International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

IP Verification of DMA Controller for OpenPOWER Processor Core Based Fabless System on Chip (SoC)

Kuncham Penchalanarasaiah*,
M.Tech Student*,
Department of ECE,
JNTUACEA, Anantapur,
Andhra Pradesh, India.

T. Prahlad Reddy, M.Tech, (Ph.D.)**,
Assistant Proffessor(Adhoc)**,
Department of ECE,
JNTUACEA, Anantapur,
Andhra Pradesh, India.

Dr. G. Mamatha, M.Tech, Ph.D.***,
Assistant Proffessor***,
Department of ECE,
JNTUACEA, Anantapur,
Andhra Pradesh, India.

***Abstract*— Direct memory access (DMA) is a memory speed-up method that enables an input/output (I/O) device to send or receive data to or from the main memory without passing via the CPU. DMA operates by "cycle stealing" memory bus access time from the CPU. It lowers CPU usage by enabling the network device to transport packet data straight into the system's memory. The device requests that the CPU retain its data, address, and control buses using a DMA controller so that it is free to transport data directly to and from the memory. Eight DMA channels, each with a 16-bit address and count registers, make up the DMA controller. This project aims to accomplish IP Verification of DMA controller, which is connected to an OpenPower processor A2O core based fabless SoC, using the AXI4 interface. The IP verification environment of DMA controller can be created by using System Verilog, Verilog, and verification methodologies like Universal Verification Methodology (UVM). Verification can be done by using software tools from Mentor Graphics (Questa®) and Xilinx Vivado®, respectively.**

Keywords—DMA, Verification, IP Verification, SoC, OpenPOWER, A2O, UVM, System Verilog, Fabless.

I. INTRODUCTION

This DMA controller supports the 8 channels for all the peripherals which are designed in this SoC and the operation of all those will be activated depending on the registers which are specified in the design of the DMA controller. There are separate registers for the interrupts handling, software commands, address paths which includes the write as well as the read transactions which will happen and in the similar manner this also includes the registers which are meant for the data path for both write as well as the read transactions. If we want to configure the external bus easily there is also an interface which is mentioned with the APB and the AXI interface has been given for the configuration of the registers of the DMA controller. Verification is the critical stage in the creation of a design. Nearly 80% of time in the design cycle is spent on verification. Technology requires a rapid and trustworthy verification mechanism in order to narrow the gap between supply and product demand. We are forced to create bigger, more capable, and more sophisticated designs by technological demands. High complexity designs are more prone to errors. Traditional verification techniques do not work well with them. The most common methodology for verifying intricate VLSI designs is UVM. UVM uses automation mechanisms including the production of random stimuli and Data and automation aspects like read, write, compare and copy are addressed by transaction-level modelling (TLM). The development of a test bench for AXI4 bus to Memory controller is to verify transactions between them. The authentication of write or read activities over Bridge is justified by UVM verification. verifying bridge transition with UVM is a crucial goals, and test bench acts as the master's for the AXI4 interface, which provide the needed input signals. As a result, The addition of a self checking mechanisms in the test bench was driven by the assertions at interface for the integration of reusable-environment into the tolerance detection approaches. When a necessary condition (or conditions) is (are) broken, assertions identify errors as well as run-time fatal errors. use of the two different interfaces in place of one, especially for bridge node with independent clocks mechanism, allows synchronization to absorb the unique qualities of the bridge more quickly. The provided testbench supports reusable environment and works with all bridge transitions.

II. DESIGN VERIFICATION

These higher-level integrations are individually confirmed once the components are tested and connected to a subsystem, and then they are combined with additional integrations to create even bigger assemblies. until all systems have been integrated and tested.

Activities:

following activities are performed:

1. Create processes for subsystem verification, If a sub-system verification plan was created, specific instructions would need to be provided in order to carry out the sub-system verification. The precise steps that will be done to validate each requirement assigned to the sub-system are defined by this method.
2. Combine the parts into a subsystem. Applications are created by combining modules and components into a subsystem.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

3. Performs the Sub-system verification. The subsystem needs are verified using the sub-system verification procedures defined.

