IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Full Chip Verification of Direct Memory Access Controller for Microcontroller SoC

Darshan S M
Dept. of ECE
R V College of Engineering
Bengaluru

Kariyappa B.S
Dept. of ECE
R V College of Engineering
Bengaluru

Sachin Kirdat
Technical Staff Engineer-Verification Microchip
Technology Pvt. Ltd Bengaluru

***Abstract*—This article focuses on the functional verification of a DMA controller within a microcontroller system-on-chip (SoC) using UVM. DMA is an integral part to modern computer systems, enhancing performance by offloading data transfer tasks from the CPU. The proposed work employs the Universal Verification Methodology (UVM) to develop a comprehensive verification environment that includes essential components like drivers, monitors, scoreboards, and sequencers. The verification covers various data transfer modes (fixed-to-fixed, fixed-to-block, block-to-fixed, and block-to-block), boundary conditions, and error cases to ensure the functional correctness of the DMA controller. Different code coverage was obtained such as FSM coverage of 100%, Expression coverage of 90.13%, Conditional coverage of 93.33%, and Statements coverage of 99.34% which leads to overall code coverage of 86% for the DMA Controller. Advanced SystemVerilog features such as assertions, coverpoints, and covergroups are incorporated into the testbench to improve its effectiveness. The proposed work also demonstrates successful verification through detailed test cases, validating the DMA controller's functionality and providing a robust basis for future enhancements in SoC design.

***Index Terms*—UVM, DMA, coverage**

I. Introduction

In the ever-evolving landscape of semiconductor design, verifying the functionality and correctness of complex in- tegrated circuits (ICs) has become increasingly challenging. Conventional test stimulus with Verilog can no longer meet the needs of verifying the chip, IC verification cycle accounts for 2/3 of the entire IC design cycle, often failing due to insufficient verification [1] . Universal Verification Method- ology (UVM) emerges as a standardized solution to address these challenges.UVM promotes a methodology-driven ap- proach to verification, emphasizing the use of object-oriented programming (OOP) principles and standardized methodolo- gies to build modular, configurable, and self-checking test- benches. while conventional verification methods are simpler and quicker to set up for smaller designs, UVM offers signifi- cant benefits in terms of reusability, scalability, automation, and overall robustness, making it the preferred choice for complex and large-scale verification projects [3] . In com- puter architecture and digital systems design, Direct Memory Access (DMA) is a critical mechanism that enhances the efficiency of data transfer between peripheral devices and main memory without involving the CPU. Traditional data transfer methods often rely on the CPU to manage data movement, which can lead to inefficient utilization of CPU resources and increased latency [5] .DMA addresses these limitations by offloading the data transfer tasks from the CPU to dedicated DMA controllers, thereby freeing up the CPU to perform other computational tasks. This parallelism in data transfer significantly improves system performance effectively. The advantage of employing a DMA controller is the improved efficiency of data transfers. The DMA controller streamlines data movement between devices and memory, ensuring that transfers are error-free. It also supports burst transfers, where multiple data blocks are transferred in a single operation, which reduces the overall transfer time. DMA controllers are especially beneficial in systems that process large amounts of data, such as video and audio streaming applications. In these scenarios, a DMA controller ensures smooth and efficient data transfers, preventing lag and interruptions [8] . In conclusion, integrating a DMA controller into your system offers numerous benefits, including enhanced performance, lower CPU utilization, and greater data transfer efficiency. It is a vital component for modern systems that handle large data volumes and can significantly optimize system performance

II. Motivation and related work

The motivation for using UVM (Universal Verification Methodology) to verify DMA (Direct Memory Access) con- trollers stems from the critical role DMA plays in enhancing system performance by offloading data transfer tasks from the CPU. Ensuring the reliability and efficiency of DMA controllers is vital, as they facilitate direct memory access for peripherals, leading to faster data processing and reduced CPU overhead.

UVM provides a robust and standardized framework for functional verification, which is essential for the complex nature of DMA controllers. With UVM [10] , verification engineers can create reusable test benches and components, improving verification efficiency and coverage. The method- ology's object-oriented approach allows for scalable and mod- ular test environments, which are crucial for handling the intricate interactions and various modes of DMA operations. By leveraging UVM, verification teams can systematically and thoroughly verify DMA controllers against a compre- hensive set of scenarios and edge cases. This ensures that the DMA controller operates correctly across all intended use cases, including different data transfer sizes, types, and speeds. Ultimately, using UVM for DMA controller verification enhances the robustness, reliability, and performance of the system, ensuring it meets the high standards required for modern computing applications.

