Viney et al. / IJAIR Vol.2 Issue 7 ISSN: 2278-7844

Verification IP for Interconnect Verification

Viney Malik[1], Rajesh Mehra[1], Surender Ahlawat[2]

[1] *ECE Department, Sector – 26, NITTTR, Chandigarh, India*

[2] *Mentor Graphics, Noida, India*

***Abstract* - Verifying complex interconnects is always a real challenge as same environment should be reused at the system level as well as block level. In this article we will take an example for verifying OCP based interconnects but same approach can be moved forward to verify other protocol based Interconnects also.**

***Keywords* - TLM – Transaction level modelling, OCP – Open Core Protocol, OVM – Open verification methodology, GPIO – General purpose IO**

I. INTRODUCTION

The Open Core Protocol [4] defines a high-performance, bus-independent interface between IP cores that reduces design time, design risk, and manufacturing costs for SOC designs. The Open Core Protocol:

- Achieves the goal of IP design reuse. The OCP transforms IP cores making them independent of the architecture and design of the systems in which they are used
- Optimizes die area by configuring into the OCP only those features needed by the communicating cores
- Simplifies system verification and testing by providing a firm boundary around each IP core that can be observed, controlled, and validated.

OCP verification IP allows a complete Open Verification Methodology (OVM) compliant OCP system to be constructed where master and slave devices can communicate using transactions or signals. The OCP verification IP includes OVM components to translate between the OVM transactions and the OCP signals for OCP master and slave devices. The OVM transactions can be randomized with protocol coverage collected by the verification IP. The OCP verification IP also includes a set of stimulus tasks the user can use to create a test or device that issues OCP OVM transactions. OCP verification IP reduces the time taken to design and test any system.

Open Verification Methodology (OVM) [1] is a verification methodology that presented the development of verification environments targeted at verifying large IP-based SoCs. Verification productivity supports the ability to develop individual verification components quickly, encapsulate them into larger reusable verification components, and reuse them in different configurations and at different levels of abstraction. OVM supports “bottom-up” reuse by allowing block-level components and environments to be encapsulated and reused as blocks that can be composed into a system [2] [3].

There are few scenarios where verification components will be used while verifying Interconnect based system, like, to verify stand alone interconnect when no master/slave cores are ready, to verify individual master/slave cores, to verify interconnect when few master/slave cores are ready, to gather the TLM activity at all the interfaces for transaction displaying/functional coverage/transaction logging when all the master/slave RTL cores are connected with Interconnect i.e. verification component working in Passive mode.

II. INTERCONNECT VERIFICATION PROBLEM

While defining the architecture it must be taken care that an exhaustive test suits can be generated, functional coverage must be collected at all the interfaces, protocol compliance checks must be there at all the interfaces and most important it must be easy to reuse this environment at the system level when one have its master/slave cores ready.

This can be easily achieved with OCP verification IP, which works at any multiple abstraction levels, one only needs to replace TLM level master / slave with RTL master / slave once cores are ready. Verification IP is system verilog based OVM compliant verification component and it supports latest version of OVM [5] – [7].

III. PROPOSED ENVIRONMENT

To verify Interconnect like above, verification IP can be used at each interface which will give following features:

1) As OCP is highly configurable and each interface might
have different configuration e.g it might be possible that burst is supported at master1 but not at master2, verification IP OCP master will generate the constraint random stimulus depending on how the interface is configured as there might be address mappings so master can be constrained to generate only legal address range

2) Functional coverage will be collected at each interface level.

3) Verification IP with XML verification plan can be used along with Questa Verification Management feature to automatically track the final coverage with respect to XML verification plan.

4) Protocol checker will be instantiated at each interface so that if any illegal behaviour is observed assertion is fired transaction level display in waveform as well in log file.

The classic interconnect verification challenge is depicted in the block diagram as shown below:

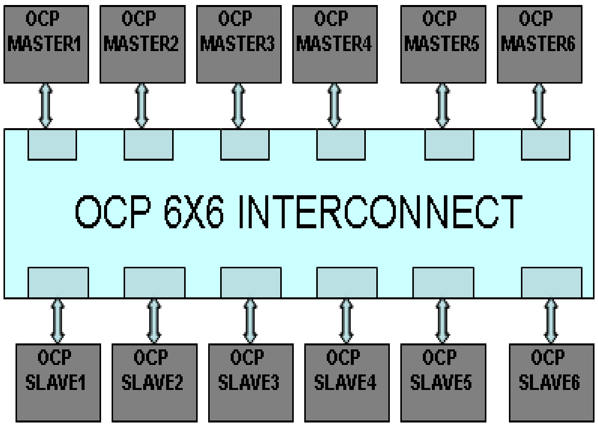


Fig.1 Shows a 6 Master 6 Slave OCP Interconnect

Now to replace single MASTER Interface, following architecture will replacing OCP Master<n> in Fig.1

