cadence®

Tackling Verification Challenges with Interconnect Validation Tool

By Hao Wen and Jianhong Chen, Spreadtrum and Dave Huang, Cadence

An interconnect, also referred to as a bus matrix or fabric, serves as the communication hub of multiple intellectual property (IP) cores inside a system on chip (SoC). As the capacity of today's SoCs continues to increase dramatically, interconnect verification complexity also grows, considering the master/slave numbers, various protocols, different kinds of transactions, and multi-layered topology. The traditional ways of firing many direct tests, or applying a divide-and-conquer strategy, do not provide a holistic verification for SoC interconnects. A systematic approach must be adopted to tackle the challenge and make the process more efficient. In this paper, we discuss how we adopted Cadence® Verification IP for AMBA® Protocols and Cadence Interconnect Validator, an industry-leading tool for fabric verification. We convey how these tools helped us to improve verification efficiency, and we discuss a verification environment that we created with the Universal Verification Methodology (UVM).

Contents

Introduction

In modern systems on chip (SoCs), where ARM® AMBA® protocols are intensively used as standard intellectual property (IP) interfaces, the interconnect is usually required to bridge and facilitate the communication between many different IP interfaces. The interconnect presents one of the biggest challenges of SoC verification, considering the different kinds of protocol interfaces, conversion of different transaction types, and the large number of masters and slaves. The verification effort becomes particularly significant when multi-layered interconnects are involved, since they are large combinational transaction paths that are very difficult to imitate using normal scoreboards or predictors. Fortunately, interconnect verification IP and validation tools are available to bring greater efficiency and accuracy to the process.

Interconnect Complexities

As shown in Figure 1, the AXI™ interconnect usually serves as a common interconnect in an SoC. Generally, the interconnect here ensures that each of the six masters can communicate to any of the slaves based on the address range mapping. To verify this block at the SoC level, we usually need a flexible stimulus generator which mimics the real master's behavior; protocol checkers that ensure each AXI bus has no protocol violations; a slave bus functional model (BFM) that can act as a device or memory based on different needs; and a system scoreboard that ensures each point-to-point communication is correct.

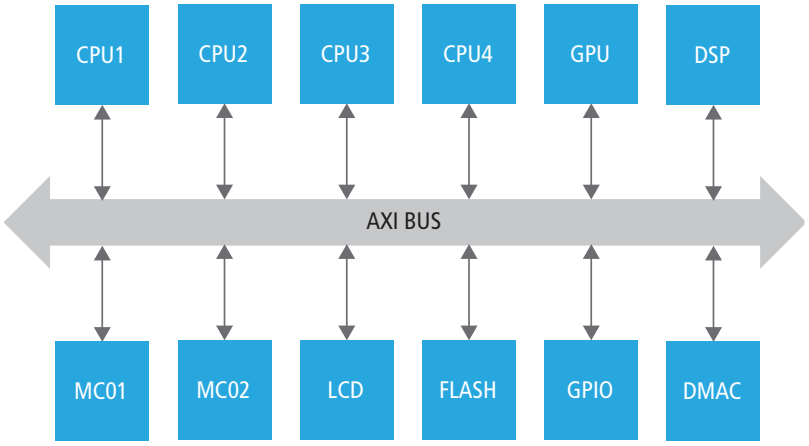

Figure 1

Reality is crueler than that, however. In an actual SoC, we may have many complex interconnects that consist of different layers and that support multiple protocols, such as APB™, AHB™, and AXI. Usually, high-speed components, like CPU and digital signal processors (DSPs), are tied into AXI fabrics, while mid-speed devices are connected to AHB fabrics. The low-speed peripherals, like UART, I2C, and MCU, might use an APB bus. All of these fabrics are chained together via bridges or a big fabric for overall communication (Figure 2). This adds more challenges for verification, since the scope of verification now increases to multiple interconnects in a system. Also, protocol conversion needs to be considered within a possible hybrid topology.