A detailed literature review has been done to get an overview and understanding of the current state of knowledge, gaps, and trends in the field, as well as to establish the theoretical and conceptual framework for a research project. It involved reviewing and critically evaluating various sources, identifying common themes or patterns, and highlighting the contributions and limitations of previous studies.

A novel DMA controller design optimized for low-area and low-power applications, specifically tailored for RISC-V microcontrollers. Introduces a DMA controller architecture that minimizes the silicon area required [11].A testbench architecture for a Direct Memory Access (DMA) memory system that utilizes the AXI protocol and employs the Universal Verification Methodology (UVM) harness technique for design verification. Proposes using the UVM harness technique to create a robust and reusable testbench architecture. This approach leverages the modularity and reusability features of UVM, making it easier to develop and maintain the test environment [12]. The design and implementation of a reconfigurable Direct Memory Access (DMA) controller aimed at enhancing the performance of high-performance computing (HPC) systems. The DMA controller is designed to be reconfigurable, allowing it to adapt to various data transfer requirements and application needs. This flexibility makes it suitable for a wide range of HPC applications [16].

The inference is that UVM's structured and modular approach streamlines the creation of verification environments for DMA controllers. By utilizing reusable components and standardized procedures, UVM reduces the time and effort required to develop and execute verification tests.UVM facilitates thorough coverage by allowing detailed monitoring and checking of DMA operations. This ensures that all functional aspects and corner cases of the DMA controller are thoroughly verified, improving the reliability of the design

III. Overview of DMA Controller

Figure 1 Shows the Architecture of DMA Controller and the basic workings of it is be briefed next

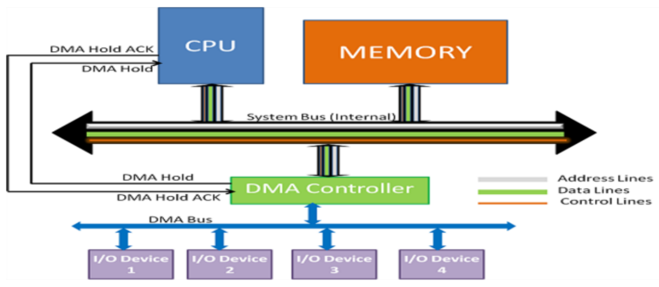

Fig. 1. DMA Controller [23]

A Direct Memory Access (DMA) controller allows peripheral devices to transfer data to and from memory without involving the CPU for each transfer. This helps improve the overall system performance by freeing up the CPU to perform other tasks while the data transfer is handled independently by the DMA controller. Figure 1 shows the working of the DMA Controller.

It involves the following steps

The CPU initializes the DMA controller by providing it with the necessary information such as the source address, the destination address, and the size of the data to be transferred. When an I/O device needs to transfer data, it sends a DMA request to the DMA controller.The DMA controller sends a DMA Hold signal to the CPU, requesting control of the system bus to perform the data transfer. Upon receiving the DMA Hold signal, the CPU completes its current operation, acknowledges the request, and then relinquishes control of the system bus by sending a DMA Hold Acknowledge (ACK) signal back to the DMA controller. Once the DMA controller receives the DMA Hold Acknowledge, it takes control of the system bus. This includes the address lines, data lines, and control lines necessary for the data transfer. The DMA controller starts transferring data between the memory and the I/O device. This transfer is done directly through the system bus, bypassing the CPU. Once the data transfer is complete, the DMA controller releases the system bus. It signals the CPU that the transfer is complete by deactivating the DMA Hold signal. After the DMA controller releases the bus, the CPU resumes its operations from where it left off.

IV. Design Methodology

The DUT(DMA) is the sub-system that is being verified and evaluated in a UVM environment. The test performs various functional and performance tests to ensure the device operates correctly and meets the required specifications. This will involve validating its compatibility, checking for proper data transmission from source to destination.

A. *Connectivity Check for DMA*

Connectivity checks are a crucial aspect of full chip verification. It is a first step before performing the Verification to ensure the input connections are proper according to the design. Connectivity checks verify that all the nets (electrical connections) in the chip design are connected as per the schematic or the intended design. This ensures that signals are routed correctly and that there are no unintended open circuits or shorts. The correct connectivity is essential for the chip to perform its intended functions. Any errors in connectivity can lead to functional failures, which can compromise the performance of the chip or the system it is part of.

