OCP – UART IP Environment using UVM Verification

Ms. Pooja J. Rajput[1], Mr. Prashant D. Karandikar[2], Mr. Gardas Naresh Kumar[3]

[1] *Research Scholar, GTU PG School, Gujarat, India*
[2] *Aurora Copper Systems LLP, Pune, Maharashtra, India*
[3] *Course Co-Coordinator, CDAC ACTS, Maharashtra, India*

ABSTRACT

This paper presents OCP-UART IP Environment using UVM Verification. As verification of the design is become most difficult task, so that verification is significant part for reaching time to market. UVM methodology reduces the time to develop verification IP by using previously built in base classes for all the required component from the UVM library and also reusing the VIP environment at different level of construct. The Open Core Protocol (OCP) Verification IP is already made highly configurable UVM verification environment suitable for design under test with OCP Interface and OCP Verification IP can generate stimuli in an OCP bus format.

Keywords: - *OCP, UART, UVM.*

1. Introduction

Now a days, with the fast development of Integrated circuit, the difficulty of the digital IC design is increasing and difficult to verify, so that development of verification methodology makes it possible to complete and effectively improve the verification efficiency. UVM is introduced from OVM (Open Verification Methodology) and VMM (Verification Methodology Manual) features. UVM is a methodology for functional verification that uses TLM standard for communication between blocks and System Verilog for its language and also it uses SV for creating components and TLM for interconnects between components. The UVM class library brings much automation to the System Verilog language such as sequences and data automation features like packing, copy, and compare. Methodology is a set of base class library which we can use to build our test bench.

2. OPEN CORE PROTOCOL

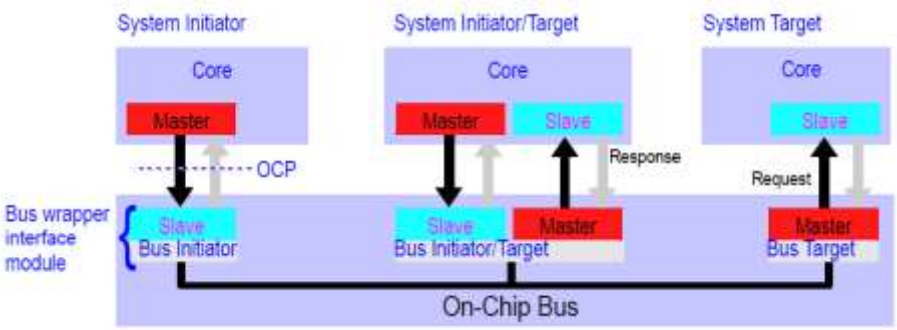

Fig. 1.Basic block diagram of OCP instance. [8]

In the block diagram, the basic operation and characteristics of OCP is explained. The OCP represents a point-to-point interface between two communicating entities such as bus interface modules and IP cores. One entity doing as

the master of the OCP instance, and another as the slave. Master can present commands and controlling entity. The slave responds to commands presented to it, either by accepting data from the master, or presenting data to the master. For two entities to communicate there need to be two instances of the OCP connecting each other such as master and slave. System consisting a wrapped bus and three IP core entities such as a system initiator, system initiator/target, and system target. The specific of the IP core decides whether the core must master, slave, or both sides of the OCP and the wrapper interface modules must act as the parallel side of the OCP for each connected entity.

2.1 OCP signals

OCP interface signals are divided into dataflow, sideband, and test signals. The dataflow signals are divided into basic signals, simple extensions, burst extensions, tag extensions, and thread extensions.

TABLE I. BASIC SIGNAL

Name	Width	Driver	Function
Clk	1	varies	Clock input
EnableClk	1	varies	Enable OCP clock
MAddr	configurable	master	Transfer address
MCmd	3	master	Transfer command
MData	configurable	master	Write data
MDataValid	1	master	Write data valid
MRespAccept	1	master	Master accepts response
SCmdAccept	1	slave	Slave accepts transfer
SData	configurable	slave	Read data
SDataAccept	1	slave	Slave accepts write data
SResp	2	slave	Transfer response

3. UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

In UART, it contain the 9 data bits mode (start bit + 8 data bits + parity + stop bits) or FIFOs for the receiver/transmitter data buffering. The transmitter implements parallel to serial conversion on the 8-bit data received from the CPU. The receiver implements serial-to-parallel conversion on the asynchronous data frame received from the serial data input.

3.1. Packet Structure

The start bit is active low signal and the stop bit is active high signal. And rest of six bits are of data bits and a parity bit. The parallel data from CPU is transformed to serial data by UART and then transmission access. Communication between more than two UARTs is based on asynchronous serial mode of transmission. Hand shaking between the UARTs is done using the synchronizing bits. Each character is sent as a start bit, a configurable number of data bits, an optional parity bit and one or more stop bits.