III. VERIFICATION PLAN

The establishment of test benches and automation are two ways that are included in the verification plan as procedures to be followed. The verification strategy is typically separate from the actual Verification Tests.

a) Design features
b) Testbench Architecture
c) Coverage models
d) Self-checking strategy
e) Test scenarios
f) Executable verification plan

The verification technique is guided by a verification plan that specifies the hardware design elements that must be confirmed. For instance, the characteristics of a system may be defined in the verification plan and then translated into the established coverage metrics. The design cannot go on to the following stage of the flow unless those coverage targets have been achieved. Although the verification plan does not include the actual tests, it may specify the strategy to be used. Using formal methods, some objectives may be simpler to achieve. Others may require execution on an emulator or virtual prototype if the anticipated runtimes are longer than what can be accommodated by simulation, while some may be better suited to simulation.

IV. TEST PLAN

A specification document called a verification test plan will include all the information needed to validate a design. Initially, the Design Verification Engineer who is in charge of establishing the test plan is familiar with the specifications of the Design under test (DUT).

Sl no	Title	Description	Generate	Testcase_name	Tags
1	Rest test	When it triggers, all the signals reset to the initial state. when it's low, then all the signals go to the active state.	Random stimulus	**reset_test**	SV, UVM
2	Base test	It will verify the different functionalities of DMA through the AXI bus. 1. task: **rd_n5_compare_mem** input parameters: addr: Memory address to read exp_data: expected data for comparing Description: Read data from the specified address and send the expected data to scoreboard. 2. task: **rd_data_frm_mem** input parameters: addr: Memory address to read rand_addr: flag to enable random address selection Description: Read data from the specified address. 3. task: **wr_read_data_f_mem** input parameters: addr: Memory address to write data: Data to write rand_addr: flag to enable random address selection rand_data: flag to enable random data write Description: Read data from the specified address. 4. task: **asset_dut** input parameters: None Description: Reset the APB intf by triggering the PRESETn signal. 5. task: **generate_mem_rd_err** input parameters: addr: Memory address to read rand_addr: flag to enable random address selection Description: Initiate read transfer with memory out of bound error. 6. task: **generate_mem_wr_err** input parameters: addr: Memory address to write data: Data to write rand_addr: flag to enable random address selection rand_data: flag to enable random data write Description: Initiate write transfer with memory out of bound error.	Random stimulus	**base_test**	SV, UVM
3	Read and write test	It will perform the read and write operation between peripherals (memory) and DMA through the AXI bus.	Random stimulus	**directed_reg_write_read_test**	SV, UVM, Memory
4	Package test	It will perform all the resets, reads and writes, and base tests one at a time.	Random stimulus	**test_pkg**	SV, UVM

Fig 1. DMA Test plan

To conduct verification with quality and within a reasonable time frame, preparation is always essential. If you don't give this enough weight, you're often setting yourself up for failure later on in the project. This could involve quality loss, bug escape, rewriting of various infrastructure tasks, and timing constraints. Details on all the design aspects that need to be confirmed should be included in the plan. Each item in the test plan should provide a brief description of each feature. Along with the features, the test plan should provide a list of all supported configurations that these features should be evaluated in. These features/configurations won't all require separate testing. The majority of the time, a mix of these features and settings has to be evaluated. Accordingly, information on stimulus infrastructure should be provided. It is wise to record particular micro-architectural examples that require verification of accuracy in addition to characteristics and combinations. This could entail explicitly identifying instances for certain test scenarios or coverage analysis. Various interface features and internal micro-architectural events are a few instances of this like fifos, arbitration, state machines, and logical block interactions. Other things to test include particular stimulus patterns, interactions, high level usage scenarios, potential deadlock or livelock situations, etc.; some of these depend on the type of design. At the end, this section should provide a high quality stimulus and coverage qualities that can ensure the stimulus' quality.