The process involves these steps. The first action is to enable the DMA (Enable = 1). which sets the enable flag. The next step is to force the source (SRC) to 1 in another way the drivers of the DMA will be triggered to 1 and the destination (DST) is expected to be 1, in another way the inputs of the DMA is driven to 1 will be checked. If DST is 1, it proceeds

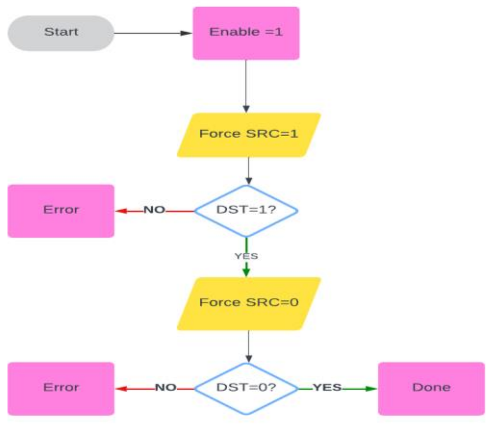

Fig. 2. Flowchart for Connectivity check

to force the source (SRC) to 0, and then the destination (DST) will be checked if 0 is driven. If all conditions are satisfied, the process moves to the Done state, indicating the check is complete.

B. *UVM testbench Architecture for DUT*

The Figure 3 shows a UVM testbench environment that interacts with the Design Under Test (DUT), which is a microcontroller SoC with a DMA controller inside.

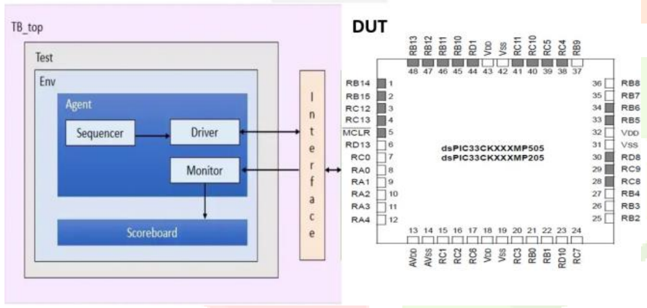

Fig. 3. UVM environment for DUT [24]

Components of the UVM Environment:

Test: The top-level UVM component that defines the overall verification scenario.

Env: The environment component that contains all other UVM components.

Agent: Encapsulates the driver, sequencer, and monitor for a specific interface.

Sequencer: Generates sequences of transactions.

Driver: Drives the transactions to the DUT.

Monitor: Observes the signals from the DUT and extracts information.

Scoreboard: Compares the actual output of the DUT with the expected output.

Here the DUT will be a microcontroller SoC as full chip verification is carried on. The stimulus will be passed through microcontroller pads into the DMA Controller.The outputs from DMA will be transferred to scoreboard via monitor and the comparison of it takes place with the reference data.

Three main test cases of DMA are used for the verification of the DMA Controller followed by code coverage.

The following test cases are discussed in later subsections

1) DMA channel trigger testcase
2) DMA idle mode testcase
3) DMA idle mode sidl testcase

C. *Testbench flow for DMA channel trigger testcase*

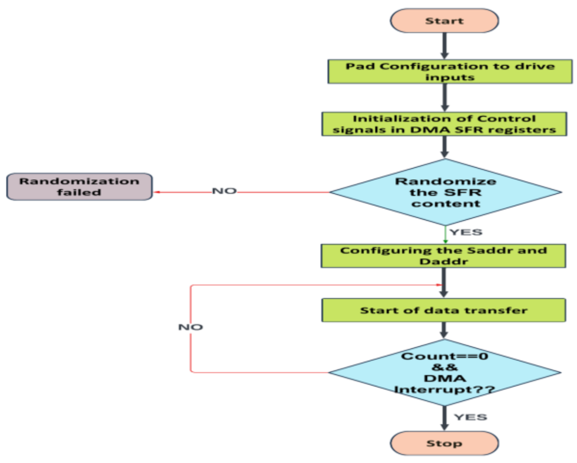
Start
Pad Configuration to drive inputs
Initialization of Control signals in DMA SFR registers
Randomize the SFR content
NO
Randomization failed
YES
Configuring the Saddr and Daddr
Start of data transfer
Count==0 && DMA Interrupt??
NO
YES
Stop

Fig. 4. Flowchart of sequence for trigger test case

This figure 4 describes a flowchart that illustrates the steps in a UVM sequence for verifying a DMA controller, focusing specifically on a DMA channel trigger test case

Below are the detailed explanation of each step:

Configure the physical pad (PAD) to drive the required input signals that might include triggers or other initial conditions necessary to start the DMA operation. Initialize the DMA Special Function Registers (SFR) with control signals such as DMA.ON, chansel,DMACNT ,CHAEN ,SIZE [1:0], TRMODE ,SAMODE ,DAMODE ,DMALOW,DMAHIGH; ,MIDVALUE.Randomize the contents of the SFR registers to ensure thorough testing with different configurations. Handle the case where randomization of the SFR content fails. The purpose is to ensure that the test does not proceed with incorrect or incomplete configuration. Configure the source (saddr) and destination (daddr) addresses for the DMA transfer.The registers related to the source and destination register will be randomized with some conditions. Initiate the data transfer process from source address to the destination address. Monitor the transfer and check if the transfer count has reached zero and if a DMA interrupt has been generated, if so then the data transfer is successfully completed.