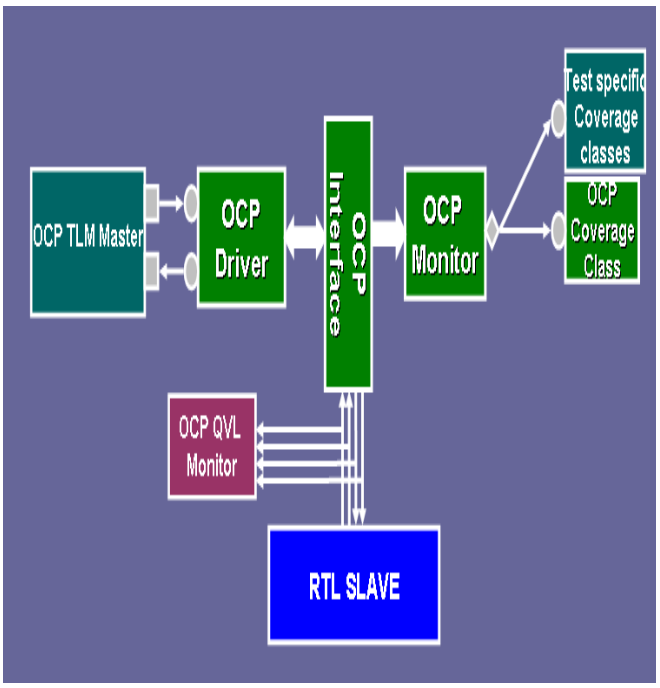


Fig. 2 OCP verification IP as Master

Now to replace single SLAVE Interface, following architecture will replacing OCP Slave<n> in Fig.1

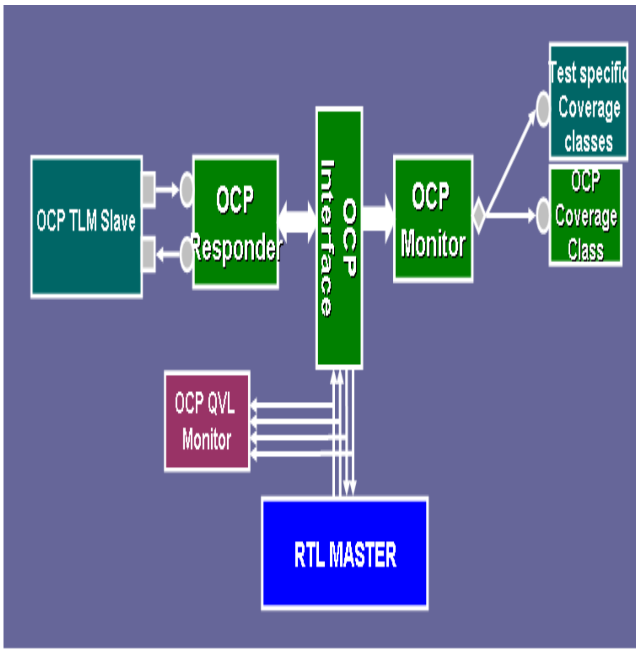


Fig. 3 OCP verification IP as Slave

Now with using OCP verification IP Master and OCP verification IP Slave, interconnect verification environment would become.