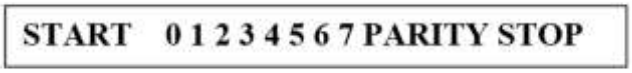

Fig. 2. Packet structure. [9]

TABLE II. UART SIGNAL

Receiver Interface		
SIN	Input	Receiver serial input
RxRDYn	Output	Receiver ready
Transmitter Interface		
SOUT	Output	Transmitter serial output
TxRDYn	Output	Transmitter ready
Modem Interface		
DCDn	Input	Data carrier detect
CTSn	Input	Clear to send
DSRn	Input	Data set ready
RIn	Input	Ring indicator
DTRn	Output	Data terminal ready
RTSn	Output	Request to send

TABLE III. UART REGISTER

Register Name	Offset	31-16	15-8	7	6	5	4	3	2	1	0
RBR (Receive buffer Register)/ THR (Transmit Holding Register)	0x0	—	—	D7	D6	D5	D4	D3	D2	D1	D0
IER (Interrupt Enable Register)	0x1			0	0	0	0	MSI	RLSI	THRI	RBRI
IIR (Interrupt Identification Register)	0X2			0	0	0	0	0	ID1	ID0	STAT
LCR (Line Control Register)	0x3			0	SB	SP	EPS	PEN	STB	WLS1	WLS0
LSR (Line Status Register)	0x5			0	TEMT	THRE	BI	FE	PE	OE	DR

4. UNIVERSAL VERIFICATION METHODOLOGY

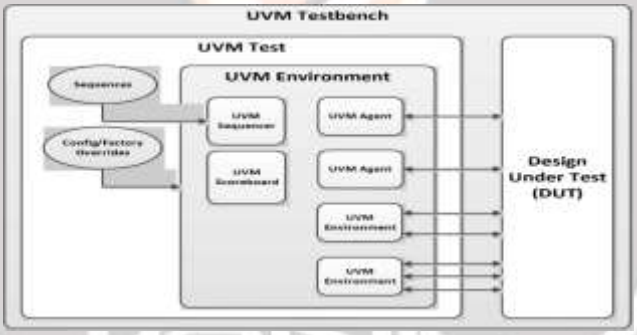

Fig. 3.UVM Test bench architecture. [7]

4.1. UVM Test bench

The UVM Test bench provides connections between Design under Test module and the UVM Test class and configures them.

4.2. UVM Test

The top level UVM Component is UVM Test in the UVM Test bench. The UVM Test performs the three main functions: Starts the Top level environment, configures the environment and applies stimulus by invoking UVM Sequences through the environment to the DUT.

4.3. UVM Environment

UVM Environment contains components are UVM Agents, UVM Scoreboards, or even other UVM Environments. UVM Environment covers all the verification components targeting the DUT.

4.4. UVM Scoreboard

The UVM Scoreboard's main function is to check the behavior of DUT. and then compares the expected output versus the actual output.

4.5. UVM Agent

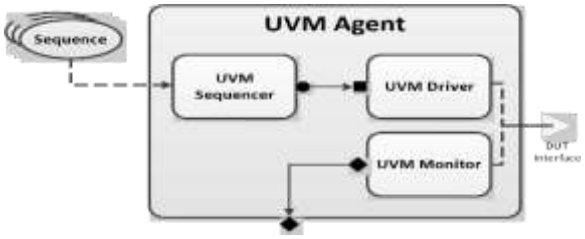

Fig. 4. UVM Agent. [7]

1) UVM Sequencer

The UVM Sequencer serves as an arbiter for controlling transaction flow from multiple stimulus sequences.

2) UVM Sequence

A UVM Sequence is an object that contains a behavior for generating stimulus.

3) UVM Driver

The UVM Driver receives separate UVM Sequence Item transactions from the UVM Sequencer and drives it on the DUT Interface.

4) UVM Monitor

The UVM Monitor samples the DUT interface and captures the information there in transactions that are sent out to the rest of the UVM Test bench for further analysis.

5. VERIFICATION ENVIRONMENT

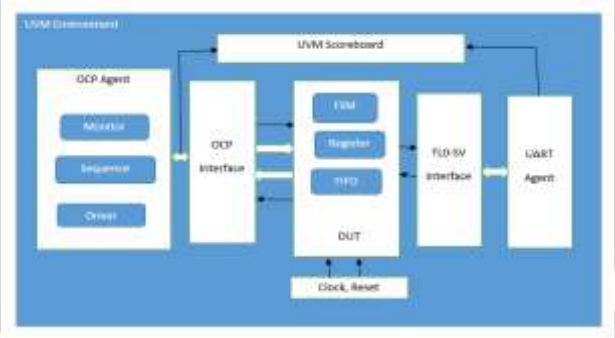

Fig. 5. OCP-UART IP Verification Environment

UART functionality issues like parity, missing data are evaluated by introducing clock jitter and stability related changes by changing clock frequency. From above fig.5 in the UART IP, Register handles toggle coverage and function coverage. RD, WR signal based on address decode and register bit actuates functionality. OCP Agent which will do RD, WR bus transaction.

The UVM Scoreboard's main functionality is to check the behavior of DUT. The UVM Scoreboard receives transactions carrying inputs and outputs of the DUT through UVM Agent analysis ports such as OCP Agent and UART Agent, runs the input transactions through model to produce expected transactions, and then compares the expected output versus the actual output.