D. *testbench flow for DMA idle mode testcase*

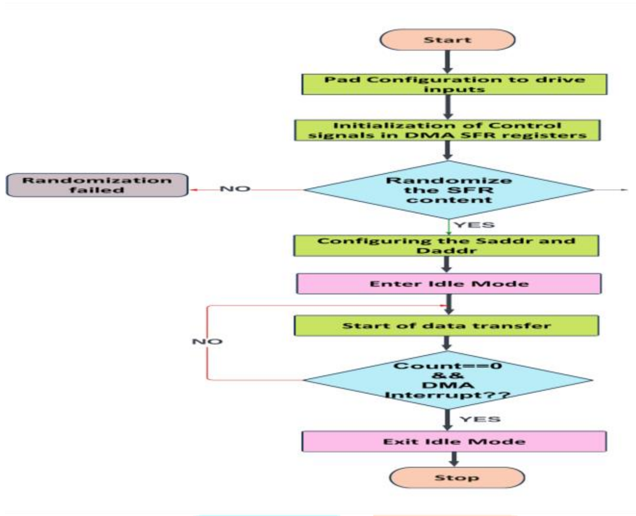
Start
Pad Configuration to drive inputs
Initialization of Control signals in DMA SFR registers
Randomize the SFR content
NO
Randomization failed
YES
Configuring the Saddr and Daddr
Enter Idle Mode
Start of data transfer
NO
Count==0 && DMA Interrupt??
YES
Exit Idle Mode
Stop

Fig. 5. flowchart of sequence for idle mode test case

This figure 5 describes the sequence of steps in a UVM testbench for verifying a DMA controller with a focus on the idle state testcase. This testcase is crucial for ensuring that the DMA controller can handle scenarios where it enters and exits an idle state properly.

Below are the detailed explanation of each step: Configure the physical pad (PAD) to drive the required input signals that might include triggers or other initial conditions necessary to start the DMA operation. Initialize the DMA Special Function Registers (SFR) with control signals such as DMA.ON, chansel,DMACNT ,CHAEN ,SIZE [1:0], TRMODE ,SAMODE ,DAMODE ,DMALOW,DMAHIGH; ,MIDVALUE.Randomize the contents of the SFR registers to ensure thorough testing with different configurations. Handle the case where randomization of the SFR content fails. The purpose is to ensure that the test does not proceed with incorrect or incomplete configuration.Write the necessary data into the RAM memory that will be used during the DMA transfer.The Purpose is to Prepare the memory with data that needs to be transferred by the DMA. Configure the source (saddr) and destination (daddr) addresses for the DMA transfer.The registers related to the source and destination register will be randomized with some conditions.After the DMA gets bus control the CPU enters into idle modeand the DMA starts data transfer.Initiate the data transfer process from source address to the destination address. Monitor the transfer and check if the transfer count has reached zero and if a DMA interrupt has been generated, if so then the data transfer is successfully completed.

V. Results and Discussion

A. *Simulation Results of connectivity check between DMA and ADC*

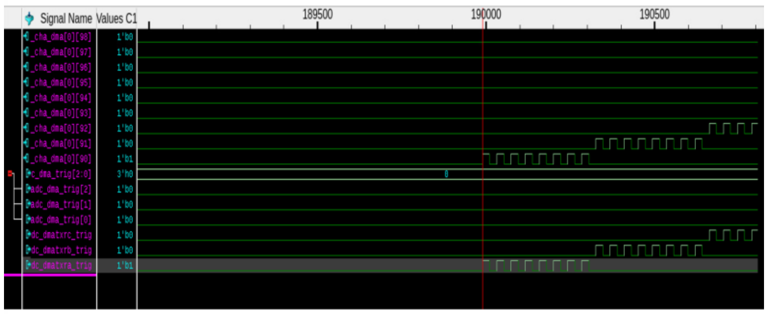

Fig. 6. Connectivity check between DMA and ADC

Figure 6 shows the simulation result of the connectivity check between DMA and ADC macro .adc_rx_trig are the drivers for the DMA input signal dma_chan[90:92] . The process involves toggling the source signal and verifying the corresponding state of the destination signals to ensure they match the expected values.It is seen when the adc signals are driven from 1 to 0 , the dma_chan signals also made the same transition.The waveforms indicate successful connectivity, as all the destination signals correctly follow the source signal's state changes.

