2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Automatically synthesizing higher level of protocol abstraction for faster debug and deeper insight into modern digital designs

Alasdair Ferro, Amar Patel, Chris Jones, Yogesh Badaya

{alasdair_ferro, amar_patel, chris_jones, yogesh_badaya}@mentor.com

Mentor Graphics, a Siemens Business

Abstract- The design and verification complexity has led to the evolution of various languages and methodologies, such as SV and UVM. This evolution happened primarily because of the raised level of abstraction at which design and verification engineers have to think and capture the intent. Such abstraction simplifies the comprehension and debug activities of the system. Still, as designs go through the flow, they are transformed into lower level of abstraction (logic synthesis and instrumentation), that could be based on context in which this data is captured, such as third-party sources, or hardware accelerator-based data capture. The analysis and debug complexity again goes up due to unavailability of higher level of data abstraction.

In this paper, we propose to synthesize the higher level of abstraction from the lower level of design data and making design data available for the debug and analysis. In our experiment, we captured AXI signal-level activity from emulation system and synthesized higher level of transactions. With this view, users can debug and analyze AXI data with the packetized transactions instead of signal-level waveforms. Though the methodology is demonstrated on AXI but is applicable to all the industry standard protocols. We have also demonstrated that this higher level of data can be easily analyzed for other insights into the subsystem like instructions coverage, address ranges, and performance analysis.

Keywords—protocol, IP, VIP, AXI, NASTI, RISC-V, SoC, SIP, emulation, prototype, signal activity, bus, transactions, processor performance evaluation

I. Introduction

There are various situations where design data is available only at lower level of abstraction. For example when waveforms are captured from emulation or FPGA Prototype or any other hardware accelerator during design verification, data is available as signal-level activity. Having only lower level of abstraction makes it difficult to understand the design behavior to debug a failure, capture coverage, and map to test plan, amongst other verification tasks.

Typically, the only option available to engineers is look at the signal-level waveforms and make progress. This requires protocol expertise to be effective, and increases the time it takes to close the task at hand.

In some other cases, engineers can use tools that have been matured to give better representation of the information, mapped to original higher level of abstraction. However, such tools typically work only in few cases and that too when the implementation of that tool started with higher level of abstraction, and has all the relevant information. For example, if the tool itself is driving UVM sequences into the emulation box, it already knows the details that can map the activity back. Still, such tools offer limited features and work for only some specific cases.
The problem keeps growing as higher number of standard IPs become part of the sub system. We have identified the following key issues with the existing flow:

1. Need of protocol expertise to work at signal level of protocols with the increase in number of protocols and the complexity of the protocols.
2. Extra effort is needed to make sense of the data. This increases the debug complexity and time to functional correctness.

3. Data analysis at various levels is a critical part of the verification closure. This includes test plan coverage, code coverage and other protocol specific analysis. Signal-level data makes it complex to work on these parameters effectively.

II. Proposed Solution

We propose methodology to synthesize higher level of protocol abstraction automatically from the signal level data and protocol configuration captured at run time. With this approach, we take care of the productivity issues with verification as described earlier.

We provide a flow as illustrated in Figure-1. Our solution reads signal-level activity and protocol configuration-specific information. Our tool can transform this information into higher level of abstraction.
This synthesis technology makes verification faster by providing the following features to users:

- Graphical visualization to comprehend and debug the data at the transaction level.
    - Transaction Viewer
    - Protocol Specific Debugger
- Additional utilities to assist with data analysis based on protocol. They help in protocol-specific checks, and scenario coverage.
- Features that make it easy to visualize design activities at the higher level of abstraction so that verification engineers do not need to have deep knowledge of all the protocols.

