer-level cache broadcasts. BROADCASTCACHEMAINT: Triggers cache maintenance broadcasts. SYSBARDISABLE: Controls system barrier functionality. Along with the mentioned signals below are a few additional lists of signals which are used for controlling the data. Since to have access to a particular system memory we need to have access over the state of the L2 so that if the state is retention, then specific signals need to be activated such as Disable Data coherency. Further to have transfer of data to L3 cache as well we have made use of some bus interface configuration signals.

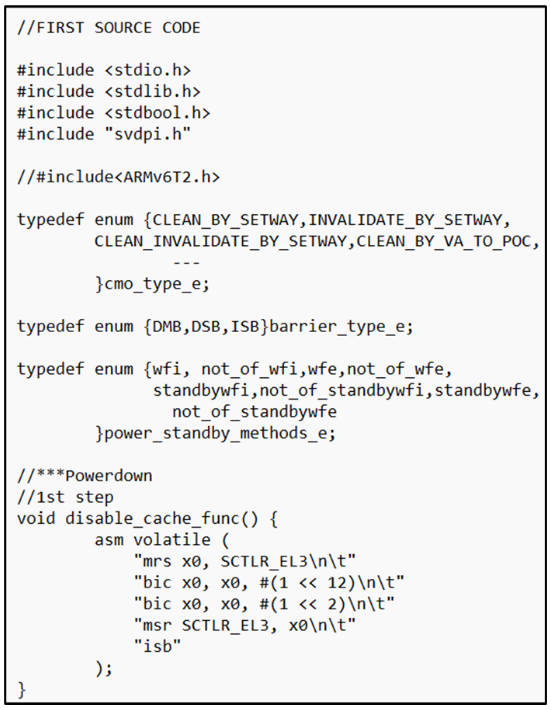


L1 and L2 CACHE POWER CONTROL SIGNALS			
1	RESET Enable/Disable	DBGL1RSTDISABLE	L1 Reset Disable State=0 (Initial) to enable L1 Reset in PD
		L2RSTDISABLE	L2 Reset Disable State=0 (Initial) to enable L2 Reset in PD
2	Disable Data Cache	SCTLR.C (C- cache enable)	System Control Register Cache line bit C = 0
		HSCTLR.C (cache enable)	Hyper System Control register bit C = 0
3	Clean and invalidate cache	DCCISW.LEVEL =3'b000(L1)	DCCISW (Data cache clean and Invalidate by set way)
		DCCISW.LEVEL =3'b111(L2)	DCCISW.LEVEL = 3'b000 (Clean and Invalidate L1 Cache) and DCCISW.LEVEL = 3'b111 Clean and Invalidate L2 Cache
	Clean and invalidate cache by Virtual Address	DCCIMVACS	
	Memory Model Feature Register (MMFRI)	L1HvdVA, L1UniVA,L1HvdSW,L1UniSW,L1Hvd, L1Uni, L1TstCln,BPred; DCCISW_s set_way; DCCIMVAC_s virtual_addr:Bpred	level 1 harvard cache by virtual address,level 1 unified cache by virtual address, level 1 harvard cache by set/way, level 1 unified cache by set/way, level 1 harvard cache, level 1 unified cache, level 1 cache test clean, Branch Predictor..
4	Disable Data Coherency	CPUECTLR.SMPEN	Low power retention state(CPU RETENTION CONTROL REGISTER .Switch Mode Power suplly Enable = 0 (Disable Data coherency))
5	L2 Cache Standby state	STANDBYWFIL2	the following conditions are met:
6	ACE READ LOCK L1 Mem	ARLOCKM	Read no snoop control signal ARLOCKM=1(For ACE Interface)(Inner/outer shareable Cache) and FOR Load/store
7	ACE WRITE LOCK L1 Mem	AWLOCKM	Write No snoop Control signal AWLOCKM= 1(For ACE Bus Interface transactions)(Inner/Outer write through) For load/store
8	Load No snoop	ReadNoSnp	Read no snoop control signal with Excl set High)(For CHI Bus Transaction Interface)
9	Store No snoop	WriteNoSnp	Write No snoop Control signal with Excl set high)(For CHI Bus Transaction Interafce)
10	Non snoopable	Non-snoopable	For non shareable cache operations
11	Bus Interface Configuration signals		Shareable, Non Shareable(Inner and Outer) Power domain control signals with / without L3 Memory
12	maintenance opertions	BROADCASTCACHEMAINT	When you set the BROADCASTINNER pin to 1 the inner shareability domain extends beyond the Cortex-A53 processor and Inner Shareable snoop and maintenance operations are broadcast externally. When you set the BROADCASTINNER pin to 0 the inner shareability domain does not extend beyond the Cortex-A53 processor. When you set the BROADCASTOUTER pin to 1 the outer shareability domain extends beyond the Cortex-A53 processor and outer shareable snoop and maintenance operations are broadcast externally. When you set the BROADCASTOUTER pin to 0 the outer shareability domain does not extend beyond the Cortex-A53 processor. When you set the BROADCASTCACHEMAINT pin to 1 this indicates to the Cortex-A53 processor that there are external downstream caches and maintenance operations are broadcast externally. When you set the BROADCASTCACHEMAINT pin to 0 there are no downstream caches external to the Cortex-A53 processor.
		BROADCASTOUTER	
		BROADCASTINNER	