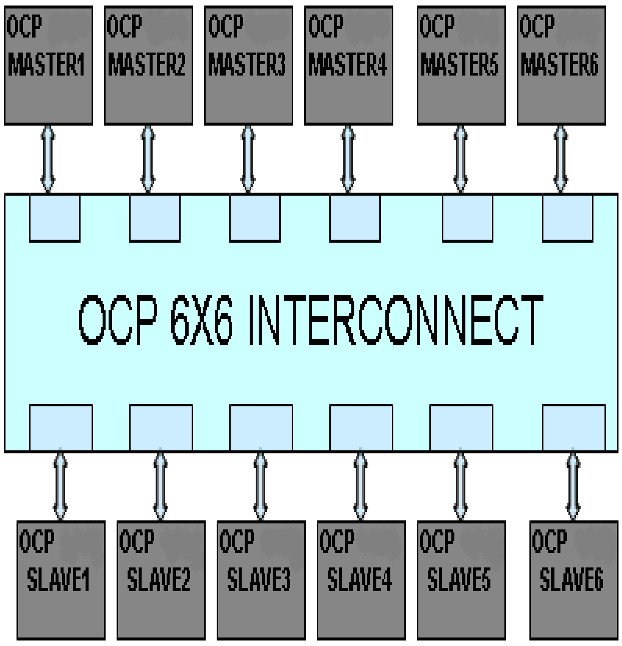
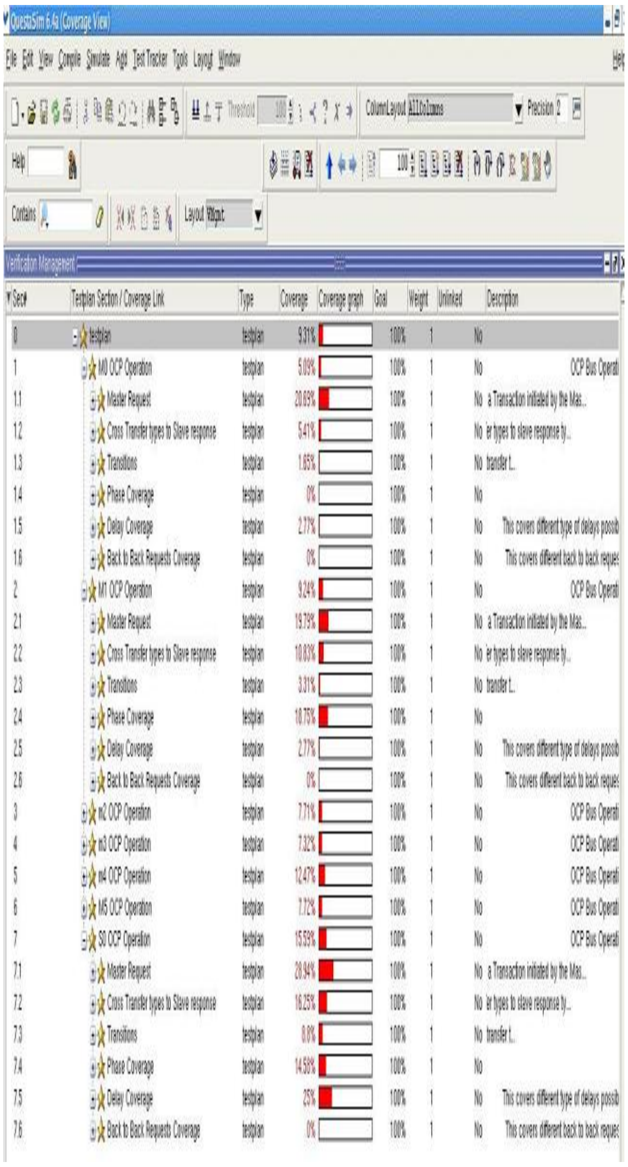


Fig. 4 OCP interconnect environment where OCP verification IP MASTER<n> corresponds to Fig 2 and OCP verification IP SLAVE <n> corresponds to Fig3

Once tests are run then functional coverage can be saved and it can be merged with XML provided verification to view how far one have gone in their verification goals As masters and slave cores gets ready they can be first verified individually using verification IP as shown in Fig 2 and Fig 3, then they can be hooked up in current verification environment of Interconnect by replacing TLM master/TLM slave with RTL master/slave core.

Rest other TLM component can be replaced later e.g if Master1 was CPU block and Master2 was DMA and Slave1 was GPIO block then after doing their block level verification with verification IPs as explained above they can replace OCP TLM master and OCP TLM Slave in figure2 and figure3, figure 2 and 3 will now become figure 6 for OCP verification IP MASTER1, OCP verification IP MASTER2 and OCP verification IP SLAVE1.


QuestaSim 6.4a (Coverage View)
File Edit View Compile Simulate Add Test Tracker Tools Layout Window
Threshold
ColumnLayout AllColumns
Precision 2
Help
Contains
Layout
Verification Management