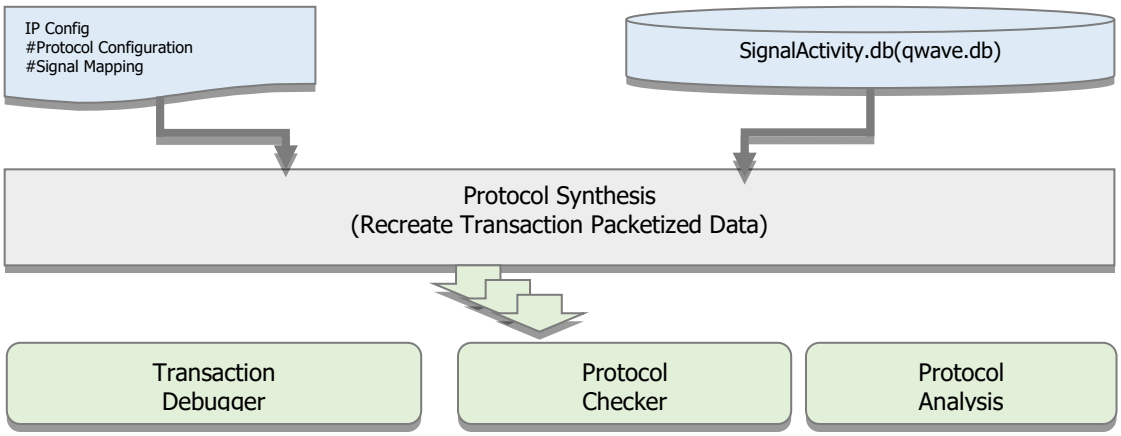

Figure 1 Protocol Synthesis

III. Experiment and Results

The experiment was done on a RISC-V processor based sub system, Figure-2 describes the block diagram of the sub system. The design is synthesizable and can be executed on hardware based verification technologies like emulation or FPGA prototype. Waveforms can be captured for any bus or a list of signals in the design.

We captured the signal-level activity at the bus as marked in Figure-2 with a star in red box. The RISC-V design and verification flow was instrumented to capture data in our propriety waveform format (qwave.db). The chosen buses are UART and AXI-based communication channel between processor (Rocket Tile) and the memory devices. The software program loaded in the RAM and the boot program are executed on the processor and data is transferred using the chosen bus.

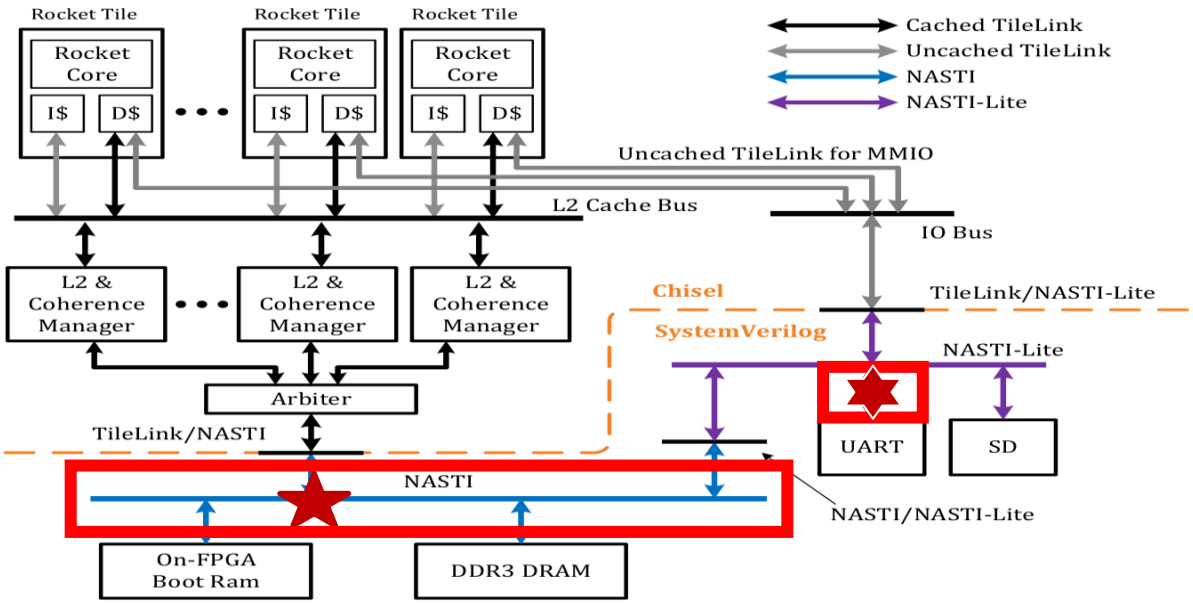

Figure 2 RISC-V based sub system

Figure 3 and Figure 4 display signal-level activity and corresponding synthesized transactions for AXI Write and AXI Read, respectively. The synthesized transactions contain packet-level information and is helpful in debugging and searching a specific transaction activity. During simulation, a transaction log is also generated in text version of the synthesis report.