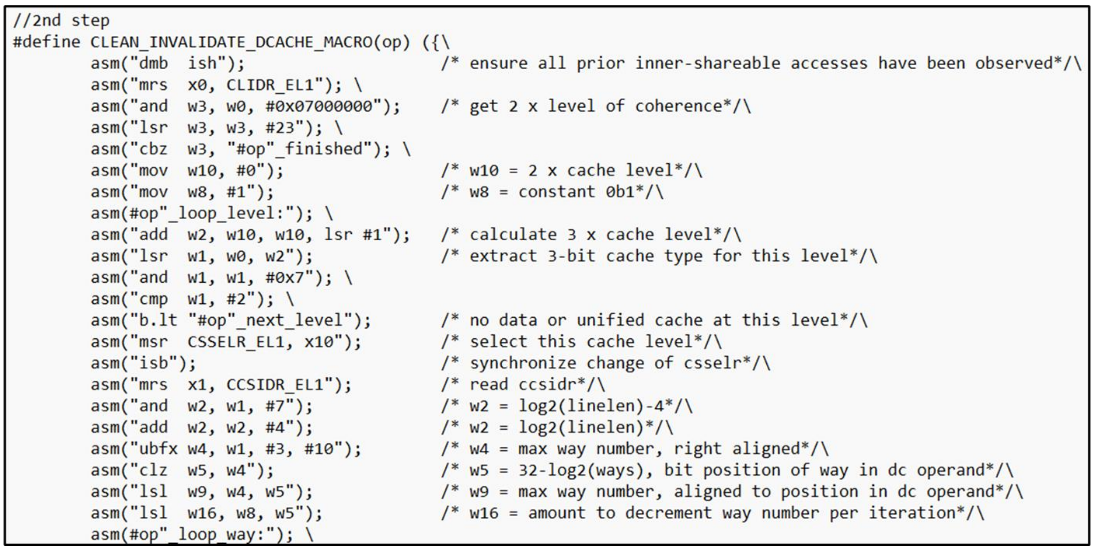
These integral signals ensure streamlined inter-core data exchange and uphold memory consistency. By seamlessly integrating these signals into UVM classes through C routines, facilitated by the DPI function, a coordinated interplay between hardware and verification realms is realized. This collaborative synergy becomes indispensable for conducting rigorous assessments and validations of cache coherency mechanisms. In effect, it substantiates the correct functioning of these mechanisms, significantly contributing to the robustness and efficiency of system performance.