B. *connectivity check between DMA and MCCP macro*

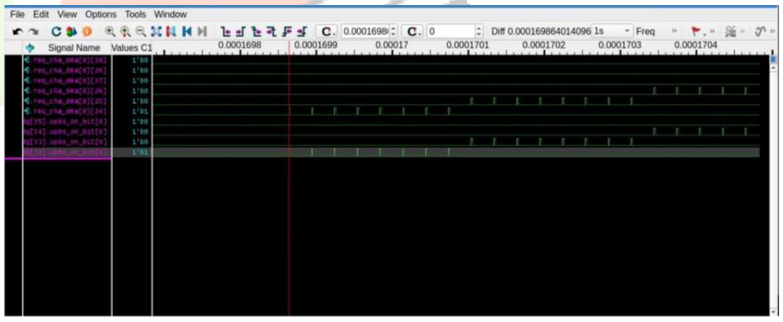

Fig. 7. Connectivity check between DMA and MCCP

Figure 7 shows the simulation result of connectivity check between DMA and MCCP macro . upbs_on_bit from MCCP macro are the drivers for the DMA input signal dma_chan[27:24] .The process involves toggling the source signal and verifying the corresponding state of the destination signals to ensure they match the expected values.It is seen when the mccp signals are driven from 1 to 0 , the dma_chan signals also made the same transition.The waveforms indicate successful connectivity, as all the destination signals correctly follow the source signal's state changes.

C. Simulation Results for DMA channel trigger test case

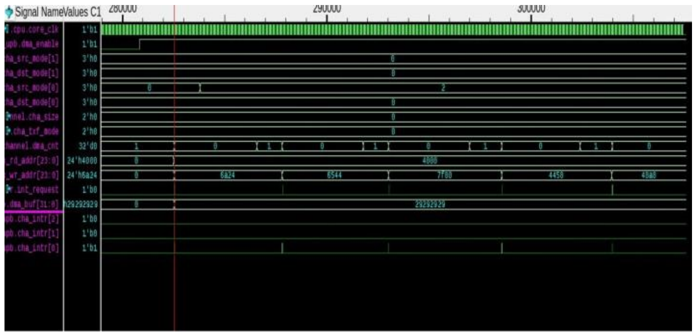

Fig. 8. Simulation result of DMA channel trigger test case

Figure 8 shows the simulation result for DMA channel trigger test case, The signal cpu_core_clk provides the clock for the DMA operation.dma_enable Set to '1' to enable the DMA operation.int_pulse[0] is connected to chan_req_dma which generates a trigger pulse for the DMA. The cha_intr[2:0] signals are the interrupt lines for the DMA channels. These signals show the status of DMA channel interrupts. When upb.cha_intr[0] goes high, it indicates that the DMA channel 0 interrupt has been triggered, initiating the transfer. The signal cha_dma[0][0] which is channel0 1st bit is set to 1 indicating a trigger generated by a peripheral.DMA_COUNT shows the amount of data packets available for data transfer .bmx_rd_addr is the address pointing to the peripheral's data and bmx_wr_addr represents the address of the memory(RAM) to which the data is supposed to be moved. Once the DMA gets access to the system bus it starts the data transfer from the read address to the write address, The COUNT value starts decrementing, for each successful data transfer the done bit will be Set to 1.dma_buffer is a buffer that stores the data to be transferred from the peripheral to memory. After each data transfer an interrupt int_request will be generated by the DMA to indicate to the CPU that the data transfer is completed and it can take back the system bus ,chan_intr[0] shows the interrupt generated by the DMA

D. Simiulation Results for DMA idle mode test case

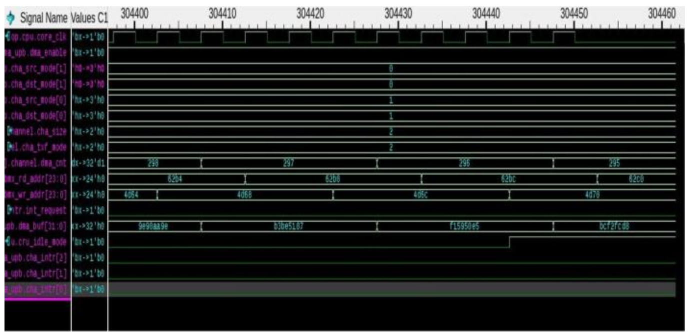

Fig. 9. Simulation of DMA idle mode testcase

The DMA transfer begins with the dma_enable signal going high, enabling the DMA operation. The core clock drives the timing of this process. Initially, src_mode,dst_mode,txf_mode will be set as discussed above. Beginning with the memory RAM will be filled with data to perform the memory-to-memory data transfer.The DMA count (channel.dma_cnt) starts at 300, indicating the total number of data packets to be transferred. As the transfer proceeds, this count decreases with each packet moved. The addresses (rd_addr and wr_addr) increment appropriately to reflect the next memory locations to be accessed. Once the DMA starts with data transfer the CPU enters into idle mode and it could also be seen the cpu_core_clk is inactive during this mode to save power.Once the channel.dma_cnt reaches 0, all data packets have been transferred from the source to the destination. At this point, the int_request signal is asserted, signaling to the CPU that the DMA transfer is complete. The CPU can now regain control of the system bus, allowing normal operations to resume.The figure 10 shows as the count reaches zero DMA generates the interrupt to CPU and thus CPU exiting from the idle mode.