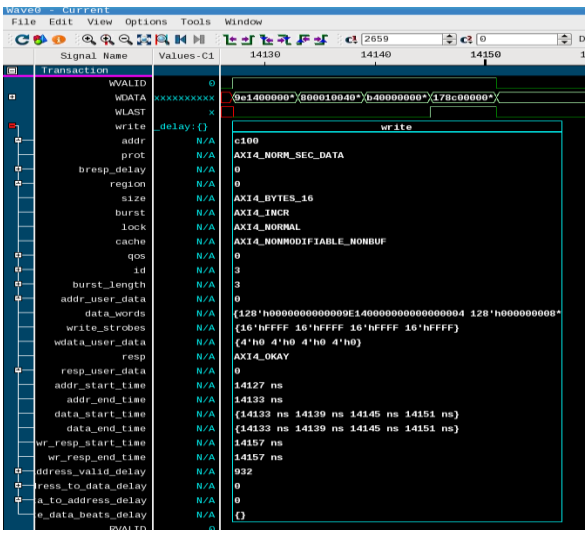

Figure 3 AXI Write Transaction

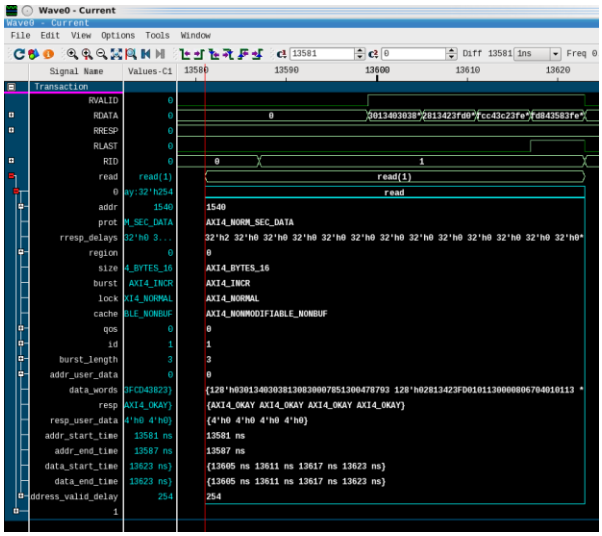

Figure 4 AXI Read Transaction

The communication bus is between the processor (Rocket Tile) and the RAM, so the transaction data contains information about the nature of instructions executed on RISC-V processor. This enables a wide range of checks and analysis, which can be done on RISC-V-based designs. For our experiment, the data was analyzed to extract instruction coverage (shown in Figure 5) and address distribution (shown in Figure 6).

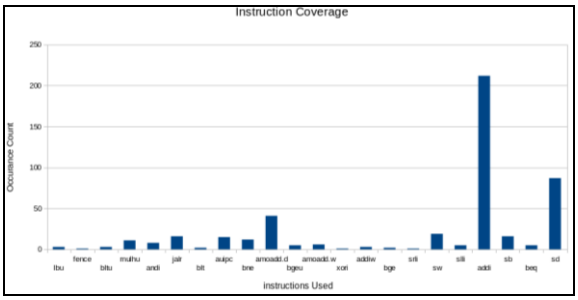

Figure 5 RISC-V Instructions Coverage

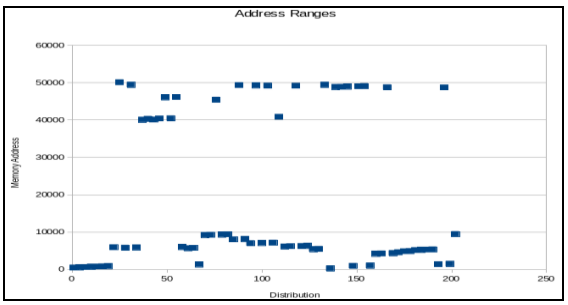

Figure 6 RISC-V Instruction addresses distribution

IV. Conclusion

Synthesis of transactions from signal-level activity provides a platform to perform design checks and design data analysis. The data collection is done during design run and synthesis is done as post process. The feature is useful for simulation, emulation and FPGA based-verification and validation tools. RISC-V, as an open ISA, is used to provide standard and independent processors. This capability can further be used for cache coherency, security, processor performance analysis and compliance checking in RISC-V based subsystems.

Though we have demonstrated our application on AXI and UART protocol, the methodology scales to all the industry standard protocols. This reduces effort and time to debug and achieve verification targets, hence verification closure of the designs.

V. References

[1] RISC-V. https://riscv.org.

[2] LowRISC https://www.lowrisc.org.

[3] ARM AMBA AXI Protocol Specifications.