IV. RESULTS AND SOURCE CODE

In sections "A," "B," "C," and "D," we showcase our method of employing ARM multi-Core L1 & L2 Power Up and Power Down sequences for low power actions. Our approach involves using ASM, encapsulated within System Verilog-based classes through DPI and C. We utilize ARM Core Assembly language for Power Down and Power Up, incorporating Low Power Management functions. ARM multi-Core functions are accessed via ASM in the ARM Development Environment, with the option to activate the "ARMv6T2.h" include statement. ARM ASM code is called within C functions, subsequently employed in System Verilog via DPI to extend classes in Low Power UVM extensions.



```
//FIRST SOURCE CODE

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "svdpi.h"

//#include<ARMv6T2.h>

typedef enum {CLEAN_BY_SETWAY,INVALIDATE_BY_SETWAY,
        CLEAN_INVALIDATE_BY_SETWAY,CLEAN_BY_VA_TO_POC,
                ---
        }cmo_type_e;

typedef enum {DMB,DSB,ISB}barrier_type_e;

typedef enum {wfi, not_of_wfi,wfe,not_of_wfe,
              standbywfi,not_of_standbywfi,standbywfe,
                not_of_standbywfe
        }power_standby_methods_e;

//***Powerdown
//1st step
void disable_cache_func() {
        asm volatile (
            "mrs x0, SCTLR_EL3\n\t"
            "bic x0, x0, #(1 << 12)\n\t"
            "bic x0, x0, #(1 << 2)\n\t"
            "msr SCTLR_EL3, x0\n\t"
            "isb"
        );
}
```



```
//2nd step
#define CLEAN_INVALIDATE_DCACHE_MACRO(op) ({\
        asm("dmb  ish");                        /* ensure all prior inner-shareable accesses have been observed*/\
        asm("mrs  x0, CLIDR_EL1"); \
        asm("and  w3, w0, #0x07000000");        /* get 2 x level of coherence*/\
        asm("lsr  w3, w3, #23"); \
        asm("cbz  w3, "#op"_finished"); \
        asm("mov  w10, #0");                    /* w10 = 2 x cache level*/\
        asm("mov  w8, #1");                     /* w8 = constant 0b1*/\
        asm(#op"_loop_level:"); \
        asm("add  w2, w10, w10, lsr #1");       /* calculate 3 x cache level*/\
        asm("lsr  w1, w0, w2");                 /* extract 3-bit cache type for this level*/\
        asm("and  w1, w1, #0x7"); \
        asm("cmp  w1, #2"); \
        asm("b.lt "#op"_next_level");           /* no data or unified cache at this level*/\
        asm("msr  CSSELR_EL1, x10");            /* select this cache level*/\
        asm("isb");                             /* synchronize change of csselr*/\
        asm("mrs  x1, CCSIDR_EL1");             /* read ccsidr*/\
        asm("and  w2, w1, #7");                 /* w2 = log2(linelen)-4*/\
        asm("add  w2, w2, #4");                 /* w2 = log2(linelen)*/\
        asm("ubfx w4, w1, #3, #10");            /* w4 = max way number, right aligned*/\
        asm("clz  w5, w4");                     /* w5 = 32-log2(ways), bit position of way in dc operand*/\
        asm("lsl  w9, w4, w5");                 /* w9 = max way number, aligned to position in dc operand*/\
        asm("lsl  w16, w8, w5");                /* w16 = amount to decrement way number per iteration*/\
        asm(#op"_loop_way:"); \
```

In essence, by relocating L1 cache data to L2 cache before core power-off, the ARM Cortex A53 processor conserves power, upholds data integrity, and sustains processor performance. Notably applied in battery-dependent devices like smartphones, tablets, and IoT devices.