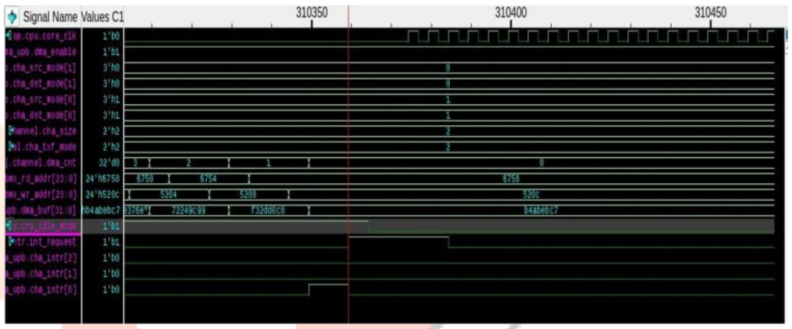

Fig. 10. Simulation of DMA idle mode testcase

E. Simulation of DMA idle mode sidle test case

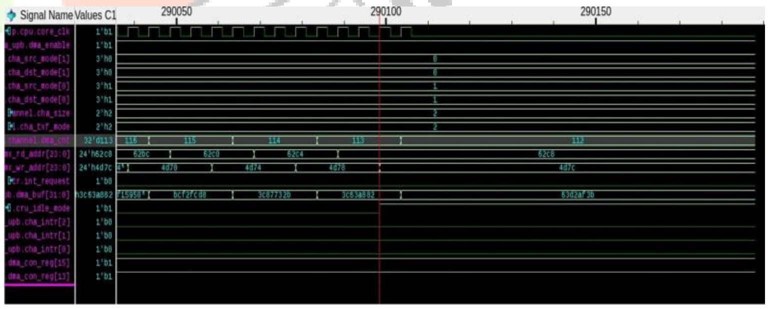

Fig. 11. Simulation of DMA idle mode sidle test case

Figure 11 shows the simulation results for DMA idle mode sidl test case.The type of data transfer method is a memory to memory type,The DMA transfer begins with the dma_enable signal going high, enabling the DMA operation. The core clock drives the timing of this process. Initially, src_mode,dst_mode, and txf_mode will be set as discussed above. Beginning with the memory RAM will be filled with data to perform the memory-to-memory data transfer.The

DMA count (channel.dma_cnt) starts at 300, indicating the total number of data packets to be transferred. con_reg[13] which is the sidl bit will be set to 1 indicates the DMA will stops working in the idle mode.As the transfer proceeds, this count decreases with each packet moved. The addresses (rd_addr and wr_addr) increment appropriately to reflect the next memory locations to be accessed. Once the DMA starts with data transfer the CPU enters into idle mode and it could also be seen the cpu_core_clk is inactive during this mode to save power.Once the CPU comes back to normal mode the DMA starts the data transfer where it left out in the idle mode .Once the channel.dma_cnt reaches 0, all data packets have been transferred from the source to the destination. At this point, the int_request signal is asserted, signaling to the CPU that the DMA transfer is complete. The CPU can now regain control of the system bus, allowing normal operations to resume.The Figure 12 shows as the count reaches zero DMA generates the interrupt to CPU and thus CPU exiting from the idle mode.

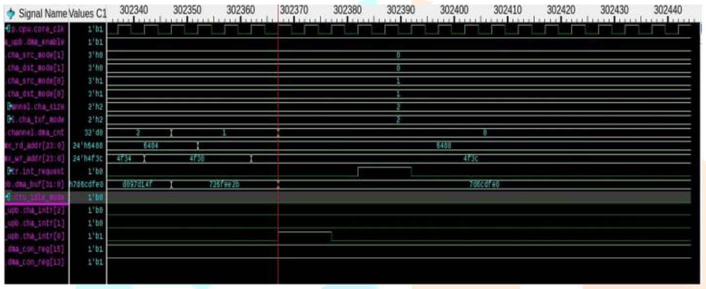

Fig. 12. Simulation of DMA idle mode testcase

F. Code Coverage

Table I shows 100% of toggle coverage which is obtained after excluding the tied high and tied low signals