V. TEST BENCH ARCHITECTURE

The VLSI industry uses the Universal Verification Methodology (UVM), which is a System Verilog language-based verification methodology, extensively. a approach called UVM was developed to create test stands for design verification. here APB VIP UVM Testbench Architecture is for DMA Register configuration and AXI VIP is for stimulus generation for memory.

A. APB VIP UVM Testbench Architecture

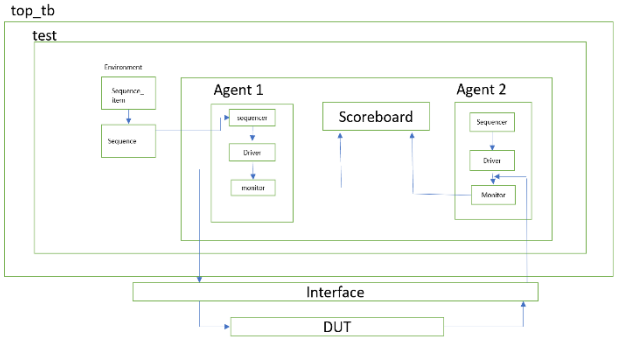

Fig 2. APB Testbench Architecture

B. AXI VIP UVM Testbench Architecture

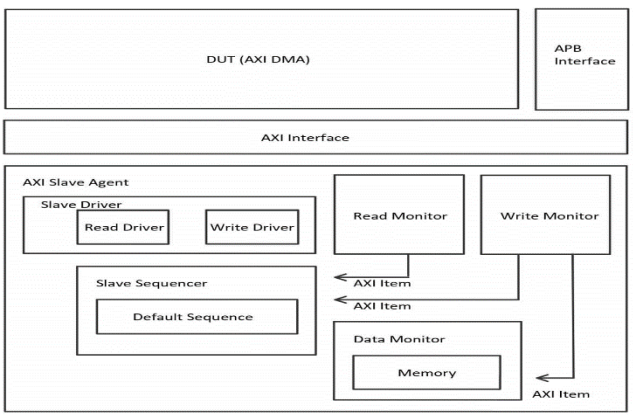

Fig 3. DMA_AXI Testbench Architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 11, November-2022

Test: Setting up the testbench. builds the environment at the next level down in the hierarchy to begin developing the test bench component (env). starts the sequences, which initiates the stimuli.
Environment: For higher level components like agents and scoreboard, it serves as a container component.
Score Board: obtains the data items from the write monitor, then compares them to the result.
Agents: The uvm component specific to an interface or protocol is part of the UVM agent group.
Sequence_item: This will specify the pin activity that the agent produces.
Monitor: analyzes the activity of the interface signal's pins and turns it into data packets for transmission to components like a scoreboard.
Driver: The stimulus from the sequencer will be received, and it will then push the packet data inside the transaction into pin level to the design that is being tested.
Sequence: This will specify the order in which the data item must be created and transferred to or from the driver.
Sequencer: It will be in responsible for sending the sequence item-generated data packets to the driver.
Interface: Contain design signals that will be driven or monitored.