A. UVM Low Power DPI Package

This Assembly Code is being enveloped using DPI calls into an SV package by declaring source file for System Verilog package "uvm_lp_core_pd_pkg". The C code shown in section A is being called inside "uvm_lp_core_pd_pkg" package using import "DPI-C" keyword. This helps to provide connectivity using ARM routines defined using C assembly language in System Verilog.

```
//SECOND SOURCE CODE
`include "arm_cortexa53_assembly_code.c"
package uvm_lp_core_pd_pkg;

        typedef enum {CLEAN_BY_SETWAY,
                INVALIDATE_BY_SETWAY,
                CLEAN_INVALIDATE_BY_SETWAY,
                CLEAN_BY_VA_TO_POC,
                CLEAN_BY_VA_TO_POU,
                CLEAN_BY_VA_TO_POP,
                INVALIDATE_BY_VA_TO_POC,
                CLEAN_INVALIDATE_BY_VA_TO_POC,
                CACHE_ZERO_BY_VA,
                INVALIDATE_ALL_TO_POUIS,
                INVALIDATE_ALL_TO_POU,
                INVALIDATE_BY_VA_TO_POU
        }uvm_lp_core_cmo_type_e;

        typedef enum {DMB,
                 DSB,
                 ISB
        }uvm_lp_core_barrier_type_e;

        typedef enum {
                wfi,
                not_of_wfi,
                wfe,
                not_of_wfe,
                standbywfi,
                not_of_standbywfi,
                standbywfe,
                not_of_standbywfe
        }uvm_lp_core_power_standby_methods_e;
```

```
import "DPI-C" function void core_status_t();
import "DPI-C" function void check_L1data_status();
import "DPI-C" function void L2_cache_operation();
import "DPI-C" function void disable_cache_func();
import "DPI-C" function void clean_invalidate_dcache_func(cmo_type);
import "DPI-C" function void cpu_extended_contrl_reg_func();
import "DPI-C" function void barrier_func(barrier);
import "DPI-C" function void transition_func(wf);
import "DPI-C" function void debug_sig_func(bit DBGPWRDUP);
import "DPI-C" function void activate_output_clamp_func(bit CLAMPCOREOUT);
import "DPI-C" function void cpu_processor_power_func(bit nCPUPORESET);
import "DPI-C" function void power_domain_cpu_func(bit PDCPU);
```

B. *UVM Low Power Scenario Package*

The uvm_power_pkg package encompasses a System Verilog file that imports DPI functions from the uvm_lp_core_pd_pkg package. Additionally, the same class is registered within the UVM factory. This package encompasses both the elements and methods responsible for executing all power-related routines.

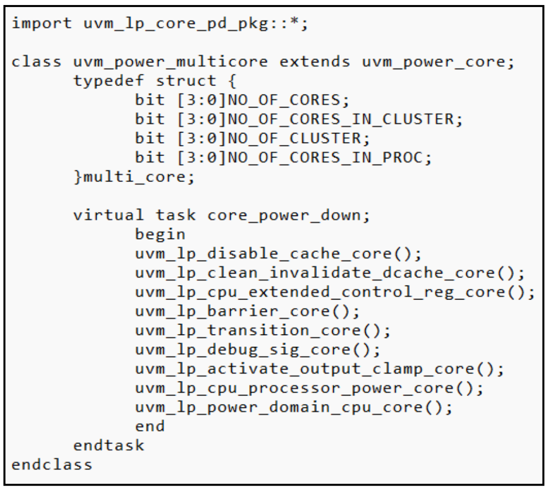
```
//THIRD SOURCE CODE

`include "uvm_lp_core_pd_pkg_dpi_c.sv"

package uvm_power_pkg;

class uvm_power;
        rand bit Wait_For_Interrupt;
        rand bit Wait_For_Event;
        rand bit Delay_time_for_power_down;
        rand bit Enable_wakeup_timer_interrupt_before_power_down;

        typedef enum {off,normal,standby,sleep,retention,dormant,
              deepsleep,ready,c0,c1,c2,c3,c4,c6,c7,c8}power_state;

  power_state state;
```

```
virtual function int powerup(state);
  begin
    case(state)
      c0: begin
        $display("It is in active mode");
      end
      c1: $display("Auto halt");
      c2: $display("Temporary state");
      c3: $display(" l1 and l2 caches will be flush");
      c4: $display("CPU is in deep sleep");
      c6: $display("Saves the core state before shutting");
      c7: $display("c6 + LLC may be flush");
      c8: $display("c7+LLC may be flush");
    endcase
  end
endfunction

virtual function sequential_power_down_up_multi_core_f();
endfunction

virtual function int power_up_another_core_f();
endfunction
```

