Verification of the PULPino SoC platform using UVM

Mahesh R., Cisma (a subsidiary of Verikwest Systems Inc)
Shamanth H. K., Cisma (a subsidiary of Verikwest Systems Inc)
Champaka Ramachandran, Verikwest Systems Inc

1. INTRODUCTION

During the last few years, there have been several implementations of the RISCV (ISA). An example of one such implementation is the PULPino SOC [1]

PULPino is a 32-bit RISCV single-core SOC targeted for low power signal processing applications. The RISCV ISA has been extended with additional instructions, including hardware loops, post/pre incrementing loads-stores, multiply-accumulate etc. The processor core is implemented as a 4-stage pipeline. It does not have a cache, DMA or memory hierarchy.The SOC has several peripherals including UART, GPIO, I2C, SPI. This is shown in figure 1.

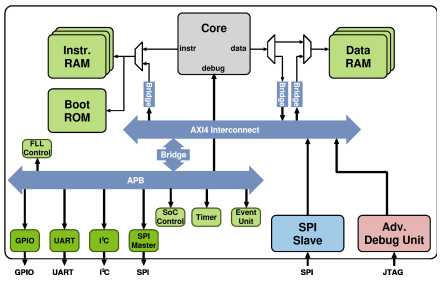

Fig. 1. Pulpino Architecture

2. SOC VERIFICATION GOALS AND CHALLENGES

The main goal of SOC verification is to ensure that the CPU core is able to communicate with all the peripherals in the chip. It offers several interesting challenges. Some of the main issues that we dealt with including

—Is the CPU able to communicate (read/write) to all the peripherals?

—Creating Verification IPs (VIP) for each of the peripheral protocols.

—Each test will have two components (i) “C”, (ii) “SV” These components have to be coordinated.

3. UVM

We used the Universal verification methodology (UVM) to create the testbench and all the tests. UVM is a verification methodology that was recently standardized by the IEEE [2].The UVM environment instantiates one UVM agent (or VIP) for each protocol. Each agent communicates to the DUT through SystemVerilog (SV) interfaces and virtual interfaces. Figure 2 shows an example of an UVM environment with two agents.

Each UVM agent contains: (a) a *driver* for driving transactions or responses on the bus, (b) a *sequencer* that sequences the transactions on the driver based on the test-writers intent, and (c) a *monitor* that monitors all the transactions on the bus and informs the scoreboard.

4. SOC VERIFICATION METHODOLOGY

4.1 Creating VIPS

The first step was to create VIPs for all the peripherals. For the PULPino SOC the VIPs were required.

(1) GPIO master and slave agents
(2) SPI slave agent
(3) UART slave agent
(4) i2C slave agent

4.2 High level APIs

To simplify our "C" test cases, high level API functions to exercise individual features of the PULPino ware created. For example, an API *send_uart("string to send")* was created to store a string in RAM, and instruct the UART peripheral to write on the bus.

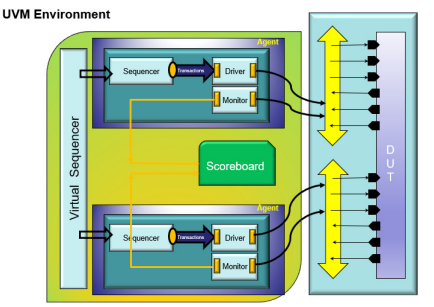

Fig. 2. UVM Environment

4.3 Test flow

—A C test is compiled using gcc from PULPino toolchain. This generates binary files for instruction and data which are loaded on the core.

—A UVM test is started on simulator along with the C test.

—Typically a test would involve read or write operation to one or more external peripherals. This involves fetching of data from Data Memory and transferring it to the appropriate peripherals. In some tests data is received from the external VIP and stored in the Data Memory.

4.4 Ending a Test

We designed two mechanisms to accomplish this.

—Unique string: At the end of a C test, a unique string is sent through a peripheral (like UART). The corresponding VIP will wait for this string to drop the UVM objection, thereby ending the test.

—GPIO pin: At the end of C test a particular GPIO pin is asserted. The GPIO slave will wait for this GPIO pin to drop the UVM objection, thereby ending the test.

4.5 Interrupt testing

—Each peripheral on the SoC is dedicated with a GPIO pin and configured as input.

—When a slave VIP needs to interrupt the CPU, the GPIO pin of that peripheral is driven by the GPIO master VIP

—We implemented interrupt service routine(ISR) dedicated for each peripheral.

—When the RISCV core observes the asserted GPIO pin, then the on-going process will be halted and the ISR dedicated to that particular pin is executed. After the completion of the ISR the main process is resumed.

5. RESULTS

—Around 30 test cases were generated (both in C and UVM).

—Testcases performed read and write(byte, bytes, halfword, word, words) operation on interfaces, between SoC and external VIPs.

—UVM objections and GPIO pins synchronises the end of simulation.

6. REFERENCES

(1) PULPINO ETH Zurich, Integrated Systems Laboratory and Universita Di Bologna, http://iis-projects.ee.ethz.ch/index.php/PULP

(2) IEEE 1800.2-2017, IEEE Standard for Universal Verification Methodology, Language Reference Manual, http://ieeexplore.ieee.org/document/7932212

(3) IEEE 1800-2017, IEEE Standard for System Verilog - Unified Hardware Design, Specification and Verification Language.