The UVM Agent is a hierarchical component that are dealing with a specific DUT interface. A UVM Agent includes a UVM Sequencer to manage stimulus flow, a UVM Driver to apply stimulus on the DUT interface, and a UVM Monitor to monitor the DUT interface. UVM Agents consists of components, like coverage collectors, protocol checkers.

6. SIMULATION RESULTS

6.1. OCP Read transaction

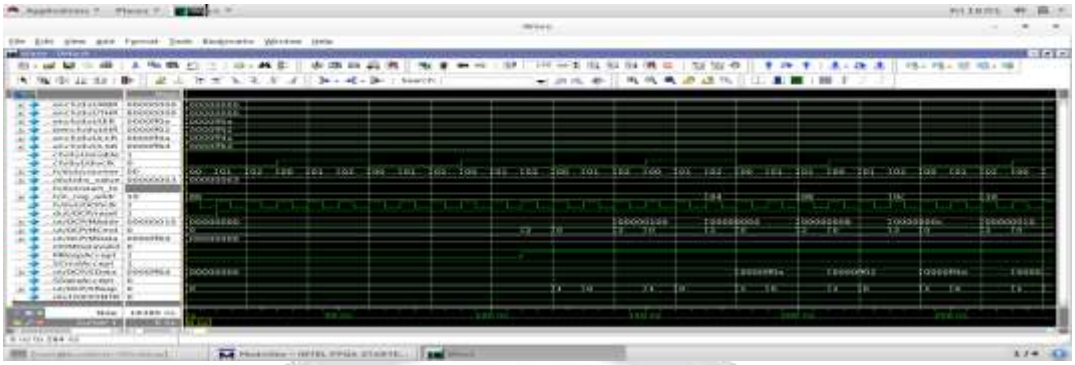

Fig. 6.Register read transaction.

6.2. OCP Write transaction

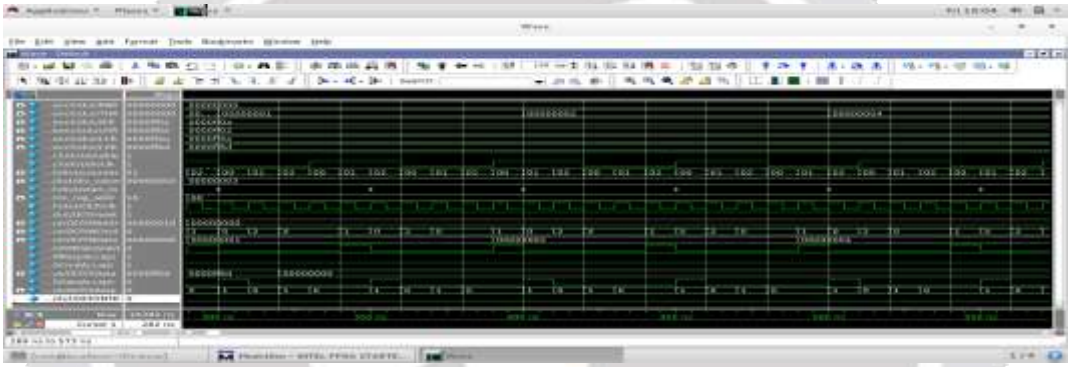

Fig. 7.Register write transaction.

7. CONCLUSION

Using this environment we were able to identify register reset value bugs with in the design using register read write test cases. OCP read write transaction has been perform using system Verilog.

8. REFERENCES

[1] SiddaramPatil, Arun Kumar, Dr.v.Venkateswarlu, "Implementation of Open Core Protocol transaction Verification using System Verilog UVM methodology," INTERNATIONAL JOURNAL OF SCIENTIFIC & ENGINEERING RESEARCH, VOLUME 5, ISSUE 9, and SEPTEMBER-2014.

[2] Shihua Zhang, Asif Iqbal Ahmed and Otmane Ait Mohamed, "A Re-Usable Verification Framework of Open Core Protocol (OCP)," Circuits and Systems and TAISA Conference, 2009. NEWCAS-TAISA '09. Joint IEEE North-East Workshop on.

[3] Wei Ni, XiaotianWang, "Functional Coverage-Driven UVM-based UART IP Verification," ASIC (ASICON), 2015 IEEE 11th International Conference on.

[4] Khaled Salah, "A UVM-Based Smart Functional Verification Platform: Concepts, Pros, Cons, and Opportunities," Design & Test Symposium (IDT), 2014 9th International.

[5] Wei Ni, Jichun Zhang, "Research of Reusability Based on UVM Verification," ASIC (ASICON), 2015 IEEE 11th International Conference on.

[6] YingkeGao, Diancheng Wu, Quanquan Li, Tiejun Zhang, ChaohuanHou, "Design and Implementation of Transaction Level Processor based on UVM," ASIC (ASICON), 2013 IEEE 10th International Conference on.

[7] uvm_users_guide_1.2.pdf

[8] "OCP specification 2.2. pdf"

[9] "UART Universal Asynchronous Receiver Transmitter User Guide.pdf"