VI. RESULT

Testcase name1: **reset_test**

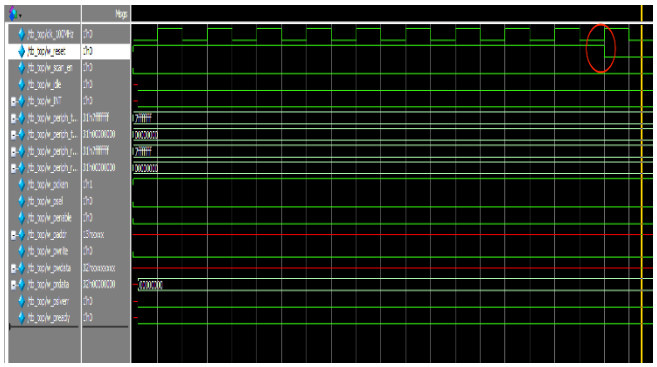

Fig 4. DMA_AXI Reset

Testcase name2: **directed_reg_write_read_test**

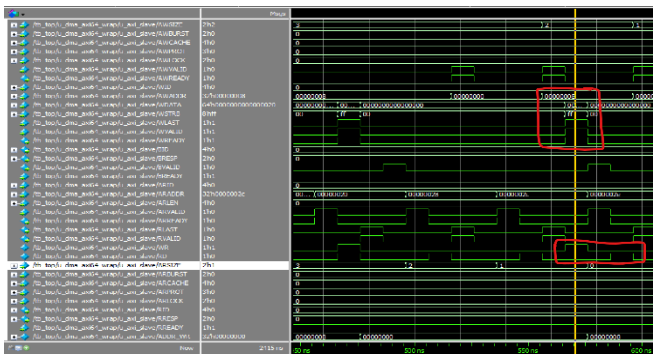

Fig 5. DMA Read and Write

VII. COVERAGE REPORT

The term "coverage" refers to how many functions or elements of the design have been put to the test. Knowing what feature has been covered by a set of tests in regressions will be helpful in restricted random verification.

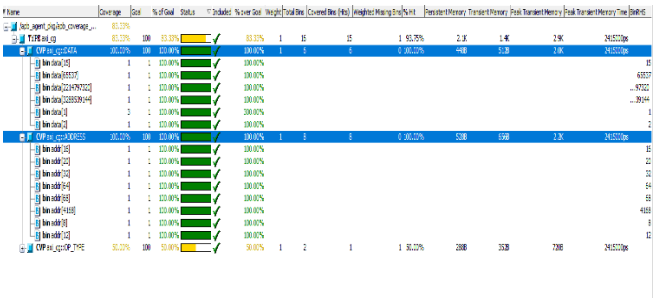

Fig 6. DMA Read and Write Coverage report

VIII. CONCLUSION

This Project developed, Simulated and Verified of DMA Controller which uses A2O processor core based fabless SoC uses SV and UVM using Questa Sim. Using AXI4, the DMA gives the commands to one of the memory or peripheral to transfer the data to another memory or peripheral. In Xilinx Vivado, the design is developed and interfaced with A2O, while Mentor Questa is used for simulation and coverage report.

IX. REFERENCES

[1] PR200 configurable Dual Core High Performance AXI DMA controller Reference Guide.

[2] Accellera organization, June 2011. universal verification methodology(UVM) 1.1 Class Reference.

[3] Glasser M, February 4, 2011. UVM: The Next Generation in Verification Methodology, Methodology Architect, Courtesy of Mentor Graphics Corporation.

[4] Young-nam Yun, Jae-Beom Kim, Nam-Do Kim, Byeong Min, 2011. Beyond UVM for Practical Soc verification, IEEE- 978-1-4577-0711-7, pp 158-162.

[5] Juan Francesconi, J. Agustin Rodriguez, Pedro M. Julian, 2014. UVM Based testbench architecture for unit Verification. ISBN: 978-987-1907-86-1 IEEE Catalog Number CFP1454E-CDR.

[6] https://opencores.org/projects/dma_axi

[7] G. Ma, H. He,Design and implementation of a advanced DMA controller on AMBA-based SoC, in 2009 IEEE 8th International Conference on ASIC (2009), pp. 419–422.

[8] K. Chen, L. Qi, H. Yu, Design of two dimensionDMAcontroller inmedia multi-processor SoC, in 2008 Second International Symposium on Intelligent Information Technology Application (2008), pp. 708–711.

[9] C. Yu, C. Liu, C. Kang, T.Wang, C. Shen, S. Tseng, An efficientDMAcontroller formultimedia application inMPUbasedSOC, in2007 IEEE International Conference on Multimedia and Expo (2007), pp. 80–83

[10] Y.J.M. Shirur, K.M. Sharma, A. Aishwarya, Design and implementation of efficient direct memory access (DMA) controller in multiprocessor SoC, in 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS) (2018), pp. 1–6.