The toggle coverage report indicates that the connectivity check for the DMA was successful, with all relevant signals achieving 100% toggle coverage. This comprehensive coverage ensures that the DMA's signal transitions are functioning correctly, providing confidence in the DMA's connectivity and basic operational reliability within the system

TABLE I
TOGGLE COVERAGE OF CONNECTIVITY CHECK

Signal	Coverage
last_pb_clk	100%
pchaen_set[0-7]	100%
req_cha_dma[0][0-76]	100%
req_cha_dma[0][89-118]	100%
req_cha_dma[1][0-76]	100%
req_cha_dma[1][89-118]	100%
req_cha_dma[2][0-76]	100%
req_cha_dma[2][89-118]	100%
req_cha_dma[3][0-76]	100%
req_cha_dma[3][89-118]	100%
req_cha_dma[4][0-76]	100%
req_cha_dma[4][89-118]	100%
req_cha_dma[5][0-76]	100%
req_cha_dma[5][89-118]	100%
req_cha_dma[6][0-76]	100%
req_cha_dma[6][89-118]	100%
req_cha_dma[7][0-76]	100%
req_cha_dma[7][89-118]	100%
sys_clk	100%
trg_done[0-7]	100%
Toggle Coverage Summary	**100%**

TABLE II
CODE COVERAGE FOR DUT

Coverage type	Coverage
Branches	77.81%
Conditions	93.33%
Expressions	90.13%
FSM states	100%
Statements	99.34%
Toggles	55.13%
Coverage Summary	**86%**

Table II provides a detailed summary of the code coverage for the Design Under Test (DUT) which is obtained of 86%. It shows various coverage metrics including branches, conditions, expressions, FSM (Finite State Machine) states, statements, and toggles.

The overall design unit coverage summary indicates a coverage of 85.96% when considering all factors. This lower overall coverage percentage is due to the fact that only three test cases were utilized. Additional test cases would be needed to achieve higher coverage and ensure thorough testing of all design aspects.

VI. CONCLUSION

Successfully developed a comprehensive UVM-based verification environment tailored for the DMA controller. This included creating detailed test plans, developing test benches, and implementing various verification components such as drivers, monitors, and scoreboards. verified the functional correctness of the DMA controller by validating all major functionalities, including different types of data transfers (fixed-to-fixed, fixed-to-block, block-to-fixed, and block-to-block). This ensures the DMA controller performs as intended across all supported operations. Performed connectivity check for DMA and successfully verified the connections. Obtained toggle coverage of 100% for connectivity check test case Achieved coverage of 86% by simulating a wide range of scenarios, including normal operation, boundary conditions, and error cases. This ensured that the DMA controller operated correctly under various conditions and met the design specifications.

REFERENCES

[1] M. Ismael, A. Hroub, and A. Abu-Issa, "Autg: An automatic uvm-based testbench generator for vlsi chip design verification," in 2023 Interna- tional Conference on Mi- croelectronics (ICM), 2023, pp. 162–167. doi: 10.1109/ICM60448.2023.10378885

[2] N. B. Harshitha, Y. G. Praveen Kumar, and M. Z. Kurian, "An intro- duction to universal verification methodology for the digital design of integrated circuits (ic's): A review," in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 1710–1713. doi: 10.1109/ICAIS50930.2021.9396034.

[3] J. Wang, S. Geng, X. Peng, X. Li, Q. Sun, and P. Li, "Dma function verification based on uvm verification platform," in 2021 6th Interna- tional Conference on Integrated Circuits and Microsystems (ICICM), 2021, pp. 276–279. doi: 10.1109/ ICICM54364.2021.9660239.

[4] S. Qamar, W. Haider, M. Anwar, F. Azam, and M. Khan, "A comprehensive investigation of universal verification methodology (uvm) standard for design verification," Feb. 2020, pp. 339–343. doi: 10.1145/3384544.3384547.

[5] H. Morales, C. Duran, and E. Roa, "A low-area direct memory access controller architecture for a risc-v based low-power microcontroller," in 2019 IEEE 10th Latin American Symposium on Circuits Systems (LASCAS), 2019, pp. 97–100. doi: 10.1109/LASCAS.2019.8667579.

[6] Anjali and J. P. Anita, "Axi based dma memory system testbench architecture using uvm harness technique," in 2019 9th International Conference on Advances in Computing and Communication (ICACC), 2019, pp. 152–157. doi: 10.1109/ICACC48162.2019.8986181.