Sec#	Testplan Section / Coverage Link	Type	Coverage	Coverage graph	Goal	Weight	Unlinked	Description
0	testplan	testplan	9.31%		100%	1	No	
1	M0 OCP Operation	testplan	5.09%		100%	1	No	OCP Bus Operati
1.1	Master Request	testplan	20.83%		100%	1	No	a Transaction initiated by the Mas...
1.2	Cross Transfer types to Slave response	testplan	5.41%		100%	1	No	er types to slave response ty...
1.3	Transitions	testplan	1.65%		100%	1	No	transfer t...
1.4	Phase Coverage	testplan	0%		100%	1	No	
1.5	Delay Coverage	testplan	2.77%		100%	1	No	This covers different type of delays possib
1.6	Back to Back Requests Coverage	testplan	0%		100%	1	No	This covers different back to back reques
2	M1 OCP Operation	testplan	9.24%		100%	1	No	OCP Bus Operati
2.1	Master Request	testplan	19.79%		100%	1	No	a Transaction initiated by the Mas...
2.2	Cross Transfer types to Slave response	testplan	10.83%		100%	1	No	er types to slave response ty...
2.3	Transitions	testplan	3.31%		100%	1	No	transfer t...
2.4	Phase Coverage	testplan	18.75%		100%	1	No	
2.5	Delay Coverage	testplan	2.77%		100%	1	No	This covers different type of delays possib
2.6	Back to Back Requests Coverage	testplan	0%		100%	1	No	This covers different back to back reques
3	m2 OCP Operation	testplan	7.71%		100%	1	No	OCP Bus Operati
4	m3 OCP Operation	testplan	7.32%		100%	1	No	OCP Bus Operati
5	m4 OCP Operation	testplan	12.47%		100%	1	No	OCP Bus Operati
6	M5 OCP Operation	testplan	7.72%		100%	1	No	OCP Bus Operati
7	S0 OCP Operation	testplan	15.59%		100%	1	No	OCP Bus Operati
7.1	Master Request	testplan	28.94%		100%	1	No	a Transaction initiated by the Mas...
7.2	Cross Transfer types to Slave response	testplan	16.25%		100%	1	No	er types to slave response ty...
7.3	Transitions	testplan	8.8%		100%	1	No	transfer t...
7.4	Phase Coverage	testplan	14.58%		100%	1	No	
7.5	Delay Coverage	testplan	25%		100%	1	No	This covers different type of delays possib
7.6	Back to Back Requests Coverage	testplan	0%		100%	1	No	This covers different back to back reques

Fig.5 OCP interconnect environment where OCP verification IP MASTER<n> corresponds to Fig 2 and OCP verification IP SLAVE <n> corresponds to Fig3

As soon as all the master and slaves are created and they are verified they will be replacing OCP TLM Master and OCP TLM Slave in verification IP, but still same environment will be reused to verify the Interconnect based system with verification IPs.

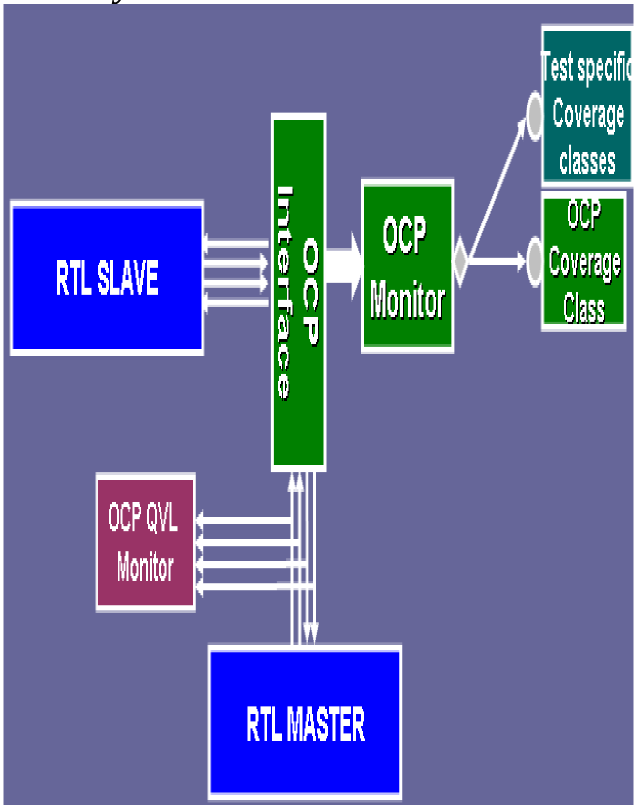


Fig.6 Verification IP hooked up to collect coverage from interface.

IV. CONCLUSIONS

To verify a block or a system, environment must be created which can be easily moved from block to system level Verification components used must have a protocol compliance checking mechanism which should be measured precisely Verification components must have a functional coverage mechanism along with detailed verification plan so that verification progress can be tracked

Verification IP comes with all the above features and also it can work at any abstraction level which makes much easy to move from one abstraction level to another As analysis components (functional coverage) is a system verilog based unencrypted component so it can be changed depending on the needs in case if a block supports only subset of complete protocol

REFERENCES

[1] Mark Glasser"Open Verification Methodology Cookbook" 1st edition Springer, 2009

[2] "OVM Golden Reference Guide", version2.0, Doulos, 2008.

[3] Open Verification Methodology, download from www.ovmworld.org

[4] Open Chip Protocol specifications ver. 2.1.

[5] Chris Spears, "System Verilog for Verification", 2nd ed., Springer, 2008.

[6] Writing test benches using system verilog, Janick Bergeron

[7] Reuse Methodology manual, Second Edition, Keating