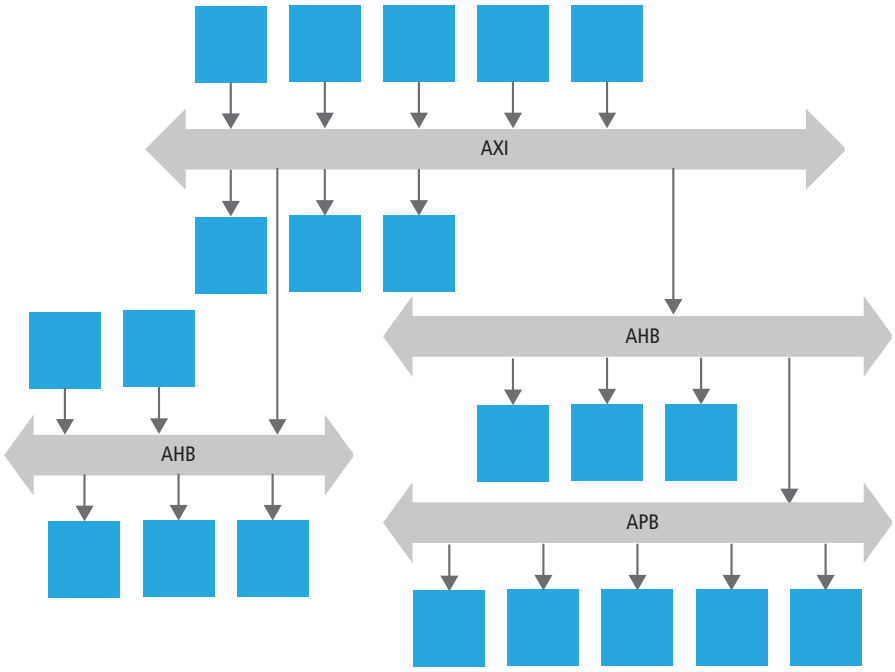

Figure 2

In the past, we built simple BFMs as masters/slaves using Verilog; this approach works well with simple protocols, like AHB and APB, around a simple fabric. However, when it comes to AXI, we felt a bit "sloppy" with this model, because it was hard to believe our created model was golden enough due to its complexity. We also had several direct test cases to verify the master-to-slave path from a point-to-point (P2P) perspective. For example, if we sent a transaction from CPU1 to DMAC, we had to have a data checker to check that the data arrived correctly and that the ID conversion was correct, and so forth. We didn't have a protocol checker to make sure there weren't violations; we just 'relied' on the interconnect. We had several functional coverage points, yet we still felt it might not

be complete. We had a test plan inherited from a previous project, and we did some incremental editing to meet our needs. Unfortunately, our new SoC was not incrementally architected. All of these factors meant that we had holes in our verification process, and we knew we would feel the pains once a new project was kicked off. At this point, we began thinking about ways to improve our verification process.

Four Fabric Verification Challenges

As we summarized in our past verification work, there are four challenges in fabric verification that we have to overcome to be able to claim that we've improved efficiency:

1. **Correctness:** an ideal verification environment should guarantee correct stimulus generation, protocol checker, and coverage collection. Our previous BFM and direct test cases were inadequate to achieve this goal.
2. **Completeness:** this covers the systematic transaction checking around layered interconnects, and it becomes more crucial when the design complexity increases. In a complex SoC, there will be multiple layered interconnects in which different protocols are involved. We need a sophisticated mechanism to check every transaction from point to point, even with different protocols, from different paths, and even in parallel execution. For example, a CPU can issue a transaction across an AXI interconnect to a slave which is tied to an AXI2AHB bridge, which is finally transferred to another AXI interconnect.
3. **Compliance:** this refers to conformity in fulfilling verification requirements in an SoC. We previously had a test plan in which direct test cases addressed this concern. With our complex interconnect system, we somehow felt it would be insufficient to fulfill the thoroughness requirement. To have verification compliance, we need fully randomized test cases and well-defined coverage points. The randomized data item should be able to exercise all the major aspects of the interconnect system, and the coverage points are used to measure the progress.
4. **Convergence:** this is an engineering-oriented and high-level goal that allows the effort spent on previous work to be easily and quickly leveraged for future projects. Simply stated, it requires the verification environment to be highly reusable and maintainable, across different projects, perhaps with different teams, with quick turn-around. This requires verification work to involve a certain methodology.

Addressing the Challenges with Verification IP

Cadence provides a rich set of proven VIP for AMBA protocols. We initially felt that these IP interfaces might be our ultimate solution, so we began our evaluation.

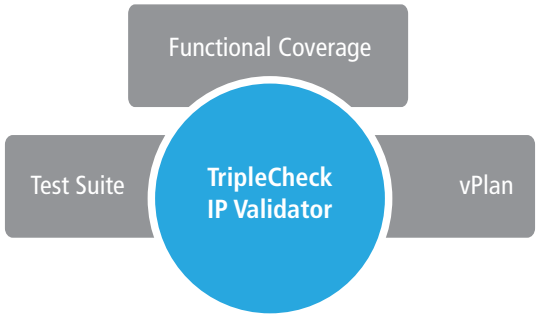

Figure 3

Cadence's AMBA VIP covers all the ARM bus protocols and provides very user-friendly interfaces for SystemVerilog users. Basically, the VIP has three main features (Figure 3) which make it more powerful than a traditional BFM and provide the plug-and-play component for bus verification. The VIP can generate random constrained stimulus, which is regarded as a golden model. It is also able to monitor or check the protocol passively any time during simulation. It has internal functional coverage containing most of the basic capabilities that users actually need. More importantly, the VIP provides a compliance system that ensures test completeness against the spec or test plan.