[7] D. Lohmann, F. Maziero, E. J. dos Santos, and D. Lettnin, "Extending universal verification methodology with fault injection capabilities," in 2018 IEEE 9th Latin American Symposium on Circuits Systems (LASCAS), 2018, pp. 1–4. doi: 10.1109/LASCAS.2018.8399945.

[8] H. K. Nguyen, K. P. Dong, and X.-T. Tran, "A reconfigurable multi- function dma controller for high-performance computing systems," in 2018 5th NAFOSTED Conference on Information and Computer Sci- ence (NICS), 2018, pp. 344–349. doi: 10.1109/NICS.2018.8606841.

[9] N. Georgoulopoulos, I. Giannou, and A. Hatzopoulos, "Uvm-based verification of a mixed-signal design using systemverilog," in 2018 28th International Symposium on Power and Timing Modeling, Optimiza- tion and Simulation (PATMOS), 2018, pp. 97–102. doi: 10.1109/PAT- MOS.2018.8464148.

[10] Y. Chi and Z. Zheng, "A design of direct memory access controller for wireless communication soc in power grid," in 2018 IEEE 2nd International Conference on Circuits, System and Simulation (ICCSS), 2018, pp. 76–79. doi: 10 . 1109 / CIRSYSSIM.2018.8525958.

[11] Y. J. M. Shirur, K. M. Sharma, and A. A, "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. doi: 10.1109/IC- NEWS.2018.8903991.

[12] T. M. Pavithran and R. Bhakthavatchalu, "Uvm based testbench architec- ture for logic sub-system verification," in 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), 2017, pp. 1–5. doi: 10.1109/ TAPENERGY.2017.8397323.

[13] F. Shanehsazzadeh and M. S. Sadri, "Area and performance evaluation of central dma controller in xilinx embedded fpga designs," in 2017 Iranian Conference on Electrical Engineering (ICEE), 2017, pp. 546–550. doi: 10 . 1109 / IranianCEE . 2017.7985100.

[14] C. Sharma and D. K. Chauhan, "High performance low power ahb dma controller with fsm decomposition technique," in 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017, pp. 456– 461. doi: 10.1109/ICPCSI.2017.8392337.

[15] A. Fiergolski, "Simulation environment based on the universal ver- ification methodology," Journal of Instrumentation, vol. 12, pp. C01001–C01001, Jan. 2017. doi: 10.1088/1748-0221/12/01/C01001.

[16] M. Mefenza, F. Yonga, and C. Bobda, "Automatic uvm environment generation for assertion-based and functional verification of systemc designs," in 2014 15th International Microprocessor Test and Verification Workshop, 2014, pp. 16–21. doi: 10.1109/MTV.2014.10.

[17] K. Salah, "A uvm-based smart functional verification platform: Concepts, pros, cons, and opportunities," in 2014 9th Interna- tional Design and Test Symposium (IDT), 2014, pp. 94–99. doi: 10.1109/IDT.2014.7038594.

[18] X. Jin-f, "Adopting universal verification methodology to achieve reusability and automation verification," Microelectronics Computer, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:63600370.

[19] W. Zhong-hai and Y. Yi-zheng, "The improvement for transaction level verification functional coverage," in 2005 IEEE International Sympo- sium on Circuits and Systems (ISCAS), 2005, 5850–5853 Vol. 6. doi: 10.1109/ISCAS.2005.1465969.

[20] A. O. Naik, E. Kuruvilla, and A. P. Chavan, "Integration and verification of ip cores on soc," in 2021 IEEE Mysore Sub Section International Conference (MysuruCon), 2021, pp. 120–125. doi: 10.1109/Mysuru- Con52639.2021.9641547.

[21] G. P, "A system verilog approach for verification of memory controller," Interna- tional Journal of Engineering Research and, vol. V9, Jun. 2020. doi: 10.17577/ IJERTV9IS050876.

[22] G. Madhura and R. Holla, "Uvm based design verification of inter- connect block," in Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems, V. Nath and J. K. Mandal, Eds., Singapore: Springer Singapore, 2021, pp. 559–569, isbn: 978-981-16-0275-7

[23] https://witscad.com/course/computer-architecture/chapter/dma-controller-and-io-processor

[24] https://vlsiverify.com/uvm/uvm-test/

[25] Amruth N and Kariyappa B S, "Verification of GDDR6 DRAM Features Using System Verilog and UVM" International Journal of Advances in Engineering and Management (IJAEM), ISSN: 2395-5252, pp: 301-305, Volume 4, Issue 9, September 2022.

[26] Bheema. T, Dr. Kariyappa. B. S "Developing Inter-Integrated Circuit Master and Slave Universal Verification Component using UVM" Inter- national Journal of Engineering Technology, Management and Applied Sciences, Volume 3, Issue 6, PP:252-256, ISSN 2349-4476, July 2015.