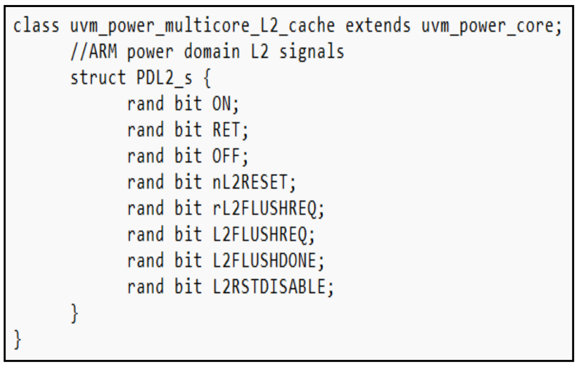
```
class core_status_for_L1 extends uvm_power;
      L1_data_cache L1_dc;
      power_state p_states;

      virtual task core_status_t;
            if(p_states = off)
                  $display("turning off L1 data cache");
            else
                  $display("looks for next transaction for the core");
      endclass

      virtual task check_L1data_status;
            if(L1_dc != 0)
                  $display("data transferring to L1 to L2 cache");
            else
                  $display("called the core routine");
      endtask
endclass
```

C. *Functional Description for Power Domains for Power Up and Power Down*

The ARM Cortex A53 architecture, employing distinct power domains like PDCORTEXA53, PDCPU, PDL1, PDL2, etc., triggers relevant power routine functions via UVM. Code in sections A, B, and C is invoked in the uvm_power_multicore class during user test cases.



```
import uvm_lp_core_pd_pkg::*;

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

      virtual task core_power_down;
            begin
            uvm_lp_disable_cache_core();
            uvm_lp_clean_invalidate_dcache_core();
            uvm_lp_cpu_extended_control_reg_core();
            uvm_lp_barrier_core();
            uvm_lp_transition_core();
            uvm_lp_debug_sig_core();
            uvm_lp_activate_output_clamp_core();
            uvm_lp_cpu_processor_power_core();
            uvm_lp_power_domain_cpu_core();
            end
      endtask
endclass
```



```
class uvm_power_multicore_L2_cache extends uvm_power_core;
      //ARM power domain L2 signals
      struct PDL2_s {
            rand bit ON;
            rand bit RET;
            rand bit OFF;
            rand bit nL2RESET;
            rand bit rL2FLUSHREQ;
            rand bit L2FLUSHREQ;
            rand bit L2FLUSHDONE;
            rand bit L2RSTDISABLE;
      }
}
```

Register this class in the UVM factory of the low power package. For complete implementation, close collaboration with ARM in the ARM Cortex Development environment is essential. This facilitates L1 & L2 Cache Power Up and Power Down. Outputs are observed via $display and C printf (using DPI-C), while the necessary assembler code for testing operates on the ARM Development Environment.

V. Conclusion

This paper introduces an extended UVM Package catering to Low Power needs for Multi-Core L1 & L2 Cache across varied low power modes. The developed ARM® multi-Core routines serve as a case study, extendable to other multi-Cores like Intel or ARC or RISC-V. It showcases that C/C++-based low power routines for Multi-Core Devices can be seamlessly integrated into the extended UVM Package featuring Low Power strategies. The paper underscores the significance of incorporating power architecture strategy and verification as intrinsic elements of the design process, thwarting the need for costly post-functional verification reiterations. A recommendation is made to incorporate multi-Core low power classes within UVM's low power extension, facilitating UVM-like verification for SOC designs.

References

[1] UVM Community (accellera.org) https://accellera.org/community/uvm.

[2] Guide to change in IEEE1801-2013(UPF2.1) (techdesignforums.com)

[3] Arm Cortex-A53 MPCore Processor Technical Reference Manual r0p4

[4] Verification Methodology Manual for Low Power https://www.synopsys.com/company/resources/synopsys_press/vmm-_low-power.html