Interconnect Validator is a system-level VIP that serves as a system scoreboard for interconnects. It's a passive component that monitors each transaction behavior within a fabric network and makes sure each transaction behaves correctly during different phases.

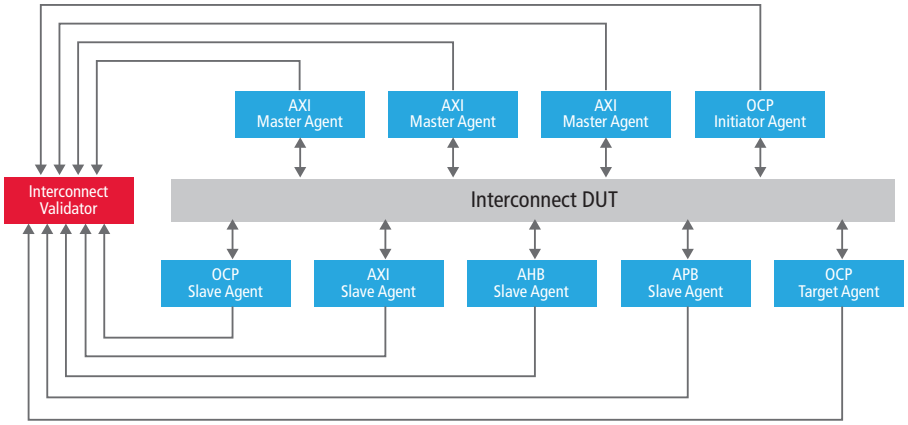

Figure 4

Theoretically, Interconnect Validator can work with any number of masters and slaves within any number of layered interconnects, if it is correctly configured regarding ID conversion, address mapping, port definition, etc. It can also handle different kinds of protocol types and transaction types. The tool also has a powerful application programming interface (API) that enables it to support proprietary protocols.

We started the evaluation with the real design under test (DUT) interconnect system in our SoC, and it is the best way to see how these new tools can meet our needs. We chose a two-layered interconnect with mixed ARM protocols. We believe that if VIP and Cadence Interconnect Validator (IVD) work well for this subsystem, the tools should be able to scale up to suit larger designs.

In this subsystem (Figure 5), we were advised to attach a VIP slave monitor to each DUT slave; then, the VIP can monitor the transaction and provide information to the Interconnect Validator. As for the master of Interconnect 01, we had to use CPU BFM to initialize the system, since we had a large initialization sequence that had to comply with strict timing. After this, the VIP master worked to replace the BFM. During normal simulation, we expected both AXI M0 and AHB M4 to generate large randomized and legal transactions simultaneously. The transactions should exercise the interconnects and also transfer to each slave without deadlock. VIP slave monitor should ensure the protocol correctness and collect coverage when possible. Interconnect Validator would survey all the transactions across different interconnects and make sure they arrive correctly from source to sink.

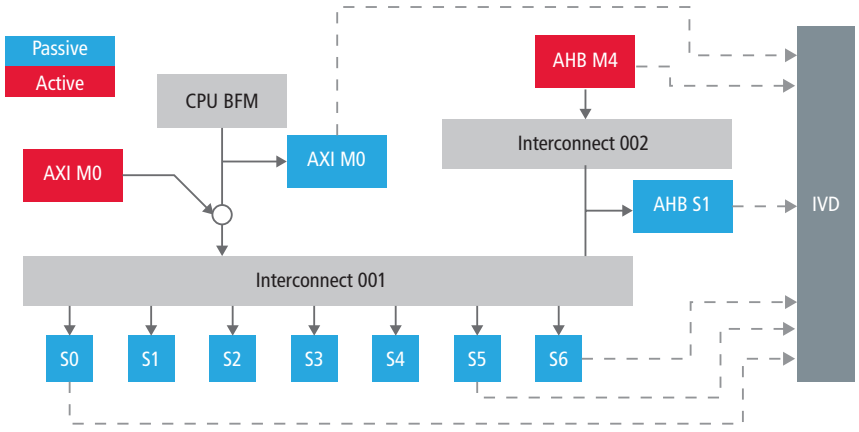

Figure 5

As stated previously, it was important for our team to be able to reuse and easily maintain our verification environment. We felt that adopting certain mature methodologies would be the only way to achieve this. Fortunately, Cadence VIP provides abundant UVM examples and the applications engineers are extremely experienced with this. Our entire verification environment was built on UVM. Using UVM, our verification environment is more structured and regular than ever. The VIP is then instantiated as an agent which contains drivers, monitors, and sequencers. All of the tests we need to execute can be written using a modular sequence library, and the data items in any transaction are already constrained and randomized to cover all attributes defined by the interface

protocol, which is very convenient when constructing large random test regressions. UVM components follow the loose coupling paradigm, and it has been our experience that they are easy to plug and play in the new environment.

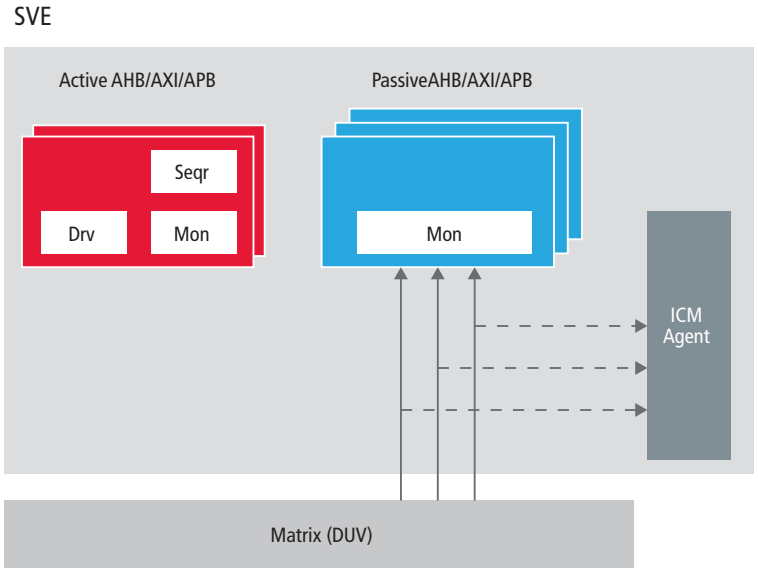

Figure 6

The basic UVM environment is shown above (Figure 6). Since we are more familiar with UVM and Cadence VIP tools, we were confident that we could create a general verification template that would suit all different designs. The main focus areas of an interconnect verification environment include: (1) interconnect system topology; (2) address/memory mapping; (3) ID conversion strategy (for AXI only). Since these three areas of meta data are already available during the system design process, we can retrieve the data and easily generate the whole interconnect verification environment in UVM. Also, while writing test sequences might require some manual work, keeping them generalized, combined with automated randomization, helps in easily creating massive test regressions. We achieved this goal with the help of the Cadence applications engineer.

Although we spent some time on integration and debugging, we consider this evaluation to be successful. We are accustomed to debugging environments and designs, and writing many direct test cases. By using Cadence tools, we have only needed to focus on whether it meets the design's intention, which has increased our efficiency significantly. Cadence VIP and Interconnect Validator demonstrated the ability to address all our interconnect verification concerns; VIP ensures the correctness and Interconnect Validator helps us to achieve the completeness and compliance. Using all these tools with UVM, we are confident that our next verification project will run much more efficiently.

Looking ahead, we still have much work to do. We will continue using these tools and integrating them into a more complex SoC environment. We know that Interconnect Validator also has a performance analysis function that we are interested in trying in the future.

Acknowledgements

Special thanks to Dave Huang and Li Chen, who educated us on building up the VIP environment around UVM. We also appreciate the support in helping us to write scripts, debug, and migrate and in transferring all of the knowledge.

Company introduction

Spreadtrum Communications, Inc. is a fabless semiconductor company that develops mobile chipset platforms for smartphones, feature phones and other consumer electronics products, supporting 2G, 3G and 4G wireless communications standards. Spreadtrum's solutions combine its highly integrated, power-efficient chipsets with

customizable software and reference designs in a complete turnkey platform, enabling customers to achieve faster design cycle with low cost. Spreadtrum's customers include global and China-based manufacturers developing mobile products for consumers in China and emerging markets around the world.

References

1. ACE VIP User Guide in VIPCAT11.3.s020

2. ICM SV User Guide in VIPCAT11.2.S020

3. Cadence VIP training slides